# **English Query Overview**

The English Query documentation contains these principal sections.

Section	Description
English Query Tutorials	Provides step-by-step instructions for creating and building English Query models. The tutorials are available from the English Query <b>Start</b> menu or from the Microsoft® Visual Studio® <b>Help</b> menu when an English Query project is opened. It is recommended that you complete the tutorials prior to developing your own English Query application.
What's New in English Query	Describes the new features in English Query.
English Query Fundamentals	Provides a basic description of English Query.
Developing and Deploying English Query Applications	Provides information about English Query models and command relationships, database normalization requirements, updating, replacing, testing and compiling models, and deploying the application to the Web.
Sample Applications and Client Interfaces	Describes where you can locate the sample applications included with English Query.
Analysis Services in English Query	Describes integrating Microsoft SQL Server™ 2000 Analysis Services with English Query.
Object Model Reference	Describes the Authoring, Engine, and Question Builder object models.
How To	Provides step-by-step instructions for tasks you can accomplish using English Query.
English Query User Interface Help Reference	Describes the shortcut keys, wizards, editors and windows, dialog boxes, icons, and buttons used in English Query.

## What's New in English Query

This section describes new features in English Query.

### **English Query Features Introduced in This Version**

This version of English Query contains these new features:

Visual Studio Integration

**Graphical Authoring** 

**SQL Project Wizard** 

Analysis Services Integration (formerly OLAP Services) and OLAP Project Wiza

Semantic Modeling Format (SMF) and Authoring Object Model

New Regression Features

Structured results output can be used as input to other tools.

Full-Text Query Support

Two-Click Deployment

Enhanced Authoring by Example

Oracle Database Support

**Graphical Question Builder** 

## **Installation Requirements and Considerations**

Installation of English Query requires the following:

- Microsoft® Windows® 95, Windows 98, Microsoft Windows NT® version 4.0 (Service Pack 6 or later), or Windows 2000.
- 40 MB of free disk space.
- Microsoft Internet Explorer version 5.0 or later (build 2615 or later is required for the display of large models and other user interface elements if running on a Windows 98 system).

**Note** Internet Explorer version 5.5 may affect how topics appear when they are printed.

Before installing English Query, consider the following:

- English Query and Windows 2000 logo requirements.
  - Because English Query appears within Microsoft Visual Studio® version 6.0, which is not compliant, English Query does not meet Windows 2000 logo requirements.
- English Query documentation.
  - All of the documentation for English Query is available in the HTMLHelp file Eqdoc.chm and the tutorial Eqtut.chm, which can be accessed from the English Query Help menu or from the English Query program group. If English Query is the only Visual Studio product you have installed, you must install MSDN® to access Visual Studio Help.

If English Query has been installed with Microsoft SQL Server<sup>™</sup> 2000, the Help system will access SQL Server Books Online, SQL80.col, instead of English Query Books Online, Eqdoc.chm. However, both documentation files contain essentially the same material and both provide context-sensitive (F1) Help for English Query. Regardless of the installation scenario, English Query Books Online is called from the

- English Query program group.
- English Query 2000 is compiled with XML 2.0, whereas SQL Server 2000 uses XML 2.6. To prevent compilation errors, select XML 2.0 in the References section of your English Query project.
- OLAP Services version 7.0 must not be running during installation.
   Installing English Query 2000 when the SQL Server 7.0 OLAP Services service is running may break connectivity to the local OLAP Client.
   This issue does not occur when SQL Server 2000 Analysis Services is running.
- If you are running OLAP Services 7.0, OLAP Manager must not be running during installation.
  - If OLAP Manager is running, English Query Setup cannot copy the files needed for English Query OLAP support. Quit OLAP Manager before running English Query Setup.
- Uninstalling English Query breaks SQL Server 7.0 OLAP Services and vice versa.
  - If you have installed SQL Server 7.0 OLAP Services, and you uninstall English Query, you must reinstall OLAP Services. Conversely, if you have installed English Query and you uninstall SQL Server 7.0 OLAP Services, you must reinstall English Query to maintain OLAP connectivity. This issue does not occur with Analysis Services.
- Microsoft Internet Information Services (IIS) must not be running with an English Query application during installation.
  - If using English Query in a Web page, shut down IIS by running **net stop iisadmin** before installing English Query.
- Microsoft Visual Basic® and Microsoft Visual C++® sample applications use either the Microsoft PivotTable® or Microsoft FlexGrid Control version 6.0 to connect to a Analysis Services server.
  - The Visual Basic and Visual C++ sample applications use either the PivotTable or the FlexGrid control to display the results of

Multidimensional Expressions (MDX) queries. At run-time, the sample application detects which of the following dynamic-link libraries (DLLs) or Microsoft ActiveX® controls are installed on your computer. The sample applications can display the results of MDX queries in a PivotTable using an MDX wrapper.

#### PivotTable

To display the query results in a PivotTable, you must have the following files registered on your computer.

File	Description
Msowc.dll	This DLL contains the PivotTable control and is installed with Microsoft Office 2000.
Owcmdx.dll	This DLL contains the MDX wrapper and is installed with English Query.

#### FlexGrid Control

If the PivotTable control is not installed on your computer, the sample applications use the FlexGrid Control 6.0 and ActiveX Data Objects (multidimensional) (ADO MD) version 2.5 to display the query results.

To display the query results using the FlexGrid control, you must have the following files registered on your computer.

File	Description
Msflxgrd.ocx	This file contains the FlexGrid control.
Msadomd.dll	This DLL is required to use ADO MD 2.5.

• The authoring tool shows the splash screen for Microsoft Visual InterDev® and Microsoft Visual J++®.

The English Query authoring tool is integrated within the Visual Studio

common shell, which also includes Visual InterDev and Visual J++. The splash screens for these products may also appear at startup.

### See Also

**English Query Usage Issues** 

## **English Query Fundamentals**

Using English Query, you can turn your relational databases into English Query applications, which allow end users to pose questions in English instead of forming a query with an SQL statement.

The English Query Model Editor appears within the Microsoft® Visual Studio® version 6.0 development environment. From there, you can choose one of the English Query project wizards, the SQL Project Wizard or the OLAP Project Wizard, to automatically create an English Query project and model. After the basic model is created, you can refine, test, and compile it into an English Query application (\*.eqd), and then deploy it (for example, to the Web).

### **Creating an English Query Project and Model**

Using the SQL Project Wizard or the OLAP Project Wizard, you incorporate the database structure (table names, field names, keys, and joins) or cube information of the database into a project and a model.

A model contains all the information needed for an English Query application, including the database structure, or schema, of the underlying SQL database or cube and the semantic objects (entities and relationships). You also define properties for an application and add entries to the English Query dictionary, as well as manually add and modify entities and relationships while testing questions and set other options to expand the model.

### **Creating Entities and Relationships**

With the wizards, semantic objects are automatically created for the model. These include entities and relationships (with phrasings such as *customers buy products* or *Customer\_Names are the names of customers*). Entities are usually represented by tables, fields, and OLAP objects.

An entity is a real-world object, referred to by a noun (person, place, thing, or idea), for example: **customers**, **cities**, **products**, **shipments**, and so forth. In databases, entities are usually represented by tables, fields, and Analysis Services objects.

Relationships describe what the entities have to do with one another, for example: *customers purchase products*. Command relationships are not represented in the database but refer to actions to be executed. For example, a command to a compact disc player can allow requests such as "Play the album with song X on it."

### **Deploying an English Query Application**

You can deploy an English Query application in several ways, including within a Microsoft Visual Basic® or Microsoft Visual C++® application and on a Web page running on Microsoft Internet Information Services (IIS). In the Web scenario, the interface of the application is with a set of Active Server Pages (ASP).

#### See Also

**Developing and Deploying English Query Applications** 

Creating an English Query Model

**Deploying an English Query Application** 

**Expanding an English Query Model** 

Testing an English Query Model

Sample Applications and Client Interfaces

### **SQL Database Normalization Rules**

SQL-based English Query applications work best with normalized databases. In general, it is easiest to create English Query applications against normalized SQL databases. In addition, the resulting applications are more flexible and powerful than those developed against databases that are not normalized.

This topic describes normalization rules as they pertain to English Query SQL applications. It describes problematic database structures that break these rules and how to solve these problems by creating views in Microsoft® SQL Server<sup>TM</sup>, which can be used in English Query just like any other table.

# Rule 1: There should be a one-to-one relationship between the instances of an entity and the rows of the table.

For every table that represents an entity, each and every row in that table should represent one and only one instance of that entity. Conversely, each and every instance of that entity should be represented by one and only one row in the table.

In this situation, this rule is not met:

Table: **Employees** 

Fields: Emp\_id, Emp\_name, Status, Position, Salary

Keys: Emp\_id, Status

This table stores information about employees. It contains their names, positions, and salaries. But sometimes employees move around from position to position, and when they do, their salaries change. So for some employees, this table also stores information about their projected position and salary. If the value of the Status field is C, the row contains the current information for the employee. If the value is P, it contains the projected information. Thus, an individual employee may appear twice in this table.

Because an employee can appear twice, you cannot use this table to represent the **employees** entity. If you were to associate the **employees** entity with this table, even simple requests, such as "Count the employees", would give the wrong answer. The solution to this problem is to create a view in the database that

contains a single row for each employee and to tell English Query about this view.

Here is what the view would look like:

CREATE VIEW Emps AS
SELECT Emp\_id, Emp\_name, Position, Salary
FROM employees
WHERE status = 'C'

You now have a view that contains exactly one row per employee. The **employees** entity can now be represented by this view.

# Rule 2: A field should have the same meaning in each row of the table.

Refer to the **Employees** table again:

Table: **Employees** 

Fields: Emp\_id, Emp\_name, Status, Position, Salary

Keys: Emp\_id, Status

This table violates the second rule as well. Remember, the Position and Salary fields contain current information if the value of the Status field is C, and projected information if the value of Status is P. The Position field really represents two distinct entities: **current position** and **projected position**, depending on the value of Status. This is true for the Salary field as well.

Notice that you have already extracted the current salary and current position when you created the **Emps** view. Do the same for the projected salary and projected position:

CREATE VIEW Projected\_info AS
SELECT Emp\_id, Position Projected\_position, Salary, Projected\_salar
FROM employees
WHERE status = 'P'

Now, you can tell English Query that the **projected position** entity is represented by the Projected\_position field in the **Projected\_info** view, and

similarly, **Projected\_salary**.

### Rule 3: Each table should represent at most one entity.

Often, if an attribute means one thing for one subset of the rows in the table and something else for another subset, those subsets of rows actually represent two different (but related) entities. For example, consider a table that represents various media productions, either movies or TV programs:

Table: **Productions** 

Fields: Prod\_id, Title, Type, Show\_date, Mpaa\_rating, Network

Keys: Prod\_id

If the value of the Type field is M, the production is a motion picture. If the value is T, it is a television show. For motion pictures, the Show\_date field contains the date it was released; for television shows, it contains the date on which the show was broadcast. The Mpaa\_rating field contains the rating of a movie (for example, G, PG, and PG-13), but is meaningless for television shows. Likewise, the Network field is the network on which a television show appears, but this field is meaningless for motion pictures.

This structure makes authoring an English Query application difficult in several ways. First, **television shows** and **motion pictures** are distinct entities. You can tell this by the fact that they participate in different relationships. Second, the Show\_date field has a different interpretation for a movie versus a TV show. Finally, the Mpaa\_rating and Network fields are meaningless for certain rows in the **Productions** table.

To define TV shows and movies as distinct entities, you must create views in SQL Server:

CREATE VIEW Tv\_shows AS
SELECT Prod\_id, Title, Show\_date, Broadcast\_date, Network
FROM Productions
WHERE type = 'T'

CREATE VIEW Movies AS SELECT Prod\_id, Title, Show\_date, Release\_date, Mpaa\_rating FROM Productions

### WHERE type = 'M'

You can then create the **television show** entity, which is represented by the **Tv\_shows** view, and the **movie** entity, which is represented by the **Movies** view.

**Note** A superficially similar table would not have the same problem. For example, consider a table of employees that contains a Name, a Salary, and a Type field. Type indicates what type of job that person has (for example, engineer, salesperson, receptionist). Because engineers, salespeople, and receptionists have the same kind of information (names and salaries), there is no need to create a view for each of these job types. Instead, use the Type field to define subsets of people. Thus, a user could ask, "How many engineers are there?" and get the right answer.

However, if you have the **Productions** table, use the Type field to define movies and TV shows as subsets of productions. Instead of creating a view for each, users can then ask nonsensical questions, such as "What is television program X rated?" and would get incorrect answers to seemingly valid questions like "When was movie X broadcast?"

# Rule 4: Multiple instances of an entity should be represented by multiple rows in a table.

When there is a many-to-one relationship between two (or more) entities, it should be represented in the database as a many-to-one join between two (or more) tables. There are two ways in which this rule is typically violated: by splitting data among multiple columns, and by splitting data among multiple tables. The solutions to both problems involve creating union views.

# Rule 4a: Multiple instances of an entity should not be represented as multiple columns.

Consider the following database of people and their pets. The designer of this database did not anticipate anyone owning more than three pets, and hard-coded three pet IDs in the **People** table:

Table: **People** 

Fields: Person\_id, Person\_name, Pet\_id1, Pet\_id2, Pet\_id3

Keys: Person\_id

Table: **Pets** 

Fields: Pet\_id, Pet\_name

Keys: Pet\_id

You can create the **person** entity, which is represented by the **People** table, and you can create the **pets** entity, represented by the **Pets** table. Unfortunately, when you try to create a relationship between people and their pets, you must specify a join path between the **People** table and the **Pets** table. You could arbitrarily choose one of the ID fields on which to join, but this would ultimately create three relationships that would force users to ask questions in awkward ways, for example, "Show the people and their first pets and second pets and third pets".

Because there is nothing that distinguishes these three pet IDs from one another, a better database design would have a many-to-many join table containing all pet IDs in a single column. To achieve this, create the following union view:

CREATE VIEW Pet\_owners AS
SELECT Person\_id, Pet\_id1 AS Pet\_id
FROM People
WHERE Pet\_id1 IS NOT NULL
UNION
SELECT Person\_id, Pet\_id2 AS Pet\_id
FROM People
WHERE Pet\_id2 IS NOT NULL
UNION
SELECT Person\_id, Pet\_id3 AS Pet\_id
FROM People
WHERE Pet\_id3 IS NOT NULL

Because this view contains a direct relationship between pets and their owners, it can now be used as a join table.

# Rule 4b: Multiple instances of an entity should not be represented as multiple tables.

In this financial database, the high-volume budget table is divided into yearly archive tables to avoid having a single table become too large to manage:

Table: **Branches** 

Fields: Branch\_id, Branch\_name

Keys: Branch\_id

Table: **Budgets\_1995** 

Fields: Branch\_id, Budget, Actual

Keys: Branch\_id

Table: **Budgets\_1996** 

Fields: Branch\_id, Budget, Actual

Keys: Branch\_id

Table: **Budgets\_1997** 

Fields: Branch\_id, Budget, Actual

Keys: Branch\_id

To create the **budget** and **actual\_expense** entities, use a union view to create a single <u>database object</u> to represent each one:

CREATE VIEW Budgets AS

SELECT Branch\_id, 1995 AS year, Budget, Actual

FROM Budgets\_1995

**UNION** 

SELECT Branch\_id, 1996 AS year, Budget, Actual

FROM Budgets\_1996

**UNION** 

SELECT Branch\_id, 1997 AS year, Budget, Actual

FROM Budgets\_1997

The Budget and Actual fields in the **Budgets** view can now represent the **budget** and **actual\_expense** entities.

# Rule 5: Joins should be based only on primary and foreign-key equality.

Sometimes, a relationship between two entities is represented in the database with a nonstandard join. Because English Query recognizes only joins based on the equality of primary and foreign keys, any nonstandard join must be translated, by using a view, into a standard join.

For example, this database contains information about people's heights, as well as a description of the ideal weight for people who fall into a certain height range:

Table: **People** 

Fields: Person\_id, Height, Weight

Keys: Person\_id

Table: **Ideal\_weights** 

Fields: Min\_height, Max\_height, Ideal\_weight

Keys: Min\_height

Because ideal weight depends on height, it is easy to learn the ideal weight for any given person. However, there is no way for English Query to make the connection between the **People** table and the **Ideal\_weights** table because there is no primary- and foreign-key join between the two tables. You must make this join explicit by creating a view that contains the ideal weight of each person.

CREATE VIEW Ideal\_weights AS SELECT Person\_id, Ideal\_weight FROM People, Ideal\_weights WHERE Height >= Min\_height AND Height <= Max\_height

This view can then be used as the join table for the relationship between the **person** entity and the **ideal weight** entity (which is represented by the **Ideal\_weight** field).

### Rule 6: Make sure keys are linked correctly.

An English Query compile error will occur if a field that represents an entity is joined to the key of another table. The entity should be represented by the other table rather than this field.

For example, if a **Branch** entity is created from the **Branch ID** field in the **Employee** table, an error occurs. The error occurs because a join exists from the Branch ID field of the **Employee** table to the ID field of the **Branch** table. The entity should be made a table entity and should point to the **Branch Table** table.

Likewise, foreign keys must point in the right direction. The subentity hierarchy cannot be circular. It must go in one direction from the subentity to the parent entity. The following examples show an incorrect hierarchy:

- **Enlisted** is a subentity of **Military Personnel**.
- **Officer** is a subentity of **Enlisted**.
- **Military Personnel** is a subentity of **Officer**.

To correct this problem, change the join for **Enlisted** to point to **Military Personnel**, and delete the join from **Enlisted** to **Officer**. Then, make **Military Personnel** a subentity of both the **Enlisted** and **Officer** entities.

#### See Also

Advanced Tab (Table/New Table Dialog Box)

## **Updating or Replacing Models**

Existing models can be replaced with a new one by simply copying the new model file over the old one. This has no effect on existing user sessions; they continue to use the old version.

For example, as system administrator you are given a new model file to replace or update an existing model. A server is running English Query with multiple models loaded and multiple user sessions attached to each domain. Each new session attaching to the model receives the new version of the model (with the first new session causing the new version model to be loaded). When the last user session attached to the old version of the model is gone, the old version of the model is automatically unloaded.

# Importing and Exporting Models to Meta Data Services

Microsoft® SQL Server™ 2000 Meta Data Services is an object-oriented repository technology that can be used to store English Query models. Meta Data Services is an installed component of SQL Server 2000. If you are using SQL Server, a repository database already exists for your use. Within Microsoft Visual Studio®, Meta Data Services supports the exchange of model data with other development tools as well.

You can use Meta Data Services for your own purposes: as a component of an integrated information system, as a native store for custom applications that process meta data, or as a storage and management service for sharing reusable models. You can also extend Meta Data Services to provide support for new tools for resale or customize it to satisfy internal tool requirements.

When importing an English Query model, you specify a connection to an instance of Meta Data Services. After the connection is established, you choose from a list of models. The model is then imported into the English Query project.

When exporting an English Query model, you make a connection to an instance of Meta Data Services from which you specify a model name. If the model name already exists in that instance, you are prompted confirm that you want to overwrite the existing model or choose a new name.

# Developing and Deploying English Query Applications

Developing an English Query application involves creating, refining, testing, compiling and deploying a <u>model</u>, based on a normalized SQL database or an OLAP cube.

Here are the basic steps for developing and deploying an English Query application:

1. Determine the questions that end users are most likely to ask.

Determining what questions must be answered prior to creating a model helps you to create the entities and relationships and to test your application.

For example, suppose you want to create a model for the **Northwind** database. It contains sales data from a fictitious company called Northwind Traders, which imports and exports specialty foods from around the world. As an owner of the company, you may want to know which suppliers are being used, how much of each product sold during the previous year, the total sales for the previous year, or how many orders a specific customer placed. As a manager, you may want to know how many sales your employees have made, who was the top salesperson in an area, or the total regional revenues for the past year.

2. Create a basic model using the SQL Project Wizard or OLAP Project Wizard.

The Project wizards automatically create a basic model by bringing in the schema of the data source (database or cube) and automatically creating entities and relationships based on the tables, fields (columns), joins, or OLAP objects.

3. Refine the model to address any questions that cannot be answered using the basic model.

After creating the basic model, you can further define it to answer user questions by adding entities and relationships. Drag the entities from

the **Entities** list in the left pane onto the Canvas pane, and then create a relationship between them. You can also drag relationships onto the Canvas pane.

4. Test the model and refine the model until it successfully returns the answers to the questions you pose.

If questions are not answered successfully with the model, click **Suggestion Wizard** on the Model Test window to view suggested relationships and to further refine the model. Continue to test the questions until you are satisfied with the entities and relationships.

**Note** A limited knowledge of the English language in English Query, incomplete authoring of an application, and the design of a database affects how many of the users' questions can be answered.

5. Build the application and then deploy it.

An English Query application can be deployed in several ways, including within a Microsoft® Visual Basic® or Microsoft Visual C++® application and on a Web page running on the Microsoft Internet Information Services (IIS). In the Web scenario, the interface of the application is with a set of Microsoft Active Server Pages (ASP).

#### See Also

**Compiling an English Query Application** 

**Creating an English Query Model** 

**Deploying an English Query Application** 

**Expanding an English Query Model** 

Testing an English Query Model

Sample Applications and Client Interfaces

## **Creating an English Query Model**

An English Query model is a collection of information about the <u>database objects</u>, the <u>semantic objects</u>, dictionary entries, and application properties. It is created within a Microsoft® Visual Studio® development environment.

In English Query, the SQL or OLAP Project wizards automatically generate the major portion of most models, including the database structure and the semantic objects that answer many basic questions.

You can expand the model by adding and modifying entities and relationships, adding tables and specifying joins, and setting other options.

To create a project using a Project wizard

### **Expanding an English Query Model**

After creating a basic English Query Model with the project wizards, you can refine and expand it by:

- Adding database objects, entities, relationships, and even other databases.
- Creating a mixed SQL/OLAP model.
- Utilizing the full-text search feature of Microsoft Search service.

To refine your model, choose among the following methods:

- Graphical authoring in the Model Editor, which is a diagramming tool that displays the entities and relationships in the English Query model. You can also drag entities onto one another to automatically create relationships between them.
- Manual authoring, which is done by starting with an empty project and adding entities and relationships individually. It is supported by commands on the English Query Model menu.
- Authoring by example, which provides the ability to start with questions
  that a user may ask. The Suggestion Wizard returns suggested entities
  and phrasings that need to be defined to answer the questions. The
  Suggestion Wizard is particularly useful in the testing phase, when you
  want to see what entities and relationships are missing from the model.
- Programmatic authoring, which is supported by Semantic Modeling Format (SMF) and the Authoring object model. SMF is an XML-based language that is used for persisting English Query model information (entities and relationships and their ties to the database).

**Note** English Query is English-only, and the user interface for the authoring tool does not support double-byte character keyboard input. However, Unicode data in the database is supported.

#### See Also

Adding Question Builder to an English Query Application

**Authoring with Semantic Modeling Format** 

Creating a Mixed English Query Model

Creating an English Query Model

Defining Entities in an English Query Model

**Enabling Analysis Services for an English Query Model** 

**Enabling SQL for an English Query Model** 

**Enabling Full-Text Search for an English Query Model** 

Testing an English Query Model

## **Creating a Mixed English Query Model**

While creating an English Query model, you may find that not all of the questions are answered using the current database. If you have created an OLAP model using the OLAP Project Wizard, you can enable SQL to access the data in the underlying SQL database.

If you create a project without using the OLAP Project Wizard, by default, both SQL and OLAP are enabled. When both Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services and SQL are enabled, English Query first tries to answer the question using multidimensional expressions (MDX) and then, if unsuccessful, generates SQL statements for the question.

Using mixed models, you can:

- Ask questions that cannot be answered using only OLAP cubes because the data is not in an OLAP cube but in the underlying SQL database.
- Speed up the retrieval of data for questions that are not about aggregation of data, such as "What products do people buy?".

#### **Considerations**

Before expanding your model by creating a mixed model, consider the following:

 Make sure that your OLAP cube is current with the underlying SQL database. If your OLAP cube and SQL database are not synchronized, you may get erroneous responses to your queries.

Each time that you ask a question, English Query creates a new query using the appropriate database. Follow-up questions provide inconsistent results if the underlying data has changed and the cube has not.

**IMPORTANT** You cannot access the underlying database structure of an OLAP cube in an English Query model unless you are a member of the OLAP Administrators group and the Microsoft Windows NT® 4.0 or

Windows® 2000 user group on the server with the Analysis Services database for which you are building the model. You must have OLAP Administrator permissions to load the tables underlying the cubes using Decision Support Objects (DSO).

To create a mixed model

# **Enabling Analysis Services for an English Query Model**

Enable Microsoft® SQL Server™ 2000 Analysis Services for an English Query model by:

• Creating a new project with the OLAP Project Wizard using an Analysis Services database.

Use the **New Project** command on the **File** menu to select the OLAP Project Wizard. If OLAP Project Wizard is selected instead of the SQL Project Wizard, the project is enabled for Analysis Services. The project is created automatically after you select the Analysis Services server, the database, and the OLAP cubes to use. Then, specify the entities and relationships you want included from the OLAP cubes.

• Importing cubes to an English Query model.

Use the select **Import Cubes** command on the **Model** menu to add an OLAP cube to your project.

• Adding an Analysis Services data source to the English Query model.

Use the **Enable OLAP** option on the **Data Connection** tab in the **Project Properties** dialog box to add OLAP cubes and other objects to the English Query model.

**To enable Analysis Services** 

### **Enabling SQL for an English Query Model**

Enable SQL for an English Query model by:

• Creating a new project with the SQL Project Wizard.

Use the **New Project** command on the **File** menu to access the SQL Project Wizard. When using the SQL Project Wizard, the project is enabled for SQL only. Initially, when selecting the OLE DB provider, you must select the data source or server, and then enter the logon information. After logging on, select a database, and then specify the entities and relationships from the tables that you want in the project. The project is automatically created.

 Importing tables from a SQL database to a previously created Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services (OLAP)—only project.

Use the **Import Tables** command on the **Model** menu to add tables from a SQL database to the project.

 Adding a SQL database to the English Query model by using the Enable SQL option on the Data Connection tab in the Project Properties dialog box.

**IMPORTANT** You cannot access the underlying database structure of an OLAP cube in an English Query model unless you are a member of the OLAP Administrators group and the Microsoft Windows NT® 4.0 or Windows® 2000 user group on the server with the Analysis Services database for which you are building the model. You must have OLAP Administrator permissions to load the tables underlying the cubes using Decision Support Objects (DSO).

To enable SQL by adding a database

# **Enabling Full-Text Search for an English Query Model**

Enable Microsoft Search service and the full-text search feature to support faster, more powerful, and more flexible searches of structured and unstructured textual data in SQL databases.

Using full-text search, users' questions can:

- Retrieve more information because the full-text searches can be for: inflections (alternate tenses and singular and plural forms); automatic combinations of successive words into search phrases; prefix stemming to search for various derived phrases; and proximity searching for words that occur near one another.
- Work much faster because they take advantage of the full-text indexes that are maintained in the database.
- Return documents that match or partially match the queries in order or relevance.

For example, using full-text search, the question "What movies released last year were about a big asteroid threatening to crash into Earth" generates the full-text FREETEXTTABLE() function. This function searches for many words and phrases (*big, asteroid, threatening, crash, Earth,*), all of the tenses of the verbs, singulars and plurals of the nouns, and phrases such as *big asteroid*.

To enable full-text searching

## **Defining Entities in an English Query Model**

After entities are automatically created using one of the Project wizards, they can be further defined in the Model Editor. A discussion on design considerations regarding entities is followed by descriptions of refinements that can be made, including entity name synonyms, entity defaults, and specifying entities as roles.

#### See Also

Creating a Mixed English Query Model

Creating an English Query Model

**Enabling Analysis Services for an English Query Model** 

**Enabling Full-Text Search for an English Query Model** 

**Expanding an English Query Model** 

Testing an English Query Model

### **Associating Entities**

When creating new entities for a model, you must consider whether to associate entities with <u>database objects</u> and how you want to do so. For example, how do you decide whether to represent the database object by a table or by one or more fields? You may also find that there are entities that are not associated with any database object. The **Northwind** database and sample project, Northwind.eqp, present a good example of the considerations to make when designing a model.

### **Associating an Entity With a Database Object**

Most entities are associated with a database object; however, there are considerations as to which database object to use for an association.

Consider the entity, **product**. Two database objects in the **Northwind** database might represent products: **Product Name** field in the **Products** table and the **Products** table as a whole. If you specify that products are represented in the database by the field only, almost all of the questions work correctly. For example, if you ask, "Show the suppliers and their products," you get the right list. You can even ask, "Who sells Chef Anton's Cajun Seasoning?" and get the right answer.

If you ask for a count of products, you may get the wrong answer, however. If you ask, "How many products were sold by New Orleans Cajun Delights?", English Query will build SQL statements that count the number of products sold by that supplier. If there is more than one product with the same name, the answer does not include the duplicates.

Therefore, it is not accurate to say that the **Product Name** field in the **Products** table represents products. In fact, the entire **Products** table represents products because each unique row in the table represents a single product. The field **Product Name**, on the other hand, represents the names of the products, not the products themselves.

### **Entities Not Associated with Database Objects**

Sometimes, there is an entity in a model that is not represented explicitly in the

database at all. Consider the question, "Do buyers like Chef Anton's Cajun Seasoning?". To make questions like this work, create a relationship containing the phrasing, *buyers like products* (for example, with some SQL condition like product\_name.1997\_sales>1000). To create this phrasing, first create the entity *buyer*. Because there is no list of buyers in the database, create an entity not represented by a database object.

#### See Also

Creating an English Query Model

**Defining Entities in an English Query Model** 

**Enabling SQL for an English Query Model** 

**Enabling Full-Text Search for an English Query Model** 

**Entity Name Synonyms** 

**Entity Defaults** 

**Entities as Roles** 

**Expanding an English Query Model** 

**SQL Database Normalization Rules** 

Testing an English Query Model

### **Entity Name Synonyms**

Use a name synonym to specify "Bill" as a synonym for "William" (an instance of the entity **author\_name**). That allows users to ask a question such as "How many books did Bill write?" and have that interpreted as "How many books does the author named William write?"

A name synonym only takes effect if the question is about names of entities. For example, it would not cause the question, "Who did we send a bill to last week?" to be interpreted as, "Who did we send a william to last week?"

Likewise, a product name may be stored in the database as "chair, wicker," but users may typically refer to it as a "wicker chair." Similarly, "Big Apple" may appear in the database as "New York."

Specifying a name synonym is useful, too, if an unknown word is stored in the database but its value has not been added to the model. This can occur when:

- None of the entities' values are being added because the Add values of entity to model check box in the Entity/New Entity dialog box was not selected.
- The number of values being added is restricted to limit the size of the model, and the specific value was not among those added. Increase the number of values being added by entering a number in the Words per entity box on the Data Connection tab of the Project Properties dialog box.

Caution Defining a global synonym using the **Read Synonym** option in the **New Dictionary Entry** dialog box overrides any synonyms you set using the **Name Synonyms** tab. For example, if you set the **Read Synonym** option to read "ship" as "boat," the question, "How many boats sailed last week?" is interpreted as "How many ships sailed last week?" But the question, "How many widgets did we ship last week?" is incorrectly interpreted as "How many widgets did we boat last week?" To avoid this error, specify a regular entity synonym with

**Entity/New Entity** dialog box that sets "ship" as a synonym of boat. This specifies that "ship" is a synonym of "boat" only when the question is about boats.

### See Also

**Associating Entities** 

Creating an English Query Model

**Defining Entities in an English Query Model** 

**Entity Defaults** 

**Entities as Roles** 

**Expanding an English Query Model** 

Testing an English Query Model

### **Entity Defaults**

Whenever an entity is used in a query, all appropriate defaults are attached to it. For example, if the entity **customer** has a default indicating that the customer must be a current customer, then the question, "Show the customers" is interpreted as "Show the current customers."

An entity default consists of a relationship in which the entity participates and, optionally, one or more conditions on the other entities in the relationship. For example, a relationship *customers have statuses* with a condition that the status must equal "current" is a default entity.

**Note** The default conditions are English conditions, not SQL conditions. For example, for the relationship *customers have statuses*, the condition is "current," not "c" (the code in the database that corresponds to "current").

#### To override the defaults:

- Users can specify "all" of an entity in their questions. For example, "Show all the customers," is interpreted by English Query as "Show the customers," without the default.
- Users ask questions that explicitly contradict a condition in the default.
   For example, "Show the inactive customers" contradicts the default status of "current."
- Specify a relationship that implicitly contradicts a default condition (for example, to specify the condition "stores that have gone out of business" contradicts the default status of "current"). When a user asks "Which stores have gone out of business," the default is not added.

You can also create entity defaults that are negated relationships. For example, an entity might have a negated relationship as a default so that a question such as "Show the customers" is interpreted as "Show the inactive customers." To create a negated relationship, *customers are not inactive*, chose *customers are inactive* and select **Invert** in the **Add/Edit Entity Default** dialog box. The default is the

inverse of the relationship.

**Note** These are English conditions and not SQL conditions.

### **See Also**

**Associating Entities** 

Creating an English Query Model

**Defining Entities in an English Query Model** 

**Entity Name Synonyms** 

**Entities as Roles** 

**Expanding an English Query Model** 

Testing an English Query Model

### **Entities as Roles**

Entities serve as roles in relationships. For example, in the relationship *employees work for employees:* 

- **employee** is an entity.
- works\_for is a relationship that has two roles (employee\_1 and employee\_2) and one phrasing (employee\_1s work for employee\_2s).
- Employee\_1 and employee\_2 are both roles that are represented by the entity **employee**.

You specify roles in the **Role> role** dialog box, which is available from the **Relationship** dialog box. In this dialog box, you add an entity if needed, select it, and then edit it to further specify its role in the relationship.

#### See Also

**Advanced Entity Properties Dialog Box** 

**Associating Entities** 

Creating an English Query Model

**Defining Entities in an English Query Model** 

**Entity Defaults** 

**Entity Name Synonyms** 

**Expanding an English Query Model** 

Name Synonyms Tab (Advanced Entity Properties Dialog Box)

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

Testing an English Query Model

### **Defining Relationship Phrasings**

In English Query, a relationship is an association between entities that describes what those entities have to do with one another. Relationships can be described as simple statements about entities (for example, *customers purchase products*). More than one join may be required to represent a single relationship.

Phrasings are ways to express a relationship in English. Types of phrasings include name, adjective, subset, preposition, verb, and trait phrasings. For example, *customers purchase products* is an example of a verb phrasing, and *department names are names of departments* is an example of a name phrasing. Using different phrasings are a good way to expand a model.

#### See Also

Creating an English Query Model

Defining Entities in an English Query Model

**Enabling SQL for an English Query Model** 

**Enabling Full-Text Search for an English Query Model** 

**Expanding an English Query Model** 

Testing an English Query Model

### **Prepositional Phrasings in Relationships**

A preposition phrasing describes a relationship between the subject and object entities that includes prepositions. For example, the phrase *patients are on medications (for illnesses) (at dosages) (for studies)* allows the following questions to be answered by English Query:

- "Who is on medication X?"
- "Who is on medication for heart disease?"
- "Who is on medication for the XYZ study?"
- "Who is on medication X for the XYZ study for heart disease?"
- "Who is on medication X at 100 mg for heart disease for the XYZ study?"

Likewise, use prepositional phrasing to answer questions about geographical locations, such as "How many cabins are in the Sierra Nevada mountain range?" and "How many cabins are in the Southern Hemisphere?" You would set up the hierarchy to answer these questions with the following phrasings:

- cabins are on mountains
- mountains are in mountain ranges
- mountain ranges are in continents
- continents are in hemispheres

Words commonly used as prepositions include about, above, across, before,

below, concerning, down, for, from, in, like of, on, over, past, regarding, since, through, till, to, toward, under, until, with, and without.

Phrasal prepositions include according to, along with, as to, because of, due to, in case of, in place of, instead of, up to, and with regard to.

### See Also

**Defining Relationship Phrasings** 

**Expanding an English Query Model** 

**Preposition Phrasing Dialog Box** 

### **Grouped Phrasings Examples**

Phrasings are a way to talk about a relationship in English. Choose phrasings that most closely reflect how users would ask their questions.

The following examples show when phrasings need to be grouped to correctly specify the relationship.

**Example 1:** Consider a database that contains information about people and their hair color. One phrasing that describes this relationship is the trait phrasing, such as *people have hair color*. However, this phrasing will not answer questions such as, "What is the color of John's hair?" For this, you need the phrasings *people have hair* and *hair has color*.

**Note** "Hair", in this case, is an entity that is not represented by a <u>database object</u>.

These two phrasings collectively describe the relationship between people and hair color. In order for English Query to treat these two phrasings as one logical unit, they need to be grouped.

**Example 2:** You have a table containing ages, sexes, and part counts that indicates how many parts were bought by people of each age and sex. You want to ask questions such as, "How many parts did young women buy?" This is a relationship among ages, sexes, parts (an entity not represented in the database) that have quantities, and people (another entity not represented in the database).

You must group the following phrasings: *people have ages* (a measurement adjective phrasing); *people have sexes* (a dynamic adjective phrasing); and *people buy parts* (a verb phrasing). These three phrasings collectively describe a single relationship.

**Example 3:** Consider a table containing suppliers, parts, and colors. You want to answer questions such as, "Who sells green parts?". This is a single relationship among suppliers, parts, and colors. You would need the following phrasings in a group: *suppliers supply parts* (verb phrasing) and *parts have colors* (adjective phrasing). Although you might consider creating separate relationships for these two phrasings, this would not supply the correct answer. In this table, the colors

of the parts are inherently dependent on who supplied them. If you were to create independent relationships for these two phrasings, then the question, "Who sells green parts," is necessarily interpreted as, "Find all of the suppliers and parts in the sales table such that the part also appears in the sales table with the color green" (in other words, "Who sells parts (in any color) that are also sold (by any supplier) in green").

Individual phrasings in a phrasing group can be used separately from the group. In the example above, a user could ask, "Who sells parts?" without referring to colors at all. However, it may not make sense to ask about some phrasings in the group without also including other required phrasings. For example, you may want the question "List the colors of the parts" to always be interpreted as "List the colors of the parts which suppliers supply." That way, the user clearly understands that colors of parts are known only in the context of suppliers supplying them. To do this, simply mark the required phrasings in the **Phrasing** dialog box.

**Example 4:** You have a table containing people and lunch times. You want to be able to ask, "What time did John eat his lunch?". This is a relationship between people, lunches (an entity not represented in the database), and lunch times. You need to group the following phrases: *people have lunches* (a trait phrasing) and *people eat lunches at lunch times*. *People each lunches at lunch times* is sufficient to answer, "What time did John eat lunch?". Because you also want to refer to "his lunch," you need the additional phrasing in the group.

#### See Also

**Defining Relationship Phrasings** 

**Expanding an English Query Model** 

### **Verb Phrasings in Relationships**

When a relationship between two entities can be expressed by an action word or verb, use verb phrasing to describe it (for example, *salespeople sell teachers briefcases from the warehouse*).

When specifying a phrasing, English Query provides the passive equivalent of the phrase. For example, specifying the verb phrasing *Salespeople sell customers products* also allows users to ask the <u>passive</u> question "Which products were sold to customers by which salespeople?".

In general, use <u>active voice</u>, rather than passive voice, in verb phrasings. For example, instead of creating the phrasing *products are sold to customers by salespeople*, create the phrasing *salespeople sell customers products*. By creating the phrasing in the active voice, you enable English Query to understand questions in both the active and passive voices. For example, it could answer both *Who sold John a lawnmower?* (active voice) and *What was sold to John by Fred?* (passive voice).

### **Converting a Passive Voice Phrasing to an Active Voice Phrasing**

Whenever a phrasing that has prepositional phrases is missing a <u>subject</u>, a <u>direct object</u>, or an <u>indirect object</u>, check to see whether any of the objects of the <u>prepositions</u> can be used to restate the phrase in the active voice. For example, in the phrasing *Cars are driven by people*, *cars* is the direct object, *are driven* is the verb, and *people* is the object of a preposition. This phrasing works better as *People drive cars*, and it can be converted by making *people* the subject. Similarly, *Customers give money to salespeople* can be converted to *Customers give salespeople money*.

**Note** Pay close attention to prepositional phrases using the prepositions *by* and *to*. Frequently, the objects of these prepositions can be changed into subjects and indirect objects, respectively.

### **Using Multiple Instances of an Entity**

In some cases, another instance of an entity is needed to create the desired

phrasing (for example, *persons migrate from countries to countries*). In this case, two instances of the **countries** entity are needed. The join condition determines your choice about which instance of the **countries** entity goes with each preposition. For the first instance, the join condition is *migration.from\_cid~country.id*. For the second instance, the join condition is *migration.to\_cid~country.id*.

#### See Also

Defining Relationship Phrasings

Expanding an English Query Model

Verb Phrasing Dialog Box

### **Adjective Phrasings in Relationships**

When a relationship between entities is expressed with an adjective, use adjective phrasing to define the relationship. The adjective can be:

- A single word not represented by a field.
- Part of a <u>database object</u> that represents an entity.
- A measurement.

When specifying the adjective phrasing that describes the relationship between the subject and the adjectives or entities that contain adjectives, you are providing information to answer questions such as:

- "Which people are old?"
- "Which people are young?"

Adjective phrasing also provides the information needed to answer trait-like questions for measurement and entity adjectives only:

• "What is John Smith's age," as well as "What is the age of John Smith?"

In addition, specifying a measurement adjective phrasing allows questions that use the <u>comparative</u> or <u>superlative</u> forms:

- "Is John older than Mary?" (comparative)
- "Who is the youngest employee?" (superlative)

#### See Also

**Defining Relationship Phrasings** 

## **Expanding an English Query Model**

# Using Command Relationships in an English Query Model

English Query interprets English questions (for example, "Who is the oldest patient?") and statements (for example, "Show the salespeople") as commands to display a set of data. In some applications, you may want to enhance English Query to understand other types of commands. Relationships that contain command phrasings are used to define these model-specific commands and are called command relationships. Command relationships do not use join tables or SQL conditions.

Similarly, entities used in command relationships do not use join paths or "how many" and "how much" properties. Entities in command relationships can be assigned mandatory roles, which means that they are required.

When command phrasings and entities are called in the object model, they require a command ID and a parameter ID, respectively.

### **Command Relationship Example**

An English Query application tracks an art collection, in conjunction with a database that contains information about the artists. A user states, "Display the paintings by xxx artist that are watercolors." This statement is translated into two commands to be executed by the application:

- The first is an SQL command, which fetches information from the database about paintings by xxx artist that are watercolors.
- The second command instructs the client application to display the actual paintings identified by the first command, rather than a display of textual data from the database.

This application uses model knowledge about paintings, artists, and medium, and a relationship containing a command phrasing. The command phrasing in this example has the verb "display" and the direct object "painting."

### **Mandatory Roles**

Mandatory roles is an enhancement to English Query that allows entities used as command relationship arguments to be required. Command relationships are not complete unless all (or a specific subset) of the parameters are specified. This is required for situations in which the target command has arguments that are mandatory.

For example, in the command, "ship (amounts of) products to customers on dates", the product and the customer and the amount must all be specified, or the command is incomplete.

For such roles, if the user does not specify a specific instance of the entity or a qualifier describing a subset of the entity, English Query prompts for such information. When a mandatory role has not been specified in the command or question, the English Query engine object model generates a clarification response with an **InputText** object for each missing mandatory role, using the prompt provided on the role by the author of the model. For example:

Question: Order 10 bottles of Coho Vineyard Merlot

InputText 1: Who should the order be sent to?

InputText 2: When should the order be sent?

In this example, the user provides a customer name for the first clarification and a date for the second clarification.

#### **Command IDs and Parameter IDs**

Command phrasings and entities require an ID when called in the object model. Command phrasings require a command ID, which is returned to the application and indicates which command relationship it is. Each entity used in a command phrasing requires a parameter ID. This ID is returned to the client application in a **Parameter** object and indicates which command parameter the object specifies.

Some command phrasings use entities that are not represented by <u>database objects</u>. These entities may simply be a static part of the phrasing, for example, "the Recycle Bin" in the command "Move *file*> into the Recycle Bin". Or, these entities may correspond to command arguments that are obtained

from the user's question rather than from the database (for example, in the command, "Raise the temperature of vat 6 by 12 degrees"). The command phrasing that is used would be "Raise the\_temperature of vats by temperature\_increase\_amounts". The entity "the\_temperature" is not represented by a database object and does not correspond to a command argument. So, you specify this by selecting "None" as the parameter type. The entity "temperature\_increase\_amounts" (also not represented by a database object) does correspond to a parameter of the command. As a result, you must indicate the data type of the parameter by using the **Parameter Type** property.

Parameters that correspond to nondatabase entities are returned to the client application through the object model as *value* **Parameter** objects, rather than as *field* **Parameter** objects. In the previous example, "12 degrees" would be sent as a **Parameter** with the **Value** property set to "12" and the **Units** property set to "degrees".

To create a command relationship

# **Verbs to Avoid in Command Relationships**

The following built-in verbs can return unexpected results if they are used in <a href="mailto:command relationships">command relationships</a>.

achieve	equal	name
add	exceed	rate
add up	fill	reach
adjust	find	relate
alphabetize	finish	reduce
ascent	help	rise
attain	include	subtract
average	gain	report
base	keep	sum
begin	go down	reprint
break down	go up	reshow
calculate	graph	save
call	know	sum up
cease	grow	search
change	identify	search out
chart	improve	select
commerce	increase	sequence
compare	locate	tell
compute	list	show
contain	lose	total
contract	look up	shrink
correlate	meet	total up
count	magnify	sort
decline	mention	unview
decrease	merge	stack
describe	miss	vary

display	order	stack up
divide	multiply	view
drop	plot	start
end	print	stop
enlarge	print out	store
expand	quit	subcount
fetch	raise	subtotal
find out	rank	widen

### **Authoring with Semantic Modeling Format**

English Query stores its information about the database structure and semantic information into an Extensible Markup Language (XML) vocabulary designed for this purpose. The objects in this XML grammar are quite general: entities and relationships, along with their (possible) associations with SQL tables, columns, and joins, and Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services (formerly OLAP) cubes, dimensions and levels.

Semantic Modeling Format (SMF) is an XML-based language that was created to allow greater accessibility for other applications to the model information in an English Query application and to enable developers to programmatically expand an English Query model. You can use SMF to create, read, and generate English Query models programmatically without writing the .eqp file format directly.

Typographic errors or the deletion of required text will cause compile errors when building the model.

#### **Universal Access to Model Information**

English Query provides Question Builder, a tool that allows presentation of the entities and relationships available in the model with any Web browser that supports Microsoft ActiveX® Script. To allow an English Query model to be navigated by any COM application, the **XMLDOMDocument** object, available with the Authoring object model, loads the SMF into a hierarchy. Thus, the English Query model information is easier to access when you are programming the application.

#### See Also

Adding Question Builder to an English Query Application

**Authoring Object Model** 

**Expanding an English Query Model** 

### **Testing an English Query Model**

Test the English Query model before compiling it into an application to make sure that the questions users are likely to ask are supported by the model and then modify it accordingly. After you are satisfied with the performance of the model, build it into a compiled English Query application (\*.eqd file).

The Suggestion Wizard is available within the Model Test window and is useful during the testing phase. You can start by asking a question that users are likely to ask. Suggestion Wizard presents suggested entities and relationships needed to answer the question.

#### **Model Test Window**

The **Model Test** window in English Query enables you to test the English-to-SQL translation of your model and the execution of the resulting database query before compiling the application.

**Note** For the data values to appear in the **Entity** list box in <u>Question Builder</u>, you must select **Sample Data** on the **Data Connections** tab before compiling the application.

In the **Model Test** window (available from the **Start** command on the **Debug** menu), you can use the **Analysis** feature to review the entities for functioning and for questions that return incorrect answers.

When testing a model, questions can be saved to a regression file with the **Save Query** command. A regression file is automatically created with each project. There can be multiple regression files, but questions are saved in the first regression file under the project in the Project Explorer window.

An output file can be added and then compared with the original file if the files have names that differ from one another. You can also promote an output file to the new regression file. Regression output is displayed in Extensible Markup Language (XML)-tagged format, which differentiates the questions, restatements, answers, and SQL statements for easier editing.

### **Suggestion Wizard**

The Suggestion Wizard, which is also available from the **Model Test** window toolbar, suggests entities and relationships for answering ambiguous questions.

The Suggestion Wizard analyzes the question and looks for unknown or ambiguous entities. For each of these entities, it displays a sentence in the form " <word> refers to an <entity list>." You may select an existing entity from the entity list, or create a new one by using the new entity button.

For pronouns or ambiguous entity names, simply associating the word with an entity is enough. For undefined entities, the Suggestion Wizard asks you to clarify the definition in one of several ways:

• As a synonym for the entity

This is appropriate if, for the purposes of the database, the word means exactly the same as some other word. For example, "films" might be a synonym for "movies."

- As an instance of the entity
   For example, "Clark Gable" might refer to an instance of entity "actor."
- As a subset of the entity type
   For example "documentaries" refers to "documentary," a subset of the entity movies.

After the entities have been added or clarified, you may have to define relationships among them. This is done by validating the correctness of the declarative sentences that the wizard generates from the question.

For relationships that are described as adjectives, the wizard displays a more complex set of sentences. If the sentence does make sense, you can define the meaning of the adjective relative to the base entity in one of the following ways:

- As an ignorable adjective, for cases in which the adjective isn't important from the standpoint of the database. For example, all customers are "valued" customers.
- As an adjective based on the value of another entity. There are two subcases of this:

- The other entity can either be a measurement of the base entity (for example, old movies are movies whose **age** entity is greater than 40).
- The other entity stores adjectives (or codes mapping to adjectives) describing the base entity (for example, "Disaster" movies are movies that have "Disaster" in their genre).
- As an adjective based on a SQL condition.

This option is not available for OLAP-only projects.

When you complete the Suggestion Wizard, it updates your model with new entities and relationships based on your answers.

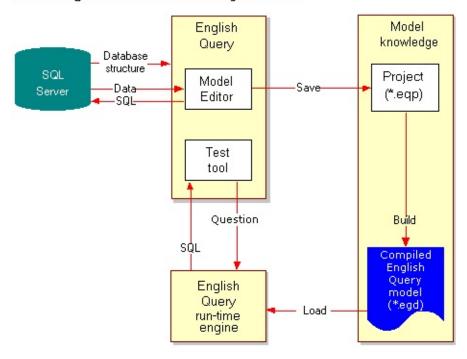
To save a regression file

## **Compiling an English Query Application**

The **Model Test** window in English Query enables you to test the English-to-SQL translation of your model and the execution of the resulting database query before compiling the application. When the model performs as expected, build it into a compiled English Query application (\*.eqd file).

**Note** For the data values to appear in the **Entity** list box in <u>Question Builder</u>, you must select **Sample Data** on the **Data Connections** tab before compiling the application.

Creating the Model and Project Files



To compile a model into an application

## **Deploying an English Query Application**

You can deploy an English Query application in several ways, including within a Microsoft® Visual Basic® or Microsoft Visual C++® application and on a Web page running on the Microsoft Internet Information Server (IIS). In the Web scenario, the interface of the application is with a set of Microsoft Active Server Pages (ASP).

**Note** English Query domain files (\*.eqd) are built to work with a specific version of the English Query engine. If you are upgrading from an earlier version of English Query, you must rebuild your English Query domain files from the project (\*.eqp) using the Model Editor.

After the English Query application is deployed, Question Builder helps the end user determine the information that is available in the English Query model.

To deploy an application to the Web

# Adding Question Builder to an English Query Application

Question Builder provides an interface for the end users of English Query applications. It uses the English Query model behind the application to provide easy question building and quick answers from the database. End users can find out what the model contains, what basic relationships are represented in the model, and what English phrases can be used to ask about the relationships.

Question Builder features a three-pane interface that makes the user's job of determining what information is available in the database easier. Question Builder graphically shows users what entities and relationships are present in the model, and provides drag-and-drop interaction and sample questions based on data in the application database. You specify Help for the end user in the English Query Model Editor.

Question Builder provides a listing of the entities, their corresponding attributes, and the relationships in which they participate. For each entity, an end user discovers:

- Its singular and plural forms.
- Any corresponding synonyms.

Using this information, the end user can reference the entity in a question.

For exploring the relations between two or more entities, Question Builder generates a list of relationships. After exploring the contents of the model, users may still need guidance in forming their questions. Question Builder provides two types of example questions:

- General questions.
- Questions about a specific instance of an entity.

#### **Integrating Question Builder into Applications**

Question Builder is a single Microsoft® ActiveX® control that can be integrated into many different applications, such as: ASP-based Web sites, Microsoft Visual Basic® applications, Microsoft Visual C++® programs, or any application that can act as an ActiveX script container. To see the Question Builder interface, run one of the sample models in \Program Files\Microsoft English Query\Samples\Models.

To see how to integrate Question Builder into an application, review the ASP applications in \Program Files\Microsoft English Query\Samples\Applications.

Question Builder provides Help for the user. To include the Question Builder Help, the following files must be in the same path as Question Builder. By default, when English Query is installed, they are in \Program Files\Common files\System\EQ80:

Mseqgrqb.ocx

Eqqbhlp.chm

#### **Programmatically Retrieving Model Information**

The Question Builder object model is a COM-automation interface used to programmatically retrieve information about an English Query model.

#### See Also

**Automatic Clarification of Questions** 

Question Builder Object Model

Redistributing the English Query Run-time DLLs

Sample Applications and Client Interfaces

## **Automatic Clarification of Questions**

The automatic clarification enhancements to English Query provide a way for a client application to automatically clarify certain classes of ambiguous questions when the user cannot or does not wish to provide the needed clarification. Although not all questions can be automatically clarified, these types of clarification questions can be automatically clarified for the user.

#### **Proper Noun Clarification**

Proper noun clarifications determine the entity to which an unknown proper noun refers, for example:

Question: How many awards did John Fortune win?

- A movie: SELECT \* FROM movies WHERE name = 'John Fortune'
- An actor: SELECT \* FROM actors WHERE fname = 'John' and lname =

### **Unique ID Clarification**

Unique ID clarifications determine the entity to which an unknown ID refers, for example:

Question: Show the total sales for 423.

Clarification question: Which of these do you mean by "423"?

- A customer:
   SELECT \* FROM customers WHERE id = 423
- An employee: SELECT \* FROM employees WHERE id = '423'

#### Name/ID Clarification

Name/ID clarifications determine whether an unknown word refers to the name or ID of the specified entity, for example:

Question: What did customer Smith order?

Clarification question: Which of these do you mean by "Smith"?

