SQL Basic

Prof. Hyuk-Yoon Kwon

https://sites.google.com/view/seoultech-bigdata

Most parts are based on slides used in Stanford
(http://web.stanford.edu/class/cs145)

Contents

- **Summary of Previous Lecture**
  - Introduction to SQL
    - SQL introduction & schema definitions

- **Today's lecture**
  - Basic single-table queries
  - Multi-table queries

1. SQL Introduction & Definitions

SQL Motivation

- **Dark times 5 years ago.**
  - Are databases dead?
- **Now, as before: everyone sells SQL**
  - Pig, Hive, Impala
- **“Not-Yet-SQL?”**

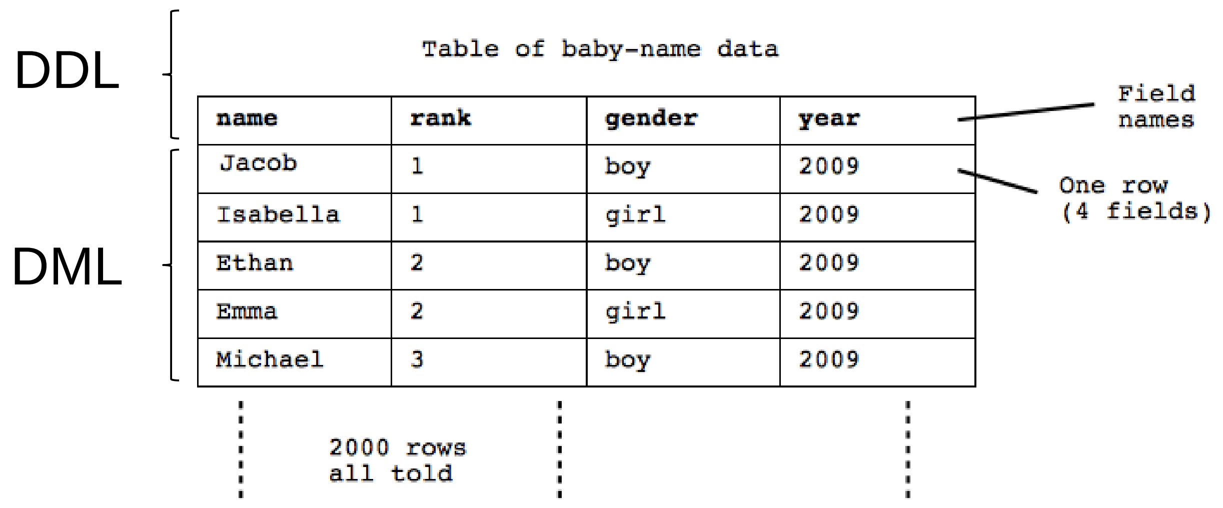
SQL is a…

Data Definition Language (DDL)

- Define relational *schemata*
- Create/alter/delete tables and their attributes

Data Manipulation Language (DML)

- Insert/delete/modify tuples in tables
- Query one or more tables



name	**rank**	**gender**	**year**
Jacob	1	boy	2009
Isabella	1	girl	2009
Ethan	2	boy	2009
Emma	2	girl	2009
Michael	3	boy	2009

Tables in SQL

Product

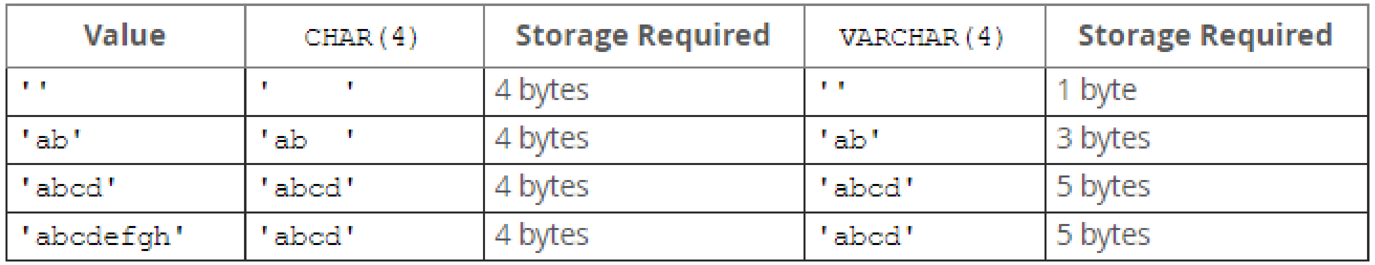
PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks
SingleTouch	$149.99	Canon
MultiTouch	$203.99	Hitachi

A **relation** or **table** is a multiset of tuples having the attributes specified by the schema

Data Types in SQL

- **Atomic types:**
  - Characters: CHAR(20), VARCHAR(50)
  - Numbers: INT, BIGINT, SMALLINT, FLOAT
  - Others: MONEY, DATETIME, …

- **Every attribute must have an atomic type**
  - Hence tables are flat



Value	CHAR(4)	Storage Required	VARCHAR(4)	Storage Required
`''`	`'    '`	4 bytes	`''`	1 byte
`'ab'`	`'ab  '`	4 bytes	`'ab'`	3 bytes
`'abcd'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes
`'abcdefgh'`	`'abcd'`	4 bytes	`'abcd'`	5 bytes

