

# CMPT 354: Database System I

Lecture 2. Relational Model

# Outline

- An overview of data models
- Basics of the Relational Model
- Define a relational schema in SQL

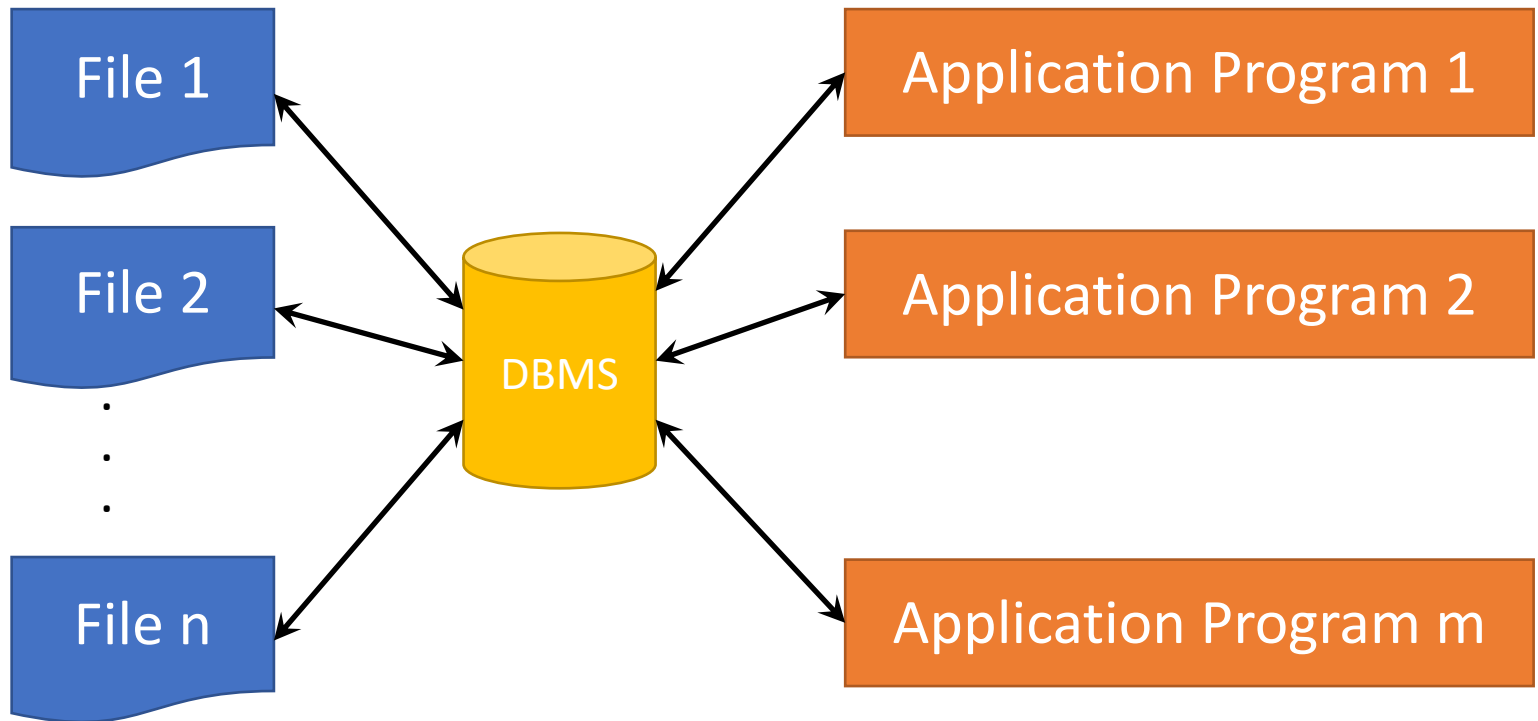
# Outline

- **An overview of data models**
- Basics of the Relational Model
- Define a relational schema in SQL

# Review

- What is a database?
  - A collection of files that store related data
- What is a DBMS?
  - A piece of software designed to store and manage databases

# Data Storage with DBMS



# Data Model

- Data Model
  - mathematical formalism (or conceptual way) for describing the data
- The description generally consists of three parts:
  - Structure of the data
  - Operations on the data
  - Constraints on the data

