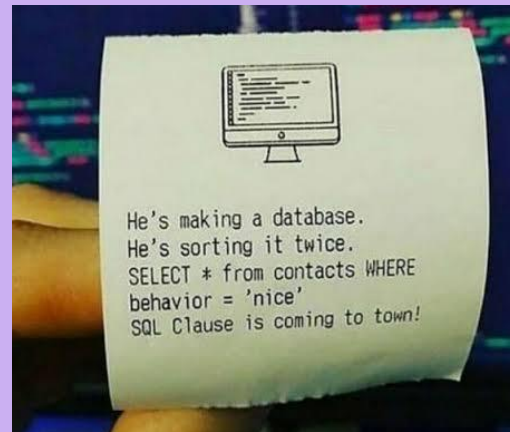# COMP3311 Week 4

By Manhua Lu

Adapted from Kyu-Sang Kim

Announcements

- Quiz 3 due Friday 6 October @11:59pm AEST

- Assignment 1 due next week! (13 October @11:59pm)
  - Autotesting now available, just run the command
    - "3311 autotest ass1"
  - If you don't know how to start, Prac Exercise 3, 4 & 5 are helpful for assignment!
    - And this tutorial!!

Learning Objectives

01 → SQL Update Statements (Q2)

02 → SQL Select Statements and Joins

03 → SQL Select Aggregates

04 → SQL Deletion Statements (Q10, Q11)

05 → Putting it all to practice...!

SQL Data Definition Language (DDL) Recap

SQL DDL Recap

```
create table Employees (
      eid      integer,
      ename    text,
      age      integer,
      salary   real,
      primary key (eid)
);
create table Departments (
      did      integer,
      dname    text,
      budget   real,
      manager integer references Employees(eid),
      primary key (did)
);
create table WorksIn (
      eid      integer references Employees(eid),
      did      integer references Departments(did),
      percent real,
      primary key (eid,did)
);
```
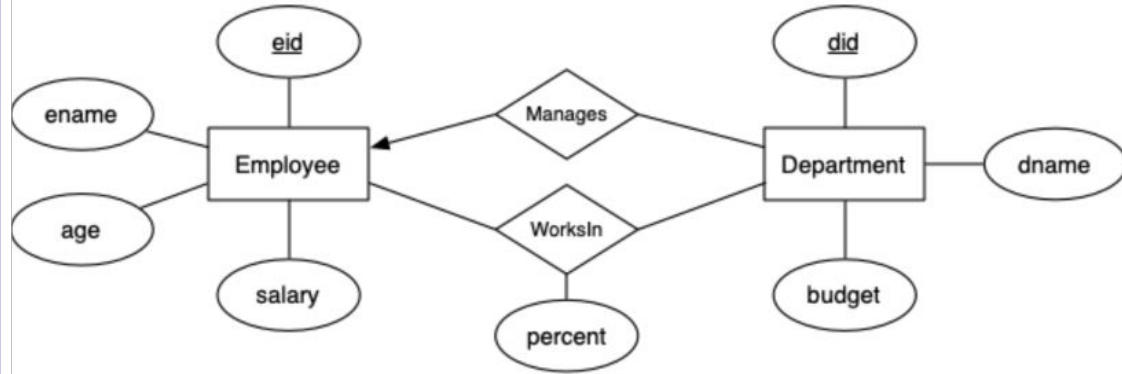

Note: Employees table is created before it is referenced
in the Departments table because order matters!!

01

SQL Update Statements

01 → SQL Update Statements

```sql
update   TableName
set      Attribute = newAttribute
where    condition
```

2. A new government initiative to get more young people into work cuts the salary levels of all workers under 25 by 20%. Write an SQL statement to implement this policy change.

