CMU SCS

Carnegie Mellon Univ.
Dept. of Computer Science
15-415/615 - DB Applications

C. Faloutsos & A. Pavlo

Lecture#6: *Rel. model - SQL part1*

(R&G, chapter 5)

CMU SCS

General Overview - rel. model

- Formal query languages
  - rel algebra and calculi
- Commercial query languages
  - SQL
  - QBE, (QUEL)

CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where, renaming
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

Relational Query Languages

- A major strength of the relational model: supports simple, powerful *querying* of data.
- Two sublanguages:
- DDL – Data Definition Language
  - define and modify schema (at all 3 levels)
- DML – Data Manipulation Language
  - Queries can be written intuitively.

CMU SCS

Relational languages

- The DBMS is responsible for efficient evaluation.
  - Query optimizer: re-orders operations and generates query plan

CMU SCS

The SQL Query Language

- **The most widely used relational query language.**
  - Major standard is SQL-1999 (=SQL3)
    - Introduced "Object-Relational" concepts
    - SQL 2003, SQL 2008 have small extensions
  - SQL92 is a basic subset

CMU SCS

SQL (cont'd)

- PostgreSQL has some "unique" aspects (as do most systems).

CMU SCS

DML

General form

select a1, a2, … an
from r1, r2, … rm
where P
[**order by** ….]
[**group by** …]
[**having** …]

CMU SCS

Reminder: our Mini-U db

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

DML - eg:

find the ssn(s) of everybody called "smith"

select ssn
from student
where name="smith"

CMU SCS

DML - observation

General form

select a1, a2, … an
from r1, r2, … rm
where P

equivalent rel. algebra query?

CMU SCS

DML - observation

General form

select a1, a2, … an
from r1, r2, … rm
where P

$$\pi_{a1,a2,...an}(\sigma_P(r1 \times r2 \times ... \times rm))$$

CMU SCS

DML - observation

General form

select distinct a1, a2, … an
from r1, r2, … rm
where P

$$\pi_{a1,a2,...an}(\sigma_P(r1 \times r2 \times ... \times rm))$$

CMU SCS

select clause

select [**distinct** | **all**] name
from student
where address="main"

CMU SCS

where clause

find ssn(s) of all "smith"s on "main"
select ssn
from student
where address="main" **and**
name = "smith"

CMU SCS

where clause

- boolean operators (**and or not** …)
- comparison operators (<, >, =, …)
- and more…

CMU SCS

What about strings?

find student ssns who live on "main" (st or str or street - ie., "main st" or "main str" …)

CMU SCS

What about strings?

find student ssns who live on "main" (st or str or street)

select ssn
from student
where address **like** "main%"

%: variable-length don't care

_: single-character don't care

CMU SCS

from clause

find names of people taking 15-415

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

from clause

find names of people taking 15-415

select name
from student, takes
where ???

CMU SCS

from clause

find names of people taking 15-415

```
select name
from student, takes
where student.ssn = takes.ssn and
   takes.c-id = "15-415"
```


CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where, **renaming**
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

renaming - tuple variables

find names of people taking 15-415

```
select name
from ourVeryOwnStudent, studentTakingClasses
where ourVeryOwnStudent.ssn =
     studentTakingClasses.ssn
 and  studentTakingClasses.c-id = "15-415"
```


CMU SCS

2 reasons to rename:

- #1) shorthand
- #2) self-joins

CMU SCS

#1) renaming - tuple variables

find names of people taking 15-415

select name

from ourVeryOwnStudent,
studentTakingClasses

where ourVeryOwnStudent.ssn
=studentTakingClasses.ssn

and studentTakingClasses.c-id = "15-415"

CMU SCS

#1) renaming - tuple variables

find names of people taking 15-415

select name

from ourVeryOwnStudent **as** S,
studentTakingClasses **as** T

where S.ssn =T.ssn

and T.c-id = "15-415"

CMU SCS

#2) renaming - self-join

- self -joins: find Tom's grandparent(s)