Study more: https://dev.mysql.com/doc/refman/5.7/en/char.html

NULL and NOT NULL

- **To say "don't know the value" we use NULL**
  - NULL has (sometimes painful) semantics, more detail later

Students(sid:string, name:string, gpa: float)

sid	name	gpa
123	Bob	3.9
143	Jim	NULL

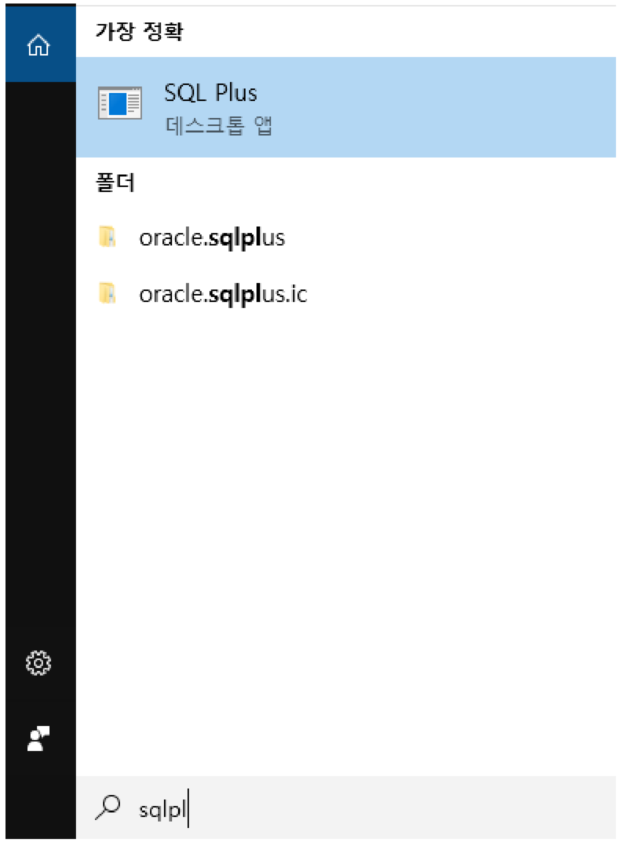
Say, Jim just enrolled in his first class.

In SQL, we may constrain a column to be NOT NULL, e.g., "name" in this table

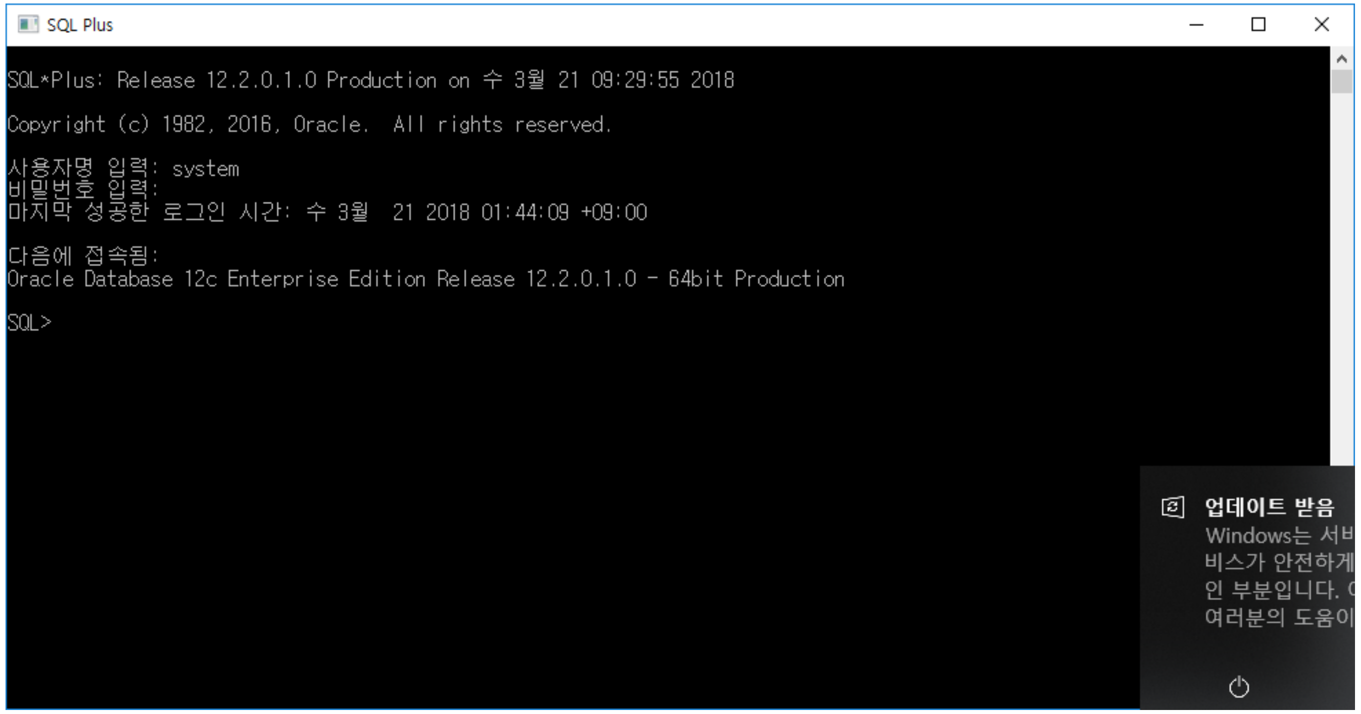
Oracle Practice #1

Start Oracle

- Execute SQL Plus



- User Authentication
  - ID: system
  - Password: oraclepractice



```
SQL*Plus: Release 12.2.0.1.0 Production on 수 3월 21 09:29:55 2018

Copyright (c) 1982, 2016, Oracle.  All rights reserved.

사용자명 입력: system
비밀번호 입력:
마지막 성공한 로그인 시간: 수 3월  21 2018 01:44:09 +09:00

다음에 접속됨:
Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production

SQL>
```

