Introduction to Databases

COMP 1531, 17s2
Aarthi Natarajan
Week 7
What is data?

- **Data** – facts that can be recorded and have implicit meaning (Elmasri & Navathe)
- Today data is being generated at an exponential rate
  - Financial market data, posts to social media sites, growing logs of web transactions, computation physics...BIG Data
Why do we need a database?

• Data by itself is not very useful.
• Give a context to data to transform data into information e.g., the numbers 45, 55, 67 do not mean much, but given a context such as these are the marks of students in COMP 1531, this is now information

DATA -> INFORMATION -> DECISION

• This data needs to be:
  – Stored
  – Manipulated
  – Shared
  – Transmitted
• Red text handled by databases; green by networks.
• **Databases are everywhere...**
  
  – The Internet uses databases extensively
    
    • Google, Ebay, Amazon, iTunes Shop
    • Library catalogues, Train time tables, Airline bookings
    • Bank accounts, credit card, debit card
    • Medical records (Medicare), Tax Office
    • Facebook, Twitter, ...
  
  – Every time you use a loyalty card, you're inputting information about your buying habits into the database of the company you are buying from

• **Challenges in building effective databases**

  – efficiency, security, scalability, maintainability, availability, integration, new media types (e.g. music), ...
What is a database?

• A **database** is a logically coherent collection of related data.

• A **database management system (DBMS)** is an application or collection of programs that allows users to:
  – create and maintain a database (DDL)
  – defining **queries** that causes data to be retrieved from the database
  – Perform **transactions** that cause some data to be written or deleted from the database (DML)

• A database and DBMS are collectively referred to as a **database system**
A DBMS provides:
- Persistence
- Concurrency
- Integrity
- Security

Examples of DBMS
- Open Source
  - SQLite
  - PostgreSQL
  - MySQL
- Commercial
  - Oracle
  - DB2 (IBM)
  - MS SQL Server
What is a relational DBMS?

A **Relational Database Management System** (RDBMS) is a special type of DBMS that:

- Stores data as *tuples* or *records* in *tables*
- Allows the user to create *relationships* between tables
Defining more database terminology

• A **data model** describes how the data is structured in the database

• A **database schema** adheres to a data model and provides a logical view of the database, defining how the data is organised and the relationships between them and is typically set up at the beginning

• A **database instance** is the state of the database at a particular instance of time
Data Models

• There are several types of data models
  – Relational model
    • a data structure where data is stored as a set of records known as tables
    • a table consists of rows of information (also called a tuple)
    • each row contains fields known as columns

      ![Student Table](image)

  – Document model
    • data is stored in a hierarchical fashion e.g., XML
  – Object-oriented model
    • a data structure where data is stored as a collection of objects
  – Object-Relational model
    • a hybrid model that combines the relational and the object-oriented database models
Goals of this course

Understand Relational Model

Understand ER model and ER-to-relational mapping

Database Application Development

- SQLite: sqlite3 (SQL shell)
- SQL (Structured Query Language, a standard that allows you to access different DBMS)
- Programming language access to databases (using Python, ORM)
Relational Data Model

The relational data model describes the world as a collection of inter-connected relations (or tables)

Goal of relational model:
• a simple, general data modelling formalism
• maps easily to file structures (i.e. implementable)

Relational model has two styles of terminology:

<table>
<thead>
<tr>
<th>mathematical</th>
<th>relation</th>
<th>tuple</th>
<th>attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>data-oriented</td>
<td>table</td>
<td>record (row)</td>
<td>field (column)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STUDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Smith</td>
</tr>
<tr>
<td>Brown</td>
</tr>
</tbody>
</table>
Relational Data Model

The relational model has one structuring mechanism ...
• a relation corresponds to a mathematical "relation"
• a relation can also be viewed as a "table"

Each relation (table) (denoted \(R, S, T, \ldots\)) has:
• a name (unique within a given database)
• a set of attributes (or column headings)

Each attribute (denoted \(A, B, \ldots\) or \(a_1, a_2, \ldots\)) has:
• a name (unique within a given relation)
• an associated domain (set of allowed values)

DB definitions also make extensive use of constraints
Example of a relation (table): Bank Account

<table>
<thead>
<tr>
<th>branchName</th>
<th>accountNo</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A–101</td>
<td>500</td>
</tr>
<tr>
<td>Mianus</td>
<td>A–215</td>
<td>700</td>
</tr>
<tr>
<td>Perryridge</td>
<td>A–102</td>
<td>400</td>
</tr>
<tr>
<td>Round Hill</td>
<td>A–305</td>
<td>350</td>
</tr>
<tr>
<td>Brighton</td>
<td>A–201</td>
<td>900</td>
</tr>
<tr>
<td>Redwood</td>
<td>A–222</td>
<td>700</td>
</tr>
<tr>
<td>Brighton</td>
<td>A–217</td>
<td>750</td>
</tr>
</tbody>
</table>
Relational Data Model

