SQL – Data Manipulation Language

Carla Teixeira Lopes

Bases de Dados
Mestrado Integrado em Engenharia Informática e Computação, FEUP

Based on Jennifer Widom slides

Agenda

~~Introduction~~

~~Basic SQL Statement~~

~~Table Variables and Set Operators~~

~~Subqueries in WHERE clauses~~

~~Subqueries in FROM and SELECT clauses~~

~~The JOIN family of operators~~

Aggregation

Null values

Data Modification statements

Aggregation

Perform aggregation over sets of values in multiple rows

Basic functions

- min, max, sum, avg, count
- Except *count*, all aggregations apply to a single attribute

SELECT $A_1, A_2, \ldots, A_n$

FROM $R_1, R_2, \ldots, R_m$

WHERE condition

GROUP BY columns

HAVING condition

} New clauses

A first aggregation query

```
SELECT  *
FROM    Student;
```

sID	sName	GPA	HS
123	Amy	3.9	1000
234	Bob	3.6	1500
345	Craig	3.5	500
456	Doris	3.9	1000
567	Edward	2.9	2000
678	Fay	3.8	200
789	Gary	3.4	800
987	Helen	3.7	800
876	Irene	3.9	400
765	Jay	2.9	1500
654	Amy	3.9	1000
543	Craig	3.4	2000

```
SELECT  avg(GPA)
FROM    Student;
```

avg(GPA)
3.566

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A second aggregation query

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT  *
FROM Student, Apply
WHERE Student.sID =
Apply.sID and major='CS';
```

sID	sName	GPA	HS	sID1	cName	major	dec
123	Amy	3.9	1000	123	Stanford	CS	Y
123	Amy	3.9	1000	123	Berkeley	CS	Y
345	Craig	3.5	500	345	Cornell	CS	Y
987	Helen	3.7	800	987	Stanford	CS	Y
987	Helen	3.7	800	987	Berkeley	CS	Y
876	Irene	3.9	400	876	Stanford	CS	N
543	Craig	3.4	2000	543	MIT	CS	N

```
SELECT  min(GPA)
FROM Student, Apply
WHERE Student.sID = Apply.sID
      and major='CS';
```

min(GPA)
3.4

Lowest GPA of students applying to CS

A third aggregation query

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT  *
FROM    Student, Apply
WHERE Student.sID =
Apply.sID and major='CS';
```

sID	sName	GPA	HS	sID1	cName	major	dec
123	Amy	3.9	1000	123	Stanford	CS	Y
123	Amy	3.9	1000	123	Berkeley	CS	Y
345	Craig	3.5	500	345	Cornell	CS	Y
987	Helen	3.7	800	987	Stanford	CS	Y
987	Helen	3.7	800	987	Berkeley	CS	Y
876	Irene	3.9	400	876	Stanford	CS	N
543	Craig	3.4	2000	543	MIT	CS	N

```
SELECT  avg(GPA)
FROM    Student, Apply
WHERE Student.sID = Apply.sID
and major='CS';
```

avg(GPA)
3.71

Average GPA of students applying to CS?

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A third aggregation query

```
SELECT  *
FROM    Student
WHERE sID in (select sID from
Apply where major='CS');
```

sID	sName	GPA	HS
123	Amy	3.9	1000
345	Craig	3.5	500
987	Helen	3.7	800
876	Irene	3.9	400
543	Craig	3.4	2000

```
SELECT  avg(GPA)
FROM    Student
WHERE sID in (select sID from
Apply where major='CS');
```

avg(GPA)
3.68

Average GPA of students applying to CS

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A first count query

```
SELECT  *
FROM    College
WHERE enr > 15000;
```

cName	state	enr
Berkeley	CA	36000
Cornell	NY	21000

```
SELECT  count(*)
FROM    College
WHERE enr > 15000;
```

count(*)
2

Number of colleges bigger than 15000

A second count query

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT  count(sID)
FROM    Apply
WHERE cName = 'Cornell';
```

```
SELECT  count(distinct sID)
FROM    Apply
WHERE cName = 'Cornell';
```

count(*)
6

count(distinct sID)
3

Number of students applying to Cornell?

Number of students applying to Cornell

Number of applications to Cornell → COUNT applies to duplicates, unless otherwise stated

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A third count query