- The customer whose name is Smith:
   SELECT \* FROM customers WHERE name = 'Smith'
- The customer whose ID is Smith:
   SELECT \* FROM customers WHERE id = 'SMITH'

#### **Measurement Clarification**

Measurement clarifications determines the value parameters used by an adjective of a specified entity, for example:

Question: List the old salespeople.

Clarification question: An old salesperson is a salesperson whose age is over \_\_\_\_

SELECT average(age) FROM salespeople

#### **See Also**

Adding Question Builder to an English Query Application
Testing an English Query Model

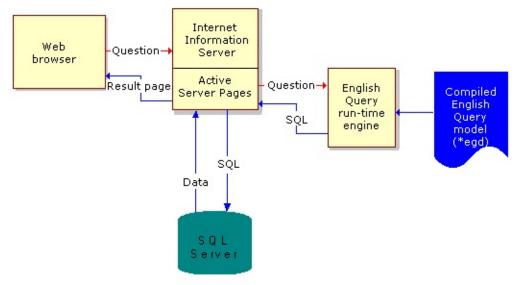
# Run-Time Environment for English Query Applications

At run-time, an end user of an English Query Web-based application connects to a Web page through Microsoft® Internet Explorer (or another Web browser), and enters a question. Internet Explorer then passes the question to Microsoft Internet Information Services (IIS), along with the URL of the Active Server Pages (ASP) page that executes Microsoft Visual Basic® Scripting Edition (VBScript).

The script passes the question to English Query for translation into SQL. English Query uses a model of the target database (in the form of a compiled English Query model) to parse the question and translate it into SQL. The script then retrieves the SQL code, executes it using Microsoft ActiveX® Data Objects (ADO), and displays the results as HTML.

**Note** English Query is not compliant with Microsoft Windows® 2000 logo requirements, because it appears within Microsoft Visual Studio® version 6.0.

English Query Application at Run-Time



#### See Also

Redistributing the English Query Run-time DLLs

## **Redistributing the English Query Run-time DLLs**

When including English Query capability in an application, the English Query run-time DLLs accompany the application. The exception to redistributing the English Query DLLs occurs when the English Query application is deployed on a Web server. In this case, the English Query run-time DLLs only need to be installed on the Web server.

Mseqgrqb.ocx and Mseqgrqb.cab are Microsoft® ActiveX® controls used to enhance the use of English Query models. For applications deployed to a Web site, end users can download these files from the site. It is neither necessary nor permissible to allow end users to download the English Query system DLLs from the Web.

Redistributing the English Query run-time DLLs involves copying all of the English Query DLLs and registering the necessary DLLs when the application is installed. If you need to redistribute the English Query DLLs, all users of English Query must be licensed Microsoft SQL Server<sup>TM</sup> users. Because the application usually accesses SQL Server data, each English Query user must have a SQL Server client access license.

You can redistribute the files shown in this table with an English Query application.

File name	Description
Mseqole.dll	English Query Server object model
Mseqbase.dll	English Query supplemental file
Mseqsql.dll	English Query supplemental file
Mseqmsg.dll	English Query supplemental file
Mseqconn.dll	English Query supplemental file
Mseqcore.eqd	English Query supplemental file
Mseqgrqb.ocx	English Query Question Builder ActiveX control
Mseqgrqb.cab	English Query cabinet file containing Mseqgrqb.ocx, the Help file (.chm); and the .inf file that tells Microsoft Internet Explorer where to copy the files, how to register the .ocx file, and how to uninstall the

control.

When English Query is installed, these files are installed by default in the \Program Files\Common Files\System\EQ80 directory. Before registering the files, the English Query application or Setup installs them in this common directory or creates the directory if it does not exist. The compiled English Query application (.eqd) (for example, Northwind.eqd) is copied to the end user's computer. By default, the .eqd file is located in the same directory that is used by the rest of the application.

#### **Registering the Necessary Files**

Both Mseqole.dll and Mseqgrqb.ocx must be registered using the Regsvr32.exe utility:

- To register Mseqole.dll, use "\path to the dll\Mseqole.dll". For example: Regsvr32 "\Program Files\Common Files\System\EQ80\Mseqc
- To register Mseqgrqb.ocx, use "\path to the location of the ActiveX controls\Mseqgrqb.ocx". For example:
   Regsvr32 "\Program Files\Common Files\System\EQ80\Mseq\xi

If you do not want the end users to see the dialog box that confirms successful registration, use the /s (silent) switch:

Regsvr32 "\Program Files\Common Files\System\EQ80\Mseqole.dll" /

If you want to clean up the end users' registries when English Query is uninstalled, use the  $/\mathbf{u}$  (uninstall) switch:

Regsvr32 "\Program Files\Common Files\System\EQ80\Mseqole.dll" /

#### See Also

Run-Time Environment for English Query Applications

## **Sample Applications and Client Interfaces**

English Query includes several sample applications and sample client interfaces, which you can modify for your own applications.

#### **Sample English Query Models**

English Query includes several sample models in the following folders in \Program Files\Microsoft English Query\Samples\Models:

#### \Pubs

Contains a sample English Query model for the Microsoft® SQL Server<sup>TM</sup> **pubs** sample database.

#### Northwind

Contains a model for the **Northwind** sample database, distributed with SQL Server version 7.0 and SQL Server 2000.

#### \FoodMart

Contains a model for the **FoodMart** sample OLAP database, distributed with SQL Server OLAP Services version 7.0, and **FoodMart 2000**, distributed with SQL Server 2000 Analysis Services.

#### **Sample Client User Interfaces**

English Query includes several sample client user interfaces, which can be modified to create your own English Query applications. These interfaces appear in \Program Files\Microsoft English Query\Samples\Applications in the following locations:

#### \ASP

Contains a complete Active Server Pages (ASP) sample application for English Query. It requires Microsoft Internet Information Server version 3.0 or later.

#### \VB

Contains a sample Microsoft Visual Basic® application for English Query. It requires Visual Basic version 6.0 or later and either the **PivotTable** or **FlexGrid** control.

#### • \VC

Contains a sample Microsoft Visual C++® application for English Query. It requires Visual C++ version 6.0 or later and either the **PivotTable** or **FlexGrid** control.

The ASP application includes a setup file, as well as an implementation of Question Builder.

The Visual Basic and Visual C++ sample applications use either the **PivotTable** or the **FlexGrid** control to display the results of Multidimensional Expression (MDX) queries. If the sample applications display the results of MDX queries in a Pivot Table (a Microsoft Office 2000 Web component), an MDX wrapper is used. At run-time, the sample application detects which of the following dynamic-link libraries (.dll) or ActiveX controls (.ocx) are installed on your computer.

To display the query results in a Pivot Table, the following files must be registered.

File	Description
Msowc.dll	This DLL contains the <b>PivotTable</b> control and is installed
	with Microsoft Office 2000.
Owcmdx.dll	This DLL contains the MDX wrapper and is installed with
	English Query.

If the **PivotTable** control is not installed, the sample applications use the **FlexGrid** control version 6.0 and Microsoft ActiveX Data Objects (Multidimensional) (ADO MD) version 2.5 to display the query results.

To display the query results using the **FlexGrid** control, the following files must be registered:

File	Description

Msflxgrd.ocx	This file contains the <b>FlexGrid</b> control.
Msadomd.dll	This DLL is required to use ADO MD 2.5.

## See Also

Compiling an English Query Application

**Deploying an English Query Application** 

Redistributing the English Query Run-time DLLs

Run-Time Environment for English Query Applications\_

## **Analysis Services in English Query**

Microsoft® SQL Server™ 2000 Analysis Services (formerly OLAP Services) is now integrated with English Query. Analysis Services is a middle-tier server for online transaction processing (OLTP).

Analysis Services includes a powerful server that constructs multidimensional cubes of data for analysis and provides rapid client access to cube information. The source data for Analysis Services is commonly stored in data warehouses in a relational database.

#### **Using Analysis Services with English Query**

With the integration of Analysis Services and English Query, client applications can produce answers to users' questions in either a multidimensional display or in traditional SQL rows.

Using Analysis Services also enhances the performance of queries processed by English Query.

When an English Query model has been enabled for Analysis Services, entities and relationships can be associated with OLAP cube information. The English Query Model Editor now includes an explicit representation of OLAP cube dimensions, levels, properties, measures, and facts.

The following example shows the resulting query and Multidimensional Expressions (MDX) of the English question "Show me the sales for stores in California for 1996 through 1998" that is run against the sample **FoodMart 2000** database:

WITH SET set\_1 AS 'filter([Time].[Year].members, [Time].currentment AND [Time].currentmember.name<="1998")'

MEMBER Measures.[Unit Sales ] AS 'sum(set\_1, Measures.[Unit Sal SELECT {Measures.[Unit Sales ]} ON COLUMNS,

filter([Store].[Store Name].members, ancestor([Store].currentmember, FROM [Sales]

For more information about Analysis Services, see **Analysis Services Overview**.

# See Also

**Enabling Analysis Services for an English Query Model** 

## **Naming Entities in the OLAP Project Wizard**

The OLAP Project Wizard automatically creates entities and relationships for OLAP cube objects. The OLAP Project Wizard suggests a set of entities and relationships for all <u>dimensions</u>, <u>levels</u>, <u>properties</u>, <u>measures</u>, and the one <u>fact table</u> for the <u>cube</u>.

Entities are named according to the dimension, level, measure, property, or cube to which they are associated. The singular form of the object name is used for dimension, level, property, and cube entities, and the plural form is used for measures.

If two entities have the same name, the object type is added to the entity that is being created. For example, because fact entities are created last, if the cube and the fact entities have the same name, the fact entity name is *<cube name>* fact.

To avoid badly named entities, such as **time** or **date** (which would cause problems for the English Query engine), the entity name of any time dimension is *<cube name> <dimension name>.* 

OLAP object	Description of generated entity
Dimension	Entity created for dimension if the dimension has more
	than one level.
Level	Entity created for each level unless the level appears in
	time dimension.
Measure	Entity created for each measure.
Property	Entity created for each member property of each level.
Fact	Entity created for the single fact table associated with each
	cube. The entity name is based on the cube name. If there
	is a name collision in this name, then entity is <i><cube< i=""></cube<></i>
	name> <fact>.</fact>

#### **Naming Sample**

From the **FoodMart 2000** sample database, the OLAP Project Wizard creates entities such as:

```
store
   store country
   store state
   store city
sales time (Because this is a time dimension, no entities are created for its
levels.)
product
   product family
   product department
   product category
   product subcategory
   brand name
   product name
customer
   country
   state
   city
   customer name
   gender
   age
   income
media type (Because this dimension is one level, no entities are created for it.
An entity is created for a single level in the Promotion Media dimension.)
```

**sale** (The cube name in singular form is the entity name associated with the fact

table.)

unit sale

store cost

store sale

sales count

store sales net

## Creating Relationships in the OLAP Project Wizard

If there is more than one <u>level</u> in a <u>dimension</u>, the OLAP Project Wizard creates <u>relationships</u> between a dimension's <u>entity</u> and the entities for each level in the dimension. In the example in which *Customer* is the dimension and the levels under it are *Country*, *State*, *City*, and *Customer Name*, the following relationships are created:

- Trait phrasings on the dimension-level relationships (dimension entity has lowest level entity) and on the level-level relationships (lower level entity has higher level entity). For example:
  - customers have cities is a dimension-level relationship, where customers is the dimension entity, and cities is the lowest level entity.
  - *Cities have states* is a level-level relationship, where **cities** is the lower-level entity, and **states** is the higher-level entity.
- Preposition phrasings on those relationships in which the lower-level entities are in higher-level entities. For example:
  - *cities are in states*, where **cities** is the lower-level entity and **states** is the higher-level entity.
- Preposition phrasings on those relationships in which the dimensions are in lower-level entities. For example:
  - *customers are in cities*, where **customers** is the dimension entity and **cities** is the lower-level entity.
- A name or ID relationship between the dimension entity and for the lowest-level of the dimension. For example:
  - *customer names are the names of customers* where **customer name** is the lowest-level entity, and **customers** is the dimension entity.

#### **Relationship Sample**

```
Here are a few relationships created by the OLAP Project Wizard:
sale
   sales have unit sales (fact to measures)
   sales have store costs
   sales have store sales
store
   stores names are the names of stores (dimension to bottom level)
   stores have sales (dimension to fact)
   stores have store managers (dimension to properties)
product
   product names are the names of products (dimension to bottom level)
   products have sales (dimension to fact)
   products have brand names
sale time (a single entity is proposed for the time dimension)
   sale times have sales
promotion media
   promotion medias have sales
promotion
   promotions have sales
customer
   customer names are the names of customers (dimension to bottom level)
   customers have sales (dimension to fact)
   customers have customer genders (dimension to properties)
education level
   education levels have sales
```

## gender

genders have sales

#### marital status

marital statuses have sales

## yearly income

yearly incomes have sales

## **Level-Level Relationships**

Users may want to ask questions about relationships between any combination of level entities in a dimension hierarchy. This can be accomplished by creating the minimum number of relationships necessary to allow a connection from any entity to any other entity, given the restriction that only two relationships can be used. Start by creating a relationship from each level entity to the next highest level entity. In the **Customer** dimension example, the following trait phrasing and preposition phrasing relationships are necessary:

- customers names are the names of customers
- customers have cities, customers are in cities
- *cities have states, cities are in states*
- states have countries, states are in countries

With the addition of *customers have states*, *customers are in states* relationship, users can ask the question, "What stores are in the United States?"

## **Additional Relationships**

Additional relationships depend on the number of level entities that exist. These additional relationships include the relationships between level and dimensions entities and property entities, levels and dimensions and the cube fact entity, and between measures and the cube fact entity.

### **Level and Dimensions Entities and Property Entities**

If the property entity is located at the bottom level of a dimension entity, create a relationship between the dimension entity and the property. If the **property** entity is not located at the bottom level of the **dimension** entity, create a relationship between the entity for the level containing the property and the property entity itself.

#### **Levels and Dimensions and the Cube Fact Entity**

Relationships are created between level and dimension entities and the entity associated with the <u>cube's fact table</u>. Although the fact table entity has relationships to each level entity of each dimension and each dimension entity, relationships are not created between the fact entity and the bottom-most level of a multilevel dimension, the name/ID level. For example, the following relationships and phrasings can occur between level (*cities*, *states*, and *countries*) and dimension entities (**customer**) and the cube fact entity (**sales**):

- customers have sales
- cities have sales
- states have sales
- countries have sales

#### **Relationships Between Measures and the Cube Fact Entity**

The OLAP Project Wizard also creates relationships between the OLAP cube

fact entity and the entities of all of the cube's <u>measures</u>. Using the previous example, the following relationships would be created between the cube fact entity (**sales**) and the measures (*unit sales*, *store cost*, *store sales*, *sales count*, and *store sales net*):

- sales have unit sales
- sales have store cost
- sales have store sales
- sales have sales count
- sales have store sales net

# **Analysis Services Requirements for (All) Level**

English Query generates Multidimensional Expressions (MDX) queries with the assumption that each <u>dimension</u> has an (All) <u>level</u>. The (All) level is the top level of a dimension that aggregates all the members of all the levels with no breakdown. When you create a dimension using Cube Editor, the dimension has an (All) level by default.

If a dimension does not have an (All) level, users may not get the results they expect. For example, for the question, "show the total unit sales", English Query generates this query:

# SELECT{Measures.[UnitSales]} ON COLUMNS FROM Sales

If the [Time] dimension has an (All) level, this query retrieves the sum of the unit sales for all of the years in the [Time] level. For example, if the [Time] dimension contains the years 1997 and 1998, this query retrieves the combined total sales for 1997 and 1998. However, if the [Time] dimension does not have an (All) level, this query retrieves the unit sales for only the first year in the [Time] dimension. For example, if the [Time] dimension contains the years 1997 and 1998, this query retrieves the total unit sales for 1997 only.

For more information about OLAP cubes and MDX syntax, see <u>Introduction to Cubes</u> and <u>Basic MDX</u>.

### **Object Model Reference**

English Query has three object models:

• Authoring Object Model

Can be used to automate the compilation of applications and enable alternate authoring tools.

• Engine Object Model

Can be used by any application that uses <u>COM</u> objects, such as a Microsoft® Visual Basic® or Microsoft Visual C++® program, or an Active Server Pages (ASP)-based Web site. The objects, methods, properties, and parameters can be used to edit Microsoft Visual Basic Scripting Edition (VBScript) sample scripts or to create scripts that make the compiled application available from ASP.

• Question Builder Object Model

Can be used to form questions that are semantically and syntactically valid for their applications.

## **Authoring Object Model**

The English Query Authoring object model is a <u>Component Object Model (COM)</u> object that provides the following services for an English Query application:

- Loading and saving an application
- Validating an object
- Compiling the English Query connection into an English Query application (.eqd)

Object creation and manipulation are performed by means of the **XMLDOMDocument** object model.

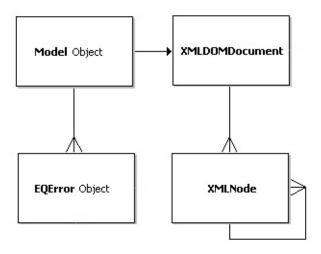
### **Automating the Compile Process of Your Applications**

When you add records to the target database, you can rebuild the English Query application from your English Query project so that it reflects the vocabulary changes. This simplifies the application building process and provides a way to automate rebuilding applications to check for errors.

### **Enabling Alternative Authoring Tools**

Using the Authoring object model, you can create and edit objects programmatically and compile and build the Semantic Modeling Format (SMF) file. In addition, in conjunction with the **XMLDOMDocument** object model, you can provide an alternative authoring environment in which other database access tools, such as Microsoft® Access, can be used with English Query.

# **Authoring Object Model Diagram**



### **Authoring Objects Summary**

This summary lists all the objects in the Authoring object model and their properties and methods.

<u>EQError Object</u>
 Properties/Methods

Description FileName PropertyIndicates the file name and/or the full path to the file with the error of the current English Query project (\*.eqp). ErrorCode PropertyIndicates a unique error code for the error. LineNumber PropertyDisplays the number of the line containing the error. ObjectID PropertyContains the ID of the object that generated the error. ObjectType PropertyIndicates the type of object that generated the error. Severity PropertyIndicates the type of error. Text PropertySpecifies the text of the error.

• EQModel Object

Properties/Methods

# **EQError Object**

Contains information about an error, warning, or hint generated during the compilation of an **EQModel** object or during the loading of a project or module.

### **Properties**

<u>FileName</u>	Indicates the file name and/or the full path to the	
	file with the error or the loaded English Query	
	project (*.eqp).	
<u>ErrorCode</u>	Indicates a unique error code for the error.	
<u>LineNumber</u>	Displays the number of the line containing the	
	error.	
ObjectID (EQError	Contains the ID of the object that generated the	
Object)	error.	
ObjectType (EQError	Indicates the type of object that generated the	
Object)	error.	
Severity	Indicates the type of error.	
Text (EQError Object)	Specifies the text of the error.	

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQModel Object** 

# **EQModel Object**

An object into which model information is loaded when a model is compiled and built.

**Note** EQModel is used as the default name of the project file when using one of the **Save** methods.

### **Properties**

<u>Errors</u>	Contains the collection of errors generated from
	the last method used on the <b>EQModel</b> object.

### **Methods**

<u>AutoModel</u>	Creates default entities and relationships for	
	tables and entities in the current project.	
Build	Builds an English Query compiled (*.eqd)	
	model from the current project.	
<u>Compile</u>	Performs all object validation checks on each	
	object in the project and returns a collection	
	of any errors, warnings, and hints found.	
<u>CreateProject</u>	Clears the contents of the current project and	
	sets the name of the new project.	
<u>FetchDatabaseStructure</u>	Retrieves schema information, such as table	
	or cubes, from the SQL or Microsoft® SQL	
	Server™ 2000 Analysis Services (OLAP)	
	database using the given Microsoft ActiveX®	
	Data Objects (ADO) connection object.	
<u>OpenModule</u>	Loads the contents of a English Query	
	module (*.eqm) and appends it to the existing	
	XMLDOMDocument object.	
<u>OpenProject</u>	Clears the <b>XMLDOMDocument</b> object and	
	then loads an English Query project (*.eqp)	

	or Semantic Modeling Format (SMF) file (*.smf).		
<u>OpenProjectFromXMLDOM</u>	Loads the contents of the current project from		
	an <b>XMLDOMDocument</b> object.		
<u>SaveModule</u>	Saves the contents of a module in the		
	XMLDOMdocument object into the		
	specified module file.		
SaveProject	Saves the contents of the		
	XMLDOMDocument object into project		
	(*.eqp) and module (*.eqm) files. The		
	SaveProject method retrieves data from the		
	<b>XMLDOMDocument</b> object and splits the		
	data into the main project file and multiple module files.		
CDurit A -CNAT			
<u>SaveProjectAsSMF</u>	Saves the contents of the current project to a merged single SMF (.smf) file.		
SaveProjectToXMLDOM	Saves the contents of the current project into		
	an <b>XMLDOMDocument</b> object.		
SaveProjectFile	Saves the content of the project file (*.eqp) in		
	the <b>XMLDOMDocument</b> object to the		
	specified file.		

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQError Object** 

# **Authoring Properties Summary**

This table lists all the properties in the Authoring object model and the objects to which they apply.

Property	Applies to
ErrorCode Property	EQError Object
FileName Property	
LineNumber Property	
ObjectID Property	
ObjectType Property	
Severity Property	
Text Property	
Errors Property	EQModel Object

# **ErrorCode Property**

Indicates a unique error code for the error.

### **Applies To**

**EQError Object** 

### **Syntax**

*EQError*.**ErrorCode** [= value]

#### **Parts**

**EQError** 

Object variable that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object or during the loading of a project or module.

value

Returned <u>integer</u> that contains the unique error code for the error.

#### Remarks

This property is read-only.

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQModel Object** 

# **Errors Property**

Contains the collection of errors generated from the last method used on the **EQModel** object.

### **Applies To**

**EQModel Object** 

### **Syntax**

*EQModel*.**Errors** [= value]

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

value

Collection of errors that were generated from the last method used on the **EQModel** object.

### **Remarks**

This property is read-only.

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQError Object** 

### FileName Property

Indicates the file name and/or the full path to the file with the error or the loaded English Query project (\*.eqp).

### **Applies To**

**EQError Object** 

### **Syntax**

*string* = *object***.Filename** 

#### **Parts**

string

Returned string that contains the full path and file name of the loaded project (\*.eqp) or file with the error.

object

Object variable that represents the **EQError** object.

### Remarks

This property is read-only. It may be empty if the error is not associated with a specific file.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

# **LineNumber Property**

Displays the number of the line containing the error.

### **Applies To**

**EQError Object** 

### **Syntax**

*EQError*.LineNumber [= value]

#### **Parts**

**EQError** 

<u>Object variable</u> that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object or during the loading of a project or module.

value

Returned <u>integer</u> indicating the number of the line containing the error.

#### Remarks

This property is read-only. It may be empty if the error is not associated with a specific line number.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQModel Object** 

### **ObjectID Property (EQError Object)**

Contains the ID of the object that generated the error.

### **Applies To**

**EQError Object** 

### **Syntax**

string = EQError.ObjectID

#### **Parts**

string

Returned <u>string</u> containing the ID of the object that generated the error.

**EQError** 

<u>Object variable</u> that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object, or during the loading of a project or module.

#### Remarks

This property is read-only. It may be empty if the error is not associated with a specific object.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQModel Object** 

ObjectID Property (HelpCmd Object)

# ObjectType Property (EQError Object)

Indicates the type of object that generated the error.

### **Applies To**

**EQError Object** 

### **Syntax**

EQError.ObjectType [= value]

#### **Parts**

**EQError** 

Object variable that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object, or during the loading of a project or module.

value

Returned enum indicating the type of object that generated the error.

Object type	Value	Description	
eqObjGlobal	0	Global property or unspecified object	
eqObjEntity	1	Entity	
eqObjRelationship	2	Relationship including model relationships	
eqObjPhrasing	3	Phrasing	
eqObjTable	4	Table	
eqObjField	5	Field	
eqObjJoin	6	Join	
eqObjRole	7	Role	
eqObjDictEntry	8	Dictionary entry	
eqObjProject	9	Project	

eqObjModule	10	Module	
eqObjCube	11	OLAP cube	
eqObjDimension	12	OLAP dimension	
eqObjLevel	13	OLAP level	
eqObjProperty	14	OLAP property	
eqObjMeasure	15	OLAP measure	

### Remarks

This property is read-only.

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQModel Object** 

ObjectType Property (HelpCmd Object)

# **Severity Property**

Indicates the type of error.

### **Applies To**

**EQError Object** 

### **Syntax**

*EQError*.**Severity** [= value]

#### **Parts**

**EQError** 

Object variable that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object or during the loading of a project or module.

value

A returned enum indicating the type of error.

Error type	Value	Description
eqSevInfo	0	Hint indicating a modification that the user may want to make to improve the model.
eqSevWarning		Warning about a potentially incorrect model, such as the presence of more than one possible join path from the join table of the relationship object to the table.
eqSevError	2	Error that must be resolved before the model can be built.

### Remarks

This property is read-only.

### See Also

Compile-Time Error Messages (Authoring Object Model)

EQModel Object

### **Text Property (EQError Object)**

Specifies the text of the error.

### **Applies To**

**EQError Object** 

### **Syntax**

string = EQError.**Text** 

#### **Parts**

string

Returned string that represents the text of the error

**EQError** 

Object variable that provides information about an error, warning, or hint generated during the compilation of an **EQModel** object or during the loading of a project or module.

#### Remarks

This property is read-only.

### See Also

Compile-Time Error Messages (Authoring Object Model)

Text Property (TextInput Object)

# **Authoring Methods Summary**

This table lists the methods of the Authoring object model and the objects to which they apply.

Method	Applies to
AutoModel Method	EQModel Object
Build Method	
Compile Method	
CreateProject Method	
FetchDatabaseStructure Method	
OpenModule Method	
OpenProject Method	
OpenProjectFromXMLDOM Method	
SaveModule Method	
SaveProject Method	
SaveProjectAsSMF Method	
SaveProjectFile Method	
SaveProjectToXMLDOM Method	

### **AutoModel Method**

Creates default entities and relationships for database objects in the current project.

### **Applies To**

**EQModel Object** 

### **Syntax**

*EQModel*.**AutoModel**(*ObjectType*, *ObjectID*, *ModuleID*)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

### *ObjectType*

<u>Integer</u> that represents the type of object and relationships that you want to generate:

Type of object	Value	Description
eqObjEntity	1	Entity
eqObjModule	10	Module file
eqObjTable	4	Table
eqObjCube	11	OLAP cube
eqObjDimension	12	OLAP dimension
eqObjProject	9	Project file

### **ObjectID**

<u>String</u> containing the ID of the module, table, or entity for which you want to automatically create entities and relationships. Ignored if **ObjectType** is **eqObjProject**.

### ModuleID

ID of the module in which you want the objects created.

### Remarks

*ModuleID* is stored as an attribute of the **MODULE** element in the **XMLDOMDocument** object.

### **See Also**

Compile-Time Error Messages (Authoring Object Model)

### **Build Method**

Builds an English Query compiled (\*.eqd) model from the current project.

### **Applies To**

**EQModel Object** 

### **Syntax**

EQModel.Build(Filename, [SQLConnection], [OLAPConnection])

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

#### **Filename**

<u>String</u> containing the full path and file name of the English Query model to which the model is written.

#### **SQLConnection**

An object variable that represents the Microsoft® ActiveX® Data Objects (ADO) connection object to use for executing SQL queries against the target database. Optional.

**Note** Although this argument is optional, if you want to load values from the database, you must include it.

#### **OLAPConnection**

An object variable that represents the ADO connection object to use for executing Microsoft SQL Server<sup>TM</sup> 2000 Analysis Services (OLAP) queries against the target database. Optional.

**Note** Although this argument is optional, if you want to load values from the database, you must include it.

### **Remarks**

Errors generated by the **Build** method are the same as the errors generated by the **Compile** method.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

### **See Also**

Compile-Time Error Messages (Authoring Object Model)

**Compile Method** 

Data Connection Tab (Project Properties Dialog Box)

**EQError Object** 

### **Compile Method**

Performs all object validation checks on each object in the project and returns a collection of any errors, warnings, and hints that are found.

### **Applies To**

**EQModel Object** 

### **Syntax**

*EQModel.***Compile(**[*SQLConnection*]**,** [*OLAPConnection*]**)** 

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**SQLConnection** 

An object variable that represents the Microsoft® ActiveX® Data Objects (ADO) connection object to use for executing SQL queries against the target database. Optional.

**Note** Although this argument is optional, if you want to load values from the database, you must include it.

#### **OLAPConnection**

An object variable that represents the ADO connection object to use for executing Microsoft SQL Server<sup>™</sup> 2000 Analysis Services (OLAP) queries against the target database. Optional.

**Note** Although this argument is optional, if you want to load values from the database, you must include it.

#### Remarks

The compiled project is loaded into the English Query engine, along with vocabulary from the database.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

### **See Also**

Compile-Time Error Messages (Authoring Object Model)

**Build Method** 

**EQError Object** 

# **CreateProject Method**

Clears the contents of the current project and sets the name of the new project.

### **Applies To**

**EQModel Object** 

### **Syntax**

*EQModel*.**CreateProject**(*ProjectName*)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

*ProjectName* 

String containing the name of the project to be used as the file name when the project is saved.

#### See Also

OpenProject Method

OpenProjectFromXMLDOM Method

SaveProject Method

SaveProjectAsSMF Method

SaveProjectFile Method

SaveProjectToXMLDOM Method

### FetchDatabaseStructure Method

Retrieves schema information, such as tables or cubes, from the SQL or Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services (OLAP) database by using the given Microsoft ActiveX® Data Objects (ADO) connection object.

### **Applies To**

**EQModel Object** 

### **Syntax**

EQModel.FetchDatabaseStructure(DBConnection, ModuleID)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

#### **DBConnection**

An object variable that represents the ADO connection object to use for executing database queries against the target database.

#### *ModuleID*

The ID of the module in which you want the objects created.

#### **Remarks**

*ModuleID* is stored as an attribute of the **MODULE** element in the **XMLDOMDocument** object.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

EQError Object

# **OpenModule Method**

Loads the contents of an English Query module (\*.eqm) and appends it to the existing **XMLDOMDocument** object.

## **Applies To**

**EQModel Object** 

## **Syntax**

EQModel.OpenModule(ModuleName)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

*ModuleName* 

String containing the full path and name of the module to be opened.

#### **Remarks**

The **OpenModule** method adds the module reference to the project section in the **XMLDOMDocument** object.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

### See Also

Compile-Time Error Messages (Authoring Object Model)

**EQError Object** 

OpenProject Method

# SaveModule Method

# **OpenProject Method**

Clears the **XMLDOMDocument** object, and then loads an English Query project (\*.eqp) or Semantic Modeling Format (SMF) file (\*.smf).

# **Applies To**

**EQModel Object** 

## **Syntax**

*EQModel.***OpenProject**(*FileName*)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

*FileName* 

String containing the full path and file name of the project to be opened.

#### **Remarks**

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

## See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

SaveProject Method

# OpenProjectFromXMLDOM Method

Loads the contents of the current project from an **XMLDOMDocument** object.

# **Applies To**

**EQModel Object** 

## **Syntax**

EQModel.OpenProjectFromXMLDOM(XMLDOMDocument)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**XMLDOMDocument** 

Object variable that contains the project to be opened.

#### Remarks

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

English Query 2000 is compiled with XML 2.0, whereas SQL Server 2000 uses XML 2.6. To prevent compilation errors, select XML 2.0 in the References section of your English Query project.

# See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

OpenProject Method
SaveProject Method

## SaveModule Method

Saves the contents of a module in the **XMLDOMdocument** object into the specified module file.

## **Applies To**

**EQModel Object** 

## **Syntax**

*EQModel*.**SaveModule**(*ModuleID*, [*SaveAsName*])

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**ModuleID** 

String variable containing the module ID.

**SaveAsName** 

A string variable containing the path and file name of the module to which you want to save the module. Optional.

**Note** If this argument is omitted, **SaveModule** uses the path and file name specified by the contents of the **MODULE** element that defines the module in the **XMLDOMDocument** object.

### **Remarks**

*ModuleID* is stored as an attribute of the **MODULE** element in the **XMLDOMDocument** object.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

# See Also

Compile-Time Error Messages (Authoring Object Model)

**EQError Object** 

OpenModule Method

# **SaveProject Method**

Saves the contents of the **XMLDOMDocument** object into the English Query project (\*.eqp) and English Query module (\*.eqm) files. The **SaveProject** method retrieves data from the **XMLDOMDocument** object and divides the data into the main project file and multiple module files.

## **Applies To**

**EQModel Object** 

### **Syntax**

EQModel.SaveProject

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

#### Remarks

Paths and file names are derived from the names of the modules as specified in the **XMLDOMDocument** object.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

## See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

OpenProjectFromXMLDOM Method

# SaveProjectAsSMF Method

Saves the contents of the current project to a merged single Semantic Modeling Format (SMF) file (\*.smf).

## **Applies To**

**EQModel Object** 

## **Syntax**

*EQModel*.SaveProjectAsSMF(Filename)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**Filename** 

String variable containing the path and file name to which you want to save the project.

#### Remarks

The collection of **EQErrors** encountered while saving the project is placed in *EQModel*.**Errors**.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

SaveProject Method

# SaveProjectFile Method

Saves the content of the English Query project file (\*.eqp) in the **XMLDOMDocument** object to the specified file.

## **Applies To**

**EQModel Object** 

## **Syntax**

EQModel.SaveProjectFile(ModuleID, [SaveAsName])

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**ModuleID** 

String variable containing the module ID.

**SaveAsName** 

A string variable containing the path and file name of the module to which you want to save the module. Optional.

**Note** If this argument is omitted, **SaveProjectFile** uses the path and file name specified by the contents of the **MODULE** element that defines the module in the **XMLDOMDocument** object.

### **Remarks**

Paths and file names are derived from the names of the modules as specified in the **XMLDOMDocument** object.

The collection of **EQErrors** encountered while saving the project is placed in *EQModel*.**Errors**.

# See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

SaveProject Method

# SaveProjectToXMLDOM Method

Saves a copy of the current project into an **XMLDOMDocument** object.

## **Applies To**

**EQModel Object** 

## **Syntax**

EQModel.SaveProjectToXMLDOM(XMLDOMDocument)

#### **Parts**

**EQModel** 

Object variable into which model information is loaded.

**XMLDOMDocument** 

Newly created object variable to which the project is saved.

#### Remarks

The changes made to the **XMLDOMDocument** are made to a private copy and do not alter the English Query model unless the project is opened and then saved. Use the **OpenProjectFromXMLDOM** method to load the private copy into English Query and update English Query's copy of the model. Then use the **SaveProject** method to save the changes.

The errors encountered while compiling the English Query project are placed in *EQModel*.**Errors**.

English Query 2000 is compiled with XML 2.0, whereas SQL Server 2000 uses XML 2.6. To prevent compilation errors, select XML 2.0 in the References section of your English Query project.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

**CreateProject Method** 

**EQError Object** 

OpenProjectFromXMLDOM Method

SaveProject Method

# English Query

# **Authoring Sample Script**

This sample script creates a new English Query model using the Authoring object model. This script:

- Loads the schema from the database.
- Runs the project wizard and uses the **AutoModel** method to create entities and relationships.
- Saves the project and then builds it into an English Query Domain (.eqd) file.

You must have a system data source name (DSN) that points to a **Northwind** database, or alternately, you can specify any database from which you want to create a model. You can copy this sample code and paste it directly into a Microsoft® Visual Basic® Scripting Edition (VBScript) file:

```
Dim adoConn, szEqmID, szEqm, szEqp, szEqd, szPath, szDatabase szPath = "c:\temp\Northwind" 'Specify a location to store the szDatabase = "Northwind" 'Specify a system DSN on your local c
```

```
Public EQModel
On Error Resume Next
Set EQModel = CreateObject("MSEQ.Model")
```

EQModel.FetchDatabaseStructure adoConn, szEqmID 'Import the CheckForErrors

Const EQOBJMODULE = 10 This constant matches constant def EQModel.AutoModel EQOBJMODULE, szEqmID, szEqmID 'Pr CheckForErrors

EQModel.SaveModule szEqmID, szEqm 'Save the .eqm file. CheckForErrors

EQModel.SaveProjectFile szEqp 'Save the .eqp file. CheckForErrors

EQModel.Build szEqd, adoConn 'Build the project to make an .ec CheckForErrors

Sub CheckForErrors()

'Check Visual Basic Err object to see whether the method call failed.

If err.Number <> 0 Then

MsgBox ("OM Method called failed: " & err.Description)

WScript.Quit (1)

End If

'Check the EQError object to see whether EQ errors, warnings, or hints Dim EqSevInfo, EqSevWarning, EqSevError, colErrors, szMsg EqSevInfo = 0: EqSevWarning = 1: EqSevError = 2: szMsg = "" 'Report only the Warnings and Errors, skip the Info hints. Set colErrors = EQModel.Errors For i = 0 To colErrors.Count - 1 Select Case colErrors(i). Severity Case EqSevInfo szMsg = szMsg + "Hint: " & colErrors(i).Text + Chr(10)

Case EqSevWarning

szMsg = szMsg + "Warning: " & colErrors(i).Text + Chr(10)

```
Case EqSevError
szMsg = szMsg + "Error: " & colErrors(i).Text + Chr(10)
End Select
Next
If szMsg <> "" Then
MsgBox (szMsg)
End If
End Sub
```

# English Query

# **Engine Object Model**

English Query objects can be used to deploy English Query applications on the Internet. Objects, methods, properties, and parameters are available for editing a sample script written in Microsoft® Visual Basic® Scripting Edition (VBScript) or for creating a new script that makes the compiled application available from Active Server Pages (ASP). For more information about this run-time system, see Deploying an English Query Application.

English Query is a multithreaded, automation server. In addition to VBScript (including the Microsoft ActiveX® Server Framework), it can be called from Microsoft Visual Basic® version 4.0 or later, or any other programming language that supports ActiveX controls.

The English Query engine object model does not presuppose any particular language, although the English Query server currently supports only the English language.

## **Parsing English Queries**

The English Query server provides the parsing service. English Query translates English questions or requests about data in a database server into query statements. For example, it converts the question, "How many products did we ship in 1996?" into a query statement. Likewise, it converts a request, such as "Show me the suppliers this year," into query statements. Therefore, an end user can easily retrieve information stored in the database without having to know the physical structure of the database.

English Query provides other responses to English questions, such as requests for clarification and error checking and handling.

In addition, when it parses an English question or request, the English Query server can respond with:

- A collection of database queries to execute.
- An answer in English.

- A request for end-user clarification.
- A parsing error (for example, an error caused by references to concepts not defined in the model).

After the English Query server parses an English question or request into one of these responses, the client acts on it. For example, English Query parses the question and may return a collection of database queries. The client then executes the queries. The client is also responsible for all interactions with the end user, including getting the initial English Query request from the user and displaying the data resulting from executing a query to the end user.

**Note** Using the database query generated by the English Query server depends on the availability of a compatible database interface, such as Microsoft ActiveX Data Objects (ADO).

# **English Query Engine Objects and Collections**

Some English Query engine object types have a corresponding collection. A collection includes all the existing objects of that type. For example, the **Commands** collection contains all open Command objects. Each collection is owned by another object at the next higher level in the hierarchy. For example, the **Commands** collection is owned by the **CommandResponse** object.

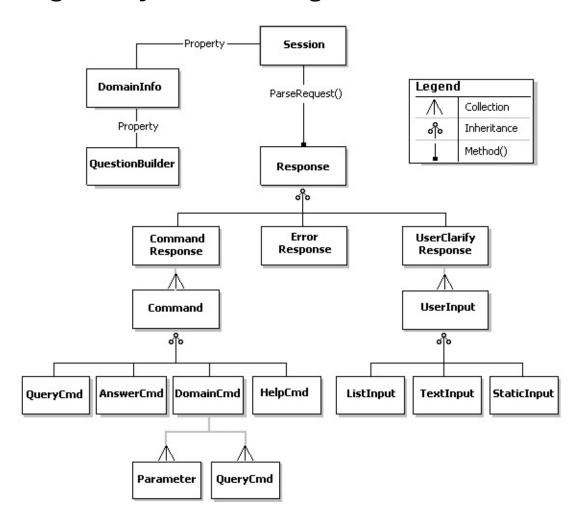
The table shows the five categories of English Query engine objects and collections.

Object category	Object/Collection name
Domain management	<b>DomainInfo</b> object
	QuestionBuilder object
End-user sessions	EndCommands collection
	Session object
Command responses	AnswerCmd object
	<b>Command</b> object
	<b>CommandResponse</b> object
	<b>Commands</b> collection
	<b>DomainCmd</b> object
	EntityInfo object

	FieldInfo object ObjectReference object
	Pivot object
	QueryCmd object
	Response object
	Substitution object
Parsing error responses	ErrorResponse object
	Response object
Clarification responses	Listlnput object
	Response object
	StaticInput object
	TextInput object
	UserInput object
	UserClarifyResponse object
	UserInputs collection

# English Query

# **Engine Object Model Diagram**



# English Query

# **Engine Objects Summary**

This summary lists all the objects in the Engine Object model and the properties and methods for each.

AnswerCmd Object

Properties/Methods

Provides the English answer text to a question that cannot be answered with data in the database.

• Command Object

Properties/Methods

• CommandResponse Object

Properties/Methods

DomainCmd Object

Properties/Methods

• DomainInfo Object

Properties/Methods

• ErrorResponse Object

Properties/Methods

• HelpCmd Object

Properties/Methods

• ListInput Object

Properties/Methods

• ObjectReference Object

Properties/Methods

• Parameter Object

# Properties/Methods

• QueryCmd Object

Properties/Methods

• Response Object

Properties/Methods

• Session Object

Properties/Methods

• StaticInput Object

Properties/Methods

• TextInput Object

Properties/Methods

• <u>UserClarifyResponse Object</u>

Properties/Methods

• <u>UserInput Object</u>

Properties/Methods

# **AnswerCmd Object**

Contains the answer to a question that is not in the database (for example, "What time is it?"). English Query can answer the question directly, without submitting the database query to the client to be executed.

# **Properties**

Answer Property	Provides the English answer text to a question
	that cannot be answered with data in the
	database.

#### Remarks

The **AnswerCmd** object has all the properties and methods of a **Command** object.

#### See Also

**Command Object** 

**CmdID Property** 

**CommandResponse Object** 

**QueryCmd Object** 

Response Object

Type Property (Response Object)

# **English Query**

# **Command Object**

Provides the client with information about what data or command the end user requested, such as a query, an answer, a user request for help, or a user request to exit command.

# **Properties**

CmdID Property	Contains an integer or a symbolic constant
	representing a command, such as a query,
	an answer, a user request for help, or a user
	request to exit.

### See Also

AnswerCmd Object

**Commands Collection** 

**DomainCmd Object** 

**QueryCmd Object** 

# **CommandResponse Object**

Returned from the **Session.ParseRequest** method when the English request is parsed into a collection of commands, which includes database commands.

## **Collections**

Commands Collection	Contains commands for the client to
	execute.

## **Properties**

Restatement Property	An English interpretation of the end user's
	question or request.

## **Remarks**

The integers that represent the type of response object are described in the **Type** property.

The **CommandResponse** object also has all the properties and methods of the **Response** object.

### See Also

**Response Object** 

Type Property

# **English Query**

# **DomainCmd Object**

Provides the client with information about the <u>command relationship</u> that the end user requested.

## **Collections**

Parameters Collection	A collection of <b>Parameter</b> objects that are
	parameters for a command relationship.
QueryCommands Collection	A collection of <b>QueryCmd</b> objects, which
	contain the queries that must be executed
	by the client application to obtain the
	parameters for a command relationship.

# **Properties**

CommandName Property	Contains a string representing a command
	relationship.

## **Remarks**

The **DomainCmd** object also has all the properties and methods of a **Command** object.

## See Also

**AnswerCmd Object** 

**CmdID Property** 

**Commands Collection** 

**QueryCmd Object** 

# **DomainInfo Object**

Encapsulates information about the entities and relationships in a particular model. It represents the information in the English Query application (\*.eqd) file. You do not use this information directly.

## **Properties**

Closed Property	Indicates the open and closed status of the
	DomainInfo object.
FullName Property	Indicates the path of the <b>DomainInfo</b> object.
QuestionBuilder Property	Contains the Question Builder information
	(the <b>QuestionBuilder</b> object) for the
	DomainInfo Object.

## **Methods**

Closes the <b>DomainInfo</b> object.	
--------------------------------------	--

#### Remarks

All information about the English Query application belongs to a model. This includes the extracted database structure, the semantic objects (such as entities and relationships), application defaults, and dictionary entries.

The **DomainInfo** object encapsulates information about the entities and relationships in a particular model. This information determines which English questions and statements the English Query server can and cannot parse.

You do not usually need to use this object directly. This object provides support for programmers who design English Query system management tools or modelinformation authoring tools.

Many end-user sessions can share the same model information object. If a

**DomainInfo** object is not already loaded when an **InitDomain** is called to initialize a session, the English Query server opens the **DomainInfo** and loads it into the English Query knowledge base. Otherwise, the session is attached to the existing **DomainInfo**. Usually, models are not unloaded when the last session attached to the **DomainInfo** is closed. Instead, you can call the **Close** method on the **DomainInfo** object.

Each end-user session is associated with a model for the duration of the session. The model must be initialized immediately after the session is created with **Session** by calling **InitDomain**.

#### See Also

**DomainInfo Property** 

**Session Object** 

# **ErrorResponse Object**

Returned from **Session.ParseRequest** method when the English request was not understood.

# **Properties**

Description Property	Contains a description of the error to be
	displayed to the end user.

#### **Remarks**

This object contains parsing errors.

The integers that represent the type of response object are described in the **Type** property.

The **ErrorResponse** object has all the properties and methods of the **Response** object.

### See Also

ParseRequest Method

Response Object

**Session Object** 

**Type Property** 

# **HelpCmd Object**

Provides the Help for an English Query application, entity, or relationship.

# **Properties**

HelpText Property	Specifies Help text of the application for an
	entity.
HelpType Property	Contains an integer representing the type of
	object for which Help is requested.
ObjectID Property (HelpCmd	Contains the ID of the object for which
Object)	Help is being requested.
ObjectType Property	Contains an integer representing the type of
(HelpCmd Object)	object for which Help is requested.

# **Remarks**

The **HelpCmd** object also has all the properties of a **Command** object.

# **See Also**

**CmdID Property** 

**Command Object** 

# **ListInput Object**

Specifies a list from which end users can choose an option to clarify their question.

# **Properties**

Caption Property	Specifies the recommended caption for one of
	three user input types: <b>ListInput</b> , <b>StaticInput</b> ,
	and <b>TextInput</b> .
Items Property	Specifies the array of strings in the list from
	which the user chooses.
ItemCount Property	Specifies the number of items in the list from
	which the end user chooses.
Selection Property	Specifies the end user's selection from the list.

### **Remarks**

The end user chooses an option from a list of available options.

# **See Also**

**StaticInput Object** 

**TextInput Object** 

Type Property (UserInput Object)

**UserInput Object** 

**UserInputs Collection** 

# **ObjectReference Object**

Refers to an object in the model (for example, an entity or relationship).

# **Properties**

ObjectID Property	Contains the ID of the object that is being
(ObjectReference	referenced.
Object)	
ObjectType Property	Contains an integer representing the type of
(ObjectReference	object that is being referenced.
Object)	

## **Remarks**

For example, the query "Who sold the most books last year" has four **ObjectReference** objects (three entities and one relationship):

- salespeople (entity)
- books (entity)
- salesdate (entity)
- *salespeople sell books on salesdate*.(relationship)

# **Parameter Object**

A parameter for a <u>command relationship</u>.

#### **Collections**

Parameters Collection	A collection of <b>Parameter</b> objects, which are
	parameters for a command relationship.

## **Properties**

FieldCount Property	Contains the number of fields in the <b>Fields</b>
	property array.
Fields Property	Contains an array of the fields in the specified
	query command, which yields the values of a
	parameter of a command relationship.
ParameterID Property	Contains an integer identifying a parameter of a
	command relationship.
ParameterType Property	Contains an integer representing a parameter for
	a command relationship.
QueryCommand	Contains an index of the query command in the
Property	<b>QueryCommands</b> collection that yields the
	values of a parameter of a command relationship.
Units Property	Contains a description that qualifies the
	parameter values, such as <i>percent</i> or <i>dollars</i> .
Value Property	Contains a parameter value that represents a
	parameter for a command relationship.

#### **Remarks**

Parameters can be of two types: **Value** and **Field**. A **Value** parameter is indicated by parameter type **nlParamValue**. A **Value** parameter has the properties of

**Value** and **Units**. A **Field** parameter is indicated by the parameter type **nlParamField**. It has the properties of **QueryCommand** and **Fields**.

See Also

**DomainCmd Object** 

# **QueryCmd Object**

Instructs the client to execute a database query.

## **Properties**

DisplayRows Property	Specifies the number of rows the client displays
	to answer the end user's question.
DisplayToUser Property	Specifies whether to display the results of the
	query to the user or to execute the database query
	without displaying the results.
DBMSType Property	Specifies the type of database used for the
	database query.
IgnoreError Property	Specifies whether the client ignores database
	errors resulting from executing the command.
QueryText Property	Returns the query text to execute.
SQL Property	Returns the query text to execute.
TableCaption Property	Specifies the suggested English caption for the
	result table.
<u>TrueFalseAnswer</u>	Specifies whether the client shows True or False
<u>Property</u>	as an answer instead of a table.

### Remarks

The **QueryCmd** object also has all the properties and methods of a **Command** object.

### See Also

AnswerCmd Command

**CmdID Property** 

**Command Object** 

## **Response Object**

Returned from the **Session.ParseRequest** method when the English request is parsed into a collection of database queries.

## **Properties**

Type Property (Response	Contains an integer or a symbolic constant
Object)	representing a type of <b>Response</b> object.

#### **Remarks**

The **Type** property contains a value or symbolic constant that represents one of three response types: **CommandResponse**, **ErrorResponse**, or **UserClarifyResponse**.

### **See Also**

AnswerCmd Object

**CommandResponse Object** 

**ErrorResponse Object** 

**QueryCmd Object** 

**Reply Method** 

**UserClarifyResponse Object** 

# **Session Object**

Represents the current state of the end-user session with the English Query server.

# **Properties**

ClarifySpellingErrors Property	Indicates whether English Query prompts
	end users for clarification of spelling errors
	or corrects spelling errors without
	prompting.
DBMSType Property (Session	Specifies the type of the primary database
Object)	used in this session.
DBMSVersion Property	Specifies the version of the database used in
	this session.
DomainInfo Property	Contains the model information (the
	<b>DomainInfo</b> object) for the <b>Session</b> object.
FallbackDBMSType Property	Specifies the type of database to use if a
	query cannot be generated for the primary
	database.
<u>FallbackDBMSVersion</u>	Specifies the version of the database to use
<u>Property</u>	if a query cannot be generated for the
	primary database.
RetainContext Property	Retains user context between questions for
	the <b>Session</b> object.

## Methods

ClearContext Method	Clears the conversation context.
InitDomain Method	Creates (if necessary) the <b>Domain</b> object
	and attaches it to the <b>Session</b> object.
	Parses an English question or request into a collection of database queries or other

	response types for the attached model.
SetDBMS Method	Sets the primary database type and version
	for which the query is generated.
SetFallbackDBMS Method	Sets the alternate database type and version
	for which a query is generated if one can be
	generated for the primary database.

#### Remarks

The **Session** object encapsulates the state of a user session with the English Query server. It is initialized for a particular model and supports parsing English Query requests for that model into a collection of well-defined commands that the client can then execute. English requests are interpreted within the context of the **Session** object.

Typically, the English Query server parses an English question into a command to execute a database query. The client then executes the query statement (by dispatching the database query to a <u>database object</u>) and displays the resulting data to the end user.

Before the **Session** object can parse a user's English questions, it must be initialized for a particular model by calling the **InitDomain** method.

The user's question can then be processed by first parsing it with the **ParseRequest** method and then taking the appropriate action, depending on the type of response returned from **ParseRequest**.

Before the **Session** object is released, the client should call **EndCommands** to retrieve a collection of commands that cleanly terminates the current user conversation with the application. These commands are necessary to remove any temporary tables that are created during the session as a result of executing SQL statements.

The English Query server supports multiple, concurrent **Session** objects, each maintaining the unique state of a particular user conversation context for clarifications, preferences, and the model. The lifetime of a **Session** object, as with all <u>COM</u> objects, is controlled by the client.

## See Also

DomainInfo Object

**EndCommands Collection** 

# **StaticInput Object**

Specifies static text to display to the user during clarification.

## **Properties**

Caption Property	Specifies the recommended caption for one of
	three user input types: <b>ListInput</b> , <b>StaticInput</b> ,
	and <b>TextInput</b> .

### **See Also**

ListInput Object

**TextInput Object** 

Type Property (UserInput Object)

<u>UserInput Object</u>

## **TextInput Object**

Specifies the text that end users have entered to clarify their questions.

## **Properties**

Caption Property	Specifies the recommended caption for one of
	three user input types: ListInput, StaticInput,
	and <b>TextInput</b> .
Text Property	Specifies the text end users provide to clarify their
	questions, statements, or errors.

#### **Remarks**

End users may have to clarify their question to get an answer. For example, they may ask, "Which employees are old?" They are asked to clarify with a **UserClarifyResponse** object, "An employee is old if his age is over what?" Then they provide the appropriate text.

#### See Also

**ListInput Object** 

**StaticInput Object** 

Type Property (UserInput Object)

**UserInput Object** 

## **UserClarifyResponse Object**

A clarification response object that is returned from **Session.ParseRequest** when English Query needs end-user input before it can complete parsing a request.

#### **Collections**

<u>UserInputs Collection</u>	A collection of <b>UserInput</b> object.
------------------------------	--

#### **Methods**

Reply Method	Continues parsing the current English request
	into a <b>Response</b> object after the end user has
	responded to a request for clarification.

#### **Remarks**

The values that represent the type of response objects are described in the **Type** property.

The **UserClarifyResponse** object has all the properties and methods of the **Response** object.

#### **See Also**

ParseRequest Method

Response Object

**Session Object** 

Type Property (Response Object)

<u>UserInputs Collection</u>

# **UserInput Object**

One of three user-clarification input objects: a list input, static text, or text input object.

## **Properties**

Type Property	Contains an integer or a symbolic constant that
	represents a type of user input object: a
	ListInput, StaticInput, or TextInput object.

#### **Remarks**

This object is read-only.

