CMSC424: Database Design
Relational Model/SQL

Instructor: Amol Deshpande
amol@cs.umd.edu
Topics covered so far

- Why Databases
  - Data Modeling
  - Importance of abstraction/independence layers

- Relational Model
  - Relations, Tuples, Primary Keys, Foreign Keys
  - Referential Integrity Constraints
  - Relational Algebra Operations

- SQL
  - Data Definition Language: How to create relations, change schemas, etc.
  - Data Manipulation Language: Simple single-table queries, aggregates, NULLs
Today

- Advanced SQL
  - Outerjoins, Views, ...

- In-class Activity focusing on how to write SQL for complex questions
  - Make sure to turn it in !!

- No class on Monday
  - Somewhat larger reading homework + video on that material
  - Should use that time to get started on the next project
  - No office hours for me on Monday – email me if you want to meet later in the week
Outline

- More SQL
  - Outerjoins
  - Views
  - Formal semantics of SQL
  - Integrity Constraints
  - Triggers
  - Transactions
Consider the query:

```
select course_id, title, count(*)
from course natural join prereq
group by course_id, title;
```

You expect:
- A list of course id/titles with number of prereqs for each

But:
- What about courses that don’t have any prereq?
- We would like to get “(course_id, title, 0)” for them
Outer Joins

<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>dept_name</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci.</td>
<td>4</td>
</tr>
<tr>
<td>CS-315</td>
<td>Robotics</td>
<td>Comp. Sci.</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>course_id</th>
<th>prereq_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>BIO-101</td>
</tr>
<tr>
<td>CS-190</td>
<td>CS-101</td>
</tr>
<tr>
<td>CS-347</td>
<td>CS-101</td>
</tr>
</tbody>
</table>

prereq information is missing for CS-315 and course information is missing for CS-437 (wouldn’t be allowed by referential integrity constraint, but let’s say it is)

<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>dept_name</th>
<th>credits</th>
<th>prereq_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
<td>BIO-101</td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci.</td>
<td>4</td>
<td>CS-101</td>
</tr>
</tbody>
</table>
## Outer Joins

<table>
<thead>
<tr>
<th>course</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th>prereq</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>course_id</td>
<td>title</td>
<td>dept_name</td>
<td>credits</td>
<td>course_id</td>
<td>prereq_id</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
<td>BIO-301</td>
<td>BIO-101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci.</td>
<td>4</td>
<td>CS-190</td>
<td>CS-101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS-315</td>
<td>Robotics</td>
<td>Comp. Sci.</td>
<td>3</td>
<td>CS-347</td>
<td>CS-101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Natural Left Outer Join `prereq`

<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>dept_name</th>
<th>credits</th>
<th>prereq_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
<td>BIO-101</td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci.</td>
<td>4</td>
<td>CS-101</td>
</tr>
<tr>
<td>CS-315</td>
<td>Robotics</td>
<td>Comp. Sci.</td>
<td>3</td>
<td>null</td>
</tr>
</tbody>
</table>

### Natural Right Outer Join `prereq`

<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>dept_name</th>
<th>credits</th>
<th>prereq_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
<td>BIO-101</td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci</td>
<td>4</td>
<td>CS-101</td>
</tr>
<tr>
<td>CS-347</td>
<td>null</td>
<td>null</td>
<td>null</td>
<td>CS-101</td>
</tr>
</tbody>
</table>
# Outer Joins

### course

<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>dept_name</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>Genetics</td>
<td>Biology</td>
<td>4</td>
</tr>
<tr>
<td>CS-190</td>
<td>Game Design</td>
<td>Comp. Sci.</td>
<td>4</td>
</tr>
<tr>
<td>CS-315</td>
<td>Robotics</td>
<td>Comp. Sci.</td>
<td>3</td>
</tr>
</tbody>
</table>

