Database

bahareh.afshinpour@univ-grenoble-alpes.fr

Université Grenoble Alpes

Main reference:
A First Course in Database Systems (and associated material) by
J. Ullman and J. Widom, Prentice-Hall

Content

- Functions of Database Management Systems
- Introduction to data models
  - Relational data model
  - Semi-structured model
- Database design
  - E/R model (UML)
- Database programming
  - Relational algebra + SQL
  - Transactions
  - Embedded SQL

Chapter 1
The words of Database Systems

What is a database?

- **Databases** today are essential to every business.
  - Major Websites (Google, Yahoo, …) or smaller sites that provide information
  - At the core of many scientific investigations
  - …..

What is a database?

A collection of information that exists over a long period of time.

- A **DBMS** is a powerful tool for creating and managing large amounts of data efficiently and allowing it to persist over long periods of time, safely.

The DBMS

- The first commercial database management system appeared in the late 1960s. These systems evolved from **file systems**.

✓Store data over a long period of time

❑Do not guarantee that data can not be lost

❑Do not control access to data from many users

❑…..

The DBMS is expected to:

- DBMSs provide many features that traditional file systems do not.

✓Allow users to create new databases

✓Give users the ability to query data and modify the data

✓Support the storage of very large amounts of data- many terabytes or more

✓Enable durability, the recovery of the database in the face of failures, errors, or intentional misuse

✓Control access to data from many users at once, without allowing unexpected interactions among users

The DBMS

A system for providing efficient, convenient, and safe multi-user storage of and access to massive amounts of persistent data.

- Example: Banking System

Data == Information on branches, accounts, customers, interest rates, transaction histories, etc.

Massive:

-Many gigabytes at least for big banks, more if keep the history of all transactions, even more, if keep images of checks.

Multiuser:

- Many people/programs accessing the same database.

Safe:

– From system failures Example: balance transfer

– From malicious users

Convenient:

– Simple commands to manipulate data: get balance, transfer funds, etc.

Efficient:

-Don’t search all files in order to: get the balance of one account, get all accounts with low balances, etc.

Outline of Database-System Studies

- Part1: Relational Database Modeling
- Part2: Relational Database programming
- Part3: Semi-Structured Data Modeling and Programming
- Part4:Database System Implementation

Chapter 2

The Relational Model of Data

What is a Data Model?

