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

Faloutsos
Lecture#7: Rel. model - SQL part2

General Overview - rel. model

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

Overview - detailed - SQL

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

General form

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

Reminder: our Mini-U db

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>234</td>
<td>15-412</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-id</td>
<td>grade</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

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”
DML - nested subqueries

find names of students of 15-415

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

• ‘in’ compares a value with a set of values
• ‘in’ can be combined other boolean ops
• it is redundant (but user friendly!):
  ```sql
  select name
  from student ...
  where c-id = "15-415" ...
  ```
DML - nested subqueries

- ‘in’ compares a value with a set of values
- ‘in’ can be combined with other boolean ops
- it is redundant (but user friendly!):
  ```
  select name
  from student, takes
  where c-id = "15-415" and
  student.ssn=takes.ssn
  ```

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")
```
DML - nested subqueries

find student record with highest ssn

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

almost correct
DML - nested subqueries

find student record with highest ssn

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

find student record with highest ssn - without nested subqueries?

```sql
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?)

DML - nested subqueries

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>STUDENT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
<td>Name</td>
</tr>
<tr>
<td>123</td>
<td>smith</td>
</tr>
<tr>
<td>234</td>
<td>jones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S1 x S2</th>
</tr>
</thead>
</table>
| 123     | 234     | ...
| 123     | 234     | ...
| 123     | 234     | ...
| 123     | 234     | ...

S1.ssn > S2.ssn
DML - nested subqueries

```sql
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!

DML - nested subqueries

```sql
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….

DML - nested subqueries

```sql
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)
```
DML - nested subqueries

\[
\begin{align*}
&\text{(select * from student) except} \\
&(\text{select S1.ssn, S1.name, S1.address from student as S1, student as S2 where S1.ssn < S1.ssn}) \\
&\text{select * from student where ssn >= all (select ssn from student)}
\end{align*}
\]

Drill: Even more readable than

\[
\begin{align*}
&\text{select * from student} \\
&\text{where ssn >= all (select ssn from student)}
\end{align*}
\]

DML - nested subqueries

Drill: Even more readable than

\[
\begin{align*}
&\text{select * from student} \\
&\text{where ssn >= all (select ssn from student)}
\end{align*}
\]

\[
\begin{align*}
&\text{select * from student where ssn in} \\
&(\text{select max(ssn) from student})
\end{align*}
\]
DML - nested subqueries

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

```sql
select ssn
from takes
where avg(grade) greater than every other GPA on file
```

DML - nested subqueries

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

```sql
select ssn, avg(grade) from takes
where
```

DML - nested subqueries

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

```sql
select ssn, avg(grade) from takes
group by ssn
having avg(grade) greater than every other GPA on file
```
DML - nested subqueries

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

\[
\begin{align*}
\text{select} & \quad \text{ssn, avg(grade) from takes} \\
& \quad \text{group by ssn} \\
& \quad \text{having avg(grade) } \geq \text{ all} \\
& \quad \left( \text{select avg(grade) from student group by ssn} \right) \} \text{ all GPAs}
\end{align*}
\]

DML - nested subqueries

- ‘in’ and ‘\( \geq \) all’ compares a value with a set of values
- other operators like these?

DML - nested subqueries

- <all>(), =all() ...
- ‘=all’ is identical to ‘not in’
- >some(), >= some () ...
- ‘= some()’ is identical to ‘in’
- exists
DML - nested subqueries

Drill for ‘exists’: find all courses that nobody enrolled in

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

<table>
<thead>
<tr>
<th>SSN</th>
<th>c-id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>A</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>c-id</th>
<th>c-name</th>
<th>units</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-413</td>
<td>s.e.</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>15-412</td>
<td>o.s.</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

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 )
```
DML - derived relations

find the ssn with the highest GPA

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

<table>
<thead>
<tr>
<th>Ssn</th>
<th>GPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>3.5</td>
</tr>
<tr>
<td>678</td>
<td>3.3</td>
</tr>
</tbody>
</table>

then what?

DML - derived relations

```
select ssn, gpa
from helpfulTable
where gpa in (select max(gpa)
              from helpfulTable)
```
DML - derived relations

find the ssn with the highest GPA

Query for helpfulTable (ssn, gpa)?

```sql
select ssn, avg(grade)
from takes
group by ssn
```

DML - derived relations

find the ssn with the highest GPA

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

DML - derived relations

find the ssn with the highest GPA

```sql
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)
```
Views

find the ssn with the highest GPA -
we can create a permanent, virtual table:

```
create view helpfulTable(ssn, gpa) as
    select ssn, avg(grade)
    from takes
    group by ssn
```

Views

- views are recorded in the schema, for ever (ie.,
  until ‘drop view…’)
- typically, they take little disk space, because
  they are computed on the fly
- (but: materialized views…)

Overview of a DBMS

- casual user
- DBA
- DML precomp.
- DML parser
- trans. mgr
- buffer mgr
- DDL parser
- catalog
- create view..
Overview - detailed - SQL

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