# Nearly Final SQL: Triggers; WITH; Views

## Triggers

Procedure that runs automatically if specified changes in DBMS happen

```
CREATE TRIGGER name
```

Event activates the trigger
Condition tests if triggers should run
Action what to do

## Triggers: Uses

Constraints (e.g. at least one)

Copy/fill data based on other tables (e.g. purchase an item; copy the current price into the purchase)

Record history of every update

## Triggers: Capabilities

Prevent an insert/update/delete (constraint)

Change the value being updated

Execute arbitrary user defined functions

## Triggers

```
CREATE TRIGGER name
    [BEFORE | AFTER | INSTEAD OF] event_list ON table
```

Event activates the trigger
Condition tests if triggers should run
Action what to do

## Triggers

```
CREATE TRIGGER name
    [BEFORE | AFTER | INSTEAD OF] event_list ON table

    WHEN trigger_qualifications
```

Event activates the trigger
Condition tests if triggers should run
Action what to do

## Triggers

```
CREATE  TRIGGER  name
    [BEFORE  | AFTER  | INSTEAD  OF]  event_list   ON  table
    [FOR  EACH  ROW]
    WHEN  trigger_qualifications
    EXECUTE  PROCEDURE   procedure
```

Event activates the trigger

Condition tests if triggers should run

Action what to do

## Copy updates into log table

```
CREATE TABLE log(
  sid int NOT NULL REFERENCES Sailors,
  t timestamp NOT NULL,
  oldAge int NOT NULL,
  newAge int NOT NULL
);
```

## Copy updates into log table

```
CREATE FUNCTION logFunc() RETURNS trigger AS
$$
BEGIN
  INSERT INTO log VALUES
    (NEW.sid, now(), OLD.age, NEW.age);
  RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

## Copy updates into log table

```
CREATE TRIGGER logChanges
AFTER UPDATE OF age ON Sailors
FOR EACH ROW EXECUTE PROCEDURE logFunc();
```

## Copy updates into log table

```
 sid |              t             | oldage | newage
-----+----------------------------+--------+-------
   1 | 2016-02-27 18:14:47.792261 |     22 |     23
(1 row)
```

## Triggers

Can be complicated to reason about

- Triggers may cause other triggers to run (recursive)
  - (e.g. trigger on sailors inserts into sailors?)

If >1 trigger match an action, which is run first?

¯_(ツ)_/¯

Arbitrary code: can't be optimized by DB

## Triggers vs Constraints

Constraint
- Statement about state of database
- Doesn't modify the database state
- Somewhat "understood" by the database

Triggers
- Operational: X should run when Y
- *Very* flexible
- Must be executed on every matching statement

## WITH (Common Table Expressions)

Large queries can get very complicated

Useful to name parts of these queries

(Rare but useful to know this exists)

## WITH

```
WITH  RedBoats(bid,   count)  AS
    (SELECT    B.bid,   count(*)
     FROM      Boats  B,  Reserves   R
     WHERE     R.bid  =  B.bid  AND  B.color  =  'red'
    GROUP  BY  B.bid)
SELECT     name,  count
FROM       Boats  AS  B,   RedBoats  AS  RB
WHERE      B.bid  =  RB.bid  AND  count  <  2
```

Names of unpopular red boats

## Views

```
CREATE VIEW view_name
AS select_statement
```

"tables" defined as query results rather than inserted base data
- Development: continue to run old apps
- Security: Grant limited access

References to *view_name* replaced with *select_statement*

Similar to WITH, lasts longer than one query

Updates: Tricky (Postgres: not permitted without triggers)

## Names of popular boats

```
CREATE VIEW boat_counts
AS  SELECT     bid, count(*)
    FROM       Reserves R
    GROUP BY   bid
    HAVING     count(*) > 10
```

Used like a normal table

```
SELECT  B.name
FROM    boat_counts bc, Boats B
WHERE   bc.bid = B.bid
```

Names of popular boats

```
SELECT  B.name
FROM
     (SELECT bid, count(*)
      FROM Reserves R
      GROUP BY bid
      HAVING count(*) > 10) bc,
     Boats B
WHERE   bc.bid = B.bid
```

Rewritten expanded query

## CREATE TABLE AS

Create table from a query

```
CREATE TABLE <table_name> AS
    <SELECT STATEMENT>
```

```
CREATE TABLE used_boats1 AS
    SELECT r.bid
    FROM   Sailors s,
           Reservations r
    WHERE  s.sid = r.sid
```

used_boats1(bid int)

```
CREATE TABLE used_boats2 AS
    SELECT r.bid as foo
    FROM   Sailors s,
           Reservations r
    WHERE  s.sid = r.sid
```

used_boats2(foo int)

How is this different than views?
- What if we insert a new record into Reservations?