```
SELECT  *
FROM     Student S1
WHERE (select count(*) from Student S2
            where S2.sID<>S1.sID and S2.GPA = S1.GPA) =
          (select count(*) from Student S2
           where S2.sID <> S1.sID and S2.sizeHS = S1.sizeHS);
```

What does it compute?

Students such that number of other students with same GPA is equal to number of other students with same HS size

A fourth count query

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT  CS.avgGPA – NonCS.avgGPA
FROM    (select avg(GPA) as avgGPA from Student
           where sID in (
              select sID from Apply where major = 'CS')) as CS,
         (select avg(GPA) as avgGPA from Student
           where sID not in (
              select sID from Apply where major = 'CS')) as nonCS;
```

CS.avgGPA-NonCS.avgGPA
0.19

What does it compute?

Amount by which average GPA of students applying to CS exceeds average of students not applying to CS

A fourth count query

Compute the “amount by which average GPA of students applying to CS exceeds average of students not applying to CS” using subqueries in the select clause

```
SELECT (select avg(GPA) from Student
            where sID in (
                select sID from Apply where major = ‘CS’)) -
           (select avg(GPA) from Student
            where sID not in (
                select sID from Apply where major = ‘CS’)) as d
FROM Student;
```

d
0.19
0.19
0.19
0.19
…

Duplicates: difference is computed for each student

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A fourth count query

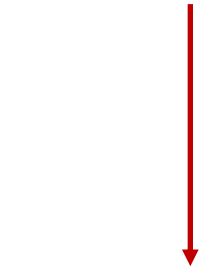
Compute the “amount by which average GPA of students applying to CS exceeds average of students not applying to CS” using subqueries in the select clause

```
SELECT DISTINCT (select avg(GPA) as avgGPA from Student
                 where sID in (
                     select sID from Apply where major = ‘CS’)) -
                (select avg(GPA) as avgGPA from Student
                 where sID not in (
                     select sID from Apply where major = ‘CS’)) as d
FROM Student;
```

d
0.19

Another example

```
SELECT SUM(price * quantity)
FROM   Purchase
WHERE  product = 'cake'
```



Sum(price*quantity)
50

(= 1∗20 + 1.50∗20)

Purchase

Product	Date	Price	Quantity
cake	21/10	1	20
banana	3/10	0.5	10
banana	10/10	1	10
cake	25/10	1.50	20

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

Group by clause

Only used in conjunction with aggregation

It partitions a relation by values of a given attribute or set of attributes

SELECT	S
FROM	$R_1,\ldots,R_n$
WHERE	C_1
GROUP BY	$a_1,\ldots,a_k$

Semantics of the query

1. Compute the FROM and WHERE clauses
2. Group by the attributes in the GROUP BY
3. Compute the SELECT clause: grouped attributes and aggregates

Group by clause

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT  *
FROM Apply
ORDER BY cName;
```

```
SELECT cName, count(*)
FROM Apply
GROUP BY cName;
```

1. FROM and WHERE

sID	cName	major	dec
123	Berkeley	CS	Y
234	Berkeley	biology	N
987	Berkeley	CS	Y
123	Cornell	EE	
345	Cornell	bio	
…	…	…	

2. Group By

sID	cName	major	dec
123	Berkeley	CS	Y
234		biology	N
987		CS	Y
123	Cornell	EE	Y
345		bioengineering	N
…		…	…

3. Select

cName	count(*)
Berkeley	3
Cornell	6
MIT	4
Stanford	6

A second query with group by

```
SELECT  *
FROM College
ORDER BY state;
```

cName	state	enr
Stanford	CA	15000
Berkeley	CA	36000
MIT	MA	10000
Cornell	NY	21000

```
SELECT state, sum(enr)
FROM College
GROUP BY state;
```

state	sum(enr)
CA	51000
MA	10000
NY	21000

College enrollments by state

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

A third query with group by

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT cName, major, GPA
FROM Student, Apply
WHERE Student.sID = Apply.sID
ORDER BY cName, major;
```

cName	major	GPA
Berkeley	biology	3.6
Berkeley	CS	3.9
Berkeley	CS	3.7
Cornell	bioengineering	3.5
…	…	…

```
SELECT cName, major, min(GPA),
max(GPA)
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY cName, major;
```

cName	major	min(GPA)	max(GPA)
Berkeley	biology	3.6	3.6
Berkeley	CS	3.7	3.9
Cornell	bioengineering	3.5	3.5
…	…	…	…

