Outline

• NULL values
• OUTER JOINs
• Constraints:
  – CHECK constraints
  – ASSERTIONS
• Triggers
NULL Values

• Field values in a tuple are sometimes unknown (e.g., a rating has not been assigned) or inapplicable (e.g., no spouse’s name).
  – SQL provides a special value null for such situations.

• The presence of null complicates many issues. E.g.:
  – Special operators needed to check if value is/is not null.
  – Is rating>8 true or false when rating is equal to null? What about AND, OR and NOT connectives?
  – We need a 3-valued logic (true, false and unknown).
  – Meaning of constructs must be defined carefully. (e.g., WHERE clause eliminates rows that don’t evaluate to true.)
  – New operators (in particular, outer joins) possible/needed.
Example: Find the names of sailors age 21 and below

```
SELECT S.sname
FROM Sailors S
WHERE S.age < 21
```

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>NULL</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

• Does this query return any row?
Outer Joins

- Regular join on sid: Sailor Lubber gets dropped.
- **Outer join**: Sailor rows without a matching Reserves row appear exactly once in the result, with the columns inherited from Reserves taking null values.
- **Left Outer Join**: Sailor rows w/o matching reservations appear in the result, but not vice versa
- **Right Outer Join**: Reservations w/o matching reservations appear in the result, but not vice versa
Example of outer join

```
SELECT S1.*, R1.*
FROM Sailors S1 NATURAL OUTER JOIN Reserves R1
```

### S1

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### R1

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

### Result

- Note the nulls

```
<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
<td>NULL</td>
<td>NULL</td>
<td>NULL</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>
```
Table or CHECK constraints

• Complex constraints over a single table
• Syntax:
  ```sql
  CHECK conditional-expression
  ```
• Example: rating must be between 1 and 10

```sql
CREATE TABLE Sailors
( sid INTEGER,
  sname CHAR(10),
  rating INTEGER,
  age REAL,
  PRIMARY KEY (sid),

  CHECK ( rating >= 1 AND rating <= 10 ))
```
Constraint: Interlake boats cannot be reserved

CREATE TABLE Reserves
    ( sid INTEGER, bid INTEGER, day DATE, 
    PRIMARY KEY (sid, bid, day), CONSTRAINT noInterlakeRes 
    CHECK (`Interlake' <> ( SELECT B.bname 
        FROM Boats B 
        WHERE B.bid=bid))))

• Recall that constraints are evaluated when a row is modified or inserted.
• Constraints can be named.
Constraints over Multiple Tables

- Example: number of boats + number of sailors < 100

```sql
CREATE TABLE Sailors ( sid INTEGER, sname CHAR(10),
                       rating INTEGER, age REAL, PRIMARY KEY (sid),
                       CHECK ( (SELECT COUNT (S.sid) FROM Sailors S)
                             + (SELECT COUNT (B.bid) FROM Boats B) < 100 )
```

- When is the constraint enforced?
- What happens if the sailors table is empty?
- Think of a case when the constraint is violated but the system never catches it.
CREATE ASSERTION

- Allows constraints that are not associated with any table.
- Evaluated whenever tables in the condition are updated

CREATE ASSERTION  smallClub
CHECK ( (SELECT COUNT (S.sid) FROM Sailors S) + (SELECT COUNT (B.bid) FROM Boats B) < 100 )
Triggers

• Trigger: procedure that starts automatically if specified changes occur to the DBMS

• Three parts:
  – Event (activates the trigger)
  – Condition (tests whether the triggers should run)
  – Action (what happens if the trigger runs)
Example of a Trigger

CREATE TRIGGER youngSailorUpdate
AFTER INSERT ON SAILORS
REFERENCING NEW TABLE NewSailors
FOR EACH STATEMENT
    INSERT INTO YoungSailors(sid, name, age, rating)
    SELECT sid, name, age, rating
    FROM NewSailors N
    WHERE N.age <= 18

• Why is “NewSailors” needed?
• What is the difference between a constraint and a trigger?
Another Example of a Trigger

• Create a trigger that will cause an error when an update occurs that would result in a salary increase greater than ten percent of the current salary.

```sql
CREATE TRIGGER RAISE_LIMIT
    AFTER UPDATE OF SALARY ON EMPLOYEE
    REFERENCING NEW AS N OLD AS O
    FOR EACH ROW
    WHEN (N.SALARY > 1.1 * O.SALARY)
    SIGNAL SQLSTATE '75000'
    SET MESSAGE_TEXT='Salary increase>10%'
```
Summary

• NULL values
  – 3-valued logic
  – Effect on WHERE-clause evaluation

• OUTER JOINs
  – Includes rows that don’t satisfy join condition

• Constraints:
  – CHECK constraints for single table constraints
  – ASSERTIONs for multi-table constraints

• Triggers
  – Allows more complex processing after certain events