SQL1: The Basics

CS411 Database Systems

Kevin C. Chang

SQL Introduction

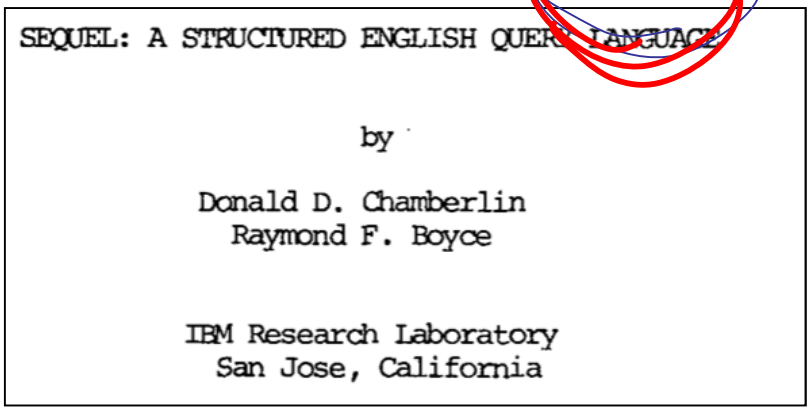
Standard language for querying and manipulating data

Structured **Q**uery **L**anguage

Many standards out there: SQL92, SQL2, SQL3, SQL99
Vendors support various subsets of these, but all of what we'll be talking about.

Behind the Scene: The Birth of SQL

- Chamberlin, D. D. and Boyce, R. F. 1974. SEQUEL: A structured English query language. In Proceedings of the 1974 ACM SIGFIDET (Now Sigmod) Workshop on Data Description, Access and Control (Ann Arbor, Michigan, May 01 - 03, 1974). FIDET '74. ACM, New York, NY, 249-264.



SEQUEL: A STRUCTURED ENGLISH QUERY LANGUAGE

by

Donald D. Chamberlin
Raymond F. Boyce

IBM Research Laboratory
San Jose, California

SQL

ABSTRACT: ... the data manipulation facility for a ... which can be used for accessing ... Without resorting to the concepts ... identifies a set of simple operations on tabular structures, which can be shown to be of equivalent power to the first order predicate calculus. A SEQUEL user is presented with a consistent set of ... which reflect how people use tables to obtain information. Moreover, the SEQUEL user is able to compose these basic templates in a structured manner in order to form more complex queries. SEQUEL is intended as a data base sublanguage for both ...

Why SQL?

- SQL is a very-high-level language, in which the programmer is able to avoid specifying a lot of data-manipulation details that would be necessary in languages like C++.
- What makes SQL viable is that its queries are “optimized” quite well, yielding efficient query executions.

Select-From-Where Statements

- The principal form of a query is:

SELECT desired attributes
FROM one or more tables
WHERE condition about tuples of the tables

Running Example

Beers(name, manf)
Bars(name, addr, license)
Drinkers(name, addr, phone)
Likes(drinker, beer)
Sells(bar, beer, price)
Frequents(drinker, bar)

Example

- Using Beers(name, manf), what beers are made by Anheuser-Busch?

```
SELECT name
FROM Beers
WHERE manf = 'Busch';
```

Meaning of Single-Relation Query

- Begin with the relation in the FROM clause.
- Apply the selection indicated by the WHERE clause.
- Apply the extended projection indicated by the SELECT clause.

“Everything” In SELECT Clause: STAR

- When there is one relation in the FROM clause, * in the SELECT clause stands for “all attributes of this relation.”
- Example using Beers(name, manf):

```
SELECT *
FROM Beers
WHERE manf = 'busch';
```

Renaming Attributes

- If you want the result to have different attribute names, use "AS <new name>" to rename an attribute.
- Example based on Beers(name, manf):

```
SELECT name AS beer, manf
FROM Beers
WHERE manf = 'Busch'
```

Expressions in SELECT Clauses

- Any expression that makes sense can appear as an element of a SELECT clause.
- Example: from Sells(bar, beer, price):

```
SELECT bar, beer, price * 120 AS priceInYen
FROM Sells;
```

Complex Conditions in WHERE Clause

- Composing conditions:
  - Use AND, OR, NOT, and parentheses.
- From Sells(bar, beer, price), find the price Joe's Bar charges for Bud:

```
SELECT bar, beer, price
FROM Sells
WHERE bar = 'joe bar' AND price < 5.0;

SELECT bar, beer, price
FROM Sells
WHERE (bar = 'joe bar' AND price < 5.0) or (beer = 'sam adam');
```

Selection Conditions

- Each condition
  - Attribute names of the relation(s) used in the FROM.
  - Comparison operators: =, <>, <, >, <=, >=
  - Apply arithmetic operations: stockprice*2
  - Operations on strings (e.g., "||" for concatenation).
  - Pattern matching: s LIKE p
    Pattern: % = "any string"; _ = "any character."

    ```
    SELECT name, phone FROM Drinkers
    WHERE phone LIKE '%333-_ _ _ _';
    ```

  - Lexicographic order on strings.

    ```
    SELECT name, city FROM Drinkers
    WHERE name < 'c';
    ```

  - Special stuff for comparing dates and times.

  - SQL is *case-insensitive*. In general, upper and lower case characters are the same, except inside quoted strings.

What Should This Query Return?

```
SELECT bar, beer
FROM Sells
WHERE price < 5.00  OR  price >= 5.00;
```

vs.

```
SELECT bar, beer FROM Sells;
```

NULL Values

- Tuples in SQL relations can have NULL as a value for one or more components.
- Meaning depends on context. Two common cases:
  - *Missing value* : e.g., we know Joe's Bar has some address, but we don't know
  - *Inapplicable* e.g., the value of attribute *spouse* for an unmarried person.

Comparing NULL's to Values

- The logic of conditions in SQL is really 3-valued logic: TRUE, FALSE, UNKNOWN
- When any value is compared with NULL, the truth value is UNKNOWN.
- But a query only produces a tuple in the answer if its truth value for the WHERE clause is TRUE (not FALSE or UNKNOWN).

Three-Valued Logic

To understand how AND, OR, and NOT work in 3-valued logic:

- TRUE = 1.
- FALSE = 0.
- UNKNOWN = ½.

Operators:

- AND = MIN.
- OR = MAX.
- NOT(*x*) = 1-*x*.

- Ex: TRUE AND (FALSE OR NOT(UNKNOWN))
  = MIN(1, MAX(0, (1 - ½)))
  = MIN(1, MAX(0, ½)
  = MIN(1, ½) = ½.

Reason: 2-Valued Laws != 3-Valued Laws

- Some common laws, like the commutatively of AND, hold in 3-valued logic.
- But others do not:
  - Ex: ," *p* OR NOT *p* = TRUE ("law of excluded middle")
  - When *p* = UNKNOWN, the left side is MAX(½, (1 – ½)) = ½ which is not 1 or TRUE.

Testing for Null

Can test for NULL explicitly:

- x IS NULL
- x IS NOT NULL

```
SELECT bar, beer
FROM Sells
WHERE price < 5.00  OR  price >= 5.00 OR price IS NULL
```

Multi-relation Queries

- Interesting queries often combine data from more than one relation.
- We can address several relations in one query by listing them all in the FROM clause.
- Distinguish attributes of the same name by “<relation>.<attribute>”

Example

- Find the beers liked by at least one person who frequents Joe's Bar.
- Tip: Always prefix with relation name to make it clear/easier to read.

```
SELECT Likes.beer
FROM Likes, Frequents
WHERE Frequents.bar = 'joe bar' AND
      Frequents.drinker = Likes.drinker;
```