Views: Idea

- Properly normalized tables not always “convenient”

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54100</td>
</tr>
</tbody>
</table>

- Career → Last First Address

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Career</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>clifton</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>clifton</td>
<td>CS54100</td>
</tr>
</tbody>
</table>
Views: Idea

- Properly normalized tables not always “convenient”

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54100</td>
</tr>
</tbody>
</table>

- select * from course where course = ‘CS34800’
  
  – *Seems simpler than a join*

Views: Idea

- Start with normalized tables

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
</tr>
</tbody>
</table>

- Create “view” for convenience

  – create view courseList as
    
    select i.Career, Last, First, Address, Course
    from instructors I, courses c
    where i.Career = c.Career

<table>
<thead>
<tr>
<th>Career</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>CS54100</td>
</tr>
</tbody>
</table>
Views: Idea

- create view courseList as
  select i.Career, Last, First, Address, Course
  from instructors I, courses c
  where i.Career = c.Career

- courseList can now be used in a query just like a table!

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54100</td>
</tr>
</tbody>
</table>

View: Semantics

- Contents of view are current *at the time it is used*
  - If base tables are updated, view is updated
- Equivalent to replacing the view with a subquery
  
  ```
  select * from courseList where course='CS34800' ≡
  select * from
  (select i.Career, Last, First, Address, Course
  from instructors I, courses c
  where i.Career = c.Career)
  where course='CS34800'
  ```

Fall 2016
Chris Clifton - CS34800
Views: Uses

- Clarity for user / developer
  - Users see what they expect/want
  - Different views for different users/uses
    - Multiple logical views of database
- Simplification
  - “abstraction” for query
- Performance
  - Don’t need to re-run the query
- Access Control
  - Give access only to view, not entire data

SQL Access Control

- grant select on <table> to <user>;
  - grant insert, delete, update
  - with grant option
    - Allows “passing on” privileges
- <table> can also be a view
  - But some caveats on updating/insert/delete
Update issue

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54100</td>
</tr>
</tbody>
</table>

- Insert into courseList values
  ('clifton', 'Clifton', 'Chris', 'LWSN 2142F', 'CS54701');

<table>
<thead>
<tr>
<th>Career</th>
<th>Last</th>
<th>First</th>
<th>Address</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td></td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td></td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS34800</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54100</td>
</tr>
<tr>
<td>clifton</td>
<td>Clifton</td>
<td>Chris</td>
<td>LWSN 2142F</td>
<td>CS54701</td>
</tr>
</tbody>
</table>

Example Views

- 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 Defined Using Other Views

- `create view physics_fall_2009 as`
  `select course.course_id, sec_id, building, room_number from course, section where course.course_id = section.course_id and course.dept_name = 'Physics' and section.semester = 'Fall' and section.year = '2009';`

- `create view physics_fall_2009_watson as`
  `select course_id, room_number from physics_fall_2009 where building= 'Watson';`

Update of a View

- Add a new tuple to `faculty` view which we defined earlier
  `insert into faculty values ('30765', 'Green', 'Music');`
  This insertion must be represented by the insertion of the tuple
  ('30765', 'Green', 'Music', null)
  into the `instructor` relation
Some Updates cannot be Translated Uniquely

- 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');
  - which department, if multiple departments in Taylor?
  - what if no department is in Taylor?

- Most SQL implementations allow updates only on simple views
  - The from clause has only one database relation.
  - The select clause contains only attribute names of the relation, and does not have any expressions, aggregates, or distinct specification.
  - Any attribute not listed in the select clause can be set to null
  - The query does not have a group by or having clause.

And Some Not at All

- create view history_instructors as
  select *
  from instructor
  where dept_name = 'History';

- What happens if we insert ('25566', 'Brown', 'Biology', 100000) into history_instructors?
Materialized View

• Remember we crossed off Performance?
• Materialized view: Create “copy” when view is created
  – Run query and save results
  – Gives performance benefits
• Problem: Need to update when base tables updated
  – Various semantics for this, depending on DBMS