Oracle Practice #1

- **Build database with real data**
  1. Download ACDB.sql from e-class
  2. Copy ACDB.sql into a specific folder (e.g., c:/work/ACDB.sql)
  3. In SQLPlus, execute the following command
     - @c:/work/ACDB.sql
     - If some problems occur, execute the following command, and then execute the command above again
       - alter session set nls_language="AMERICAN";
  4. Check if data are stored correctly
     - select * from ACDB_SECTORS;
     - select * from ACDB_PACAKGES;
     - select * from ACDB_CUSTOMERS;

2. Single-table queries

What you will learn about in this section

1. The SFW query

2. Other useful operators: LIKE, DISTINCT, ORDER BY

SQL Query

- Basic form (there are many many more bells and whistles)

```
SELECT <attributes>
FROM   <one or more relations>
WHERE  <conditions>
```

Call this a **SFW** query.

Simple SQL Query: Selection

Selection is the operation of filtering a relation's tuples on some condition

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT *
FROM   Product
WHERE  Category = 'Gadgets'
```

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks

Simple SQL Query: Projection

Projection is the operation of producing an output table with tuples that have a subset of their prior attributes

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = ‘Gadgets’
```

PName	Price	Manufacturer
Gizmo	$19.99	GizmoWorks
Powergizmo	$29.99	GizmoWorks

Notation

Input schema

Product(PName, Price, Category, Manfacturer)

```
SELECT Pname, Price, Manufacturer
FROM   Product
WHERE  Category = ‘Gadgets’
```

Output schema

Answer(PName, Price, Manfacturer)

Database Schema

```
CREATE TABLE "ACDB_SECTORS"
  (          "SECTOR_ID" NUMBER(8,0),
             "SECTOR_NAME" VARCHAR2(25 BYTE)
  ) ;
```

```
CREATE TABLE "ACDB_PACKAGES"
  (          "PACK_ID" NUMBER(8,0),
             "SPEED" VARCHAR2(10 BYTE),
             "MONTHLY_PAYMENT" NUMBER(8,0),
             "SECTOR_ID" NUMBER(8,0),
             "STRT_DATE" DATE
  ) ;
```

```
CREATE TABLE "ACDB_CUSTOMERS"
  (          "CUSTOMER_ID" NUMBER(8,0),
             "FIRST_NAME" VARCHAR2(25 BYTE),
             "LAST_NAME" VARCHAR2(25 BYTE),
             "CITY" VARCHAR2(45 BYTE),
             "STATE" VARCHAR2(25 BYTE),
             "STREET" VARCHAR2(40 BYTE),
             "MAIN_PHONE_NUM" VARCHAR2(12 BYTE),
             "SECONDARY_PHONE_NUM" VARCHAR2(12 BYTE),
             "FAX" VARCHAR2(12 BYTE),
             "MONTHLY_DISCOUNT" NUMBER(4,2),
             "PACK_ID" NUMBER(8,0),
             "BIRTH_DATE" DATE,
             "JOIN_DATE" DATE
  ) ;
```

Oracle Practice #2

- **Given each description, make the corresponding SQL and practice it using ACDB.sql**
  1. Create a query to display the internet package number, internet speed and monthly payment (*Packages* table).
  2. Create a query to display the customer number, first name, last name, primary phone number, secondary phone number and package number (*Customers* table).
  3. Display the first name, last name, and package number for all customers whose last name is “King” (*Customers* table).
  4. Display the first name, last name, package number and monthly discount for all customers with monthly discount less than 10 (*Customers* table).

Useful Expressions

a	b
10	20

- **Arithmetic operation in SELECT clause**
  - SELECT a * 2 FROM table

a
20

- **Alias in SELECT clause**
  - SELECT a as A_RESULT FROM table

A_RESULT
10

- **Concatenate in SELECT clause**
  - SELECT a || ‘,’ || b as A_AND_B FROM table

A_AND_B
10, 20

Oracle Practice #3

- **Given each description, make the corresponding SQL and practice it using ACDB.sql**

1. Create a query to display the package number, speed, strt_date (the date when the package became available), monthly payment, and monthly payment * 12, name the last column "Y_INCOME" (*Packages* table).
2. Create a query to display the last name concatenated with the first name, separated by space, and main phone number concatenated with secondary phone number, separated by comma and space. Name the column heading FULL_NAME and CONTACT_DETAILS respectively. (*Customers* table).
3. Create a query to display the first name, last name, monthly discount and city concatenated with street, separated by space. Name the column headings: FN, LN, DC and FULL_ADDRESS respectively (*Customers* table).

