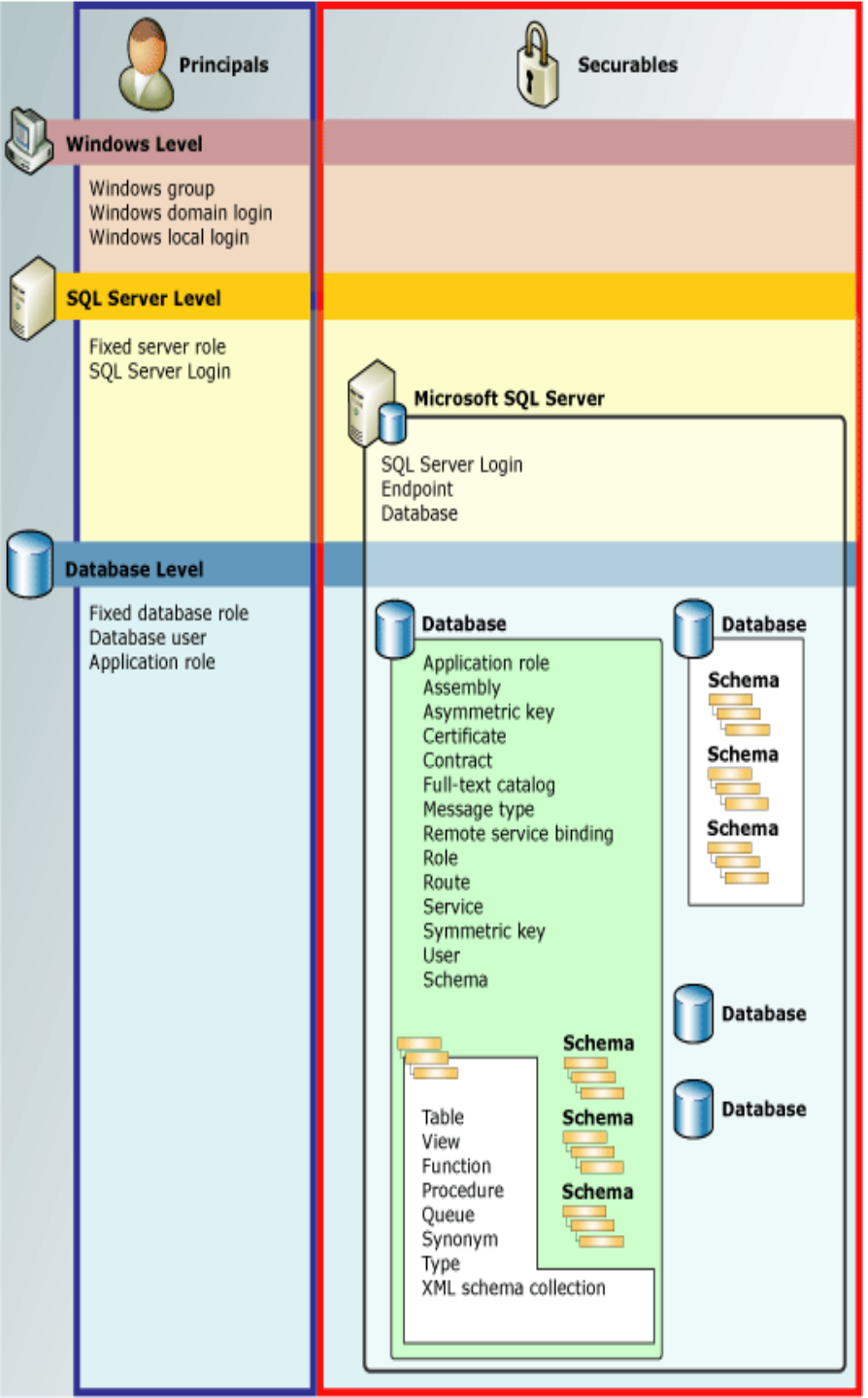
Chapter 2

RELATIONAL DATABASE

XIAMEN UNIVERSITY MALAYSIA
厦門大學馬來西亞分校



OUTLINE

Part 1 Relational Model Concepts

Part 2 Basic Structure

Part 3 Relational Algebra

Part 1 Relational Model Concepts

Relational Model Concepts

- Represent **data as a collection of relations**
- **Table** of values
  - Each **row (*tuple*)**
    - Represents a **record of related data values**
    - Facts that typically correspond to a real-world entity or relationship
  - Each **column (*attribute*)**
    - Holds a corresponding value for each row
    - Slot for a specific interpretation for a row

