SQL: Modifications and Transactions

- Problem Set #1 is due before midnight tonight.
Creating Tables

❖ CREATE TABLE is the paramount SQL command
❖ The combination of all create table commands define the database’s schema
❖ Most Integrity Constraints (ICs) are specified as part of CREATE TABLE

```sql
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] TableName (  
    Attr1 type [PRIMARY KEY [AUTOINCREMENT]] [DEFAULT value],  
    Attr2 type [DEFAULT value],  
    ...  
    AttrN type [DEFAULT value],  
    [PRIMARY KEY (AttrX,AttrY, ...), -- composite key]  
    [FOREIGN KEY(AttrX) REFERENCES Table(AttrY)  
        [ON DELETE Action],]  
    [UNIQUE (AttrX,AttrY, ...),]  
    [CHECK (expr),]  
)  
```
Official SQL Syntax

- From sqlite.org/lang_createtable.html

These diagrams are often useful when composing and debugging queries.
Example Create Table commands

❖ For our Yacht club:

CREATE TABLE IF NOT EXISTS Sailors (  
    sid INTEGER PRIMARY KEY AUTOINCREMENT,  
    sname TEXT NOT NULL,  
    rating INTEGER DEFAULT 1,  
    age REAL NOT NULL,  
    CHECK ((rating >= 1) AND (rating < 10))  
)  

CREATE TABLE Boats (  
    bid INTEGER PRIMARY KEY AUTOINCREMENT,  
    bname TEXT NOT NULL,  
    color TEXT DEFAULT ''  
)  

CREATE TABLE Reserves (  
    sid INTEGER NOT NULL,  
    bid INTEGER NOT NULL,  
    day DATE,  
    PRIMARY KEY (sid,bid,day),  
    FOREIGN KEY(sid) REFERENCES Sailors(sid),  
    FOREIGN KEY(bid) REFERENCES Boats(bid) 
)
Creating tables from queries

- Relations can be derived from other tables
  
  ```sql
  CREATE TEMPORARY TABLE BoatUses AS
  SELECT bid, COUNT(bid) AS uses
  FROM Reserves
  GROUP BY bid;
  ```

- And “SELECT * FROM BoatUses” gives:
  - Can serve as temporary relations used in complex transactions
  - Can lead to redundancy and inconsistency
  - Has no ICs

<table>
<thead>
<tr>
<th>bid</th>
<th>uses</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>2</td>
</tr>
<tr>
<td>102</td>
<td>3</td>
</tr>
<tr>
<td>103</td>
<td>3</td>
</tr>
<tr>
<td>104</td>
<td>2</td>
</tr>
</tbody>
</table>
Altering Tables

❖ Schemas can be modified and ICs added to an existing table
❖ Add a new “made” column to track the day that a reservation is made on

   \[
   \text{ALTER TABLE Reserves ADD COLUMN } \\
   \text{made DATE CHECK (made <= day)}
   \]

❖ Note: CHECK constraints are not teed against preexisting tuples in the table
❖ Rename an existing table

   \[
   \text{ALTER TABLE PlayedFor RENAME TO Roster}
   \]
Dropping Tables

❖ DROP TABLE removes a relation from a database. It is completely removed—its definition and tuples, and it can not be recovered.

❖ If FOREIGN KEY constraints are defined, a DROP TABLE will generate DELETE FROM commands for each tuple.

DROP TABLE Boats

If the RESERVES relation had a FOREIGN KEY(bid) REFERENCES Boats ON DELETE action, it would be executed.
The INSERT command adds tuples to the database. ICs are checked.

```
INSERT INTO Sailors(sid, sname, rating, age)
VALUES(81, "Dusty", 5, 23.0)
```

If all attributes are included in order the following simple form can be used

```
INSERT INTO Sailors
VALUES(80, "Crusty", 6, 32.0)
```

Fails if any IC is violated, i.e. repeating a primary key

```
INSERT INTO Sailors
VALUES(81, "Dusty", 6, 24.0)
```
Replace

❖ Can use REPLACE to change an existing tuple (primary key must appear)

  REPLACE INTO Sailors
  VALUES(81, "Dusty", 6, 24.0)

❖ “INSERT OR REPLACE” inserts a new tuple if the primary key does not already appear, and replaces a tuple if it does

  INSERT OR REPLACE INTO Sailors
  VALUES(81, "Dusty", 6, 24.5)
Update

❖ If a only a subset of relation attributes are specified in an INSERT or REPLACE command the remainder are set according to their DEFAULT clause.

❖ If one desires to change selected attributes of a tuple, the UPDATE command is provided.

```
UPDATE Sailors
SET age = age + 1
WHERE sname like "_us%"
```

```
UPDATE Sailors
SET rating = rating + 1
WHERE rating < 10

UPDATE Sailors
SET age = 46.0, rating = 10
WHERE sid = 22
```
Delete

❖ DELETE removes entire tuples from a relation that satisfy an optional condition

```
DELETE FROM Sailors
WHERE age > 5 * rating
```

❖ DELETE without a condition removes all tuples but retains the table’s definition (contrast with drop table)

❖ DELETE may cause side-effects depending on ICs
Database Transactions

❖ An important database concept
❖ Provides concurrency and durability
❖ A transaction consists of a sequence of SQL commands that might potentially change the contents of the database.
❖ These commands are considered as atomic
  ▪ Final contents of the database are as if each command was executed in sequence with no intervening changes to the database’s contents
  ▪ All or none of the commands are executed
  ▪ Database can be “Rolled back” to a state as if none of the transaction’s commands were executed
Begin, Commit, and Rollback

- No changes are made to the database until a transaction is committed.
- Any command that changes the database implicitly starts a transaction if one is not already in effect.
- One can explicitly start a transaction with the BEGIN TRANSACTION command.
- Commands within a transaction can access the intermediate changes and results of previous commands, but they do not appear in the database until an explicit COMMIT TRANSACTION command.
- If during a transaction a user decides to abort that series of changes made, a ROLLBACK TRANSACTION command be used.
Summary

- SQL provides commands for describing, querying, and modifying a database.
- A database’s schema and integrity constraints are defined by CREATE TABLE commands.
- Tuples are inserted into relations via the INSERT and REPLACE commands, and removed using DELETE.
- Specific attributes of a relation’s tuples are modified using UPDATE.
- A transaction groups a set of commands into a single “atomic” operation.