PC	
p-id	c-id
Mary	Tom
Peter	Mary
John	Tom

PC	
p-id	c-id
Mary	Tom
Peter	Mary
John	Tom

CMU SCS

#2) renaming - self-join

find grandparents of "Tom" (PC(p-id, c-id))

select gp.p-id

from PC **as** gp, PC

where gp.c-id= PC.p-id

and PC.c-id = "Tom"

CMU SCS

#2) renaming - theta join

find course names with more units than 15-415

CMU SCS

#2) renaming - theta join

find course names with more units than 15-415

select c1.c-name
from class **as** c1, class **as** c2
where c1.units > c2.units
and c2.c-id = "15-415"

CMU SCS

find course names with more units than 15-415

select c1.name
from class **as** c1, class **as** c2
where c1.units > c2.units
and c2.c-id = "15-415"

$$\{t \mid \exists c1 \in CLASS\, \exists c2 \in CLASS(c1[c-id] = 15-415 \wedge c2[units] > c1[units] \wedge t[c-name] = c2[c-name])\}$$

CMU SCS

find course names with more units than 15-415

select c2.name
from class **as** c1, class **as** c2
where c2.units > c1.units
and c1.c-id = "15-415"

$$\{t \mid \exists c1 \in CLASS\, \exists c2 \in CLASS(c1[c-id] = 15-415 \wedge c2[units] > c1[units] \wedge t[c-name] = c2[c-name])\}$$

CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

set operations

find ssn of people taking both 15-415 and 15-413

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

set operations

find ssn of people taking both 15-415 and 15-413

select ssn
from takes
where c-id="15-415" **and**
c-id="15-413"

CMU SCS

set operations

find ssn of people taking both 15-415 and 15-413

(select ssn **from** takes **where** c-id="15-415")
intersect
(select ssn **from** takes **where** c-id="15-413")

other ops: **union** , **except**

CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

Ordering

find student records, sorted in name order

select *
from student
~~**where**~~

CMU SCS

Ordering

find student records, sorted in name order

select *
from student
order by name **asc**

asc is the default

CMU SCS

Ordering

find student records, sorted in name order;
break ties by reverse ssn

select *
from student
order by name**,** ssn **desc**

CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

Aggregate functions

find avg grade, across all students

select ??

from takes

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions

find avg grade, across all students

select avg(grade)

from takes

SSN	c-id	grade
123	15-413	4
234	15-413	3

- result: a single number
- Which other functions?

CMU SCS

Aggregate functions

- A: **sum count min max (std)**

CMU SCS

Aggregate functions

find total number of enrollments

select count(*)

from takes

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions

find total number of students in 15-415

select count(*)

from takes

where c-id="15-415"

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions

find total number of students in each course

select count(*)

from takes

where ???

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions

find total number of students in each course

select c-id, **count(*)**

from takes

group by c-id

SSN	c-id	grade
123	15-413	4
234	15-413	3

c-id	count
15-413	2

CMU SCS

Aggregate functions

find total number of students in each course

select c-id, **count(*)**
from takes
group by c-id
order by c-id

SSN	c-id	grade
123	15-413	4
234	15-413	3

c-id	count
15-413	2

CMU SCS

Aggregate functions

find total number of students in each course, and sort by count, decreasing

select c-id, **count(*) as** pop
from takes
group by c-id
order by pop **desc**

SSN	c-id	grade
123	15-413	4
234	15-413	3

c-id	pop
15-413	2

CMU SCS

Aggregate functions- 'having'

find students with GPA > 3.0

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions- 'having'

find students with GPA > 3.0

select ???, **avg**(grade)
from takes
group by ???

SSN	c-id	grade
123	15-413	4
234	15-413	3

CMU SCS

Aggregate functions- 'having'

find students with GPA > 3.0

select ssn, **avg**(grade)
from takes
group by ssn
???

