SQL: Basic Queries

Database Systems

Department of Computer Science, CBNU
Prof. Nasridinov Aziz (aziz@chungbuk.ac.kr)

Table of Contents

Last Lecture

- Relational Model
  - Relational model represents the database as a collection of relations
    - Relation = table

  - Basic Terminologies
    - Relation
    - Tuple
    - Attribute
    - Degree
    - Cardinality
    - Domain
    - Relation Schema
    - Relation Instance
    - Relational Database Schema
    - NULL value

Last Lecture

- Characteristics of relational model

- Keys
  - Super key, Candidate key, Primary key, Alternate key, Foreign key

- Integrity constraints = rule
  - A set of rules used to ensure the accuracy and consistency of the relational database

  - Types of integrity constraints
    - Key constraint
    - Domain constraint
    - Referential integrity constraint

1. SQL Overview

- Structural Query Language (SQL)
  - Define queries to access and manipulate the data from database
  - A procedural language, not a programming language
  - A standard language for RDBMS
    - Oracle, MySQL, Microsoft SQL and others
  - SQL statements are divided into four major categories
    - **Data Definition Language (DDL)**
    - **Data Manipulation Language (DML)**
    - Data Control Language (DCL)
    - Transaction Control Language (TCL)

1. SQL Overview

- SQL basic syntax
  - SQL keywords are NOT case sensitive
    - select is the same as SELECT
  - Comments
    - Single line comments: -- (Put the space after --) or #
    - Multi-line comments: text between /* and */
  - Use semicolon (;) to separate each SQL statement in a database
  - List of reserved SQL keywords
    - https://www.w3schools.com/sql/sql_ref_keywords.asp

2. Data Definition Language

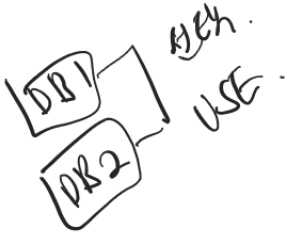
- Data Definition Language (DDL)
  - Notation for defining the database schema
    - Used to create and modify the structure of your relations and database objects
  - Statements
    - CREATE
    - DROP
    - ALTER
    - TRUNCATE

2. Data Definition Language

- Create database
  - Syntax
    - **CREATE DATABASE** *databasename;*
  - Example
    - **CREATE DATABASE** testDB;
  - You can check a list of databases using the following SQL statement
    - **SHOW DATABASES**;

2. Data Definition Language

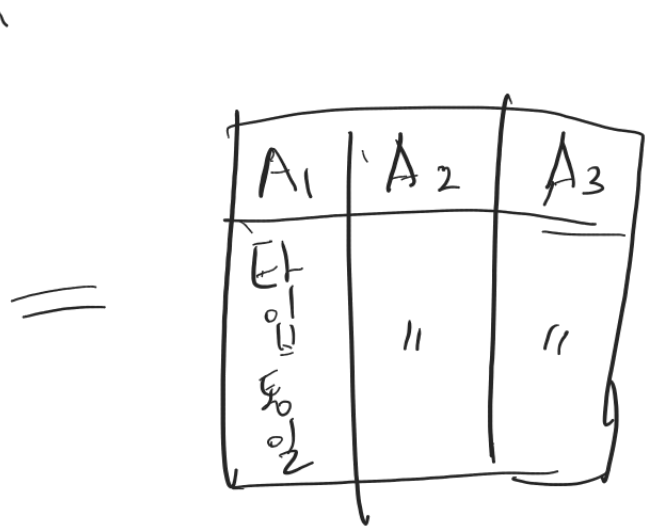
- Drop an existing database
  - Syntax
    - **DROP DATABASE** *databasename*;
  - Example
    - **DROP DATABASE** testDB;
- Set as default schema
  - Syntax
    - **USE** *databasename*;
  - Example
    - **USE** testDB;



2. Data Definition Language

- Create relation
  - Syntax

```
CREATE TABLE relation_name(
        attribute1      datatype,
        attribute2      datatype,
        attribute3      datatype,
        …               …
);
```



  - Full list of MySQL Data Types
    - https://www.w3schools.com/MySQL/mysql_datatypes.asp

2. Data Definition Language

- Create relation
  - Example

```
CREATE TABLE department(
        deptno          int,
        deptname        varchar(45),
        floor           int
);
```

PK 필수!!
= default + NN

  - The empty DEPARTMENT relation will now look like this:

2. Data Definition Language

- Drop an existing relation in a database

  테이블 삭제

  - Syntax
    - **DROP TABLE** *relation_name*;
  - Example
    - **DROP TABLE** department;

- Delete the data inside a relation, but not the relation itself

  데이터만 삭제

  테이블은 냉둥.

  - Syntax
    - **TRUNCATE TABLE** *relation_name*;
  - Example
    - **TRUNCATE TABLE** department;

2. Data Definition Language

- Alter relation – Add attribute
  - Syntax

    ALTER TABLE *relation_name*

    ADD *attribute_name datatype*;

  - Example

    ALTER TABLE department

    ADD phonenumber varchar(45);

2. Data Definition Language

- Alter relation – Drop attribute
  - Syntax

    ALTER TABLE *relation_name*

    DROP COLUMN *attribute_name*;

  - Example

    ALTER TABLE department

    DROP COLUMN phonenumber;

2. Data Definition Language

- Alter relation – Change attribute name
  - Syntax

    ALTER TABLE *relation_name*

    CHANGE *old_attribute_name new_attribute_name datatype*;

  - Example

    ALTER TABLE department

    CHANGE deptname departmentname varchar(45);

2. Data Definition Language

- Alter relation – Modify attribute
  - Syntax

    ALTER TABLE *relation_name*

    MODIFY COLUMN *attribute_name datatype*;

  - Example

    ALTER TABLE department

    MODIFY COLUMN floor varchar(45);

