Building SQL Server Applications

Building SQL Server Applications Overview

Application Programming Interfaces (APIs) are the mechanisms used by applications to access resources on the local computer or available through a network. Microsoft® SQL Server[™] 2000 supports several classes of APIs that applications can use to access SQL Server resources:

- General database access APIs allow applications to work with the data in a relational database. The APIs present results to applications in one of two forms:
 - Tabular result sets, which some APIs call rowsets.
 - XML documents, which are the preferred way of representing data in Internet applications.
- SQL Server database services APIs allow applications to administer and configure the services included with the relational database engine, such as replication and Data Transformation Services (DTS).
- The Analysis Services API gives applications access to the OLAP and data mining facilities of Analysis Services. For more information, see <u>Programming Analysis Services Applications</u>.
- The Meta Data Services API gives applications access to the repository of SQL Server meta data stored in Meta Data Services. For more information, see <u>Programming Meta Data Services Applications</u>.
- The English Query API provides applications the ability to pass customer questions, written in English, about information in a database or OLAP cube to the English Query engine. The engine returns a Transact-SQL statement or MDX query that can be executed to answer the question. For more information, see <u>Developing and Deploying English Query Applications</u>.

For information about additional considerations regarding the use of the APIs supported by SQL Server 2000, see <u>Application Development Architecture</u>.

General Database Access APIs

Database applications generally deal with data in one of two formats:

- Tabular result sets, which are sometimes called rowsets. The application uses a database API to execute a Transact-SQL statement and process any result sets that may be returned. These APIs support result set processing: ADO, OLE DB, ODBC, Embedded SQL for C, and DB-Library.
- XML documents. The application uses an API or Universal Resource Locator (URL) to execute a Transact-SQL statement or XPath query. The application then retrieves any XML document that is returned. These access methods support XML documents: ADO, URLs, OLE DB.

While result set and XML processing is typically discussed in relation to retrieving the results of a command, result sets and XML documents can both be used as the source of data for modifications of database tables:

- An application using tabular result sets can open a cursor over a result set, and use data from the cursor to modify data in tables.
- An application using XML documents can use **sp_xml_preparedocument** to add a document to the database, and then use OPENXML to retrieve data from the document. The retrieved data can be used to modify data in tables.

Most of the general database APIs supported by SQL Server are of two types:

• An object database API uses an object model comprised of objects, properties, and interfaces an application uses to connect to a database, pass commands to the database, and retrieve results.

• A C database API is a set of C functions an application calls to connect to a database, pass commands to the database, and retrieve results.

In addition, SQL Server 2000 can be accessed from URLs in Internet applications. URLs are formatted strings, or stream objects, that Internet applications use to access resources available through the Internet or an enterprises intranet. SQL Server 2000 supports URLs that specify Transact-SQL statements, query templates, or XPath queries.

Any SQL commands sent to SQL Server 2000 through the database APIs or URLs must comply with the Transact-SQL language. Transact-SQL complies with the Entry Level of the SQL-92 standard, and in addition, supports powerful extensions to SQL-92. The SQL Server OLE DB provider and SQL Server ODBC driver also support the ODBC SQL specification. For more information, see <u>Transact-SQL Overview</u>.

| Торіс | Description |
|--------------------------|--|
| Programming ADO SQL | COM API recommended as the primary API |
| Server Applications | for accessing data from general business |
| (Microsoft ActiveX® Data | applications, such as human resources, |
| Objects) | accounting, and marketing applications. ADO |
| | encapsulates the OLE DB API in a simplified |
| | object model that reduces application |
| | development and maintenance costs. The SQL |
| | Server OLE DB provider is the preferred |
| | provider to use in ADO applications that |
| | access SQL Server. ADO, similar to OLE DB, |
| | can access data from many sources, not just |
| | SQL databases. In SQL Server 2000, ADO |
| | supports XML document processing in |
| | addition to relational result set processing. |
| <u>URL Access</u> | Formatted strings or stream objects used by |
| | Internet applications to access resources |
| | available on the Internet or intranet. SQL |
| | Server 2000 supplies an ISAPI .dll that Internet |
| | Information Services (IIS) supports references |
| | |

These are the general database APIs supported by SQL Server 2000.

