Announcements

- First SQL query HW is now underway
  - Hopefully everyone has MySQL working
  - Get the latest version of the questions! (Sorry…! 😅)
- Grading is in progress for many things
  - HW #2 should be done any minute (!)
  - Other HW’s are in progress in parallel
  - Trying to get Midterm #1 done by week’s end
  - (430 is a pretty big number…)
Example Data in MySQL

**Sailors**

<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>29</td>
<td>Brutus</td>
<td>1</td>
<td>33.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>32</td>
<td>Andy</td>
<td>8</td>
<td>25.5</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>64</td>
<td>Horatio</td>
<td>7</td>
<td>35.0</td>
</tr>
<tr>
<td>71</td>
<td>Zorba</td>
<td>10</td>
<td>16.0</td>
</tr>
<tr>
<td>74</td>
<td>Horatio</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>85</td>
<td>Art</td>
<td>4</td>
<td>25.5</td>
</tr>
<tr>
<td>95</td>
<td>Bob</td>
<td>3</td>
<td>63.5</td>
</tr>
<tr>
<td>101</td>
<td>Joan</td>
<td>3</td>
<td>NULL</td>
</tr>
<tr>
<td>107</td>
<td>Johan</td>
<td>3</td>
<td>35.0</td>
</tr>
</tbody>
</table>

**Reserves**

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>1998-10-10</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>1998-10-10</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>1998-10-08</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>1998-10-07</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>1998-11-10</td>
</tr>
<tr>
<td>31</td>
<td>103</td>
<td>1998-11-06</td>
</tr>
<tr>
<td>31</td>
<td>104</td>
<td>1998-11-12</td>
</tr>
<tr>
<td>64</td>
<td>101</td>
<td>1998-09-05</td>
</tr>
<tr>
<td>64</td>
<td>102</td>
<td>1998-09-08</td>
</tr>
<tr>
<td>74</td>
<td>103</td>
<td>1998-09-08</td>
</tr>
</tbody>
</table>

**Boats**

<table>
<thead>
<tr>
<th>bid</th>
<th>bname</th>
<th>color</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Interlake</td>
<td>blue</td>
</tr>
<tr>
<td>102</td>
<td>Interlake</td>
<td>red</td>
</tr>
<tr>
<td>103</td>
<td>Clipper</td>
<td>green</td>
</tr>
<tr>
<td>104</td>
<td>Marine</td>
<td>red</td>
</tr>
</tbody>
</table>

Inner vs. Outer Joins in SQL (3)

(1) SELECT DISTINCT s.sname, r.date
    FROM Sailors s LEFT OUTER JOIN Reserves r ON s.sid = r.sid

(2) SELECT DISTINCT s.sname, r.date
    FROM Reserves r RIGHT OUTER JOIN Sailors s ON s.sid = r.sid

Variations on a theme:
- JOIN (or INNER JOIN)
- LEFT OUTER JOIN
- RIGHT OUTER JOIN
- FULL OUTER JOIN

<table>
<thead>
<tr>
<th>sname</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dustin</td>
<td>1998-10-10</td>
</tr>
<tr>
<td>Dustin</td>
<td>1998-10-08</td>
</tr>
<tr>
<td>Dustin</td>
<td>1998-10-07</td>
</tr>
<tr>
<td>Lubber</td>
<td>1998-11-10</td>
</tr>
<tr>
<td>Lubber</td>
<td>1998-11-06</td>
</tr>
<tr>
<td>Lubber</td>
<td>1998-11-12</td>
</tr>
<tr>
<td>Horatio</td>
<td>1998-09-05</td>
</tr>
<tr>
<td>Horatio</td>
<td>1998-09-08</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sname</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brutus</td>
<td></td>
</tr>
<tr>
<td>Andy</td>
<td></td>
</tr>
<tr>
<td>Rusty</td>
<td></td>
</tr>
<tr>
<td>Zorba</td>
<td></td>
</tr>
<tr>
<td>Art</td>
<td></td>
</tr>
<tr>
<td>Bob</td>
<td></td>
</tr>
</tbody>
</table>
An Algebra Side Note...

- As a side note:
  - The underlying operations are also part of the extended relational algebra, which adds...
    - Outer joins (left, right, and full)
    - Ordering (sorting)
    - Grouping (w/aggregates)
    - ....
  - You can play around with those extensions on the relational algebra (RelaX) site that you used for the recently completed RA HW (if you’re curious!)

Updates: Oh CRUD*!
(*Create, Retrieve, Update, Delete)

- Can add one or more tuples using INSERT:

  ```sql
  INSERT INTO Students (sid, name, login, age, gpa)
  VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
  ```

- Can DELETE all tuples satisfying any SQL query condition:

  ```sql
  DELETE FROM Students S
  WHERE S.sid IN (SELECT X.sid FROM Banned X)
  ```
Updates: Oh CRUD!
(Cont.)

- Can change one or more tuples using UPDATE:

  ```
  UPDATE Sailors
  SET sname = 'King Arthur',
      rating = rating + 1
  WHERE sname = 'Art';
  ```

- A few things to note:
  - LHS of SET is column name, RHS is (any) expression
  - WHERE predicate is any SQL condition, which again means SQL subqueries are available as a tool, e.g., to search for targets based on multiple tables’ content

SQL Data Integrity (Largely Review)

- An integrity constraint describes a condition that every legal instance of a relation must satisfy.
  - Inserts/deletes/updates that violate IC’s are disallowed.
  - Can be used to ensure application semantics (e.g., sid is a key, bid refers to a known boat) or prevent inconsistencies (e.g., sname has to be a string, integer age must be < 120)

- Types of IC’s: Domain constraints, primary key constraints, foreign key constraints, unique constraints, general constraints.
  - Domain constraints: Field values must be of the right type (i.e., per the schema specification). Always enforced!
SQL Data Integrity (Cont.)