#### See Also

**ListInput Object** 

StaticInput Object

**TextInput Object** 

<u>UserClarifyResponse Object</u>

<u>UserInputs Collection</u>

# **Engine Properties Summary**

The table shows the properties of the Engine object model and the objects to which they apply.

Property	Applies to
Answer Property	AnswerCmd Object
Caption Property	ListInput Object
	StaticInput Object
	TextInput Object
ClarifySpellingErrors Property	Session Object
Closed Property	DomainInfo Object
CmdID Property	Command Object
	DomainCmd Object
CommandName Property	DomainCmd Object
DBMSType Property	QueryCmd Object
DBMSType Property	Session Object
DBMSVersion Property	Session Object
Description Property	ErrorResponse Object
DisplayRows Property	QueryCmd Object
DisplayToUser Property	QueryCmd Object
DomainInfo Property	Session Object
FallbackDBMSType Property	FallbackDBMSType Property
FallbackDBMSVersion Property	FallbackDBMSVersion Property
FieldCount Property	Parameter Object
<u>Fields Property</u>	Parameter Object
FullName Property	DomainInfo Object
HelpText Property	HelpCmd Object
HelpType Property	HelpCmd Object
IgnoreError Property	QueryCmd Object

ItemCount Property	Listlnput Object
Items Property	Listlnput Object
ObjectID Property	HelpCmd Object
ObjectID Property	ObjectReference Object
ObjectType Property	HelpCmd Object
ObjectType Property	ObjectReference Object
ParameterID Property	Parameter Object
ParameterType Property	Parameter Object
QueryCommand Property	Parameter Object
QueryText Property	QueryCmd Object
QuestionBuilder Property	DomainInfo Object
Restatement Property	CommandResponse Object
RetainContext Property	Session Object
Selection Property	ListInput Object
SQL Property	QueryCmd Object
TableCaption Property	QueryCmd Object
<u>Text Property</u>	TextInput Object
TrueFalseAnswer Property	QueryCmd Object
Type Property	CommandResponse Object
	ErrorResponse Object
	Response Object
	<u>UserClarifyResponse Object</u>
Type Property	<u>UserInput Object</u>
<u>Units Property</u>	Parameter Object
Value Property	Parameter Object

## **Answer Property**

Provides the English answer text to a question that can be answered without executing a query.

## **Applies To**

AnswerCmd Object

### **Syntax**

string = AnswerCmd.Answer

#### **Parts**

*AnswerCmd* 

An <u>object variable</u> that instructs the client to execute the database query. *string* 

A returned <u>string</u> that contains the answer in English.

#### **Remarks**

This property is read-only.

## **Caption Property**

Specifies the recommended caption for one of three user input types: **ListInput**, **StaticInput**, and **TextInput**.

## **Applies To**

**ListInput Object** 

**StaticInput Object** 

**TextInput Object** 

### **Syntax**

string = UserInput.Caption

#### **Parts**

string

A returned <u>string</u> that contains the caption for the user's input.

User Input

An <u>object variable</u> that represents one of the user input types.

User input type	Description	
ListInput	The recommended caption of the list of options from	
	which the end user chooses.	
TextInput	A caption for the text that the end user provides.	
StaticInput	Any extra captions needed when clarifying an end user's	
	question or request. For example, if an end user asks,	
	"Which movies star Tanya van Dam", a caption such as,	
	"Is <i>van</i> part of the first name or the last name?" could be	
	specified.	

## Remarks

This property is read-only.

## See Also

Reply Method

## **ClarifySpellingErrors Property**

Indicates whether English Query prompts end users for clarification of spelling errors or corrects spelling errors without prompting.

## **Applies To**

**Session Object** 

### **Syntax**

Session.ClarifySpellingErrors [= value]

#### **Parts**

Session

An <u>object variable</u> that represents a **Session** object.

value

A **Boolean** value that can set or return values.

Value	Description	
TRUE	Request clarification from the user.	
FALSE	Correct spelling errors without clarification from the	
	user.	

#### **Remarks**

This property has read-write permissions.

### See Also

**DomainInfo Property** 

InitDomain Method

ParseRequest Method
RetainContext Property

## **Closed Property**

Indicates the open and closed status of the **DomainInfo** object.

## **Applies To**

**DomainInfo Object** 

### **Syntax**

DomainInfo.Closed [= value]

#### **Parts**

**DomainInfo** 

An <u>object variable</u> that represents the **DomainInfo** object.

value

A returned **Boolean** value.

Value	Description
TRUE	Closed
FALSE	Open

#### **Remarks**

This property is read-only. You can use this property to confirm that the model is actually closed after calling the **Close** method.

### See Also

**Close Method** 

## **CmdID Property**

Contains an integer or symbolic constant representing a command, such as a query, an answer, a user request for help, or a user request to exit.

## **Applies To**

**Command Object** 

**DomainCmd Object** 

### **Syntax**

Command.CmdID [= value]

#### **Parts**

Command

An <u>object variable</u> that represents one of the commands types.

value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Editing (VBScript) or symbolic constant for Visual Basic that represents one of the commands.

<b>Command type</b>	Value	Description	
NlCmdQuery	1	Query command.	
nlCmdAnswer	2	Answer command.	
NlCmdHelp	3	User requested Help.	
NlCmdExit	4	User requested to exit (for example, by typing	
		quit or bye).	
nlCmdDomain	5	Domain command.	

### Remarks

This property is read-only.

## See Also

AnswerCmd Object

QueryCmd Object

## **CommandName Property**

Contains a string that represents a <u>command relationship</u>.

## **Applies To**

**DomainCmd Object** 

#### **Syntax**

DomainCmd.CommandName [= value]

#### **Parts**

**DomainCmd** 

An <u>object variable</u> that instructs the client to execute a command relationship.

value

A returned <u>string</u> that represents a command relationship.

#### **Remarks**

The developer of an English Query application specifies this property when defining the command. This property is read-only.

## **DBMSType Property (QueryCmd Object)**

Specifies the type of database used for the database query.

### **Applies To**

**QueryCmd Object** 

## **Syntax**

QueryCmd.DBMSType [= value]

#### **Parts**

**QueryCmd** 

An <u>object variable</u> that represents a **QueryCmd** object.

value

An <u>integer</u> that represents type of database used in the query.

Value	Database type	Description
0	nlDBMSSQLServer Microsoft® SQL Server™ version 6.5 or	
		later
1	NIDBMSMSOlap	SQL Server 2000 Analysis Services
2	NIDBMSOracle	Oracle version 7.0 or later

#### **Remarks**

This property is read-only.

#### See Also

**DBMSVersion Property** 

## **DBMSType Property (Session Object)**

Specifies the type of the primary database used in this session.

## **Applies To**

**Session Object** 

### **Syntax**

Session.**DBMSType** [= value]

#### **Parts**

Session

An <u>object variable</u> that represents a **Session** object.

value

An <u>integer</u> that represents type of database used in the session.

Value	Database type	Description
0	<b>nlDBMSSQLServer</b> Microsoft® SQL Server™ version 6.5 or	
		later
1	nlDBMSMSOlap	SQL Server 2000 Analysis Services
2	nlDBMSOracle	Oracle version 7.0 or later

#### **Remarks**

This property is read-only.

### **See Also**

**DBMSVersion Property** 

## **DBMSVersion Property**

Specifies the version of the database used in this session.

## **Applies To**

**Session Object** 

### **Syntax**

string = Session.DBMSVersion

#### **Parts**

string

A returned <u>string</u> that specifies the version of database server used in this session.

Session

An <u>object variable</u> that represents a **Session** object.

#### **Remarks**

This property is read-only.

#### See Also

**ClarifySpellingErrors Property** 

**DomainInfo Object** 

InitDomain Method

ParseRequest Method

**QuestionBuilder Object** 

RetainContext Property

## SetDBMS Method

## **Description Property (ErrorResponse Object)**

Contains a description of the error to be displayed to the end user.

### **Applies To**

**ErrorResponse Object** 

### **Syntax**

string = ErrorResponse.Answer

#### **Parts**

string

A returned <u>string</u> that contains a description of the error for the end user.

#### **ErrorResponse**

An <u>object variable</u> that is returned when the question or request was not understood by English Query.

#### **Remarks**

This property is read-only.

## **DisplayRows Property**

The number of rows the client displays to answer the end user's question.

### **Applies To**

**QueryCmd Object** 

### **Syntax**

QueryCmd.DisplayRows [= value]

#### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute the database query.

value

A returned <u>integer</u> that indicates how many rows of information to return to the end user.

#### Remarks

This property is read-only. If the value of **DisplayRows** is nonzero, a user is asked a question specifying a specific number of rows to be displayed. For example, if a user asks, "Show me the top five earners last year", this property returns five rows of data, one row of data for each earner field. In Microsoft® SQL Server<sup>TM</sup> version 7.0 or SQL Server 2000, it is possible to use the SQL statement itself to limit the number of rows returned. When the SQL statement is used, the **DisplayRows** property is always zero. This property is needed for databases that do not support top n natively in SQL.

#### See Also

**DisplayToUser Property** 

IgnoreError Property

**QueryText Property** 

SetDBMS Method

**TableCaption Property** 

<u>TrueFalseAnswer Property</u>

# **DisplayToUser Property**

Specifies whether to display the results of the query to the user or to execute the database query without displaying the results.

## **Applies To**

**QueryCmd Object** 

### **Syntax**

QueryCmd.DisplayToUser [= value]

#### **Parts**

QueryCmd

An <u>object variable</u> that instructs the client to execute the database query.

value

A returned Boolean value.

Value	Description
TRUE	Results are displayed to the end user.
	Results are not displayed to the end user. (This is typical for queries that create temporary tables to be used in subsequent queries.)

#### **Remarks**

This property is read-only.

#### See Also

**DisplayRows Property** 

IgnoreError Property

**QueryText Property** 

TableCaption Property

<u>TrueFalseAnswer Property</u>

## **DomainInfo Property**

Contains the model information (the **DomainInfo** object) for the **Session** object.

### **Applies To**

**Session Object** 

### **Syntax**

DomainInfo = Session.**DomainInfo** 

#### **Parts**

**DomainInfo** 

An object variable that represents a **DomainInfo** object.

Session

An <u>object variable</u> that represents a **Session** object.

#### **Remarks**

This property is read-only. Before the **Session** object can parse an end user's English questions, it must be initialized for a particular model with the **InitDomain** method.

### **See Also**

<u>ClarifySpellingErrors Property</u>

**DomainInfo Object** 

InitDomain Method

ParseRequest Method

RetainContext Property

## FallbackDBMSType Property

Specifies the type of database to use if a query cannot be generated for the primary database.

## **Applies To**

**Session Object** 

### **Syntax**

Session.FallbackDBMSType [= value]

#### **Parts**

Session

An <u>object variable</u> that represents a **Session** object.

value

An <u>integer</u> that represents type of fallback database.

Value	Database type	Description
0	nlDBMSSQLServer	Microsoft® SQL Server <sup>TM</sup> version
		6.5 or later
1	nlDBMSMSOlap	SQL Server 2000 Analysis Services
2	nlDBMSOracle	Oracle version 7.0 or later

### Remarks

This property is read-only.

### See Also

FallbackDBMSVersion Property

## SetFallbackDBMS Method

## **FallbackDBMSVersion Property**

Specifies the version of the database to use if a query cannot be generated for the primary database.

### **Applies To**

**Session Object** 

### **Syntax**

string = Session.FallbackDBMSVersion

#### **Parts**

string

A string that specifies the version of the fallback database.

Session

An <u>object variable</u> that represents a **Session** object.

#### **Remarks**

This property is read-only.

#### See Also

FallbackDBMSType Property

SetFallbackDBMS Method

# FieldCount Property

Contains the number of fields in the **Fields** property array.

# **Applies To**

Parameter Object

# **Syntax**

Parameter.FieldCount [= value]

### **Parts**

Parameter

An <u>object variable</u> that describes a parameter for a <u>command relationship</u>.

value

A returned <u>integer</u> that represents the number of fields in the **Fields** property array.

### **Remarks**

This property is read-only.

## See Also

**DomainCmd Object** 

**Fields Property** 

# Fields Property (Parameter Object)

Contains an array of the fields in the specified query command, which yields the values of a parameter of a <u>command relationship</u>.

# **Applies To**

Parameter Object

**QueryCmd Object** 

### **Syntax**

Parameter.Fields(Index) [= value]

#### **Parts**

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship.

(Index)

The zero-based index of a specific item in the **Fields** array.

value

A returned <u>integer</u> that represents the field in the query command.

#### Remarks

This property is read-only. A single parameter can contain more than one field. For example, a jukebox application might contain the command relationship "Play tracks". If the key of the **Tracks** table in the database consists of both **TrackID** and **AlbumID**, the track parameter for the command relationship will contain both fields.

### See Also

**DomainCmd Object** 

FieldCount Property

QueryCommands Collection

# **FullName Property**

Indicates the path of the **DomainInfo** object.

# **Applies To**

**DomainInfo Object** 

# **Syntax**

string = DomainInfo.FullName

## **Parts**

string

A <u>string</u> containing the path of the model file (\*.eqd).

**DomainInfo** 

An <u>object variable</u> that represents the **DomainInfo** object.

# **Remarks**

This property is read-only.

# See Also

**Closed Property** 

# **HelpText Property (HelpCmd Object)**

Specifies Help text of the application for an entity.

# **Applies To**

HelpCmd Object

## **Syntax**

string = HelpCmd.HelpText

### **Parts**

string

A returned string that contains the Help text of the application for an entity.

HelpCmd

An <u>object variable</u> that describes the Help for a global, entity, or relationship object.

### **Remarks**

This property is read-only. If no Help is specified, this string is empty.

### See Also

**HelpType Property** 

ObjectID Property (HelpCmd Object)

ObjectType Property (HelpCmd Object)

**Reply Method** 

# **HelpType Property**

Contains an integer that represents the type of object for which Help is requested.

# **Applies To**

HelpCmd Object

# **Syntax**

HelpCmd.HelpType [= value]

### **Parts**

**HelpCmd** 

An <u>object variable</u> that describes the help for a global, entity, or relationship object.

value

A reserved  $\underline{\text{integer}}$ . The value is always 0.

### **Remarks**

This property is read-only.

## See Also

HelpText Property (HelpCmd Object)

ObjectID Property (HelpCmd Object)

ObjectType Property (HelpCmd Object)

# **IgnoreError Property**

Specifies whether the client should ignore database errors resulting from executing the command.

# **Applies To**

**QueryCmd Object** 

# **Syntax**

QueryCmd.IgnoreError [= value]

### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute the database query. *value* 

A returned Boolean value.

Value	Description
TRUE	Ignore any database error resulting from executing the command (for example, when deleting a temporary table).
FALSE	Do not ignore database errors.

## **Remarks**

This property is read-only.

## See Also

**DisplayRows Property** 

DisplayToUser Property

**QueryText Property** 

TableCaption Property

<u>TrueFalseAnswer Property</u>

# **ItemCount Property**

Specifies the number of items in the list from which the end user chooses.

# **Applies To**

**ListInput Object** 

# **Syntax**

*ListInput*.**ItemCount** [= *value*]

### **Parts**

ListInput

An <u>object variable</u> that represents one of the user input types.

value

A returned <u>integer</u> that indicates the number of items in the list.

### **Remarks**

This property is read-only.

## See Also

**Caption Property** 

**Items Property** 

**Selection Property** 

# **Items Property**

Specifies the array of strings in the list from which the user chooses.

# **Applies To**

**ListInput Object** 

# **Syntax**

array = ListInput.Items

### **Parts**

ListInput

An <u>object variable</u> that represents one of the user input types.

array

A returned array of string from which the user chooses.

## **Remarks**

This property is read-only.

## See Also

**Caption Property** 

**ItemCount Property** 

**Selection Property** 

# **ObjectID Property (HelpCmd Object)**

Contains the ID of the object for which Help is being requested.

# **Applies To**

HelpCmd Object

## **Syntax**

string = HelpCmd.ObjectID

### **Parts**

string

A returned <u>string</u> that contains the ID of the object for which Help is being requested.

HelpCmd

An <u>object variable</u> that describes the Help for a global, entity, or relationship object.

## Remarks

This property is read-only. **ObjectID** can be used in the Question Builder object model to look up an entity (or relationship) to obtain additional Help.

### See Also

**Command Object** 

HelpText Property (HelpCmd Object)

**HelpType Property** 

ObjectType Property (HelpCmd Object)

# Question Builder Object Model

# **ObjectID Property (ObjectReference Object)**

Contains the ID of the object in the model to which there is a reference.

# **Applies To**

ObjectReference Object

# **Syntax**

string = ObjectReference.ObjectID

#### **Parts**

string

A returned <u>string</u> that contains the ID of the object that is being referenced.

*ObjectReference* 

An <u>object variable</u> that represents the object being referenced.

### **Remarks**

This property is read-only.

### See Also

ObjectType Property (ObjectReference Object)

# ObjectType Property (HelpCmd Object)

Contains an integer representing the type of object for which Help is requested.

# **Applies To**

HelpCmd Object

# **Syntax**

HelpCmd.ObjectType [= value]

### **Parts**

**HelpCmd** 

An <u>object variable</u> that describes the Help for a global, entity, or relationship object.

value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Edition (VBScript) or symbolic constant for Visual Basic that represents the object type.

Value	Object type
0	nlGlobal
1	nlEntity
2	nlRelationship

### **Remarks**

This property is read-only.

### See Also

HelpText Property (HelpCmd Object)

HelpType Property
ObjectID Property (HelpCmd Object)

# **ObjectType Property (ObjectReference Object)**

Contains an integer representing the type of object that is being referenced.

# **Applies To**

ObjectReference Object

## **Syntax**

ObjectReference.**ObjectType** [= value]

### **Parts**

ObjectReference

An <u>object variable</u> that represents the object being referenced.

value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Edition (VBScript) or symbolic constant for Visual Basic that represents the object type.

Value	Object type
0	NlGlobal
1	NlEntity
2	NlRelationship

## **Remarks**

This property is read-only.

# See Also

ObjectID Property (ObjectReference Object)

# **ParameterID Property**

Contains an integer identifying a parameter of a <u>command relationship</u>.

# **Applies To**

Parameter Object

## **Syntax**

Parameter.ParameterID [= value]

### **Parts**

value

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship.

A returned <u>integer</u> that identifies a command relationship parameter.

### **Remarks**

This property is read-only. Parameter IDs are assigned to entities in relationships by the English Query application developer when defining the command relationship.

## See Also

**DomainCmd Object** 

# **ParameterType Property**

Contains an integer representing a parameter for a <u>command relationship</u>.

## **Applies To**

Parameter Object

# **Syntax**

Parameter.ParameterType [= value]

#### **Parts**

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship.

value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Edition (VBScript) or symbolic constant for Visual Basic that represents the parameter type.

Value	Parameter type	
0	nlParamValue	
1	nlParamField	

# Remarks

This property is read-only. When the property has the value **nlParamValue**, the parameter is a value parameter. A value parameter has the properties **Value** and **Units**. When the property has the value **nlParamField**, the parameter is a field parameter. A field parameter has the properties **QueryCommand** and **Fields**.

### See Also

**DomainCmd Object** 

Fields Property

**QueryCommand Property** 

**Units Property** 

Value Property

# **QueryCommand Property**

Contains an index of the query command in the **QueryCommands** collection that yields the values of a parameter of a <u>command relationship</u>.

# **Applies To**

Parameter Object

# **Syntax**

Parameter.QueryCommand [= value]

### **Parts**

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship. *value* 

A returned <u>integer</u> that represents the index of the query command that contains the values for this parameter.

### Remarks

This property is read-only.

### See Also

**DomainCmd Object** 

# **QueryText Property**

Returns the query text to execute.

# **Applies To**

**QueryCmd Object** 

## **Syntax**

QueryCmd.QueryText [= value]

### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute the database query.

value

A <u>string</u> that contains the returned query text to execute.

### **Remarks**

This property is read-only. Use in the same way that the **SQL** property is used in English Query version 7.0 and earlier.

## See Also

**DisplayRows Property** 

**DisplayToUser Property** 

<u>IgnoreError Property</u>

**TableCaption Property** 

**TrueFalseAnswer Property** 

# **QuestionBuilder Property**

Contains the Question Builder information (the **QuestionBuilder** object) for the **DomainInfo** object.

# **Applies To**

**DomainInfo Object** 

# **Syntax**

**Set** *QuestionBuilder* = *DomainInfo***.QuestionBuilder** 

#### **Parts**

QuestionBuilder

An object variable that represents a **QuestionBuilder** object.

Domain Info

An object variable that represents a **DomainInfo** object.

### **Remarks**

This property is read-only.

Before the **DomainInfo** object can parse an end user's English questions, it must be initialized with a specific model using the **InitDomain** method.

# See Also

**ClarifySpellingErrors Property** 

**DomainInfo Object** 

InitDomain Method

QuestionBuilder Object

# RetainContext Property

# **Restatement Property**

An English interpretation of the end user's question or request.

# **Applies To**

**CommandResponse Object** 

# **Syntax**

string = CommandResponse.Restatement

### **Parts**

string

A returned <u>string</u> that contains the English Query restatement of the end user's question or request.

Command Response

An <u>object variable</u> that represents a command response.

### **Remarks**

This property is read-only. Restatements confirm to end users that English Query properly interpreted their questions or requests.

# **See Also**

**Type Property** 

# **RetainContext Property**

Retains user context between questions for the **Session** object.

## **Applies To**

**Session Object** 

## **Syntax**

Session.RetainContext [= value]

#### **Parts**

Session

An object variable that represents a **Session** object.

value

A Boolean value that can set or return a value.

Value	Description	
TRUE	User context is saved between questions. This is the default.	
FALSE	User context is not saved between questions.	

### Remarks

This property is read-write. If the user context is saved, declarative statements that a user makes are retained (for example *an active person is one who exercises five days a week*). The most recent questions are also retained so that users can refer to them while asking additional question. For example, a user could ask "Show sales for 1997", and then follow it with questions such as "Include the products", and "Sort by month". The results would all be for the sales in 1997 without the user repeating the original question. Users can also ask follow-up questions with pronoun references and ellipses (for example, "Who is the oldest

salesperson" followed by "How much did he make" "Show sales in the Los Angeles branch", or "New York").

# See Also

ClarifySpellingErrors Property

**DomainInfo Object** 

InitDomain Method

ParseRequest Method

# **Selection Property**

Specifies the end user's selection from the list.

# **Applies To**

**ListInput Object** 

# **Syntax**

*ListInput*.**Selection** [= *value*]

### **Parts**

ListInput

An <u>object variable</u> that represents one of the user input types.

value

An <u>integer</u>, which can be set or returned, that represents the end user's selection from the list.

### **Remarks**

This property has read-write permissions and is zero-based.

# See Also

**Caption Property** 

**Items Property** 

**ItemCount Property** 

# **SQL Property**

Returns the query text to execute.

# **Applies To**

**QueryCmd Object** 

## **Syntax**

QueryCmd.SQL [= value]

### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute the database query. *value* 

A <u>string</u> that contains the returned query text to execute.

### **Remarks**

This property is read-only. This property is included for compatibility with earlier versions of English Query. Use the **QueryText** property for English Query.

# See Also

**DisplayRows Property** 

**DisplayToUser Property** 

**IgnoreError Property** 

**TableCaption Property** 

**TrueFalseAnswer Property** 

# **TableCaption Property**

Specifies the suggested English caption for the result table.

# **Applies To**

**QueryCmd Object** 

## **Syntax**

string = QueryCmd.TableCaption

### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute the database query. *string* 

A string that returns the caption for the result table.

### **Remarks**

This property is read-only.

### See Also

**DisplayRows Property** 

**DisplayToUser Property** 

**IgnoreError Property** 

**QueryText Property** 

**TrueFalseAnswer Property** 

# **Text Property (TextInput Object)**

Specifies the text end users provide to clarify their questions, statements, or errors.

# **Applies To**

**TextInput Object** 

# **Syntax**

*TextInput*.**Text** = *string* 

### **Parts**

*TextInput* 

An <u>object variable</u> that represents one of the user input types.

string

A <u>string</u> , which can be set or returned, that represents the text end users provide to clarify their questions or statements.

### Remarks

This property has read-write permissions.

## See Also

**Caption Property** 

# **TrueFalseAnswer Property**

Specifies whether the client should show TRUE or FALSE as an answer instead of a table.

# **Applies To**

**QueryCmd Object** 

# **Syntax**

QueryCmd.TrueFalseAnswer [= value]

#### **Parts**

**QueryCmd** 

An <u>object variable</u> that instructs the client to execute database query.

value

A returned Boolean value.

Value	Description
TRUE	When the answer to a question is expressed as Yes or No rather than a result table.
FALSE	When the answer to the question can be displayed as a result table.

## **Remarks**

This property is read-only. It indicates that the user asked a Yes/No question. If the answer is No, the database query returns no result table. If the answer is Yes, the database query returns a result table.

### See Also

**DisplayRows Property** 

DisplayToUser Property

IgnoreError Property

**QueryText Property** 

**TableCaption Property** 

# **Type Property (Response Object)**

Contains an integer or a symbolic constant representing a type of **Response** object.

# **Applies To**

**Response Object** 

# **Syntax**

*Response*.**Type** [= value]

#### **Parts**

Response

An <u>object variable</u> that represents one of the response types.

value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Edition (VBScript) or a symbolic constant for that represents one of the response types.

Value	Response type	Description
0	nlResponseCommand	Command response
3	nlResponseError	Error response
4	nlResponseUserClarify	User-clarification response

#### **Remarks**

This property is read-only.

### See Also

CommandResponse Object

ErrorResponse Object
UserClarifyResponse Object

# **Type Property (UserInput Object)**

Contains an integer or symbolic constants that represent a type of user input object: a **ListInput**, **StaticInput**, or **TextInput** object.

# **Applies To**

**UserInput Object** 

# **Syntax**

UserInput.Type [= value]

#### **Parts**

**UserInput** 

An <u>object variable</u> that represents one of the user input types.

Constant	Value	User input types
nlInputList	0	ListInput
nlInputText	1	TextInput
nlInputStatic	2	StaticInput

#### value

A returned <u>integer</u> for Microsoft® Visual Basic® Scripting Edition (VBScript) or a symbolic constant for Visual Basic that represents one of the user input types.

### See Also

**ListInput Object** 

**StaticInput Object** 

**TextInput Object** 

# **UserInputs Collection**

## **Units Property**

Contains a description that qualifies the parameter value, such as *percent* or *dollars*.

### **Applies To**

Parameter Object

#### **Syntax**

Parameter.Units [= value]

#### **Parts**

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship. *value* 

A returned <u>string</u> that represents the description of the parameter value.

#### **Remarks**

This property is read-only.

#### See Also

**DomainCmd Object** 

ParameterType Property

# Value Property

Contains a parameter value that represents a parameter for a <u>command relationship</u>.

### **Applies To**

Parameter Object

#### **Syntax**

Parameter.Value[= value]

#### **Parts**

**Parameter** 

An <u>object variable</u> that describes a parameter for a command relationship. *value* 

A returned variant that represents the parameter value.

#### **Remarks**

This property is read-only.

#### See Also

**DomainCmd Object** 

# **Engine Methods Summary**

The table shows the methods in the Engine object model and the objects to which they apply.

Methods	Applies to
ClearContext Method	Session Object
Close Method	DomainInfo Object
InitDomain Method	Session Object
ParseRequest Method	Session Object
Reply Method	UserClarifyResponse Object
SetDBMS Method	Session Object
SetFallbackDBMS Method	Session Object

### **ClearContext Method**

Clears the conversation context.

### **Applies To**

**Session Object** 

#### **Syntax**

Session.ClearContext

#### **Parts**

Session

An object variable that represents an existing model object.

#### **Remarks**

This method resets the user's session to its initial state, identical to the state immediately after the **InitDomain** method is called. This clears all declarative statements and previously saved questions (which could be used for pronoun or elliptical reference).

#### See Also

InitDomain Method

ParseRequest Method

**RetainContext Property** 

### **Close Method**

Closes the **DomainInfo** object.

### **Applies To**

**DomainInfo Object** 

#### **Syntax**

DomainInfo.Close

#### **Parts**

DomainInfo

An <u>object variable</u> that represents the **DomainInfo** object.

#### Remarks

The **DomainInfo** is removed when all attached sessions terminate.

#### See Also

**Closed Property** 

### **InitDomain Method**

Creates a **Domain** object if necessary and attaches it to the **Session** object.

#### **Applies To**

Session Object

#### **Syntax**

Session.InitDomain(DomainFileName)

#### **Parts**

Session

An object variable that represents an existing session object.

**DomainFileName** 

A <u>string</u> that contains the specific model file name (\*.eqd).

#### **Remarks**

This method must be called before any other method of the **Session** object.

#### See Also

**ClarifySpellingErrors Property** 

**DomainInfo Property** 

ParseRequest Method

### **ParseRequest Method**

Parses an English question or request into a collection of database queries or other response types for the attached model.

### **Applies To**

**Session Object** 

#### **Syntax**

**Set** Response = Session.**ParseRequest(**Request**)** 

#### **Parts**

Response

A **Response** object that represents the end user's responses.

Response object	Description
CommandResponse	An ordered sequence of database queries
	that is executed to answer the end user's
	question.
UserClarifyResponse	A clarification requiring the end user's
	input to complete processing the request.
ErrorResponse	A parsing error. This object is returned if
	the user's query is not understood in the
	context of the current model.

#### Session

An <u>object variable</u> that represents an existing **Session** object.

#### Request

A <u>string</u> that is an English question, request, or command.

#### **Remarks**

The English Query server parses a plain English question or request into a set of commands, which execute database queries and other types of responses. The client then executes the database query and displays the resulting data to the end user.

#### See Also

**ClarifySpellingErrors Property** 

**DomainInfo Property** 

InitDomain Method

### **Reply Method**

Continues parsing the current English request into a **Response** object after the end user has responded to a request for clarification.

#### **Applies To**

**UserClarifyResponse Object** 

#### **Syntax**

**Set** Response = UserClarifyResponse.**Reply** 

#### **Parts**

Response

An <u>object variable</u> that represents the response to the user.

*UserClarifyResponse* 

An object variable that contains a **UserClarifyResponse** object.

#### Remarks

There may be many clarifications needed to answer the question. After a user has responded and a client application has placed the user's responses in the **Selection** properties of the **ListInput** objects and in the **Text** properties on the **TextInput** objects, the client application calls **Reply**. **Reply** reprocesses the initial question (originally sent using **ProcessRequest**), along with all provided clarifications, and returns a new **Response** object.

There may be additional clarifications needed to answer the question, so this **Response** object might be another **UserClarifyResponse**.

#### **See Also**

**Response Object** 

### <u>UserInput Object</u>

### **SetDBMS Method**

Sets the primary database type and version for which the query is generated.

#### **Applies To**

**Session Object** 

#### **Syntax**

Session.SetDBMS(Type, Version)

#### **Parts**

Session

An object variable that represents an existing domain object.

Туре

An <u>integer</u> that represents the database type.

Value	Database type	Description
0	nlDBMSSQLServer	Microsoft® SQL Server™ 2000
1	nlDBMSMSOlap	SQL Server 2000 Analysis Services
2	nlDBMSOracle	Oracle

#### Version

A <u>string</u> in the form of *nn*, *mm*, *xx*, *yy* that represents the version of the database.

#### **Remarks**

For example, to set a model created in SQL Server 2000 to SQL Server version 6.5, use:

Session.SetDBMS(nlDBMSSQLServer, "6.5")

### See Also

**DBMSVersion Property** 

InitDomain Method

ParseRequest Method

#### SetFallBackDBMS Method

Sets the alternate database type and version for which a query is generated if none can be generated for the primary database.

#### **Applies To**

**Session Object** 

#### **Syntax**

Session.SetFallBackDBMS(Type, [Version])

#### **Parts**

Session

An object variable that represents an existing domain object.

Туре

An <u>integer</u> that represents the database type.

The values for *Type* includes the following:

Value	Database type	Description
0	nlDBMSSQLServer	Microsoft® SQL Server™ 2000
1	nlDBMSMSOlap	SQL Server 2000 Analysis Services
2	nlDBMSOracle	Oracle

#### Version

A <u>string</u> in the form of *nn*, *mm*, *xx*, *yy* that represents the version of the database. Optional.

#### **Remarks**

FallbackDBMSType and FallBackDBMSVersion properties are used to

generate a query if English Query cannot generate a query using the primary database. This occurs only if the primary database is **nlDBMSMSOlap**. The fallback database must be an SQL database such as **nlDBMSSQLServer**.

To set it so that there is no fallback database, use **SetFallbackDBMS** with no arguments. If there is no fallback database and English Query cannot generate a query using the primary database, English Query returns an error response.

#### **See Also**

FallbackDBMSType Property

FallbackDBMSVersion Property

### **Engine Collections Summary**

This summary lists all the collections in the Engine Object model and the properties and methods for each.

• Commands Collection

Properties/Methods

The number of commands in the collection. <u>Index Method</u>Returns a command from the collection by ordered position.

• EndCommands Collection

Properties/Methods

• Parameters Collection

Properties/Methods

• QueryCommands Collection

Properties/Methods

• <u>UserInputs Collection</u>

Properties/Methods

### **Commands Collection**

Contains commands for the client to execute.

### **Properties**

Count Property	Indicates the number of commands in the
	Commands, EndCommands, Parameters,
	QueryCommands, and UserInputs
	collections.

#### **Methods**

Index Method	Returns a command from the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs
	collection by ordered position.

#### **Remarks**

The index is zero-based.

#### **See Also**

**Command Object** 

### **EndCommands Collection**

A collection of commands that should be executed by the client to clean up when ending a session.

### **Properties**

Count Property	The number of commands in the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs
	collections.

#### **Methods**

Index Method	Returns a command from the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs
	collection by ordered position.

#### **Remarks**

This is a collection of the **Session** object. This collection is read-only.

#### See Also

**Session Object** 

### **Parameters Collection**

A collection of **Parameter** objects, which are parameters for a <u>command relationship</u>.

### **Properties**

Count Property	The number of commands in the <b>Commands</b> ,
	EndCommands, Parameters,
	<b>QueryCommands</b> , and <b>UserInputs</b> collections.

#### **Methods**

Index Method	Returns a command from the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs collection
	by ordered position.

#### Remarks

This collection is read-only. A particular instance of a command relationship may not specify all possible parameters. For example, the command relationship "Increase salaries by raise amounts" contains two parameters: salaries and raise amounts. If a user says, "Increase Sue's salary by \$1000", the resulting **DomainCmd** object contains both parameters. However, if a user says, "Increase Sue's salary", the **DomainCmd** has only one parameter, and "Increase the salaries" has no parameters.

#### **See Also**

**DomainCmd Object** 

Parameter Object

### **QueryCommands Collection**

A collection of **QueryCmd** objects, which contain the queries that must be executed by the client application to obtain the parameters for a command relationship.

### **Properties**

Count Property	The number of commands in the <b>Commands</b> ,
	EndCommands, Parameters,
	<b>QueryCommands</b> , and <b>UserInputs</b> collections.

#### **Methods**

Index Method	Returns a command from the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs collection
	by ordered position.

#### **Remarks**

This collection is read-only. When processing a command relationship, the client application must execute each of the queries in the **QueryCommands** collection. The results of these queries are used as parameters for the command relationship. Query commands marked as TRUE for the value of the **DisplayToUser** property are the queries containing the final results that are used as parameters. A single query command may be used for more than one parameter.

#### See Also

**DisplayToUser Property** 

### QueryCmd Object

# **UserInputs Collection**

A collection of **UserInput** objects.

### **Properties**

Count Property	The number of commands in the <b>Commands</b> ,	
	EndCommands, Parameters,	
	<b>QueryCommands</b> , and <b>UserInputs</b> collections.	

#### **Methods**

Index Method	Returns a command from the <b>Commands</b> ,
	EndCommands, Parameters,
	QueryCommands, and UserInputs collection
	by ordered position.

#### **Remarks**

This collection is read-only. **UserInputs** collection belongs to a **UserClarifyResponse** object.

#### See Also

Type Property (UserInput Object)

<u>UserClarifyResponse Object</u>

**UserInput Object** 

# **Engine Collections Properties Summary**

The table shows the properties in the Engine object model and collections to which they apply.

Property	Applies to
Count Property	Commands Collection
	EndCommands Collection
	Parameters Collection
	QueryCommands Collection
	<u>UserInputs Collection</u>

### **Count Property**

Indicates the number of commands in the **Commands**, **EndCommands**, **Parameters**, **QueryCommands**, and **UserInputs** collections.

### **Applies To**

**Commands Collection** 

**EndCommands Collection** 

**Parameters Collection** 

**QueryCommands Collection** 

**UserInputs Collection** 

#### **Syntax**

Collection.Count [= value]

#### **Parts**

Collection

A collection of **Commands**, **EndCommands**, **Parameters**, **QueryCommands** or **UserInputs**.

value

A returned <u>integer</u> that indicates how many commands there are in the collection.

#### Remarks

This property is read-only.

#### See Also

Index Method (Engine Object Model)

# **Engine Collections Methods Summary**

The table shows the methods in the Engine object model and the collections to which they apply.

Methods	Applies to
Index Method	Commands Collection
	EndCommands Collection
	Parameters Collection
	QueryCommands Collection
	<u>UserInputs Collection</u>

### **Index Method (Engine Object Model)**

Returns a command from the **Commands**, **EndCommands**, **Parameters**, **QueryCommands**, and **UserInputs** collection by ordered position.

#### **Applies To**

**Commands Collection** 

**EndCommands Collection** 

**Parameters Collection** 

**QueryCommands Collection** 

**UserInputs Collection** 

#### **Syntax**

**Set** *objItem* = *Collection***.Index**(*n*)

**Set** *objItem* = *Collection*(*n*)

#### **Parts**

objItem

An <u>object variable</u> that may represent a command object in the **Commands**, **EndCommands**, **Parameters**, **QueryCommands**, or **UserInputs** collections of commands; an entity object in the **QBEntity** collection of entities; a relationship object in the **QBRelationship** collection of relationships; or a template object in the **Templates** collection of question templates.

#### Collection

An object variable that may represent the **Commands** collection object, **EndCommands** collection object, **Parameters** collection object, **QueryCommands** collection object, **UserInputs** collection object,

**QBEntity** collection object; **QBRelationship** collection object; or **Templates** collection object.

**(n)** 

An <u>integer</u> that represents an item in a collection.

#### Remarks

The index is zero-based.

### See Also

**Count Property** 

### **Question Builder Object Model**

The Question Builder object model is a COM-automation interface used to programmatically retrieve information about an English Query model.

Question Builder allows the end user to dynamically browse model-specific knowledge and to view questions based on this knowledge.

Use the Question Builder in your application so an end user can find:

- Information in the database.
- Basic relationships represented in the database.
- English phrases that can be used to ask questions about basic relationships.

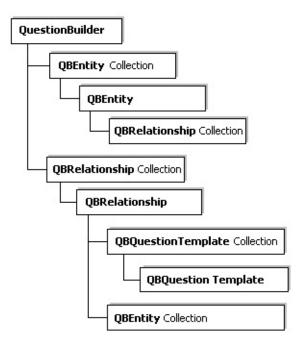
Question Builder helps end users form questions that are semantically and syntactically valid by allowing them to:

- Explore the contents in the model.
  - Question Builder provides a list of entities, their corresponding attributes, and the relationships in which they participate. For each entity, Question Builder displays:
    - The singular and plural forms.
    - Any corresponding synonyms.
    - Help information that was specified by the Model Editor.
- Generate both general questions and questions about a specific instance of an entity.

#### See Also

## Adding Question Builder to an English Query Application

# **Question Builder Object Model Diagram**



# **Question Builder Objects Summary**

This summary lists all the objects in the Question Builder object model and the properties and methods for each.

- QBEntity Object
   Properties/Methods
- QBRelationship Object
   Properties/Methods
- QBTemplate Object
   Properties/Methods
- QuestionBuilder Object
   Properties/Methods

# **QBEntity Object**

An <u>entity</u> in the application <u>model</u>.

# **Properties**

Description Property	Specifies a brief description of the entity or relationship.
HelpText Property	Specifies the Help information for the entity that is specified when the model is created.
<u>InstanceValues</u>	Specifies the known instance values for the entity.
<u>IsMajor Property</u>	Specifies whether the entity stands alone or is an attribute of another entity.
ObjectID Property	Specifies the object identifier for the entity or relationship.
ObjectType Property	Specifies whether the object is an entity or relationship.
Parent Property	Specifies the parent entity if the entity is an attribute of another entity.
Plural Property	Specifies the plural form of the main noun describing the entity.
Singular Property	Specifies the singular form of the main noun describing the entity.
SummaryText Property	Specifies a list of relationships in which the entity participates. The list is in the form of English sentences.
Synonyms Property	Specifies the array of strings that are synonyms for the entity.
Values Property	Sets the particular instance values that will be used during the generation of questions.

## Methods

<b>GetCommonRelationships</b>	Returns the relationships that involve two	
<u>Method</u>	entities: the <b>QBEntity</b> object and the entity	
	represented by the <b>ObjectID</b> property.	
FindTemplates Method	lates Method Returns a collection of templates whose subject	
	is the entity itself.	

# **QBRelationship Object**

A <u>relationship</u> in the application <u>model</u>.

# **Properties**

D ' (' D (	
Description Property	Specifies a brief description of the entity or
	relationship.
HelpText Property	Specifies the Help information for the entity that
	is specified when the model is created.
ObjectID (QBEntity	Specifies the object identifier for the relationship.
Object, QBRelationship	
Object) Property	
ObjectType Property	Specifies whether the object is an entity or
	relationship.

# **QBTemplate Object**

Represents a <u>question template</u> that can be used to help a user construct a question about concepts in the <u>model</u>.

## **Properties**

Description Property	Specifies a brief description of the question
	template.

#### **Methods**

GetExamples Method	Returns all possible examples of questions for a
	relationship in which the entity is the subject of
	the question.

# **QuestionBuilder Object**

The root object for **Question Builder**.

#### **Methods**

FindObject Method	Returns an entity or relationship based on object
	identification.

#### **Remarks**

The **QuestionBuilder** object associated with a particular model is available from the **QuestionBuilder** property of the **DomainInfo** object. This object has the following collections: **QBEntity** (all entities in the model) and **QBRelationship** (all relationships in the model).

# **Question Builder Properties Summary**

This table lists the objects and their properties in the Question Builder object model.

Property	Applies to
Description Property	QBEntity Object
	QBRelationship Object
Description Property	QBTemplate Object
HelpText Property	QBEntity Object
	QBRelationship Object
InstanceValues Property	QBEntity Object
IsMajor Property	QBEntity Object
ObjectID Property	QBEntity Object
	QBRelationship Object
ObjectType Property	QBEntity Object
	QBRelationship Object
Parent Property	QBEntity Object
Plural Property	QBEntity Object
Singular Property	QBEntity Object
SummaryText Property	QBEntity Object
Synonyms Property	QBEntity Object
Values Property	QBEntity Object

# Description Property (QBEntity Object, QBRelationship Object)

Specifies a brief description of the entity or relationship.

### **Applies To**

**QBEntity Object** 

**QBRelationship Object** 

#### **Syntax**

string = QBEntity.Description

string = QBRelationship.Description

#### **Parts**

string

A <u>string</u> that returns the description of the object.

QBEntity, QBRelationship

An object variable that represents an entity in the model.

#### **Remarks**

# **Description Property (QBTemplateObject)**

Specifies a brief description of the <u>question template</u>.

## **Applies To**

**QBTemplate Object** 

#### **Syntax**

*string* = *QBTemplate***.Description** 

#### **Parts**

string

A returned string that describes the template.

**QBTemplate** 

An <u>object variable</u> that represents a question template, which is used to help a user construct a question about concepts in the model.

#### **Remarks**

# HelpText Property (QBEntity Object, QBRelationship Object)

Specifies the Help information for the <u>entity</u> that is specified when the <u>model</u> is created.

#### **Applies To**

**QBEntity Object** 

**QBRelationship Object** 

#### **Syntax**

string = QBEntity.HelpText

#### **Parts**

string

A returned string of Help text for the entity.

QBEntity

An <u>object variable</u> that represents an entity in the model.

#### **Remarks**

# **InstanceValues Property**

Specifies the known instance values for the **QBEntity** object.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

*array* = *QBEntity*.**InstanceValues** 

#### **Parts**

array

A returned array of strings from which the user chooses an example.

**QBEntity** 

An <u>object variable</u> that represents an entity in the <u>model</u>.

# **IsMajor Property**

Specifies whether the entity stands alone or is an attribute of another entity.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

value = QBEntity.**IsMajor** 

#### **Parts**

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

value

A returned <u>Boolean</u> indicating whether an entity stands alone or is an attribute of another entity.

Value	Description
TRUE	Entity stands alone.
FALSE	Entity is an attribute of another entity.

#### **Remarks**

# ObjectID Property (QBEntity Object, QBRelationship Object)

Specifies the object identifier for the entity or relationship.

### **Applies To**

**QBEntity Object** 

**QBRelationship Object** 

#### **Syntax**

string = QBEntity.ObjectID

string = QBRelationship.ObjectID

#### **Parts**

string

A returned <u>string</u> that represents the entity or relationship.

QBEntity, QBRelationship

An <u>object variable</u> that represents an entity or relationship in the model.

#### **Remarks**

# ObjectType Property (QBEntity Object, QBRelationship Object)

Specifies the kind of object, either an entity or relationship.

#### **Applies To**

**QBEntity Object** 

**QBRelationship Object** 

#### **Syntax**

*value* = *QBEntity*.**ObjectType** 

*value* = *QBRelationship*.**ObjectType** 

#### **Parts**

QBEntity, QBRelationship

An <u>object variable</u> that represents an entity or relationship in the model.

value

An <u>integer</u>, which is returned, that represents one of the object types below.

Value	Object type	Constant
1	Entity	QB_ENTITY
2	Relationship	QB_RELATIONSHIP

#### **Remarks**

## **Parent Property**

Specifies the parent *entity* if the entity is an attribute of another entity.

#### **Applies To**

**QBEntity Object** 

#### **Syntax**

*value* = *QBEntity*.**Parent** 

#### **Parts**

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

value

An <u>integer</u> that represents the parent entity if the entity is an attribute of another entity. It returns 0 if the:

- Entity has no parent entity.
- Attribute does not have a corresponding parent entity.

#### **Remarks**

# **Plural Property**

Specifies the plural form of the main noun describing the entity.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

string = QBEntity.Plural

#### **Parts**

string

A string that returns the plural form of the entity.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

#### Remarks

# **Singular Property**

Specifies the singular form of the main noun describing the entity.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

string = QBEntity.Singular

#### **Parts**

string

A string that returns the singular form of the entity.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

#### Remarks

## **SummaryText Property**

Specifies a list, in the English sentence form, of relationships in which the entity participates.

#### **Applies To**

**QBEntity** 

#### **Syntax**

```
string = QBEntity.SummaryText
```

#### **Parts**

string

A returned string that describes a Question Builder object.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

#### **Remarks**

The following is an example of how the **SummaryText** property is called:

```
<%
   Set objEntities = objQBuilder.Entities

CRNL = Chr(13) & Chr(10)
   str2 = "&nbsp;&nbsp;&nbsp;&nbsp;
Response.Write( "fpp " & CRNL)

For each objEntity in objEntities</pre>
```

str1 = objEntity.SummaryText

```
str1 = Replace(str1, Chr(13), "<br>")
str1 = Replace(str1, Chr(9), str2)
Response.Write( str1 & CRNL)
Response.Write( "" & CRNL)
Next
%>
```

This is an example of what gets printed:

hotelname is an attribute of hotel.

hotelname participates in the following relationships:

hotels have hotelnames

hotels have price categories, airport distances, casinos, hotel names, checkouts, URLs, addresses, checkins, faxes, foreign phone numbers, latitudes, local phones, longitudes, pools, postal codes, reservation phones, and types.

*hotels* participate in the following relationships:

- hotels are within airport distances from airports
- hotels have distances from sights
- hotels have cities
- hotels feature amenities
- hotels have amenities
- hotels are located in cities
- hotels are located in countries
- hotels are located in states.

- hotels are located in sights
- hotels are located in addresses
- amenities are located in hotels
- casinos are located in hotels
- hotels offer amenities
- hotels have prices
- hotels accept credit cards
- hotels cost prices
- hotels allow amenities
- sights are near hotels
- hotels are near sights
- hotels honor credit cards
- hotels take credit cards
- hotels are within distances from sights

# **Synonyms Property**

Specifies the array of strings of synonyms for the entity.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

*array* = *QBEntity*.**Synonyms** 

#### **Parts**

array

A returned array of strings of synonyms for the entity.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

### Remarks

# **Value Property**

Sets the specific instance values that will be used in generating questions.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

*QBEntity*.**Value** = *string* 

#### **Parts**

string

A returned <u>string</u> that identifies the instance value of the entity that will be used in generating questions.

**QBEntity** 

An <u>object variable</u> that represents an entity in the <u>model</u>.

#### **Remarks**

This property is read/write.

# **Question Builder Methods Summary**

This table lists the objects and their associated methods in the Question Builder object model.

Method	Applies to
FindObject Method	QuestionBuilder Object
FindTemplates Method	QBEntity Object
<u>GetCommonRelationships</u>	QBEntity Object
<u>Method</u>	
GetExamples Method	QBTemplate Object

## **FindObject Method**

Returns an entity or relationship depending on object identification.

#### **Applies To**

**QuestionBuilder Object** 

#### **Syntax**

[variable] = QuestionBuilder.**FindObject** ObjectID

#### **Parts**

QuestionBuilder

An <u>object variable</u> that contains the root object for exploring model information.

ObjectID

A <u>string</u> that represents the entity or relationship for which you are looking. *value* 

The **QBEntity** object or the **QBRelationship** object that corresponds to *ObjectID*.

# **FindTemplates Method**

Returns a collection of templates whose subject is the entity itself.

## **Applies To**

**QBEntity Object** 

#### **Syntax**

**Set** *variable* = *QBEntity***.FindTemplates** 

#### **Parts**

variable

All the templates in which the entity participates.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model or the **ObjectID** property.

## **GetCommonRelationships Method**

Returns the relationships that involves two <u>entities</u>: the **QBEntity** object and the entity represented by the **ObjectID** property.

#### **Applies To**

**QBEntity Object** 

#### **Syntax**

array = QBEntity.GetCommonRelationships(Entity2)

#### **Parts**

array

A returned collection of <u>relationships</u>.

**QBEntity** 

An <u>object variable</u> that represents an entity in the model.

Entity2

The other entity that participates in one or more relationships with *QBEntity*.

## **GetExamples Method**

Returns all possible examples of questions for a relationship in which the <u>entity</u> is the subject of the question.

#### **Applies To**

**QBTemplate Object** 

array = QBTemplate.GetExamples(QBEntity, fdate, fneg)

#### **Parts**

array

A returned array of strings from which the user chooses an example.

**QBTemplate** 

An <u>object variable</u> that represents a question template in the <u>model</u>.

QBEntity, fdate, fneg

The subject of the question, including *fdate* and *fneg*.

#### Remarks

*QBEntity* is optional. However, if *QBEntity* is specified, *fdate* and *fneg* must both be set to **False**. The examples are generated from the templates for the relationship.

# **Question Builder Collections Summary**

This summary lists all the collections in the Question Builder object model and the properties and methods for each.

• Entities Collection

#### Properties/Methods

• RelatedEntities Collection

#### Properties/Methods

• RelatedRelationships Collection

#### Properties/Methods

• Relationships Collection

#### Properties/Methods

• Templates Collection

Properties/Methods

# **Entities Collection**

A collection of **entities**.

# **Applies To**

QuestionBuilder Object

## **Methods**

Index Method	Returns entities, relationships, and templates
	from Question Builder collections: <b>Entities</b> ,
	Relationships, RelatedEntities,
	RelatedRelationships, and Templates.

## **RelatedEntities Collection**

A collection of **entities** that participate in the relationship.

## **Applies To**

**QBRelationship Object** 

#### **Methods**

Index Method	Returns entities, relationships, and templates
	from Question Builder collections: Entities,
	Relationships, RelatedEntities,
	RelatedRelationships, and Templates.

#### **Remarks**

The **RelatedEntities** collection is a subset of the **Entities** collection.

# **Relationships Collection**

A collection of relationships.

# **Applies To**

QuestionBuilder Object

## **Methods**

Index Method	Returns entities, relationships, and templates
	from Question Builder collections: <b>Entities</b> ,
	Relationships, RelatedEntities,
	RelatedRelationships, and Templates.

# **RelatedRelationships Collection**

A collection of **ObjectID**s in relationships in which the <u>entity</u> participates.

## **Applies To**

**QBEntity Object** 

#### **Methods**

Index Method	Returns entities, relationships, and templates
	from Question Builder collections: Entities,
	Relationships, RelatedEntities,
	RelatedRelationships, and Templates.

#### **Remarks**

The **RelatedRelationships** collection is a subset of the **Relationships** collection.

# **Templates Collection**

A collection of question templates.

## **Applies To**

**QBRelationship Object** 

## Methods

Index Method	Returns entities, relationships, and templates
	from Question Builder collections: <b>Entities</b> ,
	Relationships, RelatedEntities,
	RelatedRelationships, and Templates.

# **Question Builder Collections Methods Summary**

This table lists the collections and their associated methods in the Question Builder object model.

Method	Applies to
Index Method	Entities Collection
	RelatedEntities Collection
	RelatedRelationships Collection
	Relationships Collection
	Templates Collection

## **Index Method (Question Builder Object Model)**

Returns entities, relationships, and templates from the Question Builder collections: **Entities**, **RelatedEntities**, **RelatedRelationships**, **Relationships**, and **Templates**.

## **Applies To**

**Entities Collection** 

**RelatedEntities Collection** 

RelatedRelationships Collection

**Relationships Collection** 

**Templates Collection** 

## **Syntax**

**Set** *objEntityVar* = **Entities(***n***)** 

**Set** *objRelationshipVar* = **Relationships**(*n*)

**Set** *objTemplateVar* = **Templates(***n***)** 

### **Parts**

objEntityVar

An entity object in the **QBEntity** collection.

#### **Entities**

The **QBEntity** collection object.

objRelationshipsVar

A relationship object in the **QBRelationship** collection of relationships.

## Relationships

The **QBRelationship** collection object.

objTemplateVar

A template object in the **Templates** collection of question templates.

## **Templates**

The **Templates** collection object.

**(**n**)** 

A returned command name or <u>integer</u> representing a command in a collection.

### **Remarks**

The index is zero-based.

## How To

These topics provide step-by-step instructions for performing tasks while creating English Query projects, models, and applications.

For more information, see <u>Developing and Deploying English Query Application</u>.

