Chapter 3 Design Theory for Relational Databases

Contents

- Functional Dependencies
- Decompositions
- Normal Forms (BCNF, 3NF)
- Multivalued Dependencies (and 4NF)
- Reasoning About FD′s + MVD′s

A New Form of Redundancy

- A relation is trying to represent more than one many-many relationship.

name	addr	phones	beersLiked
sue	a	p1	b1
sue	a	p2	b2
sue	a	p2	b1
sue	a	p1	b2

Then these tuples must also be in the relation.

- A drinker's phones are independent of the beers they like.
- Thus, each of a drinker's phones appears with each of the beers they like in all combinations.

Definition of MVD

- A *multivalued dependency* (MVD) on *R*, $X \text{-}{>}{>} Y$, says that if two tuples of *R* agree on all the attributes of *X*, then their components in *Y* may be swapped, and the result will be two tuples that are also in the relation.
- i.e., for each value of *X*, the values of *Y* are independent of the values of *R-X-Y*.

Example: MVD

Drinkers(name, addr, phones, beersLiked)

- A drinker's phones are independent of the beers they like.
  - name->->phones and name ->->beersLiked.
- Thus, each of a drinker's phones appears with each of the beers they like in all combinations.
- This repetition is unlike FD redundancy.
  - name->addr is the only FD.

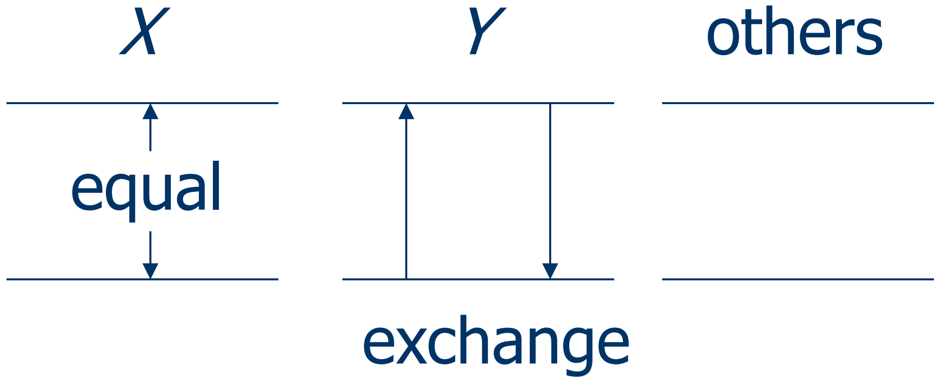
Tuples Implied by name->->phones

If we have tuples:

name	addr	phones	beersLiked
sue	a	p1	b1
sue	a	p2	b2
sue	a	p2	b1
sue	a	p1	b2

Then these tuples must also be in the relation.

Picture of MVD *X* ->->*Y*



MVD Rules

- Every FD is an MVD (*promotion*).
  - If *X* ->*Y*, then swapping *Y* ’s between two tuples that agree on *X* doesn’t change the tuples.
  - Therefore, the “new” tuples are surely in the relation, and we know *X* ->->*Y*.
- *Complementation* : If *X* ->->*Y*, and *Z* is all the other attributes, then *X* ->->*Z*.

Splitting Doesn’t Hold

- Like FD’s, we cannot generally split the left side of an MVD.
- But unlike FD’s, we cannot split the right side either --- sometimes you have to leave several attributes on the right side.

Example: Multiattribute Right Sides

Drinkers(name, areaCode, phone, beersLiked, manf)

- A drinker can have several phones, with the number divided between areaCode and phone (last 7 digits).
- A drinker can like several beers, each with its own manufacturer.

Example Continued

- Since the areaCode-phone combinations for a drinker are independent of the beersLiked-manf combinations, we expect that the following MVD's hold:

    name ->-> areaCode phone

    name ->-> beersLiked manf

Example Data

Here is possible data satisfying these MVD's:

name	areaCode	phone	beersLiked	manf
Sue	650	555-1111	Bud	A.B.
Sue	650	555-1111	WickedAle	Pete's
Sue	415	555-9999	Bud	A.B.
Sue	415	555-9999	WickedAle	Pete's

But we cannot swap area codes or phones by themselves. That is, neither name->->areaCode nor name->->phone holds for this relation.

