CMSC 508
Databases

Introduction to SQL (II)

VCU College of Engineering

Dr. Alberto Cano
Associate Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 7th Ed. by Silberschatz, Korth, Sudarshan, 2019
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003

- SQL Query
  - A typical SQL query has the form:

SELECT $A_1, A_2, \ldots, A_n$
FROM $R_1, R_2, \ldots, R_m$
WHERE P

    - A_i represents an attribute (column), a literal, a function, or an operation
    - R_i represents a relation (table), cartesian product, join
    - P is a predicate (conditions to satisfy)
  - The result of a SQL query is a **relation**
  - Relational algebra equivalency: $\Pi\, A_1, A_2, \ldots, A_n\, (\sigma_P(R_1 \text{x}\, R_2 \text{x} \ \ldots \ \text{x}\, Rm))$

- SELECT clause
  - The select clause lists the attributes desired in the result of a query, corresponds to the projection operation of the relational algebra
  - SQL names are **case insensitive**
  - SQL **allows duplicates** in relations as well as in query results
  - To force the elimination of duplicates, use the keyword **distinct**
  - An **asterisk** in the select clause denotes “all attributes”
  - May rename columns using **alias**

```
SELECT   [DISTINCT] {*, column [[as] alias],...}
FROM     table;
```

- SELECT clause

SELECT *first_name*
FROM *employees;*

SELECT DISTINCT *first_name*
FROM *employees;*

SELECT DISTINCT *first_name, last_name*
FROM *employees;*

SELECT *department_id* **AS** *ID, department_name* **AS** *Department*
FROM *departments;*

SELECT * **FROM** *employees;*

- SELECT clause
  - An attribute can be a literal or function:

    SELECT *27* **FROM** *dual;*
    SELECT now() **FROM** *dual;*

  - An attribute can be a literal with **from** clause:

    SELECT '*ASD*' **AS** "*fOo*" ← double quotes to enforce upper/lowercase formatting of column name
    FROM *departments;*

  - The select clause can contain arithmetic and string expressions on constants or attributes of tuples

    SELECT CONCAT(*first_name*, ' ', *last_name*), *salary*12* **AS** "*ANNUAL SALARY*"
    FROM *employees;*

- WHERE clause
  - The where clause specifies conditions that the results must satisfy, corresponds to the selection predicate of the relational algebra
  - Comparisons can be combined with logical connectives **AND**, **OR**, **NOT**
  - Special operators: **BETWEEN**, **IN**, **IS NULL**

```
SELECT [DISTINCT] {*, column [[as] alias], ...}
FROM   table
WHERE  operand (< | <= | = | <> | >= | >) operand [and | or | not ...];
```

- WHERE clause

```
SELECT last_name, department_id
FROM employees
WHERE department_id = 110;
```

```
SELECT last_name, salary
FROM employees
WHERE salary < 10000;
```

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id  IN (100, 145, 146);
```

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 10000 AND 12000;
```

```
SELECT last_name, manager_id
FROM employees
WHERE manager_id  = 100 OR manager_id = 145 OR manager_id = 146;
```

```
SELECT last_name, job_id, manager_id
FROM employees
WHERE manager_id IS NULL;
```

- Exercise
  - Compute the annual salary of all employees as 12 x the monthly salary (attribute *salary*) plus a commission percentage of the monthly salary (attribute *commission_pct* ranged [0-1)). Show results in the form:

Format of the column header

Name	Salary	Annual
Russell, John	14000	173600
…		

Format of the data output

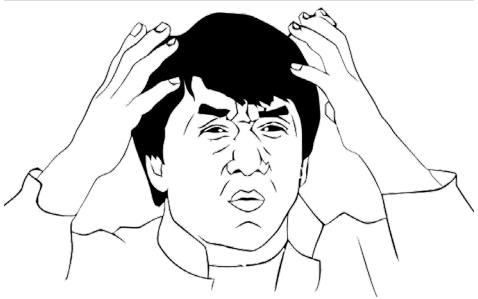
- Exercise
  - Compute the annual salary of all employees as 12 x the monthly salary (attribute *salary*) plus a commission percentage of the monthly salary (attribute *commission_pct* ranged [0-1)).

SELECT CONCAT(*last_name*, ', ', *first_name*) **AS** "Name", *salary* **AS** "Salary", *salary*12+commission_pct*salary* **AS** "Annual"
FROM *employees*;

- Exercise
  - Compute the annual salary of all employees as 12 x the monthly salary (attribute *salary*) plus a commission percentage of the monthly salary (attribute *commission_pct* ranged [0-1)).