Minimum and maximum GPAs of applicants to each college and major

A fourth query with group by

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

Find the largest spread of GPAs in colleges and majors

Step 1: Find the spread of GPAs of applicants for each college and major

```
SELECT mx-mn
FROM ( SELECT cName, major, min(GPA) as mn, max(GPA) as mx
       FROM Student, Apply
       WHERE Student.sID = Apply.sID
       GROUP BY cName, major ) M;
```

mx-mn
0
0.19
0
…

A fourth query with group by

Find the largest spread of GPAs in colleges and majors

Step 2: Find the largest spread

```
SELECT max(mx-mn)
FROM ( SELECT cName, major, min(GPA) as mn, max(GPA) as mx
    FROM Student, Apply
    WHERE Student.sID = Apply.sID
    GROUP BY cName, major ) M;
```

max(mx-mn)
0.899

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A fifth query with group by

```
SELECT Student.sID, cName
FROM Student, Apply
WHERE Student.sID = Apply.sID
ORDER BY Student.sID;
```

sID	cName
123	Stanford
123	Stanford
123	Berkeley
123	Cornell
234	Berkeley
…	…

```
SELECT Student.sID, count(distinct cName)
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY Student.sID;
```

sID	Count(distinct cName)
123	3
234	1
345	2
…	…

Number of colleges applied to by each student

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

A fifth query with group by

What if we also want the student's name?

```
SELECT Student.sID, sName, count(distinct cName)
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY Student.sID;
```

It only worked because for each sID we only have one sName

sID	sName	count(distinct cName)
123	Amy	3
234	Bob	1
345	Craig	2
…		…

A fifth query with group by

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

```
SELECT Student.sID, sName, count(distinct cName), cName
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY Student.sID;
```

sID	sName	count(distinct cName)	cName
123	Amy	3	Stanford
234	Bob	1	Berkeley
345	Craig	2	MIT
…		…	…

It chooses a random value from the group to include

Some systems throw an error in this situation

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A fifth query with group by

```
SELECT Student.sID, count(distinct cName)
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY Student.sID;
```

sID	Count(distinct cName)
123	3
234	1
345	2
…	…

What if want to list students who haven't applied anywhere with a 0 in the count?

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A fifth query with group by

```
SELECT Student.sID, count(distinct cName)
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY Student.sID
UNION
SELECT sID, 0
FROM Student
WHERE sID not in (select sID from Apply);
```

sID	Count(distinct cName)
876	2
987	2
456	0
567	0
…	…

Having clause

Applies conditions to the results of aggregate functions

- Check conditions that involve the entire group
- Where clause applies to one tuple at a time

Applied after the GROUP BY clause

SELECT	S	→ attributes $a_1,\ldots,a_k$ and/or aggregates over other attributes
FROM	$R_1,\ldots,R_n$	
WHERE	C_1	→ any condition on the attributes in $R_1,\ldots,R_n$
GROUP BY	$a_1,\ldots,a_k$	
HAVING	C_2	→ any condition on the aggregate expressions

Having clause

List the colleges with fewer than 5 applications

```
SELECT cName
FROM Apply
GROUP BY cName
HAVING count(*) < 5;
```

cName
Berkeley
MIT

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

Having clause

What if we're interested in the colleges that have fewer than 5 applicants

```
SELECT cName
FROM Apply
GROUP BY cName
HAVING count(distinct sID) < 5;
```

cName
Berkeley
Cornell
MIT

College(cName, state, enr)
Student(sID, sName, GPA, sizeHS)
Apply(sID, cName, major, decision)

College(cName, state, enr)

Student(sID, sName, GPA, sizeHS)

Apply(sID, cName, major, decision)

A final having clause

```
SELECT major
FROM Student, Apply
WHERE Student.sID = Apply.sID
GROUP BY major
HAVING max(GPA) < (SELECT avg(GPA) FROM Student);
```

major
bioengineering
psychology

What does it compute?

Majors whose applicant's maximum GPA is below the average

Kahoot time!

Any doubts?

Readings

Jeffrey Ullman, Jennifer Widom, A first course in Database Systems 3rd Edition

- Section 6.1 – Simple Queries in SQL
- Section 6.2 – Queries Involving More Than One Relation
- Section 6.3 - Subqueries
- Section 6.4 – Full-Relation Operations
- Section 6.5 – Database Modifications

Philip Greenspun, SQL for Web Nerds, http://philip.greenspun.com/sql/