Relational Model Concepts

- Relational database: a set of relations.
- **Relation: consists of two parts:-**
  1. **Schema describes table**
     - Table name, attribute names and types
       - Specifies the **relation's name**, as well as the **name** and type of each **column**.

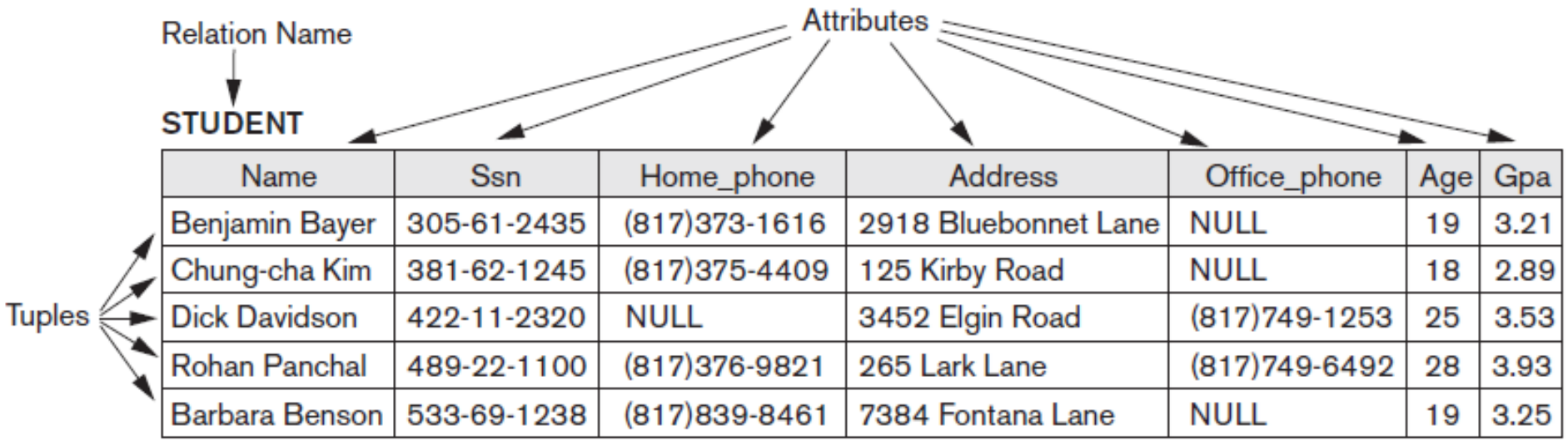
         E.g. **Students(sid: string, name: string, login: string, age: integer, gpa: real)**

  2. **The current contents of the table are indicated by instance.**
     - Table of columns and rows

       #rows = cardinality

       #fields = degree

Example



Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	(817)373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	(817)375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	(817)749-1253	25	3.53
Rohan Panchal	489-22-1100	(817)376-9821	265 Lark Lane	(817)749-6492	28	3.93
Barbara Benson	533-69-1238	(817)839-8461	7384 Fontana Lane	NULL	19	3.25

#rows = cardinality = 5 (Number of Tuples)
#fields = degree = 7 (Number of Attributes)

Part 2

Basic Structure

Characteristics Of Relations

- **A Relation initially was a mathematical concept based on the ideas of sets.**

  product_name = {keyboard, mouse, pendrive, CD, …} /* Set of all product names */

  product_origin = {KL, Perak, Kedah, …} /* Set of all product's origin*/

- **The relation above also have keys, such as:**
  - **Primary Key, Foreign Key**

CONSTRAINTS

- **Constraints are conditions** that must **hold on all valid relation states**.
- There are three main types of (**explicit schema based**) constraints that can be expressed in the relational model:

Key constraints

Entity integrity constraints

Referential integrity constraints

- Another schema based constraint is the **domain constraint**.
  - Any value in a tuple must be from the attribute's domain (or NULL if that option is available).

