COMP 7990

Principles and Practices of Data Analytics

Data Management - I

Dr. Zhang Lu

Outline

- **Introduction to Databases**
- Relational Data Model
- Introduction to SQL

Why Database?

- The purpose of a database is:
  - To store data
  - To provide an organizational structure for data
  - To provide a mechanism for interacting with data
    - **C**reate
    - **R**ead
    - **U**pdate
    - **D**elete

- A database can store information and relationships that are more complicated than storing data in a simple file.

Problems for storing data in a simple file: Redundancy

- Problems for storing data in a simple file
  - In a file, each row stands for one record for its own. As a results, the same information may be entered serval times.
    - For example, we use a file to track of the instructor of each course. This file may include the course instructor's name, ID, office.
    - If an instructor is currently teaching 4 courses, his/her information would appear in the file 4 times.
    - It costs more storage space than necessary.

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
COMP4405	Data Mining	Liu, Yang	2	RRS631

Problems with storing data in a simple file: Multiple Business Concepts

- Each row in a file may contain information on more than one business concept.

- For example, a file of courses information may include instructor information (Instructor Name, InstructionID, office) and course information (CourseID, CourseName) in the same row.

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
COMP4405	Data Mining	Liu, Yang	2	RRS631

File Modification Issues

- Redundancy and multiple business concepts create modification problems
  - **Deletion Problems (Deletion Anomalies)**

Suppose the course COMP4405 is canceled.
Deleted this row → Instructor's information is also lost

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
~~COMP4405~~	~~Data Mining~~	~~Liu, Yang~~	~~2~~	~~RRS631~~

File Modification Issues

- Redundancy and multiple business concepts create modification problems
  - **Deletion Problems (Deletion Anomalies)**
  - **Update Problems (Update Anomalies)**

If instructor Zhang, Lu is changed to LIU, YANG for course COMP4115, we need to also change Instructor ID, Office and Phone Number as well.

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
COMP4405	Data Mining	Liu, Yang	2	RRS631

File Modification Issues

- Redundancy and multiple concepts create modification problems
  - Deletion Problems (Deletion Anomalies)
  - Update Problems (Update Anomalies)
  - **Insertion Problems (Insertion Anomalies)**

Insert a row for information about a new instructor → course ID and CourseName information is missing.

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
COMP4405	Data Mining	Liu, Yang	2	RRS631
?	?	Michael	4	RRS902

Relational Databases

- Relational databases are designed to address many of the information complexity issues that arise in business.

- A relational database stores information in tables. Each concept is stored in its own **table**.

- In essence, a relational database will break up a file into several parts (tables).
  - One part for each concept in the file.
  - For example, a file of courses might be divided into a INSTRUCTOR table, a COURSE table and a COURSE_INSTRUCTOR table.

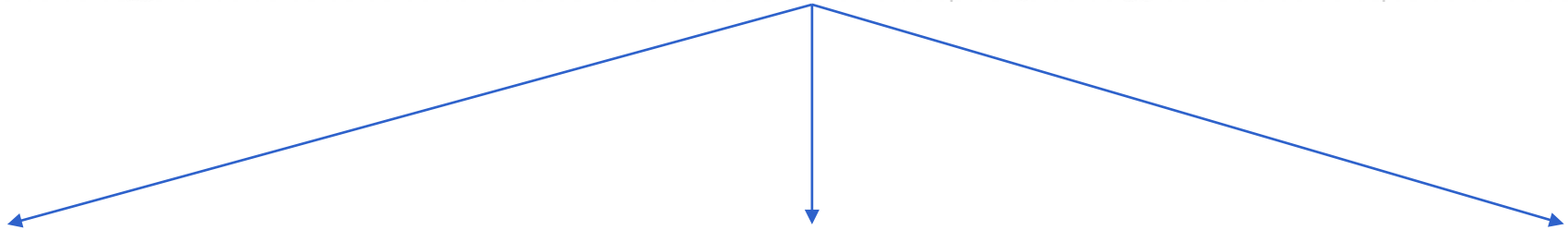
Relational Databases (con't)

- There is another type of data storage model used in applications like big data and machine learning.

- NoSQL databases store data in the same big table.

- NoSQL is useful when
  - Data structure is simple
  - The amount of data is extremely large (efficiency is important)
  - Some minor data inconsistency is acceptable

Course ID	Course Name	Instructor Name	Instructor ID	Office
COMP7990	Principles and Practices of Data Analytics	Zhang, Lu	3	RRS708
COMP1007	Introduction to Python and Its Applications	Zhang, Lu	3	RRS708
COMP7820	Visual Analytics and Decision Support	Zhang, Lu	3	RRS708
COMP4115	Exploratory Data Analysis and Visualization	Zhang, Lu	3	RRS708
COMP4405	Data Mining	Liu, Yang	2	RRS631



COURSE

Course ID	CourseName
COMP7990	Principles and Practices of Data Analytics
COMP4125	Visual Analytics
COMP7820	Visual Analytics and Decision Support
COMP4115	Exploratory Data Analysis and Visualization
COMP4405	Data Mining

COURSE_INSTRUCTOR

Course ID	Instructor ID
COMP7990	3
COMP4125	3
COMP7820	3
COMP4115	3
COMP4405	2

INSTRUCTOR

Instructor	Instructor ID	Office
Zhang, Lu	3	RRS708
LIU, Yang	2	RRS631

Relational Databases

- The information for courses and instructors can be joined back together.
- In a relational database, tables are joined together using matched pairs of data values.
  - For example, the **InstructorID** is stored in a column in **COURSE_INSTRUCTOR** table. Whenever we need information about an instructor, we can use InstructorID to look up the instructor information in the INSTRUCTOR Table.

