Name 1: Date: June 27, 2012
Name 2:

Exercise 7.2.3: Write the following constraints as tuple-based CHECK constraints on one of the relations:

Movies(<u>title, year</u>, length, genre, studioName, producer#)
StarsIn(<u>movieTitle, movieYear</u>, starName)
MovieStar(<u>name</u>, address, gender, birthdate)
MovieExec(<u>name</u>, address, cert#, netWorth)
Studio(<u>name</u>, address, presC#)

if the constraint actually involves two relations, then you should put the constraints in both relations so that whichever relation changes, the constraint will be checked on insertions and updates. Assume no deletions.

1. No two studios might have the same address.
2. The year of a movie cannot be before 1900.
3. A star may not appear in a movie made before they were born
4. A movie star cannot be a movie executive.

1) Alter Table Studio ADD CONSTRAINT uniqst UNIQUE (address)

You can also append unique to the field definition in create table.

2) ALTER TABLE Movies ADD constraint after1900 CHECK (year >= 1900);

3) ✓ ✓ StarsIn ✓ ✓ notbefore

CHECK (movieyear >= (select birthdate from stars where name = starname));

NOT supported by postgres.

4) Add constraint to movieStar

CHECK (NOT EXISTS (select * from movieExec E where E.name = name));

and to movieExec

CHECK (NOT EXISTS (select * from movieStar S where S.name = name));