SQL Queries

The SELECT-FROM-WHERE Structure

```
SELECT <attributes>
FROM <tables>
WHERE <conditions>
```

From relational algebra:

- `SELECT <attributes>` corresponds to projection
- `FROM <tables>` specifies the table in parentheses in a relational algebra expression and joins
- `WHERE <conditions>` corresponds to selection

Projection

$\pi_{first\_name,last\_name}(author)$

```
mysql> select first_name, last_name from author;
+------------+-----------+
| first_name | last_name |
+------------+-----------+
| John       | McCarthy  |
| Dennis     | Ritchie   |
| Ken        | Thompson  |
| Claude     | Shannon   |
| Alan       | Turing    |
| Alonzo     | Church    |
| Perry      | White     |
| Moshe      | Vardi     |
| Roy        | Batty     |
+------------+-----------+
9 rows in set (0.00 sec)
```

Asterisk

```
mysql> select * from author;
+-----------+------------+-----------+
| author_id | first_name | last_name |
+-----------+------------+-----------+
|         1 | John       | McCarthy  |
|         2 | Dennis     | Ritchie   |
|         3 | Ken        | Thompson  |
|         4 | Claude     | Shannon   |
|         5 | Alan       | Turing    |
|         6 | Alonzo     | Church    |
|         7 | Perry      | White     |
|         8 | Moshe      | Vardi     |
|         9 | Roy        | Batty     |
+-----------+------------+-----------+
9 rows in set (0.00 sec)
```

Notice that with no condition on select, all rows returned.

Select

$\sigma_{year=2012}(book)$

```
mysql> select * from book where year = 2012;
+---------+------------+-------+------+--------+
| book_id | book_title | month | year | editor |
+---------+------------+-------+------+--------+
|      7 | AAAI      | July | 2012 |    9 |
|      8 | NIPS      | July | 2012 |    9 |
+---------+------------+-------+------+--------+
2 rows in set (0.00 sec)
```

The FROM Clause

The FROM clause takes one or more source tables from the database and combines them into one (large) table using the JOIN operator. Three kinds of joins:

- CROSS JOIN
- INNER JOIN
- OUTER JOIN

Sicne DB designs are typically factored into many tables, the join is the most important part of a query.

CROSS JOIN

A CROSS JOIN matches every row of the first table with every row of the second table. Think of a cross join as a cartesian product.

The general syntax for a cross join is:

```
SELECT <select_header> FROM <table1> CROSS JOIN
   <table2>
```

or

```
SELECT <select_header> FROM <table1>, <table2>
```

CROSS JOIN EXAMPLE

```
mysql> select * from pub cross join book;
+--------+-----------------+---------+---------+-----
| Pub_id | title           | book_id | book_id |
   book_title | month | year | editor |
+--------+-----------------+---------+---------+-----
|      1 | LISP            |       1 |       1 | CACM
        | April    | 1960 |      8 |
|      2 | Unix            |       2 |       1 | CACM
        | April    | 1960 |      8 |
|      3 | Info Theory     |       3 |       1 | CACM
        | April    | 1960 |      8 |
|      4 | Turing Machines |       4 |       1 | CACM
        | April    | 1960 |      8 |
|      5 | Turing Test     |       5 |       1 | CACM
        | April    | 1960 |      8 |
|      6 | Lambda Calculus |       6 |       1 | CACM
        | April    | 1960 |      8 |
```

~LIMIT~ing Results

If we don't want many results to scroll past the bottom of the screen we can limit the number of results using a LIMIT clause.

```
mysql> select * from pub, book limit 3;
+--------+-------------+---------+---------+-----
| pub_id | title       | book_id | book_id |
  book_title | month | year | editor |
+--------+-------------+---------+---------+-----
|      1 | LISP        |       1 |       1 | CACM
       | April | 1960 |     8 |
|      2 | Unix        |       2 |       1 | CACM
       | April | 1960 |     8 |
|      3 | Info Theory |       3 |       1 | CACM
       | April | 1960 |     8 |
+--------+-------------+---------+---------+-----
3 rows in set (0.00 sec)
```