SSN	c-id	grade
123	15-413	4
234	15-413	3

SSN	avg(grade
123	4
234	3

CMU SCS

Aggregate functions- 'having'

find students with GPA > 3.0

select ssn, **avg**(grade)
from takes
group by ssn
having avg(grade)>3.0

SSN	c-id	grade
123	15-413	4
234	15-413	3

SSN	avg(grade
123	4
~~234~~	~~3~~

'having' <-> 'where' for groups

CMU SCS

Aggregate functions- 'having'

find students and GPA,
for students with > 5 courses

select ssn, **avg**(grade)
from takes
group by ssn
having count(*) > 5

SSN	c-id	grade
123	15-413	4
234	15-413	3

SSN	avg(grade
123	4
~~234~~	~~3~~

CMU SCS

Overview - detailed - SQL

- DML
  - select, from, where, renaming
  - set operations
  - ordering
  - aggregate functions
  - nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

DML

General form

select a1, a2, … an
from r1, r2, … rm
where P
[**order by** ….]
[**group by** …]
[**having** …]

CMU SCS

Reminder: our Mini-U db

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

DML - nested subqueries

find names of students of 15-415

select name
from student
where ...

"ssn in the set of people that take 15-415"

CMU SCS

DML - nested subqueries

find names of students of 15-415

select name
from student
where ……...
  select ssn
  from takes
  where c-id ="15-415"

CMU SCS

DML - nested subqueries

find names of students of 15-415

```
select name
from student
where  ssn in (
  select ssn
  from takes
  where  c-id =“15-415”)
```


CMU SCS

DML - nested subqueries

- **‘in’** compares a value with a set of values
- **‘in’** can be combined other boolean ops
- it is redundant (but user friendly!):

```
select name
from student …..
where c-id = “15-415” ….
```


CMU SCS

DML - nested subqueries

- **‘in’** compares a value with a set of values
- **‘in’** can be combined other boolean ops
- it is redundant (but user friendly!):

```
select name
from student, takes
where c-id = “15-415” and
    student.ssn=takes.ssn
```


CMU SCS

DML - nested subqueries

find names of students taking 15-415 and living on “main str”

```
select name
from student
where address=“main str” and ssn in
  ( select ssn from takes where  c-id =“15-415”)
```

CMU SCS

DML - nested subqueries

- **'in'** compares a value with a set of values
- other operators like '**in**' ??

CMU SCS

DML - nested subqueries

find student record with highest ssn

select *
from student
where ssn
is greater than every other ssn

CMU SCS

DML - nested subqueries

find student record with highest ssn

select *
from student
where ssn *greater than every*
select ssn **from** student

CMU SCS

DML - nested subqueries

find student record with highest ssn

select *
from student
where ssn > **all** (
select ssn **from** student)

almost correct

CMU SCS

DML - nested subqueries

find student record with highest ssn

```
select *
from student
where ssn >= all (
   select ssn   from student)
```


CMU SCS

Puzzle

DML - nested subqueries

find student record with highest ssn - without nested subqueries?

```
select S1.ssn, S1.name, S1.address
from student as S1, student as S2
where S1.ssn > S2.ssn
```

is not the answer (what does it give?)

CMU SCS

Puzzle

DML - nested subqueries

S1

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

S2

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

S1 x S2

S1. ssn	S2.ssn	....
123	123	...
234	123	...
123	234	
234	234	

S1.ssn>S2.ssn

CMU SCS

Puzzle

DML - nested subqueries

```
select S1.ssn, S1.name, S1.address
from student as S1, student as S2
where S1.ssn > S2.ssn
```

gives all but the smallest ssn -

aha!

CMU SCS

Puzzle

DML - nested subqueries

find student record with highest ssn - without nested subqueries?

select S1.ssn, S1.name, S1.address
from student **as** S1, student **as** S2
where S1.ssn < S2.ssn

gives all but the highest - therefore….

CMU SCS

Puzzle

DML - nested subqueries

find student record with highest ssn - without nested subqueries?