For information about building an English Query application step-by-step, see the English Query Tutorials. These scenario-based lessons are available from the English Query **Start** menu or from the Microsoft® Visual Studio® **Help** menu when an English Query project is opened.

## **Working with Projects**

English Query can create new <u>projects</u> by automatically extracting the schema or structure of a database using the SQL Project Wizard or the OLAP Project Wizard.

You can also begin with an empty project, add tables, fields, joins, and other data objects, and then create the entities and relationships needed for the model.

A project can be modified by:

- Adding an existing or new project.
- Deleting a project.
- Adding or deleting a data source.
- Adding or deleting modules.
- Modifying the project properties.

## How to add a new project to a solution

A new project can be added to an existing solution. A solution is a collection of projects and dependent projects. When adding a project to a solution, the current model is not affected by the new project.

### To add a new project to a solution

- 1. On the **File** menu, choose **Add Project**.
- 2. Choose the **Project** wizard to use for creating the project or choose **Empty Project**.
- 3. Enter a name for the new project in the **Name** box.
- 4. Enter a location for the new project in the **Location** box.

**Note** To select another location for the new project, click **Browse**.

- 5. To add the project to the current solution, choose **Add to current solution** to add the project to the current solution or without adding it to the current solution, choose **Close current solution**.
- 6. Click **Open** and proceed as if creating a new project.

For additional information about solutions, on the **Help** menu click **Index** and then enter *solution*.

### See Also

How to create a new project

How to create a new project using the SQL Project wizard

How to create a project using the OLAP Project wizard

## How to add an existing project to a solution

An existing project can be added to a solution. A solution is a collection of projects and dependent projects. When adding a project to a solution, the current model is not affected by the new project.

### To add an existing project to a solution

- 1. On the **File** menu, click **Add Project**.
- 2. Choose the project to add from the list shown on the **Existing** or **Recent** tab.
- 3. To add the project to the current solution, choose **Add to current solution** or choose **Close current solution** to add a project without adding it to the current solution.
- 4. Click **Open**.

For additional information about solutions, on the **Help** menu click **Index** and then enter *solution*.

## How to create a new project

A new project can be created using the SQL Project or OLAP Project wizards, or from an empty project.

### To create a new project

- 1. On the **File** menu, click **New Project**.
- 2. On the **New** tab of the **New Project** dialog box, enter a name and location for the project.
- 3. Double-click one of following three options:

### • SQL Project Wizard

Connect to the appropriate SQL database and then choose the tables to automatically create an English Query SQL model based on selected entities and relationships from the database. To create additional entities and relationships use the commands on the **Model** menu.

## • OLAP Project Wizard

Connect to the appropriate Microsoft® SQL Server™ 2000 Analysis Services database and then choose the <u>cubes</u> to automatically create an English Query OLAP model based on selected entities and relationships from the structures (cubes including <u>dimensions</u>, <u>levels</u>, <u>measures</u>, <u>properties</u>, and <u>facts</u>). To create additional entities and relationships by use the commands on the **Model** menu.

## • Empty Project

Create a new, blank project. Connect to the appropriate database. Then specify the database structure manually by creating <u>database objects</u> using the **Model** menu, or by selecting tables and views with the **Import Tables** command

### on the **Model** menu.

4. On the **File** menu, choose **Save All** to save all of the components of the project.

To save only the module, choose **Save** < *module name.eqm*>.

**Note** If the project is not saved before it is closed, all of the work will be lost when it is closed. A project is not saved until one of the **Save** commands is selected.

### See Also

How to create a new project using the SQL Project wizard

How to create a project using the OLAP Project wizard

**OLAP Project Wizard** 

**Project Wizards** 

**SQL Project Wizard** 

# How to create a new project using the SQL Project wizard

The SQL Project Wizard helps you to create an <u>English Query model</u> from a SQL database. It extracts the structure of a database and creates and defines the entities and relationships for an English Query application.

### To create a project using the SQL Project Wizard

- 1. On the **File** menu, click **New Project**.
- 2. In the **Name** box of the **New Project** dialog box, enter a name and select a location for the project.
- 3. Double-click **SQL Project Wizard**.
- 4. Connect to the database using the **Data Link Properties** dialog box.
- 5. On the **Provider** tab, double-click the OLE DB provider to use.
- 6. On the **Connection** tab, follow the on-screen instructions for the specific provider and click **OK**.
  - **Note** The options on the **Connection** tab vary with the provider chosen.
- 7. In the **New Database Tables and Views** dialog box, select the table or tables from the database to make accessible to the model and then click **OK**.

Either double-click the available table or tables or select the table and click the > button to make it accessible to the model.

**Note** To include all of the tables, click the >> button. To remove one

or more tables from the **Selected** list, double-click the tables or select the ones to remove and click the < button. To remove all of the tables from the **Selected** list, click the >> button.

8. In the **Project Wizard**, select or clear the entities and relationships to be includes or removes from the project.

Expand the entities to view the relationships that can be automatically created from the SQL database structure. Click an entity, relationship, or phrasing icon to edit the proposed entities or relationships.

9. When all of the desired entities and relationships have been selected, click **OK**.

Now, refine the model and address the questions that users are expected to ask.

### See Also

How to create a new project

**Project Wizards** 

**SQL Project Wizard** 

Table/New Table Dialog Box

# How to create a project using the OLAP Project wizard

The OLAP Project Wizard helps to create an English Query application from an Microsoft® SQL Server™ 2000 Analysis Services database. Use the OLAP Project Wizard to create either an Analysis Services project, including extracting the structure of a database and creating and defining the entities and relationships for an English Query application.

### To create a project using the OLAP Project Wizard

- 1. On the **File** menu, click **New Project**.
- 2. In the **Name** box of the **New Project** dialog box, enter a name and select a location for the project.
- 3. Double-click **OLAP Project Wizard**.
- 4. Connect to the database using the **Select An Analysis Server** dialog box, enter the name of the server and select the database to use.
- 5. In the **New OLAP Cubes** dialog box, select the OLAP cube or cubes to be made accessible to the model.

Either double-click the available cube or cubes or select the cube and click the > button.

**Note** To include all of the cubes, click the >> button. To remove one or more cubes from the **Selected** list, select the ones to remove and click the < button. To remove all of the cubes from the **Selected** list, click the << button.

6. In the **Project Wizard**, select or clear the entities and relationships to be included or removed from the project.

Expand the entities to view the relationships that can be automatically created from the Analysis Services database structure.

7. When the entities and relationships are selected, click **OK**.

You can further refine the model by adding entities and relationships as needed to answer more of the questions users are expected to ask.

### See Also

How to create a new project

New < Database Object > Dialog Box

**OLAP Project Wizard** 

**Project Wizards** 

## How to remove a database from a project

When a database is no longer needed for the model, it can be removed from the project.

### To remove the database

- 1. Select the project from which the database is to be removed.
- 2. On the **Project** menu, click **Project name Properties**.
- 3. On the **Data Connection** tab, clear the **Enable SQL** or **Enable OLAP** check box.

**Note** A database cannot be removed if there are any semantic objects that refer to that database.

### See Also

Data Connection Tab (Project Properties Dialog Box)

**Project Properties Dialog Box** 

## How to remove a project from the solution

If a project no longer works for the application that is being developed, it can be removed from the solution.

### To remove a project

- 1. In the Project Explorer window, right-click the project to be deleted and then click **Remove Project**.
- 2. When asked if the project should be removed from the solution, click **Yes**.
- 3. When asked if the changes to the file or files should be saved, click **Yes** to save the changes and **No** if the changes should not be saved.

## How to create and modify project properties

The following properties can be specified for the entire English Query project:

- Properties that determine how a model uses the database.
- Properties that apply to the entire English Query application, such as loading the vocabulary from the database, loading sample data for entities, limiting the number of words to be added to the dictionary for each field or level, and setting a date to use for regression tests. This also includes setting language options such as asking a user about spelling mistakes, dealing with unknown information, and setting the dates for a fiscal year.
- Properties that define defaults for ambiguous relationships.

### To set the project options

- 1. On the **Project** menu, click **<project name> Properties**.
- 2. Choose the various tabs, as needed, and make the changes.
- 3. When all of the changes have been made, click **OK**.

### See Also

Data Connection Tab (Project Properties Dialog Box)

<u>Default Relationships Tab (Project Properties Dialog Box)</u>

**English Tab (Project Properties Dialog Box)** 

**Project Properties Dialog Box** 

## How to add a module

An existing or new module can be added to the project. Using modules allows a model to be created once and used multiple times. Modules not only save time, but also provide consistency when creating multiple applications that have similar functions.

### To add a module

- 1. In the Project Explorer window, right-click the project to which the module is to be added, and then click **Add Module**.
- Choose a new or existing module.
   If adding a new module, enter a name for it in the **Name** box.
- 3. Click **Open**.

## How to remove a module

If a module is no longer needed in the project, it can be removed.

### To remove a module

• Right-click the module, and then click **Remove <module name>**.

## How to modify the data source in a project

To specify a database other than the one selected when creating a project, change the OLE DB data source.

## To modify the data source

1. In the Project Explorer window, double-click the module file (.eqm) for which the data source is to be changed.

**Note** The **Model Editor** becomes active and the English Query commands are displayed.

- 2. On the **Project** menu, click **<project name> Properties**.
- 3. On the **Data Connection** tab, select or clear the appropriate database connection information.

## **See Also**

Data Connection Tab (Project Properties Dialog Box)

**Model Editor** 

**Project Properties Dialog Box** 

# **Working with Models**

After creating a model, refine the model to answer the questions users are expected to ask about the database information.

Refine a model by:

- Adding entities and relationships.
- Adding or deleting a database connection.
- Adding or modifying fields.

# How to add entities and relationships using the Create Semantics command

When refining a model, entities and relationships that already exist in the database can be added to the model, either manually or by using the **Create Semantics** command.

### To add entities and relationships

1. In the Project Explorer window, double-click the module file (.eqm) to the entities and relationships are to be added.

The **Model** menu appears and the Model Editor becomes active.

- 2. On the **Model** menu, click **Create Semantics**.
- 3. Select or clear the entities and relationships to add to the project.

Expand the entities to view the relationships that can be automatically created from the database structure. Relationships where more than one exists can also be expanded.

**Note** If the **Project Wizard** is unable to find new entities or relationships to add to the model from the existing tables or cubes, add tables or cubes from the database to expand the number of questions that can be answered.

### See Also

How to add a table

How to add an Analysis Services (formerly OLAP) cube to a model

How to create a new project using the SQL Project Wizard

How to create a project using the OLAP Project Wizard

**Model Editor** 

**OLAP Project Wizard** 

Project Wizards

SQL Project Wizard

### How to add or change a SQL connection

If the current Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services database or current SQL database does not answer all of the questions that a user might ask, modify the project by adding a new or changing the current SQL database connection. There can be, at most, one SQL and one Analysis Services database per project.

**Note** Before adding a SQL connection to an Analysis Services project, make sure that you are a member of the OLAP Administrators group and the Microsoft Windows NT® 4.0 or Windows® 2000 user group on the server of the Analysis Services database for which the model is being built. You must have OLAP Administrator permissions to load the tables underlying the cubes using Decision Support Objects (DSO).

### To add or change a SQL connection

- 1. Select the project to which the database is being added.
- 2. On the **Project** menu, click **Project name Properties**.
- 3. On the **Data Connection** tab, select **Enable SQL** if adding a SQL database.

**Note** Both SQL and Analysis Services databases can be enabled.

- 4. On the **Provider** tab of the **Data Link Properties** dialog box, select the OLE DB provider and then click **Next**.
- 5. On the **Connection** tab, enter or select the server or data source name, the log on information, and the database, and then click **OK**.

### See Also

**Expanding an English Query Model** 

Data Connection Tab (Project Properties Dialog Box)

Enabling SQL for an English Query Model

Project Properties Dialog Box

# How to add or change an Analysis Services (formerly OLAP) connection

If the current SQL database does not answer all of the questions that a user may ask, modify the SQL Project by adding a Microsoft® SQL Server™ 2000 Analysis Services data connection. Questions that require Multidimensional Expressions (MDX) can then be answered.

**Note** There can be, at most, one SQL and one Analysis Services database for each project.

### To add or change an Analysis Services connection

- 1. Select the project to which the Analysis Services database is to be added.
- 2. On the **Project** menu, click **Project name Properties**.
- 3. On the **Data Connection** tab, select **Enable OLAP** if adding an Analysis Services database.

**Note** Both SQL and Analysis Services databases can be enabled.

- 4. Click Change.
- 5. In the **Select an Analysis Server** dialog box, enter the Analysis Services server, select the database, and then click **OK**.
- 6. Click **OK**.

The Analysis Services connection appears in the Project Explorer and Data View windows.

### See Also

**Expanding an English Query Model** 

Data Connection Tab (Project Properties Dialog Box)

**Enabling Analysis Services for an English Query Model** 

**Project Properties Dialog Box** 

Select An Analysis Server Dialog Box

### How to create a mixed model

An English Query application based on a mixed model allows access to the Microsoft® SQL Server<sup>TM</sup> tables that underlie an OLAP cube. English Query first attempts to answer the questions using the OLAP cube. If unsuccessful, it then accesses the data in the underlying SQL Server tables.

#### To create a mixed model

- 1. Enable SQL on the **Data Connection** tab of the **Project Properties** dialog box.
- 2. Add any tables that are necessary for the model.
- 3. Create the entities and relationships needed for the model.

### See Also

How to create a relationship using the Canvas pane

How to add a table

How to add or change a SQL connection

How to add tables and views from a SQL data source

How to create an entity

How to create relationships using the Create Relationships command

### How to delete a field from a model

A field can be deleted from a <u>model</u> when it is no longer needed. Deleting a field from a model does not delete it from the database table.

#### To delete a field

- 1. On the **SQL** tab, expand **Tables**, and then double-click the table from which the filed is to be deleted.
- 2. To select the field to delete, click the far-left gray column for the field.
- 3. Right-click anywhere in the selected field, and then click **Delete Rows**.
- 4. Click **OK**.
- 5. If prompted to delete the field, click **OK**.
- 6. To delete any entities, joins, phrasing, relationships, or roles that refer to the field, select **In addition, delete the following objects which refer to the field <fieldname>** in the **Confirm Deletion** dialog box.

**Note** Generally after deleting a field, delete the objects or the project will not compile.

### See Also

**Model Editor** 

**SQL Tab (Model Editor)** 

Table/New Table Dialog Box

### How to edit a field in a model

The properties of a field in an open <u>model</u> can be reviewed and edited. Changes affect the field as it occurs only in the project. Columns in the database table remain unchanged.

#### To edit a field

- 1. On the **SQL** tab, expand **Tables**.
- 2. Double-click the table containing the field to be edited.
- 3. In the **Table\New Table** dialog box, select the field by clicking in the gray column to the left of the field name.
- Review and edit the properties of the field.
   Changes can also be made on the **Basic** and **Advanced** tabs.
- 5. When all of the desired changes have been made, click **OK**.

### See Also

Advanced Tab (Table/New Table Dialog Box)

Basic Tab (Table/New Table Dialog Box)

**Model Editor** 

**SQL Tab (Model Editor)** 

Table/New Table Dialog Box

# **Working with Joins**

<u>Joins</u> allow an association between a <u>field</u> in one <u>table</u> or view with a field of the same data type in another table or view. Joins show how the data in one table relates to the data in another table.

By using joins, data from two or more tables or queries based on logical relationships among the tables can be retrieved. Joins can be added, edited, or deleted to retrieve the data from the database.

### How to add a join

In English Query, when a new <u>table</u> or <u>view</u> is added to the <u>model</u>, associations between the new table or view and an existing table or view must be made. The associations are made by adding a <u>join</u>.

### To add a join

- 1. On the **SQL** tab, expand **Tables**, and select the table to which the join is added.
- 2. Expand the table, right-click **Joins**, and click **Add Join**.
- 3. In the **New Join** dialog box, select the name of the **Destination Table** for the join.
- 4. Click **Add**.

**Note** If the **Add** button is not available, select a **Destination Table**.

- 5. In the **Join Condition** dialog box, select the pair of fields to be joined, and click **OK**.
- 6. Click **Allow outer join** if it is necessary to use this join as an outer join.

### See Also

Join/New Join Dialog Box

**Model Editor** 

SQL Tab (Model Editor)

### How to delete a join

Unused <u>joins</u> can be deleted from a <u>model</u> ( for example, if a join no longer represents a relationship).

### To delete a join

- 1. Select the **SQL** tab, and then expand **Tables**.
- 2. Expand the table containing the join to be deleted.
- 3. Expand **Joins**, right-click the join to be deleted, and then click **Delete**.
- 4. When asked to confirm the deletion, click **OK**.

### See Also

**Model Editor** 

**SQL Tab (Model Editor)** 

### How to edit a join

A database <u>join</u> can be modified (for example, if the fields involved in the join change).

### To edit a join

- 1. On the **SQL** tab, expand **Tables**, and expand the table containing the join to be edited.
- 2. Expand **Joins**, and then double-click the join to edit.
- 3. In the **Join/New Join** dialog, select the join to edit.
- 4. Click **Edit**.
- 5. In the **Join Condition** box, edit the source table and/or destination table fields, and click **OK**.

### See Also

Join/New Join Dialog Box

**Model Editor** 

**SQL Tab (Model Editor)** 

# **Working with Tables**

Existing database <u>tables</u> can be included in a <u>model</u>. Tables can also be deleted, refreshed, or have fields added to them.

### How to add a field to a table

Include existing database <u>fields</u> in a <u>model</u> one by one. For example, if a field has been added to a table in a database, it can be made available to and used in an <u>English Query application</u>.

#### To insert a field in a table

- 1. On the **SQL** tab, right-click the table to which a field is to be added and click **Add Field**.
- 2. In the **Field Name** box, enter the field name.

**Note** Enter the name of each new field on a separate line.

- 3. Select the **Data Type**.
- 4. If the field is to have a caption, enter one in the **Caption** box.
- 5. On the **Basic** tab, choose or enter the data type options.

**Note** The available options change with the data type selected.

6. On the **Advanced** tab, choose or enter the options for allowing Null values, denormalization, and aggregation for the selected field.

### See Also

Advanced Tab (Table/New Table Dialog Box)

Basic Tab (Table/New Table Dialog Box)

**Model Editor** 

SQL Tab (Model Editor)

Table/New Table Dialog Box

### How to add a table

Further expand a model by adding tables to it.

#### To add a table

- 1. On the **Model** menu, select **Add Database Object**, and then select **Add Table**.
- 2. For each new field, enter the name in the **Field Name** column, select the **Data Type**, and enter an optional **Caption**.
- 3. To choose the fields that constitute the primary key, select the field, right-click on the far left gray column for that field, then and choose **Primary Key**.

To select multiple fields at once, press CTRL and click in the far left gray column for each field.

4. On the **Basic** tab, select or enter the options desired.

**Note** The available options vary with the **Data Type** chosen.

- 5. Click the **Advanced** tab and set or change the options related to Null values, denormalization, and aggregation.
- 6. When finished setting the options, click **OK**.

The table is created with a default name.

- 7. On the **SQL** tab, right-click the new table, click **Rename**.
- 8. Enter a name for the table and press ENTER.

### See Also

Advanced Tab (Table/New Table Dialog Box)

Basic Tab (Table/New Table Dialog Box)

**Model Editor** 

SQL Tab (Model Editor)

Table/New Table Dialog Box

### How to add tables and views from a SQL data source

Tables or views that exist in the SQL data source can be added to SQL-enabled models.

### To add tables from a SQL data source

1. In the **Project Explorer** window, double-click the module file (.eqm) in the project to the tables are to be added.

The Model Editor specific to that project and the **Model** menu appear.

- 2. On the **Model** menu, click **Import Tables**.
- 3. In the **Available** list of the **New Databases Tables and Views** dialog box, double-click the table, tables, or views to make them accessible to the model, and then click **OK**.

To add all of the tables, click the >> button.

To remove a table, select it from the **Selected** list, and then click the < button.

### **See Also**

**Model Editor** 

New < Database Object > Dialog Box

### How to change the table options for a relationship

Change the table options for a relationship to calculate or identify the table that contains all the links to the tables (<u>join table</u>) represented by <u>entities</u>, or to specify a condition for when the relationship is true.

### To change the table options for a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, and then double-click the relationship whose table options are to be changed.
- 3. On the **Database** tab, select a table from the **Table that contains joins to all entities in this relationship box**.

As an alternative, have English Query find that table by clicking **Calculate Default Join Table**.

4. To specify when the relationship is true, select **This relationship is true only when the following SQL condition is true** and enter an appropriate SQL condition.

For example, if the relationship is *products are shipped to customers* and the SQL condition is a requirement that customers' credit be approved, type:

Approval= 'T'

5. To have English Query answers generated from the most recent data in the database, select **Show most recent data by default**.

#### See Also

Database Tab (Relationship/New Relationship Dialog Box)

Model Editor

Relationship/New Relationship Dialog Box
Semantics Tab (Model Editor)

### How to delete a table

Sometimes, it is necessary to delete a table and its fields from a project.

**Note** Deleting a table from a project does not delete it from the database.

### To delete a table from a project

- 1. On the **SQL** tab, right-click the table to be deleted and click **Delete**.
- 2. In the **Confirm Deletion** dialog box, select **In addition, delete the following objects which refer to the table** if any entities, joins, phrasing, relationships, or roles that refer to the table.

**Note** It is necessary to delete the objects or the project will not compile if the table is missing.

### See Also

**Model Editor** 

SQL Tab (Model Editor)

### How to edit a table

Sometimes it is necessary to review or edit the fields that are part of a <u>model</u>.

**Note** Changes affect only the table as it is defined in the model, not its structure in the database.

#### To modify a table

- 1. On the **SQL** tab, expand **Tables** and then double-click the table to modify.
- 2. Modify the table by adding a field, deleting field, or changing the properties of a field
- 3. On the **Basic** tab, select, enter, or change the options.

**Note** The available options vary with the **Data Type** chosen.

4. Click the **Advanced** tab and set or change the options related to Null values, denormalization, and aggregation.

If it is necessary to delete a field, click the far left gray column to select the field, and then press DELETE.

#### See Also

Advanced Tab (Table/New Table Dialog Box)

Basic Tab (Table/New Table Dialog Box)

How to add a field to a table

How to change the table options for a relationship

**Model Editor** 

Table/New Table Dialog Box

### How to enable full-text searching for a field

Full-text searches for a field can be enabled if the underlying database supports full-text indexing.

#### To enable full-text searching

- 1. In the Model Editor, click the **SQL** tab.
- 2. Select the table.

**Note** After making changes to the full-text index in any tables in the database, refresh the table or tables before proceeding to step 3.

- 3. Select the field to be enabled.
- 4. On the **Basic** tab, select **Full-text indexed**.

The **Search Type** changes to **Freetext**.

- 5. Repeat steps 1 through 4 for each field to be included.
- 6. Click **OK**.

#### See Also

Basic Tab (Table/New Table Dialog Box)

**Enabling Full-Text Search for an English Query Model** 

# How to refresh all tables in a project

Occasionally, it is necessary to refresh the database structure used in a <u>project</u> (for example, if tables or fields have been added to or deleted from the database).

### To refresh the tables in a project

• On the **Model** menu, click **Refresh Tables**.

# **Working with Entities**

<u>Entities</u> are real-world objects, referred to by a noun. They can be a person, place, thing, or idea. In English Query, use entities to create relationships. To create and refine a model, create, add, delete, or edit the entities.

### How to add an entity to a relationship

After entities are defined, they can be used to create relationships.

#### To add an entity to a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the relationship to which the entity is to be added, and click **Edit**.
- 3. On the **Semantics** tab of the **New Relationship** dialog box, click **Add** in the **Entities** section.
- 4. In the **Select Entities** dialog box, select an entity that will be involved in the relationship and click **OK**.
- 5. Double-click an entity from the list in the **Entities**.
- 6. In the **Role** dialog box, select the desired options.

#### See Also

How to create an entity

How to delete an entity from a relationship

How to edit the entity's role in a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

< Role Name > Role Dialog Box

Semantics Tab (Relationship/New Relationship Dialog Box)

### How to create an entity

Sometimes it is necessary to create an additional <u>entity</u> for a <u>model</u> in order to make a question return the right answer.

#### To create an entity

- 1. On the **Semantics** tab, right-click **Semantic Objects**, and then click **Add Entity**.
- 2. In the **Words** box, enter the word that identifies the entity and press ENTER.

As many words or phrases as needed can be added. Enter the word or phrase and press ENTER after each. For example, for an entity that is represented by an **Employee** table, enter *employee*, *worker*, and *staff member*. English Query automatically adds the words or phrases with a comma separating each to the **Words** box.

**Note** The entries should appear in the singular form, not in the plural.

- 3. Click the **Add Synonym** button, the ellipses button (...) to view a list of synonyms for the selected word or words.
- 4. Click on the words from the **Available words** list that should be moved to the **Selected words** list.

To remove a word in the **Selected words** list, click on the word to move it to the **Available words** list.

When clicking outside of the synonym list box, the words, separated by commas, automatically appear in the **Words** box.

5. Under **Entity associated with**, choose the way the entity is to be represented.

If the entity is represented by a table, select the table to which it is associated and the fields of the table to be displayed on the **Semantics** 

tab.

**Note** If **Table** is chosen, the **Name Type** option is not available.

If **Field(s)** are chosen to represent the entity, select the table to which the field or fields belong, and then select the specific field or fields.

If **OLAP** is chosen, indicate whether the entity is associated with a level, dimension, measure, property, or fact.

**Note** The availability of the remaining Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services (OLAP) options varies with the choice of level, dimension, measure, property or fact. Choose the available options.

If the entity is not associated with a table, field, or OLAP object, choose **None**.

- 6. In the **Entity Type** box, select a type.
- 7. In the **Name Type** box, select an option.

This option is available only if the entity is to be associated with **Field(s)** or an OLAP level or property.

8. Select **Add values of entity to model** if English Query is to automatically add the values in the field or set of fields represented by the entity to the model knowledge for this application.

This ensures that questions using specific database values can be answered.

**Note** The amount of time it takes to load these values when the application is compiled depends upon the number of field values in the database.

9. Enter any **Help text** to be displayed to the user.

For example, when the user asks a question such as "What is a customer", help text describing a customer is displayed.

10. Click **Create Relationships** to create new relationships for the new

entity.

11. Click **Advanced** if to further define the entity, specify a synonym for a particular instance of the entity, or set default conditions.

### See Also

**Advanced Entity Properties Dialog Box** 

**Entity/New Entity Dialog Box** 

**Model Editor** 

Relationship/New Relationship Dialog Box

## How to delete an entity

Sometimes it is necessary to delete an <u>entity</u> from a <u>model</u>. Deleting an entity affects all relationships between it and other entities.

#### To delete an entity

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Entities**, right-click the entity to delete, and then click **Delete**.
- 3. In the **Confirm Deletion** dialog, select **In addition, delete the following objects which refer to the entity <entityname>** to delete any joins, phrasing, relationships, or roles that refer to the entity.

**Note** Generally, delete the object associated with the entity; otherwise, the application will not compile if the entity is deleted and the object is not.

4. Click  $\mathbf{OK}$  when prompted to delete the entity.

#### See Also

How to create an entity

How to edit an entity

**Model Editor** 

SQL Tab (Model Editor)

## How to delete an entity from a relationship

When an entity no longer participates in a relationship, delete it from the relationship. Deleting an entity from a relationship does not delete it from the model.

### To delete an entity from a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the relationship containing the entity to delete, and then click **Edit**.
- 3. On the **Semantics** tab of the **Relationship**/**New Relationship** dialog box, select the entity in the **Entities** section, and then click **Delete**.

#### See Also

How to add an entity to a relationship

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

## How to edit an entity

In English Query an <u>entity</u> can be modified by changing its semantic properties and corresponding database properties.

#### To modify an entity

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Entities**, and then double-click the entity to modify.
- 3. In the **Entity/New Entity** dialog box, select the options to modify.
- 4. Click **Advanced** and change any options in the **Advanced Entity Properties** dialog box.

#### See Also

**Advanced Entity Properties Dialog Box** 

Entity Defaults Tab (Advanced Entity Properties Dialog Box)

**Entity/New Entity Dialog Box** 

**Model Editor** 

Name Synonyms Tab (Advanced Entity Properties Dialog Box)

Relationship/New Relationship Dialog Box

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

## How to edit the entity's role in a relationship

An entity's role in a relationship may require modification. For example, when testing a model, you may discover that a custom <u>join path</u> is required or that a measure for an entity's role needs to be modified.

### To edit an entity in a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the relationship with the entity to edit, and click **Edit**.
- 3. In the **Entities** section of the **Semantics** tab of the **Relationship/New Relationship** dialog box, select the entity to edit and click **Edit**.
- 4. In the **Role** dialog box, change the options and click **OK**.

#### See Also

< Role Name > Role Dialog Box

Model Editor

Relationship/New Relationship Dialog Box

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

## **Working with Relationships**

Relationships describe how the entities relate to one another. A new relationship may be needed to reflect the addition of a join to a database structure.

After creating relationships in a model, modify them by:

• Adding additional relationships.

To add relationships, use the **Create Relationships** or **Add Relationship** commands, click **Create Relationships** on the **Entity/New Entity** dialog box, drag entities to the Canvas pane, or click **Suggest Relationships** in the Model Test window.

- Deleting relationships.
- Modify a relationship, for example, by adding phrasings.

## How to add a relationship

Refine a model by adding relationships to those automatically generated by the Project wizard (for example, the relationship *customers have names*).

#### To add a relationship

- 1. In the hierarchical pane of the **Semantics** tab of the Model Editor, right-click the entity for which the relationship is being created.
- 2. Click **Add Relationship**.
- 3. On the **Semantics** tab under **Entities**, click **Add**.

Double-click the other entity to be included in the relationship.

Specify when the relationship occurs by setting the **When**, **Start** and **End** times. Set a **Default** date if to have the relationship assume a selected date.

To set the relationship to occur over a period of time, set **Duration**.

To set the location of the relationship, set the **Where** option.

- 4. Under **Phrasings**, click **Add** and proceed as if adding a phrasing.
  - To specify that specific phrasings be grouped together to define the relationship, select them, and then click **Group**.
- 5. In **Help Text**, enter any information about the relationship that it to be displayed to the user.
- 6. To let English Query find the table containing all the joins to the tables and fields representing the entities in the relationship, click the **Database** tab and select a table from the **Table that contains joins to all entities in this relationship** list or click **Calculate Default Join Table**.

**Note** The **Database** tab is available only when a project is enabled for SQL.

- 7. To specify when a relationship is true, on the **Database** tab select **This relationship is true only when the following SQL condition is true**, and enter the appropriate SQL condition.
- 8. To have English Query generate answers from the most recent data in the database, on the **Database** tab select **Show most recent data by default**.

#### See Also

Database Tab (Relationship/New Relationship Dialog Box)

How to add a join

How to add an entity to a relationship

How to add phrasings to a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

### How to add phrasings to a relationship

Phrasings are a way of expressing relationships among entities. When considering a phrasing type, select the phrasings that most closely reflect how users are likely to ask their questions.

### To add phrasing to a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the relationship to which the phrasing is to be added, and click **Add Phrasing**.
- 3. In the **Select Phrasing** dialog box, double-click a phrasing type.
- 4. In the dialog box specific to the phrasing type, choose the phrasing structure, and click **OK**.
- 5. Check the **Relationship**/**New Relationship** dialog box to make sure that the desired phrasing is listed at the bottom of the dialog box.

#### See Also

**Adjective Phrasing Dialog Box** 

**Grouped Phrasings Examples** 

**Model Editor** 

Name/ID Phrasing Dialog Box

<u>Preposition Phrasing Dialog Box</u>

**Select Phrasing Dialog Box** 

Semantics Tab (Model Editor)

**Subset Phrasing Dialog Box** 

Trait Phrasing Dialog Box

Verb Phrasing Dialog Box

## How to add time and location options to a relationship

Using English Query, it is possible to specify when and where a relationship occurs. The time and location options are specified in an entity (for example, a date entity that specifies when *salespeople sold products to customers*).

### To add time and location options to a relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships** and double-click an existing relationship.
- 3. On the **Semantics** tab of the **Relationship/New Relationship** dialog box, select the entity from the **Entities** box.
- 4. Select **When**, **Start**, **End**, **Default date**, and **Duration**, and **Where** this relationship occurs.

#### See Also

Model Editor

Relationship/New Relationship Dialog Box

Semantics Tab (Relationship/New Relationship Dialog Box)

### How to create a relationship using the Canvas pane

Adding relationships to those automatically generated (for example, by the SQL Project or the OLAP Project wizard) can refine a model. For example, to create the relationship *stores sell books* between **store** and **product**, drag the entities from the **Entities** folder in the left-hand hierarchical pane onto the Canvas pane.

#### To create a relationship using the Canvas pane

- 1. In the hierarchical pane of the **Semantics** tab of the Model Editor, select the entity to be included in the new relationship and drag it onto the Canvas pane.
- 2. Select any other entity in the hierarchical pane that should be included in the relationship and drag it onto the entity currently in the Canvas pane.
  - The **Relationship**/**New Relationship** dialog box appears.
- 3. On the **Semantics** tab under **Phrasings**, click **Add**.
- 4. In the **Select Phrasing** dialog box, select the desired phrasing, click **Add**, and select the appropriate options in the phrasing dialog box.
  - To create a group of phrasings that work together to define the relationship, select the phrasings and click **Group**.
- 5. Enter any **Help Text** to be displayed to the user.
- 6. On the **Database** tab, select a table from the **Table that contains joins to all entities in this relationship** box or click **Calculate Default Join Table** to have English Query find the table that contains all the joins to the tables and fields that represent entities in the relationship.
  - The **Database** tab is available only when a project is enabled for SQL.
- 7. Select **This relationship is true only when the following SQL condition is true** and enter the appropriate SQL condition to specify

when a relationship is true.

8. Select **Show most recent data by default** to have English Query answers generated from the most recent data in the database.

### See Also

Database Tab (Relationship/New Relationship Dialog Box)

How to add a join

How to add an entity to a relationship

How to add phrasings to a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

**Select Phrasing Dialog Box** 

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

## **How to create relationships from the Model Test** Window

When testing a model if a question cannot be understood or is answered incorrectly, English Query can suggest and create possible relationships.

#### To cre

modify an existing relationship.

eate	e relationships from the Model Test window
1.	Create a model and double-click it in the Project Explorer window.
2.	From the <b>Debug</b> menu, click <b>Start</b> .
3.	Enter a question and click .
4.	When there is a question cannot be understood or is answered incorrectly, click .
5.	Select the entities and identifying information, and the relationships that create the appropriate phrasing for the question.
6.	Click <b>OK</b> .
7.	Retest the question.
8.	If the question continues to be answered incorrectly or cannot be understood, repeat Steps 4 through 7.
9.	If the question continues to be answered incorrectly or cannot be

understood, create a relationship using one of the other methods, or

10. Repeat Steps 3 through 9 as needed until the model answers the question appropriately.

## See Also

How to add a relationship

How to add a relationship using the Canvas pane

How to create relationships using the Create Relationships command

How to modify a relationship

How to test a model

**Model Test Window** 

# How to create relationships using the Create Relationships command

Models may be refined by creating additional relationships. To create relationships

- 1. In the hierarchical pane of the **Semantics** tab of the Model Editor, expand **Semantics**, expand **Entities**, and then right-click the entity for which the relationship is being created.
- 2. Click **Create Relationships**.
- 3. In the SQL and OLAP Project wizards, select the relationships to add to a model.

#### See Also

**Model Editor** 

**Project Wizards** 

Semantics Tab (Model Editor)

# How to associate English Data values with a relationship

To use adjectives or subset words to describe a relationship, it is necessary to specify where they can be found: either in a database field or in a set of word-value pairs. Specify where the adjective or subset words are found by associating English Data values with the relationship.

#### To associate English Data values with a relationship

- 1. Choose or create the relationship containing the Adjective or Subset phrasing to which associate English Data values are to be added.
- 2. In the **Adjective Phrasing** or **Subset Phrasing** dialog box, click **Associate English Values**.
- 3. Decide whether the adjectives or subset words are in a field or in a lookup table, or whether they need to be specified in the application.

If the adjectives or subset words are in a field or lookup table, select **Read English from table**.

If the words are in a <u>lookup table</u>, select the table, word field, and value field from the appropriate lists.

To create a lookup table that exists only in the application, select **Specify values** and then enter the values that the codes represent.

4. Click **OK** to return to the **Adjective Phrasing** or **Subset Phrasing** dialog box.

#### See Also

Adjective Phrasing Dialog Box

English Data Values Dialog Box

How to add a relationship

How to add phrasings to a relationship

How to create relationships using the Create Relationships command

How to modify a relationship

Subset Phrasing Dialog Box

## How to delete a relationship

Delete a relationship from the model if it is no longer needed.

## To delete a relationship

- 1. In the hierarchical pane of the **Semantics** tab of the Model Editor, expand the **Relationship** folder and right-click the relationship to delete.
- 2. Click **Delete**.
- 3. Click **OK** when prompted to delete the relationship.

#### See Also

How to create a relationship using the Canvas pane

How to add a relationship

How to create relationships from the Model Test Window

How to create relationships using the Create Relationships command

**Model Editor** 

Semantics Tab (Model Editor)

## How to modify a relationship

After creating a relationship, it can be refined or expanded. For example, it is possible to add or change the entities and phrasing, or change the joins in SQL-enabled projects.

## To modify a relationship

- 1. In the hierarchical pane of the **Semantics** tab of the Model Editor, expand the **Relationships** and double-click the relationship to modify.
- 2. Make the changes on the **Semantics** and **Database** tabs.

The **Database** tab is available only for SQL-enabled project.

#### See Also

Database Tab (Relationship/New Relationship Dialog Box)

How to add phrasings to a relationship

How to add time and location options to a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

## **Working with Dictionary Entries**

English Query includes a dictionary containing thousands of common English words. This dictionary provides an English Query application with the terminology needed to answer most questions posed in English.

Creating entities (with <u>synonyms</u>) and relationships provides most of the specialized vocabulary required for an application. Create a dictionary entry if the word being defined is not associated with a particular entity or relationship. The new terms appear under the **Dictionary Entries** on **Semantics** tab in the Model Editor. To view the entries, expand **Dictionary Entries**, then add, edit, or delete dictionary entries.

## How to add a dictionary entry

Add new words to the dictionary if the application requires specialized terms that are not generally known. For example, an application used in a medical setting may require the addition of specialized medical terms.

## To add a dictionary entry

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Right-click **Dictionary Entry** and click **Add Dictionary Entry**.
- 3. Select the **Dictionary Entry Type** option.
  - Choose **Word** to define a main entry in the dictionary, or **Read synonym** or **Write synonym** to define a synonym for a word already in the dictionary.
- 4. Select or enter the specific information for the entry type.

#### See Also

New Dictionary Entry Dialog Box
Semantics Tab (Model Editor)

## How to delete a dictionary entry

Dictionary entries that have been added to an <u>English Query application</u> can be deleted when they are no longer needed.

## To delete a dictionary entry

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Dictionary Entry**, right-click the entry to delete, and then click **Delete**.

## See Also

How to add a dictionary entry

**Model Editor** 

Semantics Tab (Model Editor)

## How to edit a dictionary entry

Dictionary entries that have been added to the <u>English Query application</u> can be modified. For example, synonyms can be added to a word that is in the dictionary.

## To edit a dictionary entry

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Dictionary Entry**, right-click the entry to edit, and then click **Edit**.
- 3. Make modifications to its options in the **Dictionary Entry** dialog box.

## See Also

**Model Editor** 

New Dictionary Entry Dialog Box

Semantics Tab (Model Editor)

## **Working with Command Relationships**

After the <u>entities</u> for an English Query application have been defined, they can be used to create a <u>command relationship</u> (for example, a command relationship that allows a user to enter, "Display the paintings by xxx artist that are watercolors").

Command relationships can enhance the English Query <u>model</u>. They require a command ID and a parameter ID that can be set or modified on the **Command** tab of the **Relationship**/**New Relationship** dialog box.

Command relationships can be added, modified, or deleted. In addition, phrasings can be added or modified.

## How to add a phrasing to a command relationship

If a <u>command relationship</u> does not adequately address the questions being asked or if a <u>model</u> needs to be modified, add phrasings to the command relationship.

### To add a phrasing to a command relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Command Relationship** and then, right-click the command relationship to be edited, and then click **Add Phrasing**.
- 3. Double-click the phrasing to be added.
- 4. Create the phrasing and click **OK**.

### See Also

Using Command Relationships in an English Query Model

How to add phrasings to a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

**Select Phrasing Dialog Box** 

## How to create a command relationship

<u>Command relationships</u> consist of <u>entities</u> and command phrasings and are used to define model-specific commands. Unlike other relationships, command phrasings require both a command ID and a parameter ID.

### To create a command relationship

1. On the **Semantics** tab of the Model Editor, expand **Entities** and select the entity to be included in the command relationship.

If relevant, select **When**, **Start**, **End**, **Default Date**, **Duration**, and/or **Where**.

**Note** These options will be available depending on the type of entity choosen. If an option is dimmed, it is not available for the selected entity.

- 2. In the **Phrasings** section, click **Add**.
- 3. Select **Command Phrasing** and click **OK**.
- 4. Create the phrasing.

In command relationships, verbs are case sensitive so capitalize them the same way the users will type them in questions. To provide the most options for users, enter the verb with the first letter capitalized and then add a synonym of the same verb but with the first letter not capitalized.

- 5. Click **OK** to the warning about capitalization.
- 6. Enter **Help text** if desired.
- 7. On the **Command** tab, enter a command name.

## 8. Select the **Parameter**.

The **Parameter ID** is automatically generated

## See Also

**Command Phrasing Dialog Box** 

Using Command Relationships in an English Query Model

Command Tab (Relationship/New Relationship Dialog Box)

How to add an entity to a relationship

How to add phrasings to a relationship

Relationship/New Relationship Dialog Box

# How to create time and location options for command relationships

You can specify when and where a <u>command relationship</u> occurs.

## To create time and location options for command relationships

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships** and then double-click the command relationship to which time or location options are to be added.
- 3. On the **Semantics** tab, click **Add**, select the entities to add for time and place options, and click **OK**.
- 4. Select more than one entity by selecting the first entity, holding down the CTRL key, and selecting the additional entities.
- 5. Select the **When**, **Start**, **End**, **Default date**, **Duration**, and **Where** options wanted.

#### See Also

Using Command Relationships in an English Query Model

How to add phrasings to a relationship

**Model Editor** 

Relationship/New Relationship Dialog Box

**Select Phrasing Dialog Box** 

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

## How to delete a command relationship

When a <u>command relationship</u> is no longer needed in a <u>model</u>, delete it.

### To delete a command relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the command relationship to delete, and then, click **Delete**.
- 3. When prompted to delete the command relationship, click **OK**.

## See Also

Using Command Relationships in an English Query Model

**Model Editor** 

Semantics Tab (Model Editor)

## How to edit a command relationship

After a <u>command relationship</u> is created, it can be modified to handle specific questions in a <u>model</u>.

#### To edit a command relationship

- 1. On the **Semantics** tab, expand **Semantic Objects**.
- 2. Expand **Relationships**, right-click the command relationship to edit, and then, click **Edit**.
- 3. On the **Semantics, Database, and Command** tabs, make the desired changes, and click **OK**.

#### See Also

Using Command Relationships in an English Query Model

Command Tab (Relationship/New Relationship Dialog Box)

Database Tab (Relationship/New Relationship Dialog Box)

How to add a phrasing to a command relationship

How to add an entity to a relationship

How to create time and location options for command relationships

**Model Editor** 

Relationship/New Relationship Dialog Box

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

# Working with Analysis Services (formerly OLAP Services)

After creating a <u>project</u> using the OLAP Project wizard, add cubes to the project, modify <u>cubes</u>, <u>dimensions</u>, <u>properties</u>, and <u>levels</u>, and add entities, relationships, and <u>dictionary entries</u>.

Add, edit, and delete entities, relationships, and dictionary entries in the same way that it is done for SQL-enabled projects.

## See Also

How to add a dictionary entry

How to add a relationship

How to add an entity to a relationship

How to delete a relationship

How to delete an entity

How to delete an entity from a relationship

How to edit a dictionary entry

How to edit an entity

How to edit the entity's role in a relationship

# How to add an Analysis Services (formerly OLAP) cube to a model

To enhance a model and create questions for OLAP cube data, add one or more OLAP cubes to a model, or add an OLAP cube to an existing SQL-enabled project

**Note** The project must be either an OLAP or OLAP-enabled project.

## To add an OLAP cube to a project

- 1. On the **Model** menu, click **Import Cubes**.
- 2. If asked, enter the name of the Analysis server on which the database resides, select the name of the database.
- 3. From the **Available** list, double-click the desired OLAP cube or cubes.

#### See Also

Analysis Services in English Query

**Enabling Analysis Services for an English Query Model** 

**Model Editor** 

**OLAP Cube Dialog Box** 

## How to edit an Analysis Services (formerly OLAP) cube

The objects in the <u>OLAP</u> <u>cube</u> in the model can be changed.

#### To edit an OLAP cube

- 1. On the **OLAP** tab, expand **Cubes**.
- 2. Right-click the cube that to edit, and then click **Edit**.
- 3. Select the fact table and make the changes to **Dimensions**, **Measure**, **Aggregation**, **Field** and/or **Units**.

To delete a measure, select it by clicking in the gray column to the left of it and then click **Delete**.

4. When satisfied with the changes, click **OK**.

#### See Also

Additional Relationships

Analysis Services in English Query

**Enabling Analysis Services for an English Query Model** 

How to enable Analysis Services (formerly OLAP)

**Level-Level Relationships** 

**Model Editor** 

**OLAP Cube Dialog Box** 

**OLAP Tab (Model Editor)** 

## **How to edit Analysis Services levels**

The levels associated with each dimension can be edited to provide additional information and allow questions to be refined.

#### To edit Analysis Services levels

- 1. On the **OLAP** tab, expand **Dimensions**, and then expand the dimension whose levels are to be edited.
- 2. Right-click the level, and then click **Edit**.
- 3. Make the changes to the **Level**, **Table**, **Field**, or the level properties. To delete a level, select it by clicking in the gray column to the left of it and then, click **Delete**.
- 4. When satisfied with the changes, click **OK**.

#### See Also

**Analysis Services in English Query** 

**Enabling Analysis Services for an English Query Model** 

How to enable Analysis Services (formerly OLAP)

Level-Level Relationships

**Model Editor** 

**OLAP Dimension Dialog Box** 

**OLAP Tab (Model Editor)** 

## How to edit Analysis Services (formerly OLAP) measures

Information associated with a Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services measures, including database properties such as table and field and units, can be modified.

#### To edit Analysis Services measures

- 1. On the **OLAP** tab, expand **Cubes**.
- 2. Expand the cube that contains the measures to be edited and then expand **Measures**.
- 3. Double-click **Measures**.
- 4. Select the fact table and make the changes to **Dimensions**, **Measure**, **Aggregation**, **Field** and/or **Units**.

To delete a measure, select it by clicking in the gray column to the left of it and then, click **Delete**.

5. When satisfed with the changes, click **OK**.

### **See Also**

**Analysis Services in English Query** 

**Enabling Analysis Services for an English Query Model** 

How to enable Analysis Services (formerly OLAP)

**Model Editor** 

**OLAP Cube Dialog Box** 

# How to edit Analysis Services (formerly OLAP) properties

The properties associated with each <u>level</u> can be edited to provide additional information and allow the questions to be modified and refined.

#### **To edit Analysis Services properties**

- 1. On the **OLAP** tab, expand **Dimensions**, and then expand the dimension whose property is to be edited.
- 2. Expand the level containing the property to edit.
- 3. Right-click the property, and then click **Edit**.
- 4. Make the changes to the level, table, fields, or level properties.
- 5. When satisfied with the changes, click **OK**.

#### See Also

**Analysis Services in English Query** 

**Enabling Analysis Services for an English Query Model** 

How to enable Analysis Services (formerly OLAP)

**Model Editor** 

**OLAP Dimension Dialog Box** 

## How to enable Analysis Services (formerly OLAP)

If there are questions that cannot be answered using only the SQL database or if it is desirable to speed retrieval of data for questions that are not about aggregation of data, create a mixed model by enabling Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services and importing the appropriate cube.

### To enable Analysis Services

- 1. From the **Project** menu, choose **<project name> Properties**.
  - If the **<project name> Properties** command does not appear on the **Project** menu, double click **<project name>.eqm** in the Project Explorer window.
- 2. On the **Data Connection** tab, select **Enable OLAP**.
- 3. Enter the path to the database or click **Change** and select the path.
- 4. The **Type** and **Version** should automatically appear. If not, select **MSOLAP** for **Type** and enter the version number.

#### See Also

**Analysis Services in English Query** 

Data Connection Tab (Project Properties Dialog Box)

**Project Properties Dialog Box** 

## How to manually map OLAP objects to the underlying SQL database objects

When English Query is unable to acquire the information from a Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services server to map an OLAP cube to its underlying SQL table while creating a mixed model, the mapping information must be obtained manually or the project will not build properly.

#### To manually map OLAP objects to the underlying SQL database objects

- 1. In the Project Explorer window, double-click the module (.eqm).
- 2. On the **Model** menu, click **Import Tables**.
- 3. On the **Provider** tab of the **Data Link Properties** dialog box, double click **Microsoft OLE DB Provider for SQL Server**.
- 4. On the **Connection** tab, select or enter the name of the server on which the SQL database resides.
- 5. Enter the log on information.
- 6. Select the database containing the SQL database objects.
- 7. Click **OK**.
- 8. In the **New Database Tables and Views** dialog box, select all of the tables used by cubes and dimensions that are being mapped.
- 9. Click the **OLAP** tab and expand **Cubes**.

- 10. Double-click each cube to map.
- 11. In the **OLAP Cube** dialog box, select the appropriate table and field for each measure that needs to be mapped and then, click **OK**.
- 12. Repeat steps 10 and 11 until all of the cubes are mapped.
- 13. Expand **Dimensions**.
- 14. Double-click each dimension that to map.
- 15. In the **OLAP Dimension** dialog box, select the appropriate table and field for each level and member property and then, click **OK**.
- 16. Repeat steps 14 and 15 for each dimension to map.

**Note** All OLAP objects except calculated measures must be mapped to a SQL database object.

## How to remove an Analysis Services (formerly OLAP) cube

If an OLAP cube is no longer appropriate for a model, remove it from a project.

#### To remove an OLAP cube

- 1. On the **OLAP** tab, expand **Cubes**.
- 2. Right-click the cube to delete, and then click **Delete**.
- 3. When asked to confirm the deletion, click **OK**.

### See Also

**Analysis Services in English Query** 

**Enabling Analysis Services for an English Query Model** 

**Model Editor** 

**OLAP Tab (Model Editor)** 

## **Testing and Building a Model**

You test a <u>model</u> by asking questions that users of your application are likely to ask.

When you click **Save Query** while testing a model, you add the current question to the first regression file under the project in the Project Explorer window. There can be more than one regression text file in a project. A regression file can be added and then compared with the original file if the files have names that differ from one another.

After testing the model, build the application. The resulting file is an .eqd file, which can be made available to users inside any COM-supporting application, such as the Active Server Pages (ASP), Microsoft® Visual Basic®, or Microsoft Visual C++® sample applications that come with English Query.

**Note** When using <u>Question Builder</u>if it is desired to have the data values to appear in the entities' list in the **Relationship** pane, select **Sample Data** on the **Data Connections** tab before compiling the application.

#### See Also

Data Connection Tab (Project Properties Dialog Box)

How to set the Sample Data option

Sample Applications and Client Interfaces

## How to add a new or existing regression test file to a project

New or existing regression test files (.eqr) can be added to a project. After adding the regression test file, it is possible to view and edit the file and promote it to the new regression test file. The regression output is displayed in Extensible Markup Language (XML)-tagged format, which differentiates the questions, restatements, answers, and SQL statements for easier editing.

#### To add a regression file to the project

- 1. In the Project Explorer window, right-click the project to which the regression test file is to be added, and then click **Add Regression Test**.
- 2. In the **Add Item** dialog box, select **Regression Test** for a new file or, on the **Existing** tab, choose an existing file.
- 3. In the **Name** box, enter a name for the file or leave the default name. The default file name for a new file is Regression(*number*).eqr (for example, Regression1.eqr). Rename the regression test file if desired.
- 4. Click **Open**.

**Note** When a text (.txt) file, containing one question per line, is added as a regression test file to the project, English Query automatically converts it into regression test file format.

#### See Also

How to edit a regression test file

How to remove a regression test file

How to rename a regression test file

How to save model test information to a regression test file

## How to build an application

After the model is tested, it can be built into an application and then deployed.

### To build an application

- 1. From the **Project** menu, select **Project Properties**.
- 2. Select the **Data Connection** tab.
- 3. Select **Sample Data**.

**Note** For the data values to appear in the **Entity** list in **Question Builder**, select **Sample Data** on the **Data Connections** tab before compiling the application. Loading sample data may take some time.

- 4. Click **OK**.
- 5. In the Project Explorer window, right-click the project that need to be built into an application, and click **Build** project name>.

#### See Also

How to deploy an application to the Web

## How to edit a regression test file

The regression file can be edited in an editor that displays the questions in English Query Regression file (.eqr) format.

### To edit a regression file

- 1. In the Project Explorer window, double-click the regression test file to edit.
- 2. In the editor window, make the desired changes.
- 3. Save the file using the **Save <file name>** on the **File** menu.

### See Also

How to save model test information to a regression test file

## How to remove a regression test file

A regression test can be removed from the project when it is no longer needed.

### To remove a regression test file

- 1. In the Project Explorer window, right-click the regression test file to delete, and click **Remove <file name>**.
- 2. Click **Yes** to remove the file.

### See Also

How to add a new or existing regression test file to a project

## How to rename a regression test file

The default regression test file name is the *<name of a model>*.eqr or regression *<version number*, 1, 2>.eqr. The default name or the name of any regression test file can be changed at any time.

### To rename a regression test file

- 1. In the Project Explorer window, right-click the regression test file whose name is to be changed, and click **Rename**.
- 2. Enter the new name and press ENTER.

### See Also

How to add a new or existing regression test file to a project

## How to save model test information to a regression test file

English Query creates an empty regression file (.eqr) when a project is created. The model test information can be saved to it or to another regression file. Later the file can be compared with the current regression file.

#### To save model test information to a regression file

- 1. In the Model Test window, test a question.
- 2. Click **Save Query**.
- 3. Click **Save Query** after each question whose information is to be saved.

The results are saved in the current regression file.

#### See Also

How to add a new or existing regression test file to a project

How to edit a regression test file

How to remove a regression test file

How to rename a regression test file

How to view the output of a regression test created using the Run Regression command

**Model Test Window** 

## How to save the regression output to a regression test file

Copy the output file to a regression test file after verifying, with the **View Output** command, that the output is correct and should be used to determine whether the English Query model is functioning properly.