(Question is basically asking us to modify the contents of the SQL table)

```
create table Employees (
     eid     integer,
     ename   text,
     age     integer,
     salary  real,
     primary key (eid)
);
create table Departments (
     did     integer,
     dname   text,
     budget  real,
     manager integer references Employees(eid),
     primary key (did)
);
create table WorksIn (
     eid     integer references Employees(eid),
     did     integer references Departments(did),
     percent real,
     primary key (eid,did)
);
```

02

SQL Select Statements and Joins

```
select    attribute
from      TableName
where     condition
```

"Jack works in Bed", "John works in Clothes", …

Employees		
eid	ename	…
1	Jack	
2	John	
3	Jill	

WorksIn		
eid	did	%
1	3	0.8
2	1	0.5
3	2	0.6

Departments		
did	dname	…
1	Clothes	
2	Tech	
3	Bed	

```
create table Employees (
    eid     integer,
    ename   text,
    age     integer,
    salary  real,
    primary key (eid)
);
create table Departments (
    did     integer,
    dname   text,
    budget  real,
    manager integer references Employees(eid),
    primary key (did)
);
create table WorksIn (
    eid     integer references Employees(eid),
    did     integer references Departments(did),
    percent real,
    primary key (eid,did)
);
```

02 → SQL Select Statements and Joins

```
select   e.ename, d.dname
from     Employees e
join     WorksIn w on (e.eid = w.eid)
join     Departments d on (w.did = d.did)
where    additional conditions
```

Employees		
eid	ename	…
1	Jack	
2	John	
3	Jill	

WorksIn		
eid	did	%
1	3	0.8
2	1	0.5
3	2	0.6

Departments		
did	dname	…
1	Clothes	
2	Tech	
3	Bed	

SQL Select Statements and Joins

```
select   e.ename, d.dname
from     Employees e
join     WorksIn w on (e.eid = w.eid)
join     Departments d on (w.did = d.did)
where    additional conditions
```

Employees		
eid	ename	...
1	Jack	
2	John	
3	Jill	

WorksIn		
eid	did	%
1	3	0.8
2	1	0.5
3	2	0.6

Departments		
did	dname	...
1	Clothes	
2	Tech	
3	Bed	

The 1st join statement creates this new table →

e.eid	e.ename	w.eid	w.did	w.%
1	Jack	1	3	0.8
2	John	2	1	0.5
3	Jill	3	2	0.6

SQL Select Statements and Joins

Employees		
eid	ename	…
1	Jack	
2	John	
3	Jill	

WorksIn		
eid	did	%
1	3	0.8
2	1	0.5
3	2	0.6

Departments		
did	dname	…
1	Clothes	
2	Tech	
3	Bed	

```
select   e.ename, d.dname
from     Employees e
join     WorksIn w on (e.eid = w.eid)
join     Departments d on (w.did = d.did)
where    additional conditions
```

The 1st join statement creates this new table →

e.eid	e.ename	w.eid	w.did	w.%
1	Jack	1	3	0.8
2	John	2	1	0.5
3	Jill	3	2	0.6

The 2nd join statement creates this new table →

e.eid	e.ename	w.eid	w.did	w.%	d.did	d.dname
1	Jack	1	3	0.8	3	Bed
2	John	2	1	0.5	1	Clothes
3	Jill	3	2	0.6	2	Tech

