

“Capítol 3: Introducció al SQL”

Fitxers i bases de dades

September 19, 2016

Capítol 3: Introducció al SQL

- Resum del llenguatge de cerca SQL
- Definició de les dades
- Estructura de cerca bàsica
- Funcions bàsiques addicionals
- Conjunt d'operacions
- Valors nuls
- Funcions agregades
- Subcerques aniuades
- Modificació de la base de dades

- El llenguatge “IBM Sequel”” va ser desenvolupat com a part del Sistema del projecte R al laboratori de recerca San José d'IBM
- Redefinit com “Structured Query Language” (SQL)
- ANSI i ISO estàndard SQL:
 - SQL-86, SQL-89, SQL-92
 - SQL:1999, SQL:2003, SQL:2008
- La majoria de sistemes comercials, si no tots, ofereixen l'aplicació SQL-92
 - Potser no tots els exemples d'aquí funcionen al teu ordinador

Llenguatge de Definició de les Dades

El **llenguatge de definició de les dades (DDL)** permet l'especificació d'informació sobre relacions, incloent:

- L'esquema per a cada relació
- El domini de valors associats amb cada atribut
- Restriccions íntegres
- I com es veurà posteriorment, altra informació com ara:
 - El conjunt d'índexs a mantenir per cada relació
 - Seguretat i autorització d'informació per a cada relació
 - L'estructura d'emmagatzematge físic per a cada relació al disc dur

Tipus de dominis en SQL

- `char(n)`. Longitud de cadena de caracters fixada, definida per l'usuari n
- `varchar(n)`. Variable cadena de caracters, amb una longitud definida per l'usuari màxima de n
- `int`. Enter
- `smallint`. Enter petit
- `numeric(p,d)` Número de punt flotant, amb precisió p dígets, amb n dígets a la dreta del decimal definits per l'usuari
- `real`, `double precision`. Punt flotant i de doble precisió
- `float(n)` Número de punt flotant com a mínim amb n dígets definits per l'usuari
- Alguns més inclosos al Capítol 4

Construcció Create Table

- Una relació d'SQL és definida usant el comandament **create table**

```
create table    $r(A_1D_1, A_2D_2, \dots, A_nD_n,$   
                 $(restricció_1),$   
                 $\dots,$   
                 $(restricció_k))$ 
```

- r és el nom de la relació
 - Cada A_i és el nom d'un atribut en l'esquema de la relació r
 - D_i és el tipus de dada de valor del domini de l'atribut A_i
- Exemple:

```
create table  instructor  (  
    ID          char(5),  
    name        varchar(20) not null,  
    dept_name   varchar(20)  
    salary     numeric(8,2))
```

- Insert into** *instructor* **values** ('10211', 'Smith', 'Biology', 66000);
- Insert into** *instructor* **values** ('10211', 'null', 'Biology', 66000);

Restriccions íntegres en un Create Table

- **not null**
- **primary key** (A_1, \dots, A_n)
- **foreign key** (A_m, \dots, A_n) **references** r

Exemple: Declarar *dept_name* com a clau primària per *department*

```
create table instructor (  
    ID char(5),  
    name varchar(20) not null,  
    dept_name varchar(20)  
    salary numeric(8,2),  
    primary key (ID),  
    foreign key (dept_name) references department)
```

La declaració de **primary key** a un atribut automàticament assegura valors **not null**

Algunes definicions relacionals més

- **create table** *student* (

<i>ID</i>	varchar (5),
<i>name</i>	varchar (20) not null ,
<i>dept_name</i>	varchar (20)
<i>tot_cred</i>	numeric (3,0),
primary key	(<i>ID</i>),
foreign key	(<i>dept_name</i>) references <i>department</i>));

- **create table** *takes* (

<i>ID</i>	varchar (5),
<i>couse_id</i>	varchar (8),
<i>sec_id</i>	varchar (8),
<i>semester</i>	varchar (6),
<i>year</i>	numeric (4,0),
<i>grade</i>	varchar (2),
primary key	(<i>ID</i> , <i>course_id</i> , <i>sec_id</i> , <i>semester</i> , <i>year</i>),
foreign key	(<i>ID</i>) references <i>student</i> ,
foreign key	(<i>course_id</i> , <i>sec_id</i> , <i>semester</i> , <i>year</i>) references <i>section</i>);

