software AG

TECHniques
ISSUE 3 · JULY 2018

CONNX
SQL ACCESS TO MAINFRAME DATA

By Larry McGhaw, VP, R&D CONNX, Software AG

Techie Deep Dive

Mainframe Data

Over the decades, many have predicted the death of the mainframe, and all of those predictions were wrong. IBM®, the leader in the mainframe industry, had annual revenues of nearly $80 billion in 1997. Twenty years later, IBM still had annual revenue of $80 billion (excluding inflation). When factoring in inflation, revenue for mainframe has decreased a bit over time. However, the sheer size of the revenue number shows that mainframes will be a part of major corporations for decades to come. The average age of mainframe software engineers and administrators is increasing and reaching that critical point of retirement. Additionally, the number of software engineers learning about mainframe technology (COBOL, VSAM™, IMS™, Adabas) out of college is at an all-time low. This creates an ever-growing gap between available resources and the number of resources required to maintain and enhance these critical systems running on the mainframe.

Using SQL to bridge the gap

SQL itself is a very mature technology, but it is still taught in universities and is very much a part of modern software development. It is the primary way to access modern relational enterprise databases such as Oracle®, SQL Server® and DB2. The concepts in SQL are simple and easy to understand—and these concepts can be applied to access data from a variety of databases from disparate vendors. There are several data access APIs based on SQL. For Java® – based applications there is JDBC®. For other programming languages, there are ODBC, OLE DB and Microsoft® .NET. The combination of SQL along with these data-access APIs has enabled entire segments of the business integration market to thrive. Thousands of applications access data generically using SQL through ODBC and JDBC. CONNX bridges the gap between these modern data-access methods, and the legacy data sources that exist on the mainframe that do not support SQL nor ODBC/JDBC natively, such as Adabas, VSAM and IMS.

Using CONNX to virtualize mainframe data via SQL

All SQL-based databases have an information schema. This is a metadata repository that contains information about all of the object in the database and their attributes. In SQL databases, there is the concept of a table, which contains records with an identical structure of information. The structure is defined with a collection of columns—where each column specifies the exact attributes of a piece of data. (Is it a date? Is it a text field? What is the maximum length? etc.) The definition of the structure of tables and columns is (in general) uniform across all SQL databases regardless of vendor. It is this uniformity that enables applications developers to use SQL to access data without knowing ahead of time the particular details of the database vendor. There are of course exceptions to this, and each vendor has its own set of extensions to the ANSI SQL syntax. But all enterprise relational database vendors support the same core ANSI SQL syntax. In order for CONNX to provide SQL access to mainframe data, an equivalent SQL catalog must be created that describes the mainframe data being accessed. In CONNX, the catalog is called the CONNX Data Dictionary (CDD).

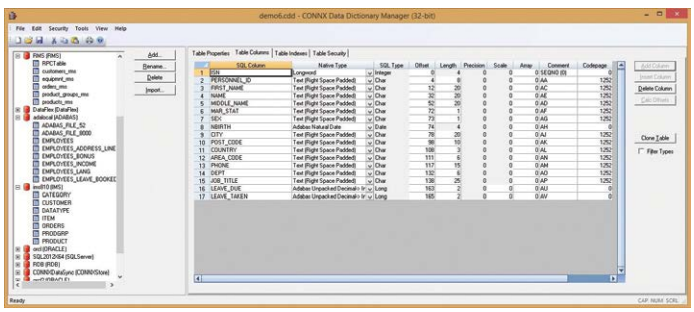

Figure 1: CONNX Data Dictionary Entry for Adabas File

The CDD contains all of the information necessary to access the mainframe data sources and all of the metadata required to represent the data as a relational table. It contains the mapping between SQL data types, SQL table names and SQL column names, and the physical field/offset/data type information necessary to access the mainframe data. Legacy mainframe applications, such as COBOL applications, actually have a very similar, much older concept called a COBOL copybook. This copybook enables COBOL applications to refer to legacy data using meaningful names and also describes the format of the data being stored or retrieved. CONNX capitalizes on the existence of these copybooks and enables imports into the CDD directly from mainframe copybooks. Mainframe Adabas has a similar concept, a Natural DDM, which provides longer descriptive names to the fields in an Adabas file. CONNX imports metadata from this format along with dozens of others.