→ # SQL Select Statements and Joins

```
select   e.ename, d.dname
from     Employees e
join     WorksIn w on (e.eid = w.eid)
join     Departments d on (w.did = d.did)
where    additional conditions
```
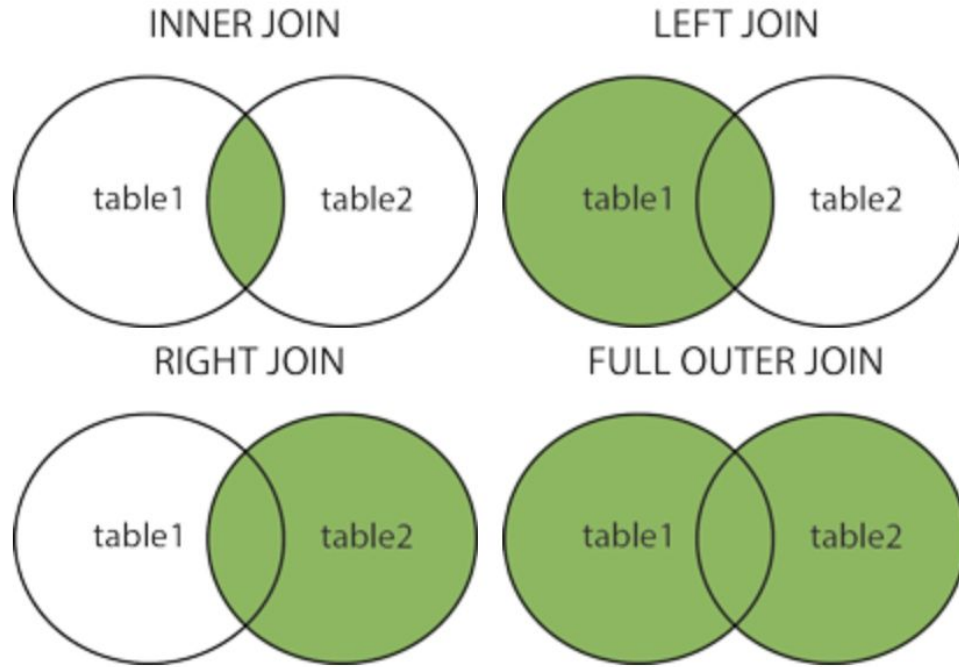
e.eid	e.ename	w.eid	w.did	w.%	d.did	d.dname
1	Jack	1	3	0.8	3	Bed
2	John	2	1	0.5	1	Clothes
3	Jill	3	2	0.6	2	Tech

The select statement displays only the ename and dname attr from the prev table →

e.ename	d.dname
Jack	Bed
John	Clothes
Jill	Tech

02 → **SQL Joins**

INNER JOIN

LEFT JOIN

table1 table2

table1 table2

RIGHT JOIN

FULL OUTER JOIN

table1 table2

table1 table2

Source: https://www.w3schools.com/sql/sql_join.asp

03

SQL Select Aggregates

- Generally a select statement returns a list of tuples that satisfy your query, sometimes we want to aggregate (combine) them
  - count
  - max
  - min
  - string_agg() → may be helpful for Ass1 👀
- e.g.

```
select   count(...)
from     TableName
where    condition
```
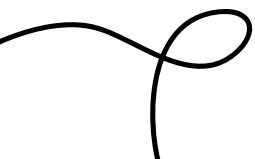
SQL Deletion Statements

04 → **SQL Deletion Statements**

- What happens when we delete something which other things refer to?
    - **Default behaviour**: Disallow the delete
    - **On Delete Cascade**: Delete all things that refer to it
    - **On Delete Set Default**: Set removed ID's to some default value

10. Consider the deletion of a department from a database based on this schema. What are the options for dealing with referential integrity between `Departments` and `WorksIn`? For each option, describe the required behaviour in SQL.

Referential integrity: Every reference to something (foreign key) must actually refer to something that exists.

The 3 approaches to dealing with referential integrity between Departments and WorksIn:

1. **Don't allow the deletion** of a Departments tuple if any WorksIn tuple refers to it. This is the **default** that results from the 'create table' definition.
2. When a Departments tuple is deleted, also **delete all WorksIn tuples that refer to it**. This requires adding an **'on delete cascade'** to the definition of WorksIn.