COURSE

Course ID	CourseName
COMP7990	Principles and Practices of Data Analytics
COMP4125	Visual Analytics
COMP7820	Visual Analytics and Decision Support
COMP4115	Exploratory Data Analysis and Visualization
COMP4405	Data Mining

COURSE_INSTRUCTOR

Course ID	Instructor ID
COMP7990	3
COMP4125	3
COMP7820	3
COMP4115	3
COMP4405	2

INSTRUCTOR

Instructor	Instructor ID	Office
Zhang, Lu	3	RRS708
LIU, Yang	2	RRS631

Course ID	CourseName
COMP7990	Principles and Practices of Data Analytics
COMP4125	Visual Analytics
COMP7820	ual Analytics and De n Support
COMP4115	Exploratory ata Analysis and Visualizati
COMP4405	Data Mining

COURSE

Instructor	Instructor ID	Office	Phone Number
LAN, Liang	3	RRS708	34115880
LIU, Yang	2	RRS631	34112798

INSTRUCTOR

Course ID	Instructor ID
COMP7990	3
COMP4125	3
COMP7820	3
COMP4115	3
COMP4405	2

COURSE_INSTRUCTOR

Relational Databases

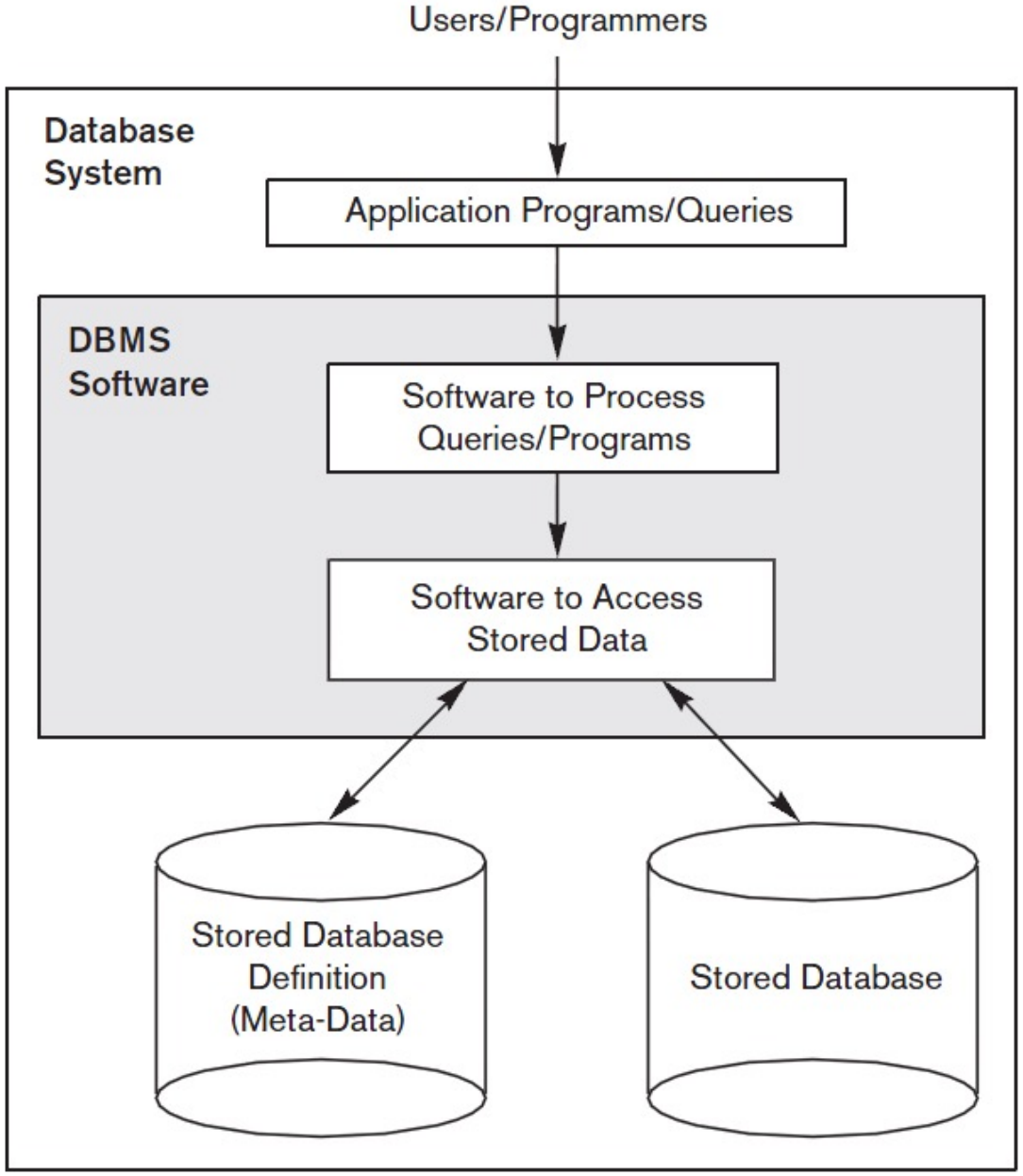
- You may think databases make things complicated.
  - Breaking file into tables
  - Obtain more complete information by joining tables (maybe some information is useless)
- **However**, a relational database minimizes data ***redundancy***, preserves ***complex relationships*** among concepts, and allows for ***partial data*** (null values).
- **Furthermore**, a relational database provides a solid foundation to create user interface forms and reports.
  - Structured Query Language (SQL)
    - An international standard language for **creating**, **processing** and **querying** databases and their tables.
    - A vast majority of data-driven applications and websites use SQL.

