DM505 Database Design and Programming
DM576 Database Systems

Panagiotis Tampakis

ptampakis@imada.sdu.dk

Relational Schema Design

- Goal of relational schema design is to avoid anomalies and redundancy
  - *Update anomaly:* one occurrence of a fact is changed, but not all occurrences
  - *Deletion anomaly:* valid fact is lost when a tuple is deleted

Example of Bad Design

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Peter	Campusvej	Odense Cl.	Alb.	Erdinger W.
Peter	???	Erdinger W.	Erd.	???
Lars	NULL	Odense Cl.	???	Odense Cl.

Data is redundant, because each of the ???'s can be figured out by using the FDs name → addr favBeer and beersLiked → manf

This Bad Design Also Exhibits Anomalies

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

name	addr	beersLiked	manf	favBeer
Peter	Campusvej	Odense Cl.	Alb.	Erdinger W.
Peter	Campusvej	Erdinger W.	Erd.	Erdinger W.
Lars	NULL	Odense Cl.	Alb.	Odense Cl.

- Update anomaly: if Peter moves to Niels Bohrs Alle, will we remember to change each of his tuples?
- Deletion anomaly: If nobody likes Odense Classic, we lose track of the fact that Albani manufactures Odense Classic

Boyce-Codd Normal Form

- We say a relation *R* is in *BCNF* if whenever $X \rightarrow Y$ is a nontrivial FD that holds in *R*, *X* is a superkey
  - Remember: *nontrivial* means *Y* is not contained in *X*
  - Remember, a *superkey* is any superset of a key

Example

Drinkers(name, addr, beersLiked, manf, favBeer)

FDs: name → addr favBeer,   beersLiked -> manf

- Only key is {name, beersLiked}
- In each FD, the left side is *not* a superkey
- Any one of these FDs shows *Drinkers* is not in BCNF

Another Example

Beers(<u>name</u>, manf, manfAddr)

FDs: name → manf,   manf → manfAddr

- Only key is {name}
- Name → manf does not violate BCNF, but manf → manfAddr does

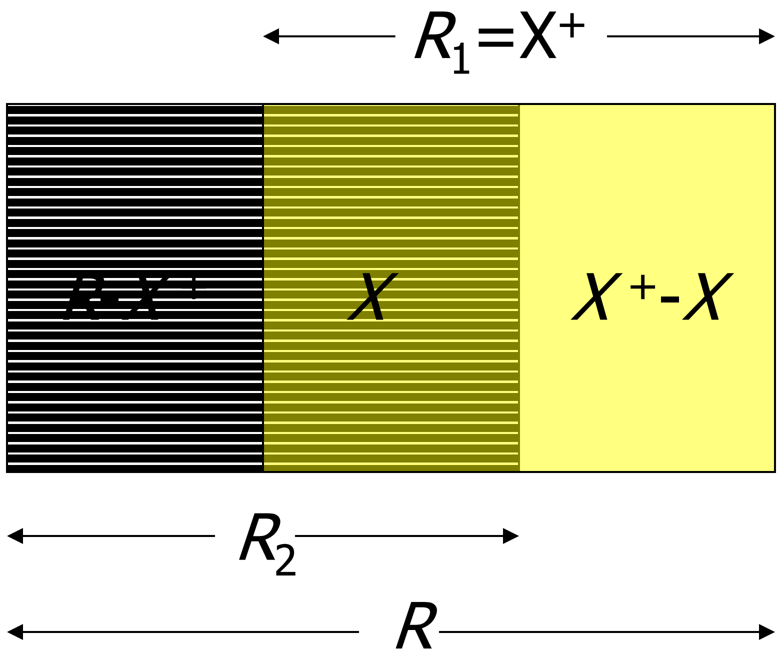
Decomposition into BCNF

- Given: relation R with FDs F
- Look among the given FDs for a BCNF violation $X \rightarrow Y$
  - If any FD following from F violates BCNF, then there will surely be a FD in F itself that violates BCNF

Decompose R Using $X \rightarrow Y$

- Compute X^+
- Replace R by relations with schemas:
  1. $R_1 = X^+$
  2. $R_2 = R - (X^+ - X)$
- *Project* given FDs F onto the two new relations

Decomposition Picture



Example: BCNF Decomposition

Drinkers(<u>name</u>, addr, <u>beersLiked</u>, manf, favBeer)