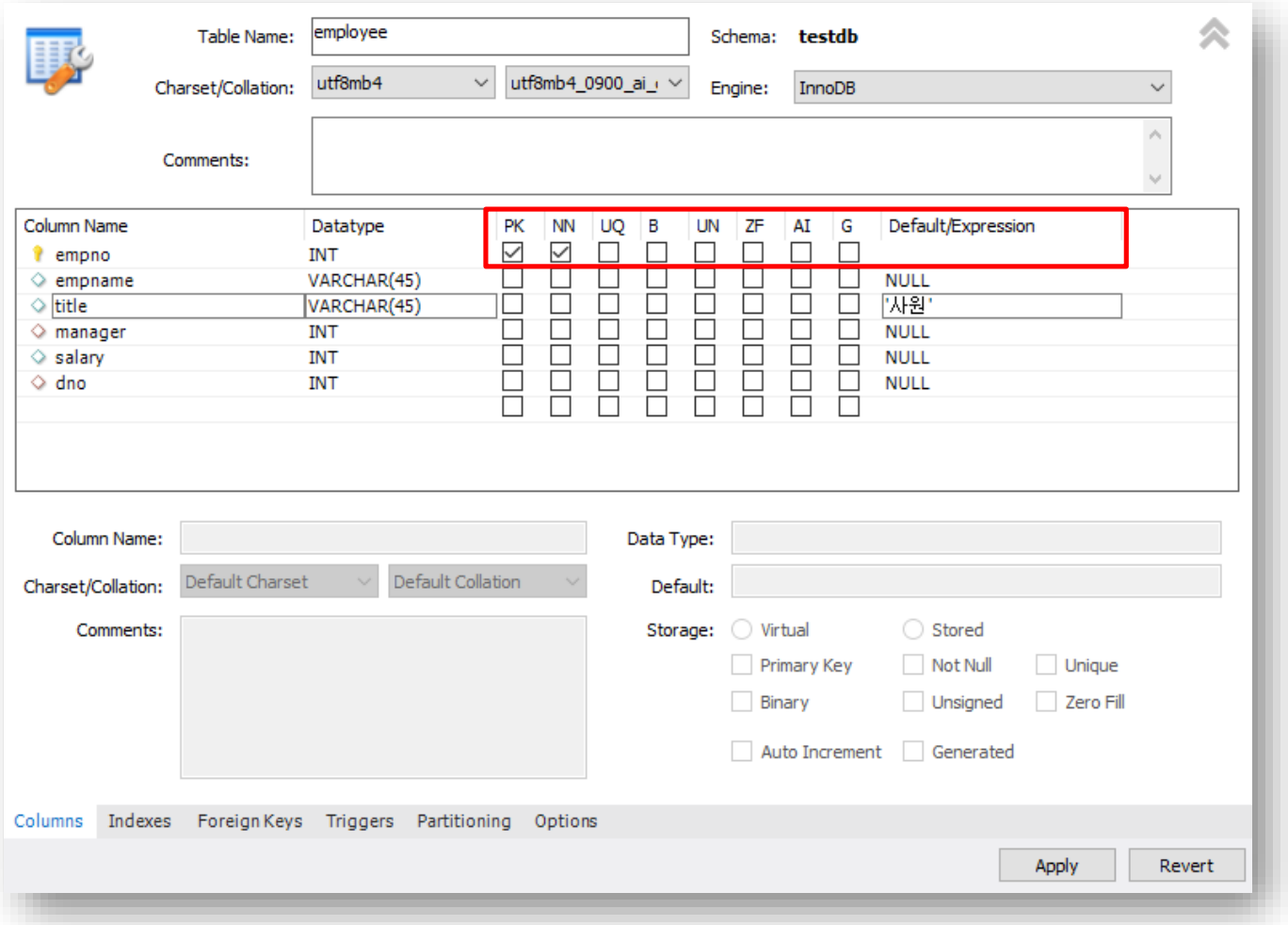
2. Data Definition Language

- MySQL constraints
  - SQL constraints are used to specify rules for the data in a relation

  - MySQL constraints
    - **NOT NULL** → 모든 데이블이 존재해야 함. PK를 유지
    - **PRIMARY KEY** → 모든 데이블이 존재해야 함.
    - **FOREIGN KEY**
    - **UNIQUE**
    - **CHECK**
    - **DEFAULT**
    - **CREATE INDEX**

2. Data Definition Language

❖ MySQL Constraints in MySQL Workbench



2. Data Definition Language

- MySQL constraints
  - Constraints can be specified when the relation is created with the CREATE TABLE statement
  - You can also specify constraints after the relation is created with the ALTER TABLE statement
  - Syntax

```
CREATE TABLE relation_name(
        attribute1      datatype        constraint,
        attribute2      datatype        constraint,
        attribute3      datatype        constraint,
        …               …
);
```

2. Data Definition Language

- NOT NULL constraint
  - By default, an attribute can hold NULL values
    - The NOT NULL constraint enforces an attribute to NOT accept NULL values

  - Example

```
CREATE TABLE department(
        deptno          int             NOT NULL,
        deptname        varchar(45)     NOT NULL,
        floor           int
);
```

  - Example

```
ALTER TABLE department
MODIFY floor int NOT NULL;
```

2. Data Definition Language

- PRIMARY KEY constraint
  - The PRIMARY KEY constraint uniquely identifies each tuple in a relation
  - Primary keys must contain unique values, and cannot contain NULL values
  - Example

```
CREATE TABLE department(
        deptno          int             NOT NULL,
        deptname        varchar(45)     NOT NULL,
        floor           int,
        CONSTRAINT PK_Department PRIMARY KEY (deptno)
);
```

2. Data Definition Language

- PRIMARY KEY constraint
  - You can also drop PRIMARY KEY constraint

    ALTER TABLE department

    DROP PRIMARY KEY;

  - To create a PRIMARY KEY constraint on the "deptno" attribute when the relation is already created

    ALTER TABLE department

    ADD CONSTRAINT PK_Department **PRIMARY KEY** (deptno);

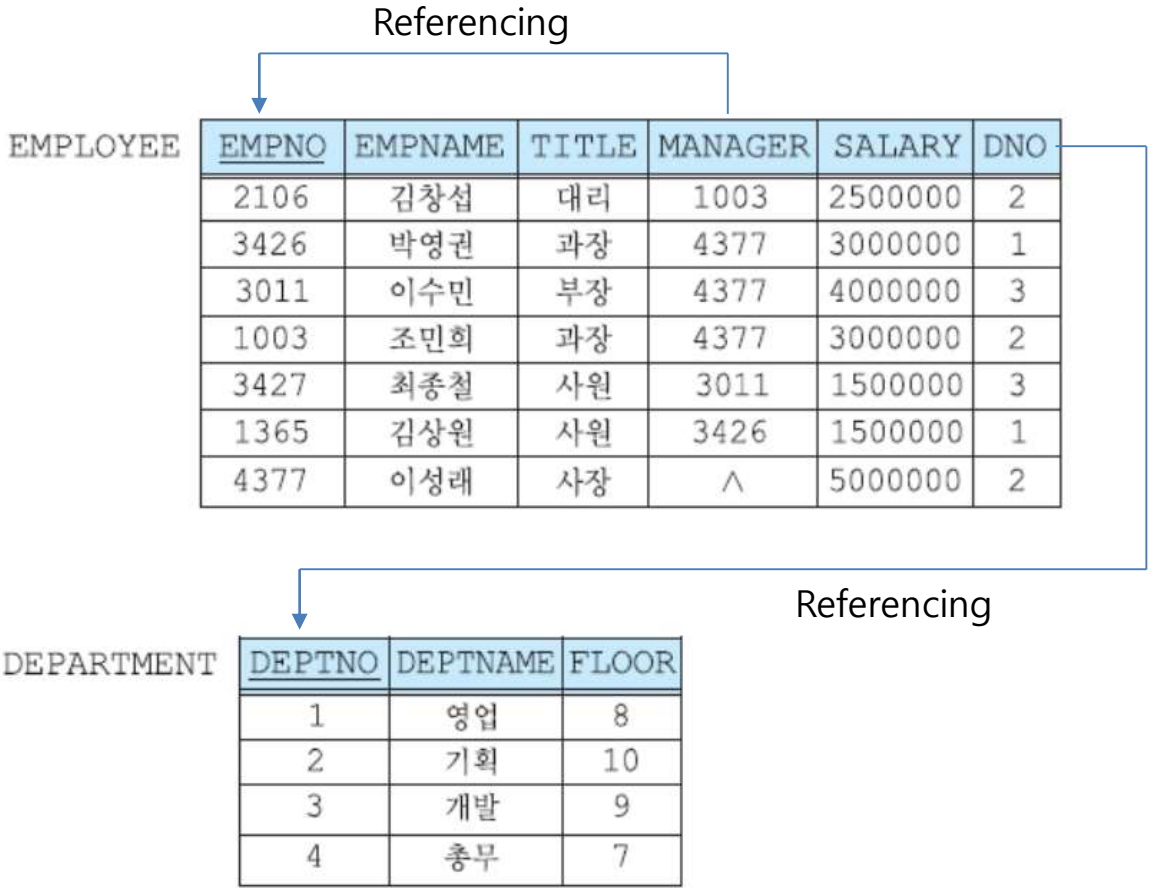
2. Data Definition Language

- FOREIGN KEY constraint
  - Used to create a relationship between two relations
  - Types of foreign key
    - Foreign key referencing the primary key of another relation
    - Foreign key referencing the primary key of its own relation
    - Foreign key that is a composite of the primary keys
  - The relation with the foreign key is called the **referencing relation** (or child relation), and the relation with the primary key is called the **referenced relation** (or parent relation)

