83

Part III: Normal Forms

83

84

Database Design Theory

- General idea:
  - Express constraints on the data
  - Use these to decompose the relations
- Ultimately, get a schema that is in a "normal form" that guarantees good properties, such as no anomalies.
- "Normal" in the sense of conforming to a standard.
- The process of converting a schema to a normal form is called normalization.

Acknowledgements: M. Papagelis, R. Johnson

84

85

Motivation for normal forms

- Identify a "good" schema
  - For some definition of "good"
  - Avoid anomalies, redundancy, etc.
- Many normal forms
  - 1st
  - 2nd
  - 3rd
  - Boyce-Codd
  - ... and several more we won't discuss...

BCNF ⊆ 3NF ⊆ 2NF ⊆ 1NF

85

86

1st Normal Form (1NF)

- No multi-valued attributes allowed
  - Imagine storing a list of values in an attribute
- Counter example
  - Course(name, instructor, [student,email]*)

Name	Instructor	Student Name	Student Email
CS 3DB3	Chiang	Alice	alice@gmail
		Mary	mary@mac
		Mary	mary@mac
SE 3SH3	Miller	Nilesh	nilesh@gmail

86

2nd normal form (2NF)

87

- Non-key attributes depend on candidate keys
  - Consider non-key attribute A
  - Then there exists an FD X s.t. X -> A, and X is a candidate key
- Counter-example
  - Movies(title, year, star, studio, studioAddress, salary)
  - FD: title, year -> studio; studio -> studioAddress; star->salary

Title	Year	Star	Studio	StudioAddr	Salary
Star Wars	1977	Hamill	Lucasfilm	1 Lucas Way	$100,000
Star Wars	1977	Ford	Lucasfilm	1 Lucas Way	$100,000
Star Wars	1977	Fisher	Lucasfilm	1 Lucas Way	$100,000
Patriot Games	1992	Ford	Paramount	Cloud 9	$2,000,000
Last Crusade	1989	Ford	Lucasfilm	1 Lucas Way	$1,000,000

87

3rd normal form (3NF)

88

- Non-prime attr. depend *only* on candidate keys
  - Consider FD X -> A
  - Either X is a superkey OR A is *prime* (part of a key)
- Counter-example:
  - studio -> studioAddr

    (*studioAddr* depends on *studio* which is not a candidate key)

Title	Year	Studio	StudioAddr
Star Wars	1977	Lucasfilm	1 Lucas Way
Patriot Games	1992	Paramount	Cloud 9
Last Crusade	1989	Lucasfilm	1 Lucas Way

88

3NF, dependencies, and join loss

89

- Theorem: always possible to convert a schema to lossless join, dependency-preserving 3NF
- Caveat: always *possible* to create schemas in 3NF for which these properties do not hold
- FD loss example 1:
  - MovieInfo(title, year, studioName)
  - StudioAddress(title, year, studioAddress)

  => Cannot enforce studioName -> studioAddress
- Join loss example 2:
  - Movies(title, year, star)
  - StarSalary(star, salary)

  => Movies ⋈ StarSalary yields additional tuples

89

Boyce-Codd normal form (BCNF)

90

- One additional restriction over 3NF
  - All non-trivial FDs have superkey LHS
- Counterexample
  - CanadianAddress(street, city, province, postalCode)
  - Candidate keys: {street, postalCode}, {street, city, province}
  - FD: postalCode -> city, province
  - Satisfies 3NF: city, province both prime
  - Violates BCNF: postalCode is not a superkey

  => Possible anomalies involving postalCode

90

Boyce-Codd Normal Form

91

- We say a relation *R* is in *BCNF* if whenever *X* → *A* is a nontrivial FD that holds in *R*, *X* is a superkey.
  - Remember: *non-trivial* means A is not contained in *X*.

91

Example: a relation not in BCNF

92

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

FD's: 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

92

Another Example

93

Beers(name, manf, manfAddr)

FD's: name->manf, manf->manfAddr

- Beers w.r.t. name->manf does not violate BCNF, but manf->manfAddr does.

In other words, BCNF requires that:
the only FDs that hold are the result of key(s).
Why does that help?