A *tuple (row)* is a set of *values (attribute or column values)*

Attribute values:

- Are **atomic** (no composite or multi-valued attributes).
- Belong to a **domain** which is given a **name, data type** and **format**
- A distinguished value **NULL** belongs to all domains.
- **NULL** has several interpretations: none, don't know, irrelevant

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Domain Name</th>
<th>Domain Data Type, Format and Constrain</th>
</tr>
</thead>
<tbody>
<tr>
<td>phone_number</td>
<td>local_phone_numbers - (set of phone numbers valid in australia)</td>
<td>character string of the format <em>(dd)</em> <code>dddddddd</code>, where each <code>d</code> is a numeric (decimal) digit and the first two digits form a valid telephone area code.</td>
</tr>
<tr>
<td>age</td>
<td>employee_age (set of possible ages for employees in the company)</td>
<td>An integer value between 15 and 80</td>
</tr>
</tbody>
</table>
Relational Data Model

- A relation(table) is a set of tuples.
- Since a relation is a set, there is no ordering on rows.
- Normally, we define a standard ordering on components of a tuple.
- The following are different presentations of the same relation:

<table>
<thead>
<tr>
<th>branchName</th>
<th>accountNo</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downtown</td>
<td>A-101</td>
<td>500</td>
</tr>
<tr>
<td>Mianus</td>
<td>A-215</td>
<td>700</td>
</tr>
<tr>
<td>Perryridge</td>
<td>A-102</td>
<td>400</td>
</tr>
<tr>
<td>Round Hill</td>
<td>A-305</td>
<td>350</td>
</tr>
<tr>
<td>Redwood</td>
<td>A-222</td>
<td>700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>accountNo</th>
<th>branchName</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-305</td>
<td>Round Hill</td>
<td>350</td>
</tr>
<tr>
<td>A-222</td>
<td>Redwood</td>
<td>700</td>
</tr>
<tr>
<td>A-215</td>
<td>Mianus</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>Perryridge</td>
<td>400</td>
</tr>
<tr>
<td>A-101</td>
<td>Downtown</td>
<td>500</td>
</tr>
</tbody>
</table>

- Each relation has a key (subset of attributes, unique over relation)
Relational Data Model

Example of a database with collection of relations (or tables)

<table>
<thead>
<tr>
<th>Account</th>
<th>Branch</th>
</tr>
</thead>
<tbody>
<tr>
<td>branchName</td>
<td>accountNo</td>
</tr>
<tr>
<td>Downtown</td>
<td>A–101</td>
</tr>
<tr>
<td>Mianus</td>
<td>A–215</td>
</tr>
<tr>
<td>Perryridge</td>
<td>A–102</td>
</tr>
<tr>
<td>Round Hill</td>
<td>A–305</td>
</tr>
<tr>
<td>Brighton</td>
<td>A–201</td>
</tr>
<tr>
<td>Redwood</td>
<td>A–222</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Customer</th>
<th>Depositor</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>address</td>
</tr>
<tr>
<td>Smith</td>
<td>Rye</td>
</tr>
<tr>
<td>Jones</td>
<td>Palo Alto</td>
</tr>
<tr>
<td>Smith</td>
<td>Brooklyn</td>
</tr>
<tr>
<td>Curry</td>
<td>Rye</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Given a relation $R$ which has:

- $n$ attributes $a_1, a_2, \ldots a_n$
- with corresponding domains $D_1, D_2, \ldots D_n$

We define:

- Relation schema of $R$ as: $R(a_1:D_1, a_2:D_2, \ldots a_n:D_n)$
- Tuple of $R$ as: an element of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. list of values)
- Instance of $R$ as: subset of $D_1 \times D_2 \times \ldots \times D_n$ (i.e. set of tuples)
- Database (instance) : a collection of relation instances.
We often use $R$ as a synonym for $R(a_1, a_1, \ldots a_n)$

e.g., the *Accounts* schema which has 3 attributes *branchName*, *accountNo*, *balance* with corresponding domains *string, string, int* can be defined as:

\[
\text{Account} \ (\text{branchName}:\text{string}, \text{accountNo}:\text{string}, \text{balance}:\text{int}) \quad \text{OR} \\
\text{Account} \ (\text{branchName}, \text{accountNo}, \text{balance})
\]

and a tuple of R (row of R) can be specified as:

