UAlg FCT

UNIVERSIDADE DO ALGARVE
FACULDADE DE CIÊNCIAS E TECNOLOGIA

Bases de Dados

INTRODUÇÃO AO SQL (PARTE 2)

Predicados da cláusula Where

- SQL inclui um operador de comparação **between**
- Exemplo: Encontre os nomes de todos os instrutores com salário entre 90.000 e 100.000 (ou seja, $\geq$ 90.000 e $\leq$ 100.000)
  - **select** *name*
    from *instructor*
    where *salary* **between** 90000 **and** 100000
- Comparação de tuplos
  - **select** *name*, *course_id*
    from *instructor*, *teaches*
    where (*instructor.ID*, *dept_name*) = (*teaches.ID*, 'Biology');

Operações de conjuntos

- Encontre as disciplinas realizadas em ‘Fall’ de 2017 ou ‘Spring’ de 2018

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2017)
union
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2018)

- Encontre as disciplinas realizadas em ‘Fall’ de 2017 e em ‘Spring’ de 2018

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2017)
intersect
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2018)

- Encontre as disciplinas realizadas em ‘Fall’ de 2017 que não ocorreram em ‘Spring’ de 2018

(**select** *course_id* **from** *section* **where** *sem* = 'Fall' **and** *year* = 2017)
except
(**select** *course_id* **from** *section* **where** *sem* = 'Spring' **and** *year* = 2018)

Operações de conjuntos(Cont.)

- As operações **union , intersect e except eliminam automaticamente os duplicados**
- Para manter todos os duplicados:
  - **union all**
  - **intersect all**
  - **except all**

Valores Nulos

- É possível existitem valores nulos, denotados por **null**, em alguns atributos
- **null** significa um valor desconhecido ou um valor que não existe.
- O resultado de qualquer expressão aritmética envolvendo **null** é **null**
  - Exemplo: 5 + **null** retorna **null**

O predicado **is null** pode ser usado para verificar valores nulos.

- Exemplo: Encontre todos os instrutores cujo salário é desconhecido .

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

- O predicado **is not null** terá êxito se o valor ao qual é aplicado não for nulo.

Valores Nulos (Cont.)

- O SQL trata como ***unknown*** o resultado de qualquer comparação envolvendo um valor ***null*** (que não seja **is null** ou **is not null**).
  - Exemplo *: 5* < **null** ou **null** <> **null** ou **null** = **null**
- O predicado de uma cláusula **where** pode envolver operações booleanas (**and** , **or** , **not**); portanto, as definições das operações booleanas são estendidas para lidar com o valor ***unknown*** .
  - **and** : *(true* **and** *unknown) = unknown,*
    (false **and** *unknown) = false,*
    (unknown **and** *unknown) = unknown*
  - **or:** (*unknown* **or** *true*) = *true*,
    (*unknown* **or** *false*) = *unknown*
    (*unknown* **or** *unknown) = unknown*
- O resultado do predicado da cláusula **where** é tratado como *false* se for avaliado como *unknown*

Funções de agregação

- Operam no conjunto de valores de uma coluna e retornam um único valor

avg : valor médio
min: valor mínimo
max: valor máximo
sum: soma dos valores
count: número de valores

Funções de agregação (exemplos)

- Encontre o salário médio dos instrutores do departamento de ‘Comp. Sci.’
  - **select avg** (*salary*)
    from *instructor*
    where *dept_name*= 'Comp. Sci.';

- Encontre o número total de instrutores que ministram uma disciplina no semestre de ‘Spring’ de 2018
  - **select count** (**distinct** *ID*)
    from *teaches*
    where *semester* = 'Spring' **and** *year* = 2018;

- Encontre o número de tuplos na relação *course*
  - **select count** (*)
    from *course*;

Agregações- Group By

Encontre o salário médio dos instrutores de cada departamento

- **select** *dept_name*, **avg** (*salary*) **as** *avg_salary*
  from *instructor*
  group by *dept_name*;

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

Agregações- Group By (Cont.)