A Few Details

- **SQL commands are case insensitive:**
  - Same: SELECT, Select, select
  - Same: Product,   product

- **Values are not:**
  - Different: ‘Seattle’, ‘seattle’

- **Use single quotes for constants:**
  - ‘abc’ - yes
  - “abc” - no

LIKE: Simple String Pattern Matching

```
SELECT *
FROM   Products
WHERE  PName LIKE '%gizmo%'
```

- **s LIKE p: pattern matching on strings**
- **p may contain two special symbols:**
  - % = any sequence of characters
  - _ = any single character

DISTINCT: Eliminating Duplicates

```
SELECT DISTINCT Category
FROM   Product
```

Category
Gadgets
Photography
Household

Versus

```
SELECT Category
FROM   Product
```

Category
Gadgets
Gadgets
Photography
Household

Oracle Practice #4

- **Given each description, make the corresponding SQL and practice it using ACDB.sql**
  1. Create a query to display unique cities from the *Customers* table.
  2. Create a query to display unique combination of cities and states from *Customers* table.
  3. Display the first name and monthly discount for all customers whose first name ends with an *e* (*Customers* table).
  4. Display the last name and package number for all customers where the second letter of their last name is *d* (*Customers* table).

ORDER BY: Sorting the Results

```
SELECT   PName, Price, Manufacturer
FROM     Product
WHERE    Category='gizmo' AND Price > 50
ORDER BY Price, PName
```

Ties are broken by the second attribute on the ORDER BY list, etc.

Ordering is ascending, unless you specify the DESC keyword.

BETWEEN Operation

a
20
40

- **BETWEEN operation in WHERE clause**
  - SELECT a FROM table WHERE a BETWEEN 10 AND 30

a
20

Oracle Practice #5

- **Given each description, make the corresponding SQL and practice it using ACDB.sql**
  1. Display the first name, join date, and package number for all customers who don’t have the letter *a* in their first name. Order the query in ascending order by package number (*Customers* table).
  2. Display the first name, join date, monthly discount, and package number for all customers whose monthly discount is over 28. Order the query in ascending order by monthly discount and package number (*Customers* table)
  3. Order the results of the previous problem (#2) in descending order by monthly discount and then in ascending order by package number (*Customers* table)
  4. Display the first name, join date, monthly discount where monthly discount is between 28 and 30 (*Customers* table)
  5. Display first name and join date where first name is between ‘B’ and ‘C’ (*Customers* table)

3. Multi-table queries

What you will learn about in this section

- **Foreign key constraints**
- **Joins: basics**
- **Joins: SQL semantics**

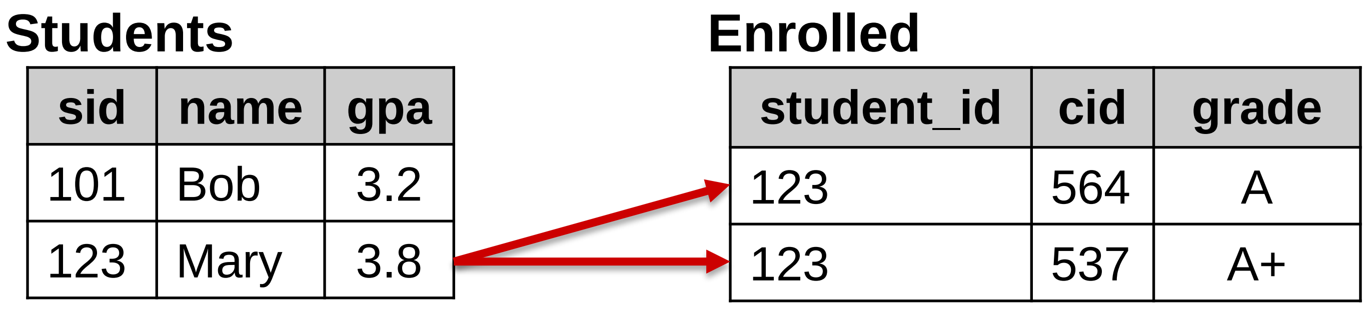
Foreign Key constraints

- **Suppose we have the following schema:**

Students(sid*: string,* name: *string*, gpa: *float*)

Enrolled(student_id: *string,* cid: *string*, grade: *string*)

- **And we want to impose the following constraint:**
  - “a student must appear in the Students table to enroll in a class”



Students

sid	name	gpa
101	Bob	3.2
123	Mary	3.8

Enrolled

student_id	cid	grade
123	564	A
123	537	A+

student_id alone is not a key- what is?

We say that student_id is a **foreign key** that refers to Students

Declaring Foreign Keys

```
Students(sid: string, name: string, gpa: float)
Enrolled(student_id: string, cid: string, grade: string)

CREATE TABLE Enrolled(
        student_id CHAR(20),
        cid             CHAR(20),
        grade   CHAR(10),
        PRIMARY KEY (student_id, cid),
        FOREIGN KEY (student_id) REFERENCES Students(sid)
)
```

Foreign Keys and update operations

Students(<u>sid</u>*: string*, name: *string*, gpa: *float*)

Enrolled(<u>student_id</u>: *string*, <u>cid</u>: *string*, grade: *string*)

- **What if we insert a tuple into Enrolled, but no corresponding student?**
  - INSERT is rejected (foreign keys are <u>constraints</u>)!