### prereq

<table>
<thead>
<tr>
<th>course_id</th>
<th>prereq_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO-301</td>
<td>BIO-101</td>
</tr>
<tr>
<td>CS-190</td>
<td>CS-101</td>
</tr>
<tr>
<td>CS-347</td>
<td>CS-101</td>
</tr>
</tbody>
</table>

**course natural full outer join prereq**
**Additional Operators: Joins**

- **Equi-join**
  - A join that only has equality conditions

- **Theta-join ($\bowtie_\theta$)**
  - $r \bowtie_\theta s = \sigma_\theta(r \times s)$

- **Left outer join ($\bowcap$)**
  - Say $r(A, B)$, $s(B, C)$
  - We need to somehow find the tuples in $r$ that have no match in $s$
  - Consider: $(r - \pi_{r.A, r.B}(r \bowcap s))$
  - We are done:
    
    $$(r \bowcap s) \cup \rho_{temp}(A, B, C) \left( (r - \pi_{r.A, r.B}(r \bowcap s)) \times \{(NULL)\} \right)$$
# Additional Operators: Join Variations

- Tables: $r(A, B), s(B, C)$

<table>
<thead>
<tr>
<th>name</th>
<th>Symbol</th>
<th>SQL Equivalent</th>
<th>RA expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>cross product</td>
<td>$\times$</td>
<td>select * from $r, s$;</td>
<td>$r \times s$</td>
</tr>
<tr>
<td>natural join</td>
<td>$\bowtie$</td>
<td>natural join</td>
<td>$\pi_{r.A, r.B, s.C}\sigma_{r.B = s.B}(r \times s)$</td>
</tr>
<tr>
<td>theta join</td>
<td>$\bowtie_\theta$</td>
<td>from .. where $\theta$;</td>
<td>$\sigma_\theta(r \times s)$</td>
</tr>
<tr>
<td>equi-join</td>
<td>$\bowtie_\theta$ (theta must be equality)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>left outer join</td>
<td>$r \bowtie^L s$</td>
<td>left outer join (with “on”)</td>
<td>(see previous slide)</td>
</tr>
<tr>
<td>full outer join</td>
<td>$r \bowtie^F s$</td>
<td>full outer join (with “on”)</td>
<td>–</td>
</tr>
<tr>
<td>(left) semijoin</td>
<td>$r \bowtie^L s$</td>
<td>none</td>
<td>$\pi_{r.A, r.B}(r \bowtie^L s)$</td>
</tr>
<tr>
<td>(left) antijoin</td>
<td>$r \bowtie^L s$</td>
<td>none</td>
<td>$r - \pi_{r.A, r.B}(r \bowtie^L s)$</td>
</tr>
</tbody>
</table>
Outline

- More SQL
  - Outerjoins
  - Views
  - Formal semantics of SQL
  - Integrity Constraints
  - Triggers
  - Transactions
Views

- Provide a mechanism to hide certain data from the view of certain users. To create a view we use the command:

```
create view v as <query expression>
```

where:

- `<query expression>` is any legal expression
- The view name is represented by `v`

- Can be used in any place a normal table can be used
- For users, there is no distinction in terms of using it
Example Queries

- A view of instructors without their salary
  
  ```sql
  create view faculty as
  select ID, name, dept_name
  from instructor
  ```

- Find all instructors in the Biology department
  
  ```sql
  select name
  from faculty
  where dept_name = 'Biology'
  ```

- Create a view of department salary totals
  
  ```sql
  create view departments_total_salary(dept_name, total_salary) as
  select dept_name, sum(salary)
  from instructor
  group by dept_name;
  ```
Views

- Is it different from DBMS’s side?
  - Yes; a view may or may not be *materialized*
  - Pros/Cons?

