Douglas Neri

February 3, 2021

IT FDN 130 A Wi 21: Foundations of Databases & SQL Programming

Assignment -06

Views

Introduction

This paper will introduce you to views in SQL, how to use them, what they are for, and how they compare to other SQL processes like Functions and Stored Procedures. As a whole Views are one of the most important aspects of SQL programming, especially once you begin working in teams or with materials that will be seen by more than just you.

Explain when you would use a SQL View.

Views are used for many reasons, but a few simple reasons are enhanced security, enhanced integrity, creating a more user friendly interface, and as a portal to make changes and updates easier.

Enhanced security and database integrity is crucial to any user, and thus so should views be crucial to users. With adding in views, a programmer can isolate the data that is visible to the various entities that intend on using the data (columns, rows, or both). Consider all of the various levels of HR functions at a company, and that some HR functions (payroll for example) don’t necessarily need access to every piece of information (like life insurance dependents) that a single employee may have on file. Scenarios like this are common for every field, but especially services where personal information is involved (financial, medical, etc).

The integrity of the database and tables is a fairly simple concept, views provide you another opportunity to ‘lock down the data’. Without a view (or some other abstraction layer) a programmer provides direct access to the database and tables, which means that if the right permissions are not in place, then there is a risk the database and tables could be changed by an outside user. This provides a significant risk to the integrity of the data.

One of the simpler concepts, but very important and powerful features is utilizing column aliasing within a view for easier data consumption. When data is collected it doesn’t necessarily have the easiest column headings. Depending on where the data comes from there could be a heavy reliance on industry jargon or acronyms that wouldn’t necessarily be easy for a common user to figure out. An easy example is the NHANES dataset. This dataset is used frequently within the medical community and is extremely acronym heavy. A simple example is that BMI is referenced fairly often. BMI stands for Body Mass Index and is essentially a comparison of a person’s height and weight (there is more to it than that, but for this example we need not go into further detail). NHANES also is a significant database, so the task of determining what each column means and their relationship to the others could be very tedious without good column heading aliasing.

One of the final ways someone would use a view in SQL would be to help with making future changes and updates within the database and tables easier. By using views a programmer can alleviate the pitfalls of a full scale database or table change by mimicking the information that will be retired while adding the new. An example would be in Major League Baseball beginning to track a new stat category. Back in 2009 a statistician determined that one could calculate a players Wins Above Replacement (WAR). Essentially the statistic determines how many wins (positive or negative) a team would have by replacing the player. Why this matters to views is that Major League Baseball began tracking a new stat and subsequently needed to change their database. I do not know how they actually changed their database, but a smart way would have been to use a view so that the entire fantasy baseball industry didn’t need to overhaul itself to adjust to the new database or tables that MLB created.

Comparing a Stored Procedure, Function, and View.

A view is a way to create a virtual table (not an actual table) that joins rows or columns to change the way the data in the table is seen. Functions and stored procedures are like each other in that they both work through steps that can have a result (or not) of something other than a view, like an integer. The individual differences of stored procedures, functions, and views is complex and has a wide array of possibilities, so only a few are listed in Figure 1 below.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Stored Procedure | Function | View |
| Must Return Value |  | X | X |
| Must use Returns Argument |  | X |  |
| Can change the database | X |  |  |
| Must have one parameter |  | X |  |
| Can use Transactions | X |  |  |
| Input Parameter allowed | X | X |  |
| Output Parameter Allowed | X |  |  |
| Can use in Select/Where/Having |  | X |  |
| Can use Try-Catch blocks | X |  |  |
| Can Restrict Access to rows or columns |  | X | X |
| Insert, Update, Delete table data | X |  | X |
| Used as a building block in a larger query |  | X | X |

Figure 1 – Table comparison of Stored Procedures, Functions, and Views

As you can see, each element has its own nuance, but a way to think about it is:

1. Use a function if you want to do one thing
2. Use a stored procedure if you want to do a few things over and over, but never individually
3. Use a view if you want to see a combination of tables without creating a new table (ie, creating redundant information).

Conclusion

Views are an easy and essential aspect of SQL that drastically expand your ability to communicate with more groups while also keeping your data organized and secure. That said, creating a View is a specific task in SQL while there are ways with Functions and Stored Procedures to mimic the same response as a View with additional function. Additional function is not always a good thing, so weighing the pros and cons of each in each scenario is a positive move forward. Similarly, those that you will be working with could be more comfortable in one method versus another, so be sure to ask the question before going too far forward in one direction.