ALTERATIONS AND TRANSACTIONS

Jed Rembold

Monday, October 14, 2024

ANNOUNCEMENTS

- Homework 6 due on Thursday
    - Mostly groups and data modeling, so you have everything already that you need
- I am working on Homework 5 feedback
- Note that we are skipping Chapter 11, so we'll be looking at Ch 12 material on Wednesday
- Terminal `psql` issues? I posted something on Discord, but will summarize.
- Polling today: polling.jedrembold.prof

CONTINUING ALTERATIONS

MULTI-UPDATES

- Sometimes you want to update several things at once
- So long as they are all in the same table, you can do this with a single `UPDATE` statement
- After the `SET` keyword, either:
  - Separate each assignment by a comma: `SET col1 = 5, col2 = col3`
  - Pair up the assignments with parentheses: `SET (col1, col2) = (5, col3)`

UNDERSTANDING CHECK

The table named `revq` to the right is acted upon by the below SQL queries. What entries in the table are left untouched once all queries have been run?

row1 *TEXT*	row2 *REAL*	row3 *INT*
A	*0.24*	*15*
B	*9.1*	*4*
C	*4*	*10*

A
B
C
D

```sql
ALTER TABLE revq ADD COLUMN row4 INT;
UPDATE revq SET row4 = row2;
UPDATE revq SET (row2,row3)=(row3, row2) WHERE row1 IN ('B','C');
UPDATE revq SET row3 = row3 - row4;
UPDATE revq SET row2 = row2 + row4 WHERE row3 > 10;
ALTER TABLE revq DROP COLUMN row4;
```

BACKUP TABLES

- Frequently, if you are about to heavily modify a table, you should consider working on a backup copy
- We actually have already seen the basic machinery for this:

```
CREATE TABLE new_table AS
SELECT * FROM og_table;
```

  - Note: Indexes and constraints are stored separately, and so are NOT copied over using this process!

- For including constraints and indexes, you can use a Postgres specific syntax, but the newly created table will initially be missing the data

```
CREATE TABLE new_table
(LIKE og_table INCLUDING ALL);
```

TABLE TO TABLE

- In some cases, you'll want to update or pass information across tables
  - Maybe one table has newer values that you want to use to update the original table
- In core SQL, you'd need to use subqueries, which we'll be talking about in a few chapters
- In Postgres, to update, you can use `FROM`:

```
UPDATE table_1
SET col_name = table_2.col2
FROM table_2
WHERE table_1.col1 = table_2.col1;
```

- To insert values from another table into another:

```
INSERT INTO new_table SELECT * FROM old_table;
```

DELETIONS

- Similar to changing tables, removing things from tables has two main keywords:
  - `DROP` for removing structural aspects of a table like columns, constraints, indexes, or the table itself
  - `DELETE FROM` for removing content (rows) from tables
- `DROP` will frequently come after an `ALTER TABLE` unless you are dropping the table itself
- `DELETE FROM` without a filter will delete all rows
  - Make absolutely sure you are using a filter if you don’t want that to happen!
  - Another good reason to back up your tables before editing them

GETTING DELETED

```sql
ALTER TABLE tname DROP COLUMN colname;
ALTER TABLE tname DROP CONSTRAINT const_name;
DROP INDEX index_name;
DROP TABLE tname;

DELETE FROM tname; -- All rows gone!
DELETE FROM tname WHERE condition;
```

- In general, unless you have an important reason, don’t remove actual data from a table
  - You can filter it, you can create new tables that are missing that data, etc.

ACID TRANSACTIONS

TRANSACTIONS

- *Atomicity* is an important aspect of most database changes
  - The idea that related changes should happen in a single, self-contained step
- Many changes you might make to a database have several steps though!
  - Need to change one value in one table and another value in another table
  - Need to create a new row and then copy some information into it
- Remember that, in general, others can access the database at the same time
  - What if they tried to access the data you were working on mid-operation?
- So solve these issues, SQL has the concept of a *transaction*