**Note** The output file must have been created using the **Run Regression** command, and it must be writable.

#### To save the output to a regression test file

- 1. Right-click on the regression test file in the Project Explorer window, and then click **Promote**.
- 2. Chose **OK** to replace the regression test file with the current output file.

### See Also

How to add a new or existing regression test file to a project Model Test Window

## How to set the Sample Data option

Set the **Sample Data** option before compiling an application that includes **Question Builder**. This will cause data values to appear in the entities' list in the **Relationship** pane.

### To set the Sample Data option

- 1. From the **Project** menu, select **Project Properties**.
- 2. Select the **Data Connection** tab.
- 3. Select **Sample Data**.
- 4. Click **OK**.

#### See Also

Data Connection Tab (Project Properties Dialog Box)

**Deploying an English Query Application** 

How to deploy an application to the Web

**Project Properties Dialog Box** 

### How to test a model

A model can be tested as many times as needed to create the application that lets users ask the questions they want.

#### To test a model

- 1. In the Project Explorer window, select the project to test.
- 2. On the **Debug** menu, click **Start**.
- 3. In the **Query** box, enter a question to be tested, and then click the **Submit Query** button.
- 4. If the restatement correctly reflects the question asked, execute the equivalent SQL statement by clicking the **View Results** button.
- 5. If the answer is correct, click **Save Query** to add the question to the regression test file.
- 6. If the question does not produce the correct answer, click the **Suggest Relationships** button to create additional relationships with the help of the Suggestion Wizard.
- 7. When testing and adding questions to the question file, click the **Close** box in the upper right-hand corner.

#### See Also

**Model Test Window** 

# How to view the differences between the current regression test file and the regression output file

The differences between the current regression file and the regression output file can be displayed and reviewed. Although the files cannot be edited, they can be searched for a specific word or phrase, and a comparison of the differences in the files can be made.

## To view the differences between current regression test file and the regression output file

- 1. Test the model and click **Save Query**.
- 2. Do this for as many questions as desired.
- 3. In the Project Explorer window, right-click on the current regression file.
- 4. Click **View Differences**.
- 5. In the **Regression Differences** dialog box, review the differences.

**Note** Deleted lines appear in blue text, changed lines in red text, and inserted lines in green text.

6. After reviewing the files, click the **Close** box in the upper right corner.

### See Also

How to add a new or existing regression test file to a project

How to edit a regression test file

How to save model test information to a regression test file

Model Test Window

### Regression Differences Dialog Box

# How to view the output of a regression test created using the Run Regression command

The results of a regression test can be viewed by using the **Run Regression** command. The output displays the restatements, SQL commands, and, if **Execute SQL** is selected, the database results. The output file can be edited but the changes do not persist if another regression test is run. To keep the changes, use the **Promote** command, which copies the output file over the original regression test file.

### To view an output file

• In the Project Explorer window, expand the project, right-click the regression test file (.eqr), and then click **View Output**.

**Note** The **Run Regression** command must have been chosen on the regression test file shortcut menu or the **View Output** command is not available.

#### See Also

How to edit a regression test file

How to save model test information to a regression test file

**Model Test Window** 

### How to deploy an application to the Web

An application can be easily deployed to the Web. English Query uses the Web Project Wizard in Microsoft® Visual InterDev® to deploy English Query applications to the Web.

### To deploy an application to the Web

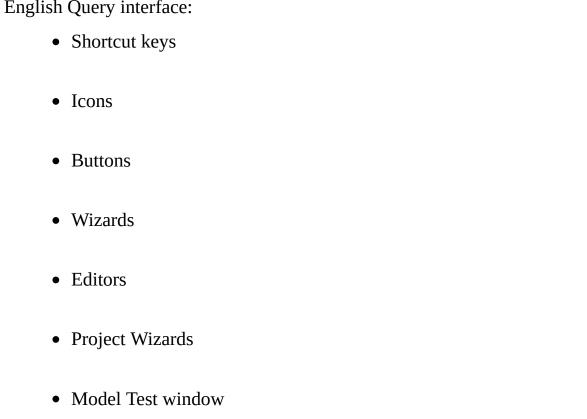
- 1. From the **Project** menu, choose **Deploy**, and then **Web**.
  - The Web Project Wizard appears.
- 2. Follow the steps of the Web Project Wizard.

**Note** It is necessary to know the name of the server to which the application is being deployed. Also the person who deploys the project must have permission to write to the root of the Web server and must be added as an operator for that server.

**Note** It is necessary to have Visual InterDev installed to create Web projects. Documentation for the Web Project Wizard is not available unless Visual InterDev Help or the MSDN® Library has been installed.

## **English Query User Interface Help Reference**

The topics in this section provide a quick reference for these elements in the English Query interface:



• Dialog boxes

## **Shortcut Keys Used in English Query**

This topic describes the shortcut keys used in English Query.

### **Global Keys**

Use these shortcut keys in all English Query windows.

Shortcut keys	Description
CTRL+N	Opens a new project.
CTRL+O	Opens an existing project.
CTRL+SHIFT+N	Opens a new file.
CTRL+SHIFT+O	Opens an existing file.
CTRL+SHIFT+S	Saves all modules and the project.
F1	Displays Help.
LEFT ARROW	Selects the previous item in a combo box.
UP ARROW	
RIGHT ARROW	Selects the next item in a combo box.
DOWN ARROW	

For more information about shortcut keys, see the Microsoft® Visual Studio® documentation.

### **Menu Keys**

Use these keys to navigate the **Menu** bar.

Shortcut keys	Description
LEFT	Selects the next menu.
RIGHT	Selects the previous menu.
UP	Selects the previous item in a menu.
DOWN	Selects the next item in a menu or expand
	menu.
ESC	Clears menus.

### **Model Editor: Navigation Keys**

Use these keys to navigate the hierarchical panes and tabs of the Model Editor.

**Note** These shortcut keys do not work in the **Canvas** pane.

Shortcut keys	Description
CTRL+PAGE UP	Selects the next tab.
CTRL+PAGE DOWN	Selects the previous tab.
UP	Selects the previous item in the tree.
DOWN	Selects the next item in the tree.
RIGHT	Expands current item in the tree.
LEFT	Collapses current item in the tree if expanded.
	Otherwise, selects parent item in the tree.

### **Model Editor: Editing Keys**

Use these keys to edit objects in the hierarchical pane of the Model Editor.

Shortcut keys	Description
ENTER	Displays a dialog box in which you can edit
Double-click an entity or	the currently selected object in the tree.
relationship.	
CTRL+C	Copies the selected object or objects and
	places them on the Clipboard.
CTRL+V	Pastes the object or objects.
CTRL+X	Cuts the selected object or objects and place
	them on the Clipboard.
DELETE	Deletes the selected object or objects.
CTRL+Click	Selects multiple objects.

### **Canvas Pane Keys**

Use these shortcut keys in the **Canvas** pane.

Shortcut keys	Description
BACKSPACE	Cancels any selection.
TAB	Selects the next object.
SPACEBAR	Starts auto layout. Stops the layout process in animation mode.
DELETE	Removes the selected object from the Canvas pane.
UP ARROW PAGE UP	Moves the Canvas pane up.
DOWN ARROW PAGE DOWN	Moves the Canvas pane down.
LEFT ARROW HOME	Moves the Canvas pane to the left.
RIGHT ARROW	Moves the Canvas pane to the right.
END	
CTRL+A	Selects all of the objects in the Canvas pane.
CTRL+F	Displays the <b>Find</b> dialog box.

### **Project Wizard Keys**

Use these shortcut keys in the SQL Project and OLAP Project wizards.

Shortcut keys	Description
SPACEBAR	Selects or clears the check boxes.
TAB	Move the focus through the various elements.
	If the relationships are expanded, TAB also
	moves the focus among the elements.
ENTER	Runs the Project Wizard and closes the
	window. Expands any item with a plus sign
	(+) and collapses an item with a minus sign (-
	).

### **Model Test Window Keys**

Use these shortcut keys in the Model Test window.

Shortcut keys	Description
ENTER	Runs the query.
CTRL+BREAK	Stops the query.
CTRL+S	Saves the query in a regression file.
CTRL+R	Displays or hides the query results.
CTRL+W	Runs the Suggestion Wizard that helps define entities and relationships to answer a question
	that currently does not work.
CTRL+T	Displays the <b>Test Properties</b> dialog box
CTRL+PAGE UP	Selects the next tab in the Model Test
	window.
CTRL+PAGE DOWN	Selects the previous tab in the Model Test
	window.

### **Command Tab (Relationship/New Relationship Dialog Box)**

Use these shortcut keys on the **Command** tab of the **Relationship/New Relationship** dialog box.

Shortcut keys	Description
ALT+P	Moves the focus to the grid.
CTRL+SPACEBAR	Selects the current row.
DELETE	Deletes the selected row.
INSERT	Inserts a row above the selected row.
UP and DOWN ARROWS	Selects the preceding and following item.
SPACEBAR	Activates the cell and selects the item.
TAB	Moves the focus to the next element in the
	tab.

### **Default Relationships Dialog Box**

Use these shortcut keys in the **Default Relationships** dialog box.

**Note** You can make only contiguous selections using the keyboard. To make noncontiguous selections, press CTRL and click the rows you want to select.

Shortcut keys	Description
CTRL+SPACEBAR	Selects the current row.
DELETE	Deletes the selected row.
INSERT	Inserts a row above the selected row.
RIGHT, LEFT, UP, and	Selects the item to the right, left, up, and
DOWN ARROWS	down of the currently selected item.
SHIFT+DOWN ARROW	Selects the next row to make a contiguous
	selection.
SHIFT+UP ARROW	Selects the previous row to make a
	contiguous selection.
SPACEBAR	Activates the cell and selects the item.
TAB	Moves the focus to the next element in the
	tab.

### **Entity/New Entity Dialog Box**

Use these keys in the **Entity/New Entity** dialog box.

Shortcut keys	Description
SPACEBAR	When the <b>Words</b> list has the focus, displays the words so you can view or edit them.
	When editing the <b>Words</b> list, moves a selected item between <b>Selected Fields</b> and <b>Available Fields</b> .
	When the <b>Display Field(s)</b> or the <b>Fields(s)</b> list has the focus, displays the <b>Selected</b>

	<b>Fields</b> and <b>Available Fields</b> lists so that you can view and edit them.
	When editing the <b>Selected Fields</b> and <b>Available Fields</b> lists, moves the selected item from one list to the other.
Comma (,) or DOWN ARROW	Moves the pointer to the next line when entering data in the <b>Words</b> list.
UP and DOWN ARROWS	Selects the previous or next item in the <b>Words</b> , synonym, and <b>Display Field(s)</b> lists.
ENTER	Closes a list.

## **Regression Differences Dialog Box Keys**

Use these keys in the **Regression Differences** dialog box.

Shortcut keys	Description
ALT+F3	Displays the <b>Find</b> dialog box where you
	specify the text you are trying to find.
F3	In the active pane, finds and selects the next
	occurrence of the text specified in the <b>Find</b>
	<b>What</b> box of the <b>Find</b> dialog box.
SHIFT+F3	In the active pane, finds and selects the
	previous occurrence of the text that is
	specified in the <b>Find What</b> box of the <b>Find</b>
	dialog box.
SHIFT+F7	Finds and moves the to the beginning of
	the previous text that has differences.
F7	Finds and moves the to the beginning of
	the next text that has differences.

### **Table/New Table Dialog Box Keys**

Use these keys to make changes in the **Table/New Table** dialog box.

**Note** You can make only contiguous selections using the keyboard. To make noncontiguous selections, press CTRL and click the rows you want to select.

Shortcut keys	Description
CTRL+K	Makes the field of the selected row a primary
	key.
CTRL+SPACEBAR	Selects the current row.
DELETE	Deletes the selected row.
INSERT	Inserts a row above the selected row.
RIGHT, LEFT, UP, and	Selects the item to the right, left, up, and
DOWN ARROWS	down from the item in the selected box.
SHIFT+DOWN ARROW	Selects the next row so you can make a
	contiguous selection.
SHIFT+UP ARROW	Selects the previous row so you can make a
	contiguous selection.

### **List Keys**

Use these shortcut keys in any list. A list is any type of box that contains a list of items from which you can select.

Keys	Descriptions
UP and RIGHT ARROW	Selects the next item in the list. If there is
	more than one column, RIGHT ARROW
	selects the item in the next column. Not
	available in all lists.
DOWN and LEFT ARROW	Selects the previous item in the list. If there is
	more than one column, LEFT ARROW selects
	the item in the previous column. Not available
	in all lists.
CTRL+SPACEBAR	Opens the list and displays all of the items.

### **Editable List Keys**

Use these shortcut keys in any editable list. An editable list consists of a rectangular box in which you can type text. If the list already contains items, you can select that default text or delete it and type new text.

Keys	Description
SPACEBAR	Displays the list and selects the first empty cell.
	When a cell with an item has the focus, also selects the text so that it can be edited.
ENTER	Accepts the changes and closes the list.
Comma (,) or DOWN	Moves the cursor to the next line when editing
ARROW	data.
UP and DOWN ARROWS	Select the previous or next line in the list.

### **Select Box Keys**

Use these shortcut keys in any select box with **Selected** and **Available Fields** columns in which you move an item from one column to the other.

Keys	Description	
SPACEBAR	When the select box has the focus, displays	
	the <b>Selected Fields</b> and <b>Available Fields</b> list	
	so that you can view and edit the lists.	
	When editing the lists, moves the selected	
	item from the current list to the other list.	
ENTER	Accepts the selection and closes the list.	
RIGHT and LEFT	Select the item in the column to the right or	
ARROWS	the left of the selected item.	
UP and DOWN ARROWS	Select the item prior to or after the selected	
	item.	

### See Also

Add/Edit Entity Default Dialog Box

**Adjective Phrasing Dialog Box** 

**Command Phrasing Dialog Box** 

Data Connection Tab (Project Properties Dialog Box)

Database Tab (Relationship/New Relationship Dialog Box)

Default Relationships Tab (Project Properties Dialog Box)

**English Data Values Dialog Box** 

English Tab (Project Properties Dialog Box)

**Entity/New Entity Dialog Box** 

**Find Dialog Box** 

Name/ID Phrasing Dialog Box

New Dictionary Entry Dialog Box

OLAP Cube Dialog Box

**OLAP Dimension Dialog Box** 

OLAP Tab (Model Editor)

**Preposition Phrasing Dialog Box** 

**Regression Differences Dialog Box** 

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

Semantics Tab (Model Editor)

Semantics Tab (Relationship/New Relationship Dialog Box)

SQL Tab (Model Editor)

**Subset Phrasing Dialog Box** 

Table/New Table Dialog Box

**Test Properties Dialog Box** 

Trait Phrasing Dialog Box

Verb Phrasing Dialog Box

## **Icons and Cursors Used in English Query**

The table shows and describes the icons used in English Query.

Icon	Description	Location
Ţ	Move selected item(s) from <b>Available</b> list to <b>Selected</b> list.	Entity/New Entity dialog box
	Move selected item(s) from <b>Selected</b> list to <b>Available</b> list.	Entity/New Entity dialog box
	Cube	OLAP Project Wizard and the <b>OLAP</b> tab of the Model Editor
<b>1</b> €	Dimension	OLAP Project Wizard and the <b>OLAP</b> tab of the Model Editor
•	Entity	SQL Project and OLAP Project Wizards, and the Semantics tab of the Model Editor
	Fact table	OLAP Project Wizard and the <b>OLAP</b> tab of the Model Editor
	Level	OLAP Project Wizard and OLAP tab of the Model Editor
# <sub>\$</sub> ¥	Measure	OLAP Project Wizard and the <b>OLAP</b> tab of the Model Editor
<b>a</b>	Member property	OLAP Project Wizard and the <b>OLAP</b> tab of the Model Editor
••••	Multiple level	OLAP Project Wizard and the

		<b>OLAP</b> tab of the Model
		Editor
<u>(S</u> )	Phrasings	SQL Project and OLAP
		Project Wizards
<u>&amp;</u>	Relationship	SQL Project and OLAP
		Project Wizards, and the
		<b>Semantics</b> tab of the Model
		Editor
	Table	SQL Project Wizard and the
		SQL tab of the Model Editor
	Current regression text	Regression Differences
	with difference	dialog box

### **See Also**

**OLAP Project Wizard** 

Regression Differences Dialog Box

SQL Project Wizard

## **Buttons Used in English Query**

The table shows the buttons used in the English Query dialog boxes and windows.

Button	Button name	Location
	Synonyms	Entity/New Entity dialog box
	Cancel Query	Model Test window
	Save Query	Model Test window
<u> </u>	Submit Query	Model Test window
	Suggest Relationships	Model Test window
	Test Properties	Model Test window
	View Results	Model Test window
<<		New < Database Object > dialog
	left column.	box
>>		New < Database Object > dialog
	right column.	box
<	Move selected item(s) to	New < Database Object > dialog
	the left column.	box
>	Move selected item(s) to	New < Database Object > dialog
	the right column.	box
	Find	<b>Regression Differences</b> dialog
		box
	Find Next	<b>Regression Differences</b> dialog
		box
	Find Previous	<b>Regression Differences</b> dialog
		box
	Next Difference	<b>Regression Differences</b> dialog
		box
	Previous Difference	<b>Regression Differences</b> dialog
		box
8	Primary Key	Table/New Table dialog box

### Wizards

This section describes the following wizards.

- Project Wizards
  - SQL Project Wizard
  - OLAP Project Wizard
- Suggestion Wizard

### **Project Wizards**

Use one of the Project wizards, either the SQL Project Wizard or the OLAP Project Wizard, to automatically create a project and model. Depending on your choice, the wizard uses an SQL database or a Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services database to build the model. These Project wizards:

- Extract all or a subset of the database based on the tables or cubes selected in the New <Database Object> dialog box.
- Create and define the entities and relationships based on the database objects selected from the database.

### **Options**

#### Create

Include or remove an entity in the model and display the relationships that will be automatically created based on the database structure.

To display or hide the relationships that are automatically created, click +.

### **Entity**

Display a list of potential entities and the relationships for each entity based on the <u>database objects</u> chosen in the **New <Database Object>** dialog box.

Select or clear a relationship to include it or remove it from the model.

### **Database Object**

Display the database object that was used to create the entity.

### See Also

Icons and Cursors Used in English Query

New < Database Object > Dialog Box

OLAP Project Wizard

### **SQL Project Wizard**

### **OLAP Project Wizard**

Use the OLAP Project Wizard to automatically create entities and relationships for all <u>OLAP</u> objects in the <u>cubes</u> selected in the **New OLAP Cubes** dialog box.

The OLAP Project Wizard suggests a set of entities and relationships for all dimensions, levels, properties, and measures and the fact table for the cube.

**Note** If the same OLAP level appears in more than one dimension or hierarchy, or if more than one OLAP level uses the same database table and fields, entities will not be created for subsequent levels using the same table or fields.

When creating a project using the OLAP Project Wizard, you enable OLAP only. Therefore, questions generate only <u>Multidimensional Expressions (MDX)</u>, not SQL statements. Also, some dialog box options will not be available. For example, you cannot create relationships that associate entities with fields so there is no **Database** tab available in the **Relationship/New Relationship** dialog box.

However, after the project is created, you can enable SQL using the **Data Connection** tab of the **Project Properties** dialog box. The resultant mixed model uses data from the OLAP cube or the SQL database tables and fields that underlie the OLAP objects. Enable SQL when questions:

- Cannot be answered using OLAP cubes only because the data is not in the cube but in the underlying SQL database.
  - For example, "What is the SKU of the product that sold the most" produces no results using only OLAP cubes because the SKU data is not in an OLAP cube. It is in the underlying SQL database.
- Are not about the aggregation of data.

Although using OLAP cubes allows the application to answer this type of question, OLAP performance may be slower than using SQL tables. In a mixed model, English Query uses the data that produces the results in the quickest manner. For example, for questions such as "What products do people buy," enabling SQL may speed up retrieval of the data. In mixed models, English Query initially tries to answer the

question using MDX. If MDX is insufficient, English Query will use SQL statements.

When using mixed models, make sure that the OLAP cube is current with the underlying SQL database. Each time that a question is asked, English Query creates a new query using the appropriate database. If the OLAP cube is not current, the user may get different results with an MDX query than with an SQL query.

### See Also

Data Connection Tab (Project Properties Dialog Box)

Database Tab (Relationship/New Relationship Dialog Box)

**Developing and Deploying English Query Applications** 

How to add or change an Analysis Services (formerly OLAP) connection

How to create a project using the OLAP Project wizard

**Icons and Cursors Used in English Query** 

Naming Entities in the OLAP Project Wizard

New < Database Object > Dialog Box

### **SQL Project Wizard**

Use the SQL Project Wizard to automatically create entities and relationships for all <u>database objects</u> in the database tables selected in the **New Database Tables and Views** dialog box.

The SQL Project wizard suggests a set of entities and relationships from those available in the database and based on the tables selected. Tables that are exclusively join tables are not included in the list of potential entities.

When creating a project using the SQL Project Wizard, tables but not views are imported. Although most views are used for reporting purposes only, any views that are used for an English Query model can be imported after a project is created.

Proposed entities are grouped by the name of the table.

### See Also

Creating an English Query Model

How to add or change a SQL connection

How to add tables and views from a SQL data source

**Icons and Cursors Used in English Query** 

New < Database Object > Dialog Box

### **Suggestion Wizard**

Use the Suggestion Wizard to create new entities and relationships for questions that cannot be answered using the current model and database structure. The Suggestion Wizard is displayed when you click the in the Model Test window.

The suggestions are displayed as sentences in which you fill in the blanks by either selecting or entering the appropriate information. You also can create a new entity by clicking the **Entity Icon** (•) to display the **Entity/New Entity** dialog box when it is available.

**Note** Any relationships created by the Suggestion Wizard that involve self-joins need a join path created after closing the Suggestion Wizard and before building the model to avoid errors when building the model. Create the join path in the **Role Name>Role** dialog box for one of the entities the relationship.

### See Also

**Entity/New Entity Dialog Box** 

How to create relationships from the Model Test Window

< Role Name > Role Dialog Box

Shortcut Keys Used in English Query

Testing an English Query Model

### **Editors**

This section describes the user interface for the following two editors:

- Model Editor
- Regression Test Editor

# **Model Editor**

Use Model Editor to easily create an English Query model. It consists of the following three tabs:

- OLAP Tab (Model Editor)
- Semantics Tab (Model Editor)
- SQL Tab (Model Editor)

# **OLAP Tab (Model Editor)**

Use the **OLAP** tab to display the <u>OLAP</u> objects in the <u>project</u>, including <u>cubes</u>, <u>levels</u>, <u>dimensions</u>, <u>measures</u>, <u>properties</u> and <u>facts</u>.

To see definitions of existing cube objects, double-click them.

Right-click a cube, dimension, level, property, or measure to display the **OLAP Dimension** or **OLAP Cube** dialog box.

#### See Also

**OLAP Cube Dialog Box** 

**OLAP Dimension Dialog Box** 

Semantics Tab (Model Editor)

**SQL** Tab (Model Editor)

# **Semantics Tab (Model Editor)**

Use the **Semantics** tab to display the semantic objects and to graphically create relationships.

The **Semantics** tab consists of two panes:

• The left pane contains a hierarchical list of the semantic objects in the selected module, including entities, relationships, and <u>dictionary entries</u>.

To see definitions of existing semantic objects, double-click the object. To create a semantic object, select one of the **Add** commands (**Entity**, **Relationship**, or **Dictionary Entry**) from the **Model** menu.

Canvas Pane

The right pane is where you graphically create and view relationships. Drag the entities from the left pane to the Canvas pane to create relationships among them. Drag relationships from the left pane to the Canvas pane to add or modify a relationship in the model.

Entities are displayed in a rectangle containing the name of the entity and relationships are displayed as an oval containing a description of the relationship.

# **Entities and Relationships**

Drag entities and relationships from the left pane to the Canvas pane to accomplish the following:

- Addition of a relationship and entities that are part of it to your model.
   Drag a relationship from the left pane to the Canvas pane to add the
  - Drag a relationship from the left pane to the Canvas pane to add the relationship and any corresponding entities to the diagram.
- Addition of an entity to an existing relationship.

Drag one or more selected entities onto a relationship that is currently part of the relationship to add entities to the existing relationship.

• Creation or modification of a relationship between two entities.

Drag an entity onto another entity in the Canvas pane to create the relationship rectangle.

• Creation of relationships among multiple entities.

Create relationships among multiple entities by pressing SHIFT and clicking the desired entities. Then drag the entities to another entity, or right-click one of the entities on the Canvas pane and click **Relate**.

• Modification of an entity or its relationship.

Display and modify an existing entity or its relationship. If you drag an entity to the Canvas pane without dragging it onto another entity, all relationships between that entity and all other entities currently on the Canvas pane are displayed. Double-clicking the entity displays the **Entity/New Entity** dialog box, where the entity can be modified and double-clicking a relationship displays the **Relationship/New Relationship** dialog box. There, the relationship can be edited.

• Repositioning of entities in the Canvas pane.

Move entities to another position on the Canvas pane by dragging them to the new location. Repositioning an entity anchors it in the new location.

• Removal of entities from the diagram.

Remove entities from the Canvas pane by dragging them from the diagram. This action does not delete them from your model.

#### **Shortcut Menu Commands**

#### **Canvas Pane**

These commands are available only when you click on the Canvas pane and affect the objects within the Canvas pane.

#### **Select All**

Select all of the entities and relationships in the Canvas pane.

#### Add Relationship

Display the **New Relationship** dialog box, where you add a relationship to your model.

#### Float All

Set all of the entities and relationships to automatically adjust their positions on the Canvas pane when a new object is added. When an entity or relationship floats, it appears with a shadow around it.

#### Anchor All

Set all of the entities and relationships to remain in position on the Canvas pane a new object is added. Anchored entities and relationships do not have a shadow.

#### Zoom In/Zoom Out

Change the view of the entities and relationships on the Canvas Pane.

#### Clear

Clear all of the objects from the Canvas pane.

### **Animate Layout**

Gradually position the entities and relationships when you add entities to the model.

# Help

Display Help for the **Semantics** tab.

### **Entities and Relationships**

These commands are available only when you right-click an entity or relationship in the Canvas pane.

# **Add Relationship**

Display the **New Relationship** dialog box, where you add a relationship to your model. Available only for entities.

## **Explode**

Add all of the relationships associated with the selected entity to the Canvas pane. All necessary entities associated with the relationships are

also added. Available only for entities.

#### **Implode**

Remove the relationships from the diagram associated with the selected entity. Available only for entities.

#### **Float**

Set the selected relationship to allow its position to be automatically adjusted on the Canvas pane when a new object is added. A relationship in float mode has a shadow around it. Available only for relationships.

#### Anchor

Set the selected entity to remain in position on the Canvas pane when a new object is added. Available only for entities.

#### **Remove from Diagram**

Remove the selected entities or relationships from the Canvas pane.

**Note** Removing an entity or relationship does not delete it from the model.

### Help

Display Help for the **Semantics** tab.

## **See Also**

**Entity/New Entity Dialog Box** 

**OLAP Tab (Model Editor)** 

Relationship/New Relationship Dialog Box

# **SQL Tab (Model Editor)**

Use the **SQL** tab to view the database schema of the <u>SQL database</u> used with the selected module.

After you open a module, select the **Tables** tree to view a list of tables, fields, and joins in the SQL database.

To see how an object has been defined in the project, double-click it.

## See Also

**Entity/New Entity Dialog Box** 

Table/New Table Dialog Box

# **Regression Test Editor**

Use the Regression Test editor to display the regression file and test your model by trying old and new questions against English Query.

Right-click on the .eqr file in the Project Explorer window to use the following commands.

#### Open

Display an editor with the original English question, the restatement, the answer, and the SQL statements.

#### **Run Regression**

Execute all of the questions available in the regression file and place the questions, their restatements, answers, and database query statements into a temporary regression output file.

An output file is a temporary file that results from running a regression. It is identical in format to a regression test file. When an output file is saved, it becomes a regression test file that can be used with multiple versions of the model.

Click **View Output** to view and edit the temporary output file. When you choose **Promote**, the contents of the temporary output file is saved as a new regression file.

A list of questions in a plain text file can also be run using this command.

# **View Output**

Display the results of a regression test run that was performed after you clicked **Run Regression**.

The results include the restatements, answers, SQL commands, and database results in Extensible Markup Language (XML) format. To edit the file and the changes, click **View Output** to open the editor before you click **Run** or **Execute SQL** again. If you click **Promote**, the changes appear in the new regression test file.

#### **View Differences**

Display the differences between the selected regression file and the output from the last time you clicked **Run Regression** on this regression file.

Use the **View Differences** command to determine whether the model is still functioning properly.

#### **Promote**

Copy the contents of the temporary output file to the original regression test file.

Click **Promote** to determine whether the English Query model is functioning properly. Do this after verifying the output and results displayed using the **View Output** command are correct.

This command is only available if you have previously selected the **Run Regression** command.

**Note** The .eqr file must be writeable. If you are working in a project that is using source control, check the file out before using this command.

#### See Also

How to add a new or existing regression test file to a project Testing and Building a Model

## **Model Test Window**

#### Use the **Model Test Window** to:

- Ask a question or state a command.
- Check the response from English Query.
- Execute the query to view the results from the database.

# **Options**

## Query

Evaluate the model by typing a test question and clicking **Submit Query**.

English Query returns a restated version of the question and the generated SQL query.

# **Save Query Button**

Save the question, answer, and any clarifications.

# **Submit Query Button**

Submit the question to English Query.

# **Cancel Query Button**

Cancel the database query that is being executed.

#### **View Results Button**

View the results of the database query.

**Note** This requires a data connection.

# **Suggest Relationships Button**

Display a suggestion for the necessary entities and relationships to make a question that cannot be understood or has been answered incorrectly work.

#### **Test Properties Button**

Display the **Test Properties** dialog box, where you set the amount of time to wait before a query that has not been completed stops, and the maximum number of rows displayed per question.

#### **Restatement pane**

View a restatement of the query in English.

#### **Answer Tab**

View a response to the query.

#### • Statement pane

View the query as a statement.

#### • SQL pane

View the SQL statements for the query.

#### • Results pane

View the query results. This option is available only if you clicked **View Results**.

# **Analysis Tab**

View the entities and phrasing used in the question. This tab appears when a query is submitted.

If the question was interpreted, you will see a description of the entities and phrasings.

#### GoTo

Display the **Entity/New Entity** or **Relationship/New Relationship** dialog box for the selected entity or relationship where you can edit it. Only available when an entity is selected.

#### **Command Tab**

Display the command structure for the command. This tab appears when a command relationship is submitted.

## **Table Tab**

Display the database query and result table for each command relationship parameter. This tab appears when a command relationship is submitted.

# **See Also**

**Buttons Used in English Query** 

**Suggestion Wizard** 

Test Properties Dialog Box

# **Dialog Boxes**

This section describes topics that are available from the user interface. For more information about Help topics, see <u>Project Wizards</u>, <u>Model Editor</u>, <u>Regression Test Editor</u>, and <u>Model Test Window</u>.

# **Add/Edit Entity Default Dialog Box**

Use this dialog box to specify a new default entity or to edit an existing default entity.

# **Options**

#### Relationship

Specify a relationship in which the entity participates as the relationship for the default.

#### Invert

Specify that the default condition is the opposite of the condition expressed in the selected relationship.

#### **Conditions**

Optional. Specify conditions on the other entities in the relationship. A condition consists of the following elements:

## Entity

Select the entity, other than the current entity, for which you want to set a condition. In the example, status equals current, the entity is status.

# Operator

Choose the operator from the list. Greater than and less than are not available for text entities. In the example, status equals current, the operator is equals.

#### Value

Enter the value for the condition. The value may be numeric or text. In the example, status equals current, the value is current.

When the entity is a date the following values are accepted:

Dates in the form MM/DD/[YY]YY

Times in the form HH:MM[:SS]

Datetimes in the form MM/DD/[YY] HH:MM:[:SS]

Months in the form MM/[YY]YY

Years [YY]YY

Special text such as "today", "this month", and "this year."

**Note** Because there is no "is null" or "is not null" available as a condition, it may not be obvious how to get defaults like:

employees whose ages are not null or employees whose termination dates are null.

However, they are not needed because creating the default *employees have ages* on the entity **employees** gives the behavior the user expects. The question, "List the employees," shows only the employees who have ages.

Similarly, creating the default employees have termination dates on the entity **employees** and setting the **Invert** on the default causes the question, "List the employees," to show only employees who do not have a termination date.

# Default conditions invalid when these relationships are used

Specify the relationships in which this entity participates and for which you do not want to use default conditions if the relationship is used in the question.

For example, if you specified *employees who have termination dates* relationship, the default relationship *employees who are active* would not be used if a user asks, "Which employees have terminated?"

# **Adjective Phrasing Dialog Box**

Use this dialog box to specify adjective phrasing for a relationship.

When a relationship between entities is expressed with an adjective, use adjective phrasing to define the relationship. The adjective can be:

- A single word not represented by a field.
- Part of a <u>database object</u> that represents an entity.
- A measurement.

# **Options**

#### **Subject**

Select an entity that represents the subject of the relationship.

# Adjective type

Select the adjective type.

### Single adjective

Select if the adjective is a word not represented by an entity. Then enter the word in the **Adjective that describes subject** box.

# **Entity contains adjectives**

Select if the adjective is contained in a field. Then enter the entity in the **Entity that contains adjective** box.

#### Measurement

Select if the adjective is a measurement of some kind, such as tall, fast, or heavy. Then enter the entity that contains it in the **Entity that contains the measurement** box.

# Adjective that describes subject

Enter an adjective that describes the subject entity in the relationship. This word or words is not represented by an entity. Available only if you click **Single Adjective. Entity that contains adjectives** 

Select an entity, which represents a table or field, containing the adjectives in the relationship.

Available only if you click **Entity contains Adjectives**.

#### **Associate English Values**

Display the **English Data Values** dialog box, where you specify English data values.

Available only if you click **Entity contains adjectives**.

#### **Entity that contains the measurement**

Select an entity containing the measurement for the relationship.

Available only when you click **Measurement**.

## Adjectives associated with higher values (Use root form)

Enter adjectives that describe the high value ranges in their <u>root form</u>. For example, in the question, "List the old salespeople," *old* is the adjective associated with a higher value.

## Numeric threshold for high values

Specify the highest value to use when users ask questions involving absolute, but subjective, measurements. For example, the question, "List the old salespeople," requires specifying what age is considered old. Only numeric values can be specified. If a threshold is not specified, English Query prompts the user for the information when the question is asked.

#### Use lower value words as the default

Select to define the lower values as the default in the <u>restatement</u> of the question.

For example, identify whether the lower value (for example, young) should be seen as the top, or higher value. The use of the lower value in this example allows questions such as "Show the people and their ages" to be restated as "Show the people and how young they are." This restatement is

not as likely as the idiom, "how old they are."

#### Adjectives associated with lower values (Use root form)

Enter adjectives that describe the lower values in their root form, for example, *young*.

#### Numeric threshold for low values

Specify the lowest value to use when users ask questions involving absolute, but subjective, measurements. For example, the question "List the young salespeople" requires you to specify what age is considered young. Only numeric values can be specified. If a threshold is not specified, English Query prompts the user for the information when the question is asked.

#### Add prepositional phrase

Enter the preposition and select an object from the **Object of preposition** list to further describe the relationship.

#### See Also

Adjective Phrasings in Relationships

# **Advanced Entity Properties Dialog Box**

Use this dialog box to refine the entities in your model. This dialog box has the following three tabs:

- Entity Defaults Tab (Advanced Entity Properties Dialog Box)
- Name Synonyms Tab (Advanced Entity Properties Dialog Box)
- Semantic Properties Tab (Advanced Entity Properties Dialog Box)

#### See Also

**Defining Entities in an English Query Model** 

# Entity Defaults Tab (Advanced Entity Properties Dialog Box)

Use this tab to specify default conditions for an entity.

Whenever an entity is used in a query, all appropriate defaults are attached to it. For example, if the entity, **customer**, has a default indicating that the customer should be a current customer, then the question, "Show the customers" is interpreted as "Show the current customers."

# **Options**

#### **Defaults for**

View the current default conditions.

#### Add

Click to display the **Add/Edit Entity Default** dialog box, where you start a new default condition.

#### **Edit/Delete**

Modify or remove the selected default condition.

#### **See Also**

**Advanced Entity Properties Dialog Box** 

**Defining Entities in an English Query Model** 

Name Synonyms Tab (Advanced Entity Properties Dialog Box)

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

# Name Synonyms Tab (Advanced Entity Properties Dialog Box)

Use this tab to specify a synonym for a particular instance of an entity and add the specified names to the model as if they were added from the database.

This is useful for situations in which certain names you want to be recognized are not present in the database or are not stored in a form that users expect. For example, a product name may be stored in the database as "chair, wicker," but users may typically refer to it as a "wicker chair." Similarly, "Big Apple" may appear in the database as "New York."

# **Options**

#### This word

Specify the synonym for the term that appears in the database. For example, specify "Bill" as a synonym for "William."

### Appears in database as

Specify the name that you want specified as a synonym. For example, the name "William" is the name stored in the application database for the synonym "Bill."

#### See Also

**Advanced Entity Properties Dialog Box** 

**Defining Entities in an English Query Model** 

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

# Semantic Properties Tab (Advanced Entity Properties Dialog Box)

Use this tab to define the entities in a model.

# **Options**

#### Is a subentity of

Specify an entity from which this entity inherits relationships. Available only if a join exists from the key of this entity's table to the key of the entity table from which the relationships are inherited. For example, click this option to have a **salesmanagers** entity inherit all the relationships for the **salesperson** entity. This option ensures that relationships supporting questions such as "List the sales people and their sales last month" will work for "List the sales managers and their sales last month," without having to define the relationship explicitly between **salesmanagers** and **sales**.

#### Load words from

Specify a smaller table or a view from which to load data values into the dictionary for this entity. Available if you select the **Add values of this entity to the model** check box and if a table is available for selection.

For example, specify a **CurrentCustomers** view from which to load customer names to avoid loading the larger **Customers** table.

The **Load words from** list contains any table that has:

• A denormalized copy of each of the fields in the entity's field list.

Set these tables using the **Denormalized** option on the **Advanced** tab of the **Table/New Table** dialog box.

For example, the **Load words from** list for the **customer\_name** entity, which refers to the **fname** and **lname** fields of the **dbo.customer** table, contains the **dbo.currentCustomers** table. This occurs because **dbo.currentCustomers.fname** is a denormalized copy of

**dbo.Customers.fname** and **dbo.currenCustomers.lname** is a denormaized cop of **dbo.Customers.lname**.

• Fields with the same names as the fields in the entity's field list.

When the table is selected, the fields in the table are automatically marked as denormalized copies of the entity's fields. For example, the **Load words from** list for the **customer\_name** entity, which refers to the **fname** and **lname** fields of the **dbo.customer** table, also contains the **dbo.BestCustomers** table because the table contains the **fname** and **lname** fields although they currently are not marked as denormalized. However, the fields will be marked as denormalized when the table is selected.

#### Unknown dates refer to

Specify a relationship that contains a date so that ambiguous questions can be answered correctly. Available only if this entity participates in more than one relationship that involves a date. For example, the question "List the 2000 customers" can be interpreted as "Show the customers who bought products in 2000" or "Show the customers that returned products in 2000." To clarify the interpretation, specify the appropriate relationship. Otherwise, English Query guesses the meaning of the question.

#### Numerical references are to

Specify a field that contains the intended numerical data in an entity that is represented by a table. For example, the **sales** entity, which is represented by a sales table, could be used for questions such as "Show the sales for last week." To allow numeric questions such as "List the sales greater than \$50" or "Show the total sales for last week," specify which field in the sales table should be used for questions that involve numeric operations like this.

# This is a standalone entity

Display data by itself, (without the accompanying data from a related table entity), when returning an answer.

Available only if this entity is represented by fields and does not participate in a name relationship. For example, to display employees' ages only, you'd specify this option for the age entity. Questions such as "What are employees' ages?" returns a list of ages, without employee names.

#### Sort by

Specify the sort order in which data from this entity is displayed.

#### Remote fields

Specify the fields from other tables to show whenever this entity is displayed. Available only when the entity is one requested by a table with joins to other tables.

The table associated with the entity must have a join to the table containing the remote field.

#### Add Field

Display the **Select Remote Fields** dialog box, where you select fields from other tables. Add a field by selecting the remote table and field you want displayed when the entity is displayed. To remove a field from the display list, select it in the **Selected** fields box and click DELETE.

**Note** The table associated with the entity must have a join to the table containing the remote field.

#### See Also

Add/Edit Entity Default Dialog Box

**Advanced Entity Properties Dialog Box** 

Advanced Tab (Table/New Table Dialog Box)

**Buttons Used in English Query** 

Entity Defaults Tab (Advanced Entity Properties Dialog Box)

**Entity/New Entity Dialog Box** 

How to add a join

Name Synonyms Tab (Advanced Entity Properties Dialog Box)

New Dictionary Entry Dialog Box

# Select Remote Fields Dialog Box

# **Command Phrasing Dialog Box**

Use this dialog box to specify a command phrasing for your command relationships.

Command phrasings are used to define commands that display a set of data or other commands such as "Show the paintings by xxx artist that are watercolors" or "Print the file."

# **Options**

#### **Sentence type**

Select the type of sentence structure:

#### Verb

An example is print or exit.

## • Verb Object

An example is *print file* or *order product*.

### • Verb Object Object

An example is *charge customers amounts for products*.

# **Examples of selected phrasing**

Display multiple examples of the selected sentence type.

#### Verb

Enter a verb that describes the action in the command relationship (for example, *sell*).

## **Indirect object**

Select an entity that contains an indirect object (for example, *teachers*).

# **Direct object**

Select an entity that contains a direct object (for example, *briefcases*).

# Add prepositional phrase

Enter the prepositions and the object of preposition that further describe the relationship (for example, *from the warehouses*).

# **See Also**

<u>Using Command Relationships in an English Query Model</u>
<a href="#">Verbs to Avoid in Command Relationships</a>

# **English Data Values Dialog Box**

Use this dialog box to specify English data values.

When the adjectives or <u>subset words</u> used to describe a relationship are defined in an entity, you can specify that they be found either in a database field or in a specified set of word-value pairs. For example, a lookup table can contain gender codes and the values they represent, M for Male and F for Female.

# **Options**

#### **Read English from table**

Indicate that the adjective or subset entity contains codes and adjectives or subset words that must be looked up in a lookup table.

#### **Table**

Select a table that contains the fields in which adjective or subset words are stored. An example is the **Gender Codes** table.

Available only if you select **Read English from table**.

#### **Word field**

Select the field that contains the descriptive words in the adjective or subset phrasing.

Available only if you select **Read English from table**.

#### Value field

Select the field that contains the codes in the adjective or subset phrasing.

Available only if you select **Read English from table**.

# **Specify values**

Specify a lookup table, which contains the values that serve as the adjectives or subset words in the relationship (for example, M, Male and F, Female). **Word** and **Value** 

Enter the word and its associated values. Although you can enter only one

entry per line, you can have multiple words or multiple values. For example, *Male* could be entered as the word, as the value, *M*, and as the value, *Y*. *Female* could be entered as the word, as the value *F*, and as the value, *X*. *Unknown* could be entered as *U*.

Available only if you select **Specify values**.

# **Entity/New Entity Dialog Box**

Use this dialog box to create or modify <u>entities</u>, which are <u>semantic objects</u>, for your <u>model</u>.

# **Options**

#### Words

Enter the words, phrases, or synonyms that identify the entity.

For example, for an entity represented by the **Employee** table, you can type: *employee*, then *worker*, and then *associate*. Each time you click **Words:**, you enter a synonym on a separate line at the bottom of the list. When you press ENTER, the synonyms are displayed in the **Words:** box with a comma separating each entry.

**Note** The words appear in their singular forms, not in the plural.

Click the browse (...) button to display a list of synonyms for all the words in the **Words** box.

# **Entity type**

Select a type from the drop-down list:

#### None

Default. Enable no additional language capabilities.

#### Who

Refer to the entity as "who" in a question. For example, indicating that customers are people lets "Who bought a lawnmower?" be interpreted as "Which customer bought a lawnmower?"

#### Where

Refer to the entity as "where" or as a "location." For example, using this option allows you to ask questions like "Show the buildings and their locations?".

#### When

Refer to the entity as "when" and allow it to be used in the **When**, **Start**, and **End** options of the **Semantics** tab of the **Relationship/New Relationship** dialog boxes.

#### Measure

Indicate that the entity is a measure quantifying something else. Using this option allows the entity to be used in the **How many** and **How much** options of the **<Role name>Role** dialog boxes.

#### Name Type

Specify the type of name for the entity.

Available only if you select **Field(s), OLAP levels**, or **OLAP properties**.

# No name type

Default. Indicate that this data does not represent a name or unique ID. Disables the **Add values of entity to model** option.

### **Proper Name**

Indicate the proper name of something. Proper names are capitalized and are not preceded by an article (*a*, *an*, or *the*) (for example, *Mary Smith*).

#### **Common Name**

Specify a general term for something. Common names are not capitalized and typically include an article: *a*, *an*, or *the* (for example, *a wrench*). An exception to the requirement for an article is "*an e-mail name*."

#### **Classifier Name**

Select a name of something commonly preceded by *a*, *an*, or *the*, and the entity word is included in the name. For example, *the Finance department* includes the name of the entity, *department*.

#### **Model Name**

Select a name that is capitalized and preceded by the word *a*, *an*, or *the* (for example, brand names such as *a Mustang*).

Note Do not include the name of the entity. For example, an entity is not

typically phrased a Mustang car.

### **Unique ID**

Specify that the entity is represented by a special type of name in the database that makes each entry unique (for example, Employee 37541).

### Add values of entity to model

Add data in the selected table or field to the model so references to instances of the entity can be recognized out of context. Depending on the number of distinct instances of the entity stored in the database, the process of loading these values can take several minutes.

Available with all **Name Type** options except **None**.

#### Attribute of

Indicate that the selected entity is an attribute of another entity. For example, name, phone number, and age are attributes of an employee.

The list contains all of the entities that are not already an attribute of another entity. The list is unavailable if the entity is associated with a table, dimension, or fact because such entities cannot be attributes of other entities. The selection is blank if the entity cannot be considered an attribute of another entity.

# **Help Text**

Specify Help about this entity.

Users view this information in the <u>Question Builder</u> or by asking questions such as "What do you know about customers?".

# **Entity associated with**

Specify that the entity is associated with:

#### **Table**

Show that the entity is represented by the rows in the table. **Table** and **Field(s)** are available when **Table** is selected.

# Field(s)

Indicate that the entity is represented by one or more fields. Table and

**Field(s)** are available when **Field(s)** is selected.

#### **OLAP**

Indicate the entity is associated with an <u>OLAP</u> object <u>Level</u> (default), <u>Dimension</u>, <u>Measure</u>, <u>Property</u>, or <u>Fact</u>.

The **OLAP object** list is unavailable until OLAP is selected. If the model is not OLAP enabled, these options are unavailable.

#### None

Indicate the entity is not associated with any database properties.

#### **Table**

Select the table you want associated with the entity. Available only if you select **Table** or **Field(s)**.

### Fields/Display Field(s)

Select the fields from the selected table that you want displayed. Available only you select **Table** or **Field(s)**.

The list is displayed in two columns, **Available Fields** and **Selected Fields**. To display a field, place the cursor over the field you want displayed and click. The field moves from the **Available Fields** list to the **Selected Fields** list.

To remove a field from the **Selected Fields** list, place your cursor over it and click. The field moves to the **Available Fields** list.

**Note** The cursor changes to a right- or left-pointing arrow depending on the column over which it is placed.

#### OLAP Cube

Display all of the OLAP <u>cubes</u> in the Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services database and select the OLAP cube to which you want to associate the entity.

Available only when you select an OLAP object.

#### Dimension

Display all of the dimensions in the selected cube and select the one for which you want to associate the entity.

Available only when you select an OLAP object (for example, **Level**, **Dimension**, or **Property**).

#### Level

Display all of the levels in the selected dimension, and select the one to which to associate the entity.

Available only when you select an OLAP object (for example, **Level** or **Property**).

### **Property**

Display all of the properties associated with the level and select the one to which to associate the entity.

Available only when you select the OLAP object, **Property**.

### **Show Relationships**

Display the **Show Relationships** dialog box with a list of relationships that involve this entity.

### **Create Relationships**

Display the **Relationship** wizard, which you use to add entities and create relationships.

#### **Advanced**

Display the **Advanced Entity Properties** dialog box, where you set semantic properties, defaults, and synonyms for the entity.

### See Also

Advanced Entity Properties Dialog Box

**Associating Entities** 

**Buttons Used in English Query** 

**Defining Entities in an English Query Model** 

**Icons and Cursors Used in English Query** 

<Role Name>Role Dialog Box

< Role Name > Role in Command Dialog Box

Semantics Tab (Relationship/New Relationship Dialog Box)

# **Find Dialog Box**

Use this dialog box to search for a specific string in the current file.

# **Options**

#### **Find What**

Enter the string you want to find.

#### Match case

Make the search case-sensitive.

### Match whole word only

Match the specified string only if the string is a whole word, not part of a word.

#### Use regular expression

Default. Use the \* and ? characters as wildcards.

For example, A?B matches ABB or AQB, and A\*B matches ADSFFDFDSB.

Clear this option to have the \* and ? characters interpreted literally. For example, A\*B matches only one string, A\*B.

#### Direction

Specify whether the search continues up or down in the file.

#### **Find Next**

Find and select the next occurrence of the specified text in the file.

### See Also

Regression Differences Dialog Box

# Join/New Join Dialog Box

Use this dialog box to specify tables involved in the join.

# **Options**

#### **Source Table**

Specify the table that contains the field (<u>foreign key</u>) from which the join originates.

#### **Destination Table**

Specify the table that contains the field (<u>primary key</u>) to which the join is made.

#### Join Conditions

Specify the fields that make up the primary and foreign keys in the join.

#### Add/Edit/Delete

Add, modify, or delete a join condition. Click **Add** or **Edit** to display the **Join Condition** dialog box, where you can select fields to be joined.

### Allow outer join

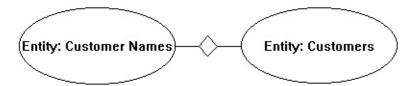
Specify an <u>outer join</u>, because all rows in the source table may not have matching rows in the destination table.

For example, the table **Employees** contains an employee ID, name, and building ID. The table **Buildings** contains a building ID and a building name. The question, "Show the employees and their buildings" usually generates a query with an outer join from **Employees** to **Buildings** so employees who are not assigned to buildings are also displayed in the result set. If all employees are assigned to buildings, clear this option to improve the performance of the query.

# **Name/ID Phrasing Dialog Box**

Use this dialog box to specify a name phrasing for your relationship.

When one entity is the name (or ID) of another entity, use <u>name phrasing</u> to describe the relationship between them.



When specifying the two entities in this dialog box, you provide information to answer the following types of questions:

- "Show me a customer named John Smith."
- "List all the employees by name."
- "What do you know about Mary Smith?"
- "Show last names of people named Tom."

# **Options**

# **Entity that is Name/ID**

Select the entity represented by a <u>database object</u> in which name or ID values are stored.

For example, in the phrasing, *Customer\_Names are the names of customers*, **customer\_name** is the entity.

# **Entity being named**

Select the entity named.

For example, in the phrasing, *Customer\_Names are the names of customers*, **customer** is the entity being named.

# **New < Database Object > Dialog Box**

Use this dialog box to select the new <u>database objects</u> you want to add to your model.

A database object can be a <u>table</u>, view, or OLAP <u>cube</u>. After choosing the database objects, use the Project Wizard to automatically add entities and relationships.

The title of this dialog box changes with the type of project that is being created or edited. In SQL projects, the name of this dialog box is the **New Database Tables and Views** dialog box. In OLAP projects, the name of the dialog box is **New OLAP Cubes** dialog box. In this Help file, it is referred to as the **New <Database Objects>** dialog box in the general sense, and the specific name otherwise.

# **Options**

#### **Available List**

Display a list of the objects in the database that can be added to a model.

#### **Selected List**

Display a list of the objects that you have chosen to add to your model.

#### > Button

Move the selected object(s) from the **Available** list to the **Selected** list.

To add multiple contiguous objects, select the first object, hold down SHIFT, and then click the last object. To select multiple noncontiguous objects, select the first object, hold down CTRL, and then click the additional objects.

#### >> Button

Move all of the objects from the **Available** list to the **Selected** list.

#### < Button

Move the selected object(s) from the **Selected** list to the **Available** list.

#### << Button

Move all of the objects from the **Selected** list to the **Available** list.

### OK

Import the database structure of the selected objects into the model.

# **See Also**

**Buttons Used in English Query** 

**OLAP Project Wizard** 

**Project Wizards** 

**SQL Project Wizard** 

# **New Dictionary Entry Dialog Box**

Use this dialog box to specify additional words to include in the English Query dictionary.

The English Query dictionary includes thousands of common English words, and provides your English Query application with the terminology it needs to answer most questions posed in English. Add new words to the dictionary if the application requires specialized terms that are not known by English Query.

Creating entities with <u>synonyms</u> and relationships will provide most of the vocabulary the system requires. Create a dictionary entry if the word being defined is not associated with a particular entity or relationship.

# **Options**

### Dictionary entry type

Specify an entry as a main word or a synonym for a word that already exists in the dictionary.

#### **Define word**

Define irregular forms of words that are not known to English Query. For example, the irregular plural form of person is people, and the irregular past tense form of shoot is shot.

# Read synonym

Indicate that a word be interpreted the same way as another word when a query is received from a user.

This option is global and overrides any synonyms set in the **Name Synonyms** tab. For example, if you set this option to read "ship" as "boat," the question "How many boats sailed last week?" would be interpreted as "How many ships sailed last week." But it would incorrectly interpret, "How many widgets did we ship last week?" as "How many widgets did we boat last week?" In this case, it is better to add the synonym "ship" to the primary entity word, "boat", in the **Entity/New Entity** dialog box.

# Write synonym

Indicate that an alternative word be used in the response to a query.

For example, click **Write: employees as: associates** to specify that if a question has been received about *employees*, the response can use the term *associates* instead.

### See Also

**Advanced Entity Properties Dialog Box** 

**Entity/New Entity Dialog Box** 

Name Synonyms Tab (Advanced Entity Properties Dialog Box)

# **OLAP Dimension Dialog Box**

Use this dialog box to display all levels in the <u>dimension</u>, including the underlying tables and fields associated with them. In addition, this dialog box displays additional properties of the level and any OLAP properties associated with it.