Key Constraints

- Constraints are the **rules to follow** when entering data into database table columns.
- Constraints ensure that the user's data in columns **meets the condition's requirement**.

NOT NULL:
- ensures that the specified ***column doesn't contain a NULL value.***

UNIQUE :
- ***provides a unique/distinct values*** to specified columns.

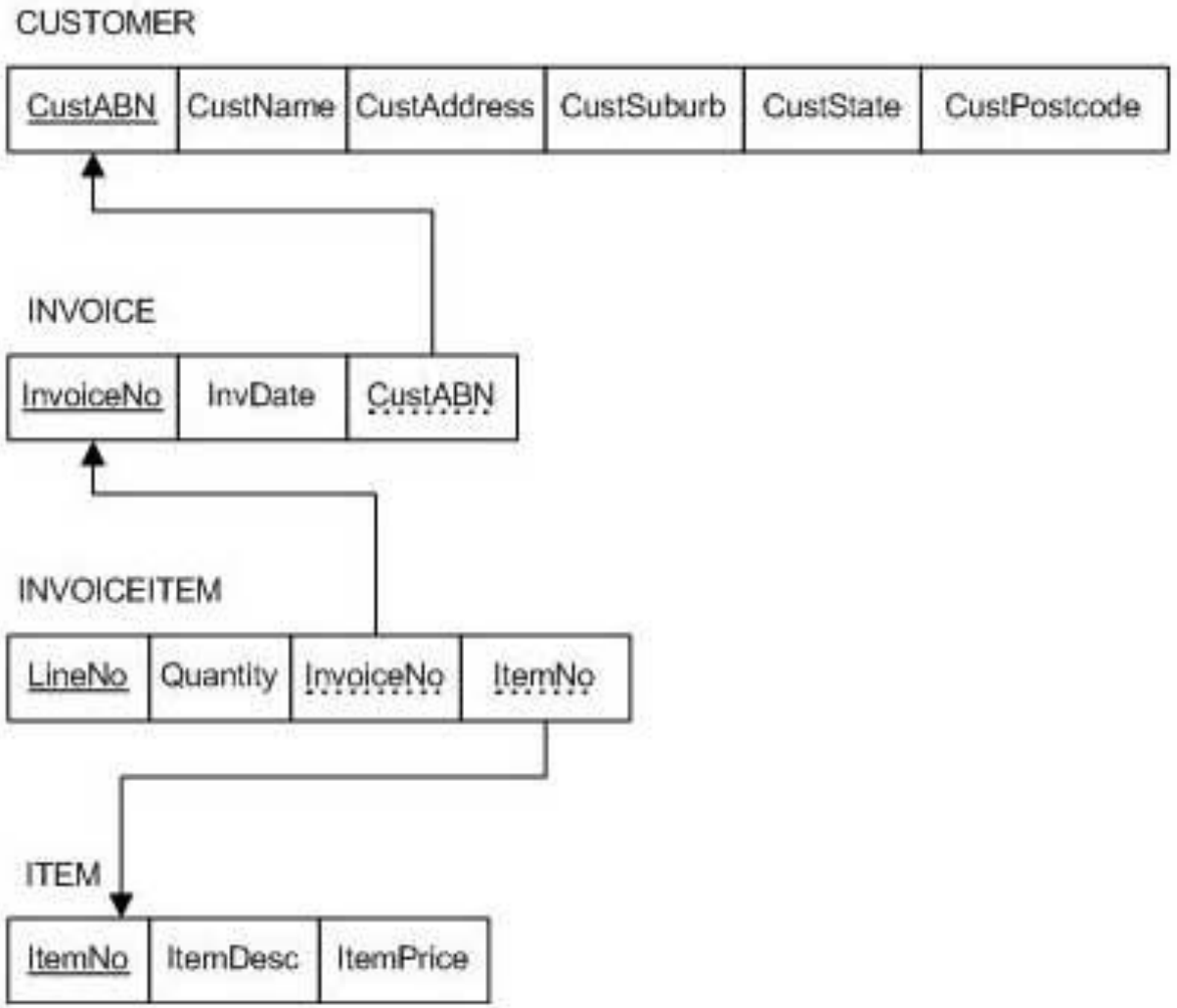
DEFAULT:
- ***provides a default value to a column*** if none is specified.