Database System

- The four components of a database system
  - Users
  - Database application(s)
  - Database Management System (DBMS)
  - Databases

Database System

- Users
- Database application(s)
  - User interacted with database applications (e.g. website)
- Database Management System (DBMS)
  - A gate-keeper for database. The database application needs go through DBMS to interact with the database
- Databases
  - Store raw data in separate tables



User

- A user of a database system will:
  - Use a **database application** to keep track of information (e.g, checking bank balance)
  - Use different user interface forms to ***enter***, ***read***, ***delete*** and ***query*** data
  - Produce reports

Database Applications

- A database application is a set of one or more computer programs or websites that serve as an intermediary between the ***user*** and the ***DBMS***.
  - Data-driven website
  - Mobile apps

- These application can not directly access data in database. They need to go through ***DBMS*** to touch the data.

Database Management System (DBMS)

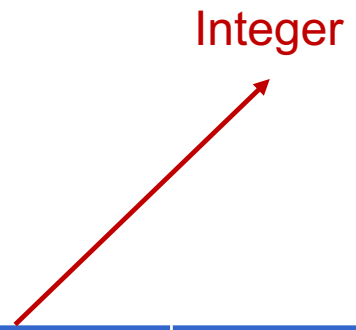
- A database management system (DBMS) serves as an intermediary (gatekeeper) between database applications and the database.
- The DBMS manages and controls database activities.
- The DBMS **creates**, **processes** and **administers** the databases it controls

The Database

- A database is a **self-describing** collection of **related records**

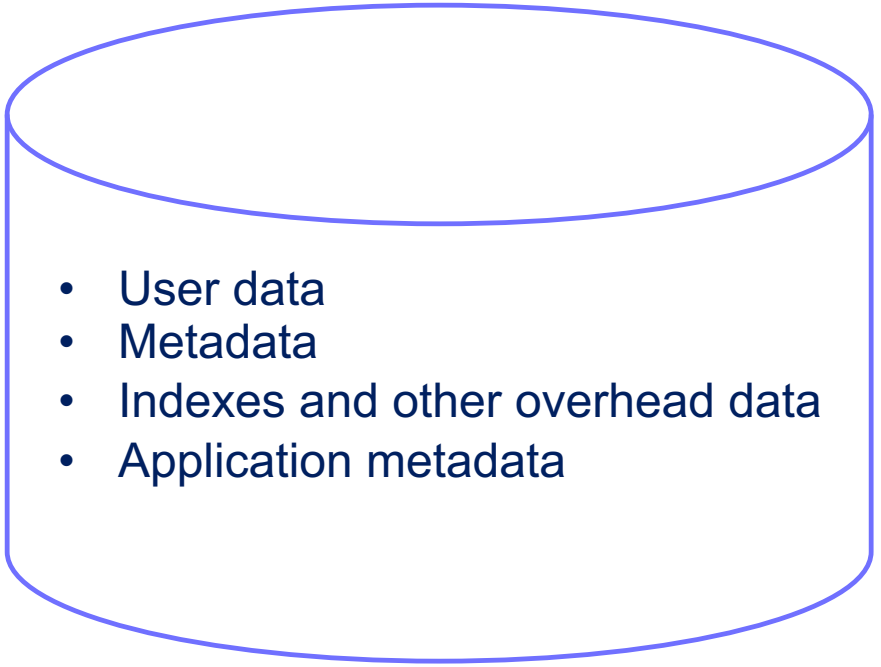
- Self-describing
  - The database itself contains the definition of its structure.
  - *Metadata* are data describing the structure of data in the database (e.g, the data type for Instructor ID is Integer)

- Tables with a relational database are related to each other in some way



Instructor	Instructor ID	Office	Phone Number
Zhang, Lu.	3	RRS708	34115880
LIU, Yang	2	RRS631	34112798

Database Contents



Functions of a DBMS

- Create databases
- Create tables
- Create relationships between tables
- Read database data
- Modify database data (insert, update, delete)
- Maintain database structures
- Enforce rules
- Control concurrency
- Provide security
- Perform data backup and recovery

Enforce Rules: Referential Integrity Constraints

- A DBMS can enforce many type of constraints.
- One of the most useful type is **Referential Integrity Constraints**.
- Referential integrity constraints ensure that the values of a column in one table are valid based on the values in another table.