Selecting **Enable SQL** on the **Data Connection** tab of the **<Project > Properties** dialog box enables you to edit the **Table** and **Field** columns.

# **Options**

#### **Date dimension**

Indicate that the dimension is of a date and/or time type.

#### Level, Table, Field

Display the **Tables** and **Fields** underlying the specified <u>level</u>.

If a table or field cell contained **None** when you imported an OLAP <u>cube</u>, English Query was not able to access the fields that underlie the OLAP levels and SQL text could not be generated.

### Capitalization

Specify the case in which the data was entered (for example, all lowercase, initial caps, and so on). If you do not know how the data was entered or the data is inconsistently capitalized, select Unknown.

For all values of **Data Capitalization** other than **Unknown**:

Regardless of how the user types in the data value in the question, the SQL statement generated by English Query matches the data capitalization values. For example, if the user asks, "Who sold the most widgets?", and the data capitalization on the **product\_name** field is "UPPERCASE", then the SQL statement will contain "**product\_name** = 'WIDGET'". An exception to this can occur when the values of **product\_name** are loaded into the model. If there are any data values that violate the capitalization convention, these are memorized so that the correct capitalization will be used in the SQL

statement.

For data capitalization that is unknown:

The capitalization in the SQL statement generated by English Query matches the user's capitalization exactly, whether or not the data values have been loaded into the model. For example, if the user asks, "How many awards did Gone With the Wind win?", the SQL statement is movie\_name = 'Gone With the Wind'. When users are unlikely to know the correct capitalization to use, pick the closest data capitalization value other than **Unknown**, and load values of the entity into the model so that all capitalization is memorized by English Query.

# **Date Type**

Specify the date type of the data in the level, if any. Available only if the field representing the level is of **date**, **string**, **integer**, or **float** data type.

#### None

Specify that the date type is unknown.

### Integer Year, Integer Month, Integer Quarter, and Integer Day

Specify that the year, month, quarter, or day is a number (for example, 2000).

# 2 Digit Year

Specify that the date uses the last two digits of the year (for example, 99).

#### 3 Letter Month or Month Name

Specify that the level contains a month in string format (for example, Mar). Not available for integers.

#### **Quarter Name**

Specify that the level contains a quarter in string format (for example, First Quarter). Not available for integers.

#### **Name Structure**

Specify the type of name data contained in the level, if any. In the last two selections, [Middle] indicates that the middle name is optional. If the structure

does not match any of the options, click None.

**Note** You can identify a Name structure of the level if the data type of the field associated with it is a **string** data type. If the field representing the level is a **date**, **string**, **integer** or **float** data type, a **date** data type can be associated it.

# **OLAP Cube Dialog Box**

Use this dialog box to display the dimensions and measures, including database properties such as tables, fields, and units of measure, associated with a <u>cube</u>.

# **Options**

#### Is virtual

Indicate whether the cube is virtual. A virtual cube can be either a combination of multiple cubes in one logical cube, somewhat like a relational database view that combines other views and tables, or a cube based on a single cube that exposes only selected subsets of its measures and dimensions.

#### **Fact Table**

Specify the fact table associated with the cube.

#### **Dimensions**

Select the dimensions associated with the cube.

#### Measure

Display and modify the measures in the cube. The **Primary Key** icon indicates the default measure.

# Aggregation

Indicate the type of <u>aggregation</u>, precalculated summary data used to create the units of measure.

#### **Units**

Specify (in singular, rather than plural form) the units of measure in which the measure data is stored (for example, year, inch, or kilometer).

#### Field

Display the fields and synonyms underlying the specified <u>level</u>.

# See Also

**Icons and Cursors Used in English Query** 

# **Preposition Phrasing Dialog Box**

Use this dialog box to specify a <u>preposition phrasing</u> for the relationship.

Preposition phrasing can be specified between subject and object entities that include prepositions. For example, the phrase *patients are on medications* (*for illnesses*) (*at dosages*) (*for studies*) allows the following questions to be answered by English Query:

- "Who is on medication X?"
- "Who is on medication for heart disease?"
- "Who is on medication for the XYZ study?"
- "Who is on medication X for the XYZ study for heart disease?"
- "Who is on medication X at 100 mg for heart disease for the XYZ study?"

# **Options**

# **Subject**

Select an entity that represents the subject of the relationship.

For example, in *patients are on medications (for illnesses) (at dosages) (for studies)*, *patients* is the subject.

### **Preposition**

Specify the preposition or prepositions that are used in the expression.

For example, in *patients are on medications (for illnesses) (at dosages) (for studies)*, *on* is the preposition.

# **Object**

Select an entity that is the object in the relationship.

For example, in *patients are on medications (for illnesses) (at dosages) (for studies), medicines* is the object.

## Add prepositional phrase

Enter the prepositions and the objects of the prepositions (represented by entities) that further describe the relationship.

For example, in *patients are on medications (for illnesses) (at dosages) (for studies), for* (preposition) *illnesses* is the prepositional phrase that further describes the relationship.

# See Also

Prepositional Phrasings in Relationships

# **Project Properties Dialog Box**

Use this dialog box to specify properties that are applicable to the entire model. This dialog box contains the following three tabs:

- <u>Data Connection Tab (Project Properties Dialog Box)</u>
- Default Relationships Tab (Project Properties Dialog Box)
- English Tab (Project Properties Dialog Box)

# **Data Connection Tab (Project Properties Dialog Box)**

Use this tab to set up how the model uses an SQL database or a Microsoft® SQL Server<sup>TM</sup> 2000 Analysis Services database.

# **Options**

#### **SQL Connection**

Create a connection to an SQL database.

### **Enable SQL**

Create entities associated with tables and fields and generate queries for SQL databases.

For an OLAP project, selecting the **Enable SQL** check box allows you to build an English Query model with entities and relationships directly associated with the SQL database tables and fields underlying the OLAP objects. This allows users to ask questions about <u>database objects</u> that may not be present in the OLAP cube.

#### **Connection**

Specify the data source connection string used to connect to the SQL database. Available only if you have enabled SQL.

# Change

Display the Data Link Properties dialog box, where you change your connection. Available only if SQL is enabled.

For information about the **Data Link Properties** dialog box, see the Microsoft Data Link API documentation.

# Type and Version

Select the database management system, such as **SQL Server**, and the version number of the database management software. Available only if SQL is enabled.

#### **Enable Outer Joins**

Specify that the English Query application should generate SQL queries containing <u>outer joins</u>.

By selecting this option, you display rows in which no matching data is found. For example, if the question is *Show the people and their salaries for 1999*, the answer will include all people whether or not they had salaries in 1999.

#### **OLAP Connection**

Create a connection to an Analysis Server database.

#### **Enable OLAP**

Create entities associated with OLAP objects and query an OLAP cube in an Analysis Services database.

If you create a project without using a Project wizard, by default, both **Enable SQL** and **Enable OLAP** are selected.

Enable both OLAP and SQL if questions cannot be answered using only an Analysis Services database or to speed up the retrieval of data for questions that are not about aggregation of data, such as "What products do people buy?"

#### Connection

Specify the data source connection string used to connect to the Analysis Services database. Available only if you have enabled OLAP.

### Change

Display the **Select An Analysis Server** dialog box, where you change your connection. Available only if you have enabled OLAP.

### Type and Version

Select the database management system, such as **MS Analysis Services**, and the version number of the database management software. Available only if OLAP is enabled.

### **Vocabulary**

Specify that the vocabulary (values for entities) should be loaded from the database.

Select this option to include entity values in the dictionary. Clear for a faster compile time.

### Words per entry

Specify a limit to the number of words to be added to the dictionary from each field or <u>level</u>. Select a lower number to decrease compile time.

### Sample data

Specify that the sample data values should be loaded for each entity. When selected prior to building an application, sample data is useful for generating instance values in Question Builder.

Clear for faster modeling and compile time.

**Note** To use Question Builder and include instance values, select **Sample data** prior to building the application.

#### See Also

Creating an English Query Model

**Expanding an English Query Model** 

# **Default Relationships Tab (Project Properties Dialog Box)**

Use this tab to specify the default relationship English Query chooses for vague questions about a particular pair of entities.

If two entities mutually participate in more than one relationship, English Query may not be able to determine which relationship users are referring to when the users ask vague questions. For example, "Show the people and their cars" could mean either "Show the people and cars they own" or "Show the people and the cars they drive." Use this option to specify a default relationship for vague questions about a particular pair of entities.

# **Options**

#### **Entities with Ambiguous Relationships**

View all pairs of entities that participate in more than one relationship.

For example, if you have both relationships, *people own cars* and *people drive cars*, then "people and their cars" and "cars and their people" would appear in this list. For either or both entries, you can select the relationship that should be used to answer vague questions about people and cars.

# **Default Relationship**

Select the relationship to be used when questions about the selected entities are vague.

**Note** Choose a different relationship for each phrasing direction. For example, you can indicate both that "people and their cars" means "people and the cars they own" and also that "cars and their drivers" means "people and the cars they drive."

# **English Tab (Project Properties Dialog Box)**

Use this tab to specify properties that are applicable to the entire English Query application, rather than particular database or semantic objects, represented by the open <u>project</u>.

# **Options**

## Ask about spelling mistakes

Specify that users of the application will be prompted to correct spelling mistakes in their questions. If this option is not selected, English Query attempts to correct spelling errors automatically.

#### **Default date**

Specify that the application should assume the selected date if the question does not specify one directly.

For example, if year is selected, then the question, "Show Ann's sales," is interpreted as "Show Ann's sales for this year." (A default date also can be set for individual relationships.)

#### **Enable user context**

Specify that English Query should retain a "conversational context" for the user.

This option allows the user to refer to previous questions using pronouns (for example, "Who was hired last year?" followed by "Which department hired them?") and ellipsis (for example, "Show the total sales for 1996" followed by "for 1997").

## **Keyword mode**

Provide a best guess of the meaning of a query that English Query does not understand because it contains improper syntax and/or unknown vocabulary.

For example, the question "List the tall customers who ordered Chai," generates different results with and without Keyword mode selected.

- With Keyword mode cleared, English Query generates an error such as
  "Based on the information I've been given about this database, I can't
  answer: 'How tall are customers.' I haven't been given any information
  on tallness." This error occurs because English Query tries to
  understand every part of the query, and tallness has nothing to do with
  who ordered Chai.
- With Keyword mode selected, English Query ignores words and concepts it cannot recognize and makes a best guess of the meaning of the query. In this example, English Query ignores "tall" and uses the question, "which customers ordered Chai?" This is a question that English Query can answer based on information from the database and the model.

#### Assume unknown names refer to

Specify an entity that should be used if it is unclear to what name the entity is referring.

For example, "Show John's appointments" would be interpreted as *show* appointments for the doctor named John.

#### Assume unknown IDs refer to

Specify an entity that should be used if it is unclear to what entity an ID is referring.

#### Assume "I" and "me" refer to

Specify an entity, which contains names that should be used if "I" or "me" is used in a question. If this option is specified, a clarification request is made when the user asks questions starting with "I" or "me" or its variations, such as "my" or "mine." A user can avoid the clarification request by entering an "I am" declarative question, such as "I am Sue." English Query will remember Sue and assume that all "I" and "me" questions refer to Sue.

#### Fiscal Year

Enter parameters for the fiscal year:

### Fiscal year start

Enter the month and day that the fiscal year begins.

#### Assume years are fiscal years

Specify that when a year is used in a question, the application will assume that it refers to a fiscal year unless otherwise specified in the question.

### **Regression Test**

Set parameters of regression:

#### **Regression date**

Specify a particular date to use when running regressions.

Using this option allows time-dependent questions to generate identical responses when regressions are run on different days. For example, the question, "How many customers bought products yesterday?" generates different SQL each day unless this option is set.

#### See Also

**Advanced Entity Properties Dialog Box** 

**Entity/New Entity Dialog Box** 

**OLAP Dimension Dialog Box** 

Select An Analysis Server Dialog Box

# **Regression Differences Dialog Box**

Use this dialog box to display the differences between the current regression file and the regression output file so you can compare them. You cannot make changes to the files from this dialog box.

The **Regression Differences** dialog box displays two files side-by-side. It uses three colors to show deleted, modified, and added lines. Deleted lines appear in blue text, modified lines are displayed in red text, and added lines appear in green text.

# **Options**

#### **Find Button**

Display the **Find** dialog box, where you specify the text you want to find. Words are found and highlighted only in the active pane.

#### **Find Next Button**

Find and select the next occurrence of the text in the active pane that is specified in the **Find What** box of the **Find** dialog box.

#### **Find Previous Button**

Find and select the previous occurrence of the text in the active pane, which is specified in the **Find What** box of the **Find** dialog box.

#### **Previous Difference Button**

Find and move the blue arrow to the beginning of the previous text with differences.

#### **Next Difference Button**

Find and move the blue arrow to the beginning of the text for the next text with differences.

## **Help Button**

Display Help for the dialog box.

#### **Regression File Pane**

Display the original version of the selected regression file.

Text that has been deleted when the next regression test is run is displayed in blue text.

## **Regression Output Pane**

Display results of the most recent regression test performed using the **Run Regression** command.

The output file display includes the restatements, answers, SQL commands, and database results, if applicable.

Text that has been changed or inserted is displayed in red and green text respectively.

#### See Also

**Buttons Used in English Query** 

**Find Dialog Box** 

**Icons and Cursors Used in English Query** 

Testing an English Query Model

# **Relationship/New Relationship Dialog Box**

Use this dialog box to refine the relationships within the model. This dialog box has the following three tabs:

- Command Tab (Relationship/New Relationship Dialog Box)
- Database Tab (Relationship/New Relationship Dialog Box)
- Semantics Tab (Relationship/New Relationship Dialog Box)

# Command Tab (Relationship/New Relationship Dialog Box)

Use this tab to specify properties for the <u>command relationship</u>.

## **Options**

#### **Command Name**

Enter a string representing a command relationship.

#### **Command parameters**

Set the parameters for the command relationship.

#### **Parameter**

Specify the parameters for a command relationship.

#### **Parameter ID**

Display an integer identifying a parameter of a command relationship.

#### See Also

Database Tab (Relationship/New Relationship Dialog Box)

Semantics Tab (Relationship/New Relationship Dialog Box)

# Database Tab (Relationship/New Relationship Dialog Box)

Use this tab to specify database options for a relationship when you want to either identify the table containing all the links to the tables represented by entities or specify a condition under which the relationship is true.

If the relationship is a command relationship, the tab title changes from **Database** tab to **Command** tab.

**Note** Not available if the model is enabled for OLAP only.

# **Options**

#### Table that contains joins to all entities in this relationship

Select the table that is the starting point for all of the joins to the tables and fields that represent entities in the relationship.

#### **Calculate Default Join Table**

Have English Query find the join table.

## This relationship is true only when the following SQL condition is true

Enter an SQL condition that specifies when the relationship is true.

For the relationship *Products are shipped to customers*, this SQL condition is a requirement that customers' credit be approved:

Sales.Approval='T'

## Show most recent data by default

Specify that English Query answers are generated from the most recent data in the database.

For example, if this option is selected, the question, "What is John's salary?" will show only John's most recent salary, although the associated table may contain several historical references to John's salary at various points in time.

This option is available if the relationship has a date that is part of the key of the join table for the relationship. It is unavailable if the date on the relationship is not a single field.

## **See Also**

Command Tab (Relationship/New Relationship Dialog Box)

Semantics Tab (Relationship/New Relationship Dialog Box)

# Semantics Tab (Relationship/New Relationship Dialog Box)

Use this tab to display the entities that participate in a new or existing relationship and to:

- Add and delete entities to and from the relationship, and edit the role the entity plays in the relationship.
- Add, modify, and delete the phrasings for the relationship.
- Specify a time or location for the relationship.

## **Options**

#### **Entities**

Add and delete entities to and from the relationship and edit the role the entity plays in the relationship.

#### **Entities List**

Display each entity that participates in the relationship.

Press CTRL or SHIFT and click the entities to select more than one entity.

#### Add/Edit/Delete

Specify a new entity to be used in the relationship, modify the role of the selected entity, or delete the role of the selected entity.

Deleting an entity from a command relationship does not delete it from the project.

#### When/Start/End

Specify when the relationship occurs, begins, or ends. Only available when a

**Date** entity is in the relationship.

For example, for the relationship, *People buy products*, where the **when** entity is **purchase\_date**, you can ask the following questions:

```
"When did John Smith buy a hammer?"
```

For the relationship, *Companies ship products*, where the start entity is **start\_date** and the end entity is **end\_date**, you can ask the following questions:

```
"When did Acme start shipping Brand X Soap?"
```

#### **Default date**

Specify that this relationship should assume the selected date. Click **None** to indicate that the relationship should use the global default setting. Available only when there is a **Date** entity in the relationship.

#### **Duration**

Specify the duration of the relationship.

For example, for the relationship, *People run races* where the duration entity is **race\_time**, you can ask the following questions:

```
"How long did Mary run?"
```

"How long did it take for Mary to run the Boston Marathon?"

#### Where

Specify the location of the relationship.

For example, for the relationship, People buy products and the location

<sup>&</sup>quot;Who bought a hammer first?"

<sup>&</sup>quot;How many hammers were bought this year?"

<sup>&</sup>quot;What did John Smith buy yesterday?"

<sup>&</sup>quot;Which products were bought in January?"

<sup>&</sup>quot;Who bought a product before December 5, 1995?"

<sup>&</sup>quot;When did Acme stop shipping Brand X Soap?"

<sup>&</sup>quot;Did Acme ship Brand X Soap in 1995?" This question can be answered only if both start and end dates are defined.

<sup>&</sup>quot;Who ran the longest?"

entity, **store**, you can ask the following questions:

"Where did Peter buy flowers?"

For the relationship *Cities are in states* and the location entity is **state**, you can ask the following question:

"Where is Chicago?"

#### **Phrasings**

List all of the phrasings added to the relationship.

Specify phrasings for a relationship. Phrasings are a way to talk about a relationship in English. Choose the phrasings that most closely reflect how users would ask their questions.

For example, *Salespeople sell customers products* is a verb phrasing that describes a relationship between *salespeople* and *product*.

#### Add/Edit/Delete

Add a phrasing type that represents the selected relationship and modify or delete the selected phrasing.

## Group/Ungroup

Specify that the selected phrasings are required to work together in order to describe the relationship, or separate the phrasings.

## Help text

Specify Help about this entity. Users view this information in the Question Builder or by asking questions such as "What do you know about customers?"

#### See Also

Command Tab (Relationship/New Relationship Dialog Box)

Database Tab (Relationship/New Relationship Dialog Box)

**Defining Relationship Phrasings** 

# **Grouped Phrasings Examples**

# <Role Name>Role Dialog Box

Use this dialog box to add entities to a new relationship or to modify entities in an existing relationship that does not have command phrasings.

**Note <Role name>** is the name of the role, not the name of the entity.

# **Options**

#### Always display <entity>

Specify that the entity is always displayed whenever the relationship is used, whether or not the question suggests its use.

For example, you can select this option from the **salesperson** role in the *customers buy products from salespeople* relationship. In response to the question, "Who bought a hammer yesterday?", English Query shows the salesperson who sold the hammer in addition to the person who bought it.

#### How many

Select any entity in the model whose entity type is **Measure**.

It indicates that the quantity of the role entity used in the relationship is indicated by another entity chosen from this list. This allows English Query to answer questions such as "How many products did xxx buy," when each purchase consists of a quantity greater than 1.

**Note** Select either **How many** or **How much**, but not both for a role.

For example, in the relationship, *customers buy products from salespeople*, the **product** role entity can be quantified by the entity, **sales quantity**, that was chosen from this list. This allows English Query to answer the question, "How many products did John buy," with a response of five, if John bought 2 boxes of Chai and 3 boxes of tea.

#### How much

Select any entity in the model whose entity type is **Measure**.

**Note** You can select either **How many** or **How much**, but not both for a role.

For example, the product role in the *customers buy products* relationship can be specified by selecting the **sales amount** entity. This enables English Query to answer the question, "How much cheese did Quick-Stop buy" and get an answer in terms of the sales amount in dollars.

#### **Join Path**

Use the join path that currently exists between the join table and the table or field that represents this entity.

## **Specify**

Display the **Specify Join Path** dialog box, where the join path can be changed. Specify a join path when there is more than one way to get to the entity from the join table specified for the relationship.

#### See Also

**Entities as Roles** 

**Specify Join Path Dialog Box** 

# < Role Name > Role In Command Dialog Box

Use this dialog box to add entities to a new role or to modify roles in an existing relationship that has command phrasings.

**Note <Role name>** is the name of the role, not the name of the entity.

#### **User must specify**

Indicate that the role is mandatory. If the user does not state a specific instance of the entity and you used the **Prompt if not present** option, English Query will ask for the information.

A mandatory role is one in which the command is not complete unless this role is supplied. The role can be a command argument. Mandatory roles allow entities used as command relationship arguments to be required. This is used for situations where the target command has arguments that are required for the command to work. For example, in the command, *ship* (amounts of) products to customers on dates, product, customer, and amount must all be specified or the phrase is incomplete.

### **Prompt if not present**

Enter the text that your English Query application displays if **User must specify <role**> is selected and the user's command does not specify this role.

For example, if the user says "Ship 10 units of Chai," mandatory roles cause English Query to ask for a customer.

## The user's command can specify a quantity ("how many") for this entity

Allow, but not require, the user to specify the quantity of the role that is being used in the command. The user only has to specify a quantity if you enter text in the **Prompt if not present** box.

Only enabled if **User must specify <role>** is selected.

**Note** You can select either **How many** or **How much**, but not both for a role.

## **Prompt if not present**

Allow English Query to ask for a quantity ("how many") if the quantity is

not specified in the command.

Only enabled if **The user's command can specify a quantity ("how many") for this** entity is selected.

For example, if you entered "Please indicate how many units" and the user says, "Ship Chai to John", the text appears and the user must enter the number of units of Chai that should be shipped to John.

## The user's command can specify an amount ("how much") for this entity

Allow the user to specify the amount the role is used in the command. The user only has to specify an amount if you enter text in **Prompt if not present**.

Only enabled if **User must specify <role>** is selected.

**Note** You can select either **How many** or **How much**, but not both for a role.

#### **Prompt if not present**

Allow English Query to ask for the amount ("how much") if it is not specified in the command.

Only enabled if **The user's command can specify an amount ("how much") for this** entity is selected.

For example, if you entered "Please indicate the amount of the shipment in dollars" and the user says, "Ship Chai to John", the text appears and the user must enter the shipment cost in dollars of Chai that should be shipped to John.

## **Entity type**

Specify the data type for non-database entities. This option is only available for entities that are not associated with a <u>database object</u>.

#### None

(Default). Indicate that no data type checking occurs when English Query responds to commands.

#### Numeric

Indicate that the entity is a **numeric** data type.

## **Text**

Indicate that the entity is a **string** data type.

#### **Date**

Indicate that the entity is a **date** data type.

# See Also

**Entities as Roles** 

**Specify Join Path Dialog Box** 

Using Command Relationships in an English Query Model

# **Select An Analysis Server Dialog Box**

Use this dialog box to select the Analysis server and database to use in the project.

# **Options**

## **Analysis Server**

Enter the name of the Analysis server where the database that you want to use resides.

#### **Database**

Select the database from the specified Analysis server.

# **Select Phrasing Dialog Box**

Use this dialog box to select a phrasing type for a relationship.

A phrasing describes a relationship between entities. In other words, phrasings are a way to talk about a relationship in English. Choose the phrasing that most closely reflects how users would ask their questions.

For example, Customers buy products from stores is a verb phrasing that describes a relationship. It allows questions to be asked such as, "What customers bought widgetX from Store #1?"

## **Options**

#### Name/ID Phrasing

Create a relationship in which an entity has a name or ID.

For example, Name/ID phrasing includes *employee\_names* are the names of *employees* or titles are the names of books.

## **Trait Phrasing**

Create a relationship in which an entity is a trait of another entity.

For example, trait phrasing include *patients have blood types*.

## **Preposition Phrasing**

Create a relationship between two entities in which one is the subject and one is the accompanying object of a preposition.

For example, preposition phrasing includes *stories* are about topics where *stories* is the subject, *about* is the preposition, and *topics* is the object.

### **Adjective Phrasing**

Create a relationship in which an entity is described by an <u>adjective</u>.

For example, adjective phrasing includes *city is hot* where **city** is an entity, and *hot* may be defined in the project or contained in a field in the database.

## **Subset Phrasing**

Create an English description of a relationship in which one entity or word is a subset of another entity.

For example, subset phrasing includes *some mountains are volcanoes where volcanoes* are a subset of *mountains*.

### **Verb Phrasing**

Create a relationship in which the entity acts.

For example, verb phrasing includes *people work* or *salespeople sell products to customers*.

#### **Command Phrasing**

Create phrasing used to define new model-specific commands in a command relationship. Command relationships can use entities that are not represented by <u>database objects</u>, such as Delete folder in the command, Move *file* into the Delete folder.

For example, command phrasing includes *print <file>* or *show paintings*.

#### See Also

**Adjective Phrasing Dialog Box** 

**Command Phrasing Dialog Box** 

**Defining Relationship Phrasings** 

Name/ID Phrasing Dialog Box

**Preposition Phrasing Dialog Box** 

**Subset Phrasing Dialog Box** 

**Trait Phrasing Dialog Box** 

**Verb Phrasing Dialog Box** 

# **Select Remote Fields Dialog Box**

Use this dialog box to specify fields from other tables to show whenever this entity is displayed.

Available when the entity is an entity associated with a table that has joins to other tables.

**Note** The table associated with the entity must have a join to the table containing the remote field.

## **Options**

#### Table and Field

Select the remote field to display whenever the entity is displayed.

#### Add

Add the remote field selected with **Table** and **Field** options.

#### **Delete**

Delete the selected remote field in the **Selected fields** box.

#### Selected fields

Display the remote fields that you selected in the **Table/Field** options and then added to the list by clicking **Add**.

# **Specify Join Path Dialog Box**

Use this dialog box to specify a join path when there is more than one way to get to the entity from the join table specified for the relationship.

For example, in a relationship called *Sell*, an entity is **branch**. The join table is the table *Sales*. There is more than one way to get from the table *Sales* to the table *Branches*.

The two ways to get to the table representing the entity are:

 Use the field Branch\_id to join from the table Sales to the table Branches.

In other words, use the join **Sales.branch\_id~Branches.id**.

• Use the field Salesrep\_id to join from the table **Sales** to the table **Salesreps**, and then use the field Branch\_id to join from **Salesreps** to the table **Branches**.

In other words, use the join **Sales.salesrep\_id~Salesreps.id** followed by the join **Salesreps.branch\_id~Branches.id**.

## **Options**

#### Joins available from the table list

Select the join to use for the entity.

This list changes to the list of joins in the other table of your selection each time you click **Add**. For example, if you choose the join between **Branch\_id** in the **Branches** table and the **Sales** table,

**Brances.Branch\_id~Sales.Branch\_id**, and click **Add**, this box will change to list the joins in the **Sales** table.

#### Add

Move the selected join from the **Joins available from the table**  list to the **Join path so far** list and change the **Joins available** list to reflect the joins available in the other table.

# Join path so far

Display a list of the joins that you have added.

#### **Delete**

Move the join selected in the **Join path so far** list to the **Joins available from the table**  list and change the list to reflect the joins available in that table.

# **Subset Phrasing Dialog Box**

Use this dialog box to specify a <u>subset phrasing</u> for a relationship when the relationship can be described as an entity having a subset.

An example is some mountains are volcanoes.

# **Options**

#### **Subject**

Select the entity that represents the universal concept for which you wish to define a subset.

For example, in the relationship *some mountains are volcanoes*, select *mountains* as the subject.

#### **Subset word (singular)**

Select **Subset word** if the subset is simply a word, not an entity.

For example, in the *relationship some mountains are volcanoes*, enter the word *volcanoes*.

## **Entity that contains category values**

Select **Entity that contains category values** if the subset is stored in another entity, and then enter the corresponding entity that contains the subset values.

### **Associate English Values**

Display the **English Data Values** dialog box, where you specify where the subset words are to be found: either in a database field or in a set of specified word-value.

For example, a lookup table can contain gender codes and the values they represent (*M* for *man* and *F* for *woman*).

### Add prepositional phrase

Enter the **Prepositions** and the **Objects of Prepositions** (represented by entities) that further describe the subset relationship.

For example, in *doctors are primary care providers of patients*, *of patients* is the prepositional phrase.

# See Also

**Defining Relationship Phrasings** 

**English Data Values Dialog Box** 

### **Table/New Table Dialog Box**

Use this dialog box to specify the properties of <u>tables</u> or <u>fields</u> that have been added to a <u>project</u>.

Existing database tables or fields can be included in a project one by one (rather than having them extracted automatically). For example, use this command when adding a table to a database structure.

### **Options**

### **Primary Key Icon**

Specify that the field is part of the <u>primary key</u> for the table. Right-click the field to change the primary key.

#### **Field Name**

Display the name of the field you want to add. You can change the field name by typing a new one.

### **Data Type**

Specify the type of data the field contains. If you do not know what kind of data the field contains, select **Unknown**.

The following describes the English Query data types and their equivalent Microsoft® SQL Server™ data types:

Integer—int, smallint, tinyint Float—real, decimal, numeric, money, small money

Date—datetime, smalldatetime

String—char, varchar

Bit—bit

Text—text

Binary—binary, varbinary, image

**Other**—**timestamp** (anything not listed above)

**GUID**—a globally unique identifier

### **Caption**

Optional. Specify the label for this field, if one exists.

For example, the **LastName** field may have a caption of Last Name, which is used on forms.

#### **Basic Tab**

Use this tab to define the fields in your project.

### **Advanced Tab**

Use this tab to further define the fields in your project.

### See Also

Advanced Tab (Table/New Table Dialog Box)

Basic Tab (Table/New Table Dialog Box)

**Icons and Cursors Used in English Query** 

### **Basic Tab (Table/New Table Dialog Box)**

Use this tab to define the fields in a model.

### **Options**

### **Capitalization**

Specify the case in which the data was entered.

The effect of using this option depends on whether the **Add values of this entity to the model** check box is selected.

### For all values of Capitalization other than Unknown:

Regardless of how the user types in the data value in the question, the SQL statement generated by English Query matches the capitalization values. For example, if the user asks, "Who sold the most widgets," and the capitalization on the **product\_name** field is "UPPERCASE", then the SQL statement will contain product\_name = 'WIDGET'. An exception to this can occur when the values of **product\_name** are loaded into the model. If there are any data values that violate the capitalization convention, these are memorized so that the correct capitalization will be used in the SQL statement.

### For capitalization that is Unknown:

The capitalization in the SQL statement generated by English Query matches the user's capitalization exactly, whether or not the data values have been loaded into the model. For example, if the user asks, "How many awards did Gone With the Wind win," the SQL statement is movie\_name = 'Gone With the Wind'. When users are unlikely to know the correct capitalization to use, pick the closest capitalization value other than Unknown, and load values of the entity into the model so that the capitalization is memorized by English Query.

If you do not know how the data was entered or the data is inconsistently capitalized, select the Unknown check box.

### **Date Type**

Specify the date type of the data in the field, if any.

### Integer Year, Integer Month, Integer Quarter, and Integer Day

Use if the field containing a year, month, quarter, or day is a number (for example, 2000).

### 2 Digit Year

Use if the field contains the last two digits of the year (for example, 99).

#### 3 Letter Month

Use if the field contains a month in String format (for example, Mar). Not available for **Integers**.

### **Month or Quarter Name**

Use if the field contains a month or quarter in String format (for example, March). Not available for **Integers**.

#### **Name Structure**

Specify the type of name data contained in the field, if any.

For example, in the last two selections, **[Middle]** indicates that the middle name is optional. If the structure does not match any of the options, choose **None**.

### **Search Type**

Specify the type of search to perform by default on the field when a user enters a question.

#### **Exact Match**

Default if the field is not full-text indexed. Display the field that matches exactly, including capitalization. For example, "where is the restaurant named Bento" would find a restaurant named *Bento* but not *The Bento Box, Bento Express*, or *Tachibana Bento*.

#### **Contains**

Display all rows with fields that have words that include the specified word or letters. For example, "Where is the restaurant named Bento"

would find *The Bento Box*, *Bento Express*, and *Tachibana Bento*.

#### **Prefix**

Display all rows with fields that have words beginning with the letters indicated. For example, "where is the restaurant named Bento" would find *Bento Express* but not *The Bento Box*.

#### **Sounds Like**

Find rows with fields containing words that have the same sound. For example, "where is the restaurant named bentow" would find *The Bento Box, Bento Express*, and *Tachibana Bento*.

#### Free Text

Default if the **Full-text indexed** check box is selected for the field. Display all the rows with fields that approximately match the words or phrase specified. For example, "Display all of the books about how I can cook seafood Japanese style" would display *Cooking Seafood New England Style, The Simple Art of Japanese Cooking,* and *Aunt Louise's Seafood Cookbook* because it would find all books about seafood, Japanese, and cooking. Only available if you select **Full-text indexed**.

#### **Units of Measure**

Specify in singular, rather than plural, form the units of measure in which the field data, if any, is stored (for example, year, inch, or kilometer).

### Field contains a fiscal year

Indicate if this field contains a fiscal year.

#### Full-text indexed

Default if the field is full-text indexed. Enable full-text searching for the field if the database supports full-text indexing.

**Note** This option is disabled if you are not using Microsoft® SQL Server<sup>™</sup> version 7.0 or later, or if the field data type is not String or Text.

When using SQL Server to create a full-text index for a table, specify a unique key column to be stored in the index. Specify the same column as the <u>primary key</u> for the table in English Query. Otherwise, the query that English

Query generates for full-text searching in that table will not retrieve any results.

Clear this option to disable full-text searching, even if the field is full-text indexed.

### **See Also**

Advanced Tab (Table/New Table Dialog Box)

### **Advanced Tab (Table/New Table Dialog Box)**

Use this tab to further define the fields in a model.

### **Options**

#### **Allow Nulls**

Specify that the field can contain nulls. Clearing this will cause the application to omit explicit null checks from the SQL database.

#### **Denormalized**

Specify that the selected field is a duplicate of a field in another table. Sometimes databases are denormalized for efficiency purposes.

For example, in a database containing **Sales**, **Customers**, and **Products** tables, the product name appears both in the **Products** table and in the **Sales** table. This allows a question such as, "Who bought a sprocket last week?", to be answered without a join to the **Products** table. This property is set on the **Product Name** field in the **Sales** table, indicating it is the same as the **Product** field in the **Products** table. English Query then can take advantage of the denormalization, rather than treating the fields as if they are two different fields that happen to have the same, or similar, names.

### Aggregate

Specify that the selected field is an aggregate for a field in another table. Available only for numeric fields (for example, an average or total field).

### **Computed Field**

Specify that the value of this field is derived from a computation, rather than occurring in a database table. Add or edit an <u>SQL expression</u> associated with the selected field. Table names in the expression are required.

An example is dbo.titles.ytd\_sales-dbo.titles.advance or datediff(year, dbo.employee.hire\_date, getdate()).

### See Also

Basic Tab (Table/New Table Dialog Box)

**SQL Database Normalization Rules** 

### **Test Properties Dialog Box**

Use this dialog box to change the time-out and the number of rows to display when executing queries in the Model Test window.

### **Options**

### **Query Time-out**

Specify the time in seconds that will be allowed to elapse before a query will time out. The range must be between 1 and 9999 seconds. The default is 120 seconds.

### Maximum rows to display

Enter the number of rows of query results that will be displayed per question when executing a database query. The number must be between 1 and 1000 rows. The default is 100 rows.

### See Also

**Model Test Window** 

Testing an English Query Model

### **Trait Phrasing Dialog Box**

Use this dialog box to specify a trait phrasing for your relationship.

An *entity* is a real-world object, referred to by a noun. Some entities are traits of other entities. For example, the entity, **blood type**, is a trait of the entity **Patient**.

When specifying the two entities in this dialog box, provide the information needed to answer these types of questions:

- "What is Mary's blood type?" and "Show me the blood type of Mary."
- "What hair color does John have?" and "Show me the hair color of John."

**Note** These questions can be formed using the <u>possessive case</u> (Mary's blood type) or by using *of* and *have*.

Prepositional phrases that further define the trait can be specified. For example, *Banks have branches in cities*. In this case, *banks* is the subject, *branches* is the trait, and *in cities* is an additional object that specifies where the branches are.

### **Options**

### **Subject**

Select an entity that represents the subject of the relationship.

For example, in the phrase, *Banks have branches in cities*, *banks* is the subject.

### **Object**

Select an entity that contains the trait

For example, in the phrase, *banks have branches in cities*, *branches* is the object and trait.

### Add prepositional phrase

Enter the prepositions and the object of prepositions that further describe the

relationship.

For example, in the phrase, *banks have branches in cities*, *in the cities* is the prepositional phrase.

### **Verb Phrasing Dialog Box**

Use this dialog box to specify a verb phrasing for your relationship.

### **Options**

### **Sentence type**

Select the type of sentence structure.

### **Subject Verb**

An example is volcanoes erupt or employees work (for companies).

### **Subject Verb Object**

An example is customers buy products or patients take medicines (for illnesses).

### **Subject Verb Object Object**

An example is salespeople sell customers products or doctors give patients medicine.

### **Object are Verb**

An example is *products* are tested or cars are driven (on roads).

### **Object are Verb Object**

An example is *products* are tested or cars are driven (on roads).

### **Examples of selected phrasing**

Display multiple examples of the selected sentence type.

### **Subject**

Select an entity that represents the subject of the relationship. For example, for the verb phrase, *salespeople sell briefcases*, the entity **salesperson** represents the subject.

#### Verb

Enter a verb that describes the action in the relationship. For example, in the

verb phrase, salespeople sell briefcases, sell is the verb.

#### **Indirect**

Select an entity that contains an indirect object. For example, in the verb phrase, *students give teachers answers*, **teachers** is the indirect object.

### **Direct object**

Select an entity that contains a direct object. For example, in the verb phrase, *salespeople sell briefcases*, **briefcases** is the direct object.

### Add prepositional phrase

Enter the **Prepositions** and the **Object of prepositions** that further describe the relationship. For example, in the verb phrasing, *products come from the warehouses*, *from the warehouse* is the prepositional phrase.

### See Also

**Defining Relationship Phrasings** 

Verb Phrasings in Relationships

### **Troubleshooting English Query Applications**

This section includes information about resolving English Query usage issues and compile-time error messages.

### **English Query Usage Issues**

The following issues may affect the performance of English Query applications.

## English Query OLAP Projects Are Not Supported on Windows 95 and Windows 98

- English Query is supported on the Microsoft® Windows® 95 and Windows 98 operating systems, but the server components required by Microsoft SQL Server<sup>TM</sup> 2000 Analysis Services are not.
- To build English Query OLAP projects on Windows 95 and Windows 98, manually install a SQL Server version 7.0 OLAP Services client (such as the MDX Sample or Microsoft Excel 2000). You still cannot build OLAP mixed mode projects that use multidimensional expressions (MDX) and SQL because those projects require Decision Support Objects (DSO), which do not work at all on Windows 95 and Windows 98. However, with this workaround, you can create or edit an English Query OLAP project on Windows 95 or Windows 98, though you cannot select the **Enable SQL** option in an OLAP project.

## "Enable SQL" in OLAP Model Requires Membership in the OLAP Administrators Group

• With Analysis Services, you cannot access the underlying database structure of an OLAP cube in an English Query model unless you are a member of the OLAP Administrators group.

That is, you must be a member of the OLAP Administrators group or a Microsoft Windows NT® 4.0 and Windows 2000 user group, on the server of the Analysis Services database for which you are building an English Query model. The user of the English Query modeling tool must have OLAP Administrator permissions to load the tables underlying the cubes using DSO.

### **Domain Files from Earlier Versions of English Query Must Be Rebuilt**

• English Query domain files (\*.eqd) are built to work with a specific

version of the English Query engine. If you are upgrading from an earlier version of English Query, you must rebuild your English Query domain files from your project (\*.eqp) using the Model Editor.

### **DBCS** (Unicode) Is Not Fully Supported

• Because English Query is English-only, the user interface for the authoring tool does not support double-byte character keyboard input. However, Unicode data in your database is supported.

### Temp Table Generation Requires SQL Server version 6.5 SP3 or Later

• SQL Server version 6.5 is supported by this release of English Query, but only when you install Service Pack 3 (SP3) or later. If you are not running SQL Server 6.5 with SP3, you may encounter problems with generated **temp** tables that are required for some queries.

## **Problems Converting English Query 1.0 Projects to this Version of English Query**

• If you encounter problems converting English Query version 1.0 to this release of English Query, convert the English Query 1.0 project to English Query version 7.0 first.

### Full-Text Search Does Not Work on Tables With uniqueidentifier Keys

• English Query is unable to build SQL statements for questions that require a full-text search on a table that has a **uniqueidentifier** key. Asking such questions may cause English Query to stop responding.

### See Also

<u>Installation Requirements and Considerations</u>

# **Compile-Time Error Messages (Authoring Object Model)**

Compile-time errors are returned by the English Query methods: **Domain**. **Compile** and **Domain**. **Build**. These compile-time errors approximate the object-level validation errors generated by the authoring tool in English Query version 7.0.

ID	Value	Type	Ol
eqIDTooLong	0x10025	eqSevError	< <b>v</b>
eqErrorInCube	0x10B00	eqSevError	eq
eqBadCubeName	0x10B01	eqSevError	eq
eqBadCubeTable	0x10B02	eqSevError	eq
eqDuplicateDimension	0x10B03	eqSevError	eq
eqDuplicateMeasure	0x10B04	eqSevError	eq
eqDuplicateCubeName	0x10B05	eqSevError	eq
eqMissingDimensions	0x10B06	eqSevError	eq
eqMissingMeasures	0x10B07	eqSevError	eq
eqMissingCubeTable	0x10B09	eqSevError	eq

eqMissingFactTable	0x10B0A	eqSevError	eq
eqDuplicateDefaultMeasure	0x10B0B	eqSevError	eq
eqMissingDefaultMeasure	0x10B0C	EqSevError	Εç
eqErrorInDictEntry	0x10800	eqSevError	eq
eqBadIrregularForm	0x10801	eqSevError	eq
eqBadIrregularFormType	0x10802	eqSevError	eq
eqBadPartOfSpeech	0x10803	eqSevError	eq
eqBadRootWord	0x10804	eqSevError	eq
eqDuplicateDictEntry	0x10805	eqSevError	eq
eqDuplicateIrregularForm	0x10806	eqSevError	eq
eqDuplicateRootWord	0x10807	eqSevError	eq
eqMissingIrregularForm	0x10808	eqSevError	eq

eqMissingIrregularFormType	0x10809	eqSevError	eq
eqMissingPartOfSpeech	0x1080A	eqSevError	eq
eqMissingRootWord	0x1080B	eqSevError	eq
eqMissingWriteWord	0x1080C	eqSevError	eq
eqMissingReadWord	0x1080D	eqSevError	eq
eqWriteSynonymSameAsWord	0x1080E	eqSevWarnin	ıg eq
eqWriteSynonymWithBadChar	0x1080F	eqSevError	eq
eqErrorInDimension	0x10C00	eqSevError	eq
eqBadDimensionName	0x10C01	eqSevError	eq
eqBadDimensionTable	0x10C02	eqSevError	eq
eqBadDimensionType	0x10C03	eqSevError	eq

0x10C04	eqSevError	eq
0x10C05	eqSevError	eq
0x10C06	eqSevError	eq
0x10C08	eqSevError	eq
0x10C09	eqSevError	eq
0x10C0A	eqSevError	eq
0x10C0B	eqSevError	eq
0x10100	eqSevError	eq
0x10101	eqSevError	eq
0x10102	eqSevError	eq
0x10103	eqSevError	eq
0x10104	eqSevError	eq
	0x10C05  0x10C06  0x10C08  0x10C0A  0x10C0B  0x10100  0x10101  0x10102  0x10103	0x10C04         eqSevError           0x10C05         eqSevError           0x10C06         eqSevError           0x10C08         eqSevError           0x10C09         eqSevError           0x10C0A         eqSevError           0x10C0B         eqSevError           0x10100         eqSevError           0x10101         eqSevError           0x10102         eqSevError           0x10103         eqSevError           0x10104         eqSevError

0x10105	eqSevError	eq
0x10106	eqSevError	eq
0x10107	eqSevError	eq
0x10108	eqSevError	eq
0x10109	eqSevError	eq
0x1010A	eqSevError	eq
0x1010B	eqSevError	eq
0x1010C	eqSevError	eq
0x1010D	eqSevError	eq
0x1010E	eqSevError	eq
0x1010F	eqSevError	eq
	0x10106  0x10107  0x10108  0x10109  0x1010A  0x1010C  0x1010C	0x10106         eqSevError           0x10107         eqSevError           0x10108         eqSevError           0x10109         eqSevError           0x1010A         eqSevError           0x1010B         eqSevError           0x1010C         eqSevError           0x1010D         eqSevError           0x1010D         eqSevError

eqBadSampleData	0x10110	eqSevError	eq
eqBadSortBy	0x10111	eqSevError	eq
eqBadSynonymInNameSynonym	0x10112	eqSevError	eq
eqBadUnknownDates	0x10113	eqSevError	eq
eqBadUnless	0x10114	eqSevError	eq
eqBadWord	0x10115	eqSevError	eq
eqBadWordInNameSynonym	0x10116	eqSevError	eq
eqDateEntityIsTable	0x10117	eqSevError	eq
eqDateFieldWithNameType	0x10118	eqSevError	eq
eqDuplicateDBObject	0x10119	eqSevError	eq
eqDuplicateDefaultCondition	0x1011A	eqSevError	eq

eqDuplicateDefaultRelationship	0x1011B	eqSevError	eq
eqDuplicateDisplayField	0x1011C	eqSevError	eq
eqDuplicateEntity	0x1011D	eqSevError	eq
eqDuplicateEntityDefaultRelationship	0x1011E	eqSevError	eq
eqDuplicateEntityID	0x1011F	eqSevError	eq
eqDuplicateEntityType	0x10120	eqSevError	eq
eqDuplicateInheritsFrom	0x10121	eqSevError	eq
eqDuplicateMemorizeNames	0x10122	eqSevError	eq
eqDuplicateNameSynonym	0x10123	eqSevError	eq

eqDuplicateNameType	0x10124	eqSevError	eq
eqDuplicateNumbersIn	0x10125	eqSevError	eq
eqDuplicateSortBy	0x10126	eqSevError	eq
eqDuplicateUnknownDates	0x10127	eqSevError	eq
eqDuplicateUnless	0x10128	eqSevError	eq
eqDuplicateWord	0x10129	eqSevError	eq
eqEntityConditionTakesDateTime	0x1012A	eqSevError	eq
eqEntityConditionTakesInteger	0x1012B	eqSevError	eq
eqEntityConditionTakesNumber	0x1012C	eqSevError	eq
eqEntityConditionWithBadDatePart	0v1012D	eqSevError	eq
equinity condition with butbutter at t	UNIUIZD	equevilion	-q

eqEntityWithBothTableAndField	0x1012E	eqSevError	eq
eqEntityWithMultipleTables	0x1012F	eqSevError	eq
eqIdNametypeOnly	0x10130	eqSevError	eq
eqLoopInSubentity	0x10131	eqSevError	eq
eqMissingDefaultRelationship	0x10132	eqSevError	eq
eqMissingEntityType	0x10134	eqSevError	eq
eqMissingEntityWord	0x10135	eqSevError	eq
eqMissingRelatedEntity	0x10136	eqSevError	eq
eqMissingSynonymInNameSynonym	0x10137	eqSevError	eq

eqMissingWordInNameSynonym	0x10138	eqSevError	eq
eqNameSynonymsWithoutNameType	0x10139	eqSevError	eq
eqNonTableEntityWithDisplayField	0x1013A	eqSevError	eq
eqWrongEntityTypeForName	0x1013B	eqSevError	eq
eqSuggestAddRemoteNameFields	0x1013C	eqSevInfo	eq
eqSuggestDifferentNameWord	0x1013D	eqSevInfo	eq
eqSuggestLinkedEntity	0x1013E	eqSevInfo	eq
eqDontUseNameAsWord	0x1013F	eqSevWarning	geq
eqInitialCapsOnEntityWord	0x10140	eqSevInfo	eq

eqUnderscoreInEntityWord	0x10142	eqSevWarning	eq
eqMissingSampleData	0x10143	eqSevWarning	eq
eqPropertyEntitySameAsLevel	0x10144	eqSevError	eq
eqBadCharacterInWord	0x10146	eqSevError	eq
eqCantLoadWordsFromAlternateTable	0x10147	eqSevError	eq
eqDateFieldEntityNotDate	0x10148	eqSevWarning	eq
eqDBEntityRequiredForMeasure	0x10149	eqSevError	eq
eqErrorLoadingInstanceValues	0x1014A	eqSevWarning	eq
eqNameStructureNotAllowedOnDates	0x1014B	eqSevError	eq
eqNametypeNotAllowed	0x1014C	eqSevError	eq

eqNameTypeNotAllowedForDateLevels	0x1014D	eqSevError	eq
eqNameTypeNotAllowedForDateProperties	0x1014E	eqSevError	eq
eqNoJoinForRemoteFields	0x1014F	eqSevError	eq
eqNoJoinForSubentity	0x10150	eqSevError	eq
eqNumberAsEntityWord		eqSevError	eq
eqPartOfNameAsEntity	0x10152	eqSevWarning	eq
eqStandaloneNotAllowed	0x10153	eqSevError	eq
eqTableEntityRequiredForNumericalReferences	0x10154	eqSevError	eq
eqTableEntityRequiredForRemoteFields	0x10155	eqSevError	eq

eqSameDateTypeFields	0x10156	eqSevError	eq
eqInvalidTimeDimEntity	0x10157	eqSevError	eq
eqMutipleFieldNameSynonym	0x10158	eqSevError	eq
eqEntityDefaultWrongDatetype	0x10159	eqSevError	eq
eqEntityDefaultBadDate	0x1015A	eqSevError	eq
eqEntityDefaultWrongDatatype	0x1015B	eqSevError	eq
eqEntityWordTooLong	0x1015D	eqSevError	eq
eqErrorInField	0x10500	eqSevError	eq

0x10501	eqSevError	eq
0x10502	eqSevError	eq
0x10503	eqSevError	eq
0x10504	eqSevError	eq
0x10505	eqSevError	eq
0x10506	eqSevError	eq
0x10507	eqSevError	eq
0x10508	eqSevError	eq
0x10509	eqSevError	eq
0x1050A	eqSevError	eq
0x1050B	eqSevError	eq
0x1050C	eqSevError	eq
	0x10502 0x10503 0x10504 0x10505 0x10506 0x10507 0x10508 0x10508 0x10508	0x10502         eqSevError           0x10503         eqSevError           0x10504         eqSevError           0x10505         eqSevError           0x10506         eqSevError           0x10507         eqSevError           0x10508         eqSevError

eqCaptionTooLarge	0x1050D	eqSevError	eq
eqDuplicateAggregate	0x1050E	eqSevError	eq
eqDuplicateCapitalization	0x1050F	eqSevError	eq
eqDuplicateCaption	0x10510	eqSevError	eq
eqDuplicateComputation	0x10511	eqSevError	eq
eqDuplicateDataType	0x10512	eqSevError	eq
eqDuplicateDateType	0x10513	eqSevError	eq
eqDuplicateDenormalizedCopy	0x10514	eqSevError	eq
eqDuplicateFieldID	0x10515	eqSevError	eq
eqDuplicateNameStructure	0x10516	eqSevError	eq
eqDuplicateUnits	0x10517	eqSevError	eq
eqMissingAggregateField	0x10518	eqSevError	eq

eqMissingAggregateType	0x10519	eqSevError	eq
eqMissingDataType	0x1051A	eqSevError	eq
eqMissingDateType	0x1051B	eqSevError	eq
eqMissingFieldID	0x1051C	eqSevError	eq
eqMissingFiscalYear	0x1051E	eqSevError	eq
eqMissingNameStructure	0x1051F	eqSevError	eq
eqInitialCapsOnUnits	0x10520	eqSevWarning	eq
eqUnderscoreInUnits	0x10521	eqSevWarning	eq
eqAggregateFieldNonNumeric	0x10522	eqSevError	eq
eqDenormalizedCopyHasWrongType	0x10523	eqSevError	eq
eqComputedPrimaryKey	0x10524	eqSevError	eq
eqWrongTableCaseInComputation	0x10525	eqSevError	eq

	010526	C	
eqWrongFieldCaseInComputation	0x10526	eqSevError	eq
eqBadNameSearchCombo	0x10527	eqSevError	eq
eqFieldWithoutTableInComputation	0x10528	eqSevError	eq
eqBadTableInComputation	0x10529	eqSevError	eq
eqBadFieldInComputation	0x1052A	eqSevError	eq
eqErrorInGlobal	0x10000	eqSevError	eq
eqBadDBMaxRows	0x10001	eqSevError	eq
eqBadDBMSType	0x10002	eqSevError	eq
eqBadDBMSVersion	0x10003	eqSevError	eq
eqBadDBTimeout	0x10004	eqSevError	eq
eqBadGlobalDefaultDate	0x10005	eqSevError	eq
eqBadDefaultID	0x10006	eqSevError	eq

eqBadDefaultModule	0x10007	eqSevError	eq
eqBadDefaultName	0x10008	eqSevError	eg
eqBadDefaultSelf	0x10009	eqSevError	eq
eqBadDSN	0x1000A	eqSevError	eq
eqBadFiscalYearDay	0x1000B	eqSevError	eq
eqBadFiscalYearMonth	0x1000C	eqSevError	eq
eqBadLoadWords	0x1000D	eqSevError	eq
eqBadRegressionDate	0x1000E	eqSevError	eq
eqBadRegressionOutFile	0x1000F	eqSevError	eq
eqBadRegressionQuestionFile	0x10010	eqSevError	eq
eqBadRegressionSaveFile	0x10011	eqSevError	eq
eqDuplicateDBMaxRows	0x10012	eqSevError	eq
eqDuplicateDBMSInfo	0x10013	eqSevError	eq
eqDuplicateDBTimeout	0x10014	eqSevError	eq
eqDuplicateDefaultDate	0x10015	eqSevError	eq
eqDuplicateDefaultID	0x10016	eqSevError	eq
eqDuplicateDefaultModule	0x10017	eqSevError	eq

eqDuplicateDefaultName	0x10018	eqSevError	eq
eqDuplicateDefaultSelf	0x10019	eqSevError	eq
eqDuplicateDSN	0x1001A	eqSevError	eq
eqDuplicateFiscalYear	0x1001B	eqSevError	eq
eqDuplicateLoadWords	0x1001C	eqSevError	eq
eqDuplicateRegressionDate	0x1001D	eqSevError	eq
eqDuplicateRegressionOutFile	0x1001E	eqSevError	eq
eqDuplicateRegressionQuestionFile	0x1001F	eqSevError	eq
eqDuplicateRegressionSaveFile	0x10020	eqSevError	eq
eqMissingFiscalYearDay	0x10021	eqSevError	eq
eqMissingFiscalYearMonth	0x10022	eqSevError	eq
eqOLAPObjectsInNonOLAPProject	0x10023	eqSevError	eq
eqUndefinedObject	0x10024	eqSevError	eq
eqUndefinedObjectInGlobal	0x10026	eqSevError	eq