Atributos na cláusula **select** que não sejam funções de agregação têm de aparecer no **group by**

- /* consulta errada */
  select *dept_name*, *ID*, **avg** (*salary*)
  from *instructor*
  group by *dept_name*;

Agregações– Having

Encontre os nomes e salários médios de todos os departamentos cujo salário médio é superior a 42.000

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

Nota: os predicados na cláusula **having** são aplicados após a formação dos grupos, enquanto os predicados na cláusula **where** são aplicados antes da formação dos grupos

Subconsultas aninhadas (nested)

- SQL fornece um mecanismo para aninhamento de subconsultas. Uma **subconsulta** é uma expressão **select-from-where** que está aninhada noutra consulta.
- O aninhamento pode ser feito na seguinte consulta SQL

select A_1, A_2, ..., A_n
from r_1, r_2, ..., r_m
where P

do seguinte modo:

- **from:** r_i pode ser substituído por qualquer subconsulta válida
- **where:** P pode ser substituída por uma expressão no formato:

  B <operação> (subconsulta)

  B é um atributo e <operação> a ser definida posteriormente.
- **select:**

  A_i pode ser substituído por uma subconsulta que gera um único valor .

Avaliar se está contido num conjunto *(Set membership)*

Set membership

- Encontre disciplinas ministradas em ‘Fall’ de 2017 e ‘Spring’ de 2018

```
select distinct course_id
from section
where semester = 'Fall' and year= 2017 and
        course_id in (select course_id
                      from section
                      where semester = 'Spring' and year= 2018);
```

- Encontre disciplinas ministradas em ‘Fall’ de 2017, mas não em ‘Spring’ de 2018

```
select distinct course_id
from section
where semester = 'Fall' and year= 2017 and
        course_id  not in (select course_id
                           from section
                           where semester = 'Spring' and year= 2018);
```

Set membership (cont.)

- Apresente o nome de todos os instrutores cujo nome não seja "Mozart" nem Einstein"

```
select distinct name
from instructor
where  name not in ('Mozart', 'Einstein')
```

- Encontre o número total de alunos (distintos) que participaram em secções de disciplinas ministradas pelo instrutor com *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: A consulta acima pode ser escrita de uma maneira muito mais simples.
A formulação acima é simplesmente para ilustrar as funcionalidades do SQL

Comparação de conjuntos

Comparação de conjuntos – cláusula *some / any*

- Encontre nomes de instrutores com salário superior ao de algum (pelo menos um) instrutor do departamento de Biologia.

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

- Mesma consulta usando a cláusula > **some**

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

Definição da cláusula *some / any*

F <comp> **some** $r \Leftrightarrow \exists\ t \in r$ tal que (F <comp> t)
Onde <comp> pode ser: $<,\ \leq,\ >,\ =,\ \neq$

(5 < **some** | 0 | 5 | 6 |) = true

5 < algum tuplo na relação)

(5 < **some** | 0 | 5 |) = false

(5 = **some** | 0 | 5 |) = true

(5≠ **some** | 0 | 5 |) = true (pois 0 ≠ 5)

(= **some**) ≡ **in**
No entanto, (≠ **some**) ≢ **not in**

Comparação de conjuntos – Cláusula *all*

- Encontre os nomes de todos os instrutores cujo salário é maior que o salário de todos os instrutores do departamento de Biologia.

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

Definição da cláusula *all*

F <comp> **all** $r \Leftrightarrow \forall\ t \in r$ (F <comp> *t)*

(5 < **all** | 0 / 5 / 6) = false

(5 < **all** | 6 / 10) = true

(5 = **all** | 4 / 5) = false

(5≠ **all** | 4 / 6) = true (pois 5 ≠ 4 e 5 ≠ 6)

(≠ **all**) ≡ **not in**

No entanto, (= **all**) ≠ **in**

Teste para relações vazias

A cláusula **exists** retorna o valor verdadeiro se a subconsulta do argumento não estiver vazia.

exists $r \Leftrightarrow r \neq \emptyset$

not exists $r \Leftrightarrow r = \emptyset$

Uso da cláusula exists

- Mais uma forma de especificar a consulta "Encontre todos os cursos ministrados no semestre do ‘Fall’ de 2017 e no semestre da ‘Spring’ de 2018"

```
select course_id
from section as S
where semester = 'Fall' and year = 2017 and
        exists  (select *
                 from section as T
                 where semester = 'Spring' and year= 2018
                         and S.course_id = T.course_id);
```

