Orange Lodge Rafting Club

Jeremy Berman, Doris Hui, and Ryan Williamson
Old Logical Data Model

Association Table:
hiker_role
Ternary Relationship:
hgt
Views:
new_hiker & returning_hiker
New Logical Data Model

- **Association Table:**
  - Rafter_Job_id

- **Ternary Relationship:**
  - RCJ

- **Views:**
  - Amateur_Rafter & Professional_Rafter

- **One to Many Relationships**
  - One Rafter can only be Captain of one Crew
DDL (RCJ Table)

- Table creation with appropriate attributes for ternary relationship
- Constraints Addition in order to specify rules and ensure matching reference data
- Sequence begun at 100 with triggers in order to allow for later data manipulation
Views DDL

- Each individual view is a virtual table based on the SQL in the DDL
- Each view is separated by the “Skill” column
- Our two views for skill were either Amateur or Professional
- DDL shows the creation of the views by checking the skill value and inserting the corresponding additional information related with that view.

```sql
create view Amateur_Rafter_view as
SELECT
    Rafter_id,
    Rafter_Name,
    Rafter_Weight,
    Captain,
    Skill,
    Coach
FROM Rafter where Skill = 'Amateur_Rafter' ;

create or replace TRIGGER Amateur_Rafter_trigger
    INSTEAD OF insert ON Amateur_Rafter_view
    FOR EACH ROW
BEGIN
    insert into Rafter( 
        Rafter_id,
        Rafter_Name,
        Rafter_Weight,
        Captain,
        Skill,
        Coach)
VALUES ( 
    :NEW.Rafter_id,
    :NEW.Rafter_Name,
    :NEW.Rafter_Weight,
    :NEW.Captain,
    'Amateur_Rafter',
    :NEW.Coach) ;
END;
/
```
To add new data via a DML we need to change our original DDL sequence to 100 and need to disable constraints.

Once we truncate our tables and insert the data we need to enable the constraints again.
Ternary Relationship Select Lists

- A rafter can go on any journey with any crew as long as they are not already on that same journey with that crew. A rafter can go on an alternate journey with the same crew, or go on the same journey with a different crew.
Foreign Key Elimination Using Joins

- For the Routes page, the names of associated Rivers and Classes are displayed, and IDs are hidden from the user’s view.

```sql
select "ROUTE_ID", "ROUTE_NAME", "ROUTE_LENGTH", "RIVER_NAME", "CLASS_NAME"
from "#OWNER#"."ROUTE" join
River on route.river_id = river.river_id join Class ON
route.class_id = class.class_id
```

- For the Crews page, the name of the Captain is shown instead of the associated Rafter ID for readability.

```sql
select CREW_ID, CREW_NAME, RAFTER_NAME
from "#OWNER#".CREW join
RAFTER on CREW.captain = RAFTER.rafter_id
```
Rafters with a weight above 249 pounds are highlighted in red, and all others in green.
Chart & Chart Drill Down

Clicking on a given bar in the bar chart will return the route’s details, including route length, corresponding river, and class level.

SQL Query

```
select null, route_name, route_length as Length from Route order by Length
```
### Master Detail Report

<table>
<thead>
<tr>
<th>Rafter Name</th>
<th>Rafter Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ryan</td>
<td>More Details</td>
</tr>
<tr>
<td>Doris</td>
<td>More Details</td>
</tr>
<tr>
<td>Tim</td>
<td>More Details</td>
</tr>
</tbody>
</table>

1 - 3

<table>
<thead>
<tr>
<th>Rafter Name</th>
<th>Rafter Weight</th>
<th>Skill</th>
<th>Sponsor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tim</td>
<td>180</td>
<td>Professional Rafter</td>
<td>Gatorade</td>
</tr>
</tbody>
</table>

1 - 1
Programming Master Detail Report/Chart Drill Down

- Create Hidden Item on Master Page
- Change Column type of redirect to link and change target to desired hidden target page
- Set the hidden variable to the selected value “#RAFTER_ID#”
- Alter the SQL of the hidden output page to limit results to selected value on master page