- **create table** *course* (
 course_id **varchar(8) primary key,**
 title **varchar50),**
 dept_name **varchar(20)**
 credits **numeric(2,0),**
 foreign key (*dept_name*) **references department**);
- La declaració de clau primària pot ser combinada amb declaracions d'atributs com s'acaba de mostrar

Deixar i modificar construccions de taules

- **drop table** *student*
 - Elimina la taula i els seus continguts
- **delete from** *student*
 - Elimina tots els continguts de la taula però no aquesta
- **alter table**
 - **alter table** *r* **add** A D
 - On A és el nom de l'atribut a afegir a la relació *r* i D és el domini d'A.
 - S'assigna *null* a totes les tuples en la relació com a valor pel nou atribut.
 - **alter table** *r* **drop** A
 - On A és el nom de l'atribut de relació *r*.
 - Eliminant els atributs no suportats per diverses bases de dades

Estructura Bàsica de Consulta

- El **llenguatge de manipulació de dades (DML)** proporciona l'habilitat de cercar informació, a més d'insertar, borrar i actualitzar tuples
- La consulta típica d'SQL té la forma:

select A_1, A_2, \dots, A_n

from r_1, r_2, \dots, r_m

where P

- A_i representa un atribut
 - R_i representa una relació
 - P és un predicat
- El resultat d'una cerca d'SQL és una relació.

La condició select

- La condició **select** llista els atributs desitjats en el resultat d'una consulta
- Exemple: trobar els noms de tots els instructors:

select *name*

from *instructor*

- NOTA: SQL no discrimina majúscules i minúscules.
 - És a dir, Name \equiv NAME \equiv name
 - Algunes persones fan servir majúscules on es fa servir negreta.

la condició Select (Cont.)

- SQL permet duplicats en relacions així com en resultats de les consultes.
- Per forçar l'eliminació de duplicats, insertar la paraula **distinct** seguit del **select**.
- Trobar els noms de tots els departaments amb instructor, i eliminar duplicats:

```
select distinct dept_name  
from instructor
```

- La paraula **all** especifica que no s'eliminaran els duplicats:

```
select all dept_name  
from instructor
```

la condició Select (Cont.)

- Un asterisc a la condició select indica “tots els atributs”.

```
select *  
  
from instructor
```

- La condició **select** pot contenir expressions aritmètiques tals com els operadors $+$, $-$, $*$ i $/$, operant en constants o atributs de tuples.
- La consulta:

```
select ID, name, salary/12  
  
from instructor
```

Retornaria una relació igual a la relació *instructor* excepte que el valor de l'atribut *salary* seria dividida per 12.

la condició where

- La clàusula **where** especifica les condicions que el resultat ha de satisfer
 - Correspon a la selecció d'atributs de l'àlgebra relacional.
- Per trobar tots els instructors en Comp. Sci. dept amb salary > 80000:

select *name*

from *instructor*

where *dept_name* = 'Comp. Sci.' **and** *salary* > 80000

- La comparació de resultats pot ser combinada fent servir els operadors lògics **and**, **or** i **not**.
- Les comparacions poden ser aplicades a resultats d'expressions aritmètiques.

- La clàusula **from** llista les relacions implicades en la consulta
 - Correspon a l'operació del producte Cartesià de l'àlgebra relacional.
- Per trobar el producte Cartesià *instructor X teaches*:

select *

from *instructor, teaches*

- Genera tots els parells possibles instructor-teacher amb tots els atributs d'ambdues relacions
- El producte Cartesià directament no és molt útil, però sí combinat amb la clàusula **where** (selecció d'operacions en l'àlgebra relacional)

Producte Cartesià: *instructor X teaches*

<i>instructor</i>				<i>teaches</i>				
<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
15151	Mozart	Music	40000	10101	CS-347	1	Fall	2009
22222	Einstein	Physics	95000	12121	FIN-201	1	Spring	2010
32343	El Said	History	60000	15151	MU-199	1	Spring	2010
...	22222	PHY-101	1	Fall	2009

<i>inst.ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>teaches.ID</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	10101	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	10101	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	10101	CS-347	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	12121	FIN-201	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	15151	MU-199	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	22222	PHY-101	1	Fall	2009
...
...
12121	Wu	Finance	90000	10101	CS-101	1	Fall	2009
12121	Wu	Finance	90000	10101	CS-315	1	Spring	2010
12121	Wu	Finance	90000	10101	CS-347	1	Fall	2009
12121	Wu	Finance	90000	12121	FIN-201	1	Spring	2010
12121	Wu	Finance	90000	15151	MU-199	1	Spring	2010
12121	Wu	Finance	90000	22222	PHY-101	1	Fall	2009
...
...