0x10027	eqSevWarnin	ıg eq
0x10028	eqSevError	eq
0x10029	eqSevError	eq
0x10600	eqSevError	eq
0x10601	eqSevError	eq
0x10602	eqSevError	eq
0x10603	eqSevError	eq
0x10604	eqSevError	eq
0x10605	eqSevError	eq
0x10606	eqSevError	eq
	0x10028  0x10029  0x10600  0x10601  0x10602  0x10603  0x10604	0x10028         eqSevError           0x10029         eqSevError           0x10600         eqSevError           0x10601         eqSevError           0x10602         eqSevError           0x10603         eqSevError           0x10604         eqSevError           0x10605         eqSevError

eqDestinationFieldUsedTwice	0x10607	eqSevError	eq
eqDuplicateJoin	0x10608	eqSevError	eq
eqDuplicateJoinCondition	0x10609	eqSevError	eq
eqDuplicateJoinID	0x1060A	eqSevError	eq
eqJoinConditionToSelf	0x1060B	eqSevError	eq
eqJoinDoesntCoverKey	0x1060C	eqSevError	eq
eqJoinFieldIsOtherDatatype	0x1060D	eqSevError	eq
eqJoinFieldsHaveDifferentDatatype	0x1060E	eqSevError	eq
eqJoinInWrongDirection	0x1060F	eqSevError	eq

eqMissingDestinationField	0x10610	eqSevError	eq
eqMissingDestinationTable	0x10611	eqSevError	eq
eqMissingJoinCondition	0x10612	eqSevError	eq
eqMissingSourceField	0x10613	eqSevError	eq
eqSameSourceJoins	0x10614	eqSevError	eq
eqDuplicateReversedJoin	0x10615	eqSevError	eq
eqErrorInLevel	0x10D00	eqSevError	eq
eqBadLevelField	0x10D01	eqSevError	eq
eqBadLevelName	0x10D02	eqSevError	eq
eqDuplicateProperty	0x10D03	eqSevError	eq
eqDuplicateLevelName	0x10D04	eqSevError	eq
eqMissingLevelField	0x10D05	eqSevError	eq

eqBadLevelDateType	0x10D07	eqSevError	eq
eqDateLevelWithoutDateField	0x10D08	eqSevError	eq
eqLevelDataCapitalizationMismatch	0x10D09	eqSevWarning	eq
eqLevelDateTypeMismatch	0x10D0A	eqSevWarning	eq
eqLevelNameStructureMismatch	0x10D0B	eqSevWarning	eq
eqMissingLevelDateType	0x10D0C	eqSevError	eq
eqMissingLevelSourceField	0x10D0D	eqSevError	eq
eqMeasureUnitsMismatch	0x11001	eqSevWarning	eq
eqNumericFieldRequiredForMeasure	0x11002	eqSevError	eq
eqErrorInPhrasing	0x10300	eqSevError	eq
eqAdjectiveIsTableWithCompoundKey	0x10301	eqSevError	eq

eqAdjectiveIsTableWithoutName	0x10302	eqSevError	eq
eqAdjPhrasingNeedsAdjOrObject	0x10303	eqSevError	eq
eqBadAdjective	0x10304	eqSevError	eq
eqBadEnglishValues	0x10305	eqSevError	eq
eqBadMainPrep	0x10306	eqSevError	eq
eqBadLowValueThreshold	0x10307	eqSevError	eq
eqBadLowValueWord	0x10308	eqSevError	eq
eqBadObject	0x10309	eqSevError	eq
eqBadPhrasingID	0x1030A	eqSevError	eq

eqBadHighValueThreshold	0x1030B	eqSevError	eq
eqBadHighValueWord	0x1030C	eqSevError	eq
eqBadPrep	0x1030D	eqSevError	eq
eqBadPrepObject	0x1030E	eqSevError	eq
eqBadSubject	0x1030F	eqSevError	eq
eqBadSubsetWord	0x10310	eqSevError	eq
eqBadValueField	0x10311	eqSevError	eq
eqBadValueInWordValuePair	0x10312	eqSevError	eq
eqBadVerb	0x10313	eqSevError	eq
eqBadWordField	0x10314	eqSevError	eq
eqBadWordInWordValuePair	0x10315	eqSevError	eq

eqDBEntityRequiredForAdjective	0x10316	eqSevError	eq
eqDBEntityRequiredForName	0x10317	eqSevError	eq
eqDBEntityRequiredForSubject	0x10318	eqSevError	eq
eqDBEntityRequiredForSubset	0x10319	eqSevError	eq
eqDuplicateAdjective	0x1031A	eqSevError	eq
eqDuplicateLowValueThreshold	0x1031B	eqSevError	eq
eqDuplicateLowValueWord	0x1031C	eqSevError	eq
eqDuplicateObject	0x1031D	eqSevError	eq
eqDuplicatePhrasing	0x1031E	eqSevError	eq

eqDuplicatePhrasingID	0x1031F	eqSevError	eq
eqDuplicateHighValueThreshold	0x10320	eqSevError	eq
eqDuplicateHighValueWord	0x10321	eqSevError	eq
eqDuplicatePrep	0x10322	eqSevError	eq
eqDuplicatePrepObject	0x10323	eqSevError	eq
eqDuplicatePrepPhrase	0x10324	eqSevError	eq
eqDuplicateSubject	0x10325	eqSevError	eq
eqDuplicateSubsetWord	0x10326	eqSevError	eq
eqDuplicateVerb	0x10327	eqSevError	eq

0x10328	eqSevError	eq
0x10329	eqSevError	eq
0x1032A	eqSevError	eq
0x1032B	eqSevError	eq
0x1032C	eqSevError	eq
0x1032D	eqSevError	eq
0x1032E	eqSevError	eq
0x1032F	eqSevError	eq
	0x10329  0x1032A  0x1032B  0x1032C  0x1032C	0x10329         eqSevError           0x1032A         eqSevError           0x1032B         eqSevError           0x1032C         eqSevError           0x1032D         eqSevError           0x1032D         eqSevError

eqMissingLookupCodeField	0x10330	eqSevError	eq
eqMissingLookupValues	0x10331	eqSevError	eq
eqMissingLookupWordField	0x10332	eqSevError	eq
eqMissingMainPrep	0x10333	eqSevError	eq
eqMissingMeasurementEntity	0x10334	eqSevError	eq
eqMissingLowValueWord	0x10335	eqSevError	eq
eqMissingNameEntity	0x10336	eqSevError	eq
eqMissingNameType	0x10337	eqSevError	eq
eqMissingObject	0x10338	eqSevError	eq
eqMissingHighValueWord	0x10339	eqSevError	eq

eqMissingPrepObject	0x1033A	eqSevError	eq
eqMissingPreposition	0x1033B	eqSevError	eq
eqMissingSubject	0x1033C	eqSevError	eq
eqMissingSubsetEntity	0x1033D	eqSevError	eq
eqMissingSubsetWord	0x1033E	eqSevError	eq
eqMissingValueInWordValuePair	0x10340	eqSevError	eq
eqMissingVerb	0x10341	eqSevError	eq
eqMissingWordInWordValuePair	0x10343	eqSevError	eq
eqNamedEntityIsField	0x10344	eqSevError	eq
eqNameEntityIsTable	0x10345	eqSevError	eq

eqNumericValueRequired	0x10346	eqSevError	eq
eqObjectHasEntityAndRole	0x10347	eqSevError	eq
eqPhrasingHasTooManyObjects	0x10348	eqSevError	eq
eqSubjectHasEntityAndRole	0x10349	eqSevError	eq
eqSubsetEntityIsMultiField	0x1034A	eqSevError	eq
eqSubsetIsTableWithoutName	0x1034C	eqSevError	eq
eqSubsetPhrasingNeedsWordOrObject	0x1034D	eqSevError	eq
eqSuggestByPrepAsSubject	0x1034F	eqSevInfo	eq
eqSuggestToPrepAsIndirectObject	0x10350	eqSevInfo	eq

eqInitialCapsOnAdjective	0x10351	eqSevWarning eq
eqInitialCapsOnLowValueWords	0x10352	eqSevWarning eq
eqInitialCapsOnHighValueWords	0x10353	eqSevWarning eq
eqInitialCapsOnSubsetWord	0x10354	eqSevWarning eq
eqInitialCapsOnVerb	0x10355	eqSevWarning eq
eqUnderscoreInAdjective	0x10356	eqSevWarning eq
eqUnderscoreInLowValueWords	0x10357	eqSevWarning eq
eqUnderscoreInHighValueWords	0x10358	eqSevWarning eq
eqUnderscoreInSubsetWord	0x10359	eqSevWarning eq

eqUnderscoreInVerb	0x1035A	eqSevWarnin	g eq
eqVerbNotRoot	0x1035B	eqSevWarnin	geq
eqAdjectiveIsMultiField	0x1035C	eqSevError	eq
eqAdjectivePhrasingWithBadOLAPEntity	0x1035D	eqSevError	eq
eqLowThresholdBiggerThanHigh	0x1035E	eqSevError	eq
eqNamePhrasingWithPrepPhrases	0x1035F	eqSevError	eq
eqNumberAsAdjective	0x10360	eqSevError	eq
eqNumberAsSubsetWord	0x10361	eqSevError	eq
eqOnlyOneThreshold	0x10362	eqSevError	eq

eqProperNameSubset	0x10363	eqSevWarning	eq
eqErrorInProperty	0x10E00	eqSevError	eq
eqBadPropertyField	0x10E01	eqSevError	eq
eqBadPropertyName	0x10E02	eqSevError	eq
eqDuplicatePropertyName	0x10E03	eqSevError	eq
eqMissingPropertyField	0x10E04	eqSevError	eq
eqPropertyDataCapitalizationMismatch	0x10E06	eqSevWarning	eq
eqPropertyNameStructureMismatch	0x10E07	eqSevWarning	eq
eqMissingPropertySourceField	0x10E08	eqSevError	eq
eqErrorInMeasure	0x10F00	eqSevError	eq

eqBadMeasureAggType	0x10F01	eqSevError	eq
eqBadMeasureField	0x10F02	eqSevError	eq
eqBadMeasureName	0x10F03	eqSevError	eq
eqDuplicateMeasureName	0x10F04	eqSevError	eq
eqMissingMeasureAggType	0x10F05	eqSevError	eq
eqMissingMeasureField	0x10F06	eqSevError	eq
eqMissingMeasureName	0x10F07	eqSevError	eq
eqErrorInRelationship	0x10200	eqSevError	eq
eqBadCommandArgumentEntity	0x10201	eqSevError	eq
eqBadCommandArgumentID	0x10202	eqSevError	eq
eqBadCommandArgumentType	0x10203	eqSevError	eq

eqBadDefaultDate	0x10204	eqSevError	eq
eqBadDomainCommandID	0x10205	eqSevError	eq
eqBadDuration	0x10206	eqSevError	eq
eqBadEnd	0x10207	eqSevError	eq
eqBadJoinTable	0x10208	eqSevError	eq
eqBadRelationshipID	0x10209	eqSevError	eq
eqBadSQLCondition	0x1020A	eqSevError	eq
eqBadStart	0x1020B	eqSevError	eq
eqBadWhen	0x1020C	eqSevError	eq
eqBadWhere	0x1020D	eqSevError	eq

eqDateEntityRequiredForEnd	0x1020E	eqSevError	eq
eqDateEntityRequiredForStart	0x1020F	eqSevError	eq
eqDateEntityRequiredForWhen	0x10210	eqSevError	eq
eqDBEntityRequiredForDuration	0x10211	eqSevError	eq
eqDBEntityRequiredForEnd	0x10212	eqSevError	eq
eqDBEntityRequiredForLocation	0x10213	eqSevError	eq
eqDBEntityRequiredForStart	0x10214	eqSevError	eq
eqDBEntityRequiredForWhen	0x10215	eqSevError	eq
eqDuplicateCommandArgumentEntity	0x10216	eqSevError	eq

eqDuplicateCommandArgumentID	0x10217	eqSevError	eq
eqDuplicateDuration	0x10218	eqSevError	eq
eqDuplicateEnd	0x10219	eqSevError	eq
eqDuplicateRole	0x1021A	eqSevError	eq
eqDuplicateJoinTable	0x1021B	eqSevError	eq
eqDuplicatePhrasingGroup	0x1021C	eqSevError	eq
eqDuplicateRelationship	0x1021D	eqSevError	eq
eqDuplicateRelationshipDefaultDate	0x1021E	eqSevError	eq
eqDuplicateRelationshipID	0x1021F	eqSevError	eq

eqDuplicateSQLCondition	0x10220	eqSevError	eq
eqDuplicateStart	0x10221	eqSevError	eq
eqDuplicateWhen	0x10222	eqSevError	eq
eqDuplicateWhere	0x10223	eqSevError	eq
eqEndIsMultifield	0x10224	eqSevError	eq
eqMissingCmdInPhrasingGroup	0x10225	eqSevError	eq
eqMissingCommandArgumentEntity	0x10226	eqSevError	eq
eqMissingCommandArgumentID	0x10227	eqSevError	eq
eqMissingCommandArgumentType	0x10228	eqSevError	eq

eqMissingCommandName	0x10229	eqSevError	eq
eqMissingEntities	0x1022A	eqSevError	eq
eqMissingJoinTable	0x1022B	eqSevError	eq
eqMissingPhrasings	0x1022C	eqSevError	eq
eqNoConditionJoinpath	0x1022D	eqSevError	eq
eqNoJoinpath	0x1022E	eqSevError	eq
eqNonCommandPhraseNotInGroup	0x1022F	eqSevError	eq
eqRelationshipWithMultiQtyOrAmt	0x10230	eqSevError	eq
eqStartIsMultifield	0x10231	eqSevError	eq

eqTooManyCmdInPhrasingGroup	0x10232	eqSevError	eq
eqSuggestAugmentRels	0x10233	eqSevInfo	eq
eqSuggestMergeRelationships	0x10234	eqSevInfo	eq
eqSuggestWhen	0x10235	eqSevInfo	eq
eqSuggestWhere	0x10236	eqSevInfo	eq
eqAmbiguousJoinPath	0x10237	eqSevWarning	geq
eqUnsupportedAmountOrQuantity	0x10238	eqSevWarning	geq

eqUnsupportedDuration	0x10239	eqSevWarning	geq
eqUnsupportedEnd	0x1023A	eqSevWarning	geq
eqUnsupportedStart	0x1023B	eqSevWarning	geq
eqUnsupportedWhen	0x1023C	eqSevWarning	geq
eqUnsupportedWhere	0x1023D	eqSevWarning	geq
eqBothStartAndWhen	0x1023E	eqSevError	eq
eqBothEndAndWhen	0x1023F	eqSevError	eq

eqMostRecentNotSupported	0x10240	eqSevError	eq
eqNonDBRequiredForDate	0x10241	eqSevError	eq
eqNonDBRequiredForDuration	0x10242	eqSevError	eq
eqNonDBRequiredForEndDate	0x10243	eqSevError	eq
eqNonDBRequiredForStartDate	0x10244	eqSevError	eq
eqNonLocationUsedAsLocation	0x10245	eqSevError	eq
eqTooManyPhrasingsInRelationship	0x10246	eqSevError	eq
eqTooManyRolesInRelationship	0x10247	eqSevError	eq
eqNoCubePath	0x10248	eqSevError	eq

eqDuplicateRoleInTimeAndLocation	0x10249	eqSevError	eq
eqTwoMeasurePhrasing	0x1024A	eqSevError	eq
eqWrongTableCaseInCondition	0x1024B	eqSevError	eq
eqWrongFieldCaseInCondition	0x1024C	eqSevError	eq
eqTwoMeasurePhrInTwoRel	0x1024D	eqSevError	eq
eqRoleNotUsedInPhrasingGroup	0x1024E	eqSevError	eq

eqFieldWithoutTableInSQLCondition	0x1024F	eqSevError	eq
eqBadTableInSQLCondition	0x10250	eqSevError	eq
eqBadFieldInSQLCondition	0x10251	eqSevError	eq
eqErrorInRole	0x10700	eqSevError	eq
eqAlwaysShowForDomainCmd	0x10/01	eqSevError	eq
eqBadAmountField	0x10702	eqSevError	eq
eqBadRoleID	0x10703	eqSevError	eq
eqBadEntity	0x10704	eqSevError	eq
eqBadJoin	0x10705	eqSevError	eq

eqBadJoinInJoinpath	0x10706	eqSevError	eq
eqBadJoinRef	0x10707	eqSevError	eq
eqBadNonDBType	0x10708	eqSevError	eq
eqBadQuantityField	0x10709	eqSevError	eq
eqDuplicateAmount	0x1070A	eqSevError	eq
eqDuplicateRoleID	0x1070B	eqSevError	eq
eqDuplicateJoinPath	0x1070C	eqSevError	eq
eqDuplicateJoinRef	0x1070D	eqSevError	eq
eqDuplicateNonDBType	0x1070E	eqSevError	eq
eqDuplicateQuantity	0x1070F	eqSevError	eq
eqMissingEntity	0x10710	eqSevError	eq
eqMissingJoinPath	0x10711	eqSevWarning	geq

eqMissingJoinRef	0x10712	eqSevError	eq
eqMissingNonDBType	0x10713	eqSevError	eq
eqNonDBTypeForDBEntity	0x10714	eqSevError	eq
eqNonDBTypeForNonDomainCmd	0x10715	eqSevError	eq
eqMandatoryTableEntityWithoutName	0x10716	eqSevError	eq
eqMandatoryFieldEntityWithoutName	0x10717	eqSevError	eq
eqRoleNotUsedInPhrasing	0x10718	eqSevError	eq
eqRoleNotUseInPhrasingGroups	0x10719	eqSevError	eq

eqAmountEntitySameAsRole	0x1071A	eqSevError	eq
eqMeasureEntityRequiredForAmount	0x1071B	eqSevError	eq
eqMeasureEntityRequiredForQuantity	0x1071C	eqSevError	eq
eqQuantityEntitySameAsRole	0x1071D	eqSevError	eq
eqSingleFieldEntityRequiredForAmount	0x1071E	eqSevError	eq
eqSingleFieldEntityRequiredForQuantity	0x1071F	eqSevError	eq
eqErrorInTable	0x10400	eqSevError	eq
eqBadTableName	0x10401	eqSevError	eq
eqDuplicateField	0x10402	eqSevError	eq
eqDuplicateTableName	0x10403	eqSevError	eq
eqMissingFields	0x10404	eqSevError	eq
eqMissingKeys	0x10405	eqSevError	eq

## English Query

## **Cube Error Messages**

This section provides expanded text for the cube error messages.

# **Duplicate default measure for cube <ObjectID>.**

# **Message Text**

This cube has more than one default measure. Remove the duplicates.

#### **See Also**

# **Duplicate dimension for cube <ObjectID>.**

## **Message Text**

The same dimension is used more than one time in this cube. Remove the duplicates.

#### **See Also**

# **Duplicate measure for cube <ObjectID>.**

## **Message Text**

The same measure is used more than one time in this cube. Remove the duplicates.

#### **See Also**

# **Error in cube <ObjectID>.**

# **Message Text**

This cube contains an unexpected error.

## **See Also**

# **Invalid name for cube <ObjectID>.**

# **Message Text**

The name for this cube is not valid.

#### **See Also**

# **Invalid table for cube <ObjectID>.**

# **Message Text**

The table for this cube is not valid.

## See Also

# Missing default measure in cube <ObjectID>.

# **Message Text**

The default measure for this cube is not specified.

## See Also

# Missing dimensions for cube <ObjectID>.

# **Message Text**

The dimensions for this cube are not specified.

## See Also

# Missing fact table for cube <ObjectID>.

# **Message Text**

The fact table on this cube is not specified.

## See Also

# **Missing measures for cube <ObjectID>.**

# **Message Text**

The measures for this cube are not specified.

#### **See Also**

# Missing table for cube <ObjectID>.

# **Message Text**

The table for this cube is not specified.

## See Also

# **Multiple cubes with ID <ObjectID>.**

## **Message Text**

The same ID is assigned to more than one cube. Rename or remove the duplicates.

#### See Also

# **Dictionary Entry Error Messages**

This section provides expanded text for the dictionary entry error messages.

# **Duplicate irregular form in dictionary entry** < ObjectID>.

# **Message Text**

This dictionary entry has more than one irregular form. Remove the duplicates.

#### **See Also**

# **Duplicate root word in dictionary entry <ObjectID>.**

## **Message Text**

This dictionary entry has more than one root word. Remove the duplicates.

## See Also

# **Error in dictionary entry <ObjectID>.**

# **Message Text**

This dictionary entry contains an unexpected error.

## See Also

# Invalid character in write synonym for dictionary entry <ObjectID>.

## **Message Text**

The write synonym for this dictionary entry contains a nonalphanumeric character.

#### **See Also**

# Invalid irregular form for dictionary entry <ObjectID>.

# **Message Text**

The irregular form for this dictionary entry is not valid.

#### **See Also**

# Invalid irregular form type for dictionary entry <ObjectID>.

# **Message Text**

The irregular form type for this dictionary entry is not valid.

#### **See Also**

# Invalid part of speech for dictionary entry <ObjectID>.

# **Message Text**

The part of speech for this dictionary entry is not valid.

#### **See Also**

# **Invalid root word for dictionary entry <ObjectID>.**

## **Message Text**

The root word for this dictionary entry is not valid.

#### **See Also**

# Missing irregular form in dictionary entry <ObjectID>.

# **Message Text**

The irregular form for this dictionary entry is not specified.

#### **See Also**

# Missing irregular type in dictionary entry <ObjectID>.

# **Message Text**

The irregular type for this dictionary entry is not specified.

#### **See Also**

# Missing part of speech in dictionary entry <ObjectID>.

# **Message Text**

The part of speech for this dictionary entry is not specified.

#### **See Also**

# Missing read word for dictionary entry <ObjectID>.

## **Message Text**

This dictionary entry is a read synonym, but the read word is not specified.

## See Also

# Missing root word in dictionary entry <ObjectID>.

# **Message Text**

The root word for this dictionary entry is not specified.

## See Also

# Missing write word for dictionary entry <ObjectID>.

## **Message Text**

This dictionary entry is a write synonym, but the write word is not specified.

## See Also

# Multiple dictionary entries with ID <ObjectID>.

## **Message Text**

This dictionary entry defines the same root word as another dictionary entry. Remove the duplicates.

#### See Also

# Write synonym in dictionary entry <ObjectID> is identical to the root word.

## **Message Text**

The write synonym for this dictionary entry is the same as the word for which it is a write synonym.

#### See Also

# **Dimension Error Messages**

This section provides expanded text for the dimension error messages.

# Date types of levels in time dimension <ObjectID> are not in descending order.

## **Message Text**

The date types on the levels of this dimension are not in descending order (for example, year then month then day).

#### See Also

# **Duplicate level for dimension <ObjectID>.**

## **Message Text**

The same level is used more than one time in this dimension. Remove the duplicates.

#### **See Also**

# **Error in dimension <ObjectID>.**

# **Message Text**

This dimension contains an unexpected error.

## **See Also**

# **Invalid name for dimension <ObjectID>.**

# **Message Text**

The name for this dimension is not valid.

#### **See Also**

# **Invalid table for dimension <ObjectID>.**

# **Message Text**

The table for this dimension is not valid.

#### **See Also**

# **Invalid type for dimension <ObjectID>.**

# **Message Text**

The type for this dimension is not valid.

# See Also

# Missing levels for dimension <ObjectID>.

# **Message Text**

The levels for this dimension are not specified.

## **See Also**

# Missing source table for dimension <ObjectID>.

# **Message Text**

The source table on this dimension is not specified.

#### **See Also**

# Missing table for dimension <ObjectID>.

# **Message Text**

The table for this dimension is not specified.

## **See Also**

# **Missing type for dimension <ObjectID>.**

# **Message Text**

The type for this dimension is not specified.

## **See Also**

# Multiple dimensions with ID <ObjectID>.

# **Message Text**

The same ID is assigned to more than one dimension. Rename or remove the duplicates.

#### See Also

# **Entity Error Messages**

This section provides expanded text for the entity error messages. In some cases, an action is recommended to resolve the errors.

# A database object is not specified for measure entity <ObjectID>.

# **Message Text**

This measure entity does not have a database object. Measure entities must be represented by a database object.

#### **See Also**

# Comparing {datepart} to entity that has no {datepart} in entity default for entity <ObjectID>.

#### **Message Text**

In an entity default on this entity, a date value is being compared with an entity, which does not contain dates of that type.

#### See Also

# Comparing date entity {entityid} to nondate in entity default for entity <ObjectID>.

# **Message Text**

In an entity default on this entity, a nondatetime value is being compared with a datetime entity.

#### **See Also**

# Comparing integer entity {entityid} to noninteger in entity default for entity <ObjectID>.

# **Message Text**

In an entity default on this entity, a noninteger value is being compared with an integer entity.

# See Also

# Comparing numeric entity {entityid} to nonnumber in entity default for entity <ObjectID>.

## **Message Text**

In an entity default on this entity, a nonnumeric value is being compared with a numeric entity.

#### See Also

# Date entity {entityid} in a default condition on entity <ObjectID> cannot be compared with a {date type}.

#### **Message Text**

This entity has an entity default with a default condition that compares a constant value to an entity with a date type not compatible with that value. For example, comparing the value "1/1/2000" with a month entity is not valid.

#### See Also

# Date entity {entityid} in a default condition on entity <ObjectID> cannot be compared with a {value}.

#### **Message Text**

This entity has an entity default with a default condition that compares a constant value that is not a recognized date to an entity that is a date.

#### See Also

# Date entity <ObjectID> has a name type.

# **Message Text**

This entity is represented by a date field but has a name type. Date entities cannot contain names or IDs.

#### See Also

# **Date entity <ObjectID> is a table.**

# **Message Text**

This entity is a date but is represented by a table. Dates can be represented only by fields.

#### See Also

# Display fields for entity <ObjectID> should include fields from name entity {entityid}.

#### **Message Text**

This entity is not defined to show its name fields. The fields in its name entity should be added as remote display fields.

#### **See Also**

# **Duplicate database object for entity <ObjectID>.**

# **Message Text**

The same database object is used more than one time in this entity. Remove the duplicates.

#### See Also

# **Duplicate default condition in entity default for entity** < **ObjectID**>.

## **Message Text**

This entity has an entity default that contains more than one occurrence of the same default condition. Remove the duplicates.

# See Also

# Duplicate default relationship for entity <ObjectID> and entity {entityid}.

# **Message Text**

This entity has more than one default relationship for the same entity. Remove the duplicates.

#### **See Also**

# **Duplicate display field for entity <ObjectID>.**

# **Message Text**

The same field is used more than one time in display fields of this entity. Remove the duplicates.

#### See Also

# **Duplicate entity type for entity <ObjectID>.**

# **Message Text**

This entity has more than one entity type. Remove the duplicates.

## **See Also**

# **Duplicate name synonym {synonym}/{value} for entity <ObjectID>.**

# **Message Text**

The same name synonym is used more than one time in this entity. Remove the duplicates.

#### **See Also**

# **Duplicate name type for entity <ObjectID>.**

# **Message Text**

This entity has more than one name type. Remove the duplicates.

## **See Also**

# **Duplicate numbers in entity <ObjectID>.**

# **Message Text**

This entity has more than one field containing numeric references to the entity. Remove the duplicates.

#### See Also

# **Duplicate relationship on entity default for entity** <**ObjectID>.**

# **Message Text**

An entity default for this entity has more than one relationship. Remove the duplicates.

#### **See Also**

# **Duplicate sort-by field for entity <ObjectID>.**

# **Message Text**

The same field is used more than one time in sort-by fields of this entity. Remove the duplicates.

#### See Also

# **Duplicate table to memorize names from for entity** <**ObjectID>.**

# **Message Text**

This entity has more than one table from which database values are loaded. Remove the duplicates.

# See Also

# **Duplicate unknown date relationship for entity** < ObjectID >.

# **Message Text**

This entity has more than one unknown date relationship. Remove the duplicates.

#### **See Also**

# Duplicate unless relationship {relationshipid} on entity default for entity <ObjectID>.

#### **Message Text**

This entity has an entity default in which a relationship is used as an unless relationship more than one time. Remove the duplicates.

#### See Also

# **Duplicate word** "{word}" for entity < ObjectID>.

## **Message Text**

The same word is used more than one time on this entity. Remove the duplicates.

## **See Also**

# Entity <ObjectID> and entity {entityid} have the same underlying database object(s).

#### **Message Text**

This entity and another entity are represented by the same database objects. Remove one of the entities and add its words as synonyms of the other entity.

#### See Also

# Entity <ObjectID> cannot load words from table {tableid}.

#### **Message Text**

This entity has an alternate table specified to load words from, but that table does not contain the fields that represent this entity.

#### **Explanation**

While building an application, English Query loads the vocabulary in the database either from the table that contains the entity or from an alternate table. If an alternate table is used, it must contain denormalized copies of the fields that represent the entity.

For example, the database for a project contains a **Customers** table and an alternative table called **Current Customers**, which contains fields that are marked as denormalized. If the **Customers** table contains a **Suite number** field and the **Suite number** field is missing from the **Current Customers** table, English Query will not be able to load words from the table.

This error can occur when:

- The Semantic Modeling Format (SMF) text file has been edited and a field is missing from one table but not the other.
- A field is marked as a denormalized copy of a field and deleted in one table without making the same changes in the corresponding field of the other table.

#### Action

To correct this error:

• Mark one field as a denormalized copy of the other field.

- Add the missing field to the appropriate table.
- Edit the SMF text file to either add the field or remove it from the other table.
- Refresh the tables.

#### See Also

Advanced Tab (Table/New Table Dialog Box)

Compile-Time Error Messages (Authoring Object Model)

How to add a field to a table

How to add tables and views from a SQL data source

How to edit a table

How to refresh all tables in a project

**SQL Database Normalization Rules** 

# Entity <ObjectID> cannot show fields from table {tableid}.

## **Message Text**

This entity has remote fields specified, but there is no join from this entity to the table that contains the remote fields.

#### **See Also**

# Entity <ObjectID> has both table and field as database objects.

## **Message Text**

This entity is represented by both a table and a field. Entities can be represented only by a single table, or by one or more fields.

#### See Also

# Entity <ObjectID> has display fields but is not a table entity.

## **Message Text**

This entity is not represented by a table but has display fields. Only table entities can have display fields.

#### **See Also**

# Entity <ObjectID> has multiple tables as database objects.

## **Message Text**

This entity is represented by more than one table. Entities can be represented only by a single table, or by one or more fields.

#### See Also

# Entity <ObjectID> has the same words as its {nametype} {entityid}.

## **Message Text**

This entity has the same words as another entity that is its name or ID. It is recommended that you choose other words to differentiate them.

#### See Also

## **Entity <ObjectID>** is a subentity of itself.

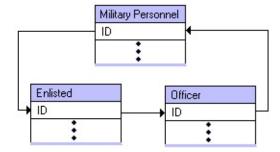
#### **Message Text**

This entity is a subentity of itself, due to a loop in the subentity hierarchy. One of the entities in this hierarchy should not be a subentity.

## **Explanation**

The subentity hierarchy cannot be circular. It must go in one direction from the subentity to the parent entity. The following examples show an incorrect hierarchy:

- **Enlisted** is a subentity of **Military Personnel**.
- **Officer** is a subentity of **Enlisted**.
- **Military Personnel** is a subentity of **Officer**.



This error occurs when:

• An error is introduced while editing the Semantic Modeling Format (SMF).

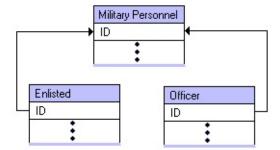
#### **Action**

To prevent this error:

• Redirect the foreign keys when you make one entity a subentity of the other.

To correct this error, redirect the join and make one entity a subentity of the other:

1. In the **Join/New Join** dialog box, correct the joins between the table so that they point in the right direction. In the Military Personnel example, point the join for **Enlisted** to **Military Personnel** and delete the join from **Enlisted** to **Officer** as seen in the following diagram:



2. In the **Advance Entity Properties** dialog box for the subentity, on the **Semantics Properties** tab, in the **Is subentity of** list, click the parent entity. In this example, **Military Personnel** would be listed in the **Is subentity of** option for both the **Enlisted** and **Officer** entities.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to add a join

How to delete a join

How to edit a join

Join/New Join Dialog Box

Semantic Properties Tab (Advanced Entity Properties Dialog Box)

**Specify Join Path Dialog Box** 

**SQL Database Normalization Rules** 

Table/New Table Dialog Box

# Entity <ObjectID> is name/ID of entity {entityid}, but has entity type other than None.

#### **Message Text**

This entity has an entity type other than "None". Because it is the name or ID of another entity, the entity type should be "None".

#### See Also

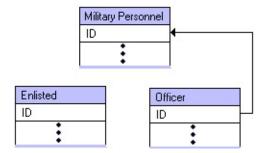
# Entity <ObjectID> is subentity of {entityid}, but has no join.

#### **Message Text**

This entity is a subentity of another entity, but there is no join from the key of the subentity (child) table to the key of the entity (parent) table.

#### **Explanation**

The subentities (child) table has no join to connect it to the parent entities table. In the following example, both **Enlisted** and **Officer** are subentities of **Military Personnel** but only **Officer** has a join to the **Military Personnel** table.



This error occurs when:

• The underlying join between the child table and the parent table is missing.

#### **Action**

To prevent this error:

- Make sure that a valid join exists between the child and parent tables and that the join is in the right direction.
- Maintain all desired joins when you edit the Semantic Modeling Format (SMF).

- Check for errors in changes to the SMF (for example, typographical errors, deletion of essential material and incorrect SMF).
- If a parent table is deleted, delete the joins to that table.

#### To correct this error:

Create a join between the child and parent table by using the **Table/New Table** dialog box or by editing the SMF. For example, create a join between **Enlisted** and **Military Personnel**.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to add a join

How to delete a join

How to edit a join

Join/New Join Dialog Box

**Specify Join Path Dialog Box** 

**SQL** Database Normalization Rules

Table/New Table Dialog Box

# Entity <ObjectID> represented by date field, but not marked as date.

## **Message Text**

This entity is represented by date/time fields, but it is not marked as a date.

#### **See Also**

# Entity <ObjectID> should be represented by level {levelid}.

#### **Message Text**

This entity is represented by a member property whose underlying field is the same as the underlying field for some level in the model. If a level and a property have the same underlying field, only the level can represent an entity.

#### **Explanation**

In an OLAP or mixed SQL/OLAP model, every OLAP object has an underlying SQL object. You cannot have one entity representing the OLAP member property and another entity representing the OLAP level if the member property and the level have the same underlying database fields. Only one entity representing the level should exist.

#### Action

To correct this error:

• Delete the member property entity. English Query will determine the appropriate SQL statements from the OLAP level.

#### See Also

<u>Additional Relationships</u>

Compile-Time Error Messages (Authoring Object Model)

**Expanding an English Query Model** 

How to delete an entity

How to edit an entity

How to manually map OLAP objects to the underlying SQL database objects

## Naming Entities in the OLAP Project Wizard

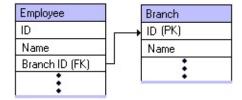
## Entity <ObjectID> should be represented by table <tableid>.

#### **Message Text**

The field that represents this entity is joined to the key of another table. The entity should be represented by the other table rather than this field.

#### **Explanation**

This entity is represented by a field that has a join to the key of another table. For example, if a **Branch** entity is created from the **Branch ID** field in the **Employee** table, an error occurs. The error occurs because a join exists from the **Branch ID** field of the **Employee** table to the **ID** field of the **Branch** table. Make the entity a table entity and point it to the **Branch** table as seen in the following diagram:



#### Action

To prevent this error:

• Represent the **Branch** entity by the **Branch** table.

To correct this error:

• In the **Entity/New Entity** dialog box, click **Table**. In the **Table** list, click the table containing the foreign key and in the **Field** list, click the fields you want to display.

**Note** Make sure that the appropriate join exists.

## **See Also**

Compile-Time Error Messages (Authoring Object Model)

**Entity/New Entity Dialog Box** 

How to add a join

How to edit a join

Join/New Join Dialog Box

**Specify Join Path Dialog Box** 

**SQL Database Normalization Rules** 

## **Entity <ObjectID> uses the word "name".**

## **Message Text**

The word "name" on an entity can cause English Query to misunderstand questions about names. Use a more specific word (for example, customer name) on this entity.

#### See Also

## **Error in entity <ObjectID>.**

## **Message Text**

This entity contains an unexpected error.

## See Also

## **Error loading instance values for entity <ObjectID>.**

## **Message Text**

There was an error loading instance values from the database for this entity. Check permissions on the database object that represents this entity.

#### **See Also**

# Fields {fieldid} and {fieldid} in entity <ObjectID> have the same date type.

## **Message Text**

There is more than one field in this entity with the same date type. Date entities must have only one field of each date type.

#### **See Also**

# Fields from name entity {objectid} are reused in entity <ObjectID>.

#### **Message Text**

This entity is represented by fields that represent a different name entity. Generally, fields used in name entities do not require additional entities created from them.

#### **Explanation**

In English Query, a name entity can represent a single field or set of fields. Therefore, it is recommended that another entity not be represented by one of the previously used fields. For example, if the **Customer Name** entity is represented by the **FirstName** and **LastName** fields from the **Customer** table, do not use a new entity, **Customer First Name**, that uses the **FirstName** field of the **Customer** table.

#### Action

To correct this error:

• Delete the entity represented by only one of the fields.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

**Entity/New Entity Dialog Box** 

How to delete an entity

How to edit an entity

# Invalid character {char} in word for entity <ObjectID>.

## **Message Text**

This entity contains an invalid character in one of its words.

#### **See Also**

## **Invalid database object for entity <ObjectID>.**

## **Message Text**

The database object for this entity is not valid.

## **See Also**

# Invalid database value on name synonym for entity <ObjectID>.

## **Message Text**

The database value of a name synonym on this entity is not valid.

#### **See Also**

## **Invalid default relationship for entity <ObjectID>.**

## **Message Text**

The default relationship for this entity is not valid.

## **See Also**

## Invalid display field for entity <ObjectID>.

## **Message Text**

A display field for this entity is not valid.

## **See Also**

# Invalid entity on entity default condition for entity <ObjectID>.

## **Message Text**

The entity in a condition on an entity default of this entity is not valid.

#### **See Also**

## **Invalid entity type for entity <ObjectID>.**

## **Message Text**

The entity type for this entity is not valid.

## **See Also**

## Invalid entity used in the option Is subentity of.

## **Message Text**

The inherits-from entity for this entity is not valid.

## **See Also**

## **Invalid field for entity <ObjectID>.**

## **Message Text**

The field specified as the database object for this entity is not valid.

## **See Also**

## **Invalid ID for entity <ObjectID>.**

## **Message Text**

The ID for this entity is not valid.

## **See Also**

## **Invalid name type for entity <ObjectID>.**

## **Message Text**

This entity has a name type but is not represented by a string or text field.

## **See Also**

## Invalid numerical references specified.

## **Message Text**

The numeric reference field for this entity is not valid.

#### **See Also**

# Invalid operator on entity default condition for entity <ObjectID>.

## **Message Text**

The operator for a condition on an entity default on this entity is not valid.

#### **See Also**

## **Invalid related entity for entity <ObjectID>.**

## **Message Text**

The related entity for a default relationship of this entity is not valid.

## **See Also**

## **Invalid sample data for entity <ObjectID>.**

## **Message Text**

The sample data for this entity is not valid.

## **See Also**

## **Invalid sort field for entity <ObjectID>.**

## **Message Text**

One of the sort fields for this entity is not valid.

## **See Also**

# **Invalid synonym word for entity <ObjectID>.**

# **Message Text**

The synonym word on a name synonym for this entity is not valid.

## **See Also**

# **Invalid table for entity <ObjectID>.**

# **Message Text**

The table specified as the database object of this entity is not valid.

## **See Also**

# Invalid table specified from which to load words.

# **Message Text**

The table to memorize names from for this entity is not valid.

## **See Also**

# Invalid unknown dates relationship for entity <ObjectID>.

# **Message Text**

The unknown dates relationship for this entity is not valid.

#### **See Also**

# Invalid unless relationship on entity default for entity <ObjectID>.

# **Message Text**

The unless relationship for an entity default of this entity is not valid.

#### **See Also**

# Invalid value on entity default condition for entity <ObjectID>.

# **Message Text**

The condition for a default of this entity contains a value that is not valid.

#### **See Also**

# **Invalid word for entity <ObjectID>.**

# **Message Text**

One of the words for this entity is not valid.

## **See Also**

# Missing database value on name synonym for entity <ObjectID>.

# **Message Text**

The database value for a name synonym on this entity is not specified.

#### **See Also**

# Missing entity type for entity <ObjectID>.

# **Message Text**

The entity type for this entity is not specified.

## **See Also**

# Missing related entity on default relationship for entity <ObjectID>.

# **Message Text**

The related entity for a default relationship of this entity is not specified.

#### **See Also**

# Missing relationship on default relationship for entity <ObjectID>.

# **Message Text**

The relationship for a default relationship of this entity is not specified.

# See Also

# Missing sample data for entity <ObjectID>.

# **Message Text**

The sample data for this entity is not specified.

## **See Also**

# Missing synonym on name synonym for entity <ObjectID>.

# **Message Text**

The synonym for a name synonym on this entity is not specified.

#### **See Also**

# Missing words for entity <ObjectID>.

# **Message Text**

The words for this entity are not specified.

## **See Also**

# **Mixed-case word for entity <ObjectID>.**

# **Message Text**

Entity words are case-sensitive. Verify that the words on this entity are capitalized in the same way that users will type them in questions.

#### See Also

# More than one entity used in the option Is a subentity of.

## **Message Text**

This entity has more than one inherits-from entity. Remove the duplicates.

#### **See Also**

# **Multiple entities with ID <ObjectID>.**

# **Message Text**

The same ID is assigned to more than one entity. Rename or remove the duplicates.

#### See Also

# Multiple field enity <ObjectID> has name synonyms.

# **Message Text**

This entity is represented by multiple fields but has name synonyms. Only single-field entities can have name synonyms.

#### See Also

# Name structure is not allowed for date entity <ObjectID>.

# **Message Text**

This date entity has a name structure specified. Date entities may not have a name structure.

#### **See Also**

# Name synonym but no name type on entity <ObjectID>.

# **Message Text**

The name type for this entity is not specified. Entities without a name type cannot have name synonyms.

# See Also

# Name type is incompatible with database object type on entity <ObjectID>.

#### **Message Text**

This entity has a name type but is represented by a table or an OLAP dimension, cube, or measure. Only field entities and OLAP level and property entities can have name types.

#### **See Also**

# Name type is incompatible with level date type on entity <ObjectID>.

# **Message Text**

This entity has a name type but is represented by a member property that has a date type. Entities represented by date properties cannot be names.

#### See Also

# Name type is incompatible with property date type on entity <ObjectID>.

# **Message Text**

This entity has a name type but is represented by a member property that has a date type. Entities represented by date properties cannot be names.

#### See Also

# Numeric entity <ObjectID> has name type other than Unique ID.

## **Message Text**

This entity is represented by a numeric field, but has a name type other than Unique ID. Numeric entities cannot contain names.

#### See Also

# Numeric references are not allowed for nontable entity <ObjectID>.

# **Message Text**

This entity has a numeric references property, but is not a table entity. Only table entities can have a numeric references property.

#### See Also

# **Numeric word on entity <ObjectID>.**

# **Message Text**

This entity has a word that is a number. Numbers cannot be used as words on entities.

#### See Also

# Remote fields are not allowed for nontable entity <ObjectID>.

# **Message Text**

This entity has remote fields specified, but is not a table entity. Only table entities can have remote fields.

#### **See Also**

# Stand-alone entity <ObjectID> must be either a field entity or a level entity.

#### **Message Text**

This entity is marked as a stand-alone entity but either is not a field entity or is used as the name of another entity. Only nonname field entities can be standalone entities.

#### See Also

# The dimension {objectid} for the time entity <ObjectID> is not a time dimension.

## **Message Text**

This date entity is represented by a dimension that is not a time dimension. Only time dimensions can represent date entities.

#### **See Also**

# The entity <ObjectID> in default condition can only be compared with a {datatype}.

#### **Message Text**

This entity has an entity default with a default condition that compares a constant value to an entity with a data type not compatible with that value. For example, comparing the value "XYZ" with an integer entity is not valid.

#### See Also

# **Underscores used in word for entity <ObjectID>.**

# **Message Text**

Underscores are not commonly used in English nouns. Verify that the words on this entity are entered in the same way that users will type them in questions.

#### See Also

# Word for entity <ObjectID> is too long.

# **Message Text**

One of the words for this entity is too long.

## See Also

# **Field Error Messages**

This section provides expanded text for the field error messages.

# Aggregate field for field <ObjectID> is not numeric.

# **Message Text**

The aggregate field for this field is not numeric.

## **See Also**

# **Caption is too long for field <ObjectID>.**

# **Message Text**

Caption for this field exceeds the length supported by your database.

## See Also

# Case mismatch for field {fieldname} in computation for field <ObjectID>.

#### **Message Text**

The computation for this field references a field for which there is no exact match in the model. There are more than one case-insensitive matches. This makes the SQL condition ambiguous. Change the case to match the desired field exactly.

#### See Also

# Case mismatch for table {tablename} in computation for field <ObjectID>.

#### **Message Text**

The computation for this field references a table for which there is no exact match in the model. There are more than one case-insensitive matches. This makes the SQL condition ambiguous. Change the case to match the desired table exactly

#### See Also

## **Computed field <ObjectID> cannot be in key of table.**

## **Message Text**

This field is a computed field but is marked as a part of the primary key of the table. Computed fields cannot be used in primary keys.

#### See Also

# Denormalized copy field for field <ObjectID> has incorrect data type.

## **Message Text**

The data type of this field does not match the data type of its denormalized copy field.

## See Also

# **Duplicate aggregate field for field <ObjectID>.**

## **Message Text**

This field has more than one aggregate field. Remove the duplicates.

## **See Also**

# **Duplicate capitalization convention for field** < ObjectID >.

## **Message Text**

This field has more than one capitalization convention. Remove the duplicates.

#### **See Also**

# **Duplicate caption for field <ObjectID>.**

## **Message Text**

This field has more than one caption. Remove the duplicates.

## **See Also**

# **Duplicate computation for field <ObjectID>.**

## **Message Text**

This field has more than one computation. Remove the duplicates.

## **See Also**

# **Duplicate data type for field <ObjectID>.**

## **Message Text**

This field has more than one data type. Remove the duplicates.

## **See Also**

# **Duplicate date type for field <ObjectID>.**

## **Message Text**

This field has more than one date type. Remove the duplicates.

## **See Also**

# **Duplicate denormalized copy field for field** <**ObjectID>.**

## **Message Text**

This field has more than one denormalized copy field. Remove the duplicates.

#### **See Also**

## **Duplicate name structure for field <ObjectID>.**

## **Message Text**

This field has more than one name structure. Remove the duplicates.

## **See Also**

## **Duplicate units of measure for field <ObjectID>.**

## **Message Text**

This field has more than one unit of measure. Remove the duplicates.

#### **See Also**

# Error in field <ObjectID>.

## **Message Text**

This field contains an unexpected error.

## See Also

# Field {fieldname} without table name in computation for field <ObjectID>.

## **Message Text**

The computation for this field uses a field without a fully specified table name.

#### **See Also**

# Invalid aggregate field for field <ObjectID>.

## **Message Text**

The aggregate field for this field is not valid.

## See Also

# **Invalid aggregate type for field <ObjectID>.**

## **Message Text**

The aggregate type for this field is not valid.

## See Also

# Invalid capitalization convention for field <ObjectID>.

## **Message Text**

The capitalization convention for this field is not valid.

#### **See Also**

# **Invalid caption for field <ObjectID>.**

## **Message Text**

The caption for this field is not valid.

## **See Also**

# Invalid combination of name structure and search type for field <ObjectID>.

## **Message Text**

The combination of name structure and search type specified for this field is not supported.

#### See Also

# **Invalid computation for field <ObjectID>.**

## **Message Text**

The computation for this field is not valid.

## **See Also**

# **Invalid data type for field <ObjectID>.**

## **Message Text**

The data type for this field is not valid.

## **See Also**

# **Invalid date type for field <ObjectID>.**

## **Message Text**

The date type of this field is incompatible with the field's data type.

## **See Also**

# Invalid denormalized copy field for field <ObjectID>.

## **Message Text**

The denormalized copy field for this field is not valid.

## **See Also**

# **Invalid ID for field <ObjectID>.**

## **Message Text**

The ID for this field is not valid.

#### **See Also**

# **Invalid name for field <ObjectID>.**

## **Message Text**

The name for this field is not valid.

#### **See Also**

# **Invalid name structure for field <ObjectID>.**

## **Message Text**

The name structure for this field is not valid.

#### **See Also**

# **Invalid units of measure for field <ObjectID>.**

## **Message Text**

The units of measure for this field are not valid.

#### **See Also**

# Missing aggregate field for field <ObjectID>.

## **Message Text**

The aggregate field for this field is not specified.

## **See Also**

# Missing aggregate type for field <ObjectID>.

## **Message Text**

The aggregate type for this field is not specified.

## See Also

# Missing data type for field <ObjectID>.

## **Message Text**

The data type for this field is not specified.

## **See Also**

## Missing date type for field <ObjectID>.

## **Message Text**

Date type is not specified for this field, which represents a date entity. All fields representing a date entity must have a date type.

#### See Also

# **Missing ID for field <ObjectID>.**

## **Message Text**

The ID for this field is not specified.

## **See Also**

## Missing name structure for field <ObjectID>.

## **Message Text**

This field is used in name entity; therefore, it must have a name structure.

## **See Also**

## **Mixed-case unit word for field <ObjectID>.**

## **Message Text**

Units are case-sensitive. Verify that the units on this field are capitalized in the same way that users will type them in questions.

#### See Also

## **Multiple fields with ID <ObjectID>.**

## **Message Text**

The same ID is assigned to more than one field. Rename or remove the duplicates.

#### See Also

# No fiscal year specified, but field <ObjectID> is fiscal.

## **Message Text**

This field is marked as fiscal, but fiscal year is not specified in the global properties.

#### See Also

# Undefined field {fieldname} in computation for field <ObjectID>.

# **Message Text**

The computation for this field references an undefined field.

#### **See Also**

# Undefined table {tablename} in computation for field <ObjectID>.

# **Message Text**

The computation for this field references an undefined table.

#### **See Also**

# **Underscores used in unit word for field <ObjectID>.**

## **Message Text**

Underscores are not commonly used in English nouns. Verify that the units of measure on this field are entered in the same way that users will type them in questions.

#### See Also

# **Global Project Error Messages**

This section provides expanded text for the global project error messages. In some cases, an action is recommended to resolve the errors.

# Belongs in nonvalidation compile method errors.

# **Message Text**

Unable to connect to the Analysis server.

## **See Also**

# Default ID entity <ObjectID> has no name phrasing with ID name type.

#### **Message Text**

The entity specified for "Assume unknown IDs refer to" does not have a name phrasing with an entity whose name type is 'Unique ID.' Only entities with unique ID name phrasings can be used as the default ID entity.

#### See Also

# Default name entity <ObjectID> has no name type and no name phrasing.

#### **Message Text**

The entity specified for "Assume unknown names refer to" has no name type and no name phrasing. Only entities with name phrasings or name types can be used as the default name entity.

#### See Also

# **Duplicate database information.**

# **Message Text**

Database information is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate database time-out.**

# **Message Text**

Database time-out is specified more than one time. Remove the duplicates.

## **See Also**

# Duplicate default date.

# **Message Text**

Default date is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate default ID entity.**

# **Message Text**

Default ID entity is specified more than one time. Remove the duplicates.

## **See Also**

# Duplicate default module name.

# **Message Text**

The default module is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate default name entity.**

# **Message Text**

Default name entity is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate default self entity.**

# **Message Text**

Default self entity is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate DSN.**

# **Message Text**

DSN is specified more than one time. Remove the duplicates.

## See Also

# **Duplicate fiscal year start information.**

# **Message Text**

Fiscal year is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate number of words to load.**

## **Message Text**

The number of words to load from the database is specified more than one time. Remove the duplicates.

#### See Also

# **Duplicate regression test date.**

# **Message Text**

Regression test date is specified more than one time. Remove the duplicates.

## **See Also**

# **Duplicate regression test output file.**

## **Message Text**

The regression test output file is specified more than one time. Remove the duplicates.

#### See Also

# **Duplicate regression test question file.**

## **Message Text**

The regression test question file is specified more than one time. Remove the duplicates.

#### See Also

# **Duplicate regression test save file.**

## **Message Text**

The regression test save file is specified more than one time. Remove the duplicates.

#### See Also

# **Invalid database time-out.**

# **Message Text**

The database time-out is not valid.

#### **See Also**

# Invalid database type.

# **Message Text**

The database type is not valid.

## See Also

# Invalid database version.

# **Message Text**

The database version is not valid.

#### **See Also**

# Invalid default date.

# **Message Text**

The default date is not valid.

## **See Also**

# Invalid default ID entity.

# **Message Text**

The default ID entity is not valid.

## See Also

# Invalid default module name.

# **Message Text**

The default module name is not valid.

## **See Also**

# Invalid default name entity.

# **Message Text**

The default name entity is not valid.

# See Also

# Invalid default self entity.

# **Message Text**

The default self entity is not valid.

## See Also

# **Invalid DSN.**

# **Message Text**

The DSN is not valid.

## See Also

# Invalid fiscal year start day.

# **Message Text**

The fiscal year start day is not valid.

## **See Also**

# Invalid fiscal year start month.

# **Message Text**

The fiscal year start month is not valid.

## **See Also**

# Invalid maximum number of rows set.

# **Message Text**

The maximum numbers of rows to return is not valid.

#### **See Also**

# Invalid number of words to load.

# **Message Text**

The number of words to load from the database is not valid.

## **See Also**

## **Invalid regression test date.**

### **Message Text**

The regression test date is not valid.

