CMSC 508
Database Theory

Introduction to SQL (III)

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

- DATE type
  - Stores point-in-time values (dates and times) including the year, the month, the day, the hours, the minutes, and the seconds

select *SYSDATE* **from** *dual;*

select *last_name, (SYSDATE-hire_date)/7* **as** *weeks* **from** *employees;*

  - TO_CHAR(date, format) and TO_DATE(date, format)

select *TO_CHAR(SYSDATE, 'DD MONTH YYYY')* **as** *Today* **from** *dual*;

select *TO_DATE('2003/07/09', 'yyyy/mm/dd')* **from** dual;

insert into *foo (bname, bday)* **values** *('ANDY',TO_DATE('13-AUG-66 12:56 A.M.','DD-MON-YY HH:MI A.M.'));*

- DATE type

Format	Description
YYYY	4-digit year
MM	Month (01-12; JAN = 01)
MONTH	Name of month, padded with blanks to length of 9 characters
DAY	Name of day
DD	Day of month (1-31)

- Date functions:

select *hire_date*, *ADD_MONTHS(hire_date,1)* **from** *employees*;

select *NEXT_DAY(sysdate,'TUESDAY')* **as** "NEXT TUESDAY" **from** *dual*;

select *MONTHS_BETWEEN(TO_DATE('02-02-1995','MM-DD-YYYY'), TO_DATE('01-01-1995','MM-DD-YYYY'))* **as** "Elapsed" **from** *dual*;

- FROM clause
  - Lists the relations involved in the query, corresponds to the Cartesian product operation of the relational algebra

select * **from** *employees, departments;*

Remember! Cartesian product multiplies the number of rows in every table involved! Then, 107 employees x 27 departments = 2889 rows!

  - Cartesian product is very useful when combined with where-clause predicates

select * **from** *employees, departments*
where *employees.department_id = departments.department_id;*

Provides useful information about 106 employees and their dept inf

* There is one employee omitted because his department_id is null

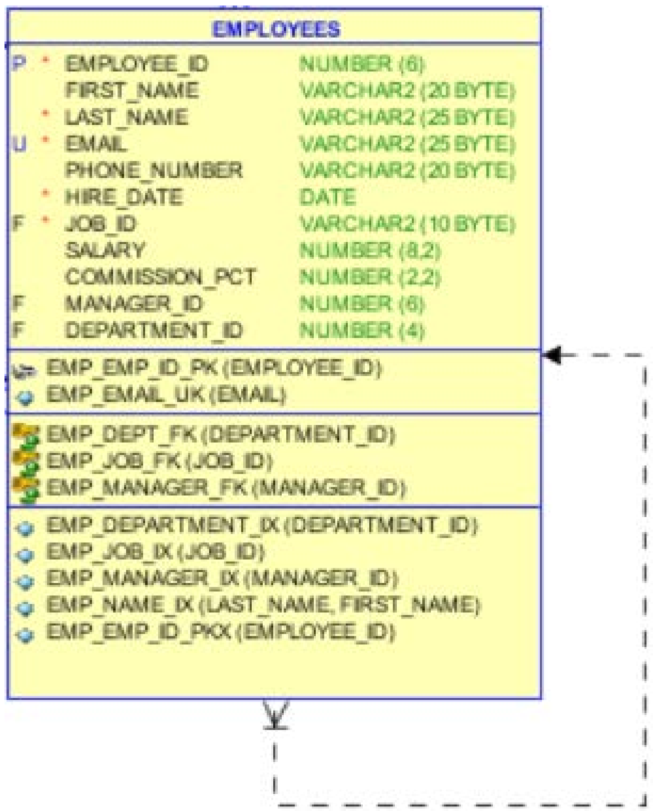
- FROM clause

select *last_name, department_id, department_name*
from *employees, departments*
where *employees.department_id = departments.department_id;*

```
ORA-00918: column ambiguously defined
00918. 00000 -  "column ambiguously defined"
```

select *e.last_name, d.department_id, d.department_name*
from *employees e, departments d*
where *e.department_id = d.department_id;*

- Exercise
  - Show for all employees their last name and their manager last name



EMPLOYEES

P	*	EMPLOYEE_ID	NUMBER (6)
		FIRST_NAME	VARCHAR2 (20 BYTE)
	*	LAST_NAME	VARCHAR2 (25 BYTE)
U	*	EMAIL	VARCHAR2 (25 BYTE)
		PHONE_NUMBER	VARCHAR2 (20 BYTE)
	*	HIRE_DATE	DATE
F	*	JOB_ID	VARCHAR2 (10 BYTE)
		SALARY	NUMBER (8,2)
		COMMISSION_PCT	NUMBER (2,2)
F		MANAGER_ID	NUMBER (6)
F		DEPARTMENT_ID	NUMBER (4)