Figure: Exemple de producte cartesià.

Unions

- Per a tots els instructors que han ensenyat algun curs, trobar els seus noms i l'identificador del curs que han donat *instructor X teaches*:

```
select name, course_id
from instructor, teaches
where instructor.ID = teaches.ID
```

- Trobar l'identificador del curs, semestre, any i títol de cada curs que ofereix el departament de Comp. Sci:

```
select section.course_id, semester, year, title
from section, course
where section.course_id = course.course_id and
dept_name = 'Comp. Sci.'
```



Intenteu escriure algunes consultes en SQL

- Escribiu algunes consultes per fer...

Unió Natural

- La Unió natural enganxa tuples amb els mateixos valors per a tots els atributs comuns, i només conserva una còpia per a cada columna comuna
- select ***
from instructor natural join teaches;

ID	name	dept_name	salary	course_id	sec_id	semester	year
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010

Figure: Exemple d'unió natural.

Exemple Unió Natural

- Llistar els noms dels instructors juntament amb l'identificador de curs que han ensenyat
 - **select** *name, course_id*
from *instructor, teaches*
where *instructor.ID = teaches.ID;*
 - **select** *name, course_id*
from *instructor natural join teaches;*

Natural Join (Cont.)

- Perill en una unió natural: atenció amb els atributs no relacionats amb el mateix nom però incorrectament equivalents
- Llistar els noms dels instructors amb els títols dels cursos que han donat

- Versió incorrecte (makes `course.dept_name = instructor.dept_name`)

```
select name, title
```

```
from instructor natural join teaches natural join course;
```

- Versió correcte

```
select name, title
```

```
from instructor natural join teaches, course
```

```
where teaches.course_id = course.course_id;
```

- Una altra versió correcte

```
select name, title
```

```
from (instructor natural join teaches)
```

```
join course using (course_id);
```

L'operació Rebatejar

- SQL permet rebatejar relacions i atributs utilitzant l'expressió **as**:

nom-antic as nom-nou

- Exemple:

- **select** *ID, name, salary/12 as monthly_salary*
from *instructor*

- Trobar el nom de tots els instructors que tinguin un salary més elevat que qualsevol instructor del departament 'Comp. Sci.'.

- **select distinct** *T.name*
from *instructor as T instructor as S*
where *T.salary > S.salary and S.dept_name = 'Comp. Sci.'*

- La paraula **as** és opcional i es pot ometre

instructor as T \equiv instructor T

- La paraula **as** pot ser eliminada en Oracle

Operacions cadena de caracters

- SQL inclou l'opció de combinar operadors de caracters per comparar cadenes de caracters. L'operador "like" utilitza patrons que són descrits utilitzant dos caracters especials:
 - Percentatge (%). El % combina qualsevol subcadena de caracters.
 - Guionet baix (_). El _ combina qualsevol caracter.
- Trobar els noms de tots els instructors els quals els seus noms incloguin la subcadena "dar".

```
select name  
from instructor  
where name like '%dar%'
```

- Combinar la cadena "100 %"
like '100 \% ' **escape** '\'

Operacions cadena de caracters (Cont.)

- Els patrons discriminen majúscules i minúscules.
- Exemple de combinació de patrons:
 - 'Intro%' combina qualsevol cadena que comenci per "Intro".
 - '%Comp%' combina qualsevol cadena que contingui "Comp" com a subcadena.
 - ' _ _ ' combina qualsevol cadena d'exactament 3 caracters.
 - ' _ _ _ %' combina qualsevol cadena d'almenys 3 caracters.
- SQL suporta una varietat d'operacions amb cadenes tals com
 - Concatenació (fent servir "||")
 - Convertir majúscula en minúscula i a l'inversa
 - Trobar el tamany d'una cadena, extreure subcadenaes, etc...

