Databases

CS 210: Data Management for Data Science

Databases

A database (DB) is some collection of organized information.

- spreadsheet
- document
- related information

DBMS

A database management system (DBMS) is software that manages a database.

- MySQL
- MongoDB

Cartesian products

Let A, B be sets.

The Cartesian product $A \times B$ is

$$\{(a, b) : a \in A, b \in B\}$$

A relation R is a subset of $A \times B$.

Relational (SQL) databases

- MySQL
- PostgreSQL
- MSSQL
- MariaDB

Non-relational (NoSQL) databases

- Key-value stores
- Documents
- Graphs

ACID

Some common properties we may want our DBMS to have:

- Atomicity: transactions are an atomic unit
- Consistency: invariants are preserved
- Isolation: running concurrently = serial execution
- Durability: completed transactions persist

SQL

Structured query language (SQL) is a language for manipulating databases.

It’s made up of multiple sublanguages:

- data definition language (DDL): modify DB objects
- data manipulation language (DML): modify information in DB
- data control language (DCL): permissions
- transaction control language (TCL): commit transactions

Programming paradigms

- Imperative
- Object-oriented
- Declarative
- Functional

CRUD

There are four common operations on databases:

- Create
- Read
- Update
- Delete

Setting up mysql

To log in the first time, you may have to use `sudo`:

```
sudo mysql -u root
```

or set the root password first:

```
mysqladmin -u root password 'myPassword'
```

Setting up mysql

To log in as root:

```
mysql -u root -p
```

Set password:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'myPassword';
```

User management

Add a user:

```
CREATE USER 'bob'@'localhost' IDENTIFIED BY 'myPassword';
```

Delete a user:

```
DROP USER 'bob'@'localhost';
```

User management

List users:

```sql
SELECT user, host FROM mysql.user;
```

List databases

To see all databases:

```
SHOW DATABASES;
```

To select a database:

```
USE myDatabase;
```

Create a database

Create a new database:

```sql
CREATE DATABASE myDatabase;
```

Delete a database:

```sql
DROP DATABASE myDatabase;
```

Permissions

To give permissions:

```
GRANT ALL ON myDatabase.* TO 'bob'@'localhost';
```

To query permissions:

```
SHOW GRANTS FOR 'bob'@'localhost';
```

Log in as user

To log in as your user:

```
mysql -u bob -p [database]
```

(or log in as before, then `USE myDatabase;`)

Tables

A database is made up of a number of tables.

Name	Major	GPA
Alan	CS	3.2
Emmy	Math	3.7
Daler	Dance	4.0

Each column has values of a single data type.

MySQL data types

Some common data types:

CHAR(n)	String w/ fixed length n
VARCHAR(n)	String w/ variable length, max length of n
TINYINT	8-bit integer
INT	32-bit integer
FLOAT	32-bit floating-point number
DOUBLE	64-bit floating-point number
DECIMAL	Fixed-point decimal number
DATE	Date in YYYY-MM-DD format
TIME	Time in HH:MM:SS format
YEAR	Year in YYYY format

Tables

To create a table:

```
CREATE TABLE students (name VARCHAR(20), major CHAR(5));
```

To delete a table:

```
DROP TABLE students;
```

Describe a table:

```
SHOW COLUMNS FROM myTable;
DESC myTable; -- shorthand for SHOW COLUMNS
```

Tables

To modify a table:

```sql
ALTER TABLE students ADD COLUMN gpa FLOAT;
```

List all tables:

```sql
SHOW TABLES;
```

CRUD

- Create: INSERT
- Read: SELECT
- Update: UPDATE
- Delete: DELETE

Insert

To insert a row:

```
INSERT INTO students VALUES ('Albert', 'Phys', 3.5);
```

We can insert partial rows:

```
INSERT INTO students (name, gpa) VALUES ('Marie', 2.9);
```

Select

To list all records:

```
SELECT * FROM students;
```

We can be more particular:

```
SELECT * FROM students WHERE gpa > 3.8;
```

We can ask for only certain columns:

```
SELECT name, gpa FROM students;
```

Update

To change a value:

```sql
UPDATE students SET gpa = 3.9 WHERE name = 'Emmy';
```

Delete

To delete a record:

```
DELETE FROM students WHERE name = 'Emmy';
```

Or we can delete multiple records:

```
DELETE FROM students WHERE gpa < 2.0;
```

Count and distinct

We can count rows:

```
SELECT COUNT(*) FROM students;
```

We can show only unique values:

```
SELECT DISTINCT name FROM students;
```

And combine these to count unique values:

```
SELECT COUNT(DISTINCT name) FROM students;
```

Limit

To avoid seeing too many results:

```
SELECT * FROM students LIMIT 5;
```

Where expressions

The where clause can be more complex:

```
SELECT * FROM students WHERE major <> 'CS';

SELECT * FROM students WHERE major = 'CS' AND gpa > 3.0;

SELECT * FROM students WHERE major = 'CS' OR major = 'Math';

SELECT * FROM students WHERE major = 'CS'
                         AND gpa BETWEEN 2.0 AND 3.0;

SELECT * FROM students WHERE NOT major = 'CS'
                         AND NOT major = 'Math';
```