376a. Database Design

Dept. of Computer Science
Vassar College

http://www.cs.vassar.edu/~cs376
Class 12: Functional Dependencies

3NF and BCNF problems

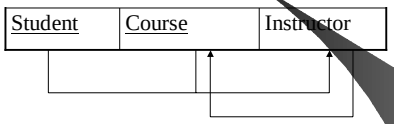
- Sometimes it's difficult to preserve dependencies

{Student, Course} → Instructor
Instructor → Course

(assuming each instructor teaches only one course)

Example

COURSE_TAKEN



Relation is in 3NF (why?)
How to convert it to BCNF?

Three possible solutions

{Student, Instructor} → {Student, Instructor}
{Student, Course} → {Student, Course}
Or
Instructor->Course
{Student, Course} →{Student, Course}
Or
Instructor->Course
{Student, Instructor} →{Student, Instructor}

All three lose {Student,Course} → Instructor FD

Non-additive join

Relations should be decomposed to meet this property (when joined, no extra tuples.)

Remember

R is universal relation schema.

Attribute preservation property

$$\bigcup_{i=1}^{m} R_i = R$$

All new relations should have the same attributes as the starting relation.

Three additional conditions

NF are not sufficient to guarantee good design.

Will use the following conditions to help:

- Dependency preservation condition
- Loss-less join

Dependency preservation

Either the dependency exists in a relation or can be inferred from a set of relations in the decomposition.

Use the relational synthesis algorithm to accomplish this.

Relational synthesis algorithm

Find minimal cover G for F

For each X in X →Y create the relational schema {X ∪ Y} for all dependencies in G with X as left hand side (X is key of relation).

Place remaining attributes (unplaced) in a single relational schema to ensure attribute preservation property.

Lossless (nonadditive) join

No spurious tuples is a condition of the extension of a relational schema (must hold for every instance of the database)

Lossless refers to loss of information not to loss of tuples (tuples may be very different).

Lossless (nonadditive) join test

Create matrix S. Set all elements to $b_{i,j}$. Rows are relations R_i and columns are attributes A_j.

For each relation (R_i), change $b_{i,j}$ to A_j if A_j is in R_i.

For each FD X → Y, for all Ri that have the same X symbols, make all the Ys the same. (Use A's first, otherwise use b's)

If one row has all A's, D has lossless join prop.

Do an example

R(A,B,C,D,E,F)

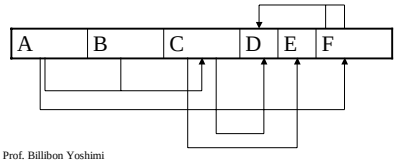
FDs

- {A, B} →C
- A →F
- C →D
- C →E
- {F, E} →D

Convert to BCNF.

Find the key for the relation

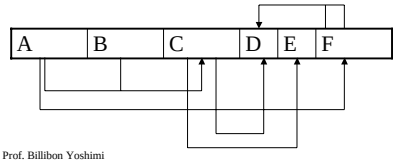
Start with K = set of all attributes
For each attribute Ai
If $\{K - A\}^+$ is equivalent to $\{K\}^+$
Set K = $\{K-A\}^+$



Key

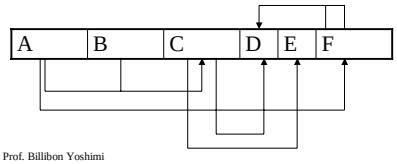
$\{A, B\}^+ \rightarrow \{A, B, C, D, E, F\}$

Find a FD $X \rightarrow Y$ in R that violates BCNF
Remove Y from R and create new relation $X \cup Y$

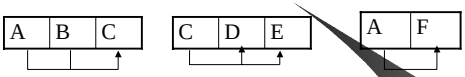


Remember BCNF is 1NF, 2NF and

If $X \rightarrow Y$, then X must be a super key.



Lossless join preserved?