– For example, if **4** was entered as an InstructorID in the COURSE_INSTRUCTOR table, a Instructor having a InstructorID value of 4 **MUST exist** in INSTRUCTOR table.

– If Instructor ID 4 does not exist, the DBMS would not allow us to add the record.

Course ID	Instructor ID
COMP7990	3
COMP4125	3
COMP7820	3
COMP4115	3
COMP4405	2
COMP4005	4

New record

Instructor	Instructor ID	Office
Zhang, Lu	3	RRS708
LIU, Yang	2	RRS631

Does InstructorID 4 exist in INSTRUCTOR table?

Outline

- Introduction to Databases
- **Relational Data Model**
- Introduction to SQL

Entity

- An entity is something of importance to a user of organization that needs to be represented in a database.
- Database entity is a thing, person, place, unit, object or any item about which the data should be captured and stored in the form of properties, workflow and tables.
- An entity represents one concept.
  - –e.g., course, instructor …

Instructor ID	Instructor	Office
3	Zhang, Lu.	RRS708
2	LIU, Yang	RRS631

Relation

- A relation is a two-dimensional table that has specific characteristics
  - Rows contain data about instances of an entity
  - Columns contain data about attributes of the entity
  - Cells of the table hold a single value
  - All values in a column are of the same data type
  - Each column has a unique name
  - The order of the columns is unimportant
  - The order of the rows is unimportant
  - No two rows can be identical

Each column is an attribute.

Each row represents an instructor →

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS708	34115880
2	LIU, Yang	RRS631	34112798

A Non-relation Example

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS708	34115880
2	LIU, Yang	RRS631	34112798
4	ZHANG, Wei	RRS707	34112001,51091111
3	Zhang, Lu	RRS708	34115880

A Non-relation Example

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS708	34115880
2	LIU, Yang	RRS631	34112798
4	ZHANG, Wei	RRS707	34112001,51091111
3	Zhang, Lu	RRS708	34115880

Identical rows

Multiple values in a cell

All relations are tables. However, not all tables are relations

Synonyms

Table	Row	Column
Relation	Record	Field
File (rarely)	Tuple (rarely)	Attribute

Keys

- A key is one (or more) columns of a relation whose values are used to identify a row

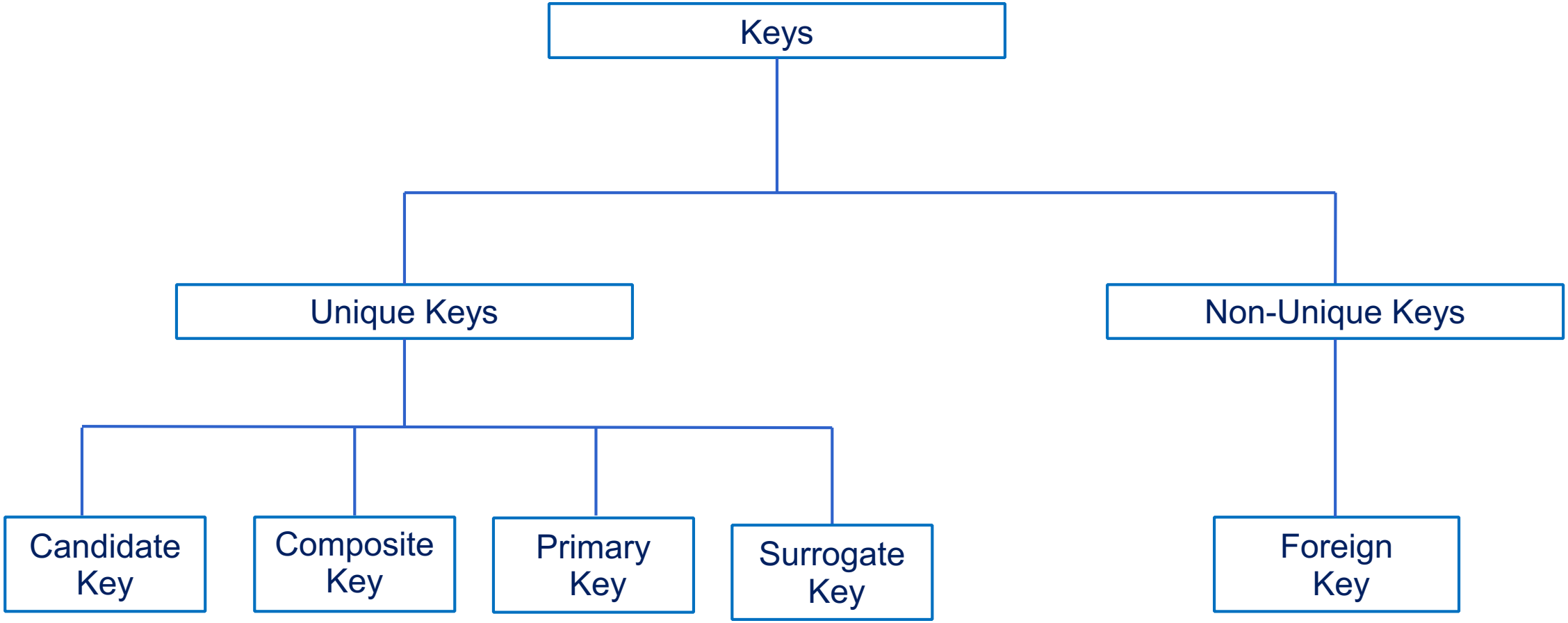
Unique Key	Non-unique Key
Data value is unique for each row. Therefore, the key will uniquely identify a row.	Data value may be shared among multiple rows. Therefore, the key will identify a set of rows.