EMP_EMP_ID_PK (EMPLOYEE_ID)
EMP_EMAIL_UK (EMAIL)

EMP_DEPT_FK (DEPARTMENT_ID)
EMP_JOB_FK (JOB_ID)
EMP_MANAGER_FK (MANAGER_ID)

EMP_DEPARTMENT_IX (DEPARTMENT_ID)
EMP_JOB_IX (JOB_ID)
EMP_MANAGER_IX (MANAGER_ID)
EMP_NAME_IX (LAST_NAME, FIRST_NAME)
EMP_EMP_ID_PKX (EMPLOYEE_ID)

- Exercise
  - Show for all employees their last name and their manager last name

select *e.last_name* **as** Employee, *m.last_name* **as** Manager
from *employees* e, *employees* m
where *e.manager_id* = *m.employee_id*;

106 rows returned.

King is not returned because it has no manager.

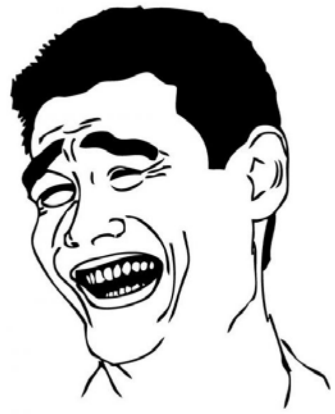
Edit the query to include King.

- Exercise
  - Show for all employees their last name and their manager last name

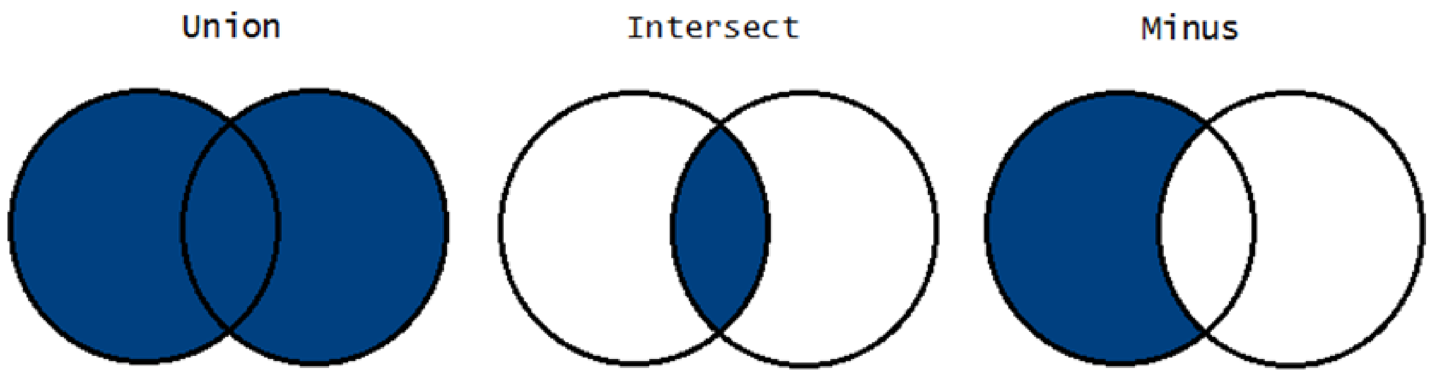
select *e.last_name* **as** Employee, *m.last_name* **as** Manager
from *employees* e, *employees* m
where *e.manager_id* = *m.employee_id*
or *e.manager_id* **is null**;

213 rows returned.

Why?



- Set operators
  - Set operations **union**, **intersect**, and **minus**
  - Each of the above operations automatically eliminates duplicates
  - To retain all duplicates use **union all**



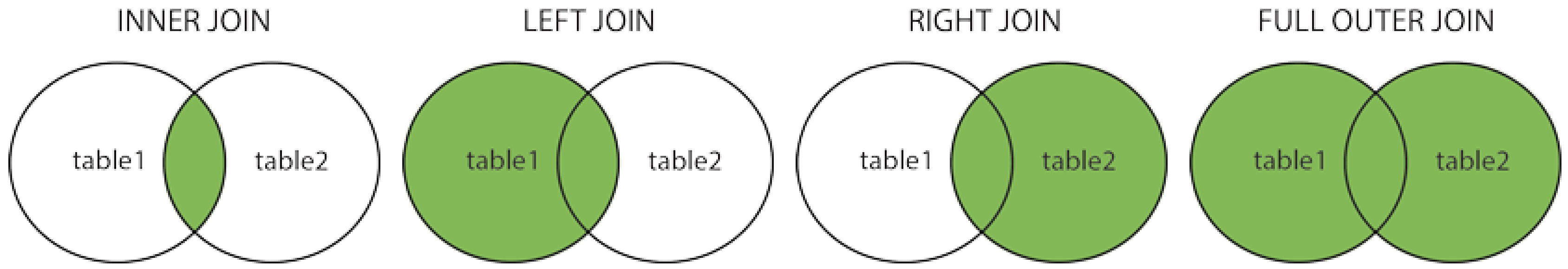
- Exercise
  - Show for all employees their last name and their manager last name using set operations