# Structure of the data

- Schema (e.g., table names, attribute names)
  - Describe the **conceptual** structure of the data
- Different from data structure (e.g., list, array)
  - Data structure can be seen as a **physical** data model

# Operations on the data

- Query language (e.g., SQL)
  - Describe what operations that can be performed on data
- Two kinds of operations
  - operations that retrieve information
  - operations that change the database
- Different from programming languages (e.g., C, Java)
  - Support a set of limited operations
  - Allow for query optimizations



# Constraints on the data

- Constraints (e.g.,  $\text{age} > 0$ , student# is unique)
  - describe limitations on what the data can be.
- Different kinds of constraints
  - Domain constraints
  - Integrity constraints
- Why does it matter?
  - Ensure the correctness of data

# Commonly Used Data Models

- Relational Data Model
- Key-Value Data Model
- Semi-structured Data Model (e.g., Json, XML)

# The Relational Model in Brief

**Students**

<b>Id</b>	<b>Name</b>	<b>Age</b>	<b>GPA</b>
1000	Mike	21	3.8
1001	Bill	19	3.4
1002	Alice	20	3.6

- Structure of the data
  - Table structure
- Operations on the data
  - SQL
- Constraints on the data
  - E.g., id is unique, age > 10, name is not NULL

# The Key-Value Model in Brief

Key → Value
1000 → (Mike, 21, 3.8)
1001 → (Bill, 19, 3.4)
1002 → (Alice, 20, 3.6)

- Structure of the data
  - (Key, Value) pairs
  - Key is an integer/string, value can be any object
- Operations on the data
  - get(key), put(key, value)
- Constraints on the data
  - E.g., key is unique, value is not NULL

# The Semistructured Model in Brief

- Structure of the Data
  - Tree structure
- Operations on the data
  - XPath
- Constraints
  - E.g., <Age> has to be integer, each <Student> has a <Name> element nested within in

```
<Students>
  <Student id=1000>
    <Name>Mike</Name>
    <Age>20</Age>
    <GPA>3.8</GPA>
  </Student>
  <Student id=1001>
    <Name>Bill</Name>
    <Age>19</Age>
    <GPA>3.4</GPA>
  </Student>
  <Student id=1002>
    <Name>Alice</Name>
    <Age>21</Age>
    <GPA>3.6</GPA>
  </Student>
</Students>
```