The general form of the LIMIT clause is LIMIT start, count,

Inner Joins

A simple inner join uses an ON condition.

```
mysql> select * from pub join book on
   pub.book_id = book.book_id;
+--------+-----------------+---------+---------+------
| pub_id | title           | book_id | book_id |
   book_title | month | year | editor |
+--------+-----------------+---------+---------+------
|      1 | LISP            |       1 |       1 | CACM
        | April    | 1960 |      8 |
|      2 | Unix            |       2 |       2 | CACM
        | July     | 1974 |      8 |
|      3 | Info Theory     |       3 |       3 | BST
         | July     | 1948 |      2 |
|      4 | Turing Machines |       4 |       4 | LMS
         | November | 1936 |      7 |
|      5 | Turing Test     |       5 |       5 | Mind
        | October  | 1950 |   NULL |
```

Natural Joins

The USING clause, also called a natural join, equijoins on a like-named column from each table and includes the join column only once.

```
mysql> select * from pub join book using
   (book_id);
+---------+--------+----------------+------------+---
| book_id | pub_id | title          | book_title |
  month   | year | editor |
+---------+--------+----------------+------------+---
|       1 |      1 | LISP           | CACM       |
  April     | 1960 |      8 |
|       2 |      2 | Unix           | CACM       |
  July      | 1974 |      8 |
|       3 |      3 | Info Theory    | BST        |
  July      | 1948 |      2 |
|       4 |      4 | Turing Machines  | LMS      |
  November  | 1936 |      7 |
```

Many to Many Relationships

A single author can write many publications, and a single publication can have many authors. This is a many-to-many relationship, which is modeled in relational databases with a relationship (or link or bridge) table.

```
CREATE TABLE IF NOT EXISTS author_pub (
  author_id INTEGER NOT NULL REFERENCES
     author(author_id),
  pub_id INTEGER NOT NULL REFERENCES
     publication(pub_id),
  author_position INTEGER NOT NULL, -- first
     author, second, etc?
  PRIMARY KEY (author_id, pub_id)
);
```

author_pub tables links the author and pub tables

- author_id and pub_id are foreign keys to author and pub tables

Joining Multiple Tables

We can join all three tables by chaining join clauses:

```
mysql> select *
    -> from author join author_pub using
       (author_id)
    ->  join pub using (pub_id);
+--------+-----------+------------+-----------+-------
| pub_id | author_id | first_name | last_name |
  author_position | title     | book_id |
+--------+-----------+------------+-----------+-------
|      1 |         1 | John       | McCarthy  |
               1 | LISP              |       1 |
|      2 |         2 | Dennis     | Ritchie   |
               1 | Unix              |       2 |
|      2 |         3 | Ken        | Thompson  |
               2 | Unix              |       2 |
|      3 |         4 | Claude     | Shannon   |
               1 | Info Theory       |       3 |
```

String Matching with LIKE

Our where condition can match a pattern with like. Use a % for wildcard, i.e., matching any character sequence.

Which publications have "Turing" in their titles?

```
elect * from pub where title like 'Turing%';
+--------+-----------------+---------+
| pub_id | title           | book_id |
+--------+-----------------+---------+
|      4 | Turing Machines |       4 |
|      5 | Turing Test     |       5 |
+--------+-----------------+---------+
2 rows in set (0.00 sec)
```

Note that strings are not case-sensitive.

Simple Database: Dorms

1. Download [dorms.sql](../resources/dorms.sql)
2. On the command line, go to the directory where you downloaded dorms.sql
3. Make sure your MySQL server is running:

   ```
   $ mysql.server start
   Starting MySQL
   SUCCESS!
   ```

4. Run the `dorms.sql` script like this:

   ```
   $ mysql -u root -p < dorms.sql
   Enter password:
   ```

Running Queries on the Dorms Database

Start MySQL's client and use the `dorms` database.

```
$ mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with
   ; or \g.
...
mysql> use dorms
...
Database changed
mysql>
```

Exploring the Database

Get a list of the tables:

```
mysql> show tables;
+-----------------+
| Tables_in_dorms |
+-----------------+
| dorm            |
| student         |
+-----------------+
2 rows in set (0.00 sec)
```

See the structure of a table:

```
mysql> describe dorm;
+---------+---------+------+-----+---------+-------+
| Field   | Type    | Null | Key | Default | Extra
   |
+---------+---------+------+-----+---------+-------+
| dorm_id | int(11) | NO   | PRI | NULL    |       |
```

Simple Queries on Dorms Database

- What are the names of all the dorms?
- Which students have GPAs greater than 3.0?
- Which students are in Armstrong?
- Rank students by GPA.
- Which student has the top GPA?