2. Data Definition Language

❖ FOREIGN KEY constraint

- Referencing relation: EMPLOYEE; Referenced relation : DEPARTMENT



2. Data Definition Language

- FOREIGN KEY constraint
  - Syntax

    CONSTRAINT *FK_name*

    FOREIGN KEY (*FK_attribute*) **REFERENCES** *parent_relation(PK_attribute)*

  - Example

```
CREATE TABLE employee(
        empno           int             NOT NULL,
        empname         varchar(45),
        title           varchar(45),
        manager         int,
        salary          int,
        dno             int,
        CONSTRAINT PK_Employee PRIMARY KEY (empno),
        CONSTRAINT FK_Employee_Manager
        FOREIGN KEY (manager) REFERENCES employee(empno),
        CONSTRAINT FK_Department_Employee
        FOREIGN KEY (dno) REFERENCES department(deptno)
);
```

FOREIGN KEY name

2. Data Definition Language

- FOREIGN KEY constraint
  - To drop FOREIGN KEY constraint

    ALTER TABLE employee

    DROP FOREIGN KEY FK_Department_Employee;

  - To add FOREIGN KEY constraint

    ALTER TABLE employee

    ADD CONSTRAINT FK_Department_Employee

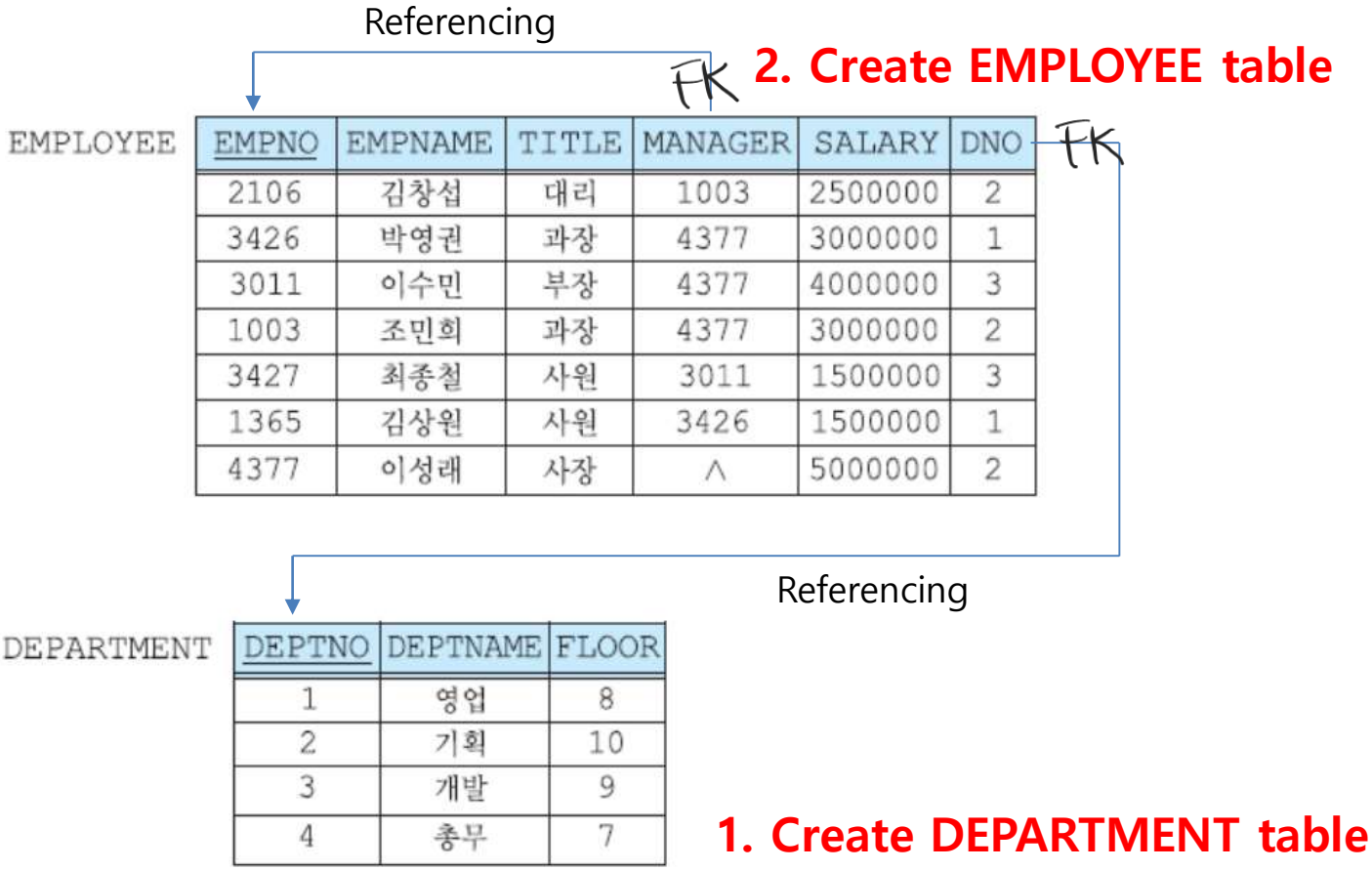
    FOREIGN KEY (dno) **REFERENCES** department(deptno);

FOREIGN KEY
= 한 테이블에서 다른 테이블을 참조하는 키.

2. Data Definition Language

❖ Summary

- Referencing relation: EMPLOYEE; Referenced relation : DEPARTMENT



2. Data Definition Language

- Summary
  - First create DEPARTMENT table with PRIMARY KEY constraint

  - DEPARTMENT Table

```
CREATE TABLE department(
        deptno          int             NOT NULL,
        deptname        varchar(45)     NOT NULL,
        floor           int,
        CONSTRAINT PK_Department PRIMARY KEY (deptno)
);
```

2. Data Definition Language

- Summary
  - Then create EMPLOYEE table with FOREING KEY constraint

```
CREATE TABLE employee(
        empno           int             NOT NULL,
        empname         varchar(45),
        title           varchar(45)
        manager         int,
        salary          int,
        dno             int,
        CONSTRAINT PK_Employee PRIMARY KEY (empno),
        CONSTRAINT FK_Employee_Manager
        FOREIGN KEY (manager) REFERENCES employee(empno),
        CONSTRAINT FK_Department_Employee
        FOREIGN KEY (dno) REFERENCES department(deptno)
);
```

3. Data Manipulation Language

- Data Manipulation Language (DML)
  - A language for searching, modifying, inserting and deleting the desired data in the database
    - DML also known as query language
  - Statements
    - INSERT
    - UPDATE
    - DELETE
    - SELECT

3. Data Manipulation Language

- The INSERT INTO statement
  - Used to insert tuples into the relation
  - Specify both the attributes names and the values to be inserted

    INSERT INTO *relation_name* (*attribute1*, *attribute2*, *attribute3*, ...)