(Downtown, A-101, 500)

and an account instance (set of tuples or rows)

* No duplicates

The **degree (or arity)** of a relation is the number of attributes $n$ of its relation schema, so a relation schema $R$ of degree $n$ is denoted by $R(a_1, a_2, \ldots, a_n)$.

e.g. A relation of degree seven, which stores information about university students, would contain seven attributes describing each student. as follows:

$$\text{Student} \ (\text{name}, \ \text{ssn}, \ \text{home\_phone}, \ \text{address}, \ \text{office\_phone}, \ \text{age}, \ \text{gpa})$$

OR as

$$\text{Student} \ (\text{name: string,} \ \text{ssn: string,} \ \text{home\_phone: string,} \ \text{address: string,} \ \text{office\_phone: string,} \ \text{age: integer,} \ \text{gpa: int})$$
Changing Relation Schema and Relations

In making changes to relations, it is …

• easy to add new tuples (rows) \((\text{relation insert})\)
• east to change attribute values in tuples \((\text{relation update})\)
• but, difficult to add new attributes (columns) \((\text{schema update})\)

The reasons:

• \text{relation update} \Rightarrow \text{insertion of one new tuple into a set}
  \hspace{1cm} \text{(in file terms: writing one record to the end of a data file)}
• \text{schema update} \Rightarrow \text{insertion of new data into every tuple}
  \hspace{1cm} \text{(in file terms: re-writing the entire file to modify each record)}
To remember:

- DBMS-level ... database names must be unique
- database-level ... schema names must be unique
- schema-level ... table names must be unique
- table-level ... attribute names must be unique

Sometimes it is convenient to use same name in several tables

We distinguish which attribute we mean using qualified names

\texttt{e.g. Account.branchName} \texttt{vs} \texttt{Branch.branchName}
Relational Model vs DBMS

The relational model is a mathematical theory
• giving a representation for data structures
• with constraints on relations/tuples
• and an algebra for manipulating relations/tuples (union, intersect...)

Relational DBMSs
• provide an implementation of the relational model
• using SQL as language for:
  – data definition (creating, deleting relations i.e. tables)
  – query (selecting tuples)
  – update (changing relations)
Describing Relational Schemas

• SQL (Structured Query Language) provides the formalism to express relational schemas

• SQL provides a Data Definition Language (DDL) for creating relations

```sql
CREATE TABLE TableName (  
attrName_1 domain_1 constraints_1 ,  
attrName_2 domain_2 constraints_2 , ...  
PRIMARY KEY (attr_i, attr_j, ...)  
FOREIGN KEY (attr_x, attr_y, ...)  
REFERENCES OtherTable (attr_m, attr_n, ...)  
);
```
SQL Syntax in a NutShell

• Comments: everything after -- *is a comment*
• Identifiers: alphanumeric (a la C), but also "An Identifier"
• Reserved words: many e.g. CREATE, SELECT, TABLE, ...
• Strings: e.g. 'a string', 'don''t ask', but no '\n'
• Numbers: like C, e.g. 1, -5, 3.14159, ...
• Identifiers and reserved words are case-insensitive:
  TableName = tablename = TaBLeNamE != "TableName"
• Types: integer, float, char(n), varchar(n), date, ...
• Operators: =, <>, <, <=, >, >=, AND, OR, NOT, ...
Defining tables:

```
CREATE TABLE Name ( Attributes, Constraints )
```

Defining attributes:

```
Name Domain [ Constraint ]  (constraint is optional)
```

Defining keys:

```
PRIMARY KEY ( AttrNames )
FOREIGN KEY ( AttrNames )
    REFERENCES Table ( AttrNames )
```

Defining constraints:

```
[ CONSTRAINT Name ] CHECK ( ExpressionOnAttributes )
```
Exercise: Simple Relational Schema (i)

Express the following in SQL DDL:

Assume only two domains: string and integer
Creating a database

- Choose a **data model** for the database
- Set up the structure of the database by defining a **database schema** for the database (e.g., for a relational data model, define the tables, rows and columns or field names and types of fields, constraints and relationships)
- Create the initial state of the data by loading data into the database
- After this, typically schema doesn’t change much, but data changes rapidly as new data is loaded or existing data is updated
Entity-Relationship Model
Entity-Relationship Data Modelling

The world is viewed as a collection of inter-related entities.

ER has three major modelling constructs:

- **attribute**: data item describing a property of interest
- **entity**: collection of attributes describing object of interest
- **relationship**: association between entities (objects)
  - e.g. employee works-on project
- The ER model is not a standard, so many variations exist.