LECTURE 4

Unit 3

Boyce-Codd Normal form (BCNF)

Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF.

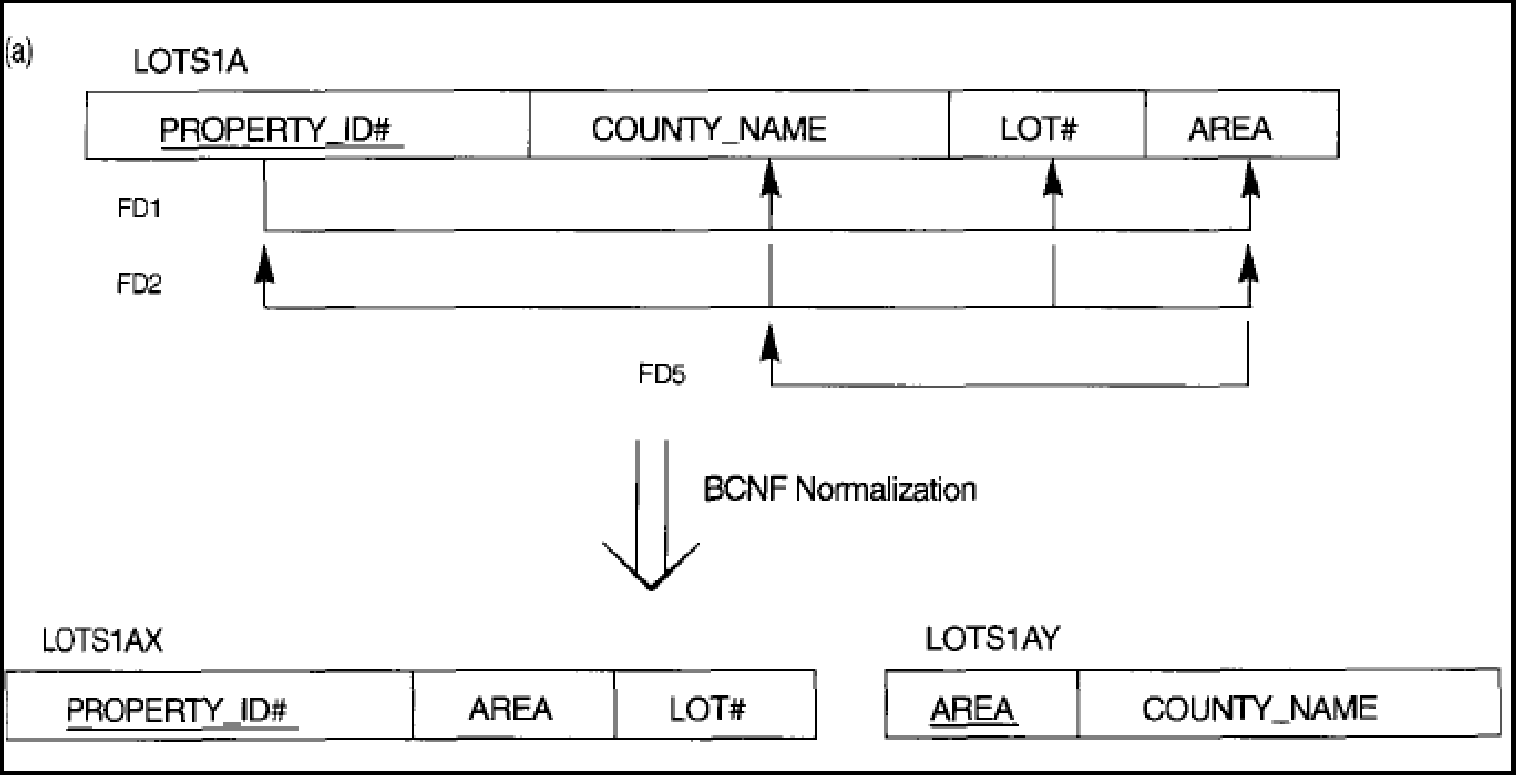
Every relation in BCNF is also in 3NF; however, a relation in 3NF is not *necessarily* in BCNF.

FD5: AREA→COUNTY_NAME

A relation schema R is in BCNF if whenever a functional dependency X→A holds in R, then X is a superkey of R.

The only difference between the definitions of BCNF and 3NF is that condition (b) of 3NF, which allows A to be prime, is absent from BCNF.

EXAMPLE – CONVERT TO BCNF



Example (1/4)

Consider the relation ADVISER (<u>SID, Major</u>, Fname)

Primary key: (SID, Major)

Candidate key: (SID, Fname)

Suppose, requirements are :

A student (SID) can have one or more majors (Major), a major can have several faculty members (Fname) as advisers, and faculty member advises in only one major area.

Also, assume no Fname have the same name

Example 2 (2/4)

Since, students can have several majors, SID does not determine Major

Since students can have several advisers, SID does not determine Fname

Therefore, SID by itself is not a key

Hence, FDs are

(SID, Major) → Fname

(SID, Fname) → Major

These either could be key

Example 2 (3/4)

Besides, candidate keys, another FD:

Fname→ Major (i.e. any faculty member advises in only one major)

Fname is the **determinant**

ADVISER relation is in 1NF, 2NF and 3NF

Reason:

Suppose, SID 300 drops out of school. If we delete SID 300 tuple, we lose the fact that Smith advises in Psychology – deletion anomaly

Also, cannot store John advises in economics, if no student has enrolled yet – insertion anomaly

Example 2 (4/4)

The relation is in BCNF if **every determinant is a candidate key**

Therefore, ADVISER relation is not in BCNF, since Fname is not a candidate key

Normalise the relation as follows:

STD_ADV (<u>SID, Fname</u>)

ADV_MAJ (<u>SID, Major</u>)

Now, these relations are in BCNF

Properties of relational decompositions (1/5) (please go through by Yourself)

Relation Decomposition and Insufficiency of Normal Forms

The universal relation assumption, which states that every attribute name is unique.

Using the FDs, the algorithms decompose the universal relation schema R into a set of relation schemas D={R1, R2, ..Rm} that will become the relational database schema

D is called a decomposition of R.

Properties of relational decompositions (2/5)

We must make sure that each attribute in R will appear in at least one relation schema Ri in the decomposition so that no attributes are "lost“

This is called the attribute preservation condition of a decomposition

Properties of relational decompositions (3/5)

Dependency Preservation Property of a Decomposition

If each FD X → Y specified in F either appeared directly in one of the relation schemas Ri in the decomposition D or could be inferred from the dependencies that appear in some Ri.

This is the *dependency preservation condition*.

Each dependency in F represents a constraint

Properties of relational decompositions (4/5)

Not necessary that the exact dependencies specified in F appear themselves in individual relations of the decomposition D

Sufficient that the union of the dependencies that hold on the individual relations in D be equivalent to F

Properties of relational decompositions (5/5)

Lossless (Nonadditive) Join Property of a Decomposition

Ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to the relations in the decomposition

Because this is a property of a decomposition of relation *schema,* the condition of no spurious tuples should hold on *every legal relation* state-that is, every relation state that satisfies the FD in F

The lossless join property is always defined w.r.t. a specific set F of dependencies.

Exercise

Reduce the following to BCNF, showing all the steps involved.

Supplier (sno, sname, saddress, (partno, partdesc, (custid, custname, custaddr, quantity)))

sno -> sname, saddr

sno, partno -> partdesc

sno, partno, custid -> quantity

sname -> sno

custid -> custname, custaddr

Suppliers supply many parts to many customers. Each customer deals with only one supplier. Supplier names are unique. Customer names are not unique