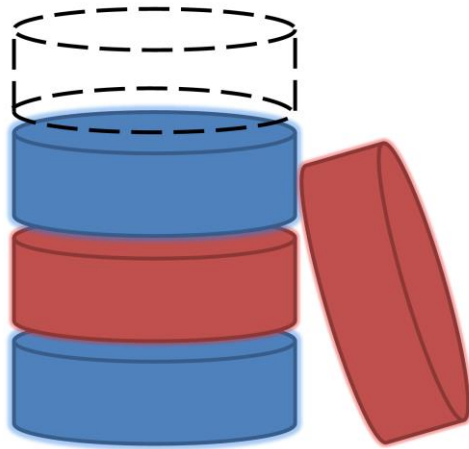


Interogări SQL



2

Interogări

- Posibile informații pe care dorim să le obținem din baza de date anterioară (*Faculty Database*) :
 - Care este numele studentului cu *sid* = 2833?
 - Care este salariul profesorilor care predau cursul *Alg100*?
 - Câți studenți sunt înscriși la cursul *Alg100*?
- Astfel de întrebări referitoare la datele stocate într-un SGBD se numesc *interogări*.
- → *limbaj de interogare*

Limbaje SGBD

- *Data Definition Language (DDL)*
 - Definesc structura **conceptuală**
 - Descriu **constrângerile de integritate**
 - Influențează **structura fizică** (în anumite SGBD-uri)
- *Data Manipulation Language (DML)*
 - Operații aplicate instanțelor unei baze de date
 - DML procedural (**cum?**) vs. DML declarative (**ce?**)
- *Limbaj gazdă*
 - Limbaj de programare obișnuit ce permite utilizatorilor să includă comenzi DML în propriul cod

Limbaje de interogare pentru BD relaționale

SQL (Structured Query Language)

SELECT *name* **FROM** *Students* **WHERE** *age* > 20

Algebra

$\pi_{name}(\sigma_{age > 20}(Students))$

Domain Calculus

$\{ \langle X \rangle \mid \exists V \exists Y \exists Z \exists T : Students(V, X, Y, Z, T) \wedge Z > 20 \}$

T-uple Calculus

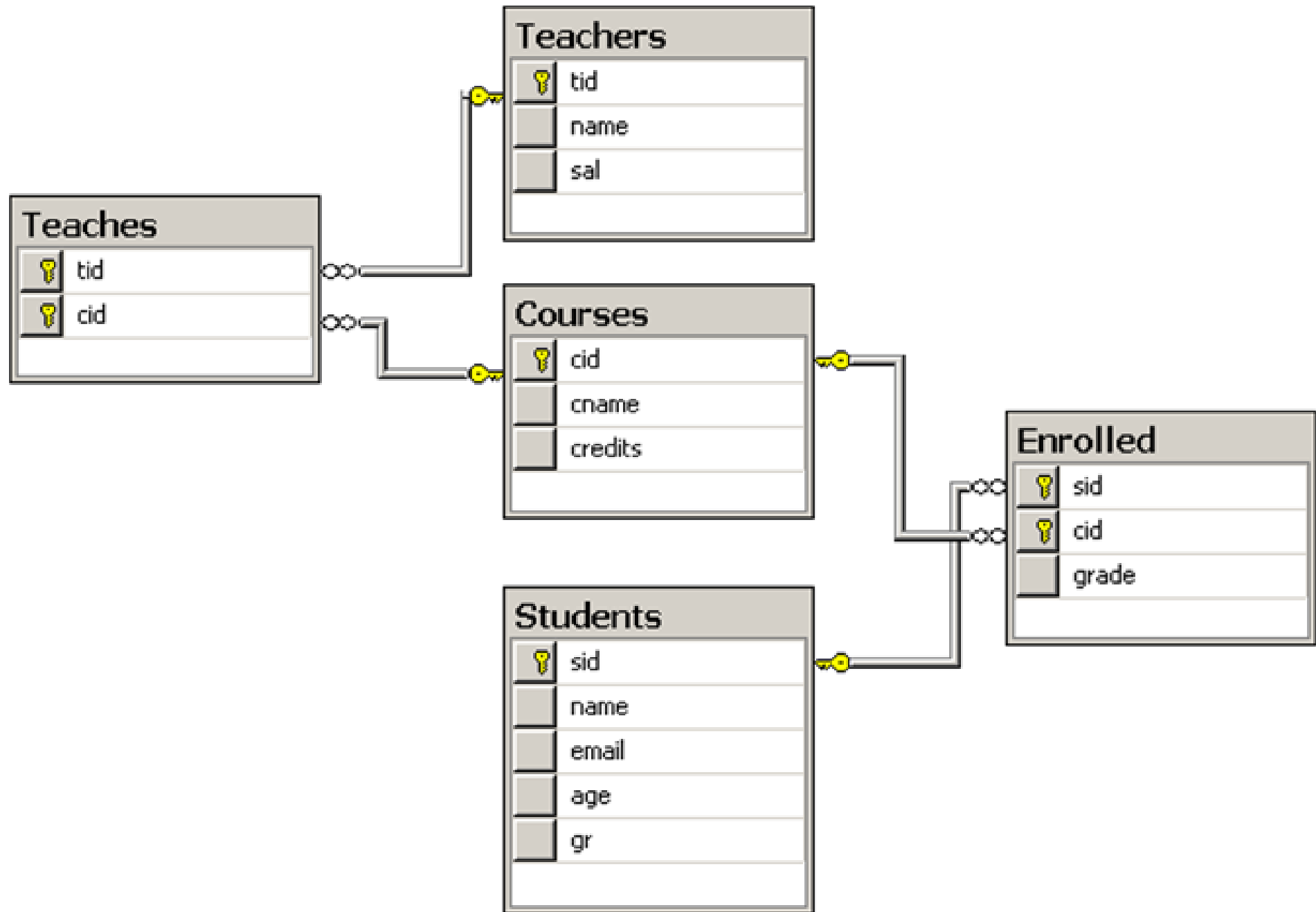
$\{ X \mid \exists Y : Y \in Students \wedge Y.age > 20 \wedge X.name = Y.name \}$

