



THE UNIVERSITY
of ADELAIDE

Web and Database Computing •

adelaide.edu.au

Intro to Relational Databases: Structured Query Language

Database Design

Step 1: Specify requirements

~~**Step 2:** Model the data~~ (optional - we'll skip this)

Step 3: Develop a relation schema

Step 4: Implement relation schema **(TODO)**

We will cover these steps in further detail after the break,
but for now we'll look at how we can work with a database.

Database Implementation

Once we have our schema, we need a way of

- Loading it into a RDBMS
- Adding data
- and interacting with that data

The most common way of doing this in relational databases is using SQL

Structured Query Language (SQL)

- Most popular commercial relational query language
- Originally developed by IBM in the 70s
- Versions
 - ISO standards 1986, 1989, 1992, 1999
 - Although there are standards, there are many vendor specific dialects
 - Slight variations in syntax and functions specific to a given implementation.
 - We are using SQL supplied by MySQL.
 - A popular open source relational database

SQL Commands

SQL provide commands for us to do operations on our databases:

- Create databases
- Create and modify tables (entity sets)
- Add data (rows)
- Update data
- Search for data
- Remove data, tables, and databases

Getting Started

Start our database (and install/setup if not already done)

```
sql_start
```

Connect to your database using the mysql command:

```
mysql --host=127.0.0.1
```

Create a Database

Create a database with the CREATE DATABASE statement:

```
CREATE DATABASE dbname;
```

- Our database server can have multiple databases, so we may need to create one if we don't already have one set up.
- Skip the above step if you already have one.

Use Database with the USE statement:

```
USE dbname;
```

- Sets the database that we'll be working with.

Create tables

Create a table with the CREATE TABLE statement:

```
CREATE TABLE tablename (  
  columnname1 datatype constraint,  
  columnname2 datatype constraint,  
  columnname3 datatype constraint,  
  ....  
);
```

- A table in database is an entity set
- A column header is an attribute
- Datatype defines what type of data will be stored in each column.
 - INT, FLOAT, CHAR, DATE, etc
- Constraints allow us to specify additional properties for a column.
 - Primary Key, Foreign Key, Weak Entity, Checks on data entry etc.
 - We will look at how to use these constraints later on

Removing Databases & Tables

Use the DROP DATABASE or DROP TABLE statement:

```
DROP DATABASE databasename;
```

```
DROP TABLE table_name;
```

Modifying tables

Modify a table with the ALTER TABLE statement:

```
/* Add a column */  
ALTER TABLE tablename  
    ADD columnname1 datatype constraint;  
  
/* Change a column */  
ALTER TABLE tablename  
    ALTER columnname1 newdatatype newconstraint;  
  
/* Delete a column */  
ALTER TABLE tablename  
    DROP columnname1;
```

- If major change is required, the table needs to be dropped and re-created

Adding data

Add a row of data to a table with the INSERT INTO statement:

```
INSERT INTO tablename (columnname1, columnname2, columnname3)  
  VALUES (value1, value2, value3);
```

Column names can be skipped if the row is inserted with all columns in order:

```
/* Okay to leave out columns */  
INSERT INTO tablename  
  VALUES (value1, value2, value3);  
  
/* Bad: wrong number of columns and out of order */  
INSERT INTO tablename  
  VALUES (value3, value1);
```

Updating & Deleting data

Use the UPDATE statement + SET clause to modify the data in the specified table

```
UPDATE table_name  
    SET columnname = value;
```

Use the DELETE statement to delete the data in the specified table

```
DELETE FROM table_name;
```

- Both of these affect all rows in the given table unless we filter the rows...

Filtering data with WHERE

The WHERE clause is used to filter records

- It can be used with update and querying statements
- Only records that fulfil a specified condition are affected by the statement

Use the UPDATE + WHERE to only update some rows

```
UPDATE table_name  
  SET columnname = value  
  WHERE columnname > 5;
```

Use the DELETE + WHERE to only delete some rows

```
DELETE FROM table_name  
  WHERE columnname IS NULL;
```

Filtering data with WHERE

- Comparison operators
`=, <, >, <=, >=, <>, between`
- Logical connectives
`not, and, or`
- Arithmetic expressions
`/, *, +, -`
- Operator precedence as shown above
 - Can be overridden with parentheses `()`
- NULL value test
`IS NULL, IS NOT NULL`

Example combining:

```
DELETE FROM table_name
WHERE column1 > 1200 AND (column2 = 'hello' OR column3 IS NOT NULL);
```

Filtering data with WHERE

Pattern matching in strings using LIKE:

- Match any substring (including empty string) %
 - Use \ to escape
- Match any single character _

Example:

```
UPDATE table_name  
SET column1 = value1  
WHERE lower(column2) LIKE '%foo_ar%';
```

Finding data

Use the SELECT ... FROM statement to query the database

```
SELECT column1, column2 FROM table_name;
```

- Retrieves all rows for the specified columns from the table
- Can use ***** operator to get all columns

Use the WHERE clause filter to specific rows

```
SELCT * FROM table_name WHERE column1 IS NULL;
```

- Both of these affect all rows in the given table unless we filter the rows...

Finding Data Across Tables

Use a JOIN to combine tables.

- A join combines tables on a common column, matching the values.
- The most common type of join is an **inner join** which only returns rows that match.
- We will look at other joins later in the course.

```
SELECT * FROM tableA INNER JOIN tableB  
ON tableA.column1 = tableA.column2  
WHERE column3 = 'Hello';
```

- The ON clause is used to specify which columns should be matched



THE UNIVERSITY *of* ADELAIDE

CRICOS PROVIDER NUMBER 00123M