F = name → addr, name → favBeers
beersLiked → manf

- Pick BCNF violation name → addr
- Close the left side:
  $\{name\}^+ = \{name, addr, favBeer\}$
- Decomposed relations:
  1. Drinkers1(<u>name</u>, addr, favBeer)
  2. Drinkers2(<u>name</u>, <u>beersLiked</u>, manf)

Example: BCNF Decomposition

- We are not done; we need to check Drinkers1 and Drinkers2 for BCNF
- Projecting FDs is easy here
- For Drinkers1(name, addr, favBeer), relevant FDs are name -> addr and name -> favBeer
  - Thus, {name} is the only key and Drinkers1 is in BCNF

Example: BCNF Decomposition

- For Drinkers2(name, beersLiked, manf), the only FD is beersLiked -> manf, and the only key is {name, beersLiked}
  - Violation of BCNF
- beersLiked^+ = {beersLiked, manf}, so we decompose *Drinkers2* into:

1. Drinkers3(beersLiked, manf)
2. Drinkers4(name, beersLiked)

Example: BCNF Decomposition

- The resulting decomposition of *Drinkers:*

1. Drinkers1(name, addr, favBeer)
2. Drinkers3(beersLiked, manf)
3. Drinkers4(name, beersLiked)

- Notice: *Drinkers1* tells us about drinkers, *Drinkers3* tells us about beers, and *Drinkers4* tells us beers the drinkers like
- Compare with our running example:

1. Drinkers(name, addr, phone)
2. Beers(name, manf)
3. Likes(drinker,beer)

Exercise

Let us consider the following relation R(A,B,C,D) with FD's:

- $A \rightarrow B$
- $B \rightarrow C$
- $C \rightarrow D$

Examine if this relation is in BCNF and if it is not decompose it into BCNF

3rd Normal Form

Third Normal Form – Motivation

- Properties of Decomposition
  1. Elimination of Anomalies
     - Redundancy
     - Update Anomalies
     - Deletion Anomalies
  2. Recoverability of Information
  3. Preservation of Dependencies
- BCNF provides (1) and (2) but not necessarily (3).

Testing for a Lossless Join

- If we project R onto R_1, R_2,…, R_k, can we recover R by rejoining?
- Any tuple in R can be recovered from its projected fragments
- So, the only question is: When we rejoin, would we retrieve entries we didn't see before (i.e. create phantom/ghost entries)?

Example: Phantom entries

- Let $R = ABC$, and the decomposition be AB and BC
- Let the given FD be $A \rightarrow B$

A	***B***	***C***
1	2	3
4	5	6
7	2	8

Our table R.

A	***B***
1	2
4	5
7	2

B	***C***
2	3
5	6
2	8

Decompositions R1 and R2.

A	***B***	***C***
1	2	3
4	5	6
7	2	8
1	**2**	**8**
7	**2**	**3**

R1 join R2.

Example: No phantom entries

- Let *R* = *ABC*, and the decomposition be *AB* and *BC*
- Let the given FD be *A* -> *B* and *B* -> *C*

A	***B***	***C***
1	2	3
4	5	6
7	2	3

Our table R.

A	***B***
1	2
4	5
7	2

B	***C***
2	3
5	6

Decompositions R1 and R2.

A	***B***	***C***
1	2	3
4	5	6
7	2	3

R1 join R2.

Chase Test for Lossless Join

- Let R(A,B,C,D)
  - with FD's A→B, B→C and CD→A
- R_1(A,D), R_2(A,C), R_3(B,C,D)
- Tableau

A	B	C	D
a	b_1	c_1	d
a	b_2	c	d_2
a_3	b	c	d

Chase Test for Lossless Join

- Our goal is to use the given set of FD's F to prove that t is really in R.
  - to do so, we apply the FD's in F to equate symbols in the tableau whenever we can.
  - If we discover that one of the rows is actually the same as t, then we have proved that any tuple t in the join of the projections was actually a tuple of R.

Chase Test for Lossless Join

- To avoid confusion, when equating two symbols,
  - if one of them is unsubscripted, make the other be the same.
  - if we equate two symbols, both with their own subscript, then you can change either to be the other.
- You must change all occurrences of one to be the other, not just some of the occurences.