- **What if we delete a student?**
  1. Disallow the delete
  2. Remove all of the courses for that student
  3. *SQL allows a third via NULL (not yet covered)*

Keys and Foreign Keys

Company

CName	StockPrice	Country
GizmoWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

What is a foreign key vs. a key here?

Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

Joins

Product(<u>PName</u>, Price, Category, Manufacturer)

Company(<u>CName</u>, StockPrice, Country)

Ex: Find all products under $200 manufactured in Japan; return their names and prices.

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
          AND Country=‘Japan’
     AND Price <= 200
```

Joins

Product(PName, Price, Category, Manufacturer)

Company(CName, StockPrice, Country)

Ex: Find all products under $200 manufactured in Japan;
return their names and prices.

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
       AND Country=‘Japan’
    AND Price <= 200
```

A **join** between tables returns all unique combinations of their tuples **which meet some specified join condition**

Joins

Product(PName, Price, Category, Manufacturer)

Company(CName, StockPrice, Country)

Several equivalent ways to write a basic join in SQL:

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
          AND Country='Japan'
     AND Price <= 200
```

```
SELECT PName, Price
FROM   Product
JOIN   Company ON Manufacturer = Cname
                 AND Country='Japan'
WHERE  Price <= 200
```

A few more later on…

Joins

Product

PName	Price	Category	Manuf
Gizmo	$19	Gadgets	GWorks
Powergizmo	$29	Gadgets	GWorks
SingleTouch	$149	Photography	Canon
MultiTouch	$203	Household	Hitachi

Company

Cname	Stock	Country
GWorks	25	USA
Canon	65	Japan
Hitachi	15	Japan

$\Downarrow$

```
SELECT PName, Price
FROM   Product, Company
WHERE  Manufacturer = CName
         AND Country='Japan'
      AND Price <= 200
```

PName	Price
SingleTouch	$149.99

Primary Key and Foreign Key

```
ALTER TABLE "ACDB_SECTORS" ADD CONSTRAINT "SECTOR_ID_PK" PRIMARY KEY ("SECTOR_ID");
```

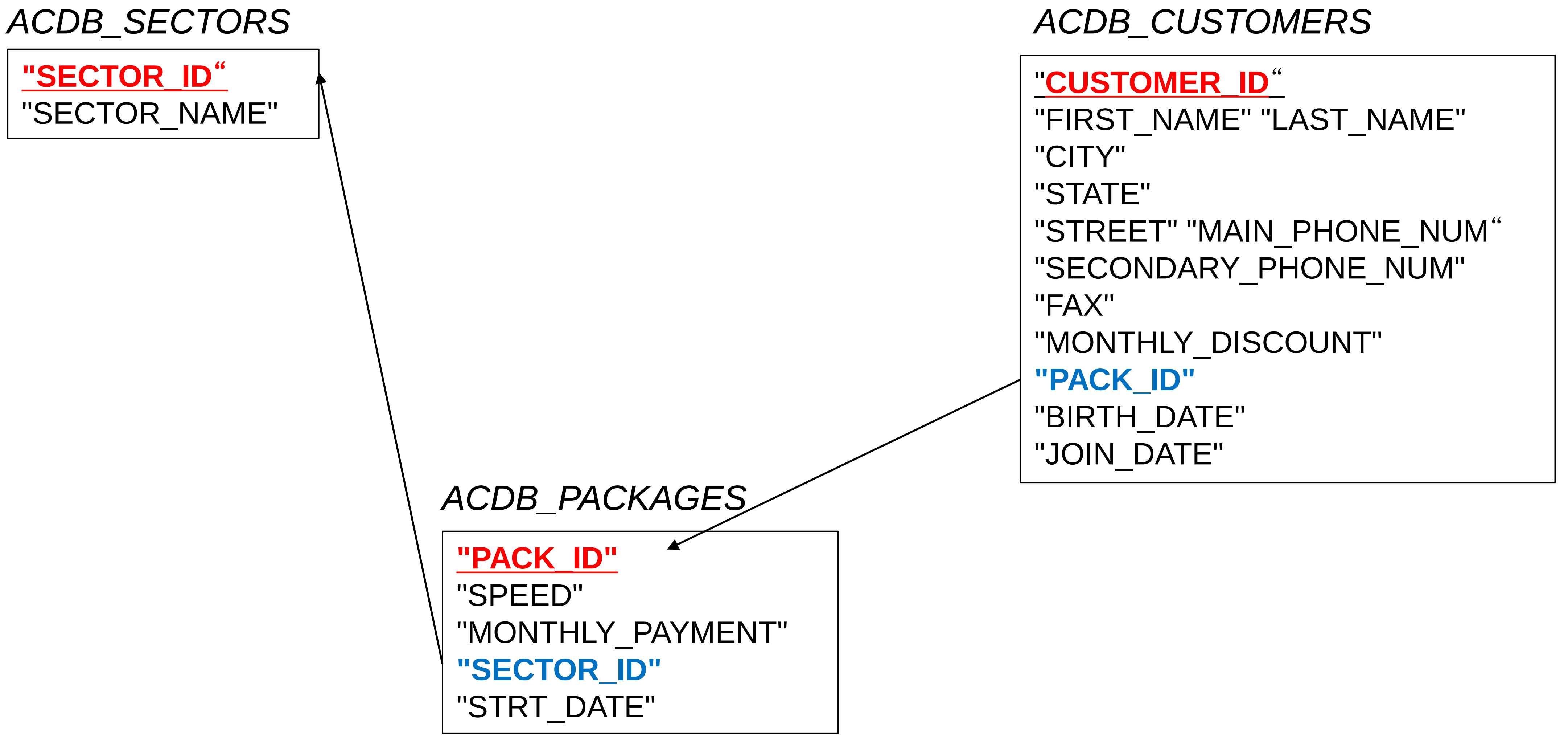
```
ALTER TABLE "ACDB_PACKAGES" ADD CONSTRAINT "PACK_ID_PK" PRIMARY KEY ("PACK_ID");
```