SELECT CONCAT(*last_name*, ', ', *first_name*) **AS** "Name", *salary* **AS** "Salary", *salary*12+commission_pct*salary* **AS** "Annual"
FROM *employees*;

Name	Salary	Annual
King, Steven	24000.00	*NULL*
Kochhar, Neena	17000.00	*NULL*
De Haan, Lex	17000.00	*NULL*
Hunold, Alexander	9000.00	*NULL*
Ernst, Bruce	6000.00	*NULL*
Austin, David	4800.00	*NULL*
Pataballa, Valli	4800.00	*NULL*
Lorentz, Diana	4200.00	*NULL*



- IFNULL for NULL values
  - IFNULL(*expr1,expr2*) replaces ***NULL*** with a value in the results of a query.
    - If *expr1* is **NOT *NULL***, then IFNULL returns *expr1*
    - If *expr1* is ***NULL***, then IFNULL returns *expr2*

SELECT CONCAT(*last_name*, ', ', *first_name*) **AS** "Name", *salary* **AS** "Salary",
*salary*12+* IFNULL*(commission_pct,0)*salary* **AS** "Annual"
FROM *employees*;

SELECT *last_name,* IFNULL*(commission_pct, 'Not Applicable')*
AS "*COMMISSION*" **FROM** *employees;*

- **NULL** signifies an unknown value or that a value does not exist

- STRING operations
  - The operator **LIKE** uses patterns (**case insensitive**) (use **LIKE BINARY** for **case sensitive**) for string-matching operations using two special characters:
    - percentage (%) matches **any substring** (none or many characters)
    - underscore (_) matches **any single character**
  - Examples:

'Intro%'	matches any string beginning with “Intro”
'%Comp%'	matches any string containing “Comp” as a substring
'_ _ _'	matches any string of exactly three characters
'_ _ _ %'	matches any string of at least three characters
'%_ a _ '	same as before but the second to the last letter is 'a'

- STRING operations

```
SELECT last_name FROM employees
WHERE last_name LIKE 'Mc%';

SELECT last_name FROM employees
WHERE last_name LIKE BINARY 'MC%';

SELECT first_name FROM employees
WHERE first_name LIKE 'D__i%';

SELECT phone_number FROM employees
WHERE phone_number LIKE '%123%';

SELECT phone_number FROM employees
WHERE phone_number LIKE '%.123.%';
```

- Logical operators
  - AND, OR, NOT as in Boolean algebra
  - Operator precedence:

INTERVAL
BINARY, COLLATE
!
- (unary minus), ~ (unary bit inversion)
^
*, /, DIV, %, MOD
-, +
<<, >>
&
|
= (comparison), <=>, >=, >, <=, <, <>, !=, IS, LIKE, REGEXP, IN
BETWEEN, CASE, WHEN, THEN, ELSE
NOT
AND, &&
XOR
OR, ||
= (assignment), :=

- Logical operators

SELECT *last_name*, *job_id*, *salary* **FROM** *employees*
WHERE *job_id* = 'SA_MAN' **OR** *job_id* = 'AD_PRES' **AND** *salary* >= 14000;

SELECT *last_name*, *job_id*, *salary* **FROM** *employees*
WHERE (*job_id* = 'SA_MAN' **OR** *job_id* = 'AD_PRES') **AND** *salary* >= 14000;

- **NOT IN** and **NOT NULL**

SELECT *last_name*, *job_id* **FROM** *employees*
WHERE *job_id* **NOT IN** ('SA_MAN' , 'AD_PRES') **AND** *department_id* = 50;

SELECT *last_name*, *commission_pct* **FROM** *employees*
WHERE *commission_pct* **IS NOT NULL**;

** **NOT IN** (NULL, A, B, C, …) is **ALWAYS** FALSE --> careful with this operation

- Sort the output

SELECT DISTINCT *last_name* **FROM** *employees*
ORDER BY *last_name*;

SELECT *last_name, first_name* **FROM** *employees*
ORDER BY *last_name, first_name;*

SELECT *employee_id, manager_id* **FROM** *employees*
ORDER BY *manager_id ASC, employee_id DESC;*

- Manipulation functions (there're maaaaany, here's some of them)

Function	Results
LOWER ('BD sql')	bd sql
UPPER ('BD sql')	BD SQL
INITCAP ('BD sql')	BD Sql
CONCAT ('BD', 'SQL')	BDSQL
SUBSTR ('MYSQL',1,3)	MYS
INSTR ('MYSQL','Y')	2
LPAD (salary, 10, '*')	******5000
ROUND (7.968, 2)	7.97
TRUNC (7.968, 2)	7.96
MOD (1600, 300)	100

- Exercises

1. Find the department names containing IT
2. Find the employees names working for the Purchasing department
3. Find the employees names whose manager is the president: Steven King
4. Find the employees not working for any department
5. Find the locations of the offices sorted by country, state, and city (use ORDER BY)
6. Find the employees working for any of the IT departments and earning more than $5k. Show their salary with the $ symbol first.
7. Find the employees hired before Jan 1st, 2005 and whose first name ends with 'n'
8. Build the email of the employees by combining the first letter of their first name plus the last name plus "@dundermifflinpaper.com" (Yes, https://dundermifflinpaper.com/ exists)

CMSC 508
Databases

Introduction to SQL (II)

VCU College of Engineering

Dr. Alberto Cano
Associate Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 7th Ed. by Silberschatz, Korth, Sudarshan, 2019
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003