Chase Test for Lossless Join

- Replace b_2 with b_1

A	B	C	D
a	$\boxed{b_1}$	c_1	d
a	$\boxed{b_1}$	c	d_2
a_3	b	c	d

Chase Test for Lossless Join

- Replace c_1 with c

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a_3	b	c	d

Chase Test for Lossless Join

- Replace a_3 with a

A	B	C	D
$\boxed{a}$	b_1	c	d
a	b_1	c	d_2
$\boxed{a}$	b	c	d

Chase Test for Lossless Join

- Replace a_3 with a

A	B	C	D
a	b_1	c	d
a	b_1	c	d_2
a	b	c	d

Third Normal Form – Motivation

- There is one structure of FDs that causes trouble when we decompose ABC: *AB*-> *C* and *C*-> *B*
  - Example:
    A = street, *B* = city, *C* = post code
- There are two keys, {*A*,*B* } and {*A*,*C* }
- *C*-> *B* is a BCNF violation, so we must decompose ABC into *AC*, *BC*

We Cannot Enforce FDs

- The problem is that if we use *AC* and *BC* as our database schema, we cannot enforce the FD $AB \rightarrow C$ by checking FDs in these decomposed relations
- Example with *A* = street, *B* = city, and *C* = post code on the next slide

An Unenforceable FD

street	post
Campusvej	5230
Vestergade	5000

city	post
Odense	5230
Odense	5000

Join tuples with equal post codes

street	city	post
Campusvej	Odense	5230
Vestergade	Odense	5000

No FDs were violated in the decomposed relations and FD street city -> post holds for the database as a whole

An Unenforceable FD

street	post
Hjallesevej	5230
Hjallesevej	5000

city	post
Odense	5230
Odense	5000

Join tuples with equal post codes

street	city	post
Hjallesevej	Odense	5230
Hjallesevej	Odense	5000

Although no FDs were violated in the decomposed relations, FD street city -> post is violated by the database as a whole

Another Unenforcable FD

- Departures(time, track, train)
- time track -> train and train -> track
- Two keys, {time,track} and {time,train}
- train -> track is a BCNF violation, so we must decompose into
  Departures1(time, train)
  Departures2(track,train)

Another Unenforceable FD

time	train
19:08	ICL54
19:16	IC852

track	train
4	ICL54
3	IC852

Join tuples with equal train code

time	track	train
19:08	4	ICL54
19:16	3	IC852

No FDs were violated in the decomposed relations,
FD time track -> train holds for the database as a whole

Another Unenforceable FD

time	train
19:08	ICL54
19:08	IC 42

track	train
4	ICL54
4	IC 42

Join tuples with equal train code

time	track	train
19:08	4	ICL54
19:08	4	IC 42

Although no FDs were violated in the decomposed relations, FD time track -> train is violated by the database as a whole

3NF avoids this problem

- 3rd Normal Form (3NF) modifies the BCNF condition so we do not have to decompose in this problem situation
- An attribute is *prime* if it is a member of any key
- $X \rightarrow A$ violates 3NF if and only if X is not a superkey, and also A is not prime
- Or: $X \rightarrow A$ holds 3NF iff X is a superkey or A is prime (i.e. member of any key)

Example: 3NF

- In our problem situation with FDs $AB \rightarrow C$ and $C \rightarrow B$, we have keys AB and AC
- Thus A, B, and C are each prime
- Although $C \rightarrow B$ violates BCNF, it does not violate 3NF

What 3NF and BCNF Give You

- There are two important properties of a decomposition:
  1. *Lossless Join:* it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original
  2. *Dependency Preservation:* it should be possible to check in the projected relations whether all the given FDs are satisfied

3NF and BCNF – Continued

- We can get (1) with a BCNF decomposition
- We can get both (1) and (2) with a 3NF decomposition
- But we can't always get (1) and (2) with a BCNF decomposition
  - street-city-post is an example
  - time-track-train is another example

3NF Synthesis Algorithm

- We can always construct a decomposition into 3NF relations with a lossless join and dependency preservation
- We need a *minimal basis* for the FDs:
  1. Right sides are single attributes
  2. No FD can be removed
  3. No attribute can be removed from a left side

Constructing a Minimal Basis

1. Split right sides
2. Repeatedly try to remove a FD and see if the remaining FDs are equivalent to the original
3. Repeatedly try to remove an attribute from a left side and see if the resulting FDs are equivalent to the original

3NF Synthesis

- One relation for each FD in the minimal basis
  - Schema is the union of the left and right sides
- If no key is part of a FD: add one relation whose schema is some key

Example: 3NF Synthesis

- Relation R = ABCD
- FDs *A -> B* and *A -> C*
- Decomposition: AB and AC from the FDs, plus AD for a key

Summary 5

More things you should know:

- Update Anomaly, Deletion Anomaly
- BCNF, Closure, Decomposition
- 3rd Normal Form