# CECS 323 LAB BCNF

**OBJECTIVE:** Get some first-hand experience with moving from 3NF to BCNF.

**INTRODUCTION:** We are going to revisit Jo Anne's Fabrics and Crafts with a couple of simplifying assumptions.

- Each District includes several cities.
- A city belongs to only one District.
- We will not worry about the Region and state level.
- Assume that the city has a city ID surrogate so that we do not have to worry about migrating the state code and city name around.
- Each store is within both a city and a district.
- The store name (just called store here) is only unique within a district. But since the city functionally determines the district, the store name is also unique within a city.

Assume that one city could have multiple stores in it. That tells you that the city does not functionally determine the store.

Start with the relation: JoAnneStore: {city_ID, district_name, store_name, store_manager, store_revenue}. The functional dependencies are:

- {city_ID} → {district_name}
- {district_name, store_name} → {store_manager, store_revenue, city_ID}
- {city_ID, store_name} → {store_manager, store_revenue, district_name}

Both city_ID and district_name are part of a candidate key, so the original relation is in 3rd Normal Form. Remember the homey definition of 3rd NF is "every non key attribute depends upon the key, the whole key, and nothing but the key". {city_ID} is not a superkey, but that is not a problem for 3rd NF since district_name is part of a candidate key. Unfortunately, it still has some redundancy:

| City_ID | District_name | Store_name | Store_manager | Store revenue |
|---|---|---|---|---|
| 1 | South Bay | Palo Verde | Eliza Doolittle | $500,005.82 |
| 1 | South Bay | Towne Center | Milly Cyrus | $382,234.88 |
| 2 | South Bay | Fashion Center | David Duchovny | $128,000.83 |
| 2 | South Bay | South Mall | Michael J. Fox | 512,384.22 |

- Clearly, the correspondence between the city_ID and the district_name needs to come out into a separate table to bring an end to this senseless redundancy. But if we do that, the city_id will migrate into the new JoAnneStore relation, and district_name will remain in the new City table. Now we can no longer make sure that we support the {city_ID, store_name} → {store_manager, store_revenue, district_name}. When this was all one relation, we could just put a uniqueness constraint across {city_ID, store_name}. But that's not open to us anymore.

# CECS 323 LAB BCNF

**PROCEDURE:** Build the two tables: JoAnneStore and City with the proper foreign key constraint from City to JoAnneStore.

1. Come up with a trigger to enforce the functional dependency: {district_name, store_name} → {store_manager, store_revenue, city_ID}
2. Load up your tables with some convincing data and demonstrate that your trigger prevents a violation of that functional dependency for inserts.
3. Then do the same thing for updates.
4. If it has not already occurred to you, build a stored procedure that your on update trigger and your on insert triggers will call, so that you do not commit the mortal sin of redundant coding.

**WHAT TO TURN IN:**

- The DDL for the two tables.
- The DML that you use for inserting rows into the two tables.
- The code for your triggers and the stored procedure.
- A demonstration that the triggers work:
  - A table of the data in the two tables when you start the insert to test the trigger.
  - The console output from the trigger failing.
- Your team's filled out collaboration document. You can find a template for that here.