Points:

- SQL stands for *Structured Query Language*.
- SQL statements are divided into two main branches: **DDL** and **DML**.
- SQL is declarative (non-procedural).
- SQL was created by Don Chamberlin and Ray Boyce of IBM.

---

**What's “SQL”?**

“SQL” stands for “*Structured Query Language*”. Over the years, almost all relational database management systems have moved to using SQL. The DBMS uses SQL for all communications with the world outside itself. Even if you use a GUI to build a table, the system translates the GUI representation to SQL statements to send to the database.

---

**Pronunciation**

Before we get started, SQL can be pronounced either S-Q-L or “Sequel”. I don’t believe there’s a consensus on which pronunciation is correct, or even which is more common. When ANSI first standardized the language in 1986, they officially called it S-Q-L. But the earlier IBM language from which SQL was derived was named “SEQUEL” for “Structured English Query Language”. For me, S-Q-L just doesn’t roll off the tongue the way Sequel does. I also say “Gooey” instead of G-U-I.

---

**DDL and DML**

There are two branches of SQL statements:

- **Data Definition Language (DDL)** The set of SQL statements we use to create and alter tables, their columns and constraints. (*Works on structure*)

- **Data Manipulation Language (DML)** The set of SQL statements we use to query and change the data in tables. (*Works on data*)

You might hear “I’d like to see the DDL used to build this table”, but not “Where’s the DML you wrote to read this table?”. You'd hear “Where’s the SQL you wrote to read this table?”. 


**A Declarative (Non-Procedural) Language**

The first thing to notice about SQL statements is they’re *declarative*, or *non-procedural*.

A story: Before SQL was widespread, I performed data access using an ordinary functional language. Reading data meant writing a loop, checking for the end of the data, and testing for records to print. I had to declare and increment variables for counters and accumulators. SQL frees you from all of that coding.

When I started using SQL, I rebelled and wished I could just work my way through the records the old-fashioned way, using code I could control. After getting the hang of SQL, I’d *never* go back to using code. SQL it is *so much easier!* You can pop out a query to compare tables, find missing records, get a quick sum on an account within a date range, etc. You’d never have the time to do this using old procedural code!

**Standard Versions**

There have been eight versions of SQL adopted by ANSI (the American National Standards Institute). These are accompanied by corresponding ISO standards. The ANSI standard numbers correspond to the year of adoption:

1. SQL-86 (based on SEQUEL, standardized the languages in the marketplace)
2. SQL-89 (minor revision to the original standard)
3. SQL-92 (also known as SQL2, this major revision is very widely used)
4. SQL:1999 (also known as SQL3, supports OO features and recursive queries)
5. SQL:2003 (supports XML features)
6. SQL:2006 (more complete XML support)
7. SQL:2008 (adds INSTEAD OF triggers, expands ORDER BY and TRUNCATE)
8. SQL:2011 (improved support for temporal data)

**Some History**

The SQL that ANSI first standardized in 1986 was based on the SEQUEL language developed in IBM’s System R project. Remember that E. F. Codd worked at IBM developing his relational ideas while the rest of the information world was pursuing navigational (pointer-connected) databases. This is approximately how things happened:

The most advanced type of database in the 1960’s was Bachman’s Network Database. In 1969 a language for querying the Network Database was proposed by the CODASYL DBTG (Conference on Data Systems Languages - Database Task Group), an industry consortium for developing computer languages. This language
became known as the CODASYL Data Model. One of IBM’s experts on this language was an engineer named Don Chamberlin. Here’s how Chamberlin described the CODASYL Data Model:

“My job was to study this CODASYL DBTG proposal and learn about it and give presentations on it and figure out what needed to be done to it and things like that. So I became an expert on DBTG and I just loved it and thought it was neat. It had all sorts of real complicated pointers and set-oriented selection rules and you could just study it all day. It was a real puzzle.”

After Codd’s Relational Model started gaining attention, IBM started an internal research project named “System R” with the aim of developing a working Relational Database.

Chamberlin, together with Ray Boyce (whose name you’ll hear again when we study the Boyce-Codd Normal Form), joined this project around 1972. The two of them had the task to create a language for the new Relational Database. To help them, they developed a way of thinking they called “the Query Game” to help them derive simple ways to express complicated queries using Relational principles.

The result was SEQUEL (Structured English Query Language), which evolved to become the SQL that we know today.

In a 1995 reunion of System R project members, Chamberlin told a story relating his first encounter with the new Relational Model. (Boyce was absent from the reunion, having passed away as a young man in 1974 after working for IBM for only two years).

“We knew sort of peripherally that there was some work going on in the provinces, in San Jose. There was this guy Ted Codd who had some kind of strange mathematical notation, but nobody took it very seriously. Ray Boyce was hired at about this time, and we kind of got into this game called the Query Game where we were thinking of ways to express complicated queries.

But actually before the Query Game started, I had a conversion experience, and I still remember this. Ted Codd came to visit Yorktown… He gave a seminar and a lot of us went to listen to him. This was as I say a revelation for me because Codd had a bunch of queries that were fairly complicated queries and since I'd been studying CODASYL, I could imagine how those queries would have been represented in CODASYL by programs that were five pages long that would navigate through this labyrinth of pointers and stuff. Codd would sort of write them down as one-liners. These would be queries like, "Find the employees who earn more than their managers." [laughter]

---

1 Photo from http://www.mcjones.org/System_R/people.html
2 Photo from http://www.almaden.ibm.com/cs/people/chamberlin/chamberlin-publist.html
He just whacked them out and you could sort of read them, and they weren't complicated at all, and I said, "Wow." This was kind of a conversion experience for me, that I understood what the relational thing was about after that."  

Codd, the mathematician, used Relational Calculus to express his queries. Chamberlin and Boyce used this as a foundation to build plain English expressions that were, in effect, Relational Calculus word-problems for the same queries.

Chamberlin continues about how he and Boyce went from relational mathematics to an English-like language they called SQUARE, which was the first step to SQL:

We called the notation SQUARE; it stands for Specifying Queries as Relational Expressions. We had this idea, that Codd had developed two languages, called the relational algebra and the relational calculus. In the relational algebra, the basic objects were tables, and you combined these tables with operations like joins and projections and things like that. The relational calculus was a kind of a strange mathematical notation with a lot of quantifiers in it.

We thought that what we needed was a language that was different from either one of those, in which the basic objects that you worked on were sets of values, and the things you did to those sets of values were you mapped one set of values into another using some kind of a table.

So we had the usual database of sales and departments and items being located on different floors and we would take a value like two and map it through this notation into the departments that were on that floor, and then we'd map it again into the items that were sold by those departments. We would try to show that this mapping notation was simpler than some of the complex ways that you'd have to express this query in relational calculus, or of course far worse, using something like CODASYL.  

As Chamberlin and Boyce developed their notation, they changed the name from SQUARE to SEQUEL (Structured English QUEry Language). This has come down to the present day as SQL.