BUNDLING UP

- A *transaction* is essentially a bundling of several statements into one, discrete change to the database
- Commands within the transaction have not yet modified the database, but exist only in local memory
- Changes get written to the database all at once upon the conclusion of the transaction
- Starting a transaction?
  - `START TRANSACTION;` or `BEGIN;`
- Ending a transaction?
  - `COMMIT;` actually makes the changes
  - `ROLLBACK;` throws out everything within the transaction

USES OF TRANSACTIONS

- Protecting against system faults
  - What if you have a system crash in the middle of an operation?
    - What commands had been run? What commands had not?
  - Transactions actually write to a log what they are *going* to do before they actually do it. So in case of a crash, then the transaction can then simply be rerun
- Protecting against simultaneous access
  - Changes occur all at once, so it is impossible for another database user to access data "mid-change"
  - Other users of the database will see none of your changes until actually committed
- Testing changes
  - Sometimes it is useful to check to see that some changes look the way you wanted before actually changing the database
  - Embedding within a transaction block always gives you the option to rollback

PRESIDENTIAL CLEANING

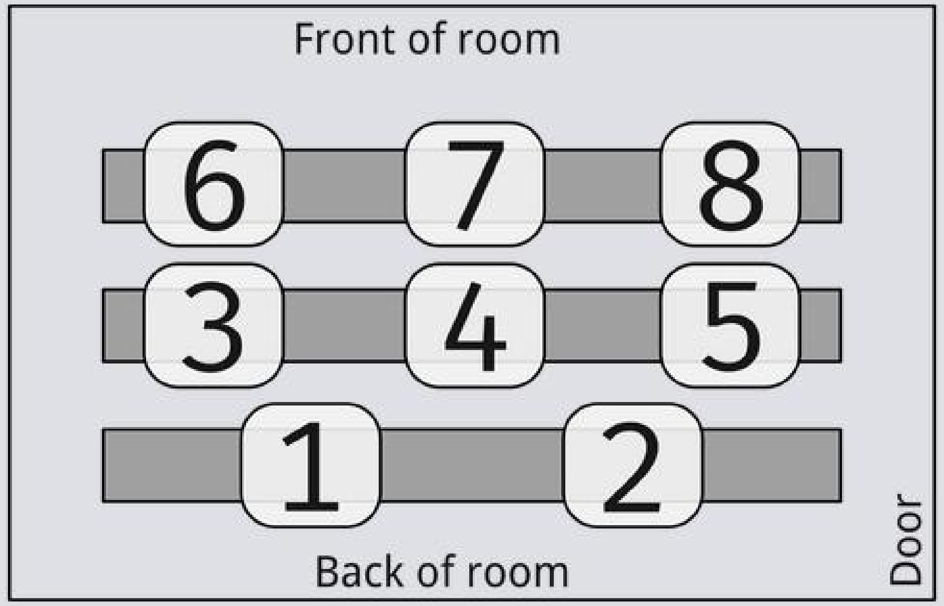
PRACTICE ACTIVITY

- There is a simple CSV of presidents and debt here
- It has some data consistency problems which you should determine and fix before answering the following questions:
  - What are the top 5 presidents to have the greatest average annual increase in national debt over the years of their presidency?
  - How do the median values of annual increases in national debt compare across party lines?
  - Trickier: What is the average *change* in the annual increase percentage of national debt overall all the years?

GROUPS

- You'll be working in small groups, only 1 computer interacting with a database
  - Others can have slides, documentation, etc. open

- Groups:
  - Group 1: Connor, Hannah, Aurora
  - Group 2: Tiffany, Michael, Jordan
  - Group 3: Sergio, Jack, Tippy
  - Group 4: Nick, Jerrick, Myles, Matthew
  - Group 5: Grace, Greg, Sam H
  - Group 6: Dayton, Finn, Sam J
  - Group 7: Evan, Marcus, AJ
  - Group 8: Haley, Mallory, Harleen



Group Locations

QUESTION ANSWERS

- What are the top 5 presidents to have the greatest average annual increase in national debt over the years of their presidency?
  - Reagan, HW Bush, Ford, Carter, Obama
- How do the median values of annual increases in national debt compare across party lines?
  - Democrats: 3.65%
  - Republicans: 7.40%
- Trickier: What is the average *change* in the annual increase percentage of national debt overall all the years?
  - Only about 0.05%, but seemingly a slight steady increase