Example

InstructorID	InstructorName	Office	Phone Number	Department ID
3	Zhang, Lu	RRS708	34115880	101
2	LIU, Yang	RRS631	34112798	101
9	WANG, Alan	AAB556	34110001	102

Unique Key

Non-unique Key


Keys
Unique Keys
Non-Unique Keys
Candidate Key
Composite Key
Primary Key
Surrogate Key
Foreign Key

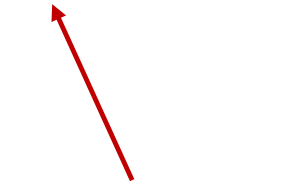
Keys

- Unique Keys
  - Candidate Key
    - A candidate key is called “candidate” because it has the potential to become the **primary key**.
  - Primary Key
    - A primary key is a candidate key chosen to be the main key for the relation.
  - Composite Key
    - A composite key is a key that is composed of two or more attributes for uniqueness.
    - e.g., Flight number + Date
  - Surrogate Key
    - A surrogate key is a unique, numeric value that is added to a relation to serve as the primary key.
    - Surrogate key values have no meaning to users.
    - e.g., Instructor ID
- Non-unique Keys
  - Foreign Key

Primary Key

- If you know the value of the primary key, you will be able to uniquely identify a single row within table.

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS709	34115880
2	LIU, Yang	RRS631	34112798
9	WANG, Alan	AAB556	34110001



Primary Key

Composite Key

- A composite key is a key that combines two or more attributes for uniqueness.

Flight Number	Date
KA852	20 Aug 2018
KA850	20 Aug 2018
KA852	21 Aug 2018
KA770	21 Aug 2018

Surrogate Key

- A surrogate key is a unique, numeric value that is intentionally added to a relation by database administrator for serving as the primary key.
- Surrogate key values have no meaning to users.
- Surrogate key are often used when there is no columns in a relation that can naturally served as a primary key.

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS708	34115880
2	LIU, Yang	RRS631	34112798
9	WANG, Alan	AAB556	34110001

Primary Key (Surrogate key)

Surrogate Key

- To avoid the use of composition key

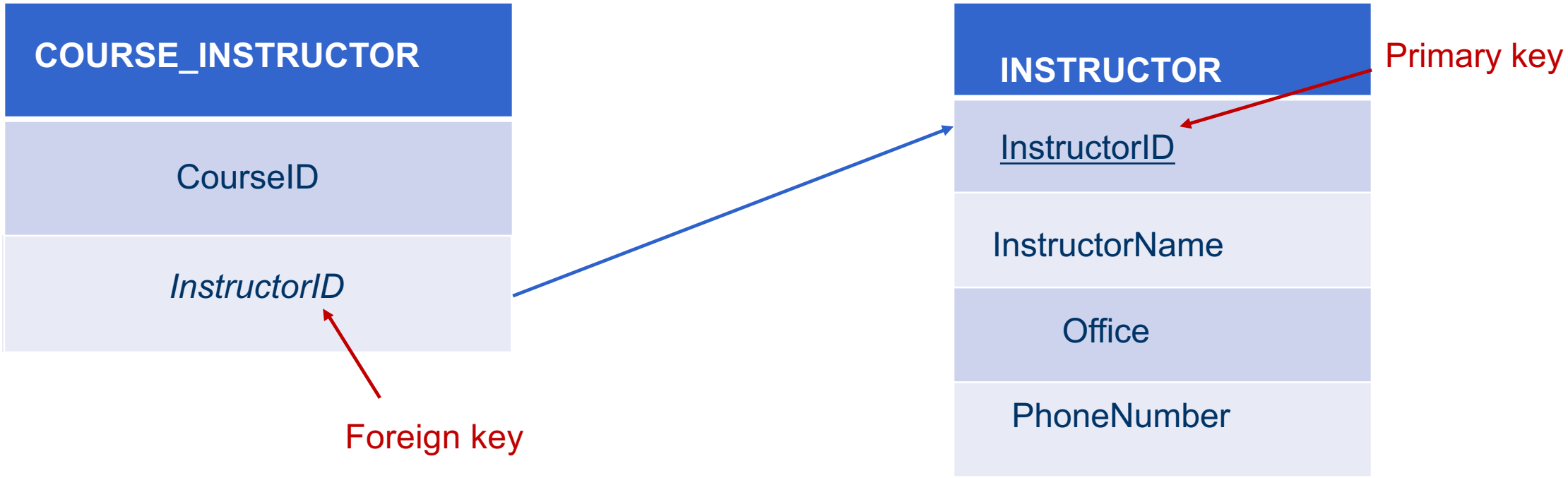
FlightID	Flight Number	Date
1	KA852	20 Aug 2018
2	KA850	20 Aug 2018
3	KA852	21 Aug 2018
4	KA770	21 Aug 2018

Relationship Between Tables

- A table may be related to other tables
- For example
  - An instructors teaches multiple courses
  - A manger controls multiple projects

A Foreign Key

- To establish relationships, we need to implement a foreign key.
- A **foreign key** is a primary key from one table that is placed into another table.
- The key is called a foreign key in the table that receives the key.