- Updates into views have to be treated differently
  - In most cases, disallowed.
# Views vs Tables

<table>
<thead>
<tr>
<th>Creating</th>
<th>Create view V as (select * from A, B where …)</th>
<th>Create table T as (select * from A, B where …)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Can be used</td>
<td>In any select query. Only some update queries.</td>
<td>It's a new table. You can do what you want.</td>
</tr>
</tbody>
</table>
| Maintained as | 1. Evaluate the query and store it on disk as if a table.  
2. Don’t store. Substitute in queries when referenced. | It’s a new table. Stored on disk. |
| What if a tuple inserted in A? | 1. If stored on disk, the stored table is automatically updated to be accurate.  
2. If we are just substituting, there is no need to do anything. | T is a separate table; there is no reason why DBMS should keep it updated. If you want that, you must define a trigger. |
Views vs Tables

- Views strictly supercede “create a table and define a trigger to keep it updated”
- Two main reasons for using them:
  - Security/authorization
  - Ease of writing queries
- Perhaps the only reason to create a table is to force the DBMS to choose the option of “materializing”
  - That has efficiency advantages in some cases
  - Especially if the underlying tables don’t change
Update of a View

Add a new tuple to faculty

```
insert into faculty values ('30765', 'Green', 'Music');
```

This insertion must be represented by the insertion of the tuple

```
```

into the instructor relation

Updates on more complex views are difficult or impossible to translate, and hence are disallowed, e.g.:

- `create view instructor_info as
  select ID, name, building
  from instructor, department
  where instructor.dept_name= department.dept_name;`

- `insert into instructor_info values (’69987’, ’White’, ’Taylor’);`

Many SQL implementations allow updates only on simple views (without aggregates) defined on a single relation.
Outline

More SQL

- Outerjoins
- Views
- Formal semantics of SQL
- Integrity Constraints
- Triggers
- Transactions
Transactions

- A transaction is a sequence of queries and update statements executed as a single unit
  - Transactions are started implicitly and terminated by one of
    - *commit work*: makes all updates of the transaction permanent in the database
    - *rollback work*: undoes all updates performed by the transaction.

- Motivating example
  - Transfer of money from one account to another involves two steps:
    - deduct from one account and credit to another
  - If one steps succeeds and the other fails, database is in an inconsistent state
  - Therefore, either both steps should succeed or neither should

- If any step of a transaction fails, all work done by the transaction can be undone by rollback work.

- Rollback of incomplete transactions is done automatically, in case of system failures
In most database systems, each SQL statement that executes successfully is automatically committed.

- Each transaction would then consist of only a single statement.
- Automatic commit can usually be turned off, allowing multi-statement transactions, but how to do so depends on the database system.
- Another option in SQL:1999: enclose statements within

  ```sql
  begin atomic
  ...
  end
  ```
Outline

- More SQL
  - Outerjoins
  - Views
  - Formal semantics of SQL
  - Integrity Constraints
  - Triggers
  - Transactions
Triggers

- A trigger is a statement that is executed automatically by the system as a side effect of a modification to the database.

- Suppose that instead of allowing negative account balances, the bank deals with overdrafts by:
  1. setting the account balance to zero
  2. creating a loan in the amount of the overdraft
  3. giving this loan a loan number identical to the account number of the overdrawn account
create trigger overdraft-trigger after update on account
  referencing new row as nrow
  for each row
  when nrow.balance < 0
  begin atomic
    actions to be taken
  end
create trigger overdraft-trigger after update on account
  referencing new row as nrow
  for each row
  when nrow.balance < 0
  begin atomic
    insert into borrower
      (select customer-name, account-number
         from depositor
         where nrow.account-number = depositor.account-number);
    insert into loan values
      (nrow.account-number, nrow.branch-name, nrow.balance);
    update account set balance = 0
      where account.account-number = nrow.account-number
  end
Triggers...

- External World Actions
  - How does the DB order something if the inventory is low?

- Syntax
  - Every system has its own syntax

- Careful with triggers
  - Cascading triggers, Infinite Sequences...

- More Info/Examples:
  - Google: “create trigger” oracle download-uk