```
ALTER TABLE "ACDB_PACKAGES" ADD CONSTRAINT "SECTOR_ID_FK" FOREIGN KEY ("SECTOR_ID")
            REFERENCES "ACDB_SECTORS" ("SECTOR_ID") ENABLE;
```

```
ALTER TABLE "ACDB_CUSTOMERS" ADD CONSTRAINT "CUSTOMER_ID_PK" PRIMARY KEY ("CUSTOMER_ID");
```

```
ALTER TABLE "ACDB_CUSTOMERS" ADD CONSTRAINT "PACK_ID_FK" FOREIGN KEY ("PACK_ID")
            REFERENCES "ACDB_PACKAGES" ("PACK_ID") ENABLE;
```

Primary Key and Foreign Key



Oracle Practice #6

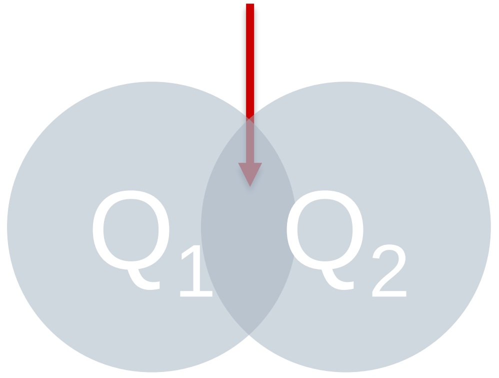
- **Given each description, make the corresponding SQL and practice it using ACDB.sql**
  1. Write a query to display first name, last name, package number and internet speed for all customers. (*Customers* and *Packages*)
  2. Display the package number, internet speed, monthly payment and sector name for all packages (*Packages* and *Sectors* tables).
  3. Display the customer name, package number, internet speed, monthly payment and sector name for all customers (*Customers*, *Packages* and *Sectors* tables).
  4. Display the customer name, package number, internet speed, monthly payment and sector name for all customers in the ‘Business’ sector (*Customers*, *Packages* and *Sectors* tables).

  * You may check the data stored in Sectors table

Explicit Set Operators: INTERSECT

```
SELECT R.A
FROM   R, S
WHERE  R.A=S.A
   INTERSECT
SELECT R.A
FROM   R, T
WHERE  R.A=T.A
```

$$\{r.A \mid r.A = s.A\} \cap \{r.A \mid r.A = t.A\}$$



■ **Constraint for Intersect**

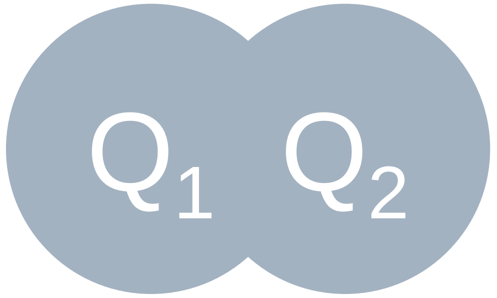
```
SELECT Name, BirthDate FROM Employee
INTERSECT
SELECT Name, BirthDate FROM Customer
```

```
SELECT Name, BirthDate FROM Employee
INTERSECT
SELECT Age, BirthDate, Name FROM Customer
```

UNION

$$\{r.A \mid r.A = s.A\} \cup \{r.A \mid r.A = t.A\}$$

```
SELECT  R.A
FROM    R, S
WHERE   R.A=S.A
UNION
SELECT R.A
FROM    R, T
WHERE   R.A=T.A
```



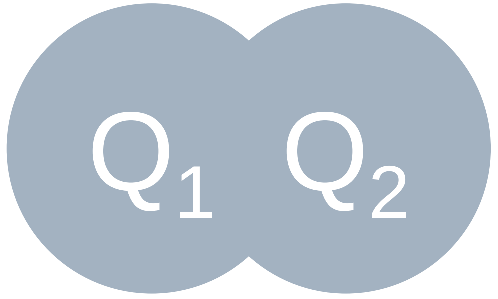
Why aren’t there duplicates?

What if we want duplicates?

UNION ALL