| | to SQL Server 2000 from URLs. |
|------------------------|--|
| OLE DB and SQL Server | Strategic, low-level, COM API for accessing |
| | data. OLE DB is recommended for developing |
| | tools, utilities, or low-level components that |
| | need high performance. The SQL Server OLE |
| | DB provider is a native, high performance |
| | provider that accesses the SQL Server TDS |
| | protocol directly. In SQL Server 2000, OLE |
| | DB supports XML document processing in |
| | addition to relational result set processing. |
| Programming ODBC SQL | Open C API designed to access data in SQL |
| Server Applications | databases. The SQL Server ODBC driver is a |
| (Open Database | native, high-performance driver that directly |
| Connectivity) | accesses the SQL Server TDS protocol. |
| Programming Embedded | Standard API defined for accessing SQL |
| <u>SQL for C</u> | databases from C or COBOL applications. |
| Programming DB-Library | Legacy C API designed to work with SQL |
| <u>for C</u> | Server. |

Through its support of ODBC, SQL Server 2000 also supports applications written to the Remote Data Objects (RDO) and Data Access Objects (DAO) APIs. These are object APIs that encapsulate ODBC. They are not discussed further in SQL Server Books Online; programmers using RDO and DAO should refer to ODBC and SQL Server 2000 for implementation details for the SQL Server ODBC Driver.

SQL Server Books Online topics about ADO, OLE DB, and ODBC do not cover the full functionality of those APIs. The topics cover only the issues specific to those APIs when you are using the SQL Server OLE DB provider or the SQL Server ODBC driver. They assume that you are familiar with the general concepts for the API you are using, and that you have access to the documentation for the API. You can download the documentation for ADO, OLE DB, and ODBC at <u>Microsoft Web site</u>.

Microsoft Distributed Transaction Coordinator (MS DTC) is a component that allows applications to define distributed transactions. Distributed transactions protect the integrity of a series of updates made against multiple servers. SQL

Server 2000 database applications can initiate distributed transactions themselves by calling the MS DTC API directly, but the SQL Server database engine can also call MS DTC to implement the functionality required by distributed Transact-SQL statements executed by applications. For more information, see <u>MS DTC Distributed Transactions</u>

SQL Server Database Services APIs

SQL Server 2000 supports APIs that allow applications to configure and administer the DTS and replication components of SQL Server. Applications can use the same administration and configuration API, SQL-DMO, which the SQL Server tools use when managing instances of SQL Server.

| Торіс | Description |
|-------------------------|---|
| DTS Programming | Set of COM interfaces (based on OLE DB) for |
| <u>Reference</u> | defining and executing complex data |
| (Data Transformation | transformations between OLE DB data |
| Services) | providers. |
| MS DTC Distributed | Component that allows applications to define |
| <u>Transactions</u> | distributed transactions that protect the |
| | integrity of a series of updates made against |
| | multiple servers. Applications use the |
| | transaction commands of an API or Transact- |
| | SQL, the API or SQL Server 2000 interface |
| | with MS DTC to implement the distributed |
| | transactions. |
| Programming Extended | C API for writing SQL Server extended stored |
| Stored Procedures | procedures. |
| Getting Started with | Set of COM interfaces for defining and |
| Replication Programming | managing replication between instances of |
| | SQL Server databases. You can also replicate |
| | data from heterogeneous third-party databases |
| | to SQL Server. |
| Developing SQL-DMO | Set of COM interfaces for managing and |
| Applications | administering SQL Server 2000. |
| (SQL Distributed | |
| Management Objects) | |

For information about additional considerations regarding the use of the APIs discussed, see <u>Application Development Architecture</u>.

Through its support of ODBC, SQL Server 2000 also supports applications written to the Remote Data Objects (RDO) and Data Access Objects (DAO) APIs. These are object APIs that encapsulate ODBC. They are not discussed further in SQL Server Books Online; programmers using RDO and DAO should refer to ODBC and SQL Server 2000 for implementation details for the SQL Server ODBC Driver.