CHECK :
- ***checks for the predefined conditions before inserting*** the data inside the table.

PRIMARY KEY:
- it ***uniquely identifies a row*** in a table.

FOREIGN KEY:
- ensures ***referential integrity*** of the relationship

Relational Database Schema



Update Operation in Relation

- **INSERT a tuple.**
- **DELETE a tuple.**
- **MODIFY a tuple.**

Class Exercise

Consider the following relations for a database that keeps track of student enrollment in courses and the books adopted for each course:

STUDENT (SID, Name, Major, Bdate)

COURSE (Course#, Cname, Dept, TID)

ENROLL (Enroll#, Course#, Quarter, Grade, SID, Semester)

TEACHER (TID, Qualification, Position)

Draw a relational database schema diagram specifying the foreign keys for this schema. Do underline the primary key.

Part 3 (Week 3)

Relational Algebra

Query Language

- A Language which is used to **store and retrieve data** from database is known as query language.
  - For example – **SQL**

Procedural Query language

- **User give direction** to the system to perform a series of operations to produce the desired output.
- **User tells what data to be retrieved** from database and **how to** retrieve it.

Non-procedural query language

- **User tell the system** to come up with output **without telling how to** do it.
- Users **tells what data** to be retrieved from database but **doesn't tell how to** retrieve it.

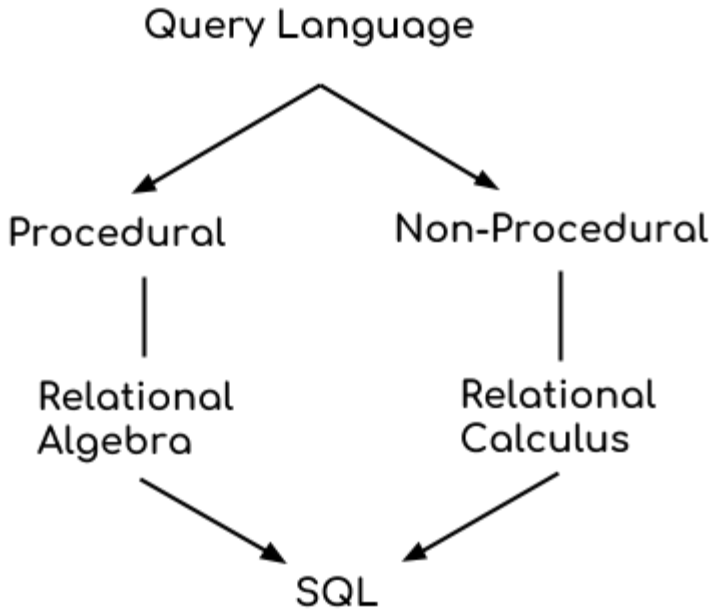
Relational Algebra & Relational Calculus

- **Relational Algebra:**
  - **Relational algebra** is a conceptual **procedural query language** used on relational model.

- **Relational Calculus:**
  - **Relational calculus** is a conceptual **non-procedural query language** used on relational model.

Difference between Relational Algebra, Calculus, RDBMS & SQL:

- **Relational algebra and calculus** are the **theoretical concepts used on relational model**.
- **RDBMS** is a **practical implementation of relational model**.
- SQL is a **practical implementation of relational algebra and calculus**.