### See Also

## Invalid regression test output file.

### **Message Text**

The regression test output file name is not valid.

#### See Also

## Invalid regression test question file.

### **Message Text**

The regression test question file name is not valid.

#### See Also

## Invalid regression test save file.

### **Message Text**

The regression test save file name is not valid.

#### See Also

## Loading more than 10,000 words per entity is not recommended.

#### **Message Text**

Loading more than 10,000 words per entity is not recommended, because the size of the compiled English Query model increases with the number of values memorized from the database.

#### See Also

# Maximum number of rows to display is specified more than one time.

#### **Message Text**

The maximum number of result rows is specified more than one time. Remove the duplicates.

#### **See Also**

## Missing fiscal year day.

### **Message Text**

Fiscal year start day is not specified.

#### See Also

## Missing fiscal year month.

### **Message Text**

Fiscal year start month is not specified.

#### See Also

## **OLAP objects in non-OLAP project.**

### **Message Text**

The project contains OLAP objects, but OLAP is not enabled for the project.

#### See Also

# Undefined {objecttype} {objectid} is referenced in {objecttype} <ObjectID>.

#### **Message Text**

This object references another object that has not been defined.

#### **Explanation**

This error occurs when there is a dangling reference. A dangling reference can result from:

- Making errors while creating or editing the Semantic Modeling Format (SMF) (for example, typing *adress* instead of *address*).
- Keeping objects referenced by an object that is being deleted. When deleting an entity, delete the objects that refer to it. This can be done in the **Confirm Deletion** dialog box.
- Importing a model that contains a reference to an object that has been deleted.
- Referencing a role in a phrasing that does not exist in the relationship.

#### Action

To prevent this error:

 Delete all objects that refer to the object. By default, English Query deletes the objects which refer to an entity or role that is being deleted. In the Confirm Deletion dialog box, do not clear the In addition, delete the following objects which refer to the entity <entityname> check box. • Check all typing when editing SMF and make sure that the name of the object does not contain typographical errors.

#### To correct this error:

- Delete the objects that reference this deleted object.
- Create the missing object.
- Import a model that contains the missing object.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to add an entity to a relationship

How to add tables and views from a SQL data source

How to create an entity

How to create relationships using the Create Relationships command

How to delete an entity

How to edit the entity's role in a relationship

# Undefined {objecttype} {objectid} is referenced in global project settings.

#### **Message Text**

The global project settings contain a reference to an object that has not been defined.

#### See Also

## Unexpected error in global settings.

### **Message Text**

The global settings for the project contain an unexpected error.

#### **See Also**

## Unexpected error.

### **Message Text**

An unexpected error has occurred.

#### **See Also**

## **Join Error Messages**

This section provides expanded text for the join error messages. In some cases, an action is recommended to resolve the errors.

# Computed field {fieldid} used in join condition for join <ObjectID>.

#### **Message Text**

This join contains a join condition that uses a computed field. Computed fields cannot be the source or target of a join.

#### **See Also**

# Conditions for join <ObjectID> do not cover primary key fields of destination table {tableid}.

#### **Message Text**

The join conditions in this join do not cover all the fields in the primary key of the destination table.

#### See Also

# Destination field {fieldname} in join <ObjectID> is not a key of destination table {tableid}.

#### **Message Text**

One of the destination fields in this join is not part of the key of the destination table.

#### See Also

# Destination field {fieldname} is used more than once in join <ObjectID>.

#### **Message Text**

The same destination field is used more than one time in this join. Remove the duplicates.

#### **See Also**

# **Duplicate join condition {condition} in join <ObjectID>.**

#### **Message Text**

This join contains the same join condition more than one time. Remove the duplicates.

#### **See Also**

## Error in join <ObjectID>.

### **Message Text**

This join contains an unexpected error.

#### See Also

## **Invalid destination field for join <ObjectID>.**

### **Message Text**

The destination field for this join is not valid.

#### **See Also**

## **Invalid destination table for join <ObjectID>.**

### **Message Text**

The destination table for this join is not valid.

#### **See Also**

## **Invalid ID for join <ObjectID>.**

### **Message Text**

The ID for this join is not valid.

#### **See Also**

## **Invalid source field for join <ObjectID>.**

### **Message Text**

The source field for this join is not valid.

#### **See Also**

# Join <ObjectID> and join {joinid} have the same source fields.

#### **Message Text**

This join and another join have the same source fields. Delete one of these joins.

#### **See Also**

# Join <ObjectID> contains a condition with identical source and destination fields.

#### **Message Text**

This join contains a join condition with identical source and destination fields.

#### See Also

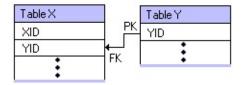
### Join <ObjectID> is backward.

#### **Message Text**

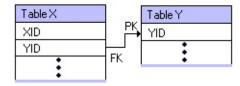
This join is backward. Based on the primary keys of the tables, the join should go from the current destination table to the current source table.

#### **Explanation**

According to the primary keys of the tables, the destination and the source tables of this join are reversed. Joins must be from the source table to the destination table or from the entity to the subentity. For example, if the join is from YID in table Y to the YID in table X, you will have a backward join, as seen in the following diagram.



However, **Table X** is the source table and **Table Y** is the destination table; YID is the primary key in **Table Y** and a foreign key in **Table X**. The correct direction of the join is from YID in **Table X** to YID in **Table Y**, as seen in the following diagram.



This error can occur:

- When editing the Semantic Modeling Format (SMF).
- If a change is made to a primary key that causes a join to be redirected after it is created.

#### **Action**

To correct this error, do one of the following:

- Edit the SMF so that the join goes from the source table to its destination table.
- Edit the join in the Model Editor to reverse the direction of the join.
- Change the primary key to redirect the join.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to change the table options for a relationship

How to edit a join

Join/New Join Dialog Box

**Specify Join Path Dialog Box** 

Table/New Table Dialog Box

**Working with Joins** 

## Join <ObjectID> is identical to join {joinid} except the source and destination are reversed.

#### **Message Text**

This join is identical to another join, except their source and destination are opposite. Remove one of the duplicates.

#### See Also

### Join <ObjectID> is identical to join {joinid}.

#### **Message Text**

This join is identical to another join. Remove one of the duplicates.

#### **See Also**

# Join field {fieldid} and {fieldid} in join <ObjectID> do not have the same data types.

#### **Message Text**

One of the source and destination join fields in this join does not have the same data type.

#### **See Also**

# Join field {fieldid} in join <ObjectID> has data type "Other".

#### **Message Text**

A join field in this join has a data type specified as Other. Joins cannot use Other fields in join conditions.

#### **See Also**

# Missing destination field for join condition in join <ObjectID>.

### **Message Text**

The destination field for a join condition on this join is not specified.

#### **See Also**

## Missing destination table in join <ObjectID>.

### **Message Text**

The destination table for this join is not specified.

#### **See Also**

# Missing join conditions in join <ObjectID>.

## **Message Text**

The join conditions for this join are not specified.

## **See Also**

# Missing source field for join condition in join <ObjectID>.

## **Message Text**

The source field for a join condition of this join is not specified.

## See Also

# Multiple joins with ID <ObjectID>.

## **Message Text**

The same ID is assigned to more than one join. Rename or remove the duplicates.

#### See Also

# **Level Error Messages**

This section provides expanded text for the level error messages.

# Data capitalization does not match field for level <ObjectID>.

## **Message Text**

The data capitalization specified for this level does not match the data capitalization specified for the field associated with it.

#### **See Also**

## Date type does not match field for level <ObjectID>.

## **Message Text**

The date type specified for this level does not match the date type specified for the field associated with it.

#### **See Also**

# Date/time level <ObjectID> does not refer to a date or time field.

## **Message Text**

This level is part of a date dimension, but the corresponding field is not a date or time field. Levels in date dimensions must refer to date/time fields.

#### See Also

## **Duplicate member property for level <ObjectID>.**

## **Message Text**

The same member property is used more than one time in this level. Remove the duplicates.

#### **See Also**

## Error in level <ObjectID>.

## **Message Text**

This level contains an unexpected error.

## **See Also**

# **Invalid date type for level <ObjectID>.**

## **Message Text**

The date type of this level is not valid.

## **See Also**

# **Invalid field for level <ObjectID>.**

## **Message Text**

The field for this level is not valid.

#### **See Also**

# **Invalid name for level <ObjectID>.**

## **Message Text**

The name for this level is not valid.

#### **See Also**

# Missing date type for level <ObjectID>.

## **Message Text**

This level is part of a date dimension, but has no date type.

## **See Also**

# Missing field for level <ObjectID>.

## **Message Text**

The field for this level is not specified.

## **See Also**

# Missing source field for level <ObjectID>.

## **Message Text**

The source field on this level is not specified.

## **See Also**

## **Multiple levels with ID <ObjectID>.**

## **Message Text**

The same ID is assigned to more than one level. Rename or remove the duplicates.

#### **See Also**

# Name structure does not match field for level <ObjectID>.

## **Message Text**

The name structure specified for this level does not match the name structure specified for the field associated with it.

#### See Also

# **Measure Error Messages**

This section provides expanded text for the measure error messages.

# Nonnumeric field is associated with measure <ObjectID>.

## **Message Text**

This measure is associated with a nonnumeric field. Measures must be associated with numeric fields.

#### See Also

## **Units does not match field for measure <ObjectID>.**

## **Message Text**

The units specified for this measure does not match the units specified for the field associated with it.

#### **See Also**

# **Module Error Messages**

This section provides expanded text for the module error messages.

## **Multiple modules with ID <ObjectID>**

## **Message Text**

The same ID is assigned to more than one module. Rename or remove the duplicates.

#### **See Also**

# **Phrasing Error Messages**

This section provides expanded text for the phrasing error messages. In some cases, an action is recommended to resolve the errors.

# Adjective entity {entityid} in phrasing <ObjectID> has no name.

#### **Message Text**

The adjective entity in this phrasing is represented by a table but does not have a name relationship. Table entities can be adjective entities only if they have a name relationship.

#### **See Also**

# Adjective entity for phrasing <ObjectID> is a multifield entity.

## **Message Text**

The adjective entity in this phrasing is represented by multiple fields. Field entities can be adjectives only if they are single fields.

#### **See Also**

# Adjective entity for phrasing <ObjectID> is not a database entity.

## **Message Text**

The adjective entity for this phrasing is not represented by a database object.

#### **See Also**

# Dimension or cube entity is object of adjective phrasing <ObjectID>.

## **Message Text**

This adjective phrasing has a dimension or cube entity as its object. The object of an adjective phrasing cannot be a cube or dimension entity.

#### See Also

## **Duplicate adjective for phrasing <ObjectID>.**

## **Message Text**

The same adjective is used more than one time in this phrasing. Remove the duplicates.

#### See Also

# **Duplicate high-value adjective for phrasing** <**ObjectID>.**

## **Message Text**

The same high-value adjective is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate high-value threshold for phrasing** <**ObjectID>.**

## **Message Text**

This phrasing has more than one high-value threshold. Remove the duplicates.

#### **See Also**

# **Duplicate low-value adjective for phrasing** <**ObjectID>.**

## **Message Text**

The same low-value adjective is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate low-value threshold for phrasing** <**ObjectID>.**

## **Message Text**

This phrasing has more than one low-value threshold. Remove the duplicates.

#### **See Also**

# **Duplicate object entity for phrasing <ObjectID>.**

# **Message Text**

This relationship has more than one object entity. Remove the duplicates.

#### See Also

# **Duplicate object entity in prepositional phrase for phrasing <ObjectID>.**

## **Message Text**

A prepositional phrase in this phrasing has more than one object entity. Remove the duplicates.

#### **See Also**

# **Duplicate preposition (prep) in prepositional phrase** for phrasing <ObjectID>.

## **Message Text**

The same preposition is used more than once in a prepositional phrase on this phrasing. Remove the duplicates.

#### **See Also**

# Duplicate prepositional phrase "{prep} {entityid}" for phrasing <ObjectID>.

## **Message Text**

The same prepositional phrase is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate role** "{roleid}" for phrasing <ObjectID>.

## **Message Text**

The same role is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate subject entity for phrasing <ObjectID>.**

## **Message Text**

This relationship has more than one subject entity. Remove the duplicates.

#### **See Also**

# **Duplicate subset word "{subset}" for phrasing <ObjectID>.**

## **Message Text**

The same subset word is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate verb** "{verb}" for phrasing <ObjectID>.

## **Message Text**

The same verb is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# Duplicate word/value pair "{word} / {value}" for phrasing <ObjectID>.

## **Message Text**

The same word/value pair is used more than one time in this phrasing. Remove the duplicates.

#### **See Also**

# **Duplicate word/value table for phrasing <ObjectID>.**

## **Message Text**

This phrasing has more than one word/value table. Remove the duplicates.

#### **See Also**

# Entity {objectid} has a proper name but is used as the object of subset phrasing <ObjectID>.

#### **Message Text**

Do not use entities with proper names as the object of a subset phrasing. Such entities include: field and level entities with a proper name type, and table and cube entities that have a name phrasing with a name entity that has a proper name type.

#### See Also

# **Error in phrasing <ObjectID>.**

# **Message Text**

This phrasing contains an unexpected error.

## See Also

# High threshold is lower than low threshold in phrasing <ObjectID>.

## **Message Text**

This phrasing has a high-value threshold that is lower than its low-value threshold.

#### **See Also**

# High-value threshold for phrasing <ObjectID> is not a number.

## **Message Text**

The high-value threshold for this phrasing is not a number.

#### **See Also**

# **Invalid adjective for phrasing <ObjectID>.**

# **Message Text**

The adjective for this phrasing is not valid.

#### **See Also**

# **Invalid English values for phrasing <ObjectID>.**

## **Message Text**

The English values for this phrasing are not valid.

#### **See Also**

# Invalid high value adjective for phrasing <ObjectID>.

#### **Message Text**

An adjective associated with high values for this phrasing is not valid.

#### **See Also**

# Invalid high value threshold for phrasing <ObjectID>.

## **Message Text**

The threshold for high values on this phrasing is not valid.

#### **See Also**

# **Invalid ID for phrasing <ObjectID>.**

# **Message Text**

The ID for this phrasing is not valid.

#### **See Also**

# Invalid lookup value for phrasing <ObjectID>.

## **Message Text**

One of the lookup values for this phrasing is not valid.

#### **See Also**

# **Invalid lookup word for phrasing <ObjectID>.**

## **Message Text**

One of the lookup words for this phrasing is not valid.

#### **See Also**

# **Invalid low value adjective for phrasing <ObjectID>.**

## **Message Text**

An adjective associated with low values on this phrasing is not valid.

#### **See Also**

# Invalid low value threshold for phrasing <ObjectID>.

## **Message Text**

The low value threshold for this phrasing is not valid.

#### **See Also**

# **Invalid main preposition for phrasing <ObjectID>.**

# **Message Text**

The main preposition for this phrasing is not valid.

#### **See Also**

# **Invalid object entity for phrasing <ObjectID>.**

# **Message Text**

The object entity for this phrasing is not valid.

#### **See Also**

# **Invalid preposition entity for phrasing <ObjectID>.**

# **Message Text**

The preposition entity for this phrasing is not valid.

#### **See Also**

# **Invalid preposition for phrasing <ObjectID>.**

# **Message Text**

One of the prepositions for this phrasing is not valid.

#### **See Also**

# **Invalid subject entity for phrasing <ObjectID>.**

# **Message Text**

The subject entity of this phrasing is not valid.

#### **See Also**

# **Invalid subset noun for phrasing <ObjectID>.**

# **Message Text**

The subset noun for this phrasing is not valid.

#### **See Also**

# Invalid value field of lookup table for phrasing <ObjectID>.

# **Message Text**

The value field for the lookup table on this phrasing is not valid.

# See Also

# **Invalid verb for phrasing <ObjectID>.**

# **Message Text**

The verb for this phrasing is not valid.

#### **See Also**

# Invalid word field of lookup table for phrasing <ObjectID>.

# **Message Text**

The word field for the lookup table on this phrasing is not valid.

# See Also

# Low-value threshold for phrasing <ObjectID> is not a number.

## **Message Text**

The low-value threshold for this phrasing is not a number.

#### **See Also**

# Measurement entity for phrasing <ObjectID> is a table entity.

## **Message Text**

The measurement entity in this phrasing is represented by a table. Only field entities can be measurements.

#### **See Also**

# Missing adjective entity for phrasing <ObjectID>.

# **Message Text**

The adjective entity for this phrasing is not specified.

#### **See Also**

# Missing adjective for phrasing <ObjectID>.

# **Message Text**

The adjective for this phrasing is not specified.

#### **See Also**

# Missing adjective or object entity for phrasing <ObjectID>.

## **Message Text**

This adjective phrasing has neither an adjective nor an object entity. One or the other is required.

#### **See Also**

# Missing code field in lookup table for phrasing <ObjectID>.

## **Message Text**

The code field in the lookup table for this phrasing is not specified.

## See Also

# Missing high-value adjectives for phrasing <ObjectID>.

## **Message Text**

The high-value adjectives for this phrasing are not specified.

#### **See Also**

# Missing low-value adjectives for phrasing <ObjectID>.

## **Message Text**

The low-value adjectives for this phrasing are not specified.

#### **See Also**

## Missing measurement entity for phrasing <ObjectID>.

## **Message Text**

The measurement entity is not specified for this phrasing.

## **See Also**

## Missing name entity for phrasing <ObjectID>.

## **Message Text**

The name entity for this phrasing is not specified.

## **See Also**

# Missing name type for name entity in phrasing <ObjectID>.

## **Message Text**

The name type for the name entity in this phrasing is not specified.

#### **See Also**

## Missing object entity for phrasing <ObjectID>.

## **Message Text**

The object entity for this phrasing is not specified.

## **See Also**

# Missing preposition entity for prepositional phrase in phrasing <ObjectID>.

## **Message Text**

The preposition entity for one of the prepositional phrases in this phrasing is not specified.

#### **See Also**

# Missing preposition for prepositional phrase in phrasing <ObjectID>.

## **Message Text**

The preposition for a prepositional phrase in this phrasing is not specified.

#### **See Also**

# Missing prepositional phrase for phrasing <ObjectID>.

## **Message Text**

A prepositional phrase for this phrasing is not specified .

#### **See Also**

## Missing subject entity for phrasing <ObjectID>.

## **Message Text**

The subject entity for this phrasing is not specified.

## **See Also**

## Missing subset entity for phrasing <ObjectID>.

## **Message Text**

The subset entity for this phrasing is not specified.

## **See Also**

## Missing subset word for phrasing <ObjectID>.

## **Message Text**

The subset word for this phrasing is not specified.

## **See Also**

# Missing value in word/value pair for phrasing <ObjectID>.

## **Message Text**

The value for a word/value pair on this phrasing is not specified.

#### **See Also**

## Missing verb for phrasing <ObjectID>.

## **Message Text**

The verb for this phrasing is not specified.

## **See Also**

# Missing word field in lookup table for phrasing <ObjectID>.

## **Message Text**

The word field in the lookup table for this phrasing is not specified.

## See Also

# Missing word in word/value pair for phrasing <ObjectID>.

## **Message Text**

The word in a word/value pair for this phrasing is not specified.

#### **See Also**

## Missing word/code pairs for phrasing <ObjectID>.

## **Message Text**

Word/code pairs in the lookup table for this phrasing are not specified.

## **See Also**

## Mixed-case adjective used for phrasing <ObjectID>.

## **Message Text**

Adjectives are case-sensitive. Verify that the adjectives in this phrasing are capitalized in the same way that users will type them in questions.

#### See Also

## Mixed-case subset word used for phrasing <ObjectID>.

## **Message Text**

Subset words are case-sensitive. Verify that the words on this phrasing are capitalized in the same way that users will type them in questions.

#### See Also

## Mixed-case verb used for phrasing <ObjectID>.

## **Message Text**

Verbs are case-sensitive. Verify that the verbs on this phrasing are capitalized in the same way that users will type them in questions.

#### See Also

## Mixed-case, high-value adjective used for phrasing <ObjectID>.

#### **Message Text**

Words associated with high values are case-sensitive. Verify that the words on this phrasing are capitalized in the same way that users will type them in questions.

#### **See Also**

## Mixed-case, low-value adjective used for phrasing <ObjectID>.

#### **Message Text**

Words associated with low values are case-sensitive. Verify that the words on this phrasing are capitalized in the same way that users will type them in questions.

#### **See Also**

## **Multiple phrasings with ID <ObjectID>.**

## **Message Text**

The same ID is assigned to more than one phrasing. Rename or remove the duplicates.

#### See Also

# Name entity {entityid} is table entity for phrasing <ObjectID>.

## **Message Text**

The object of this name phrasing is a table entity. Only field entities can be used as the object of a name phrasing.

#### **See Also**

# Name entity for phrasing <ObjectID> is not a database entity.

## **Message Text**

The name entity for this phrasing is not represented by a database object.

#### **See Also**

## **Numeric adjective in phrasing <ObjectID>.**

## **Message Text**

This phrasing has an adjective that is a number. Numbers cannot be used as adjectives.

#### See Also

## **Numeric subset word in phrasing <ObjectID>.**

## **Message Text**

This phrasing has a subset word that is a number. Numbers cannot be used as subset words.

#### See Also

## Object for phrasing <ObjectID> has both an entity and a role.

## **Message Text**

The object of this phrasing refers to both an entity and a role. It can refer only to one.

#### **See Also**

## Phrasing <ObjectID> has a low threshold or a high threshold but not both.

## **Message Text**

This phrasing has either a low or high value threshold. If one threshold is specified, they both must be specified.

#### **See Also**

# Phrasing <ObjectID> has a table adjective entity with a multifield key.

#### **Message Text**

The adjective entity in this phrasing is represented by a table that has a multi-field key. Table entities can be adjectives only if they have single-field keys.

#### See Also

# Phrasing <ObjectID> is identical to phrasing {phrasingid}.

## **Message Text**

This phrasing is identical to another phrasing in a different relationship. Remove the duplicate.

#### **See Also**

## Prepositional phrases in name phrasing <ObjectID>.

## **Message Text**

This name phrasing contains one or more prepositional phrases. Prepositional phrases are not allowed on name phrasings.

#### See Also

## Relationship <ObjectID> uses a dative alternation.

#### **Message Text**

This phrasing has no indirect object but has both a direct object and the preposition "to". Typically, this indicates the entity that is the object of the preposition should be used as the indirect object instead. For example, instead of "salespeople sell products to customers", it should be "salespeople sell customers products".

#### See Also

## Relationship <ObjectID> uses a passive voice.

## **Message Text**

This phrasing has no subject but has the preposition "by". Typically, this indicates the entity that is the object of the preposition should be used as the subject instead. For example, instead of "products are bought by customers", it should be "customers buy products".

#### See Also

# Subject entity {entityid} is field entity for phrasing <ObjectID>.

#### **Message Text**

The subject of this name phrasing is a field entity. Only table entities can be used as the subject of a name phrasing.

#### **See Also**

# Subject entity for phrasing <ObjectID> is not a database entity.

#### **Message Text**

The subject entity for this phrasing is not represented by a database object.

#### **See Also**

## Subject for phrasing <ObjectID> has both an entity and a role.

#### **Message Text**

The subject of this phrasing refers to both an entity and a role. It can refer only to one.

#### **See Also**

## Subset entity {entityid} in phrasing <ObjectID> has no name.

#### **Message Text**

The subset entity in this phrasing is represented by a table, but does not have a name relationship. Table entities can be subset entities only if they have a name relationship.

#### **See Also**

# Subset entity for phrasing <ObjectID> is a multifield entity.

#### **Message Text**

The subset entity in this phrasing is represented by more than one field. Only single-field entities and table entities can be subset entities.

#### See Also

# Subset entity for phrasing <ObjectID> is not a database entity.

#### **Message Text**

The subset entity in this phrasing is not represented by a database object. Only database entities can be subset entities.

#### **See Also**

# Subset phrasing <ObjectID> has no subset word or object entity.

### **Message Text**

This subset phrasing has neither a subset word nor an object entity.

#### **See Also**

# Text value is not allowed for associated value in phrasing <ObjectID>.

#### **Message Text**

This phrasing contains a word/value pair that has a text value. Because the field representing the adjective or subset entity is numeric, the value is not valid.

#### See Also

## Too many object entities for phrasing <ObjectID>.

#### **Message Text**

This phrasing has too many object entities. Verb phrasings can have no more than two object entities. All other phrasings can have no more than one.

#### See Also

# Underscores used in adjective for phrasing <ObjectID>.

#### **Message Text**

Underscores are not commonly used in English adjectives. Verify that the adjectives on this phrasing are entered in the same way that users will type them in questions.

#### **See Also**

# Underscores used in high-value word for phrasing <ObjectID>.

#### **Message Text**

Underscores are not commonly used in English adjectives. Verify that the high-value words on this phrasing are entered in the same way that users will type them in questions.

#### **See Also**

# Underscores used in low-value word for phrasing <ObjectID>.

#### **Message Text**

Underscores are not commonly used in English adjectives. Verify that the low-value words on this phrasing are entered in the same way that users will type them in questions.

#### **See Also**

# Underscores used in subset word for phrasing <ObjectID>.

### **Message Text**

Underscores are not commonly used in English nouns. Verify that the subset words on this phrasing are entered in the same way that users will type them in questions.

#### **See Also**

### **Underscores used in verb for phrasing <ObjectID>.**

#### **Message Text**

Underscores are not commonly used in English verbs. Verify that the verb on this phrasing are entered in the same way that users will type them in questions.

#### See Also

## Verb "{verb}" is not in root form for phrasing <ObjectID>.

#### **Message Text**

This phrasing uses a verb that is not in its <u>root form</u>. Verbs must be entered in their root forms.

#### **Explanation**

For English Query to create a phrase correctly, verbs must be in their root form. English Query takes the root form and creates the additional forms as necessary. For example, to create a relationship to answer the question "show the customers that bought products", use the root form **buy** for the verb in your phrase. The phrasing would be *customers buy products* not *customers bought products*.

#### Action

To prevent this error:

• Always use the root form of a verb when creating a phrasing. If you are not sure of the root form, consult a dictionary.

To correct this error:

• Edit the phrasing in the relationship to change the verb to its root form.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to add a relationship

How to add phrasings to a relationship

How to create relationships using the Create Relationships command

How to modify a relationship

# Relationship/New Relationship Dialog Box Verb Phrasing Dialog Box

## **Property Error Messages**

This section provides expanded text for the property error messages.

# Data capitalization does not match field for member property <ObjectID>.

#### **Message Text**

The data capitalization specified for this member property does not match the data capitalization specified for the field associated with it.

#### See Also

## **Error in measure <ObjectID>.**

### **Message Text**

This measure contains an unexpected error.

#### **See Also**

## **Error in member property <ObjectID>.**

### **Message Text**

This member property contains an unexpected error.

#### **See Also**

## **Invalid aggregate type for measure <ObjectID>.**

### **Message Text**

The aggregate type for this measure is not valid.

#### **See Also**

## **Invalid field for measure <ObjectID>.**

### **Message Text**

The field for this measure is not valid.

#### **See Also**

## **Invalid field for member property <ObjectID>.**

### **Message Text**

The field for this member property is not valid.

#### **See Also**

## **Invalid name for measure <ObjectID>.**

### **Message Text**

The name for this measure is not valid.

#### **See Also**

## **Invalid name for member property <ObjectID>.**

### **Message Text**

The name for this member property is not valid.

#### **See Also**

## Missing aggregate type for measure <ObjectID>.

### **Message Text**

The aggregate type for this measure is not specified.

#### **See Also**

## Missing field for measure <ObjectID>.

### **Message Text**

The field for this measure is not specified.

#### **See Also**

## Missing field for member property <ObjectID>.

### **Message Text**

The field for this member property is not specified.

#### **See Also**

## Missing name for measure <ObjectID>.

### **Message Text**

The name for this measure is not specified.

#### **See Also**

# Missing source field for member property <ObjectID>.

### **Message Text**

The source field on this member property is not specified.

#### **See Also**

## **Multiple measures with ID <ObjectID>.**

#### **Message Text**

The same ID is assigned to more than one measure. Rename or remove the duplicates.

#### See Also

## **Multiple properties with ID <ObjectID>.**

#### **Message Text**

This member property has the same name as another member property. Remove one of the duplicates.

#### See Also

# Name structure does not match field for member property <ObjectID>

#### **Message Text**

The name structure specified for this member property does not match the name structure specified for the field associated with it.

#### See Also

## **Relationships Error Messages**

This section provides expanded text for relationship error messages. In some cases, an action is recommended to resolve the errors.

# Case mismatch for field {fieldname} in SQL condition for relationship <ObjectID>.

#### **Message Text**

The SQL condition for this relationship references a field for which there is no exact match in the model. There are more than one case-insensitive matches. This makes the SQL condition ambiguous. Change the case to match the desired field exactly.

#### See Also

# Case mismatch for table {tablename} in SQL condition for relationship <ObjectID>.

#### **Message Text**

The SQL condition for this relationship references a table for which there is no exact match in the model. There are more than one case-insensitive matches. This makes the SQL condition ambiguous. Change the case to match the desired table exactly.

#### See Also

# Database entity used as date for the command relationship <ObjectID>.

### **Message Text**

The date entity for this command relationship is represented by a database object. Only nondatabase entities can be used as dates in command relationships.

#### See Also

# Database entity used as duration for the command relationship <ObjectID>.

### **Message Text**

The duration entity for this command relationship is represented by a database object. Only nondatabase entities can be used as durations in command relationships.

#### See Also

# Database entity used as end date for the command relationship <ObjectID>.

### **Message Text**

The end date entity for this command relationship is represented by a database object. Only nondatabase entities can be used as end dates in command relationships.

#### See Also

# Database entity used as start date for the command relationship <ObjectID>.

### **Message Text**

The start date entity for this command relationship is represented by a database object. Only nondatabase entities can be used as start dates in command relationships.

#### See Also

# Date entity for relationship <ObjectID> is not a database entity.

# **Message Text**

The date entity for this relationship is not represented by a database object.

#### **See Also**

# **Date entity for relationship <ObjectID> is not a date.**

# **Message Text**

The date entity for this relationship is not a date.

### **See Also**

# Date unsupported by phrasing in relationship {ObjectID}.

### **Message Text**

This relationship contains a date but one of its phrasings does not support dates. Dates cannot be used with name phrasings or subset phrasings.

#### **See Also**

# **Duplicate argument {entityid} + {argtype} for relationship <ObjectID>.**

# **Message Text**

The same entity is specified more than one time in the arguments of the command relationship <ObjectID>. Remove the duplicates.

#### See Also

# **Duplicate argument ID {argid} for relationship <ObjectID>.**

# **Message Text**

The same argument is specified more than one time in this command relationship. Remove the duplicates.

#### **See Also**

# **Duplicate date entity for relationship <ObjectID>.**

# **Message Text**

Date entity for this relationship is specified more than one time. Remove the duplicates.

#### See Also

# **Duplicate default date for relationship <ObjectID>.**

# **Message Text**

This relationship has more than one default date. Remove the duplicates.

### **See Also**

# **Duplicate duration entity for relationship** < ObjectID>.

# **Message Text**

This relationship has more than one duration entity. Remove the duplicates.

#### **See Also**

# **Duplicate end date entity for relationship <ObjectID>.**

# **Message Text**

This relationship has more than one end date entity. Remove the duplicates.

### **See Also**

# **Duplicate join table for relationship <ObjectID>.**

# **Message Text**

This relationship has more than one join table. Remove the duplicates.

### **See Also**

# **Duplicate location entity for relationship <ObjectID>.**

# **Message Text**

This relationship has more than one location entity. Remove the duplicates.

### **See Also**

# **Duplicate phrasing group for relationship** < ObjectID > .

# **Message Text**

This relationship contains two identical phrasing groups. Remove one of the duplicates.

#### **See Also**

# **Duplicate role for relationship <ObjectID>.**

# **Message Text**

The same entity is specified more than one time in this relationship. This is valid only if this entity has a different join path each time it is used.

#### See Also

# **Duplicate SQL condition for relationship <ObjectID>.**

# **Message Text**

This relationship contains more than one SQL condition. Remove the duplicates.

### **See Also**

# **Duplicate start date entity for relationship** < ObjectID > .

# **Message Text**

This relationship has more than one start date entity. Remove the duplicates.

#### **See Also**

# Duration entity for relationship <ObjectID> is not a database entity.

# **Message Text**

The duration entity for this relationship is not represented by a database object.

#### **See Also**

# Duration unsupported by phrasing in relationship {ObjectID}.

#### **Message Text**

This relationship contains a duration, but one of its phrasings does not support durations. Durations can be used only with verb, preposition, and trait phrasings.

#### See Also

# End date entity for relationship <ObjectID> is not a database entity.

# **Message Text**

The end date entity for this relationship is not represented by a database object.

#### **See Also**

# End date entity for relationship <ObjectID> is not a date.

# **Message Text**

End date entity for this relationship is not a date.

#### **See Also**

# End date unsupported by phrasing in relationship {ObjectID}.

#### **Message Text**

This relationship contains an end date, but one of its phrasings does not support end dates. End dates can be used only with verb, preposition, and trait phrasings.

#### See Also

# Entity {entityid} should be the date/time of relationship <ObjectID>.

#### **Message Text**

This relationship contains an entity that is known to be a date or time but the relationship does not have a date or time specified. To enable questions about locations, the entity should be made the date/time of the relationship.

#### See Also

# Entity {entityid} should be the location of relationship <ObjectID>.

#### **Message Text**

This relationship contains an entity that is known to be a location, but the relationship does not have a location specified. To enable questions about locations, the entity should be made the location of the relationship.

#### **See Also**

# **Error** in relationship < ObjectID>.

# **Message Text**

This relationship contains an unexpected error.

### **See Also**

# Field {fieldname} without table name in SQL condition for relationship <ObjectID>.

# **Message Text**

The SQL condition for this relationship uses a field without a fully specified table name.

#### **See Also**

# Invalid argument ID on command argument for relationship <ObjectID>.

# **Message Text**

The argument ID for a command argument on this command relationship is not valid.

#### **See Also**

# Invalid argument type on command argument for relationship <ObjectID>.

# **Message Text**

Argument type for a command argument on this command relationship is not valid.

# See Also

# **Invalid command ID for relationship <ObjectID>.**

# **Message Text**

The command ID for this command relationship is not valid.

### **See Also**

# Invalid date entity for relationship <ObjectID>.

# **Message Text**

The date entity for this relationship is not valid.

### **See Also**

# Invalid default date for relationship <ObjectID>.

# **Message Text**

The default date for this relationship is not valid.

### **See Also**

# **Invalid duration entity for relationship <ObjectID>.**

# **Message Text**

The duration entity for this relationship is not valid.

### **See Also**

# Invalid end date entity for relationship <ObjectID>.

# **Message Text**

The end date entity for this relationship is not valid.

### **See Also**

# Invalid entity on command argument for relationship <ObjectID>.

# **Message Text**

The entity for a command argument of this command relationship is not valid.

#### **See Also**

# **Invalid ID for relationship <ObjectID>.**

# **Message Text**

The ID for this relationship is not valid.

### **See Also**

# **Invalid join table for relationship <ObjectID>.**

# **Message Text**

The join table for this relationship is not valid.

### **See Also**

# **Invalid location entity for relationship <ObjectID>.**

# **Message Text**

The location entity for this relationship is not valid.

### **See Also**

# **Invalid SQL condition for relationship <ObjectID>.**

# **Message Text**

The SQL condition on this relationship contains an error. Verify that the SQL condition is valid and that it uses the complete [owner.]tablename.fieldname syntax.

#### See Also

# **Invalid start date entity for relationship <ObjectID>.**

# **Message Text**

The start date entity for this relationship is not valid.

### **See Also**

# Location entity for relationship <ObjectID> is not a database entity.

# **Message Text**

The location entity for this relationship is not represented by a database object.

#### **See Also**

# Location unsupported by phrasing in relationship {ObjectID}.

#### **Message Text**

This relationship contains a location, but one of its phrasings does not support locations. Locations can be used only with verb, preposition, and trait phrasings.

#### See Also

# Missing argument ID in argument for relationship <ObjectID>.

# **Message Text**

The argument ID for an argument of this command relationship is not specified.

#### **See Also**

# Missing argument type in argument for relationship <ObjectID>.

# **Message Text**

The argument type for an argument of this command relationship is not specified.

#### **See Also**

# Missing command name for relationship <ObjectID>.

# **Message Text**

The command name for this command relationship is not specified.

### **See Also**

# Missing command phrasing in phrasing group for command relationship <ObjectID>.

#### **Message Text**

This command relationship has a phrasing group that contains no command phrasings. Each phrasing group must contain exactly one command phrasing.

#### See Also

# Missing entities for relationship <ObjectID>.

# **Message Text**

Entities for this relationship are not specified.

### **See Also**

# Missing entity in argument for relationship <ObjectID>.

# **Message Text**

The entity for an argument of this command relationship is not specified.

#### **See Also**

# Missing join path to field {fieldid} in relationship <ObjectID>.

#### **Message Text**

This relationship has a SQL condition that references a field that cannot be reached from the join table of the relationship. This indicates that either the SQL condition is incorrect or there are one or more joins missing from tables in the model.

#### See Also

# Missing join table for relationship <ObjectID>.

# **Message Text**

The join table for this relationship is not specified.

### **See Also**

# Missing path from join table of relationship <ObjectID> to entity {entityid}.

# **Message Text**

There is no join path from the join table of this relationship to one of the entities in this relationship.

#### **See Also**

# Missing phrasings for relationship <ObjectID>.

# **Message Text**

The phrasings for this relationship are not specified.

### **See Also**

# Most recent data is not supported for the relationship <ObjectID>.

#### **Message Text**

This relationship is marked to show the most recent data by default. This feature can be used only on relationships where the date entity is represented by a single field that is part of the primary key of the join table.

#### See Also

# Multifield end date entity for relationship <ObjectID>.

# **Message Text**

The end date entity in this relationship is represented by more than one field. Only single-field entities can be end dates.

# See Also

# Multifield start date entity for relationship <ObjectID>.

# **Message Text**

The start date entity is represented by more than one field in this relationship. Only single-field entities can be start dates.

# See Also

# Multiple command phrasings in phrasing group for relationship <ObjectID>.

### **Message Text**

This command relationship has a phrasing group that contains more than one command phrasing. Each phrasing group must contain exactly one command phrasing.

#### **See Also**

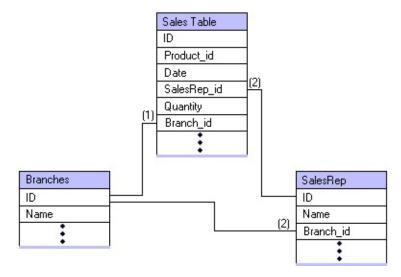
# Multiple possible paths from join table of relationship <ObjectID> to entity {entityid}.

#### **Message Text**

There is more than one possible join path from the join table of this relationship to one of the entities used in the relationship.

#### **Explanation**

English Query chooses the join path from the join table for the relationship to each entity in the relationship. If there is more than one possible join path, English Query calculates a default path unless an explicit join path is set. For the relationship, *branches sell customers products*, there is more than one path from the join table, **Sales**, to the **Branches** table. The first path (1) uses the join, **Sales.branch\_id~Branches.id**. The second path (2) uses the joins **Sales.salesrep\_id~Salesrep.id** and **Salesrep.branch\_id~Branches.id**.



#### Action

To prevent this error:

• Specify a join path if there is more than one join path from the join table of a relationship to one of the entities used in that relationship.

To correct this error, set the desired join path:

- 1. In the **Relationship/New Relationship** dialog box, on the **Semantics** tab, in the **Entities** list, double-click the role (entity) that has the ambiguous join path.
- 2. In the **<role> Role** dialog box, click **Specify**.
- 3. In the **Specify Join Path** dialog box, add the correct join path.

**Note** You must specify the complete join path from beginning to end. You may have to add multiple joins to accomplish this.

#### See Also

Compile-Time Error Messages (Authoring Object Model)

How to edit the entity's role in a relationship

< Role Name > Role Dialog Box

Semantics Tab (Relationship/New Relationship Dialog Box)

Specify Join Path Dialog Box

# Multiple quantity/amount in relationship <ObjectID>.

# **Message Text**

This relationship contains more than one entity with a quantity or amount. Only one quantity or amount can be specified in a relationship.

#### See Also

# **Multiple relationships with ID <ObjectID>.**

# **Message Text**

The same ID is assigned to more than one relationship. Rename or remove the duplicates.

#### See Also

# No relationship in OLAP cube between <ObjectID> and {objectid}.

#### **Message Text**

The OLAP database contains no relationship between these two objects. Therefore, they cannot be used together in a relationship.

#### **Explanation**

This error occurs when there are multiple OLAP cubes in a project and a relationship is created that uses entities that do not have an underlying relationship within the same OLAP cube. The relationship must contain only entities represented by OLAP objects in the same cube.

For example, in the **FoodMart** database, there is a **Sales** cube and a **Warehouse** cube. The **Sales** cube contains a **customer** dimension and a **product** dimension. The **Warehouse** cube contains a **product** dimension and a **warehouse** dimension. Using these two cubes, there can be a relationship between:

- **customer** and **product** in the **Sales** cube
- **product** and **warehouse** in the **Warehouse** cube.

There cannot be a relationship between **customers** and **warehouses** because no cube that contains both **customers** and **warehouses** exists.

#### Action

To prevent the error, create entities from:

• The same cube.

To correct this error:

• Edit or create the necessary relationships using entities that come from

the same cube.

For more information, see <u>Introduction to Dimensions</u>.

#### See Also

**Additional Relationships** 

**Analysis Services in English Query** 

Compile-Time Error Messages (Authoring Object Model)

How to create relationships using the Create Relationships command

How to edit the entity's role in a relationship

How to modify a relationship

Relationship/New Relationship Dialog Box

# Nonlocation entity used as location in relationship <ObjectID>.

### **Message Text**

The location entity for this relationship has entity type "Who" or "When". Only entities with entity type "Where" or "None" can be used as locations.

#### See Also

# Relationship <ObjectID> and relationship {relationshipid} are identical.

### **Message Text**

This relationship is identical to another relationship except for its phrasings. Create a single relationship with multiple phrasings instead of multiple relationships.

#### **See Also**

# Relationship <ObjectID> and relationship {relationshipid} have measurement phrasings that use the same measurement.

# **Message Text**

This relationship and another relationship have measurement phrasings that use the same measurement entity. These phrasings must be combined into a single phrasing in a single relationship.

#### See Also

# Relationship <ObjectID> has both a when date and an end date.

### **Message Text**

This relationship contains both a when date and an end date. End date cannot be used in a relationship with a when date.

# See Also

# Relationship <ObjectID> has both a when date and start date.

### **Message Text**

This relationship contains both a when date and a start date. Start date cannot be used in a relationship with a when date.

# See Also

# Relationship <ObjectID> has two measurement phrasings that use the same measurement.

#### **Message Text**

This relationship has two measurement phrasings that use the same measurement entity. These phrasings must be combined into a single phrasing.

#### See Also

# Relationship <ObjectID> is identical to relationship {relationshipid}.

# **Message Text**

This relationship is identical to another relationship. Remove one of the duplicates.

#### **See Also**

# Relationships <ObjectID> and {relationshipid} each cover only part of the underlying database relationship.

#### **Message Text**

Based on the keys of the join table, this relationship covers only part of the underlying relationship represented in the database. There is another relationship that also covers only part of this underlying relationship. These relationships probably contain phrasings that operate as a group. If so, these relationships should be merged, and their phrasings merged into a phrasing group.

#### **Explanation**

There are two or more relationships that include only part of the underlying database relationship. If these relationships remain independent of each other, some questions will not result in an accurate response. For example, the entities **manufacturer**, **product**, **color**, **year**, and **quantity** are used to create the following relationships:

- manufacturers make quantities of products in years
- products have colors

The relationships are based on the following underlying table.

Table: Manufactured Products

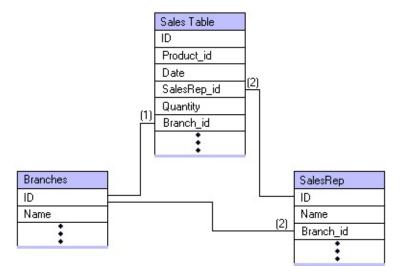
Manufacturer	Product Name	Color		Qty
Acme	bike	green	1999	6
Acme	skateboard	red	1999	4
Spacely	bike	green	1999	12
Spacely	skateboard	blue	1999	4
Acme	skateboard	green	1999	45

P Indicates primary key

If these phrases are put into two separate relationships, they will be used by English Query in the following order to generate a response to the question "How many green products did Acme make last year?":

- 1. The relationship *products have colors* will be used to find all of the products that have the color green. The results include the products, both bikes and skateboards, from all manufacturers. This phrase determines that bikes and skateboards are green.
- 2. The relationship *manufacturers make quantities of products in years* will be used to find all of the manufacturers named Acme who make bikes and skateboards and the number of these products that they made last year, regardless of color. (The color was already accounted for in the first phrase.) The second phrase does not incorporate anything about color.

The relationships are based on the:



The total quantity of products is 55. This is not the desired result because it includes red skateboards in the total.

This situation arises when the database tables are structured in a particular way and the tables are not normalized. Although this may be by design, in this case, the color of the bikes is inherently dependent on the product and the manufacturer. The only way for English Query to correctly respond to this question is for it to process both of these phrases simultaneously, instead of in sequence. If both phrases are processed simultaneously, English Query will look for green products that are made by Acme.

#### Action

#### To prevent this error:

- Create or group phrasings to cover the underlying relationship in the database.
- To correct this error, group the two phrases in one relationship:
  - 1. Double-click one of the relationships.
  - 2. In the **Relationship** dialog box, create the second phrasing.
  - 3. Select one phrasing, press CTRL, and then select the other phrasing.
  - 4. Click **Group**.
  - 5. Select the *products have colors* check box.

A red check mark appears to the left of the phrasing, indicating that the phrase is mandatory. The phrase must be part of this relationship when the relationship is used.

6. Delete the other relationship.

In the example above, you would:

- 1. Create one relationship that contains both of the phrasings, *products* have colors and manufacturers make quantities of products in years.
- 2. Make them a group.
- 3. Specify both phrases as mandatory.

Creating a group forces both phrases to be used simultaneously if either one of the phrases is used. Then, English Query will search for products that are both green and made by Acme.

### See Also

Compile-Time Error Messages (Authoring Object Model)

**Grouped Phrasings Examples** 

How to add a relationship

Relationship/New Relationship Dialog Box

# Role {roleid} is not marked to always display and does not appear in each phrasing group in relationship <ObjectID>.

#### **Message Text**

This role does not appear in all the phrasing groups in the relationship and is not marked as Always Show. Every role that is not used must be marked as Always Show.

#### See Also

# Start date entity for relationship <ObjectID> is not a database entity.

#### **Message Text**

The start date entity for this relationship is not represented by a database object.

#### **See Also**

# Start date entity for relationship <ObjectID> is not a date.

### **Message Text**

The start date entity for this relationship is not a date.

#### **See Also**

# Start date unsupported by phrasing in relationship {ObjectID}

#### **Message Text**

This relationship contains a start date, but one of its phrasings does not support start dates. Start dates can be used only with verb, preposition, and trait phrasings.

#### **See Also**

# The entity {entityid} is used as both the {usage} and the {usage} in the relationship <ObjectID>.

#### **Message Text**

The same entity is used more than once as a time or location property of this relationship. Each entity can be used once as a time or location property in a relationship.

#### See Also

# Too many phrasings in relationship <ObjectID>.

### **Message Text**

This relationship contains more than 32 phrasings. A maximum of 32 phrasings is allowed in each relationship.

#### See Also

# **Too many roles in relationship <ObjectID>.**

### **Message Text**

This relationship contains more than 32 roles. A maximum of 32 roles is allowed in each relationship.

#### See Also

# Undefined field {fieldname} in SQL condition for relationship <ObjectID>.

### **Message Text**

The SQL condition for this relationship references an undefined field.

#### **See Also**

# Undefined table {tablename} in SQL condition for relationship <ObjectID>.

#### **Message Text**

The SQL condition for this relationship references an undefined table.

#### **See Also**

# **Ungrouped noncommand phrasing in command relationship <ObjectID>.**

#### **Message Text**

This command relationship contains a noncommand phrasing that is not part of a phrasing group. All noncommand phrasings must be grouped with a command phrasing.

#### **See Also**

# Unsupported quantity/amount in phrasing {phrasingid} in relationship <ObjectID>.

#### **Message Text**

One of the phrasings in this relationship does not support quantities and amounts. However, one of the roles in the relationship has a quantity or amount. Amounts and quantities can be used only with verb, preposition, and trait phrasings.

#### See Also

# **Role Error Messages**

This section provides expanded text for the role error messages.

# **Duplicate amount entity for role <ObjectID>.**

### **Message Text**

This role has more than one amount entity. Remove the duplicates.

#### **See Also**

# **Duplicate join in join path for role <ObjectID>.**

### **Message Text**

The same join is used more than one time in the join path for this role. Remove the duplicates.

#### See Also

# **Duplicate join path for role <ObjectID>.**

# **Message Text**

This role has more than one join path. Remove the duplicates.

#### **See Also**

# **Duplicate nondatabase entity data type for role** <**ObjectID>.**

### **Message Text**

This role has more than one nondatabase entity data type. Remove the duplicates.

#### **See Also**

# **Duplicate quantity entity for role <ObjectID>.**

# **Message Text**

This role has more than one quantity entity. Remove the duplicates.

#### **See Also**

# Entity {entityid} used as amount of itself in role <ObjectID>.

### **Message Text**

This role has an amount entity that is identical to the role's entity. The amount entity must be different from the role entity.

### See Also

# Entity {entityid} used as quantity of itself in role <ObjectID>.

### **Message Text**

This role has a quantity entity that is identical to the role's entity. The quantity entity must be different than the role entity.

#### **See Also**

# Error in role <ObjectID>.

# **Message Text**

This role contains an unexpected error.

# See Also

# **Invalid amount field for role <ObjectID>.**

# **Message Text**

The amount field for this role is not valid.

#### **See Also**

# **Invalid entity for role <ObjectID>.**

# **Message Text**

The entity for this role is not valid.

#### **See Also**

# **Invalid ID for role <ObjectID>.**

# **Message Text**

The ID for this role is not valid.

#### **See Also**

# **Invalid join in join path for role <ObjectID>.**

# **Message Text**

A join in the join path for this role is not valid.

#### **See Also**

# **Invalid join path for role <ObjectID>.**

# **Message Text**

The join path for this role is not valid.

#### **See Also**

# Invalid nondatabase entity data type for role <ObjectID>.

# **Message Text**

The nondatabase entity data type for this role is not valid.

#### **See Also**

# **Invalid quantity entity for role <ObjectID>.**

# **Message Text**

The quantity entity for this role is not valid.

#### **See Also**

# Missing entity for role <ObjectID>.

# **Message Text**

The entity for this role is not specified.

#### **See Also**

# Missing join path for role <ObjectID>.

### **Message Text**

More than one possible path exists to this role from the join table of the relationship; therefore, no default join path is available. Add a join path to the role.

#### See Also

# Missing joins in join path for role <ObjectID>.

# **Message Text**

The join path for this role is not specified.

#### **See Also**

# Missing nondatabase entity data type for role <ObjectID>.

# **Message Text**

The nondatabase entity data type for this role is not specified.

#### **See Also**

# Multifield entity {entityid} used as amount on role <ObjectID>.

# **Message Text**

This role has an amount entity that is represented by more than one field. Amount entities must be single-field entities.

# See Also

# Multifield entity {entityid} used as quantity on role <ObjectID>.

# **Message Text**

This role has a quantity entity that is represented by more than one field. Quantity entities must be single-field entities.

#### **See Also**

# **Multiple roles with ID <ObjectID>.**

# **Message Text**

The same ID is assigned to more than one role. Rename or remove the duplicates.

#### **See Also**

# Nonmeasure entity {entityid} used as amount on role <ObjectID>.

# **Message Text**

This role has an amount entity that does not have the entity type "measure". Amount entities must be measure entities.

#### See Also

# Nonmeasure entity {entityid} used as quantity on role <ObjectID>.

# **Message Text**

This role has a quantity entity that does not have the entity type "measure". Quantity entities must be measure entities.

#### See Also

# Role <ObjectID> has a nondatabase entity data type in a noncommand relationship.

### **Message Text**

This role has a nondatabase entity data type. Nondatabase entity types are valid only for roles in command relationships.

#### See Also

# Role <ObjectID> is a database entity but has a nondatabase entity data type.

# **Message Text**

This role has a nondatabase entity data type but is not a nondatabase entity.

#### **See Also**

# Role <ObjectID> is mandatory, but has no name phrasing.

# **Message Text**

This role is mandatory, but it is a table entity and does not have a name phrasing.

#### **See Also**

# Role <ObjectID> is mandatory, but is not a number, name, or date.

# **Message Text**

This role is mandatory, but the entity is not a number, name, or date.

#### **See Also**

# Role <ObjectID> is marked to always display in a command relationship.

# **Message Text**

This role is marked as Always Show. Always Show is not valid in command relationships.

#### **See Also**

# Role <ObjectID> is not marked to always display and does not appear in each phrasing group.

# **Message Text**

This role is not marked as Always Show, and it does not appear in each phrasing group in the relationship.

#### See Also

# Role <ObjectID> is not marked to always display and does not appear in phrasing {phrasingid}.

#### **Message Text**

This role does not appear in all phrasings in the relationship and is not marked as Always Show. Each role that does not appear in all phrasings must be marked as Always Show.

#### See Also

# Source/destination mismatch in join path for role <ObjectID>.

# **Message Text**

The source table of a join in the join path on this role does not equal the destination table of the previous join (or join table).

# See Also

# Table Error Messages

This section provides expanded text for the table error messages.

# **Duplicate field in table <ObjectID>.**

# **Message Text**

The same field is used more than one time in this table. Remove the duplicates.

### **See Also**

# Error in table <ObjectID>.

# **Message Text**

This table contains an unexpected error.

# See Also

# **Invalid name for table <ObjectID>.**

# **Message Text**

The name for this table is not valid.

#### **See Also**

# **Missing fields for table <ObjectID>.**

# **Message Text**

The fields for this table are not specified.

### **See Also**

# Missing keys for table <ObjectID>.

# **Message Text**

The keys for this table are not specified.

### **See Also**

# **Multiple tables with ID <ObjectID>.**

# **Message Text**

The same ID is assigned to more than one table. Rename or remove the duplicates.

#### See Also

# **Various Error Messages**

This section provides expanded text for error messages that occur in a variety of situations.

# The ID of {objecttype} <ObjectID> is too long.

# **Message Text**

This object has an ID that exceeds the maximum ID length.

### **See Also**