```
SELECT  R.A
FROM    R, S
WHERE   R.A=S.A
   UNION ALL
SELECT R.A
FROM    R, T
WHERE   R.A=T.A
```

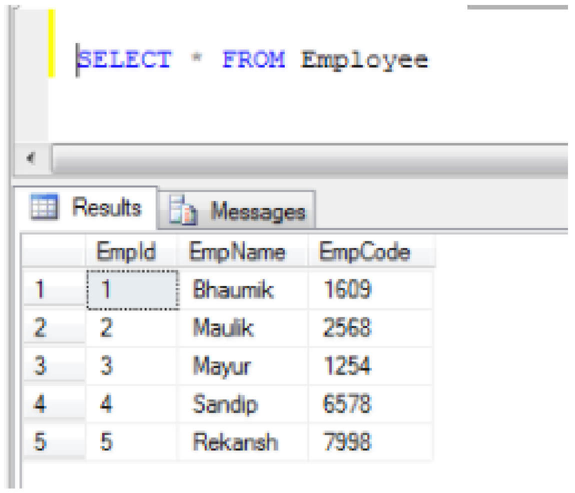
$\{r.A \mid r.A = s.A\} \cup \{r.A \mid r.A = t.A\}$



By default:
SQL uses set semantics

UNION vs. UNION ALL

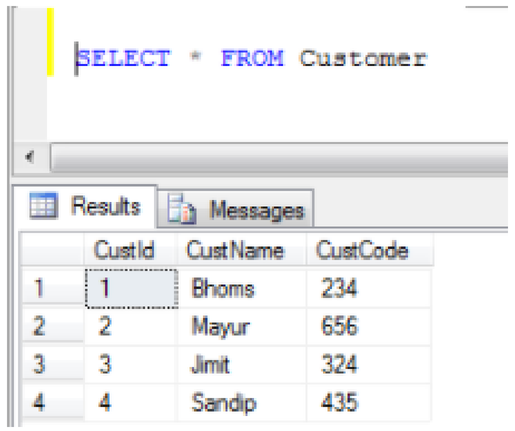
1. Employee table data:



```
SELECT * FROM Employee
```

	EmpId	EmpName	EmpCode
1	1	Bhaumik	1609
2	2	Maulik	2568
3	3	Mayur	1254
4	4	Sandip	6578
5	5	Rekansh	7998

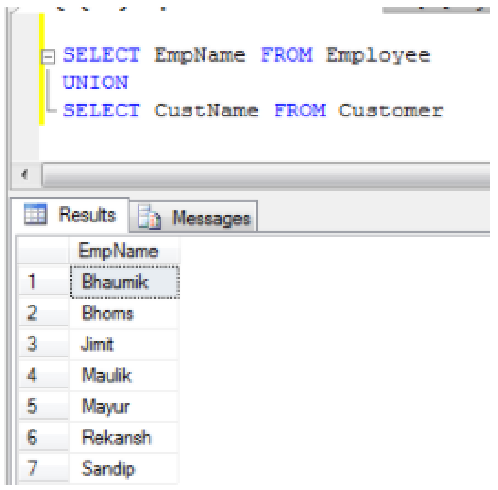
2. Customer table data:



```
SELECT * FROM Customer
```

	CustId	CustName	CustCode
1	1	Bhoms	234
2	2	Mayur	656
3	3	Jimit	324
4	4	Sandip	435

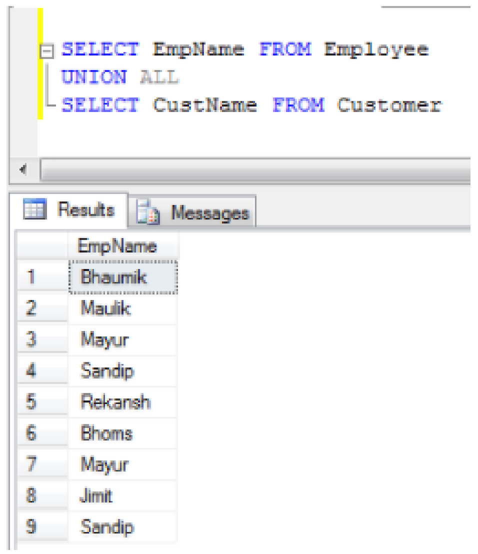
3. UNION Example (It removes all duplicate records):