    VALUES (*value1*, *value2*, *value3*, ...);
  - If you are adding values for all the attributes of the relation, you do not need to specify the attribute names in the SQL query

    INSERT INTO *relation_name*

    VALUES (*value1, value2, value3, ...*);

3. Data Manipulation Language

- The INSERT INTO statement
  - Example of inserting a single tuple

```
INSERT INTO department (deptno, deptname, floor)
VALUES (1, '영업', 8);
```

  - Example of inserting multiple tuples

```
INSERT INTO department
        (deptno, deptname, floor)
VALUES
        (2, '기획', 10),
        (3, '개발', 9),
        (4, '충무', 7);
```

3. Data Manipulation Language

- The INSERT INTO statement
  - The INSERT INTO statement without attribute names

```
INSERT INTO employee
VALUES
        (4377, '이성래', '사장', NULL, 5000000, 2),
        (3011, '이수민', '부장', 4377, 4000000, 3),
        (3426, '박영권', '과장', 4377, 3000000, 1),
        (1003, '조민희', '과장', 4377, 3000000, 2),
        (2106, '김창섭', '대리', 1003, 2500000, 2),
        (3427, '최종철', '사원', 3011, 1500000, 3),
        (1365, '김상원', '사원', 3426, 1500000, 1);
```

3. Data Manipulation Language

- The UPDATE statement
  - Used to modify the existing tuples in a relation

    UPDATE *relation_name*

    SET *attribute1* = *value1*, *attribute2* = *value2*, ...

    WHERE *condition*;

    optional

    PK의 인스턴스화 (ex: empno = ?)

  - Example: Suppose that a person with an employee number 1365 received the promotion from '사원' to '대리' and his salary is increased from 1500000 to 2500000. Use UPDATE statement to make this change:

    UPDATE employee

    SET title = '대리', salary = 2500000

    WHERE empno = 1365;

3. Data Manipulation Language

- The DELETE statement
  - Used to delete existing tuples in a relation

    DELETE FROM *relation_name* **WHERE** *condition*;

  - Example: Suppose that a person with an employee number 2106 quit the job. Use DELETE statement to remove the tuple related to 2106

    DELETE FROM employee **WHERE** empno = 2106;

3. Data Manipulation Language

- The SELECT statement
  - Used to select/retrieve tuples from a database
  - Syntax

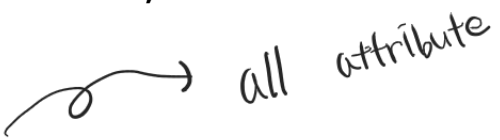
SELECT	[**DISTINCT**] attribute(s)	(1)	Compulsory
FROM	relation(s)	(2)	
[**WHERE**	condition]	(3)	Optional
[**GROUP BY**	attribute(s)]	(4)	
[**HAVING**	condition]	(5)	
[**ORDER BY**	attribute(s)]	(6)	
[**LIMIT**	number];	(7)	

3. Data Manipulation Language

- The SELECT statement
  - Example: Select all attributes of department relation

SELECT deptno, deptname, floor

FROM department;



SELECT *

FROM department;

DEPTNO	DEPTNAME	FLOOR
1	영업	8
2	기획	10
3	개발	9
4	총무	7

3. Data Manipulation Language

- The INSERT INTO SELECT statement
  - Copies data from one table and inserts it into another table
  - Example
    - **INSERT INTO** Customers (CustomerName, City, Country)
    - **SELECT** SupplierName, City, Country FROM Suppliers;
- The SELECT INTO statement
  - Copies data from one table and inserts it into a new table
  - Example
    - **SELECT** * **INTO** CustomersBackup2017
    - **FROM** Customers;

3. Data Manipulation Language

- The SELECT statement
  - Example: Search the titles of all employees

    SELECT title

    FROM employee;

TITLE
대리
과장
부장
과장
사원
사원
사장

3. Data Manipulation Language

- The SELECT statement
  - Example: Search only distinct titles of all employees

```
SELECT DISTINCT title
FROM employee;
```

remove duplicates.

TITLE
대리
과장
부장
사원
사장

3. Data Manipulation Language

- The SELECT statement
  - Example: Retrieve all information about employees in department 2

```
SELECT *
FROM employee
WHERE DNO = 2;
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1003	조민희	과장	4377	3000000	2
2016	김창섭	대리	1003	2500000	2
4377	이성래	사장	NULL	5000000	2

3. Data Manipulation Language

- The SELECT statement – AND operator
  - The AND operator displays a tuple if all the conditions separated by AND are TRUE

  - Example: Search for the names and salaries of employees with a manager position and working in department 1

```
SELECT empname, salary
FROM employee
WHERE title = '과장' AND dno = 1;
```

EMPNAME	SALARY
박영권	3000000

3. Data Manipulation Language

❖ The SELECT statement – AND operator

- Example: Search for the names and salaries of employees whose job title is a manager and do not belong to department 1

```
SELECT empname, salary
FROM employee
WHERE title = '과장' AND dno <> 1;
```

!=

EMPNAME	SALARY
조민희	3000000

- Note that in some versions of SQL, <> operator may be written as !=

3. Data Manipulation Language

- The SELECT statement – AND operator
  - Example: Search for the names, positions, and salaries of employees with a salary of 3,000,000 won or more and less than 4,500,000 won

```
SELECT empname, title, salary
FROM employee
WHERE salary BETWEEN 3000000 AND 4500000;

SELECT empname, title, salary
FROM employee
WHERE salary >= 3000000 AND salary <= 4500000;
```

EMPNAME	TITLE	SALARY
박영권	과장	3000000
이수민	부장	4000000
조민희	과장	3000000

3. Data Manipulation Language

- The SELECT statement – OR operator
  - The OR operator displays a tuple if any of the conditions separated by OR is TRUE

  - Example: Search for employees who have title '대리' or '사원'

```
SELECT *
FROM employee
WHERE title = '대리' OR title = '사원';
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
2106	김창섭	대리	1003	2500000	2
3427	최종철	사원	3011	1500000	3
1365	김상원	사원	3426	1500000	1

3. Data Manipulation Language

- The SELECT statement – NOT operator
  - The NOT operator displays a tuple if the condition(s) is NOT TRUE

  - Example: Search all information related to employees except the owner

```
SELECT *
FROM employee
WHERE NOT title = '사장';
```

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
2106	김창섭	대리	1003	2500000	2
3426	박영권	과장	4377	3000000	1
3011	이수민	부장	4377	4000000	3
1003	조민희	과장	4377	3000000	2
3427	최종철	사원	3011	1500000	3
1365	김상원	사원	3426	1500000	1