Ordenant la visualització de les Tuples

- Llistar en ordre alfabètic el nom de tots els instructors

select distinct *name*

from *instructor*

order by *name*

- Podem especificar **desc** per ordenar de manera descendent i **asc** per ascendent; aquesta última és per defecte.
 - Exemple: **order by** *name* **desc**
- Es pot ordenar per múltiples atributs
 - Exemple: **order by** *dept_name, name*

Opció amb la condició Where

- SQL inclou l'operador comparador **between**
- Exemple: Trobar els noms de tots els instructors amb salari entre 90.000\$ i 100.000\$ (és a dir, $\geq 90.000\$$ i $\leq 100.000\$$)

```
select name  
from instructor  
where salary between 90000 and 100000
```

- Comparant Tuples

```
select name, course_id  
from instructor, teaches  
where (instructor.ID, dept_name) = (teaches.ID, 'Biology');
```

Duplicats

- A les relacions amb duplicats, SQL pot definir quantes còpies de tuples han d'aparèixer al resultat.
- Versions **multiset** d'alguns operadors d'àlgebra relacionals - donades relacions r_1 i r_2 :
 - 1 $\sigma_\theta(r_1)$: Si hi ha c_1 còpies de tuples t_1 a r_1 i t_1 satisfà la selecció σ_θ , aleshores hi ha c_1 còpies de t_1 en $\sigma_\theta(r_1)$.
 - 2 $\Pi_a(r)$: Per cada còpia de tuple t_1 a r_1 , hi ha una còpia de tuple $\Pi_a(t_1)$ a $\Pi_a(r_1)$ on $\Pi_a(t_1)$ denota la projecció de l'única tuple t_1 .
 - 3 $r_1 \times r_2$: Si hi ha c_1 còpies de tuples t_1 a r_1 i c_2 còpies de tuples t_2 a r_2 , hi ha $c_1 \times c_2$ còpies de tuples $t_1 \cdot t_2$ a $r_1 \times r_2$

Duplicats (Cont.)

- Exemple: Suposem un conjunt múltiple de relacions r_1 (A,B) i r_2 (C) següents:

$$r_1 = \{(1, a), (2, a)\} \quad r_2 = \{(2), (3), (3)\}$$

- Aleshores $\Pi_B(r_1)$ seria $\{(a), (a)\}$, mentre $\Pi_B(r_1) \times r_2$ seria

$$\{(a, 2), (a, 3), (a, 3)\}$$

- SQL duplica semàntica:

```
select distinct  $A_1, A_2, \dots, A_n$   
from  $r_1, r_2, \dots, r_m$   
where  $P$ 
```

és equivalent a la versió multiconjunt de l'expressió:

$$\Pi_{A_1, A_2, \dots, A_n}(\sigma_P(r_1 \times r_2 \times \dots \times r_m))$$

Conjunt d'operacions

- Trobar cursos que es van impartir la tardor del 2009 **o** primavera del 2010
(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
union
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)
- Trobar cursos que es van impartir la tardor del 2009 **i** primavera del 2010
(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
intersect
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)
- Trobar cursos que es van impartir la tardor del 2009 **però no** primavera del 2010
(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2009)
except
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2010)

Conjunt d'operacions

- Conjunt d'operacions **union**, **intersect** i **except**
 - Cadascuna de les operacions anteriors **eliminen automàticament els duplicats**
- Per **conservar tots els duplicats** s'han de fer servir les corresponents versions multiconjunts **union all**, **intersect all** i **except all**

Suposem que una tuple passa m vegades en r i n en s . Aleshores passa:

- $m + n$ vegades en r **union all** s
- $\min(m, n)$ vegades en r **intersect all** s
- $\max(0, m - n)$ vegades en r **except all** s

Null values

- És possible l'existència de tuples amb valor nul, denotat per *null*, per alguns atributs.
- *null* significa un valor desconegut o inexistent.
- Els resultats d'operacions aritmètiques que continguin l'expressió *null* és *null*
 - Exemple: $5 + \text{null}$ retorna null
- L'afirmació **is null** es pot fer servir per comprovar valors nuls.
 - Exemple: Trobar tots els instructors els quals el seu salari sigui nul.

```
select name
from instructor
where salary is null
```


