Today

Joins and other advanced features of SQL
- Recap
- Loose ends
- Joins
- Optimization and relational algebra
- Summary statistics in SQL
Recap

Relational model

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

In the relational model, IM stores set of relations or tables

- TELL
- INFORMATION MANAGER
- DEFINE
- ASK
**SQL Language**

- **DEFINE**
  - SQL: create table

- **TELL**
  - SQL: insert values

- **INFORMATION MANAGER**

- **ASK**
  - SQL: select queries

**Relational Model: Example**

**Definition statement**

```sql
create table books
    ( isbn char(13) not null primary key,
      author char(80),
      title char(100),
      price float(4,2)
    );
```
Relational model example

Tell statement

insert into books values
('0-672-31697-8',
 'Michael Morgan',
 'Java 2 for Professional Developers',
 34.99);

Relational model example

Tell statement
– Adds a row to the specified table in the information manager to include the specified entity or relationship.
Relational model example

Query example

SELECT author, title
FROM books
WHERE price > 30;

This returns a new table

<table>
<thead>
<tr>
<th>author</th>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Michael</td>
<td>Java 2 for Professional Developers</td>
</tr>
<tr>
<td>Morgan</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Options

not null
primary key
references table(field)

Interaction

use database;
show tables;
describe table;
Loose ends

Null values and primary keys

create table nullness (  
id integer primary key,  
stuff text  
);  

insert into nullness values  
(null, "you lose");

Loose ends

Updating an existing row

update table  
set column=expression  
where restriction
Update

changes columns in existing table rows
– *Set* clause indicates which columns to modify and the values they should be given.
– *Where* clause specifies which rows should be updated.

Example

Book gets “new edition”

Current row of books:
– isbn='0-672-31697-8',
– author='Michael Morgan',
– title='Java 2 for Professional Developers',
– price=34.99
Example

Command:

Update books
Set title = 'Java 3 for Professional Developers',
price = 39.99
where isbn='0-672-31697-8'

Result: row changed to:
- isbn='0-672-31697-8',
- author='Michael Morgan',
- title='Java 3 for Professional Developers',
- price=39.99

Loose ends

Removing an existing row

delete from table
where restriction
Example

Book goes out of print

```
delete from books
where isbn='0-672-31697-8`
```

Loose ends

Adding or deleting columns: `alter table`

Add a column, give all rows null value:

```
alter table books
add column publisher char(40)
```
Loose ends

Adding or deleting columns: alter table

Get rid of a column:

alter table books
drop column publisher

Loose ends

Adding or deleting columns: alter table

Lots of other ways to use this command.
Adding or deleting columns

Why should you not have to do this?

Loose ends

Discarding whole tables from the database

drop table books
Demo break

Any questions?

Joins - Motivation

How do you combine information from multiple tables?

Example, from book domain:

who ordered what titles?
Recap

Not useful:
select C.name, O.isbn
from customers C, order_items O

- performs cross product on tables
- no connections between rows

Recap

Need to establish relationships
select C.name, I.isbn
from customers C, orders O,
  order_items I
where C.customerid = O.customerid
and O.ordernumber = I.orderid
Joins

How you evaluate these queries is very important.

– Database designers describe algorithms using idea of a join – an operation that combines two tables together to give a new table.

Relational algebra

Describes operations to build relations

– Used in DB to represent query
– Can find equivalent expressions
– Can estimate how long evaluation will take
Selection

Extract rows from a relation

\[ \sigma_{\text{condition}}(R) \]

extract all the rows from relation \( R \) that satisfy condition

Example

Get the rows from \( S \) where rating > 8

\[ \sigma_{\text{rating}>8}(S) \]

Corresponds to

select * from \( S \) where rating > 8
**Projection**

Extract columns from a relation

$$\pi_{\text{columns}}(R)$$

make a smaller table from $R$ with just the specified *columns*

**Example**

Extract sailor names and ratings

$$\pi_{\text{sname, rating}}(S)$$

corresponds to

select sname, rating from $S$
Set operations

Union \( R \cup S \)
Intersection \( R \cap S \)
Difference \( R - S \)
Cross-product \( R \times S \)

General Joins

Accepts:
- join condition
- two relations

Returns
- new relation

\[ R \hat{A}^c S = \sigma_c(R \times S) \]
Equijoins

Join conditions contain only equalities
Duplicated fields are dropped

\[ R \Join_{R.I=S.J} S \]

Natural join: special case
– all fields in common are equated

Equivalences

Cascading of selections

\[ \sigma_{c \land d}(S) = \sigma_c(\sigma_d(S)) \]

Commutativity of selections

\[ \sigma_c(\sigma_d(S)) = \sigma_d(\sigma_c(S)) \]
Equivalences

Cascading projections

$$\pi_c(\pi_d(S)) = \pi_c(S)$$

Equivalences

Commutativity of joins

$$R \otimes S = S \otimes R$$

Associativity of joins

$$R \otimes (S \otimes T) = (R \otimes S) \otimes T$$
Equivalences

Depending on the conditions at play
Selections and projections commute

\[ \pi_c(\sigma_c(R)) \sqcup \sigma_c(\pi_c(R)) \]

Selection and join commute

\[ \sigma_c(R \otimes S) \sqcup \sigma_c(R) \otimes S \]

What this means in practice

The DB implementation can search for a good way to evaluate a query!