History

**Creation**
- language for relational databases
- IBM
  - Early 1970’s
  - SEQUEL → SQL

**Modern day**
- many implementations
  - SQL Server
  - SQLite
  - MySQL
  - PostgreSQL
  - Oracle SQL
MySQL

- FOSS
- popular
- scalable
  - Facebook
  - Twitter
- procedural support
Syntax

comments
- -- line comment
- ( { block comment } )
- /* C-style block comment */

symbols
- SET @bob = 6; -- Sets bob to 6
- DECLARE var1 INT; SET var1 = 0;

Expressions
SET @a = 1;
SET @b = 2;
SET @c = 3;
SET @d = 4;
IF d < a OR b < c OR b < d THEN @q = 8;
Common Programming Statements

Core functions of persistent storage
- **C**: INSERT
- **R**: SELECT
- **U**: UPDATE
- **D**: DELETE

MySQL Procedural statements
- CREATE PROCEDURE
- CREATE FUNCTION
- IF...THEN...ELSE
- label:LOOP...END LOOP
Executing SQL statements

Interpreted or compiled?
- Depends on implementation
  - SQLite (prepare() functions)
- SQL optimization
  - SQL String → (Optimizer) → Execution Plan
  - Execution Plan → (Execution) → Result

Execute procedures/functions
- CALL testproc(@a) -- stored procedure
- SELECT name(in) -- stored function
Memory management

- Server managed
- Specify memory tables with memory engine

```
CREATE TABLE students ENGINE=MEMORY;
```
Interesting and Identifying Features of SQL

- non-procedural
- implicitly parallel
- declarative
- case-insensitive
● Malicious exploitation of poorly written applications
Given frontend to school's database, input string name:

```
SELECT * FROM Students WHERE (student_name=name);
```

name = "Robert’); DROP TABLE Students; --"

Resulting query:

```
SELECT * FROM Students WHERE (student_name='Robert'); DROP TABLE Students;-- ')
```
How to prevent it?

- End goal: Sanitize inputs
- (Don't do it yourself!)
- Solution: **parameterized statements**
- See bobby-tables.com for language-specific examples.

**Bad** - vulnerable to injection

```python
cmd = ("SELECT * FROM Students WHERE" 
      "(user_name = '%s')" % student_name) 
curs.execute(cmd)
```

**Good** - uses parameterized statements

```python
cmd = "SELECT * FROM Students WHERE (student_name='%s')" 
curs.execute(cmd, student_name)
```