Create matrix S. Set all elements to $b_{i,j}$. Rows are relations R_i and columns are attributes A_i.
For each relation (R_i), change $b_{i,j}$ to A_j if A_j is in R_i.
For each FD $X \rightarrow Y$, for all Ri that have the same X symbols, make all the Ys the same. (Use A's first, otherwise use b's)
If one row has all A's, D has lossless join prop.

Example

	A	B	C	D	E	F
R(ABC)	A1	A2	A3	b14	b15	b16
R(CDE)	b21	b22	A3	A4	A5	b26
R(AF)	A1	b32	B33	b34	b35	A6

$\{A, B\} \rightarrow C$, $A \rightarrow F$, $C \rightarrow D$, $C \rightarrow E$, $\{F, E\} \rightarrow D$

Some other problems

Lossless joins decompositions assume no null values for joint attributes!
General problems with nulls for foreign keys (must do outer join (left or right) to maintain information)
Also problem with nulls for grouping values (average, sum, etc.)
Dangling tuples - when decomposed too far (natural join loses tuples.)

Multi-valued dependencies

X⇒Y (⇒ is used instead of ->> shown in the book.)

There exists three tuples t, u, and v s.t.

t[X] = u[X] = v[X]

t[Y] = u[Y]

u[Z] = v[Z]

Where Z = R-(X ∪ Y)

Means

For every pair of tuples, t and u, we can find a v s.t.

- v has the same X as t and u
- v has the same Y as t
- v has the same Z as u

From this relation we can infer information that does not exist!

Example

Movie star relation

R[Name, Street, City, Title, Year, Type]

Star can appear in multiple movies.

Star can have multiple addresses

Star may make more than one movie/yr.

Star can be in original and remake.

Go through Dependency Synth process.

Dependencies:

- {Name} -> {Street, City}
- {Name}-> {Title, Year, Type}

Keys:

- Name

Is this BCNF?

Add a star with two addresses and two movies

How many tuples?

Name	Street	City	Title	Year
Fisher	123	Holly	StarWars	1983
Fisher	1	NJ	StarWars	1983
Fisher	123	Holly	Empire	1987
Fisher	1	NJ	Empire	1987

Problem

- Redundant information
- But in BCNF!

Fourth Normal Form (4NF)

Removes multi-valued dependencies.
(removes insertion, deletion anomalies)
For all non-trivial dependency, X ⇒ Y,in F^+, X is a super key of R.

A trivial MVD is
1. Y is subset of X or
2. X ∪ Y is R

Remember our example

R[Name, Street, City, Title, Year, Type]
Dependencies:
{Name} -> {Street, City}
{Name}-> {Title, Year, Type}

Name is not super key!

Relational Decomposition to 4NF

Find a MVD X ⇒ Y in R that violates 4NF.
Replace R with R-Y and create new relation (X u Y)
Repeat until all MVD are trivial.

Note: any decomposition created with this algorithm exhibits the lossless join property

Lossless Join Property

R1 and R2 form a lossless join decomposition of R iff (R1 ∩ R2) ⇒ (R1 - R2) or (R1 ∩ R2) ⇒ (R2-R1)

Join dependencies

Join dependency (JD) state that for
*(R1, R2, … Rn) = R
The natural join over all relations.

MVD is just n = 2 or JD(R1, R2) implies (R1 ∩ R2) ⇒ (R1 - R2)
Trivial JD if one of Rs is R (the entire relation).
But what happens when n>2

Fifth normal form (5NF)

Project-join normal form (PJNF).
For each JD(R1, R2..Rn) each Ri is a superkey of R.
Natural join on ALL N preserves relation.
Show how this works on

Sname	Part	Proj
Smith	Bolt	X
Smith	Nut	Y
Adam	Bolt	Y
Walt	Nut	Z
Adam	Nail	Y

5NF

Dependency is difficult to detect in practice, normally not done in practice.

Other dependencies

Inclusion dependency
Template dependency

Domain-key normal form - the ultimate in normal forms (very difficult to implement in practice).