Structured Query Language (SQL)

- Dezvoltat de IBM (*system R*) în anii 1970
- Ulterior a apărut nevoia de standardizare
- Standarde (ANSI):
 - SQL-86
 - SQL-89 (minor revision)
 - SQL-92 (major revision) - *1,120 pagini*
 - SQL-99 (major extensions) - *2,084 pagini*
 - SQL-2003 (secțiuni SQL/XML) - *3,606 pagini*
 - SQL-2006
 - SQL-2008
 - SQL-2011
 - SQL-2016
 - SQL-2019

Nivele SQL

- Data-definition language (DDL):
 - Creare / stergere / modificare *tabele* și *views*.
 - Definire *constrangeri de integritate* (CI's).
- Data-manipulation language (DML)
 - Permit formularea de interogari
 - Inserare / ștergere / modificare înregistrări.
- *Controlul accesului:*
 - Asignează sau elimină drepturi de acces si modificare a *tabelelor* și a *view*-urilor.



Students

<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

SELECT

Lista tuturor studenților având vârsta 21 de ani:

```
SELECT *  
FROM Students S  
WHERE S.age = 21
```

<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1236	Anne	a@cs.ro	21	332

Returnează doar numele și adresele de e-mail:

```
SELECT S.name, S.email  
FROM Students S  
WHERE S.age = 21
```

<i>name</i>	<i>email</i>
John	j@cs.ro
Anne	a@cs.ro

Interogare SQL simplă

```
SELECT [DISTINCT] target-list  
FROM relation-list  
WHERE qualification
```

- *relation-list* - lista de nume de relații/tabele.
- *target-list* - listă de attribute ale relațiilor din *relation-list*
- *qualification* - comparații logice (*Attr op const* sau *Attr1 op Attr2*, unde *op* is one of $<$, $>$, $=$, \leq , \geq , \neq) combinate cu AND, OR sau NOT.
- *DISTINCT* (optional) - indică faptul că rezultatul final nu conține duplicate.

Evaluare conceptuală

```
SELECT [DISTINCT] target-list  
FROM relation-list  
WHERE qualification
```

- Calcul produs cartezian al tabelelor din *relation-list*.
- Filtrare înregistrări ce nu verifică *qualifications*.
- Ștergere attribute ce nu aparțin *target-list*.
- Dacă **DISTINCT** e prezent, se elimină înregistrările duplicate.

1. PRODUS CARTEZIAN

2. ELIMINA LINII

3. ELIMINA COLOANE

4. ELIMINA DUPLICATE

Această strategie e **doar**
la nivel *conceptual*!