3. Data Manipulation Language

❖ The SELECT statement – LIKE operator ← Exam Alert

- Example: Search for the names, titles, and department numbers of employees with the Lee surname

```
SELECT empname, title, dno
FROM employee
WHERE empname LIKE '이%';
```

EMPNAME	TITLE	DNO
이수민	부장	3
이성래	사장	2

3. Data Manipulation Language

- The SELECT statement – LIKE operator
  - Other cases of LIKE operator with ‘%’ and ‘_’ keywords

LIKE Operator	Description
WHERE CustomerName LIKE 'a%'	Finds any values that start with "a"
WHERE CustomerName LIKE '%a'	Finds any values that end with "a"
WHERE CustomerName LIKE '%or%'	Finds any values that have "or" in any position
WHERE CustomerName LIKE '_r%'	Finds any values that have "r" in the second position
WHERE CustomerName LIKE 'a_%'	Finds any values that start with "a" and are at least 2 characters in length
WHERE CustomerName LIKE 'a__%'	Finds any values that start with "a" and are at least 3 characters in length
WHERE ContactName LIKE 'a%o'	Finds any values that start with "a" and ends with "o"

3. Data Manipulation Language

❖ The SELECT statement – IN operator

- Example: Retrieve all information about employees belonging to department 1 or department 3

```
SELECT *
FROM employee
WHERE dno IN (1, 3);
```

→ 이거 'or' 니까 헷갈리지 말 것!!

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
1365	김상원	사원	3426	1500000	1
3011	이수민	부장	4377	4000000	3
3426	박영권	과장	4377	3000000	1
3427	최종철	사원	3011	1500000	3

3. Data Manipulation Language

❖ The SELECT statement – Arithmetic Operators

- Example: Retrieve the names, current salaries, and salaries increased by 10% for employees with a manager position

```
SELECT empname, salary, salary * 1.1 AS newsalary
FROM employee
WHERE title = '과장';
```

별칭

EMPNAME	SALARY	NEWSALARY
박영권	3000000	3300000
조민희	3000000	3300000

3. Data Manipulation Language

- The SELECT statement – Aggregation Functions
  - Applied to one attribute of a relation and returns a single value
  - **Can only appear in SELECT and HAVING clauses**

  - Aggregation functions
    - COUNT – counts the number of tuples or values
    - SUM – sums the values
    - AVG – returns the average of values
    - MAX – returns maximum value
    - MIN – returns min value

3. Data Manipulation Language

one tuple

- The SELECT statement – Aggregation Functions
  - Example: Show the maximum and average salary of all employees

    SELECT AVG(salary) **AS** avgsalary, **MAX**(salary) **AS** maxsalary

    FROM employee;

AVGSAL	MAXSAL
2928571	5000000

  - Example: Count all tuples in employee relation

    SELECT COUNT(*) **AS** allTuples

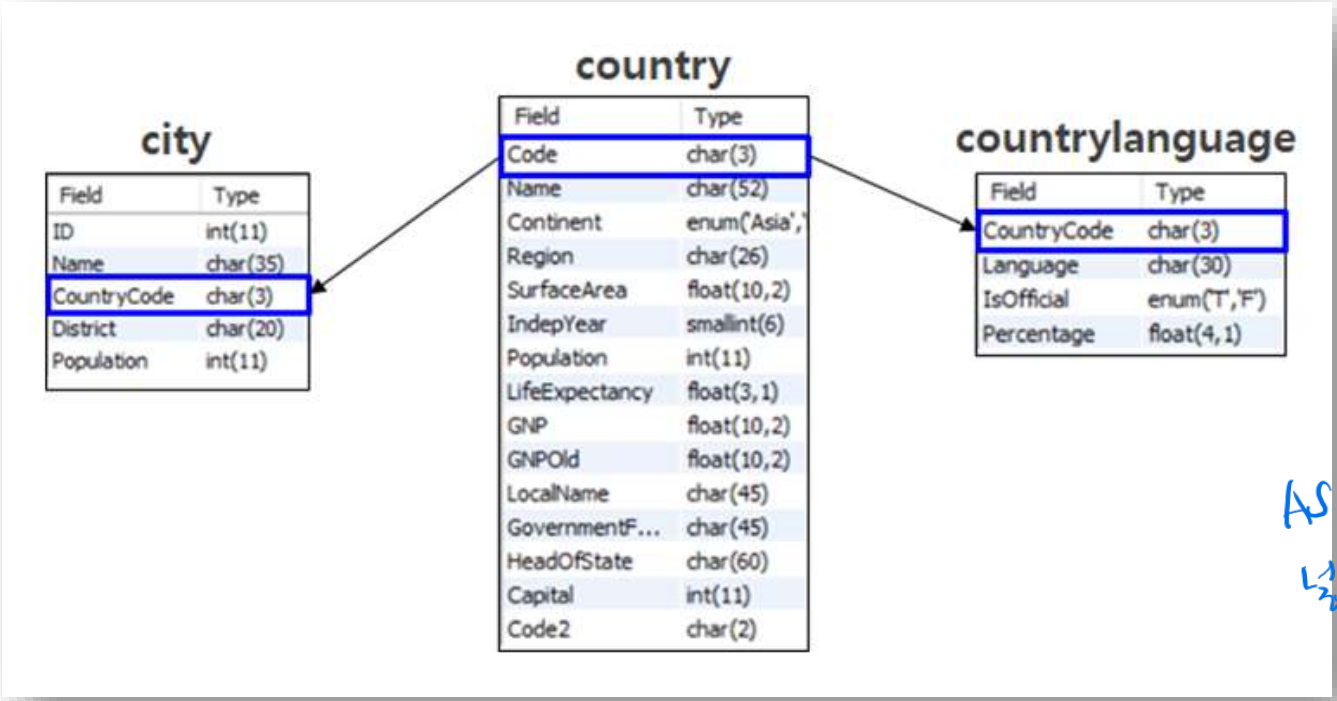
    FROM employee;

allTuples
7

3. Data Manipulation Language

❖ Quiz 1

- Given the following WORLD database



- Find how many languages are spoken in the USA

AS ~
나중에 다 공부.

SELECT count(*)
FROM countrylanguage
WHERE CountryCode = 'USA';