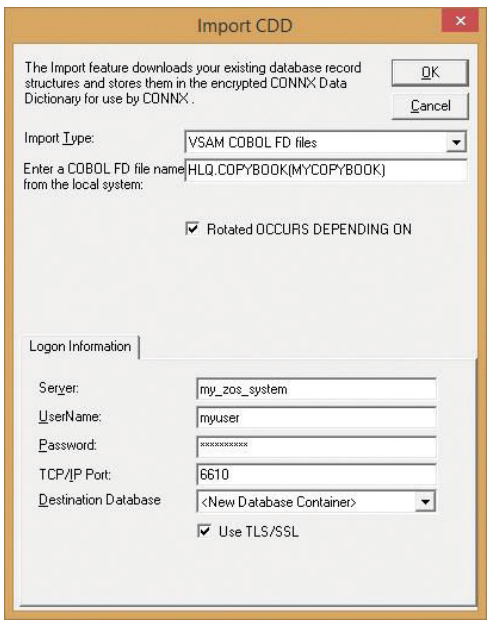

Figure 2: Importing Data from COBOL Copybook

While the data for many companies is on the mainframe, the integration points may not be on the mainframe. An integration point could be a data lake/mart/warehouse or a web-based application. If the integration point is not on the mainframe, then additional challenges arise. Linux®, UNIX® and Windows® (LUW) systems store data differently than mainframe systems. For example, most mainframe text-based data is encoded in something call Extended Binary Coded Decimal Interchange Code (EBCDIC for short). On LUW systems, either Unicode® or ASCII is more common for text data. The bytes of integer numbers are stored in opposite order on the mainframe versus most LUW systems. And, there are many other difference between the platforms. CONNX smooths over all of these differences and complexities by storing all of the required information to properly read and write the numeric, character and date data in the native mainframe format, while also providing the data in a standard SQL format.

Application developers do not have to worry or care about the fact that the data is physically being stored differently on the mainframe; it makes it completely transparent to them once the CDD has been defined.

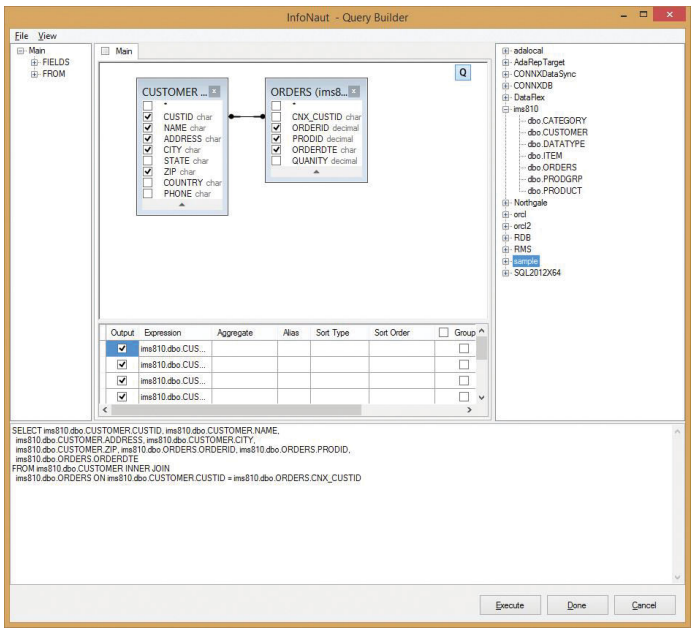

Figure 3: Query Builder to Join SQL Tables

CONNX not only provides singular access to a particular mainframe data source, but it also makes all of the mainframe data sources appear as if they are in a single virtual database. Organizations that have data in both VSAM and in IMS, for example, can use a single connection to query, combine or move data from the data sources.