Fourth Normal Form

- The redundancy that comes from MVD's is not removable by putting the database schema in BCNF.
- There is a stronger normal form, called 4NF, that (intuitively) treats MVD's as FD's when it comes to decomposition, but not when determining keys of the relation.

4NF Definition

- A relation *R* is in *4NF* if: whenever *X* ->->*Y* is a nontrivial MVD, then *X* is a superkey.
  - *Nontrivial MVD* means that:
    1. *Y* is not a subset of *X*, and
    2. *X* and *Y* are not, together, all the attributes.
  - Note that the definition of "superkey" still depends on FD's only.

BCNF Versus 4NF

- Remember that every FD *X ->Y* is also an MVD, *X ->->Y*.
- Thus, if *R* is in 4NF, it is certainly in BCNF.
  - Because any BCNF violation is a 4NF violation (after conversion to an MVD).
- But *R* could be in BCNF and not 4NF, because MVD's are "invisible" to BCNF.

Decomposition and 4NF

Input: relation R + FDs for R + MVDs for R

Output: decomposition of R into 4NF relations

with "lossless join"

1. Compute keys for R
2. Repeat until all relations are in 4NF:

Pick any R' with nontrivial A→→ B that violates 4NF

Decompose R' into R1(A, B) and R2(A, rest)

Compute FDs and MVDs for R1 and R2

Computer keys for R1 and R2

Example: 4NF Decomposition

Drinkers(<u>name</u>, addr, <u>phones</u>, <u>beersLiked</u>)

FD: name -> addr

MVD's: name ->-> phones

name ->-> beersLiked

- Key is {name, phones, beersLiked}.
- All dependencies violate 4NF.

Example Continued

- Decompose using name -> addr:

1. Drinkers1(<u>name</u>, addr)
   - In 4NF; only dependency is name -> addr.
2. Drinkers2(<u>name</u>, <u>phones</u>, <u>beersLiked</u>)
   - Not in 4NF. MVD's name ->-> phones and name ->-> beersLiked apply. No FD's, so all three attributes form the key.

Example: Decompose Drinkers2

- Either MVD name ->-> phones or name ->-> beersLiked tells us to decompose to:
  - Drinkers3(<u>name</u>, <u>phones</u>)
  - Drinkers4(<u>name</u>, <u>beersLiked</u>)

Reasoning About MVD's + FD's

- Problem: given a set of MVD's and/or FD's that hold for a relation *R*, does a certain FD or MVD also hold in *R* ?
- Solution: Use a tableau to explore all inferences from the given set, to see if you can prove the target dependency.

Why Do We Care?

1. 4NF technically requires an MVD violation.
   - Need to infer MVD’s from given FD’s and MVD’s that may not be violations themselves.
2. When we decompose, we need to project FD’s + MVD’s.

Example: Chasing a Tableau With MVD's and FD's

- To apply a FD, equate symbols, as before.
- To apply an MVD, generate one or both of the tuples we know must also be in the relation represented by the tableau.
- We'll prove: if A->->BC and D->C, then A->C.

The Tableau for A->C

Goal: prove that $c_1 = c_2$.

A	B	C	D
a	b_1	$\cancel{c_1}$ c_2	d_1
a	b_2	c_2	d_2
a	b_2	c_2	d_1

Use A->->BC (first row's *D* with second row's *BC*).

Use D->C (first and third row agree on *D*, therefore agree on *C*).

Example: Transitive Law for MVD's

- If A->->B and B->->C, then A->->C.
  - Obvious from the complementation rule if the Schema is *ABC*.
  - But it holds no matter what the schema; we'll assume *ABCD*.

The Tableau for A->->C

Goal: derive tuple (a,b_1,c_2,d_1).

A	B	C	D
a	b_1	c_1	d_1
a	b_2	c_2	d_2
a	b_1	c_2	d_2
a	b_1	c_2	

d_1

Use A->->B to swap B from the first row into the second.

Use B->->C to swap C from the third row into the first.

