Transact SQL Programming

Reference Guide/Summary

Introduction

This document is to record my progress through the Transact SQL guidebook from O’Reilly by Kevin Kline, Lee Gould & Andrew Zanevsky.

Each chapter will have two sections. The first, a list of bullet points summarising the subsections of the chapter. Not all subsections will be included, only the ones I think will be relevant for creating a good reference guide later. History sections for example are, while interesting to read, very verbose and lacking value in summarising here. The same is true for out of date information regarding Sybase and Microsoft. Some of these subsections will be titled the same as they are in the book, while some will not, and in some cases, information will be in different sections here than they are in the book.

The second section will be listing the practise and additional research I have done surrounding each subsection. This includes SQL scripts, research excerpts from the web, with links to source material. The SQL scripts listed will be included in a ZIP file with this doc so that each can be opened and tested. All scripts will be rerunnable and compatible with Azure SQL Server as that is the platform I am using to practise

Key

* Text marked in Yellow are references to either SQL scripts or other of my own supporting documents to help explain a concept.
* Text marked in green are information directly copy and pasted from the internet.

Chapter 1 summary

**Introducing SQL/T-SQL**

* IBM introduced SQL; Oracle brought to market first
* ANSI (American National Standards Institute) Standardizes behaviours and syntax for vendors to adhere to
* Sybase introduced T-SQL to extend functionality

**The relational Database Model**

* *Relational Database*: a system whose users view data as a collection of tables related to each other through common data values (keys)
* E.F. Codd first proposed Relational Data Theory in 1970
* Important concepts for relational data include *Normalisation* and *Row vs Set processing*

**Normalisation**

* It’s a database design technique used to minimise programming problems
* Unnormalized data is harder to delete from, insert into and update
* First normal from (1NF) eliminates repeating groups by splitting subgroups into their own tables
* This often causes concatenated keys to be required
* Second normal form (2NF) requires partial dependencies to be eliminated, which are data that does not depend on the primary key to be uniquely identified.
* Third normal form eliminates transitive dependencies. These are hidden dependencies where one column more uniquely identifies its information than the other. This is difficult to explain, hence see practise example.
* Denormalization is sometimes used to speed up queries where joins are often used to get the result. If a BAU process routinely joins two tables to get information, it might be better for those tables to be denormalized into one so that the query is faster, although it will take up more storage space as a trade-off.

**Row vs Set processing**

* Row processing performs logic on one record after the other and the programmer needs to tell the program exactly how to treat the data
* Set processing requires only that the programmer specify what they want and not how to get it, while performing logic on multiple rows of data at once
* Cursors are an extension to SQL that allows the normally set processed language to manipulate data at an individual row level

**Tracing and Debugging**

* Using transaction control functions like BEGIN TRAN, COMMIT TRAN and ROLLBACK TRAN, you can define exactly what the transaction is rather than it being implied.
* DBCC commands can be used to check database consistency, database and server configuration, debugging and performance tuning and monitoring
* SET can be used to enable or disable specific ANSI standards or more generally, force certain behaviours within a given session

**Special Aggregate Functions**

* WITH CUBE makes the result set include summary rows of all combinations of the keys in the GROUP BY clause
* WITH ROLLUP returns a subset of the summaries returned by CUBE, depending on the order of the keys in the GROUP BY clause

Chapter 1 Practise work/examples

**Normalization**

* Create\_circus\_table.sql showing unnormalized table
* circusanimals1nf.sql + tricks1nf.sql showing 1nf tables using SELECT INTO and UNPIVOT techniques
* Trickstable2nf.sql + Animaltrickstable2nf.sql showing 2nf tables
* tents3nf.sql + animals\_lodging3nf.sql showing 3nf tables

**Row vs set processing**

Row-based:

for each employee meeting criteria  
update table where emplid = x  
end-for

>>>> number of database calls = number of employees meeting criteria.  
If  
you have 100,000 employees, that’s 100,000 db calls.

Set-based:

update table where employee meets criteria

>>>>> number of database calls = 1 (regardless of query)

Example Credit: <https://www.toolbox.com/tech/question/difference-between-set-processing-and-row-processing-011807/#:~:text=12%20Answers&text=Set%20processing%20acts%20on%20a,all%20depends%20on%20the%20requirement>

By: Neil.Paku Posted January 18, 2007

**Special Aggregate Functions**

* CUBE.sql showing CUBE aggregate
* ROLLUP.sql showing ROLLUP aggregate

**MISC/Extras**

* Temp table example tmptables.sql
* Cursor example cursorquery.sql
* GOTO and WAITFOR examples goto&waitforstatements.sql
* IF statement example ifstatement.sql
* WHILE statement example whilestatement.sql