Relational Algebra in Database

- Its based on the **procedural query language**.
  - to retrieve data from database or perform various operations such as **insert, update, delete** on the data.
- There are TWO categories of procedural language operation :

Basics
• Select (σ) • Project (∏) • Union (∪) • Set Difference (-) • Cartesian product (X) • Rename (ρ)

Derived
• Natural Join (⋈) • Left, Right, Full outer join (⟕, ⟖, ⟗) • Intersection (∩) • Division (÷)

Basics

- Select (σ)
- Project (∏)
- Union (∪)
- Set Difference (-)
- Cartesian product (X)
- Rename (ρ)

Select (σ)

Represented by sigma (**σ**) and it is used to **find the rows** in a **table** which satisfy the **given condition** **(WHERE statement in SQL)**.

```
σ Condition (Table name)
```

σ City= “Shenzen" (STUDENT)

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

Stu_Id	Stu_Name	City	Stu_Age
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

Select (σ)

Display the information about the Student from Shenzen and age his/her above 20.

σ City = "Shenzen" AND Stu_Age > 20 (STUDENT)

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

Stu_Id	Stu_Name	City	Stu_Age
103	Liyau	Shenzen	21

Project (∏)

- Used to **select specific columns** (or attributes) from a table (or relation).
- Similar to **SELECT** statement in SQL.

```
∏ column_name1, column_name2, ....
(table_name)
```

Lets say we want to **select only two column** form Student table

∏ City, Stu_Age (STUDENT)

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

City	Stu_Age
Kuala Lumpur	21
Selangor	19
Shenzen	21
Shenzen	20

Union (∪)

Used to **select all the rows** (tuples) **from two tables** (relations).

```
table_name1 ∪ table_name2
```

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

COURSE

Stu_Id	Stu_Name
104	Jason
105	Maria
106	Xuan

$\prod$ Stu_Name (STUDENT)

$\cup \prod$ Stu_Name (COURSE)

Stu_Name
Hamzah
Ajeet
Liyau
Jason
Maria
Xuan

Intersection (∩)

- Used to **select common rows** (tuples) **from two tables** (relations).

```
table_name1 ∩ table_name2
```

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

COURSE

Stu_Id	Stu_Name
104	Jason
105	Maria
106	Xuan

Stu_Name
Jason

∏ Stu_Name (STUDENT) ∩ ∏ Stu_Name (COURSE)

Difference -

Used to **select all tuples(rows)** that are **present in Relation R1 but not present in Relation R2**

```
table_name1 - table_name2
```

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

COURSE

Stu_Id	Stu_Name
104	Jason
105	Maria
106	Xuan

∏ Stu_Name (STUDENT) - ∏ Stu_Name (COURSE)

Stu_Name
Hamzah
Ajeet
Liyau

Cartesian Product (X)

If we have two relations R1 and R2 then the Cartesian product of these two relations (R1 X R2) would **combine each tuple of first relation R1** with the **each tuple of second relation R2**.

R1 X R2

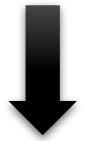
STUDENT : B

Stu_Id	Stu_Name
101	Hamzah
102	Ajeet

COURSE : C

Course_Id	Course_Name
CST403	Database
CST104	Intro

B X C



Stu_Id	Stu_Name	Course_Id	Course_Name
101	Hamzah	CST403	Database
101	Hamzah	CST104	Intro
102	Ajeet	CST403	Database
102	Ajeet	CST104	Intro

Matrix 2 x 2

Rename (ρ)

Used to change or **rename the table** (relation name) or **attributes** of the table.

ρ(new_relation_name, old_relation_name)

STUDENT

Stu_Id	Stu_Name	City	Stu_Age
101	Hamzah	Kuala Lumpur	21
102	Ajeet	Selangor	19
103	Liyau	Shenzen	21
104	Jason	Shenzen	20

Rename the attributes (Stu_Age) from table above to "Age"

ρ(Age, ∏(Stu_Age)(STUDENT))

Age
21
19
21
20

Links to study

https://www.youtube.com/watch?v=KaIRmVD-v3U → Basic operator of relational algebra

https://www.youtube.com/watch?v=xZELQc11ACY → Intersection and Union Set Concepts

https://www.mathsisfun.com/sets/venn-diagrams.html → Venn Diagram explanation