Referential Integrity

- Referential integrity states that every value of a foreign key must match a value of an existing primary key.

Foreign key

Course ID	*Instructor* ID
COMP7990	3
COMP4125	3
COMP7820	3
COMP4115	3
COMP4405	2
COMP4005	4

New record

Primary key

InstructorID	InstructorName	Office	Phone Number
3	Zhang, Lu	RRS709	34115880
2	LIU, Yang	RRS631	34112798
9	WANG, Alan	AAB556	34110001

Does InstructorID 4 exist in INSTRUCTOR table?

Null Values

- A null value means that no data exists
  - An empty cell in the table
- This is different from a zero, space character, empty string, or tab character
- The problem of null values
  - A null if often ambiguous. It could mean
    - The column value is unknown
    - The column value has not be decided yet
    - The column value is not applicable
  - Each may have entirely different implications

Outline

- Introduction to Databases
- Relational Data Model
- **Introduction to SQL**

Structured Query Language (SQL)

- Structured Query Language
  - Acronym: SQL
  - Pronounced “sequel” or “S-Q-L”
  - Original developed by IBM as the SEQUEL language in the 1970s
  - It is the standard query language in accessing a relational database

SQL

- SQL is not a programming language, but rather is a data sub-language
- SQL is comprised of
  - A Data Definition Language (DDL)
    - Used to define and manage database structures
  - A Data Manipulation Language (DML)
    - Data definition and updating
    - Data retrieval (Queries)
  - A Data Control Language (DCL)
    - For creating user accounts, managing permissions, etc

An Example of SQL

```
SELECT Name, Id FROM Employee;
```

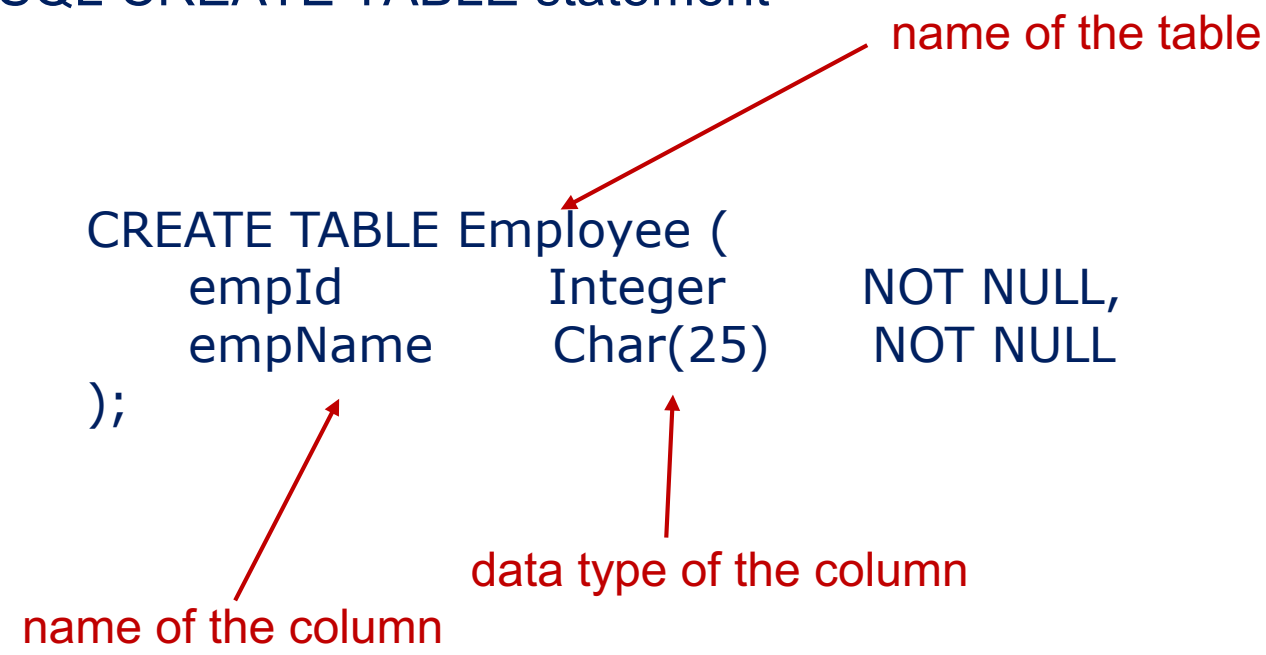
- The above statement is an example of SQL query.
- Type it in a platform (e.g. phpMyAdmin) allows you to retrieve the record.
- Depends on different platforms and different setup, table's name and column's name may be case sensitive or case insensitive. In our course, please write your query following the case in the question.
- The last semicolon ; is not required.