3. Data Manipulation Language

- More about data

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	1.9
AFG	Uzbek	F	8.8

3. Data Manipulation Language

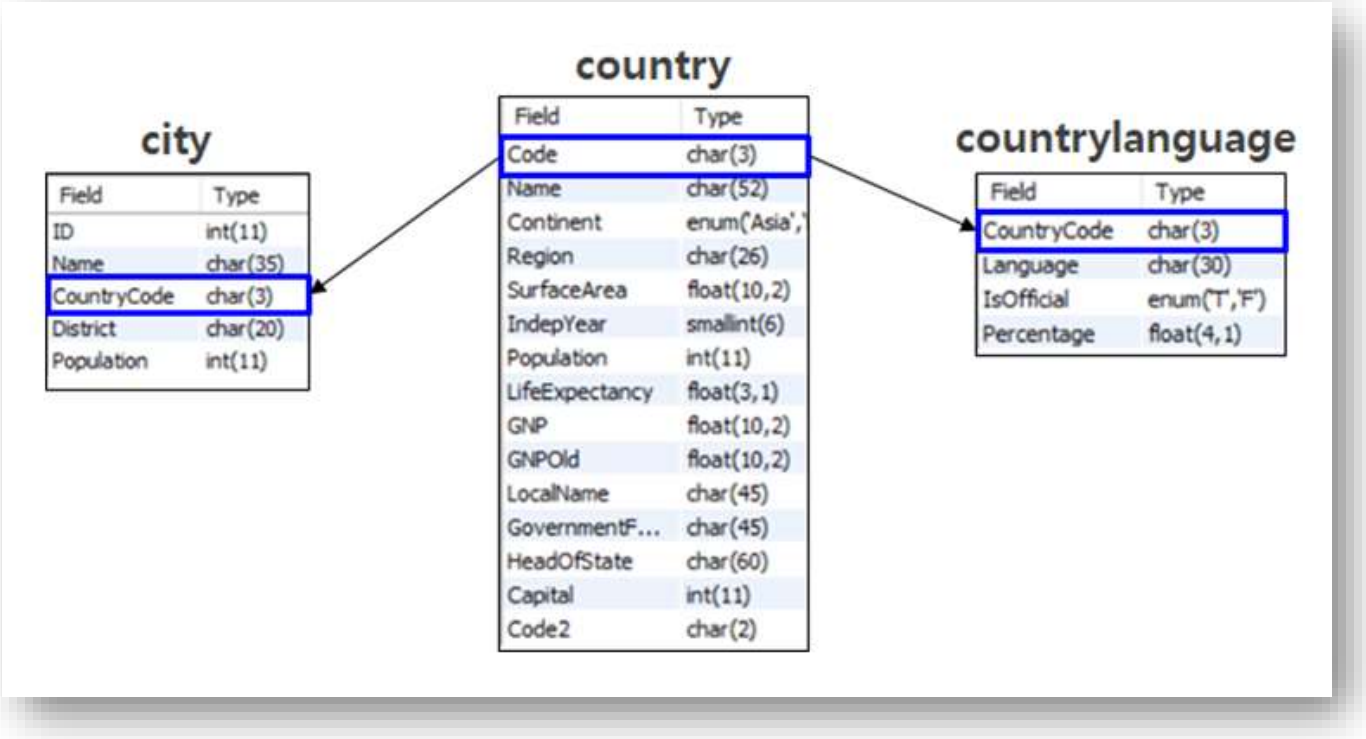
- Answer to Query 1

```
SELECT COUNT(*) AS LanguagesInUSA
FROM CountryLanguage
WHERE CountryCode = 'USA';
```

	LanguagesInUSA
▶	12

3. Data Manipulation Language

- Quiz 2
  - Given the following WORLD database



  - Find total amount of languages in the world

3. Data Manipulation Language

- Answer to Query 2: Count how many languages are in the world

```
SELECT COUNT(Language)
FROM CountryLanguage ;
```

	COUNT(Language)
▶	984

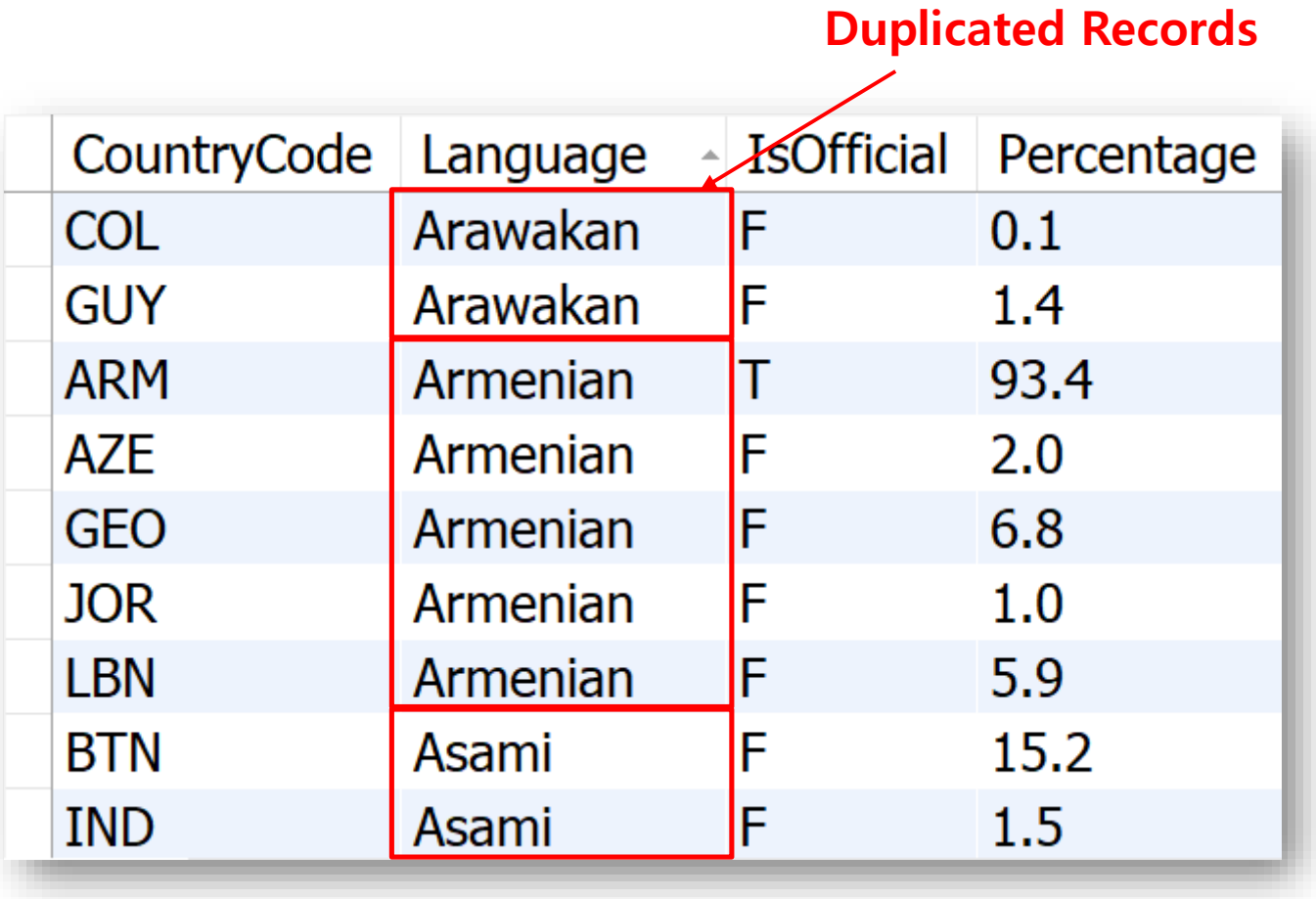
- Answer to Query 2: Count how many languages are in the world

```
SELECT COUNT( DISTINCT Language)
FROM CountryLanguage;
```

	COUNT(DISTINCT Language)
▶	457

3. Data Manipulation Language

- Query 2 Explanation
  - There are many duplicated records in Language attribute



CountryCode	Language	IsOfficial	Percentage
COL	Arawakan	F	0.1
GUY	Arawakan	F	1.4
ARM	Armenian	T	93.4
AZE	Armenian	F	2.0
GEO	Armenian	F	6.8
JOR	Armenian	F	1.0
LBN	Armenian	F	5.9
BTN	Asami	F	15.2
IND	Asami	F	1.5

