Using ANSI SQL

by Sophia Tutorial

WHAT'S COVERED

This tutorial explores how the American National Standards Institute created an SQL standard that is required to be implemented by commercial database management systems, in three parts:

1. Standard SQL
2. Changing Databases
3. Standards in the Real World

1. Standard SQL

The American National Standards Institute has defined an ANSI SQL standard. It is a standard that is accepted by the International Organization for Standardization (ISO), which is a consortium of national standard bodies that is accepted by more than 150 countries. Although database vendors try to keep to the ANSI SQL standards, each database has its own enhancements. Fo example, as we saw in the last tutorial, SQL Server and Oracle have their own added features and functionalities in T-SQL and PL/SQL.

2. Changing Databases

In almost any switch between databases for applications, there are some changes that are needed to accommodate the new database. It is important to note that PostgreSQL, the database that we work with, attempts to stick to the ANSI SQL standards as closely as possible. So, it will be much easier moving from PostgreSQL to another database option versus starting with a different dialect.

To be compliant with the ANSI standards, there are a specific set of commands that must be implemented by the database in the same way. This includes the SELECT, UPDATE, DELETE, INSERT, and WHERE commands. Every database has its own dialect, but in some databases like MySQL, you can enable strict ANSI SQL mode to avoid users being able to use non-ANSI SQL code. This can be especially useful if there may be unwanted results that may be possible in non-standard code. It also allows an easier shift to other dataset products if needed.

One common difference in data types between ANSI SQL and other SQL dialects is the implementation of date and time data types. ANSI SQL defines them to be separate data types. However, databases like Oracle and SQL Server have a date data type that has both a date and time stored in it, but no separate time data type. As such, the code in these databases cannot be ported between the databases without modifications to reflect them.

3. Standards in the Real World

In a real-world setting, it is not crucial to focus on ANSI compliance in an application, as you will be using the SQL code specifically for a product and purpose. One question you may have is why databases don't support pure ANSI SQL and leave it at that. One of the big reasons is that the standards generally lag behind technological needs. Different database vendors want to be able to respond to their customer needs and differentiate themselves from the competition. There may be things like encryption, XML, or JSON support that may not make it into the standards, but the need is there.

The opposite is also true, where the standards have gotten so complex that not all standards are implemented. The standards do not include certain behaviors like indexing or file storage, but rather allow the database vendors to decide how those should work. And since the standards are always being updated, database vendors may have already implemented certain features that conflict with a newer standard. If they made changes to support the new standard, it could have a large impact on customers that depend on that feature.

So, from a database vendor perspective, it's not always in their best interests to stick only with pure ANSI SQL standards. Doing so would also make it easy for customers to move from their database to another database. Having features and functionality that are only available on their database allows them to lock an application in. Any changes that must be implemented will take a lot more work and effort to make that transition.

SUMMARY

ANSI SQL is an SQL standard that all vendors build off of within their database.

Source: Authored by Vincent Tran