SQL for Data Definition

- The SQL data definition statements includes
  - CREATE
    - To create database objects
  - ALTER
    - To modify the structure and/or characteristics of existing database objects
  - DROP
    - To delete existing database objects

SQL for Data Definition: CREATE

- Creating database tables
  - The SQL CREATE TABLE statement



empId	empName

SQL for Data Definition: Common Data Type

Data Type	Description	Example Data	Usage
CHARACTER(n) / CHAR(n)	Character string, fixed at length n	"abc", "comp7990", "99"	studentEmail CHARACTER(25)
CHARACTER / CHAR	same as CHARACTER(1)	"x", "y"	gender CHARACTER
VARCHAR(n)	Variable length character string of max length n		
INTEGER	integer numerical.	7990, 12345	enrollment INTEGER
DECIMAL(p, s)	Exact numerical, maximum *p* number of digits and maximum *s* number of decimal place	1234.5678	price DECIMAL(10,2)
DATE	Date, the format is YYYY-MM-DD	1999-12-20	birthday DATE

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables with PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

empId	empName

```
CREATE TABLE Employee (
        empId           Integer         NOT NULL,
        empName         Char(25)        NOT NULL,
        CONSTRAINT      empPk           PRIMARY KEY(empId)
);
```

primary key in **bold and underscore**

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables with composite primary keys using PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE TABLE EmployeeSkill (
        empId           Integer         NOT NULL,
        skillId         Integer         NOT NULL,
        skillLevel      Integer         NULL,
        CONSTRAINT      empSkillPk      PRIMARY KEY(empId, skillId)
);
```

primary key in **bold and underscore**

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE TABLE EmployeeSkill (
        empId           Integer         NOT NULL,
        skillId         Integer         NOT NULL,
        skillLevel      Integer         NULL,
        CONSTRAINT      empSkillPk      PRIMARY KEY(empId, skillId),
        CONSTRAINT      empFk           FOREIGN KEY(empId) REFERENCES Employee(empId),
        CONSTRAINT      skillFk         FOREIGN KEY(skillId) REFERENCES Skill(skillId)
);
```

foreign key in *italic*
primary key in **bold and underscore**

Employee

empId	empName
1	Dan
2	Penny
3	Sheldon

EmployeeSkill

empId	***skillId***	skillLevel
1	101	5
1	102	3
2	101	4
2	103	5
		3
		5

Skill

skillId	skillName
101	C
102	Java
103	Python

```
CREATE TABLE Skill (
        skillId         Integer         NOT NULL,
        skillName       VarChar(25)     NOT NULL,
        CONSTRAINT      skillPk   PRIMARY KEY(skillId)
);
```

foreign key in *italic*
primary key in **bold and underscore**

Quick Question: What is the SQL statement to create Skill table?

SQL for Data Definition: CREATE with CONSTRAINT

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword
  - ON UPDATE CASCADE and ON DELETE CASCADE

```
CREATE TABLE EmployeeSkill (
        empId           Integer         NOT NULL,
        skillId         Integer         NOT NULL,
        skillLevel      Integer         NULL,
        CONSTRAINT      empSkillPk      PRIMARY KEY(empId, skillId),
        CONSTRAINT      empFk           FOREIGN KEY(empId)
                REFERENCES Employee(empId) ON DELETE CASCADE,
        CONSTRAINT      skillFk         FOREIGN KEY(skillId)
                REFERENCES Skill(skillId) ON UPDATE CASCADE
);
```

Cascade Delete

```
CONSTRAINT  empFk  FOREIGN KEY(empId)
    REFERENCES Employee(empId) ON DELETE CASCADE
```

Employee

empId	empName
1	Dan
2	Penny
3	Sheldon

EmployeeSkill

empId	***skillId***	SkillLevel
1	101	5
1	102	3
2	101	4
2	103	5
3	102	3
3	102	5

Skill

skillId	skillName
101	C
102	Java
103	Python

foreign key in *italic*
primary key in **bold and underscore**

When we delete a record (e.g. employee 1) from Employee table, “cascade delete” will automatically delete related records from other related tables.

Cascade Update

```
CONSTRAINT    skillFk          FOREIGN KEY(skillId)
    REFERENCES Skill(skillId) ON UPDATE CASCADE
```

Employee

empId	empName
1	Dan
2	Penny
3	Sheldon

EmployeeSkill

empId	***skillId***	SkillLevel
1	105	5
1	102	3
2	105	4
2	103	5
3	102	3
3	102	5

Skill

skillId	skillName
105	C
102	Java
103	Python

foreign key in *italic*
primary key in **bold and underscore**

When we update the skillId for ‘101’ to ‘105’ in skill table, “cascade update” will automatically update related records in other related tables.