- Exercise
  - Show for all employees their last name and their manager last name using set operations

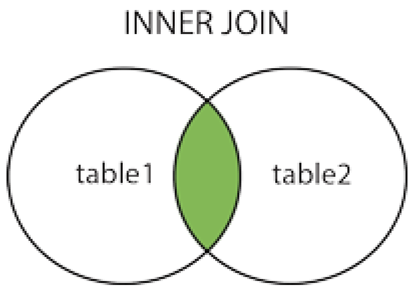
select *e.last_name* **as** Employee, *m.last_name* **as** Manager
from *employees* e, *employees* m
where *e.manager_id* = *m.employee_id*
union
select *e.last_name* **as** Employee, *null* **as** Manager
from *employees* e
where *e.manager_id* **is null***;*

107 rows returned.

- Join operators
  - Join takes two relations and returns as a result another relation
  - A join operation is a Cartesian product which requires that tuples in the two relations match under some condition
  - Typically used as subquery expressions in the from clause
  - **(inner) join, left (outer) join, right (outer) join, full (outer) join**



- Join operators



select *
from *employees* **inner join** *departments*
on *employees.department_id = departments.department_id*;

equivalent to

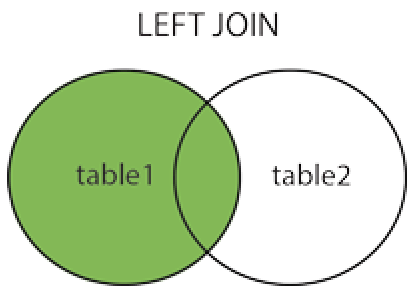
select *
from *employees*, *departments*
where *employees.department_id = departments.department_id*;

not equivalent to (because there are two columns for matching the natural join)

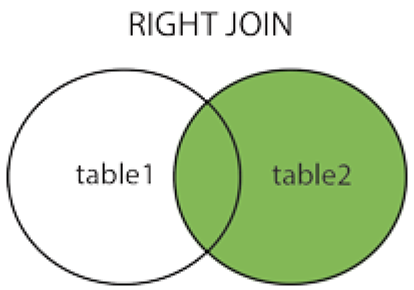
select *
from *employees* **natural inner join** *departments;*

- Join operators

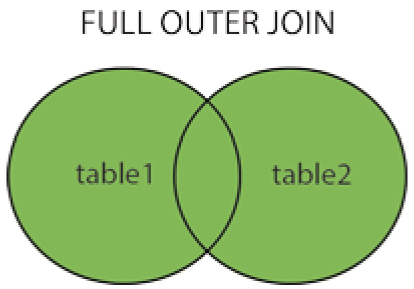
select *
from *employees* **left join** *departments*
on *employees.department_id = departments.department_id*;



select *
from *employees* **right join** *departments*
on *employees.department_id = departments.department_id*;



select *
from *employees* **full join** *departments*
on *employees.department_id = departments.department_id*;

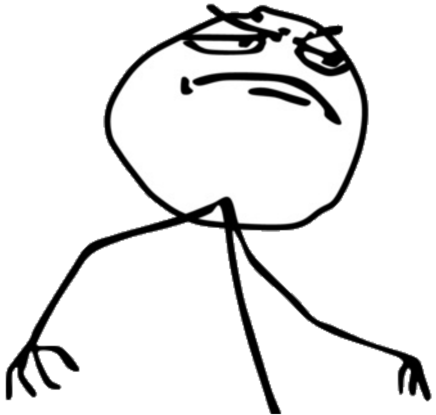


- Exercise
  - Show for all employees their last name and their manager last name using join operations

- Exercise
  - Show for all employees their last name and their manager last name using join operations

select *e.last_name* **as** Employee, *m.last_name* **as** Manager
from *employees* e **left join** *employees* m
on *e.manager_id* = *m.employee_id*;

107 rows returned.



CMSC 508
Database Theory

Introduction to SQL (III)

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