SQL Server Books Online topics about ADO, OLE DB, and ODBC do not cover the full functionality of those APIs. The topics cover only the issues specific to those APIs when you are using the SQL Server OLE DB provider or the SQL Server ODBC driver. They assume that you are familiar with the general concepts for the API you are using, and that you have access to the documentation for the API. You can download the documentation for ADO, OLE DB, and ODBC at <u>Microsoft Web site</u>. Building SQL Server Applications

Connecting Early Version Clients to SQL Server 2000

Microsoft® SQL Server[™] 2000 supports connections from client applications that use the client software of SQL Server version 7.0 or earlier. SQL Server 2000 supports these connections in a compatibility mode. If a database is migrated to SQL Server 2000 from SQL Server 4.21a, 6.0, 6.5, or 7.0, the applications using the database can continue working with few, if any, changes.

SQL Server uses an application level protocol called Tabular Data Stream (TDS) to communicate between applications and instances of SQL Server. SQL Server 2000 supports these versions of TDS:

- Applications using the SQL Server 2000 versions of the OLE DB provider for SQL Server or the SQL Server ODBC driver communicate using TDS 8.0.
- Applications using the SQL Server 7.0 versions of the OLE DB provider for SQL Server or the SQL Server ODBC driver communicate using TDS 7.0.
- Applications using the SQL Server ODBC driver from SQL Server versions 6.5, 6.0, or 4.21a communicate using TDS 4.2. These versions of SQL Server did not include OLE DB providers.
- Application using any version of DB-Library or Embedded-SQL for C communicates using TDS 4.2. The DB-Library .dll used by these two APIs has not been enhanced since SQL Server 6.5, so even the .dll versions included with SQL Server 2000 and SQL Server 7.0 still use TDS 4.2.
- Application using any version of DB-Library or Embedded-SQL for C communicates using TDS 4.2. The DB-Library .dll used by these two APIs has not been enhanced since SQL Server 6.5, so even the .dll versions included with SQL Server 2000 and SQL Server 7.0 still use

TDS 4.2.

Applications using TDS 7.0 cannot access all features introduced in SQL Server 2000. The new features not available to TDS 7.0 clients include:

- TDS 7.0 does not support **sql_variant** data. **sql_variant** data values are returned as **nvarchar(4000)** values to TDS 7.0 clients.
- TDS 7.0 does not support **bigint** data. **bigint** data values are returned as **decimal(19,0)** values to TDS 7.0 clients.
- TDS 7.0 does not support column-level collations. SQL Server 2000 always reports the instance default collation back to TDS 7.0 clients.

Applications using TDS 4.2 cannot access all features introduced in SQL Server 7.0 and SQL Server 2000. The new features that are not available to TDS 4.2 clients include:

- TDS 4.2 does not support **sql_variant** data. **sql_variant** data values are returned as **varchar(255)** values to TDS 4.2 clients.
- TDS 4.2 does not support **bigint** data. **bigint** data values are returned as **float** values to TDS 4.2 clients.
- TDS 4.2 does not support column-level collations. SQL Server 2000 always reports the instance default collation back to TDS 4.2 clients.
- TDS 4.2 does not support XML document processing. TDS 4.2 applications attempting to execute SELECT statements with a FOR XML clause will receive an error.
- **char**, **varchar**, **nchar**, **nvarchar**, **binary**, and **varbinary** values longer than 255 bytes are truncated to 255 bytes.

- TDS 4.2 does not support Unicode. **nchar** and **nvarchar** values are converted to **char** and **varchar** using the non-Unicode Windows® code page of the server, with possible loss of extended characters. **ntext** values cannot be retrieved.
- **uniqueidentifer** data types are converted to **varbinary(16)**.
- NULL values in **bit** columns are returned as 0. Catalog and meta data functions report all **bit** columns as NOT NULL because NULL was not allowed for **bit** columns in SQL Server version 6.5 and earlier.
- Catalog and meta data functions do not report the SQL Server 2000 data types (**nchar**, **nvarchar**, **ntext**, or **uniqueidentifier**).
- In SQL Server 6.5 and earlier, **sysname** was defined as **varchar(30)**. In SQL Server 2000, **sysname** is defined as **nvarchar(128)** to allow for longer identifiers that contain more extended characters. TDS 4.2 clients cannot access SQL Server 2000 objects that have names more than 30 characters in length or that include characters not represented in the Windows code page on the client computer.

If any of these features are introduced into a SQL Server database, applications running with earlier versions of the SQL Server client software must be upgraded before they can access these new features.