Programming and Database Fundamentals for Data Scientists

Database Fundamentals

Varun Chandola

School of Engineering and Applied Sciences
State University of New York at Buffalo
Buffalo, NY, USA
chandola@buffalo.edu

University at Buffalo
School of Engineering
and Applied Sciences

Outline

Databases

Why?

- Why not store everything in files?
- Use Python to manipulate files

What?

- What is a Database, a Database Management System, Data Model?

How?

- How to load data into a database?
- How to *interact* with the data?

Overview

- Design of databases
  - Entity Relationship Model
  - Chapters 2, 4 (until section 4.5)
- Database programming
  - SQL
  - Chapters 6,7, and 8
- SQL in a server environment
  - Embedding SQL in Python
  - Chapter 9 (partly)

Book

Database Systems, The Complete Book (2nd Ed.), Hector Garcia-Molina, Jeffrey D. Ullman, Jennifer Widom (2009), Prentice Hall.

What is a Data Model?

- Mathematical representation of data
  - Examples: Relational, Semi-structured, Hierarchical, Network
- Operations on data
- Constraints

Relational Model - A Relation is a Table

- Data arranged as rows in a table, each row has related information about one data entity
- Consider the following relation (or table) – *Movies*

title	*year*	*length*	*genre*
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

- Attributes (column headers)
- Tuples (rows)
- Relation name (*movies*)

Schemas

- Relation schema – Relation name and attribute list
  - Type of each attribute
  - E.g., title - *String*, year - *Integer*, etc.
- A **Database** is a collection of relations (tables).
- The collection of all relation schemas in the database is the **database schema**

Why Relational Model

- Most popular - simple and limited
- Allows for highly efficient implementations to operate on the data
    - Allows implement high-level languages, such as SQL

Relational Model Basics

- Relation
- Attributes
- Tuples
- Schemas
- Domains
- Relation instance
- Relation keys

title	*year*	*length*	*genre*
Gone with the wind	1939	231	drama
Star wars	1977	124	sciFi
Wayne's world	1992	95	comedy

A Simple Database – Movies

Movies

```
title: string,
year: int,
length: int,
genre: string,
studioName: string,
producerCertificateNum:
int
```

Studio

```
name:  string,
address: string
presidentCertificateNum:
int
```

StarsIn

```
movieTitle:
string,
movieYear: int,
starName: string
```

MovieStar

```
name: string,
birthdate: date,
address: string,
gender: string
```

MovieExecutive

```
name: string,
certificateNum:
int,
address: string,
netWorth: int
```

Starting with SQL

- **Structure Query Language** or SQL is the language to interact with a relational database management system
- Has two uses
  1. Data definition – creating database schemas, etc.
  2. Data manipulation – querying, modifying database tables

Creating a Database

```
CREATE DATABASE moviedb;
```

Creating/Deleting Tables

```
-- create movies table
CREATE TABLE movies (
   title  VARCHAR(128) NOT NULL,
  year  INT,
  length  INT,
  studioname VARCHAR(128),
  executivenumber INT
);
-- deleting a table
DROP TABLE movies;
```

SQL Types - Numeric

Type	Storage	Min Signed/Unsigned	Max Signed/Unsigned
TINYINT	1	-128 0	127 255
SMALLINT	2	-2^{15} 0	$2^{15}-1$ 2^16-1
MEDIUMINT	3	-2^{23} 0	$2^{23}-1$ $2^{24}-1$
INT	4	-2^{31} 0	$2^{31}-1$ $2^{32}-1$
BIGINT	8	-2^{63} 0	$2^{63}-1$ $2^{64}-1$

SQL Types - Optional Width Argument

- One can optionally set the display width - `INT(4)`

SQL Types - Floating Points

- `FLOAT` and `DOUBLE` keywords to specify fields with single and double precision values, respectively.

SQL Types - Date and Time Types

- `DATE` - Only date and no time.
  - `DATE` values are displayed as `'YYYY-MM-DD'`
  - The supported range is '1000-01-01' to '9999-12-31'.
- `DATETIME` - Both date and time
  - `DATETIME` values are displayed as `'YYYY-MM-DD HH:MM:SS'`
  - The supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'.
- `TIMESTAMP` - Full timestamp (stored as UTC but displayed using current time zone)
  - `TIMESTAMP` values are displayed as `'YYYY-MM-DD HH:MM:SS'`
  - The supported range is `'1970-01-01 00:00:01'` UTC to `'2038-01-19 03:14:07'` UTC.

SQL Types - Text

- `CHAR` and `VARCHAR` are declared with a length specifying the maximumum length string that can be stored in that field
- Difference between the two
  - Maximum length for a `CHAR` field is 255 bytes, while maximum length for a `VARCHAR` field is 65,535 bytes
  - `CHAR(4)` will always use 4 bytes (shorter strings are padded with empty space)
  - `VARCHAR(4)` will use one byte to store the length of the stored string but only use the exact length
  - Example: The string `'ab'` will use 4 bytes when stored as `CHAR(4)` and 3 bytes when stored as `VARCHAR(4)`
  - Example: The string `'abcd'` will use 4 bytes when stored as `CHAR(4)` and 5 bytes when stored as `VARCHAR(4)`

SQL Types - More Text

- `BINARY` and `VARBINARY` – For binary data (length specified as number of bytes)
- `TEXT` and `BLOB` – For very long strings and binary data, respectively

Modifying Schema

- `DROP` – already seen
- Adding or deleting columns

```
-- add a new column to an existing table
ALTER TABLE movies ADD genre VARCHAR(16);

-- change type of an existing column
ALTER TABLE movies MODIFY genre VARCHAR(32);

-- delete an existing column from a table
ALTER TABLE movies DROP genre;
```

Defining Keys

- An attribute or list of attributes (say S) may be declared `PRIMARY KEY` or `UNIQUE`.
- For `UNIQUE`, two tuples cannot agree on all of the attributes in S, unless the values for S in one of the tuple is `NULL`
- For `PRIMARY KEY`, attributes in S are not allowed to have `NULL` as a value for their components

Example

```
-- create movies table with UNIQUE attribute
CREATE TABLE movies (
   title VARCHAR(128) UNIQUE,
  year INT,
  length INT,
  studioname VARCHAR(128),
  executivenumber INT
);
-- create movies table with primary key as (title, year)
CREATE TABLE movies (
   title VARCHAR(128),
  year INT,
  length INT,
  studioname VARCHAR(128),
  executivenumber INT,
  PRIMARY KEY (title,year)
);
```

References