3. For every WorksIn tuple that refers to the deleted department, **set the dId field to the department id** of some existing 'default' department.

```
create table WorksIn (
    eid         integer,
    did         integer default 1,
    foreign key (eid) references Employees(id),
    foreign key (did) references Departments(did) on delete set default
    primary key (eid, did)
```

```
create table WorksIn (
    eid      integer,
    did      integer,
    percent real,
    primary key (eid,did),
    foreign key (eid) references Employees(eid) on delete cascade,
    foreign key (did) references Departments(did) on delete cascade
);
```

04 → SQL Deletion Statements Question

11. For each of the possible cases in the previous question, show how deletion of the Engineering department would affect the following database:

```
EID ENAME              AGE    SALARY
----- ---------------- ----- ----------
    1 John Smith        26     25000
    2 Jane Doe          40     55000
    3 Jack Jones        55     35000
    4 Superman          35     90000
    5 Jim James         20     20000

DID DNAME              BUDGET   MANAGER
----- ---------------- ---------- --------
    1 Sales             500000      2
    2 Engineering      1000000      4
    3 Service           200000      4

EID   DID  PCT_TIME
----- ----- ----------
    1   2     1.00
    2   1     1.00
    3   1     0.50
    3   3     0.50
    4   2     0.50
    4   3     0.50
    5   2     0.75
```

SQL Deletion Statements Answer

a. Disallow ... The database would not change. The DBMS would print an error message about referential integrity constraint violation.

b. ON DELETE CASCADE ... All of the tuples in the WorksIn relation that have did = 2 are removed, giving:

```
DID DNAME                BUDGET  MANAGER
----- ---------------- ----------- ---------
  1 Sales                 500000      2
  3 Service               200000      4


EID   DID  PCT_TIME
----- ----- ----------
  2     1      1.00
  3     1      0.50
  3     3      0.50
  4     3      0.50
```

c. ON DELETE SET NULL ... All of the tuples in the WorksIn relation that have did = 2 have that attribute modified to NULL, giving:

```
DID DNAME                BUDGET  MANAGER
----- ---------------- ----------- ---------
  1 Sales                 500000      2
  3 Service               200000      4


EID    DID  PCT_TIME
-----  ----- ----------
  1   NULL     1.00
  2     1      1.00
  3     1      0.50
  3     3      0.50
  4   NULL     0.50
  4     3      0.50
  5   NULL     0.75
```

d. ON DELETE SET DEFAULT ... All of the tuples in the WorksIn relation that have did = 2 have that attribute modified to the default department (1), giving:

```
DID DNAME                BUDGET  MANAGER
----- ---------------- ----------- ---------
  1 Sales                 500000      2
  3 Service               200000      4


EID    DID  PCT_TIME
-----  ----- ----------
  1     1      1.00
  2     1      1.00
  3     1      0.50
  3     3      0.50
  4     1      0.50
  4     3      0.50
  5     1      0.75
```
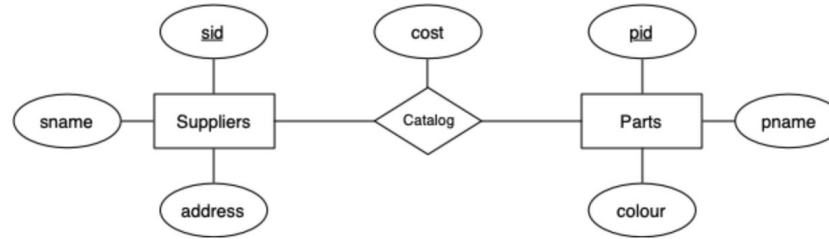
05

Putting it all to practice...!

05 → Putting it all to practice...!

Consider the following data model for a a business that supplies various parts:

Based on the ER design and the above considerations, here is a relational schema to represent this scenario:

```
create table Suppliers (
    sid     integer primary key,
    sname   text,
    address text
);
create table Parts (
    pid     integer primary key,
    pname   text,
    colour  text
);
create table Catalog (
    sid     integer references Suppliers(sid),
    pid     integer references Parts(pid),
    cost    real,
    primary key (sid,pid)
);
```

Write SQL statements to answer each of the following queries ...