Modul actual de evaluare
a unei interogări
e **mult** optimizat

Utilizare *alias* pentru

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade=10
```



```
SELECT name, cid  
FROM Students, Enrolled  
WHERE Students.sid=Enrolled.sid  
AND grade=10
```

Interogare: *Studentii care au cel puțin o notă*

```
SELECT S.sid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid
```

- Rezultatul e diferit cu **DISTINCT**?
- Ce efect are înlocuirea *S.sid* cu *S.sname* în clauza **SELECT**?

Rezultatul e diferit cu **DISTINCT** în acest caz?

Expresii și *string*-uri

- *Obține triplete (cu vârsta studenților + alte două expresii) pentru studenții al căror nume începe și se termină cu B și conține cel puțin trei caractere.*

```
SELECT S.age, age1=S.age-5, 2*S.age AS age2  
FROM Students S  
WHERE S.name LIKE 'B_ %B'
```

- **AS** și **=** sunt două moduri de redenumire a câmpurilor în rezultat.
- **LIKE** e folosit pentru comparații pe siruri de caractere. **'_'** reprezintă orice caracter și **'%'** stands reprezintă 0 sau mai multe caractere arbitrare.

INNER JOIN

```
SELECT S.name, C.cname
FROM Students S,
Enrolled E, Courses C
WHERE S.sid = E.sid
AND E.cid = C.cid
```



```
SELECT S.name, C.cname
FROM Students S
INNER JOIN Enrolled E ON
S.sid = E.sid
INNER JOIN Courses C ON
E.cid = C.cid
```

Students

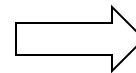
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1

LEFT OUTER JOIN

- Daca dorim sa regasim și studentii fără nici o notă la vreun curs:

```
SELECT S.name, C.cname
FROM Students S
LEFT OUTER JOIN Enrolled E
ON S.sid = E.sid
LEFT OUTER JOIN Courses C
ON E.cid = C.cid
```

Students

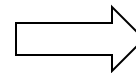
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
Anne	NULL

RIGHT OUTER JOIN

- Pentru a gasi notele asiguate unor studenti inexistenti:

```
SELECT S.name, C.cname
FROM Students S
RIGHT OUTER JOIN Enrolled E
ON S.sid = E.sid
INNER JOIN Courses C ON
E.cid = C.cid
```

Students

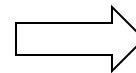
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
NULL	Databases2

FULL OUTER JOIN

- LEFT+RIGHT OUTER JOIN
- In majoritatea SGBD **OUTER** e optional

```
SELECT S.name, C.cname  
FROM Students S
```

```
FULL OUTER JOIN Enrolled E  
ON S.sid = E.sid
```

```
FULL OUTER JOIN Courses C  
ON E.cid = C.cid
```

Students

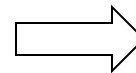
<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1237	DB2	9



<i>name</i>	<i>cname</i>
John	Algorithms1
Smith	Algorithms1
NULL	Databases2
NULL	Databases1
Anne	NULL

NULL values

- Field values in a tuple are sometimes *unknown* (e.g., a rating has not been assigned) or *inapplicable*.
 - SQL provides a special value *null* for such situations.
- The presence of *null* complicates many issues. E.g.:
 - Special operators needed to check if value is/is not *null*.
 - Is *rating* > 8 true or false when *rating* is equal to *null*?
What about **AND**, **OR** and **NOT** connectives?
 - We need a 3-valued logic (true, false and *unknown*).
 - Meaning of constructs must be defined carefully. (e.g., WHERE clause eliminates rows that don't evaluate to true.)
 - New operators (in particular *outer joins*) possible/needed.

Valoarea NULL

- În anumite situații valorile particulare ale unor attribute (câmpuri) pot fi *necunoscute* sau *inaplicabile* temporar.
 - SQL permite utilizarea unei valori speciale null pentru astfel de situații.
- Prezența valorii *null* implică unele probleme suplimentare:
 - E necesară implementarea unei logici cu 3 valori: *true*, *false* și *null* (de exemplu o condiție de tipul *rating*>8 va fi întotdeauna evaluată cu *false* dacă valoarea câmpului *rating* este *null*)
 - E necesară adăugarea unui operator special IS NULL / IS NOT NULL.

Operatori de agregare

COUNT (*)
COUNT ([DISTINCT] A)
SUM ([DISTINCT] A)
AVG ([DISTINCT] A)
MAX (A)
MIN (A)

atribut

```
SELECT COUNT (*)  
FROM Students S
```

```
SELECT AVG (S.age)  
FROM Students S  
WHERE S.gr=921
```

```
SELECT COUNT (DISTINCT S.gr)  
FROM Students S  
WHERE S.name='Bob'
```

```
SELECT S.name  
FROM Students S  
WHERE S.age = ANY  
      (SELECT MAX(S2.age)  
       FROM Students S2)
```

GROUP BY / HAVING

For $i = 221, 222, 223, 224 \dots$:

```
SELECT MIN(S.age)
FROM   Students S
WHERE  S.gr =  $i$ 
```


GROUP BY / HAVING