3. Data Manipulation Language

- The SELECT statement
  - Used to select/retrieve tuples from a database

  - Syntax

SELECT	[**DISTINCT**] attribute(s)	(1)	Compulsory
FROM	relation(s)	(2)	
[**WHERE**	condition]	(3)	Optional
[**GROUP BY**	attribute(s)]	(4)	
[**HAVING**	condition]	(5)	
[**ORDER BY**	attribute(s)]	(6)	
[**LIMIT**	number];	(7)	

3. Data Manipulation Language

- The GROUP BY statement
  - Used to group tuples with the same value in the attribute
    - **In the result, one tuple is created for each group**
  - We often use GROUP BY with aggregation functions (COUNT, SUM, AVG, MAX, MIN)
  - Example: Group employees by department number, and show the department number, average salary, and maximum salary in each department

```
SELECT      dno, AVG(salary) as avgsal, MAX(salary) as maxsal
FROM        employee
GROUP BY    dno;
```

3. Data Manipulation Language

❖ The GROUP BY statement

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
3426	박영권	과장	4377	3000000	1
1365	김상원	사원	3426	1500000	1
2106	김창섭	대리	1003	2500000	2
1003	조민희	과장	4377	3000000	2
4377	이성래	사장	∧	5000000	2
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3

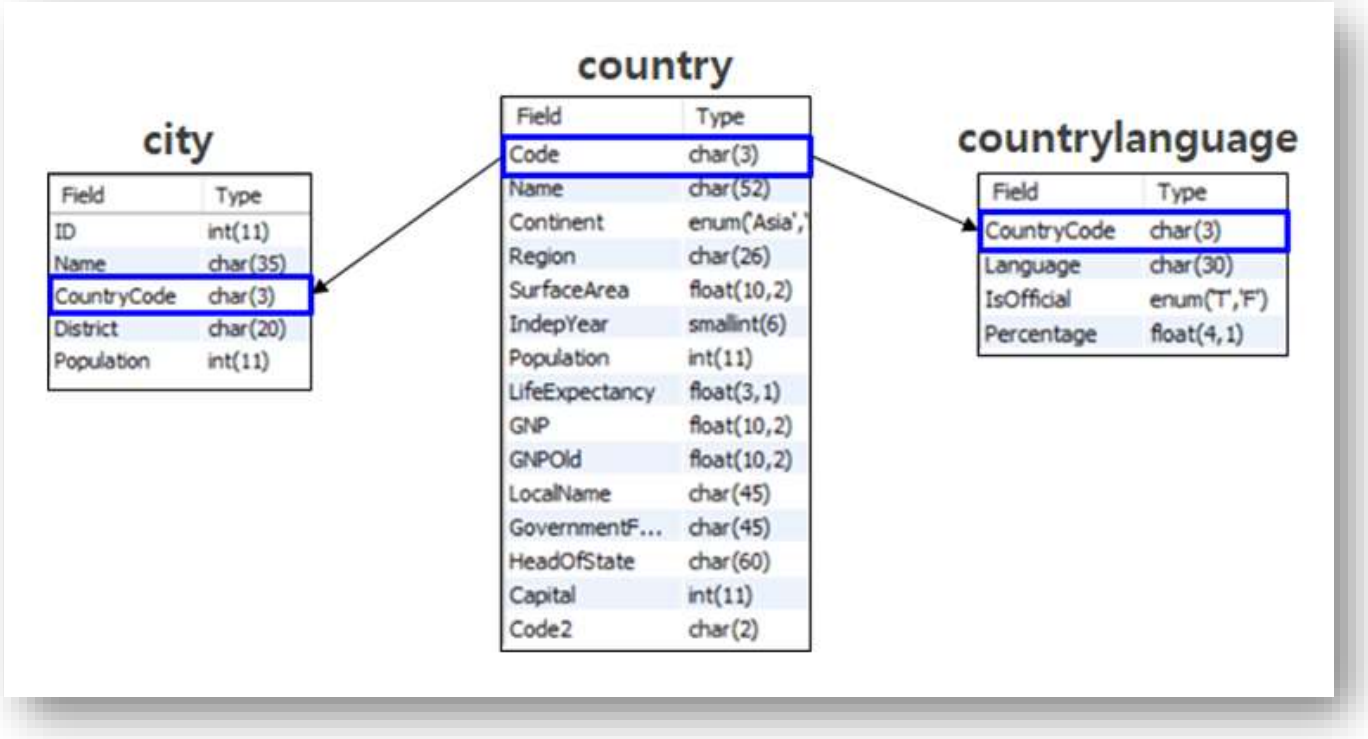
Grouped

DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

3 tuples

3. Data Manipulation Language

- Quiz 3
  - Given the following WORLD database



  - How many countries speak each language

3. Data Manipulation Language

- Answer to Query 3: How many countries speak each language
  - **SELECT** Language, **COUNT**(CountryCode)
  - **FROM** CountryLanguage
  - **GROUP BY** Language

Language	COUNT(CountryCode)
Dutch	5
English	60
Papiamento	2
Spanish	28
Balochi	4
Dari	1
Pashto	2
Turkmenian	3
Uzbek	6

3. Data Manipulation Language

- Quiz 3 Explanation
  - Group by Language attribute: Arawakan, Armenian, Asami, etc

CountryCode	Language	IsOfficial	Percentage
COL	Arawakan	F	0.1
GUY	Arawakan	F	1.4
ARM	Armenian	T	93.4
AZE	Armenian	F	2.0
GEO	Armenian	F	6.8
JOR	Armenian	F	1.0
LBN	Armenian	F	5.9
BTN	Asami	F	15.2
IND	Asami	F	1.5

3. Data Manipulation Language

+ GROUP BY

- The HAVING statement
  - Used to create a condition on a group of tuples
  - Attributes appearing in the HAVING statement must appear in the GROUP BY clause or be included in a aggregate function
  - Example: Group employees by department number, and show the department number, average salary, and maximum salary for the department with the average salary of 2,500,000 Korean Won.

```
SELECT      dno, AVG(salary) as avgsal, MAX(salary) as maxsal
FROM        employee
GROUP BY    dno
HAVING      avgsal >= 2500000;
```

3. Data Manipulation Language

❖ The HAVING statement

EMPLOYEE

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
3426	박영권	과장	4377	3000000	1
1365	김상원	사원	3426	1500000	1
2106	김창섭	대리	1003	2500000	2
1003	조민희	과장	4377	3000000	2
4377	이성래	사장	∧	5000000	2
3011	이수민	부장	4377	4000000	3
3427	최종철	사원	3011	1500000	3

Grouped

그룹화

GROUP BY →

DNO	AVGSAL	MAXSAL
1	2250000	3000000
2	3500000	5000000
3	2750000	4000000

HAVING 조건으로 조건 추가

HAVING →

DNO	AVGSAL	MAXSAL
2	3500000	5000000
3	2750000	4000000

3. Data Manipulation Language

- The ORDER BY statement
  - By default, the tuples are presented to the user in the order inserted in the relation
  - The ORDER BY statement is used to sort the resulting relation in ascending or descending order
  - Types of ORDER BY statement
    - Default sort order is ascending (ASC)
    - Sort order can be specified in descending order by specifying DESC
  - The null value appears first in ascending order and last in descending order

3. Data Manipulation Language

- The ORDER BY statement
  - Example: Search for the salary, title, and name of employees in department 2 and sort them in ascending order of salaries

```
SELECT      salary, title, empname
FROM        employee
WHERE       dno = 2
ORDER BY    salary;
```

오름차순 정렬

SALARY	TITLE	EMPNAME
2500000	대리	김창섭
3000000	과장	조민희
5000000	사장	이성래

3. Data Manipulation Language

- The LIMIT statement
  - Used to specify the number of records to return

"top" tuple or record 가 필요할 경우 유용하다

  - The LIMIT statement is useful on large tables with thousands of records
    - Returning a large number of records can impact performance

  - Example: Show the first three records from the employee table
    - **SELECT** *
    - **FROM** employee
    - **LIMIT** 3;

tuple 개수 제한.