Securing access to the mainframe data

A foremost concern companies have is that someone will obtain unauthorized access to the data on the mainframe. CONNX addresses that concern head-on with several security features and concepts. Authentication is the first line of defense. In order to access the data on the mainframe via CONNX, you must first provide the correct mainframe authentication credentials. Another way of putting this is, if you do not have the credentials to log on to the mainframe via a green screen/terminal emulator, you cannot do so with CONNX either. This ensures that any existing user-based security restrictions on the mainframe are honored. The second line of defense is the CONNX global security setting for the CDD. All CDDs are read/only by default, and no-write permission is allowed unless explicitly granted. This can be configured at the CDD level. The third line of defense is the CONNX SQL security layer. Relational databases have the concept of security—where you can specify which users have select, insert, update and delete access to which table. CONNX provides

this same SQL security mechanism to all of the mainframe data sources. CDD administrators can define exact access permissions for either individual users or groups of users. CONNX views are the fourth line of defense. Some organizations want to tightly control not only what columns of data individual users can see but also which rows are visible. This is possible with a CONNX view, where the name/ID of the currently logged on user can be joined to a field within a table to filter which rows are returned. This type of view will return different rows depending on who has logged on—ensuring that the logged-on user can only see the rows belonging to them. The fifth line of defense is the encryption of sensitive data when transmitted across TCP/IP. All CONNX TCP/IP connections can be encrypted with state-of-the-art TLS/SSL. And even when TLS/SSL is not enabled, any sensitive information (like username and password) is encrypted at all times.

Unified access from SQL-based applications

After a CONNX CDD has been established and secured, it enables the querying of mainframe data. Java applications and frameworks use the JDBC interface to access data. Establishing a connection to CONNX just requires authentication information, the name of the data source (CDD) and the name of the JDBC Server. The JDBC Server is the CONNX service for Java that does all of the data access and SQL query processing. This service runs on any LUW platform. The follow code snippet shows one way a Java application connects to CONNX (other connection methods are possible as well):

```
try {
 Class.forName("com.Connx.jdbc.TCJdbc.
TCJdbcDriver").newInstance();
} catch (Exception ex) {
 ex.printStackTrace();
}
Properties info = new Properties();
String connString =
"jdbc:connx:DD=<connx_datasource_
name>;port=7500;GATEWAY=<connx_jdbcserver_name_
or_ip>";
info.put("user", "my_userid");
info.put("password", "my_password");
connectionObj = DriverManager.
getConnection(connString, info);
```

Once the connection has been established, the application can use standard SQL to query the data, or update the data if permissions permit. CONNX delivers transaction capability to all mainframe data sources that support rollback capability (which is true of Adabas, VSAM under CICS, and IMS).

CONNX also supports ODBC access to the data from all LUW platforms, and the same authentication and encryption features are available through this data access API.

CONNX provides a query tool called Infonaut, which visualizes the data returned from SQL statements.

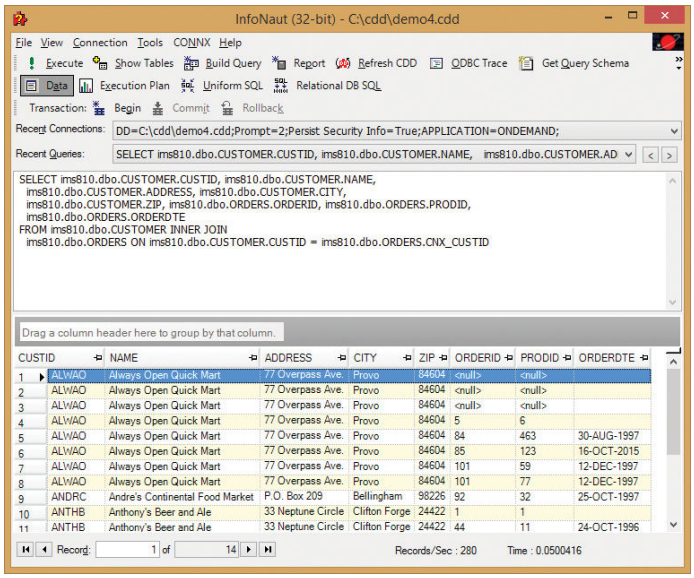

Figure 4: Showing Query Results

Bridging the gap

CONNX enables a new generation of software engineers who are training and familiar with SQL to access mainframe data. Regardless of whether a company is building a new application, extending an existing one, or integrating mainframe data into a data lake, CONNX makes the task secure, easy and fast.

To find out more about CONNX, contact your Software AG sales representative and visit www.softwareag.com/connx.

SAG_CONNX_SQL_Access_TECHniques_Jul18

software AG