It's time for another installment of....

Friday Nights With Databases!
Announcements

- HW and exams:
  - HW #4 info
    - Solution now available on the CS122a wiki page
  - HW #5 info
    - Underway, on-time deadline is Tuesday (5 pm)
  - Midterm Exam 1
    - Available today! (6-7 pm if you missed 3-4 pm)

- Next week’s meetings
  - Monday is a holiday! (No class, no discussion!)

- Today’s lecture plan:
  - STILL more about SQL...!

Midterm #1 Statistics

```
<table>
<thead>
<tr>
<th>Score</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>1</td>
</tr>
<tr>
<td>65</td>
<td>1</td>
</tr>
<tr>
<td>68</td>
<td>1</td>
</tr>
<tr>
<td>71</td>
<td>1</td>
</tr>
<tr>
<td>73</td>
<td>2</td>
</tr>
<tr>
<td>75</td>
<td>2</td>
</tr>
<tr>
<td>76</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
</tr>
</tbody>
</table>

Statistics:
- Mean: 85.97
- Median: 87
- Mode: 87
- Maximum: 100
- Minimum: 60
- Std. dev.: 10.61
- # scores: 330
```

[Graph showing the distribution of midterm scores]
### Nulls and SQL's 3-Valued Logic

#### AND

<table>
<thead>
<tr>
<th></th>
<th>true</th>
<th>false</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>false</td>
<td>unknown</td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>false</td>
<td>unknown</td>
</tr>
</tbody>
</table>

#### OR

<table>
<thead>
<tr>
<th></th>
<th>true</th>
<th>false</th>
<th>unknown</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
<td>unknown</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
</tr>
</tbody>
</table>

#### NOT

<table>
<thead>
<tr>
<th></th>
<th>false</th>
<th>true</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>unknown</td>
<td>unknown</td>
<td>unknown</td>
</tr>
</tbody>
</table>

**Note:** SQL arithmetic expressions involving null values will yield null values (Ex: EMP.sal + EMP.bonus)

(Ex: Sailors w/Some Null Values)

<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>Johannes</td>
<td>NULL</td>
<td>35.0</td>
</tr>
</tbody>
</table>
**Basic SQL Queries w/Nulls**

```sql
SELECT * 
FROM Sailors S 
WHERE age > 35.0
```

```sql
SELECT * 
FROM Sailors S 
WHERE age <= 35.0
```

```sql
SELECT COUNT(*) 
FROM Sailors S 
WHERE age > 35.0 
OR age <= 35.0 
OR age IS NULL
```

---

**Ex: Sailors and Reserves w/Nulls**

```sql
SELECT * 
FROM Sailors S 
WHERE age > 35.0 
OR age <= 35.0 
OR age IS NULL
```
**Nulls w/Aggregates**

```
SELECT COUNT(rating)
FROM Sailors
(11)
```

```
SELECT COUNT(DISTINCT rating)
FROM Sailors
(7)
```

```
SELECT SUM(rating), COUNT(rating), AVG(rating)
FROM Sailors
(70, 11, 6.3636)
```

(Useful – but logically “wrong”!)

---

**Nulls w/Aggregates & Grouping**

```
SELECT COUNT( DISTINCT bid)
FROM Reserves
(4)
```

```
SELECT bid, COUNT(*)
FROM Reserves
GROUP BY bid
```

```
bid   COUNT(*)
NULL  2
101   2
102   3
103   4
104   2
```

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

9

10
Nulls w/Joins $\Rightarrow$ Inner/Outer Joins

<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>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>Johannes</td>
<td>NULL</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<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>1998–10–08</td>
</tr>
<tr>
<td>22</td>
<td>103</td>
<td>1998–10–07</td>
</tr>
<tr>
<td>22</td>
<td>104</td>
<td>1998–11–10</td>
</tr>
<tr>
<td>31</td>
<td>102</td>
<td>1998–11–06</td>
</tr>
<tr>
<td>31</td>
<td>103</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>
<tr>
<td>1</td>
<td>NULL</td>
<td>2001–01–11</td>
</tr>
<tr>
<td>1</td>
<td>NULL</td>
<td>2002–02–02</td>
</tr>
</tbody>
</table>

Inner vs. Outer Joins in SQL

```
SELECT DISTINCT s.sname, r.date
FROM Sailors s, Reserves r
WHERE s.sid = r.sid
```
**Inner vs. Outer Joins in SQL (2)**

SELECT DISTINCT s.sname, r.date  
FROM Sailors s INNER JOIN Reserves r  
ON s.sid = r.sid

<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>

**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

(Varies from RDBMS to RDBMS)

(See: [http://dev.mysql.com/doc/refman/5.7/en/join.html](http://dev.mysql.com/doc/refman/5.7/en/join.html) for MySQL’s join syntax)
A Few Side Notes...

- As a side note:
  - The underlying operations are also part of the extended relational algebra, which adds...
    - Outer joins
    - Grouping
    - Ordering
    - ...
  - You can actually play with those extensions on the relational algebra RelaX site that we used for the last HW assignment (if 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:
  
  ```sql
  UPDATE Sailors
  SET sname = '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

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

- 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!)

Some Integrity Related Examples

- CHECK 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 ( (SELECT B.bname FROM Boats B WHERE B.bid=bid)<> (Interlake') )
```
Enforcing Referential Integrity (RI)

- Consider Sailors and Reserves; sid in Sailors is a foreign key that references Reserves.

- 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.
  - Disallow deletion of a Sailors tuple that’s being referred to.
  - Set sid in Reserves tuples that refer to it to a default sid.
  - (In SQL, also: Set sid in Reserves tuples that refer to it to null, denoting ‘unknown’ or ‘inapplicable’.)

- Similar if primary key of Sailors tuple 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)
One more SQL lecture ahead....