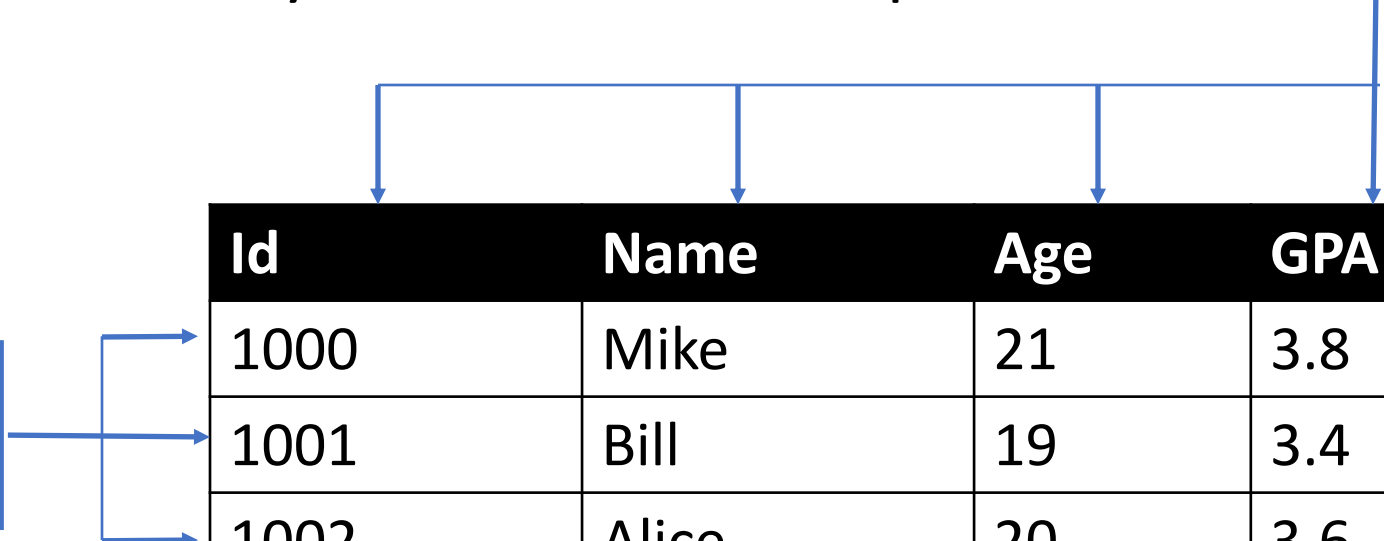
# Outline

- An overview of data models
- **Basics of the Relational Model**
- Define a relational schema in SQL

# Terminology

- Relations/Tables
- Columns/Attributes/Fields
- Rows/Tuples/Records
- Degree (arity) of a relation = #attributes
- Cardinality of a relation = #tuples

Columns/  
Attributes/  
Fields



Id	Name	Age	GPA
1000	Mike	21	3.8
1001	Bill	19	3.4
1002	Alice	20	3.6

Rows/  
Tuples/  
Records

# Schema

- Relation schema
  - The name of a relation + The set of attributes for a relation

**Student(id, sname, age, gpa)**

- Database schema
  - The set of schemas for the relations of a database

Suppose your database has 3 relations.

**Student (sid, sname, age, gpa)**

**Take (sid, cid)**

**Course (cid, cname, credit)**



# Domains

- Each attribute has a domain (data type)
- Examples
  - Text: CHAR(20), VARCHAR(50), TEXT
  - Integer: INT, SMALLINT
  - Real: DOUBLE, FLOAT
  - Few more that are vendor specific

**Student(id INT, sname VARCHAR(50), age INT, gpa FLOAT)**

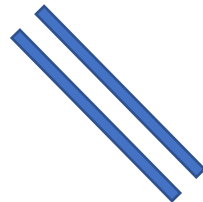
# Equivalent Representations of a Relation

- Order does NOT matter!

Id	Name	Age	GPA
1000	Mike	21	3.8
1001	Bill	19	3.4
1002	Alice	20	3.6



Id	Name	GPA	Age
1000	Mike	3.8	21
1001	Bill	3.4	19
1002	Alice	3.6	20



Id	Name	Age	GPA
1000	Mike	21	3.8
1002	Alice	20	3.6
1001	Bill	19	3.4

# Exercise-1: Terminology

## Accounts

AcctNo	Type	Balance
12345	savings	12000
23456	checking	1000
34567	savings	25

## Customers

fname	lname	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

1. List two other terms of “rows”
2. List two other terms of “columns”
3. List another term of “table”

# Exercise-2: Terminology

## Accounts

AcctNo	Type	Balance
12345	savings	12000
23456	checking	1000
34567	savings	25

## Customers

fname	lname	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

4. Indicate the attributes of each relation
5. Indicate the tuples of each relation
6. Indicate the degree of each relation
7. Indicate the cardinality of each relation

# Exercise-3: Terminology

## Accounts

AcctNo	Type	Balance
12345	savings	12000
23456	checking	1000
34567	savings	25

## Customers

fname	lname	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

8. Indicate the schema for each relation
9. Indicate the database schema
10. Specify a suitable domain for each attribute

# Exercise-4: Terminology

## Customers

fname	lname	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456

11. Indicate another equivalent way to represent this relation
12. How many different ways to represent this relation?

# Discussions

- Tables are NOT ordered
  - They are sets or multisets (bags)
- Tables DO NOT prescribe how they are implemented/stored on disk
  - This is called physical data independence