Null values i tres valors lògics

- Qualsevol comparació amb *null* retorna *unknown*.
 - Exemple: $5 < null$ o $null <> null$ o $null = null$
- Tres valors lògics fent servir el valor veritats *unknown*:
 - **OR**: $(unknown \text{ or } true) = true$,
 $(unknown \text{ or } false) = unknown$,
 $(unknown \text{ or } unknown) = unknown$,
 - **AND**: $(true \text{ and } unknown) = unknown$,
 $(false \text{ and } unknown) = false$,
 $(unknown \text{ and } unknown) = unknown$,
 - **NOT**: $(not \ unknown) = unknown$
 - “*P* is unknown” evalua a true si el predicat *P* evalua a unknown
- El resultat de l’afirmació **where** és tractat com a *false* si evalua a *unknown*

Funcions agregades

- Aquestes funcions operen sobre un conjunt de valors de relació per columna i retornen un valor

avg: mitjana del valor

min: valor mínim

max: valor màxim

sum: suma dels valors

count: nombre de valors

Funcions agregades (Cont.)

- Trobar la mitjana del salari dels instructors del departament de Computer Science
 - **select avg** (*salary*)
from *instructor*
where *dept_name* = 'Comp. Sci.';
- Trobar el nombre total d'instructors que van donar classes el semestre de primavera del 2010
 - **select count (distinct ID)**
from *teaches*
where *semester* = 'Spring' **and** *year* = 2010;
- Trobar el nombre de tuples a la relació *course*
 - **select count (*)**
from *course*;

Funcions agregades - Group By

- Trobar la mitjana del salari dels instructors de cada departament
 - `select dept_name, avg (salary)`
`from instructor`
`group by dept_name;`
- Nota: Els departaments sense instructors no apareixeran al resultat

ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

dept_name	avg_salary
Biology	72000
Comp. Sci.	77333
Elec. Eng.	80000
Finance	85000
History	61000
Music	40000
Physics	91000

Figure: Exemple de funcions agregades - Group By.

- Attributes in **select** clause outside of aggregate functions must appear in **group by** list
 - /* erroneous query */
select *dept_name*, *ID*, **avg** (*salary*)
from *instructor*
group by *dept_name*;

Funcions agregades - Condició Having

- Trobar el nom i la mitjana del salari de tots els departaments els quals la mitjana del salari sigui més gran de 42000

```
select dept_name, avg salary
from instructor
group by dept_name
having avg (salary) > 42000;
```

Nota: Predicates in the **having** clause are applied after the formation of groups whereas predicates in the **where** clause are applied before forming groups.

Null Values and Aggregates

- Trobar el total de tots els salaris

```
select sum (salary)
from instructor
```

- Els arguments anteriors ignoren les quantitats nul·les
- El resultat és *null* si no hi ha quantitats no nul·les
- Totes les operacions agregades excepte **count(*)** ignoren tuples amb valors nuls als atributs agregats
- Què passa si només hi ha valors nuls?
 - Count retorna 0
 - Totes les altres agregacions retornen *null*

Subconsultes aniuades

- SQL proporciona un mecanisme per fer subconsultes aniuades.
- Una **subquery** és una expressió **select-from-where** que és aniuada dins d'una altra consulta.
- Un ús habitual de subconsultes és portar a terme tests per conjunts de membres, conjunt de comparacions and set cardinality.

Exemple Query

- Trobar els cursos oferts a la tardor del 2009 i primavera del 2010

```
select distinct course_id
from section
where semester = 'Fall' and year = 2009 and
       course_id in (select course_id
                       from section
                       where semester = 'Spring' and year = 2010);
```

- Trobar els cursos oferts a la tardor del 2009 però no a la primavera del 2010

```
select distinct course_id
from section
where semester = 'Fall' and year = 2009 and
       course_id not in (select course_id
                              from section
                              where semester = 'Spring' and year = 2010);
```

Exemple Query

- Trobar el nombre total (**distinct**) d'estudiants els quals el seu instructor hagi sigut el *ID* 10101

```
select count (distinct ID)  
from takes  
where (course_id, sec_id, semester, year) in  
      (select course_id, sec_id, semester, year  
       from teaches  
       where teaches.ID = 10101);
```

- **Nota:** La consulta anterior pot ser escrita d'una manera molt més simple. Només es posa l'exemple per il·lustrar funcions de l'SQL.