Rules for Inferring MVD's + FD's

- Start with a tableau of two rows.
  - These rows agree on the attributes of the left side of the dependency to be inferred.
  - And they disagree on all other attributes.
  - Use unsubscripted variables where they agree, subscripts where they disagree.

Inference: Applying a FD

- Apply a FD X->Y by finding rows that agree on all attributes of *X*. Force the rows to agree on all attributes of *Y*.
  - Replace one variable by the other.
  - If the replaced variable is part of the goal tuple, replace it there too.

Inference: Applying a MVD

- Apply a MVD X->->Y by finding two rows that agree in *X*.
  - Add to the tableau one or both rows that are formed by swapping the *Y*-components of these two rows.

Inference: Goals

- To test whether U->V holds, we succeed by inferring that the two variables in each column of V are actually the same.
- If we are testing U->->V, we succeed if we infer in the tableau a row that is the original two rows with the components of V swapped.

Inference: Endgame

- Apply all the given FD′s and MVD′s until we cannot change the tableau.
- If we meet the goal, then the dependency is inferred.
- If not, then the final tableau is a counterexample relation.
  - Satisfies all given dependencies.
  - Original two rows violate target dependency.

Relationships Among Normal Forms

Property	3NF	BCNF	4NF
Eliminates Redundancy due to FDs	Most	Yes	Yes
Eliminates redundancy Due to MVDs	No	No	Yes
Preserves FDs	Yes	maybe	maybe
Preserves MVD	maybe	maybe	maybe
Equal to original relation	Yes	Yes	Yes

Relational Database Design

- Normal forms – “good” relation
- Design by decomposition, usually intuitive and works well
- Some Shortcomings:

a) Over-decomposition

b) Query workload

c) Dependency enforcement

Summary of the chapter

- Functional dependencies
- Keys of a relation
- Minimal basis for a set of FD’s
- BCNF and 3NF
- BCNF decomposition and 3NF synthesis (with lossless-join and dependency-preserving)
- Multivalued dependencies
- 4NF
- Reasoning about MVD and FD

Classroom exercise 3.7.1

Use the chase test to tell whether each of the following dependencies hold in a relation R(A,B,C,D,E) with the dependencies A→→BC, B→D, and C→→E

a) A→D

d) A→→E

Classroom Exercise 3.3.1 a) c) and 3.5.1

R(A,B,C,D) with

a) FD's AB→C, C→D, D→A

c) AB→C, BC→D, CD→A, AD→B

1. Indicate all the BCNF violations.
2. Decompose the relations, as necessary into collections of relations that are in BCNF
3. Indicate all the 3NF violations
4. Decompose the relations into 3NF.

Solution (a)

1) C->A, C->D, D->A, AC->D, CD->A

2) Key are AB, BC, and BD

BCNF: R1(AC), R2(BC), R3(CD)

Or R1(CD),R2(BC),R3(AD) or…

3) No 3NF violations and why?

4) R(A,B,C,D) is already in 3NF

Solution c

i) indicate all the BCNF violations

- Consider the closures of all 15 nonempty subsets of {A,B,C,D}.
- A+=A, B+=B, C+=C, and D+=D. Thus we get no new FD's.
- AB+=BC+=CD+=AD+=ABCD, AC+=AC, and BD+=BD. Thus we get new nontrivial FD's: $AB \rightarrow D$, $BC \rightarrow A$, $CD \rightarrow B$, $AD \rightarrow C$.
- ABC+=ABD+=ACD+=BCD+=ABCD. Thus we get new nontrivial FD's: $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCE \rightarrow A$.
- ABCD+=ABCD, so we get no new FD's.
- To sum up, we can deduce 8 new nontrivial FD's from the given 4 FD's.
- From above, we find that AB, BC, CD and AD are keys, and that all the nontrivial FD's for R contain a key on the left side. Thus there're no BCNF violations.

ii) R(A,B,C,D) is already in BCNF.

iii) No 3NF violations.

iv) R(A,B,C,D) is already in 3NF.

Homework

- Exercise 3.2.1
- Exercise 3.5.2
- Exercise 3.6.3 a), c)