Quick Question

```
CONSTRAINT    empFk          FOREIGN KEY(empId)
        REFERENCES Employee(empId) ON DELETE CASCADE,
CONSTRAINT    skillFk        FOREIGN KEY(skillId)
        REFERENCES Skill(skillId) ON UPDATE CASCADE
```

Employee

empId	empName
1	Dan
2	Penny
3	Sheldon

EmployeeSkill

empId	***skillId***	SkillLevel
1	101	5
1	102	3
2	101	4
2	103	5
3	102	3
3	102	5

Skill

skillId	skillName
101	C
102	Java
103	Python

foreign key in *italic*
primary key in **bold and underscore**

What happen if we delete that data? Any cascade update/delete?

Primary Key Constraint: ALTER

- Adding primary key constraints to an existing table
  - The SQL ALTER statement

```
ALTER TABLE Emplyee
        ADD CONSTRAINT empPk PRIMARY KEY(empId);
```

PRIMARY KEY →

Employee

empId	empName
1	Dan
2	Penny
3	Sheldon

Composite Primary Key Constraints: ALTER

- Adding a composite primary key constraint to an existing table
  - The SQL ALTER statement

```
ALTER TABLE EmplyeeSkill
        ADD CONSTRAINT empSkillPk
                PRIMARY KEY(empId, skillId);
```

PRIMARY KEY

EmployeeSkill

empId	skillId	SkillLevel
1	101	5
1	102	3
2	101	4
2	103	5
3	102	3
3	102	5

Foreign Key Constraint: ALTER

- Adding foreign key constraints to an exisiting table
  - The SQL ALTER statement

```
ALTER TABLE EmplyeeSkill
        ADD CONSTRAINT empFk FOREIGN KEY(empId)
                REFERENCES Employee(empId)
```

Modifying Data using SQL

- INSERT INTO
  - Add a new row to a table
- UPDATE
  - Update the rows in a table which match the specified criteria
- DELETE FROM
  - Delete the rows in a table which match the specified criteria

Adding Data: INSERT INTO

- To add a new row to an existing table
- Non-numeric data must be enclosed in single quotes (‘’)

```
INSERT INTO Employee (empId, salaryCode, lastName)
        VALUES (62, 11, 'Halpert');
```

Employee

empId	salaryCode	lastName
1	2	Brown
22	7	Smith

Employee

empId	salaryCode	lastName
1	2	Brown
22	7	Smith
62	11	Halpert

Quick Question

```
INSERT INTO Employee (empId, salaryCode, lastName)
VALUES (10, 3, "Kevin")
```

- How to insert the record for "Kevin" with salaryCode 3, empId 10
- Which of the following records can be insert successfully?
  - INSERT INTO Employee (empId, salaryCode, lastName) VALUES (3, 2, "Brown“)
  - INSERT INTO Employee (empId, salaryCode, lastName) VALUES (4, 5, “Steve“)
  - INSERT INTO Employee (empId, salaryCode, lastName) VALUES (22, 7, “Brady“)

Salary

salaryCode	salary
2	23000
3	26000
7	35000

Employee

empId	*salaryCode*	lastName
1	2	Brown
22	7	Smith
62	11	Halpert

Changing Data Values: UPDATE

- To change the data values in an existing row (or a set of rows) use the UPDATE statement
- The WHERE clause specifies the matching or filtering criteria for the records (rows) that are to be displayed

```
UPDATE      Employee
SET         phone = '34111001'
WHERE       empId = 29;
```

Update the phone number to '34111001' for the employee whose empID is 29

Employee

empId	phone
18	69981245
29	12345678

Employee

empId	phone
18	69981245
29	34111001

Changing Data Values: UPDATE

- To change the data values in an existing row (or a set of rows) use the UPDATE statement
- The WHERE clause specifies the matching or filtering criteria for the records (rows) that are to be displayed

```
UPDATE      Employee
SET         deptId = 4
WHERE       empName LIKE 'Da%';
```

Update the deptID to 4 for the employee names starts with "Da"

WHERE, LIKE are some keywords that will be covered more next week!

Employee

deptId	empName
18	Dave
29	John
29	Dan

Employee

deptId	empName
4	Dave
29	John
4	Dan

Quick Question

- What if we say

```
UPDATE      Employee
SET         deptId = 5
WHERE       empName Like '%k'
```

- And what if we say

```
UPDATE      Employee
SET         deptId = 5
```

Employee

deptId	empName
18	Dave
29	John
29	Dan

WHERE, LIKE are some keywords that will be covered more next week!

Be careful when updating records. If you omit the `WHERE` clause, ALL records will be updated!

Deleting Data: DELETE

- To delete a row (or a set of rows) from a table use the DELETE FROM statement

```
DELETE FROM   Employee
WHERE         empId = 29;

DELETE FROM   Employee
WHERE         empName LIKE 'Da%';

DELETE FROM Employee;
```

Employee

empId	phone
18	69981245
29	34110000

Employee

empId	phone
18	69981245

Note: Be careful when deleting records in a table! Notice the `WHERE` clause in the `DELETE` statement. The `WHERE` clause specifies which record(s) should be deleted. If you omit the `WHERE` clause, all records in the table will be deleted!

Next Week:

- SQL Query
- SELECT
- WHERE
- GROUP BY
- HAVING
- join tables