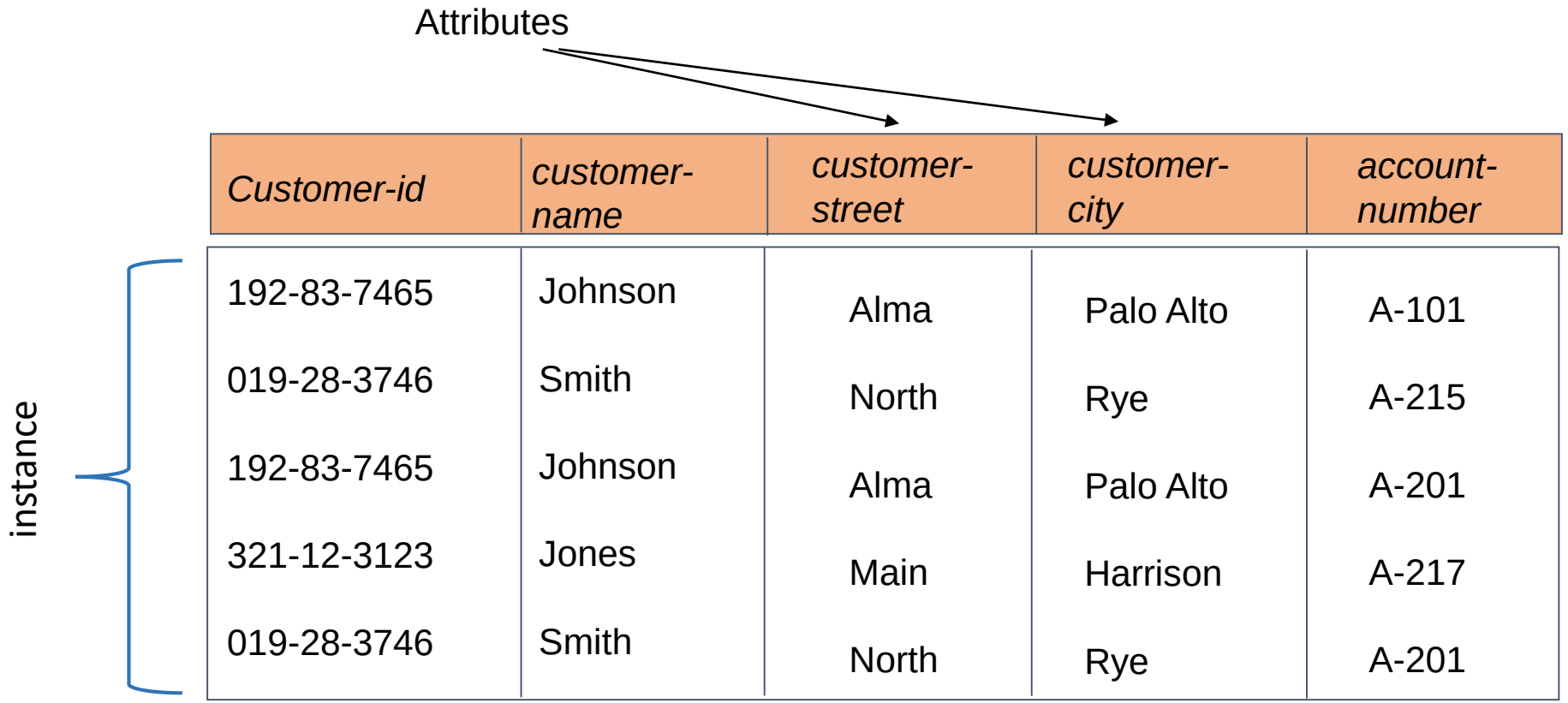
A data model is :

- A notation for describing data or information.
- It describes the conceptual structuring of data stored in the database.

- The two data models of preeminent importance for database systems are:
  - -The relational model
  - -The semi-structured data model

Relational Model

- Example of tabular data in the relational model



Customer-id	*customer-name*	*customer-street*	*customer-city*	*account-number*
192-83-7465	Johnson	Alma	Palo Alto	A-101
019-28-3746	Smith	North	Rye	A-215
192-83-7465	Johnson	Alma	Palo Alto	A-201
321-12-3123	Jones	Main	Harrison	A-217
019-28-3746	Smith	North	Rye	A-201

Basics of the Relational Model

- **Attributes**
  - The columns of a relation are named by attributes.
  - Usually, an attribute describes the meaning of entries in the column below.
- **Schemas**
  - The name of a relation and the set of attributes for a relation is called the schema for the relation.

Bank1(customer-name, customer-name, customer-street, customer-city, account-number)

The set of schemas for the relations of a database is called a **relational database schema**, or a **database schema**.

- **Tuples**
  - The rows of a relation are called tuples
  - Relations are a set of tuples not a list of tuples. (the order is immaterial)

Basics of the Relational Model

- **Instances**
  - We shall call a set of tuples for a given relation an instance of the relation.
- **Keys of relations**
  - A set of attributes forms a Key for a relation if we do not allow two tuples in a relation instance to have the same value in all the attributes of the key.
  - We indicate the attribute or attributes that form a key for a relation by underlining the key attribute(s)
  - ➢Ex: Social-security number, Student Id

Movies(title, year, genre,length)

year	*genre*	*title*	*length*
1977	sciFi	Star Wars	124
1992	comedy	Wayne's World	95
1939	drama	Gone With the Wind	231

An example database schema

```
Movies(
    title:string,
    year:integer,
    length:integer,
    genre:string,
    studioName:string,
    producerC#:integer
)
MovieStar(
    name:string,
    address:string,
    gender:char,
    birthdate:date
)
StarsIn(
    movieTitle:string,
    movieYear:integer,
    starName:string
)
```

Exercise

Exercise 2.2.2: In Section 2.2.7 we suggested that there are many examples of attributes that are created for the purpose of serving as keys of relations. Give some additional examples.