Set Comparison

- Trobar els noms dels instructors amb salari més gran que algun instructor (almenys un) del departament de biologia

```
select distinct T.name)  
from instructor as T, instructor as S  
where T.salary > S.salary and S.dept_name = 'Biology';
```

- La mateixa consulta fent servir `>` la condició **some**

```
(select name  
from instructor  
where salary > some (select salary  
                        from instructor  
                        where dept_name = 'Biology');
```

Definició d'algunes clàusula

- $F < comp > \text{some } r \Leftrightarrow \exists t \in r \text{ tal que } (F < comp > t)$

On $< comp >$ **pot ser**: $<, \leq, >, =, \neq$

$(5 < \text{some } \{0, 5, 6\}) = \text{true}$. (read: $5 <$ alguna tupla en la relació)

$(5 < \text{some } \{0, 5\}) = \text{false}$

$(5 = \text{some } \{0, 5\}) = \text{true}$

$(5 \neq \text{some } \{0, 5\}) = \text{true}$ (*since* $0 \neq 5$)

$(=\text{some}) = \text{in}$

però, $(\neq \text{some}) \neq \text{not in}$

Exemple Query

- Trobar els noms dels instructors amb salari més gran que tots els instructors del departament de biologia

```
select name)  
from instructor  
where salary > all (select salary  
                        from instructor  
                        where dept_name = 'Biology');
```

Definició de tota clàusula

- $F < comp > \mathbf{all} \ r \Leftrightarrow \forall t \in r \ (F < comp > t)$
 $(5 < \mathbf{all} \ \{0, 5, 6\}) = \text{false}$
 $(5 < \mathbf{all} \ \{6, 10\}) = \text{true}$
 $(5 = \mathbf{all} \ \{4, 5\}) = \text{false}$
 $(5 \neq \mathbf{all} \ \{4, 6\}) = \text{true} \ (\text{since } 5 \neq 4 \text{ and } 5 \neq 6)$
 $(\neq \mathbf{all}) = \mathbf{not \ in}$
però, $(= \mathbf{all}) \neq \mathbf{in}$

Test per relacions buides

- El constructe **exist** retorna el valor **true** si l'argument de la subconsulta no és buit.
- **exists** $r \Leftrightarrow r \neq \emptyset$
- **not exists** $r \Leftrightarrow r = \emptyset$

- Una altra manera per fer la consulta anterior: “Trobar els cursos fets a la tardor del 2009 i primavera del 2010”

```
select course_id
from section as S
where semester = 'Fall' and year = 2009 and
      exists (select *
              from section as T
              where semester = 'Spring' and year = 2010
                  and S.course_id = T.course_id);
```

- Subconsulta correlacionada
- Correlation name or correlation variable

- Trobar tots els estudiants que han rebut tots els cursos oferts pel departament de Biologia

```
select distinct S.ID, S.name  
from student as S  
where not exists ((select course_id  
                    from course  
                    where dept_name = 'Biology')  
                  except  
                  (select T.course_id  
                    from takes as T  
                    where S.ID = T.ID));
```

- Noteu que $X - Y = \emptyset \Leftrightarrow X \subseteq Y$
- *Nota:* Aquesta consulta no es pot escriure fent servir = **all** i les seves variants

Test per absència de duplicats de tuples

- The **unique** construct tests whether a subquery has any duplicate tuples in its result.
 - (Evaluates to “true” on an empty set)
- Trobar tots els cursos que han estat impartits almenys una vegada al 2009

```
select T.course_id
from course as T
where unique (select R.course_id
                  from section as R
                  where T.course_id = R.course_id
                      and R.year = 2009);
```

Subqueries in the From Clause

- SQL permet una expressió de **subconsulta** utilitzada a la condició **from**
- Trobar la mitjana del salari dels instructors dels departaments on la mitjana del salary és més gran que 42000\$

```
select dept_name, avg_salary
from (select dept_name, avg (salary) as avg_salary
      from instructor
      group by dept_name)
where avg_salary > 42000;
```

- Noteu que no s'ha necessitat la condició **having**
- Una altra manera d'escriure la consulta anterior

```
select dept_name, avg_salary
from (select dept_name, avg (salary)
      from instructor
      group by dept_name)
as dept_avg (dept_name, avg_salary)
where avg_salary > 42000;
```

Subqueries in the From Clause (Cont.)

- I encara una altra manera d'escriure-la usant la condició **lateral**

```
select name, salary, avg_salary
from instructor1,
      lateral (select avg (salary) as avg_salary
               from instructor2
               where i2.dept_name = i1.dept_name);
```

- Lateral clause permits later part of the **from** clause (after the lateral keyword) to access correlation variables from the earlier part.
- Note: lateral is a part of the SQL standard, but is not supported on many database systems; some databases such as SQL Server offer alternative syntax.
- Note: Això no funciona sempre!