Nome da correlação – variável S na consulta

Subconsulta correlacionada – a consulta *nested*

Uso da cláusula **not exists**

- Encontre todos os alunos que frequentaram todas as disciplinas oferecidas no departamento 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));
```

  - A primeira consulta aninhada lista todas as disciplinas oferecidas em Biologia
  - A segunda consulta aninhada lista todas as disciplinas que um aluno específico fez

Observe que $X - Y = \emptyset \Leftrightarrow X \subseteq Y$

Nota: Não é possível escrever esta consulta usando = all e suas variantes. Correponde ao operador divisão de álgebra relacional.

Teste para ausência de duplicados

- A cláusula **unique** testa se uma subconsulta possui duplicados no seu resultado.
- A cláusula **unique** é avaliada como "true" se uma determinada subconsulta não contiver duplicados.
- Encontre todos os cursos que foram oferecidos no máximo uma vez em 2017

```
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 = 2017);
```

Cláusula with

- A cláusula **with** fornece uma forma de definir uma relação temporária cuja definição está disponível apenas para a consulta na qual a cláusula **with** ocorre.

- Encontre todos os departamentos com o orçamento máximo

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

Consultas complexas com a cláusula With

Encontre todos os departamentos onde o salário total é maior que a média do salário total em todos os departamentos

```
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
```

Modificação da base de dados

- Eliminação de tuplos de uma determinada relação.
- Inserção de novos tuplos numa determinada relação
- Atualização de valores em algumas tuplos de uma determinada relação

Eliminação

- Excluir todos os instrutores

```
delete from instructor
```

- Excluir todos os instrutores do departamento de ‘Finance’

```
delete from instructor
where dept_name= 'Finance’;
```

- *Exclua todos os instrutores associados a um departamento localizado no edifício ‘Watson’.*

```
delete from instructor
where dept name in (select dept name
                    from department
                    where building = 'Watson');
```

Inserção

- Adicione um novo tuplo nas disciplinas

```
insert into course
        values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

- ou equivalente

```
insert into course (course_id, title, dept_name, credits)
        values ('CS-437', 'Database Systems', 'Comp. Sci.', 4);
```

Adicione um novo *aluno* com *tot_creds* definido como nulo

```
insert into student
    values ('3003', 'Green', 'Finance', null);
```

Inserção (Cont.)

- Faça com que cada aluno do departamento de Música que tenha ganho mais de 144 horas de crédito seja um professor do departamento de Música com um salário de 18.000.

```
insert into instructor
        select ID, name, dept_name, 18000
        from   student
        where  dept_name = 'Music' and total_cred > 144;
```

- A instrução **select from where** é avaliada completamente antes de qualquer um dos seus resultados ser inserido na relação.

  Caso contrário, consultas como

  insert into *table*1 **select** * **from** *table*1

  causariam um problema

Atualizações

- Dê um aumento salarial de 5% a todos os professores

```
update instructor
    set salary = salary * 1.05
```

Dê um aumento salarial de 5% aos instrutores que ganham menos de 70.000

```
update instructor
    set salary = salary * 1.05
    where salary < 70000;
```

- Conceder um aumento salarial de 5% aos instrutores cujo salário seja inferior à média

```
update instructor
    set salary = salary * 1.05
    where salary <  (select avg (salary)
                     from instructor);
```

Atualizações (Cont.)

Aumentar os salários dos instrutores cujo salário é superior a 100.000 em 3% e todos os outros em 5%

- Escreva duas instruções **de atualização** :

```
update instructor
    set salary = salary * 1.03
    where salary > 100000;
update instructor
    set salary = salary * 1.05
    where salary <= 100000;
```

- A ordem é importante
- Pode ser feito de forma mais elegante usando a cláusula **case** (próximo slide)

Case para atualizações condicionais

- A consulta do slide anterior pode ser escrita usando **case**

```
update instructor
        set salary = case
                        when salary <= 100000 then salary * 1.05
                        else salary * 1.03
                       end
```

Atualizações com subconsultas escalares

- Recalcular e atualizar o valor tot_creds para todos os alunos

```
update student S
    set tot_cred = (select sum(credits)
                    from takes, course
                    where takes.course_id = course.course_id  and
                          S.ID= takes.ID.and
                          takes.grade <> 'F' and
                          takes.grade is not null);
```