Carnegie Mellon Univ.
Dept. of Computer Science
15-415 - Database Applications

C. Faloutsos
Rel. model - SQL part I

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>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>TAKES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
</tr>
<tr>
<td>123</td>
</tr>
<tr>
<td>234</td>
</tr>
</tbody>
</table>

DML - eg:
find the ssn(s) of everybody called “smith”
select ssn
from student
where name = "smith"
DML - observation

General form

\[
\begin{align*}
\text{select} & \quad a_1, a_2, \ldots, a_n \\
\text{from} & \quad r_1, r_2, \ldots, r_m \\
\text{where} & \quad P
\end{align*}
\]

equivalent rel. algebra query?

\[
\pi_{a_1, a_2, \ldots, a_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))
\]

DML - observation

General form

\[
\begin{align*}
\text{select} & \quad \text{distinct } a_1, a_2, \ldots, a_n \\
\text{from} & \quad r_1, r_2, \ldots, r_m \\
\text{where} & \quad P
\end{align*}
\]

\[
\pi_{a_1, a_2, \ldots, a_n} (\sigma_P (r_1 \times r_2 \times \ldots \times r_m))
\]

select clause

\[
\text{select [distinct | all] Name} \\
\text{from student} \\
\text{where address = "main"}
\]

where clause

find ssn(s) of all “smith’s on “main”

\[
\begin{align*}
\text{select} & \quad \text{ssn} \\
\text{from student} \\
\text{where address = "main" and name = "smith"}
\end{align*}
\]

where clause

- boolean operators (and or not ...)
- comparison operators (<, >, =, ...)
- and more...
What about strings?

find student ssn's who live on "main" (st or str or street - ie, "main st" or "main str" ...)

What about strings?

find student ssn's 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

from clause

find names of people taking 15-415

<table>
<thead>
<tr>
<th>STUDENT</th>
<th>CLASS</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>TAKES</th>
<th>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>

from clause

find names of people taking 15-415
select name
from student, takes
where ??

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"
renaming - tuple variables

find names of people taking 15-415
select name
from curVeryOwnStudent as S,
studentTakingClasses as T
where S.ssn = T.ssn
and T.c-id = "15-415"

renaming - self-join

• self-joins: find Tom’s grandparent(s)

<table>
<thead>
<tr>
<th>PC</th>
<th>P-id</th>
<th>C-id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mary</td>
<td>Tom</td>
<td>(Tom)</td>
</tr>
<tr>
<td>Peter</td>
<td>Mary</td>
<td>Peter</td>
</tr>
<tr>
<td>John</td>
<td>Tom</td>
<td>John</td>
</tr>
</tbody>
</table>

renaming - self-join

find grandparents of “Tom” (PC(p-id, c-id))
select gpa-pid
from PC as gp, PC
where gpa-pid = PC-pid
and PC-c-id = "Tom"

renaming - theta join

find course names with more units than 15-415
select c.c-name
from class as c1, class as c2
where c1.units > c2.units
and c2.c-id = "15-415"

{ t \exists c1 \in \text{CLASS} \exists c2 \in \text{CLASS} ( c1[c - id] = 15 - 415 \land c2[units] > c1[units] \land t[c - name] = c2[c - name] )}
Overview - detailed - SQL

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

set operations

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

<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>

set operations

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

(select ssn from takes where c_id='15-415')
intersection
(select ssn from takes where c_id='15-413')

other ops: \texttt{union}, \texttt{except}

Overview - detailed - SQL

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

Ordering

find student records, sorted in name order

\texttt{select *}
from student
---where---
Ordering

find student records, sorted in name order
select * from student
order by name asc

asc is the default

Ordering

find student records, sorted in name order;
break ties by reverse ssn
select * from student
order by name, ssn desc

Overview - detailed - SQL

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

Aggregate functions

find avg grade, across all students
select avg(??) from takes

<table>
<thead>
<tr>
<th>SSN</th>
<th>coid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions

• A: sum count min max (std)

find avg grade, across all students
select avg(grade) from takes

<table>
<thead>
<tr>
<th>SSN</th>
<th>coid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

result: a single number
Which other functions?
Aggregate functions

find total number of enrollments
select count(*)
from 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>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions

find total number of students in 15-415
select count(*)
from takes
where c-id="15-415"

<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>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions

find total number of students in each course
select count(*)
from takes
where ???

<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>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions

find total number of students in each course
select c-id, count(*)
from takes
group by c-id

<table>
<thead>
<tr>
<th>c-id</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-413</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>c-id</th>
<th>pop</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-413</td>
<td>2</td>
</tr>
</tbody>
</table>
Aggregate functions - ‘having’

find students with GPA > 3.0

<table>
<thead>
<tr>
<th>SSN</th>
<th>id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions - ‘having’

find students with GPA > 3.0

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

<table>
<thead>
<tr>
<th>SSN</th>
<th>id</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>15-413</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
<td>3</td>
</tr>
</tbody>
</table>

Aggregate functions - ‘having’

find students with GPA > 3.0

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

<table>
<thead>
<tr>
<th>SSN</th>
<th>avg/grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>3</td>
</tr>
</tbody>
</table>

“having” === “where” for groups

Aggregate functions - ‘having’

find students and GPA,
for students with > 5 courses

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

<table>
<thead>
<tr>
<th>SSN</th>
<th>avg/grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>4</td>
</tr>
<tr>
<td>234</td>
<td>3</td>
</tr>
</tbody>
</table>

Overview - detailed - SQL

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