93

Decomposition into BCNF

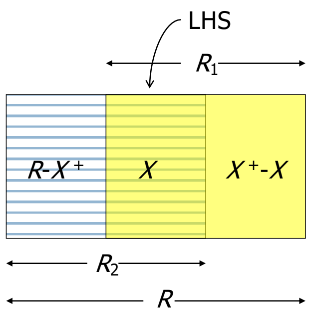
94

- Given: relation *R* with FDs *F*
- Look among the given FDs for a BCNF violation $X \rightarrow Y$ (i.e., X is not a superkey)
- Compute X^+.
  - Find $X^+ \neq X \neq$ all attributes, (o.w. X is a superkey)
- Replace *R* by relations with:
  - $R_1 = X^+$.
  - $R_2 = R - (X^+ - X) = R - X^+ \cup X$
- Continue to recursively decompose the two new relations
- *Project* given FDs *F* onto the two new relations.

94

Decomposition on X → Y

95



95

Example: BCNF Decomposition

96

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

F = {name→addr, name → favBeer, beersLiked→manf}

Key = name, beersLiked

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

96

Example -- Continued

97

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

97

Example -- Continued

98

- For Drinkers2(<u>name</u>, <u>beersLiked</u>, 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:
  - Drinkers3(<u>beersLiked</u>, manf)
  - Drinkers4(<u>name</u>, <u>beersLiked</u>)

98

Example -- Concluded

99

- The resulting decomposition of *Drinkers* :
  - Drinkers1(name, addr, favBeer)
  - Drinkers3(beersLiked, manf)
  - Drinkers4(name, beersLiked)
- Notice: *Drinkers1* tells us about drinkers, *Drinkers3* tells us about beers, and *Drinkers4* tells us the relationship between drinkers and the beers they like.

99

What we want from a decomposition

100

- *Lossless Join* : it should be possible to project the original relations onto the decomposed schema, and then reconstruct the original, i.e., get back exactly the original tuples.
- *No anomalies*
- *Dependency Preservation* : All the original FDs should be satisfied.

100

What we get from a BCNF decomposition

101

- *Lossless Join* : ✓
- *No anomalies* : ✓
- *Dependency Preservation* : ✗

101

Example: Failure to preserve dependencies

102

- Suppose we start with R(*A,B,C*) and FDs
  - *AB* → C and C → *B*.
- There are two keys, {*A,B* } and {*A,C* }.
- C → *B* is a BCNF violation, so we must decompose into *AC*, *BC*.

The problem is that if we use *AC* and *BC* as our database schema, we cannot enforce the FD *AB* → C in these decomposed relations.

102

3NF: Helps Us Avoid this Problem

103

- *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.
- $X \to A$ violates 3NF if and only if X is not a superkey, and also A is not prime.
- i.e., it's ok if X is not a superkey, as long as A is prime.

103

Example: 3NF

104

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

104

What we get from a 3NF decomposition

105

- *Lossless Join* : ✓
- *No anomalies* : ✗
- *Dependency Preservation* : ✓

Why?

Unfortunately, neither BCNF nor 3NF can guarantee all three properties we want.

105

3NF Synthesis Algorithm

106

- We can always construct a decomposition into 3NF relations with a lossless join and dependency preservation.
- Need *minimal basis* for the FDs (same as used in projection)
  - Right sides are single attributes.
  - No FD can be removed.
  - No attribute can be removed from a left side.

106

3NF Synthesis – (2)

107

- One relation for each FD in the minimal basis.
  - Schema is the union of the left and right sides.
- If no key is contained in an FD, then add one relation whose schema is some key.

107

Example: 3NF Synthesis

108

- Relation R = ABCD.
- FDs $A \rightarrow B$ and $A \rightarrow C$.
- Decomposition: AB and AC from the FDs, plus AD for a key.

108

Limits of decomposition

109

- Pick two…
  - Lossless join
  - Dependency preservation
  - Anomaly-free
- 3NF
  - Provides lossless join and dependency preserving
  - May allow some anomalies
- BCNF
  - Anomaly-free, lossless join
  - Sacrifice dependency preservation

Use domain knowledge to choose 3NF vs. BCNF

109