CMSC 508 Database Theory

Introduction to SQL (II)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
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)
    - R_i represents a relation (table)
    - P is a predicate (conditions to satisfy)
  - The result of a SQL query is a **relation**
  - Relational algebra equivalency: $\Pi\, A_1, A_2, \ldots, A_n\, (\sigma_P(R_1 \text{ x } R_2 \text{ x } \ldots \text{ x } R_m))$

- 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 *department_id* ***as*** *ID, department_name* ***as*** *Department*
from *departments;*

select *
from *employees;*

- SELECT clause
  - An attribute can be a literal with **no from** clause:

    select *'27'* **from** *dual;*

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

    select *'ASD'* **as** "*fOo*"
    from *departments;*

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

    select *first_name* || ' ' || *last_name,*
    *salary*12* **as** "*ANNUAL SALARY*",
    2(300 + salary)* **as** "*BONUS SALARY*"
    from *employees;*

- WHERE clause
  - The where clause specifies conditions that the result 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;
```

- 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, salary
from employees
where salary between 10000 and 12000;
```

```
select last_name, manager_id
from employees
where manager_id  in (100, 145, 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 times 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

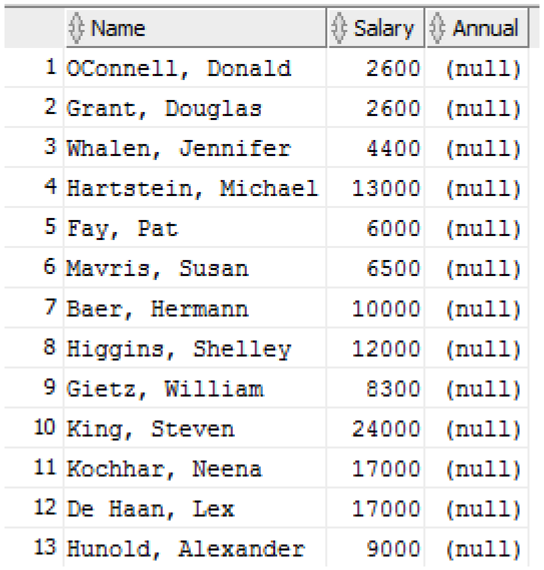
- Exercise
  - Compute the annual salary of all employees as 12 times the monthly salary (attribute *salary*) plus a commission percentage of the monthly salary (attribute *commission_pct* ranged [0-1)).

select *last_name* || ', ' || *first_name* **as** "Name", *salary* **as** "Salary", *salary*12+commission_pct*salary* **as** "Annual" **from** *employees*;

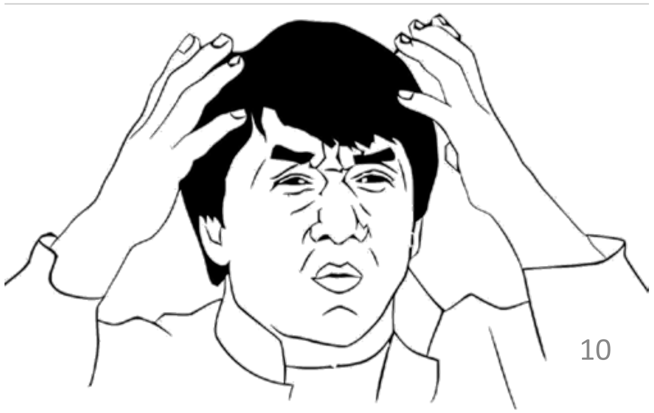
- Exercise

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

select *last_name* || ', ' || *first_name* **as** "Name", *salary* **as** "Salary", *salary*12+commission_pct*salary* **as** "Annual" **from** *employees*;



	Name	Salary	Annual
1	OConnell, Donald	2600	(null)
2	Grant, Douglas	2600	(null)
3	Whalen, Jennifer	4400	(null)
4	Hartstein, Michael	13000	(null)
5	Fay, Pat	6000	(null)
6	Mavris, Susan	6500	(null)
7	Baer, Hermann	10000	(null)
8	Higgins, Shelley	12000	(null)
9	Gietz, William	8300	(null)
10	King, Steven	24000	(null)
11	Kochhar, Neena	17000	(null)
12	De Haan, Lex	17000	(null)
13	Hunold, Alexander	9000	(null)



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

select *last_name* || ', ' || *first_name* **as** "Name", *salary* **as** "Salary",
*salary*12+NVL(commission_pct,0)*salary* **as** "Annual"
from *employees*;

select *last_name, NVL(TO_CHAR(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 sensitive**) for string-matching operations using two special characters:
    - percent (%) 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 but the second last letter is ‘a’

- STRING operations

  select *last_name* **from** *employees*
  where *last_name* **like** '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;

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**;

- Ordering the output
  - An ORDER BY clause allows you to specify the order in which rows appear in the result set (ascending, descending)
  - Can sort according to multiple attributes

```
SELECT          [DISTINCT] {*, column [[as] alias], ...}
FROM            table
[WHERE          condition(s)]
[ORDER BY      {column+} [ASC | DESC]];
```

select distinct *last_name* **from** *employees*
order by *last_name*;

select *last_name, first_name* **from** *employees*
order by *last_name, first_name*;

- Manipulation functions (there’re many, 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 (‘ORACLE’,1,3)	ORA
INSTR (‘ORALCE’,’R’)	2
LPAD (salary, 10, ‘*’)	******5000
ROUND (7.968, 2)	7.97
TRUNC (7.968, 2)	7.96
MOD (1600, 300)	100

CMSC 508
Database Theory

Introduction to SQL (II)

Dr. Alberto Cano
Assistant Professor
Department of Computer Science

Chapter 3 from Database System Concepts, 6th Ed. by Silberschatz, Korth, Sudarshan, 2011
Chapter 5 from Database Management Systems, 3rd Ed. by Ramakrishnan, Gehrke, 2003