Condicíó With

- La condició **with** proporciona una manera de definir temporalment on les seves definicions són disponibles només en la consulta on la condició **with** apareix
- Trobar tots els departaments amb el pressupost màxim

```
with max_budget(value) as  
    (select max (budget)  
     from department)  
select budget  
from department, max_budget  
where department.budget = max_budget.value
```

Consultes complexes fent servir la condició With

- La condició With és molt útil per fer queries complexes
- Suportat per la majoria de sistemes de base de dades, amb variacions de sintaxi menors
- Trobar tots els departaments on el total del salari és mes gran que la mitjana total del salari de tots els departaments

```
with dept_total(dept_name, value) as
  (select dept_name, sum(salary)
   from instructor
   group by dept_name),
dept_total_avg(value) as
  (select avg(value)
   from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value >= dept_total_avg.value;
```

Scalar Subquery

- **Scalar subquery** is one which is used where a single value is expected
- Ex.:

```
select dept_name,  
       (select count(*)  
        from instructor  
        where department.dept_name = instructor.dept_name)  
as num_instructors  
from department;
```
- Ex.:

```
select name,  
       from instructor  
where salary * 10 >  
       (select budget from department  
        where department.dept_name = instructor.dept_name);
```
- Runtime error if subquery returns more than one result tuple

Modificació de la base de dades

- **Deletion** of tuples from a given relation
- **Inserció** de noves tuples en una relació donada
- **Actualitzant valors** d'algunes tuples en una relació donada

Modificació de la base de dades - Eliminació

- Borrar **tots** els instructors

```
delete from instructor
```

- Borrar **tots** els instructors del departament de **finances**

```
delete from instructor
```

```
where dept_name = 'Finance';
```

- Borrar totes les tuples a la relació *instructor* per aquells instructors associats amb el departament localitzat a l'edifici Watson

```
delete from instructor
```

```
where dept_name in (select dept_name
```

```
from department
```

```
where building = 'Watson');
```

- Borrar tots els instructors els quals el seu salari sigui menor que la mitjana

delete from *instructor*

where *salary* < (**select avg** (*salary*) **from** *instructor*)

- Problema: En el moment en què es borren tuples, la mitjana del salari canvia
- Solució utilitzada en SQL:
 - 1 Primer, calcular la mitjana del salari **avg** i trobar totes les tuples a borrar
 - 2 Després, borrar totes les tuples trobades (sense recalculer **avg** o repetir les tuples)

Modificació de la base de dades - Inserció

- **Afegir** una nova tuple a *course*

```
insert into course
```

```
values ('CS-437','Database Systems', 'Comp. Sci.', 4)
```

- O de manera equivalent

```
insert into course (course_id, title, dept_name, credits)
```

```
values ('CS-437','Database Systems', 'Comp. Sci.', 4)
```

- Afegir una nova tuple a *student* amb *tot_creds* amb valor nul

```
insert into student
```

```
values ('3003','Green', 'Finance', null)
```

- Afegir tots els instructors a la relació *student* amb tot_creds fixats a 0

insert into *student*

select ID, name, dept_name, 0

from instructor

- La declaració **select from where** és evaluada totalment abans any of its results are inserted into the relation (otherwise queries like

insert into table1 select * from table1

would cause problems, if *table1* did not have any primary key defined).

Modificació de la base de dades - Actualització

- Augmentar els salaris dels instructors els quals el seu salari sigui més de 100.000\$ un 3%, a tota la resta incrementar-los en un 5%

- Cal escriure dues declaracions a **actualitzar**:

```
update instructor
```

```
  set salary = salary * 1.03
```

```
  where salary > 100000;
```

```
update instructor
```

```
  set salary = salary * 1.05
```

```
  where salary <= 100000;
```

- L'ordre és important
- Es pot fer millor fent servir la instrucció **case** (pròxima transparència)

Declaració Case o Actualitzacions Condicionades

- Una consulta similar però amb la condició **case**:

```
update instructor
```

```
  set salary = case
```

```
    when salary <= 100000 then salary * 1.05
```

```
    else salary * 1.03
```

```
  end
```