# Keys


- Key = one (or multiple) attributes that uniquely identify a record

AcctNo	Type	Balance
12345	savings	12000
23456	checking	1000
34567	savings	25



# Keys

- Key = one (or multiple) attributes that uniquely identify a record

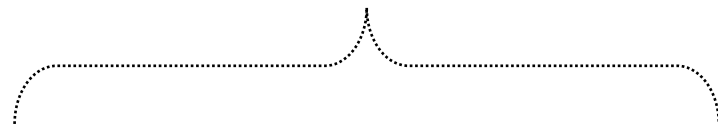


AcctNo	Type	Balance
12345	savings	12000
23456	checking	1000
34567	savings	25

# Multiple-attribute Key

- Multiple-attribute Key = multiple attributes that uniquely identify a record

Key = fname, lname



<b>fname</b>	<b>lname</b>	<b>age</b>	<b>salary</b>
Robbie	Banks	20	10k
Alice	Banks	30	8k
Alice	Smith	25	12k

# Multiple Keys



SIN	fname	lname	age	salary
123-456-789	Robbie	Banks	20	10k
222-111-709	Alice	Banks	30	8k
345-498-712	Alice	Smith	25	12k

- We can choose one key as primary key (e.g., SSN)

# Foreign Key

- Attribute(s) whose value is a key of a record in some other relation

**Accounts**

acctNo	type	balance
12345	savings	12000
23456	checking	1000
34567	savings	25

**Customers**

fname	lname	idNo	account
Robbie	Banks	901-222	12345
Lena	Hand	805-333	12345
Lena	Hand	805-333	23456



Foreign key to  
Accounts.acctNo

# Outline

- An overview of data models
- Basics of the Relational Model
- **Define a relational schema in SQL**

# SQL DDL

- *SQL* stands for *Structured Query Language*
- SQL is divided into two parts
  - *Data Manipulation Language (DML)* which allows users to create, modify and query data
  - *Data Definition Language (DDL)* which is used to define external and conceptual schemas
- The *DDL* supports the creation, deletion and modification of tables
  - Including the specification of domain constraints and other constraints

# Creating Tables

- To create a table use the **CREATE TABLE** statement
  - Specify the table name, field names and domains

```
CREATE TABLE Customer (  
    sin          CHAR(11),  
    firstName    CHAR(20),  
    lastName     CHAR(20),  
    age          INTEGER,  
    income       REAL)
```

Question – is SQL case sensitive?

Answer – SQL keywords (create and table for example) are not case sensitive.

Named objects (tables, columns etc.) *may* be.

# Inserting Records

- To insert a record into an existing table use the **INSERT** statement
  - The list of column names is optional
    - If omitted the values must be in the same order as the columns

```
INSERT INTO Customer(sin, firstName, lastName, age, income)  
VALUES ('111', 'Sam', 'Spade', 23, 65234)
```



# Deleting Records

- To delete a record use the **DELETE** statement
  - The **WHERE** clause specifies the record(s) to be deleted

```
DELETE  
FROM Customer  
WHERE sin = '111'
```

- Be careful, the following SQL query deletes *all* the records in a table

```
DELETE  
FROM Customer
```

# Modifying Records

- Use the **UPDATE** statement to modify a record, or records, in a table
  - Note that the **WHERE** statement is evaluated *before* the **SET** statement
- Like **DELETE** the **WHERE** clause specifies which records are to be updated

```
UPDATE Customer  
SET age = 37  
WHERE sin = '111'
```

# Deleting Tables

- To delete a *table* use the **DROP TABLE** statement
  - This not only deletes all of the records but also deletes the table schema

```
DROP TABLE Customer
```

# Modifying Tables

- Columns can be added or removed to tables using the **ALTER TABLE** statement
  - **ADD** to add a column and
  - **DROP** to remove a column

```
ALTER TABLE Customer  
ADD height INTEGER
```

```
ALTER TABLE Customer  
DROP height
```

# Acknowledge

- Some lecture slides were copied from or inspired by the following course materials
  - “W4111: Introduction to databases” by Eugene Wu at Columbia University
  - “CSE344: Introduction to Data Management” by Dan Suciu at University of Washington
  - “CMPT354: Database System I” by John Edgar at Simon Fraser University
  - “CS186: Introduction to Database Systems” by Joe Hellerstein at UC Berkeley
  - “CS145: Introduction to Databases” by Peter Bailis at Stanford
  - “CS 348: Introduction to Database Management” by Grant Weddell at University of Waterloo