!! **Exercise 2.2.3:** How many different ways (considering orders of tuples and attributes) are there to represent a relation instance if that instance has:

a) Three attributes and three tuples

b) Four attributes and five tuples?

c) n attributes and m tuples?

Relations in SQL

- SQL also pronounced (sequel) is the principal language used to describe and manipulate the relational database.
- SQL distinguishes three kinds of relations :
  - **Stored relations (tables):** these are tables that exist in the database we can query and modify them.
  - **Views:** are relations defined by a computation. They are not stored but constructed when needed. We just query them.
  - **Temporary tables:** are constructed by SQL language processors during optimization. These are not stored nor seen by the user.

Data Types

- Char(n): a fixed-length string of up to n characters.
- Varchar(n): a variable-length string of up to n characters
- Bit(n), Varbit(n) fixed, and a variable string of up to n bits.
- Boolean: True False and although it would surprise George Boole Unknown
- Int or Integer: typical integer values
- Float or real: typical real values Decimal(6,2) could be 0123.45
- Date and time: essentially char strings with constraints.

Data Types

Character strings of fixed or varying length. The type `CHAR(n)` denotes a fixed-length string of up to n characters. `VARCHAR(n)` also denotes a string of up to n characters. The difference is implementation-dependent; typically `CHAR` implies that short strings are padded to make n characters, while `VARCHAR` implies that an endmarker or string-length is used. SQL permits reasonable coercions between values of character-string types.

Dates and times can be represented by the data types `DATE` and `TIME`, respectively (see the box on "Dates and Times in SQL"). These values are essentially character strings of a special form. We may, in fact, coerce dates and times to string types, and we may do the reverse if the string "makes sense" as a date or time.

Simple Table Declarations

- The simplest form of declaration of a relation schema consists of the keyword CREATE TABLE followed by the name of the relation and parenthesized, comma-separated list of the attribute names and their types.

```
CREATE TABLE Movies (
    title       CHAR(100),
    year        INT,
    length      INT,
    genre       CHAR(10),
    studioName  CHAR(30),
    producerC#  INT
);
```

Modifying Relation Schemas

- We can delete a table R by the following SQL command

  Drop table R;

- We can modify a table by the command

  Alter Table MovieStar **ADD** phone CHAR(16);

  Alter Table MovieStar **Drop** birthdate;

- Defaults values

  Gender CHAR(1) **DEFAULT** '?',

  Birthdate DATE **DEFAULT** '0000-00-00',

  ALTER TABLE MovieStar ADD phone CHAR (16) **DEFAULT** ' unlisted';

Declaring Keys

- There are two ways to declare an attribute or set of attributes to be a key:

```
CREATE TABLE MovieStar (
    name CHAR(30) PRIMARY KEY,
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE
);
```

```
CREATE TABLE MovieStar (
    name CHAR(30),
    address VARCHAR(255),
    gender CHAR(1),
    birthdate DATE,
    PRIMARY KEY (name)
);
```

However, in a situation where the key has more than one attribute, we must use this style

- Replace **primary** with **unique** in examples to get the example with unique

We could also substitute `UNIQUE` for `PRIMARY KEY` in this declaration. If we did so, then two or more tuples could have `NULL` as the value of `name`, but there could be no other duplicate values for this attribute.

2.3.7 Exercises for Section 2.3

Exercise 2.3.1: In this exercise we introduce one of our running examples of a relational database schema. The database schema consists of four relations, whose schemas are:

```
Product(maker, model, type)
PC(model, speed, ram, hd, price)
Laptop(model, speed, ram, hd, screen, price)
Printer(model, color, type, price)
```

Write the following declarations:

a) A suitable schema for relation `Product`.

b) A suitable schema for relation `PC`.

c) A suitable schema for relation `Laptop`.

d) A suitable schema for relation `Printer`.

e) An alteration to your `Printer` schema from (d) to delete the attribute `color`.

f) An alteration to your `Laptop` schema from (c) to add the attribute `od` (optical-disk type, e.g., cd or dvd). Let the default value for this attribute be `'none'` if the laptop does not have an optical disk.