Actualitzacions amb subconsultes escalars

- Recalcular i actualitzar el valor de `tot_creds` per tots els estudiants

update *student* as *S*

set *tot_cred* = (**select** **sum**(*credits*)
from *takes* **natural join** *course*
where *S.ID* = *takes.ID* **and**
takes.grades <> 'F' **and**
takes.grades **is not null**);

- Assigna null *tot_creds* als estudiants que no han fet cap curs

- En comptes de **sum** (*credits*), fer servir:

case *student S*

when **sum** (*credits*) = **is not null** **then** **sum**(*credits*)
else 0

end

FINAL DEL CAPÍTOL 3

- Create a table with the same schema as an existing table:
create table temp_account **like** account

Figures

<i>name</i>
Srinivasan
Wu
Mozart
Einstein
El Said
Gold
Katz
Califieri
Singh
Crick
Brandt
Kim

Figure: Exemple.

Figures

<i>dept_name</i>
Comp. Sci.
Finance
Music
Physics
History
Physics
Comp. Sci.
History
Finance
Biology
Comp. Sci.
Elec. Eng.

Figure: Exemple.

Figures

<i>name</i>
Katz
Brandt

Figure: Exemple.

Figures

<i>name</i>	<i>dept_name</i>	<i>building</i>
Srinivasan	Comp. Sci.	Taylor
Wu	Finance	Painter
Mozart	Music	Packard
Einstein	Physics	Watson
El Said	History	Painter
Gold	Physics	Watson
Katz	Comp. Sci.	Taylor
Califieri	History	Painter
Singh	Finance	Painter
Crick	Biology	Watson
Brandt	Comp. Sci.	Taylor
Kim	Elec. Eng.	Taylor

Figure: Exemple.

Figures

<i>name</i>	<i>Course_id</i>
Srinivasan	CS-101
Srinivasan	CS-315
Srinivasan	CS-347
Wu	FIN-201
Mozart	MU-199
Einstein	PHY-101
El Said	HIS-351
Katz	CS-101
Katz	CS-319
Crick	BIO-101
Crick	BIO-301
Brandt	CS-190
Brandt	CS-190
Brandt	CS-319
Kim	EE-181

Figure: Exemple.

Figures

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>	<i>course_id</i>	<i>sec_id</i>	<i>semester</i>	<i>year</i>
10101	Srinivasan	Comp. Sci.	65000	CS-101	1	Fall	2009
10101	Srinivasan	Comp. Sci.	65000	CS-315	1	Spring	2010
10101	Srinivasan	Comp. Sci.	65000	CS-347	1	Fall	2009
12121	Wu	Finance	90000	FIN-201	1	Spring	2010
15151	Mozart	Music	40000	MU-199	1	Spring	2010
22222	Einstein	Physics	95000	PHY-101	1	Fall	2009
32343	El Said	History	60000	HIS-351	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-101	1	Spring	2010
45565	Katz	Comp. Sci.	75000	CS-319	1	Spring	2010
76766	Crick	Biology	72000	BIO-101	1	Summer	2009
76766	Crick	Biology	72000	BIO-301	1	Summer	2010
83821	Brandt	Comp. Sci.	92000	CS-190	1	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-190	2	Spring	2009
83821	Brandt	Comp. Sci.	92000	CS-319	2	Spring	2010
98345	Kim	Elec. Eng.	80000	EE-181	1	Spring	2009

Figure: Exemple.

Figures

<i>course_id</i>
CS-101
CS-347
PHY-101

Figure: Exemple.

Figures

<i>course_id</i>
CS-101
CS-315
CS-319
CS-319
FIN-201
HIS-351
MU-199

Figure: Exemple.

Figures

<i>course_id</i>
CS-101
CS-315
CS-319
CS-347
FIN-201
HIS-351
MU-199
PHY-101

Figure: Exemple.

Figures

<i>course_id</i>
CS-101

Figure: Exemple.

Figures

<i>course_id</i>
CS-347
PHY-101

Figure: Exemple.

Figures

<i>dept_name</i>	<i>count</i>
Comp. Sci.	3
Finance	1
History	1
Music	1

Figure: Exemple.

Figures

<i>dept_name</i>	<i>avg(salary)</i>
Physics	91000
Elec. Eng.	80000
Finance	85000
Comp. Sci.	77333
Biology	72000
History	61000

Figure: Exemple.