- So far we have been making good use of:
  - PRIMARY KEY
  - UNIQUE
  - NOT NULL
  - FOREIGN KEY

- Other features for ensuring field value integrity:
  - DEFAULT (alternative to NULL for missing values)
  - CHECK (called “general” in the book, kind of...)

- More powerful integrity features include
  - ASSERTION (called “general” in the book, correctly 😊)
  - TRIGGER (a sledge hammer to use when all else fails!)

Note: MySQL with InnoDB actually permits a foreign key to reference any indexed column(s)...

Some Integrity Related Examples

- CHECK is useful when more general ICs than just keys are involved.
- Could use SQL subqueries to express richer constraints (if supported 😊).
- Constraints can be named (to manage them).

CREATE TABLE Sailors
(sid INTEGER,
sname CHAR(10),
rating INTEGER,
age REAL DEFAULT 18.0,
PRIMARY KEY (sid),
CHECK (rating >= 1 AND rating <= 10))

CREATE TABLE Reserves
(sname CHAR(10),
bid INTEGER,
day DATE,
PRIMARY KEY (bid, day),
CONSTRAINT noInterlakeRes
CHECK (Interlake'<>(
SELECT B.bname
FROM Boats B
WHERE B.bid=bid)))

Note: Unfortunately, MySQL currently ignores CHECK constraints…
Enforcing Referential Integrity (RI)

- Consider Sailors and Reserves; sid in Reserves is a foreign key that references Sailors.
- What should be done if a Reserves tuple with a non-existent sailor id is inserted? (A: Reject it!)
- What should be done if a Sailors tuple is deleted?
  - Also delete all Reserves tuples that refer to it, or
  - Disallow deletion of a Sailors that’s being referred to, or
  - Set sid in Reserves tuples that refer to it to some default sid.
  - (In SQL, could also: Set sid in Reserves tuples that refer to it to null, denoting ‘unknown’ or ‘inapplicable’.)
- Similar issue if the primary key of a Sailor is updated.

RI Enforcement in SQL (Reminder)

- SQL/92 and SQL:1999 support all 4 options on deletes and updates.
  - Default is NO ACTION (delete/update is rejected)
  - CASCADE (also delete all tuples that refer to the deleted tuple)
  - SET NULL / SET DEFAULT (set foreign key value of referencing tuple)

Ex:
CREATE TABLE Reserves
(sid INTEGER,
bid INTEGER,
date DATE,
....
FOREIGN KEY (sid)
REFERENCES Sailors
ON DELETE CASCADE
ON UPDATE SET NULL)

Odd combo; just illustrating what’s possible here...
**Triggers in SQL**

- Trigger: a procedure that runs automatically if specified changes occur to the DBMS
- Three parts:
  - Event (activates the trigger)
  - Condition (tests if the trigger should run)
  - Action (what happens if the trigger runs)
- Can be used to do “whatever”!
  - One SQL statement or sequence/flow of statements; can also cause the current update to bail out.
  - Details vary WIDELY from vendor to vendor (!)
  - Major source of “vendor lock-in”, along with the *stored procedure language* (= trigger action language)

**Trigger Syntax (MySQL)**

```sql
CREATE [DEFINER = { user | CURRENT_USER }] TRIGGER trigger_name
trigger_time  trigger_event
ON tbl_name
FOR EACH ROW
[trigger_order]
trigger_body
```

- `trigger_time`: { BEFORE | AFTER }
- `trigger_event`: { INSERT | UPDATE | DELETE }
- `trigger_order`: { FOLLOWS | PRECEDES } other_trigger_name

Trigger Example (MySQL)

```
DELIMITER $$
-- Necessary to make semicolons great again... 😊
-- (Prevents them from ending the input statement!)

CREATE TRIGGER youngSailorUpdate
AFTER INSERT ON Sailors
FOR EACH ROW
BEGIN
    IF NEW.age < 18 THEN
        INSERT INTO YoungSailors (sid, sname, age, rating)
        VALUES (NEW.sid, NEW.sname, NEW.age, NEW.rating);
    END IF;
END;
```

Note: `FOR EACH ROW` provides less power than `FOR EACH STATEMENT` (e.g., can’t compute average new age)

Trigger Example (MySQL, cont’d.)

- INSERT INTO Sailors(sid, sname, rating, age)
  VALUES (777, 'Lucky', 7, 77);

- INSERT INTO Sailors(sid, sname, rating, age)
  VALUES (778, 'Lucky Jr', 7, 7);

(Note: Look at YoungSailors table content after each one!)
Another Trigger Example (MySQL)

-- Let’s implement a poor man’s CHECK constraint!
DELIMITER $$

CREATE TRIGGER checkSailorAge
AFTER INSERT ON Sailors
FOR EACH ROW
BEGIN
  IF NEW.age < 18 THEN
    SIGNAL SQLSTATE '02000'
    SET MESSAGE_TEXT = 'Warning: Sailors can not be under 18!';
  END IF;
END;
END;