Constraints review

- Constraints we've discussed:
  - ???
Referential triggered actions

- **Trigger conditions:**
  - ON DELETE
  - ON UPDATE

- **Actions:**
  - CASCADE
  - SET NULL
  - SET DEFAULT

```sql
CREATE TABLE Students
(
  ID INTEGER,
  Name VARCHAR(20),
  Major VARCHAR(10),
  GPA DECIMAL(3,2),
  CONSTRAINT Students_PK PRIMARY KEY (ID)
);

CREATE TABLE Enrollment
(
  Stud_ID INTEGER,
  Course VARCHAR(15),
  CONSTRAINT Enrollment_FK FOREIGN KEY (Stud_ID) REFERENCES Students(ID)
  ON DELETE CASCADE
);
```
The chicken and the egg problem...

- CREATE TABLE Chicken (ID INT PRIMARY KEY,
  eID INT REFERENCES Egg(ID));
- CREATE TABLE Egg(ID INT PRIMARY KEY,
  cID INT REFERENCES Chicken(ID));

- We can't create these tables using these commands!
  - Do we know commands that could create these tables?
- But how can we insert into either table??
  - Need to treat two inserts into both tables as one logical unit of work...
Transactional DBMS properties

- ???
Transactions

- Begin Transaction
- Active
- Partially committed
- Committed
- Failed
- Terminated
- Read /Write
- End
- Commit
- Abort
Basic outline of transactions in SQL

- **START TRANSACTION** [ *transaction_mode*, ... ]
  - Where *transaction_mode* is one of:
    - READ WRITE | READ ONLY
    - [ NOT ] DEFERRABLE
    - ISOLATION LEVEL { SERIALIZABLE | REPEATABLE READ | READ COMMITTED | READ UNCOMMITTED }

- **SET TRANSACTION** [ *transaction_mode*, ... ]
  - Modifies set *transaction_mode*

- **COMMIT**;
  - End the transaction, committing all changes as permanent

- **ROLLBACK**;
  - End the transaction, undoing all changes made

We'll come back to these two...
Standard SQL transactions vs PostgreSQL

- START TRANSACTION … should be unnecessary according to the SQL standard
  ○ Each SQL statement should implicitly start a transaction
- In PostgreSQL, you cannot effectively have a multi-statement transaction without issuing a START TRANSACTION
  ○ However, each SQL statement still implicitly starts a transaction
    - Unless START TRANSACTION is issued, PostgreSQL implicitly issues a COMMIT after each SQL statement
      - This functionality is sometimes referred to as autocommit
Constraints within a transaction

● Demo time

○ `CREATE TABLE Students`  
  ( ID INTEGER,  
    Name VARCHAR(20),  
    Major VARCHAR(10),  
    GPA DECIMAL(3,2),  
    CONSTRAINT Students_PK  
    PRIMARY KEY (ID)  
  )

○ `CREATE TABLE Enrollment`  
  ( Stud_ID INTEGER,  
    Course VARCHAR(15),  
    CONSTRAINT Enrollment_FK  
    FOREIGN KEY (Stud_ID) REFERENCES Students(ID)  
    ON DELETE CASCADE  
  )
Keywords that can be added to set how constraints are evaluated within a transaction

- Constraints that are NOT DEFERRABLE are immediately evaluated after each SQL statement
- DEFERRABLE constraints can be evaluated after the transaction is ended, before committing

  - If DEFERRABLE, constraint can INITIALLY be:
    - DEFERRED
      - Evaluated after the transaction is ended
    - IMMEDIATE
      - Evaluated after each SQL statement

  - Initial setting can be changed within a transaction
    - SET CONSTRAINTS \textit{cname} [,...] \{DEFERRED \mid IMMEDIATE\};
    - SET CONSTRAINTS ALL \{DEFERRED \mid IMMEDIATE\};
Constraints within a transaction take 2

- **Demo time**
  - CREATE TABLE Students
    (  ID INTEGER,
    Name VARCHAR(20),
    Major VARCHAR(10),
    GPA DECIMAL(3,2),
    CONSTRAINT Students_PK
      PRIMARY KEY (ID)
    );
  - CREATE TABLE Enrollment
    (  Stud_ID INTEGER,
    Course VARCHAR(15),
    CONSTRAINT Enrollment_FK
      FOREIGN KEY (Stud_ID) REFERENCES Students(ID)
      ON DELETE CASCADE
      DEFERRABLE INITIALLY DEFERRED
    );
CREATE TABLE Chicken (ID INT PRIMARY KEY, eID INT);
CREATE TABLE Egg(ID INT PRIMARY KEY, cID INT);

ALTER TABLE Chicken ADD CONSTRAINT Chicken_FK
FOREIGN KEY (eID) REFERENCES Egg(ID)
DEFERRABLE INITIALLY IMMEDIATE;

ALTER TABLE Egg ADD CONSTRAINT Egg_FK
FOREIGN KEY (cID) REFERENCES Chicken(ID)
DEFERRABLE INITIALLY IMMEDIATE;