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

C. Faloutsos
Rel. model - SQL part 3

General Overview - rel. model
- Formal query languages
  - Rel algebra and calculus
- Commercial query languages
  - SQL
  - QBE, (QUEL)

Overview - detailed - SQL
- DML
  - Select, from, where, renaming, ordering,
  - Aggregate functions, nested subqueries
  - Insertion, deletion, update
- Other parts: DDL, embedded SQL, auth etc

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>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ssn</td>
<td>Cx</td>
</tr>
<tr>
<td>123</td>
<td>15-413</td>
</tr>
<tr>
<td>234</td>
<td>15-413</td>
</tr>
</tbody>
</table>

DML - insertions etc

insert into student
values ("123", "smith", "main")

insert into student(ssn, name, address)
values ("123", "smith", "main")

DML - insertions etc

Bulk insertion: how to insert, say, a table of ‘foreign student’s, in bulk?
**DML - insertions etc**

- bulk insertion:

  ```sql
  insert into student
  select ssn, name, address
  from foreign_student
  ```

**DML - deletion etc**

- delete the record of 'smith':

  ```sql
  delete from student
  where name='smith'
  ```

  (careful - it deletes ALL the 'smith's!)
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>15-413</td>
</tr>
</tbody>
</table>

inner join

<table>
<thead>
<tr>
<th>TAKES</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>15-413</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-name</td>
</tr>
<tr>
<td>123</td>
<td>Sub</td>
</tr>
<tr>
<td>234</td>
<td>Sub</td>
</tr>
</tbody>
</table>

outer join

<table>
<thead>
<tr>
<th>TAKES</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>15-413</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TAKES</th>
<th>CLASS</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN</td>
<td>c-name</td>
</tr>
<tr>
<td>123</td>
<td>Sub</td>
</tr>
<tr>
<td>234</td>
<td>Sub</td>
</tr>
</tbody>
</table>

outer join

```
select ssn, c-name
from takes outer join class on takes.c-id=class.c-id
```

outer join

- left outer join
- right outer join
- full outer join
- natural join

Overview - detailed - SQL

- DML
  - select, from, where, renaming, ordering,
  - aggregate functions, nested subqueries
  - insertion, deletion, update
- other parts: DDL, embedded SQL, auth etc
Data Definition Language

create table student
(ssn char(9) not null,
 name char(30),
 address char(50),
 primary key (ssn))

Data Definition Language

create table r(A1 D1, ..., An Dn, integrity-constraint1, ...
 integrity-constraint-n)

Data Definition Language

Domains:
- char(n), varchar(n)
- int, numeric(p,d), real, double precision
- float, smallint
- date, time

Data Definition Language

integrity constraints:
- primary key
- foreign key
- check(P)

Data Definition Language

create table takes
(ssn char(9) not null,
c-id char(5) not null,
grade char(1),
primary key (ssn, c-id),
check grade in ("A", "B", "C", "D", "F")

Data Definition Language

delete a table: difference between drop table student
delete from student
Data Definition Language

modify a table:

```
alter table student drop address
```

```
alter table student add major char(10)
```

Overview - detailed - SQL

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

Embedded SQL

from within a 'host' language (eg., 'C', "VB")

```
EXEC SQL <emb. SQL stmt> END-EXEC
```

Q: why do we need embedded SQL??

Embedded SQL

SQL returns sets; host language expects a
tuple - impedance mismatch!

solution: 'cursor', i.e., a 'pointer' over the set of tuples.

e.g.,

```
example:
```

Embedded SQL

```
main0()
...
EXEC SQL declare c cursor for
select * from student
END-EXEC
...
```

Embedded SQL - ctn’d

```
... EXEC SQL open c END-EXEC ...

while( $qerror ){
  EXEC SQL fetch c into cssn, cname, cadd
  END-EXEC
  fprintf( ..., cssn, cname, cadd);
}
```
 Embedded SQL - ctn’d

  ...  
  EXEC SQL. close END-EXEC  
  ... 
  } /* end main() */

dynamic SQL

  main() { /* set all grades to user’s input */ 
  ... 
  char *sqlcmd=" update takes set grade = ?";  
  EXEC SQL. prepare dynsql from :sqlcmd ;  
  char inputgrade[5]="a";  
  EXEC SQL. execute dynsql using :inputgrade;  
  ... 
  } /* end main() */

Overview - detailed - SQL

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

SQL - misc

  Later, we’ll see  
  • authorization:  
    grant select on student to <user-id>  
  • transactions  
  • other features (triggers, assertions etc)  
  • see, e.g.:  
    http://www.cs.colorado.edu/~shadow/sql.html

General Overview - rel. model

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

Rel. model - QBE

  • Inspired by the R.D.C.  
  • “P.” -> print (ie., ‘select’ of SQL)  
  • __x, y__: domain variables (ie., attribute names)  
  • Example: find names of students taking 15-415
Rel. model - QBE

STUDENT
Ssn  Name    Address
123  smith  main str
234  forbes  ave

CLASS
s-cid  c-name  units
15-113  s.e.   2
15-12   o.s.   2

TAKES
SSN  c-id  grade
123 15-113  A
234 15-12   B

Rel. model - QBE

names of students taking 15-415

STUDENT
Ssn  Name    Address
X    P.

CLASS
s-cid  c-name  units
15-415

Rel. model - QBE

aggregate: avg grade overall:

STUDENT
Ssn  Name    Address

CLASS
s-cid  c-name  units

TAKES
SSN  c-id  grade
P.AVG.ALL

Rel. model - QBE

aggregate: avg grade per student:

STUDENT
Ssn  Name    Address

CLASS
s-cid  c-name  units

TAKES
SSN  c-id  grade
P.G.
P.AVG.ALL
General Overview - rel. model

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

Rel. model - QUEL

Used in INGRES only - of historical interest.
Eg.: find all ssn's in mini-U:

\[
\text{range of } s \text{ is student; } \\
\text{retrieve (s, ssn); } \\
\{\{s | s \text{ is student} \land s[ssn] = t[ssn]\}\
\]

Rel. model - QUEL

general syntax:
- SQL
- range of .... is i-name
- select attr. list
- retrieve (attribute list)
- from i-name
- where condition
- where condition

Rel. model - QUEL

- very similar to SQL
- also supports aggregates, ordering etc

General Overview

- Formal query languages
  - rel algebra and calculi
- Commercial query languages
  - SQL
  - QBEs (QUEL)
- Integrity constraints
- Functional Dependencies
- Normalization - “good” DB design