(**select** * from student) **except**
(**select** S1.ssn, S1.name, S1.address
from student as S1, student as S2
where S1.ssn < S2.ssn)

CMU SCS

Puzzle

DML - nested subqueries

(**select** * **from** student) **except**
(**select** S1.ssn, S1.name, S1.address
from student as S1, student as S2
where S1.ssn < S2.ssn)

---

select *
from student
where ssn >= all (select ssn from student)

CMU SCS

DML - nested subqueries

Drill: Even more readable than

select * **from** student
where ssn >= **all** (**select** ssn **from** student)

CMU SCS

DML - nested subqueries

Drill: Even more readable than

select * **from** student
where ssn >= **all** (**select** ssn **from** student)

select * **from** student
where ssn **in**
(**select max**(ssn) **from** student)

CMU SCS

DML - nested subqueries

Drill: find the ssn of the student with the highest GPA

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CMU SCS

DML - nested subqueries

Drill: find the ssn and GPA of the student with the highest GPA

select ssn, **avg**(grade) **from** takes
~~**where**~~

CMU SCS

DML - nested subqueries

Drill: find the ssn and GPA of the student with the highest GPA

select ssn, **avg**(grade) **from** takes
group by ssn
having avg(grade**) …...**

greater than every other GPA on file

CMU SCS

DML - nested subqueries

Drill: find the ssn and GPA of the student with the highest GPA

```
select ssn, avg(grade) from takes
group by ssn
having avg( grade) >=  all
   ( select avg( grade )
     from student group by ssn )
```

} all GPAs

CMU SCS

DML - nested subqueries

- **'in'** and **'>= all'** compares a value with a set of values
- other operators like these?

CMU SCS

DML - nested subqueries

- **<all(), <>all()** ...
- '**<>all**' is identical to '**not in**'
- **>some()**, **>= some** () ...
- '**= some()**' is identical to '**in'**
- **exists**

CMU SCS

DML - nested subqueries

Drill for **'exists'**: find all courses that nobody enrolled in

select c-id **from** class ….*with no tuples in 'takes'*

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

CMU SCS

DML - nested subqueries

Drill for **'exists'**: find all courses that nobody enrolled in

select c-id **from** class
where not exists
(**select** * **from** takes
where class.c-id = takes.c-id)

CMU SCS

DML - derived relations

find the ssn with the highest GPA

select ssn, **avg**(grade) **from** takes
group by ssn
having avg(grade) >= **all**
(select avg(grade **)**
from takes **group by** ssn **)**

CMU SCS

DML - derived relations

find the ssn with the highest GPA

Query would be easier, if we had a table like: helpfulTable (ssn, gpa):

HelpfulTable	
Ssn	Gpa
123	3.5
678	3.3

then what?

CMU SCS

DML - derived relations

HelpfulTable	
Ssn	Gpa
123	3.5
678	3.3

select ssn, gpa
from helpfulTable
where gpa **in** (**select** **max**(gpa)
from helpfulTable)

CMU SCS

DML - derived relations

find the ssn with the highest GPA -
Query for helpfulTable (ssn, gpa)?

HelpfulTable	
Ssn	Gpa
123	3.5
678	3.3

CMU SCS

DML - derived relations

find the ssn with the highest GPA
Query for helpfulTable (ssn, gpa)?

HelpfulTable	
Ssn	Gpa
123	3.5
678	3.3

select ssn, **avg**(grade)
from takes
group by ssn

CMU SCS

DML - derived relations

find the ssn with the highest GPA

helpfulTable(ssn,gpa)

```
select ssn, gpa                 select ssn, avg(grade)
  from helpfulTable             from takes
where gpa = (select  max(gpa)   group by ssn
             from helpfulTable)
```


CMU SCS

DML - derived relations

find the ssn with the highest GPA

```
select ssn, gpa
from (select ssn, avg(grade)
        from takes
        group by ssn)
        as helpfulTable(ssn, gpa)
where gpa in (select  max(gpa)
                from helpfulTable)
```

CMU SCS

Overview - detailed - SQL

- DML
  - ✔ select, from, where, renaming
  - ✔ set operations
  - ✔ ordering
  - ✔ aggregate functions
  - ✔ nested subqueries
- other parts: DDL, embedded SQL, auth etc

CMU SCS

Next lecture:

- DML
  - ✔ select, from, where, renaming
  - ✔ set operations
  - ✔ ordering
  - ✔ aggregate functions
  - ✔ nested subqueries
- other parts: DDL, embedded SQL, auth etc