```
SELECT [DISTINCT] target-list
FROM    relation-list
WHERE   qualification
GROUP BY grouping-list
HAVING  group-qualification
```

GROUP BY / HAVING

```
SELECT [DISTINCT] target-list
FROM   relation-list
WHERE  qualification
GROUP BY grouping-list
HAVING   group-qualification
```

- Un *grup* este o mulțime de tupluri care au aceeași valoare pentru toate atributele din *grouping-list*.
- *target-list* conține (i) nume de atribut sau (ii) termeni ce utilizează operatori de agregare (e.g., MIN (*S.age*)).
 - numele de atribut (i) trebuie să fie o submulțime a *grouping-list*.
 - Intuitiv, fiecare tuplu din rezultat corespunde unui *grup*, și toate atributele vor avea o singură valoare per grup.

Numarul studentilor cu nota la cursurile cu 6 credite si media notelor acestora

```
SELECT  C.cid, COUNT (*) AS scount, AVG(grade)
FROM    Enrolled E, Courses C
WHERE   E.cid=C.cid AND C.credits=6
GROUP BY C.cid
```

Courses

<i>cid</i>	<i>cname</i>	<i>credits</i>
Alg1	Algorithms1	7
DB1	Databases1	6
DB2	Databases2	6

Students

<i>sid</i>	<i>name</i>	<i>email</i>	<i>age</i>	<i>gr</i>
1234	John	j@cs.ro	21	331
1235	Smith	s@cs.ro	22	331
1236	Anne	a@cs.ro	21	332

Enrolled

<i>sid</i>	<i>cid</i>	<i>grade</i>
1234	Alg1	9
1235	Alg1	10
1234	DB1	10
1234	DB2	9
1236	DB1	7

Enrolled

Courses

<i>sid</i>	<i>cid</i>	<i>grade</i>	<i>cid</i>	<i>cname</i>	<i>credits</i>
1234	Alg1	9	Alg1	Algorithms1	7
1234	Alg1	9	DB1	Databases1	6
1234	Alg1	9	DB2	Databases2	6
1235	Alg1	10	Alg1	Algorithms1	7
1235	Alg1	10	DB1	Databases1	6
1235	Alg1	10	DB2	Databases2	6
1234	DB1	10	Alg1	Algorithms1	7
1234	DB1	10	DB1	Databases1	6
1234	DB1	10	DB2	Databases2	6
1234	DB2	9	Alg1	Algorithms1	7
1234	DB2	9	DB1	Databases1	6
1234	DB2	9	DB2	Databases2	6
1236	DB1	7	Alg1	Algorithms1	7
1236	DB1	7	DB1	Databases1	6
1236	DB1	7	DB2	Databases2	6

```
SELECT C.cid,  
COUNT(*)AS scount,  
AVG(grade)AS average  
FROM Enrolled E,  
Courses C  
WHERE E.cid=C.cid  
AND C.credits=6  
GROUP BY C.cid
```

<i>Enrolled</i>			<i>Courses</i>		
<i>sid</i>	<i>cid</i>	<i>grade</i>	<i>cid</i>	<i>cname</i>	<i>credits</i>
1234	Alg1	9	Alg1	Algoritmics 1	7
1235	Alg1	10	Alg1	Algoritmics 1	7
1234	DB1	10	DB1	Databases1	6
1234	DB2	9	DB2	Databases2	6
1236	DB1	7	DB1	Databases1	6

```

SELECT C.cid,
COUNT(*)AS scount,
AVG(grade)AS average
FROM    Enrolled E,
        Courses C
WHERE   E.cid=C.cid
AND
C.credits=6
GROUP BY C.cid

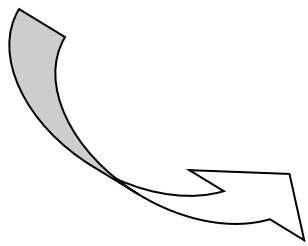
```

<i>sid</i>	<i>cid</i>	<i>grade</i>	<i>cid</i>	<i>cname</i>	<i>credits</i>
1234	DB1	10	DB1	Databases1	6
1234	DB2	9	DB2	Databases2	6
1236	DB1	7	DB1	Databases1	6

```

SELECT C.cid
COUNT(*) AS scount,
AVG(grade) AS average
FROM   Enrolled E,
       Courses C
WHERE  E.cid=C.cid
AND
      C.credits=6
GROUP BY C.cid
HAVING MAX(grade) = 10

```



<i>cid</i>	<i>scount</i>	<i>average</i>
DB1	2	8.5
DB2	1	9