

The Big Ideas of SQL

by

Joe Celko

copyright 2006

Big Idea #1

- Separate data and code
- Data used to be tied to the programs which use it.
- If you separate data from any particular application, you can share it among many applications.
- But now the data has to take care of itself.

Host Language vs Database

- The first result of Big Idea #1
- SQL sits between the database and the host program which runs the application
 - SQL has no I/O of its own
 - SQL datatypes are not the same as the host program datatypes

Big Idea #2

- Logical versus Physical
- SQL has no idea how data is physically stored in the hardware
- All references are made by identifiers and logical descriptions of the data.
- No two SQL product are anything alike inside

Keys

- Result of Big Idea #2
- You cannot locate a row in a table by any physical references, so you have to use logic
- If you want to find one row in particular, then you need a way to identify it from the other rows

Big Idea #3

- Declarative versus Procedural
- Procedural languages like Cobol, Fortran, Pascal, C tell the machine *what to do*
- Declarative languages like SQL tell the machine *what you want*, and let the database engine figure out how to do it.
 - As long as the results are correct, the engine does not have to do the same query the same way each time

Big Idea #4

- Operators, Relationships and Closure
- Closure means that operations on tables produce tables as a result
 - You can nest expressions
 - You get nice mathematical properties
- Operators let you work tables as single units - no row-at-a-time processing
- Relationships restrict you from screwing up things.

Big Idea #5

- We do not always know everything
- We need some kind of token for missing data.
- There are several schemes for doing this, but SQL settled on the NULL
- Smaller idea: there is a difference between missing data values and an empty table

The Dreaded NULL

- Not a value, has no datatype!
- Missing, Unknown, N/A and Error
- NULLs propagate in operation
- Comparing NULLs is always UNKNOWN, not TRUE or FALSE

Logic and SQL

- Comparison on NULL is UNKNOWN
- Boolean operators for Three Values
- AND - a single FALSE dominates
- OR - a single TRUE dominates
- NOT - reverse only TRUE and FALSE
- IS [NOT] [TRUE | FALSE | UNKNOWN]
operator in SQL-92

Big Idea #6

- Normalization
 - The goal of a database was to remove redundancy from the old file systems
 - Normalization removes redundancy from the schema
- One fact, one place, one time, one way
- This lets the data take care of itself via referential actions

Basic Idea

- A database is normalized when all its tables are normalized
- A table is normalized when each fact appears
 - One place
 - One time
 - One way

Big Idea # 7

- A small, simple set of operators can build new tables from old ones (or “Algebra was a good idea”)
- The results can be quite complex
- The expressions can be optimized
- The operators can be driven by logic

Big Idea #8

- Forests and trees do not belong in the same table
- Sets are made of elements of the same type, so you must have elements at the same level of aggregation.
- This leads to the GROUP BY clause and aggregate functions

Big Idea #9

- Computers should work, people should think
- If this relational stuff is based on formal ideas, then you can use software tools to verify database designs.
- Designing files was dumb, blind luck and a lot of artistry

Big Idea #10

- Transactions: Start what you finish
- A user has a session with zero or more transactions in it.
- A transaction is a unit of work that either fails or completes as a whole
- This "All or Nothing" philosophy is an important part of SQL

Transactions

- Classic example: I want to move \$10.00 from Savings to Checking at an ATM
- If the ATM machine crashes during the transfer, I could either be ahead or behind the \$10.00
- Both debit and credit must occur or the deal is off

User Sessions

- CONNECT attaches user to database
- COMMIT makes work permanent, ends transaction
- ROLLBACK restores database, ends transaction
- SAVEPOINT <id > is a "mini-commit"
- DISCONNECT ends a session

CONNECT

- CONNECT attaches user to database
- Usually with a simple password scheme
- System finds all of the privileges for the user, so you cannot get to everything
- You are connects to the whole database and not just some of the tables

COMMIT

- COMMIT makes work permanent and ends a transaction
- Until the COMMIT is executed, the changes to the database are not permanent
- Some constraints can be deferred, but none can be dropped
- The whole database must be consistent with all constraints after the commit is done.

ROLLBACK

- ROLLBACK restores database to state it was in before the transaction began and then ends transaction
- Users can execute a ROLLBACK
- The system can reply to an error with a ROLLBACK

Big Idea #11

- Concurrency Control: We can drive on the same roads if everyone plays by the same rules
- A database has multiple users, so you have to regulate the traffic
- Priority systems for allowing users in the database at the same time
- Resolve conflicts
 - Optimistic: take care of it after the fact
 - Pessimistic take care of it before the fact

Concurrency in SQL

- Databases are used by more than one person at a time!
- Pessimistic Concurrency assumes there will be conflicts (uses locks)
- Optimistic Concurrency handles conflicts as exceptions (uses generation copies)
- Logical Concurrency avoids conflicts by not allowing conflicting work in the database at the same time

Three Phenomena

- Dirty Reads = you see other user's changes to the database as they happen
- Unrepeatable Reads = you get one result set the first read, but it changes the next time you use the query
- Repeatable Reads = you read the same data each time, but the rest of the database changes.

Four Isolation Levels

- SET TRANSACTION <keyword>
- Serializable = looks like you own the whole database
- Repeatable Read = (Cursor stability) The part of the database you are using stays the same for your session
- Read Committed = see other user's committed work
- Read Uncommitted = you see other user's work as it happens, even though it might not be committed later

Deadlocks

- Mr. A needs X and Y and has X.
- Mr. B needs X and Y and has Y.
- Neither A nor B will give up his treasure
- Possible outcomes
 - Mr. A and Mr. B starve to death
 - One of them has priority over the other and takes his resource
 - An outside force (the DBA) gives one of them priority over the other.

Livelocks

- Mr. A needs to get X, but every time he tries, someone else has beat him to it. The busy telephone line problem
- Possible outcomes
 - Mr. A starves to death
 - Mr. A gets lucky
 - Mr. A gets priority from the DBA
 - Mr. A gets higher priority as he waits until he can bump someone