CS144 Notes: Relational Table Design

- Main question: How do we design "good" tables for a relational database?

Boyce-Codd Normal Form (BCNF)

- Example:

    Students take classes
    StudentClass(sid, name, addr, dept, cnum, title, unit)

- Q: Is it a good table design?

    <Example instance slides>

- REDUNDANCY: The same information mentioned multiple times
    - Redundancy leads to potential anomaly
        * update anomaly, insertion anomaly, deletion anomaly

- Q: Is there a better design? What tables would you use?

Let us study this question more carefully

- Q: Where is the redundancy from?
    - (Slide on "guessing" missing info)

- Some attributes are "determined" by other attrs: FUNCTIONAL DEPENDENCY
    - ex) sid -> (name, addr), (dept, cnum) -> (title, unit)

- Some notations for formal definition
    - u[X] - values for the attributes X of tuple u
    - <e.g, u = (sid: 100, name: James, addr: Wilshire) u[sid, name] = (100, James)>

- FUNCTIONAL DEPENDENCY X -> Y (where X = A1, ..., An, Y = B1, ..., Bm):
    - For any u1, u2 in R, if u1[X] = u2[X], then u1[Y] = u2[Y]
    - No two tuples in R can have the same X values but different Y values

    - <e.g., StudentClass(sid, name, addr, dept, cnum, title, unit)>

- Q: sid -> name?

- Q: dept, cnum -> title, unit?

- Q: dept, cnum -> sid?

- Functional dependency and redundancy

```
sid -> name, addr
dept,cnum -> title,unit
```

  (301, James, 11 West) is stored redundantly. So is (cs, 143, database, 04).

  - NOTE: When there is a FD, we may have redundancy.

- DECOMPOSITION: When there is a FD, no need to store this relationship multiple times. Store it once in a separate table

```
<StudentClass example>
       <Intuitively explain normalization of StudentClass table>
       Two FDs: sid->(name,addr),  (dept,cnum)->(title,unit)
```

  1. sid -> (name, addr) no need to store it multiple time. separate it out

  2. (dept, cnum) -> (title, unit). separate it out

- Basic idea of NORMAL FORM and DECOMPOSITION
  - Whenever there is a FD, the table may be "bad" (not in normal form)
  - we use FDs to "split" or "decompose" table and remove redundancy

- Redundancy and keys in functional dependency

  StudentClass(sid, name, addr, dept, cnum, title, unit)
  - Q: sid->(name,addr). Does it cause redundancy?

  After decomposition, Student(sid, name, addr)
  - Q: sid->(name,addr). Does it still cause redundancy?

- Q: Why does the same FD cause redundancy in one case, but not in the other?
  A: Main difference
    - sid is a key for the second example, but not for the first.
    - Because sid is a key, (sid, name, addr) is stored only once in the second. no redundancy
    - Because sid is not a key, (sid, name, addr) may be stored multiple times in the first.

- IN GENERAL, FD X->Y IS BAD IF X DOES NOT CONTAIN A KEY.

- TRIVIAL functional dependency: X -> Y where Y SUBSET X: always true
  - (diagram)

  - We focus on non-trivial functional dependency

- Boyce-Codd Normal Form (BCNF)
  - R is in BCNF with regard to F, iff for every non-trivial X -> Y, X contains a key
  - a "good" table design (no redundancy due to FD)

- Q: S(dept, cnum, title, unit). BCNF?
  dept,cnum->title,unit

- BCNF NORMALIZATION: decompose tables until all tables are in BCNF
  - For each FD that violates the condition, use it to decompose the table and remove the redundancy from it.

Fourth Normal Form (4NF)

- **Example:** Classes, students and TAs. Every TA is for every student.

  cnum: 143, TA: (tony, james), sid: (100, 101, 103)
  cnum: 248, TA: (tony), sid: (100, 102).

  (entity-relationship diagram)

(Potentially good table design)

cnum	ta
143	tony
143	james
248	tony

Cnum	sid
143	100
143	101
143	103
248	100
248	102

(Potentially bad table design)

cnum	ta	sid
248	tony	100
248	tony	102
143	tony	100
143	tony	101
143	tony	103
143	james	100
143	james	101
143	james	103

– **Q:** Does it have redundancy?

– **Q:** Does it have a FD?

– **Q:** Is it in BCNF?

– **Q:** Where does the redundancy come from?

Note:

- Two indepedent information (cnum, ta) and (cnum, sid) are put together in one table.
- No direct connection between ta and student. The connection is through class.

– **Q:** How can we detect this kind of "bad" design?

- **Q:** Assume that we have seen only the first 7 tuples in the table. Just based on these, can we "predict" that the table should also contain the 8th tuple?

  - Note:
    - In each class, every ta appears with every student (ta × student)
    - For C_1, if TA_1 appears with S_1 and TA_2 appears with S_2, then TA_1 also appears with S_2.

- Multivalued dependency X -» Y:

  for every tuple u, v ∈ R:

  If u[X] = v[X], then there exists a tuple w such that:

  1. w[X] = u[X] = v[X]
  2. w[Y] = u[Y]
  3. w[Z] = v[Z] where Z is all attributes in R except X and Y

  (Explanation using canonical database)

  MVD requires that tuples of a certain form exist: "tuple generating dependency"
  (Explation using Y circle diagram)

  - X -» Y means that if two tuples in R agree on X, we can swap Y values of the tuples and the two new tuples should still exist in R.

- Example: Class(cnum, ta, sid). cnum -» ta? cnum -» sid?

- TRIVIAL MVD
  - X -» Y is trivial MVD if

1. $Y \subset X$ or
2. $X \cup Y = R$

(Prove by canonical database)

X	Y	Z
x_1	y_1	z_1
x_1	y_2	z_2
x_1	y_1	z_2

- Fourth Normal Form (4NF)
  - R is in 4NF if for every nontrivial FD $X \rightarrow Y$ or MVD X -» Y , X contains a key
- 4NF decomposition
  - First, using all functional dependencies, normalize tables into BCNF.
  - Once the tables are in BCNF, apply the following algorithm to normalize them further into 4NF.