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
2106	김창섭	대리	1003	2500000	2
3426	박영권	과장	4377	3000000	1
3011	이수민	부장	4377	4000000	3

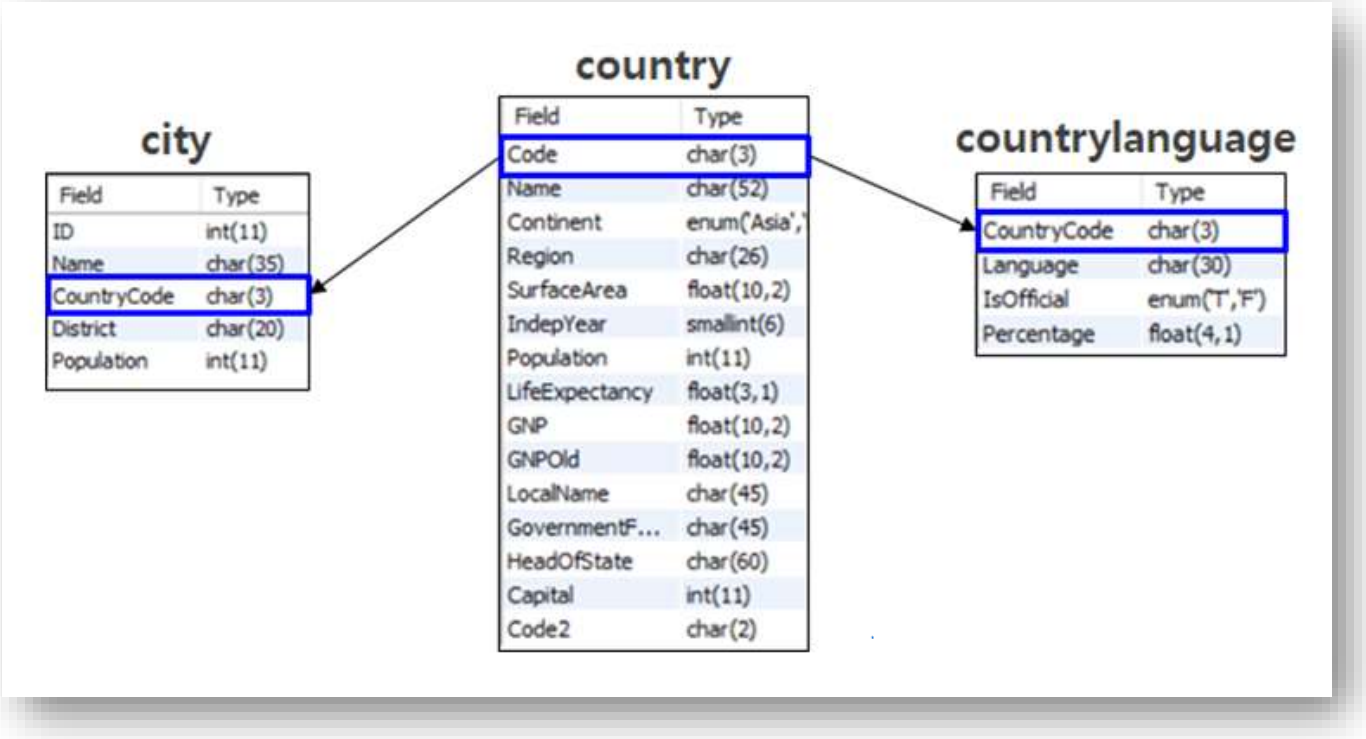
3. Data Manipulation Language

- The LIMIT statement
  - You can also use LIMIT to show the top or the bottom tuples
  - Example: Show an employee with the highest salary
    - **SELECT** *
    - **FROM** employee
    - **ORDER BY** salary **DESC**
    - **LIMIT** 1;

EMPNO	EMPNAME	TITLE	MANAGER	SALARY	DNO
4377	이성래	사장	NULL	5000000	2

3. Data Manipulation Language

- Quiz 4
  - Given the following WORLD database



  - Find languages that are spoken in at least 3 different countries with percentage at least 50

3. Data Manipulation Language

- More about data

CountryCode	Language	IsOfficial	Percentage
ABW	Dutch	T	5.3
ABW	English	F	9.5
ABW	Papiamento	F	76.7
ABW	Spanish	F	7.4
AFG	Balochi	F	0.9
AFG	Dari	T	32.1
AFG	Pashto	T	52.4
AFG	Turkmenian	F	1.9
AFG	Uzbek	F	8.8

3. Data Manipulation Language

- Answer to Query 4: Find languages that are spoken in at least 3 different countries with percentage at least 50.
  - **SELECT** Language, COUNT(CountryCode) AS N
  - **FROM** CountryLanguage
  - **WHERE** Percentage >= 50
  - **GROUP BY** Language
  - **HAVING** N > 2
  - **ORDER BY** N **DESC** ;

SELECT Language, COUNT(CountryCode) AS N
FROM CountryLanguage
WHERE Percentage >= 50
GROUP BY Language
HAVING N > 2
ORDER BY N DESC;

	Language	N
▶	Spanish	20
	Arabic	16
	English	11
	Creole English	8
	Creole French	6
	German	4
	Serbo-Croatian	3

4. Summary and Discussions

- Data Definition Language (DDL)
  - Notation for defining the database schema
    - Used to create and modify the structure of your relations and database objects
  - Statements
    - CREATE
    - DROP
    - ALTER
    - TRUNCATE

4. Summary and Discussions

- MySQL constraints
  - SQL constraints are used to specify rules for the data in a relation

  - MySQL constraints
    - **NOT NULL**
    - **PRIMARY KEY**
    - **FOREIGN KEY**
    - **UNIQUE**
    - **CHECK**
    - **DEFAULT**

4. Summary and Discussions

- Data Manipulation Language (DML)
  - A language for searching, modifying, inserting and deleting the desired data in the database
    - DML also known as query language
  - Statements
    - INSERT
    - UPDATE
    - DELETE
    - SELECT

4. Summary and Discussions

- The SELECT statement
  - Used to select/retrieve tuples from a database
  - Syntax

SELECT	[**DISTINCT**] attribute(s)	(1)	Compulsory
FROM	relation(s)	(2)	
[**WHERE**	condition]	(3)	Optional
[**GROUP BY**	attribute(s)]	(4)	
[**HAVING**	condition]	(5)	
[**ORDER BY**	attribute(s)]	(6)	
[**LIMIT**	number];	(7)	