```
SELECT EmpName FROM Employee
UNION
SELECT CustName FROM Customer
```

	EmpName
1	Bhaumik
2	Bhoms
3	Jimit
4	Maulik
5	Mayur
6	Rekansh
7	Sandip

4. UNION ALL Example (It just concatenate records, not eliminate duplicates, so it is faster than UNION):



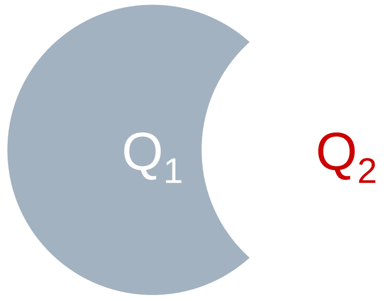
```
SELECT EmpName FROM Employee
UNION ALL
SELECT CustName FROM Customer
```

	EmpName
1	Bhaumik
2	Maulik
3	Mayur
4	Sandip
5	Rekansh
6	Bhoms
7	Mayur
8	Jimit
9	Sandip

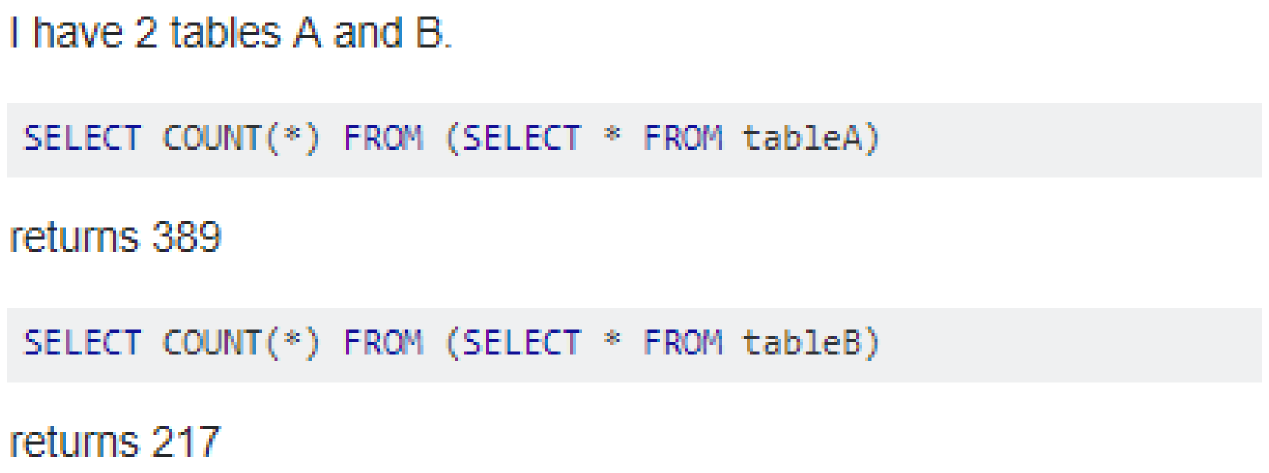
MINUS

```
SELECT R.A
FROM   R, S
WHERE  R.A=S.A
   MINUS
SELECT R.A
FROM   R, T
WHERE  R.A=T.A
```

$$\{r.A \mid r.A = s.A\} \setminus \{r.A \mid r.A = t.A\}$$



Interesting Result of MINUS!



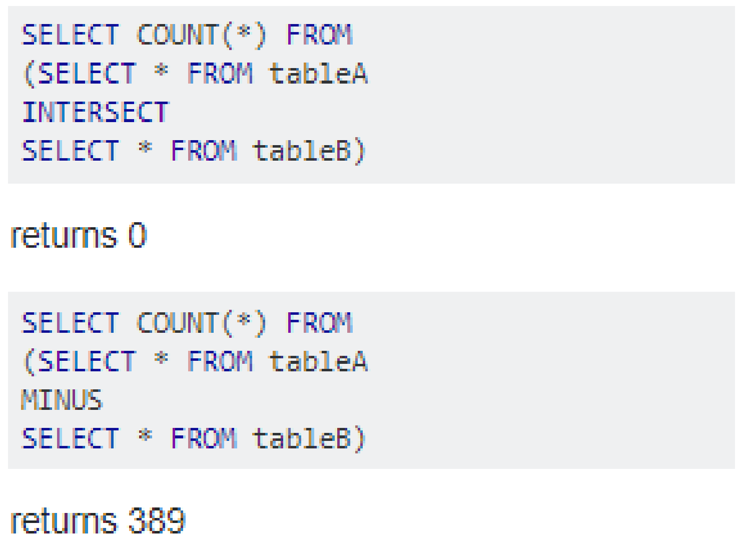
I have 2 tables A and B.

```
SELECT COUNT(*) FROM (SELECT * FROM tableA)
```

returns 389

```
SELECT COUNT(*) FROM (SELECT * FROM tableB)
```

returns 217

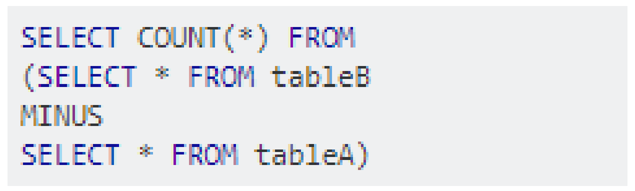


```
SELECT COUNT(*) FROM
(SELECT * FROM tableA
INTERSECT
SELECT * FROM tableB)
```

returns 0

```
SELECT COUNT(*) FROM
(SELECT * FROM tableA
MINUS
SELECT * FROM tableB)
```

returns 389



```
SELECT COUNT(*) FROM
(SELECT * FROM tableB
MINUS
SELECT * FROM tableA)
```

Oracle Practice #7 – Multiset operators

1. **When we have the following two conditions,**
   - A. IDs of customers who get discount monthly over $5 (i.e., > 5)
   - B. IDs of customers who pay monthly over $100 (i.e., > 100)
   - Find the following results and compare them: 1) A, 2) B, and 3) A INTERSECT B

2. **When we have the following two conditions, solve the problem**
   - A. Cities of customers who live in ‘California’
   - B. Cities of customers who include in ‘Business’ sector
   - Find the following results and compare them: 1) A, 2) B, 3) A UNION B, 4) A UNION ALL B

3. **When we have the following two conditions, solve the problem**
   - A. Names of customers who include in ‘Private’ sector
   - B. Names of customers who live in ‘Seattle’ or ‘San Franciso’
   - Find the following results and compare them: 1) A, 2) B, and 3) A MINUS B