Normalization Algorithms

CS157A
Chris Pollett
Nov. 21, 2005.

Outline

- General Definitions of Normal Forms
- Boyce-Codd Normal Form
- Properties of Relational Decomposition

General Definitions of Normal Forms

- The original definitions of normal forms we gave last day were defined in terms of the primary key.
- Today, we'll briefly consider definitions which are with respect to any key.

General Definition 2NF

- A relation schema R is in 2NF if every nonprime attribute A in R is not partially dependent on *any* key of R.
- For example, suppose we had the table:

  LOTS(PropertyID, CountyName, Number, Area, Price, TaxRate)

  and we had the functional dependencies:

  PropertyID --> CountyName, Number, Area, Price, TaxRate

  CountyName, Number --> PropertyID, Area, Price, TaxRate

  CountyName --> TaxRate

  Area --> Price
- In this situation, TaxRate is nonprime and partially depends on the candidate key CountryName, Number.
- To put this in 2NF, we would set up a new relation for each partial key with its dependent attributes.

General Definition 3NF

- Recall a *trivial dependency* is a dependency of the form X-->A where $X \supseteq A$.
- A relation schema *R is in 3NF* if whenever a nontrivial FD X-->A holds in R, either (a) X is a superkey of R or (b) A is a prime attribute of R.
- For instance, if we had split LOTS from the last slide into:

  LOTS1(PropertyID, CountyName, Number, Area, Price)

  LOTS2(CountyName, TaxRate)

  It would be in 2NF but not 3NF, because Area --> Price in LOTS1 but Area is not a superkey and Price is not prime.
- Conditions (a) and (b) guarantee both being in 2NF and there being no transitive dependencies.

Boyce-Codd Normal Form

- The general definition of 3NF can be further simplified to give an even stronger normal form:
- A relation schema *R is in Boyce-Codd Normal Formal* (BCNF) if whenever a nontrivial FD X-->A holds in R, then X is a superkey of R.
- For example, suppose we put LOTS1 of the last slide into 3NF by making two relation schemas:

  LOTS1a(PropertyID, CountyName, Number, Area)

  LOTS1b(Area, Price)
- Now suppose we added the FD Area -->CountyName, since maybe from the plots size we can figure out which county we are in.
- Even with this new FD our decomposition is still in 3NF.
- However, it is not in BCNF.

Properties of Relational Decomposition

- We are now about to describe algorithms for putting our tables in normal forms.
- We assume we start with a *universal relation* which has all the attributes of the DB we want to store.
- We assume we also have a list of FDs, F, for these attributes.
- The goal is to decompose this universal relation R=(A_1,…, A_n) into relation schemas D = {R_1,…, R_m} in some normal form.
- We also want the decomposition to be *attribute preserving*:

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

- We also describe some other properties for the decomposition on the next couple of slides.

Dependency Preservation

- FDs contain important semantics of the data.
- It is useful that for each FD X-->Y our decomposition either has a table R_i which contains the attributes of X and Y or this dependency is inferable from the other dependencies that do appear.
- Let $\pi_{R_i}(F)$ denote those FDs in F^+ all of whose attributes lie in R_i.
- D = $\{R_1, \ldots, R_m\}$ is dependency preserving if:

$$((\pi_{R_1}(F)) \cup \ldots \cup (\pi_{R_n}(F)))^+ = F^+$$

Lossless Join Property

- Another property we like our decompositions to have is that we join all the tables together of our decomposition we do not get any spurious tuples.
- Let r be a relation state for our universal relation R. Formally, D = {R_1,…, R_m} has the *lossless join property* if it satisfies:

$$\pi_{R_1}(r) * \cdots * \pi_{R_m}(r)$$

Testing for the Lossless Join Property

Input: A universal relation R, a decomposition D = {R_1,…, R_m} of R and a set F of FDs.

1. Create an initial matrix S with one row i for each relation R_i in D, and one column j for each attribute A_j in R.
2. Set S(i,j) := b_{ij} for all matrix entries.
3. For each row i representing relation R_i

   {for each column j with attribute A_j

   {if (relation R_i includes attribute A_j) then set S(i,j) :=a_j;};};
4. Repeat the following loop until a complete loop execution results in no change to S:

   {for each FD X-->Y in F

   {for all rows in S that have the same symbols in the columns corresponding to attributes in X

   {make the symbols in the columns that correspond to an attribute in Y be the same in all these rows as follows: If any of the rows has an “a” symbol for the column, set the other rows to that same “a” symbol. If no “a” symbol exists, choose one of the “b” symbols and set the other rows to the same “b” symbol in that column ;};};};
5. If a row is made up entirely of “a” symbols then the decomposition has the lossless join property.

Example

- Suppose

  R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS}

  D= {R_1, R_2, R_3 }

  R_1=EMP ={SSN, ENAME}

  R_2=PROJ={PNUMBER, PNAME, PLOCATION}

  R_3,=WORKS_ON={SSN, PNUMBER, HOURS}

  F={SSN-->{ENAME}; PNUMBER-->{PNAME,PLOCATION}; {SSN,PNUMBER}-->HOURS}

  Initially, after step 3 get:

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R1	a1	a2	b13	b14	b15	b16
R2	b21	b22	a3	a4	a5	b26
R3	a1	b32	a3	b34	b35	a6

  after step 4 get:

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS	
R1	a1	a2	b13	b14	b15	b16	
R2	b21	b22	a3	a4	a5	b26	
R3	a1	a2	a3	a4	a5	a6	So has lossless join property

Binary Test for LJP

- A decomposition D={R1, R2} of R has the lossless join property with respect to F iff
  - The FD (R1 ∩ R2) --> (R1 -R2) is in F^+, or
  - The FD (R1 ∩ R2) --> (R2 -R1) is in F^+
- IF D has the lossless join property and D' is obtained from D by replacing an R_i with a decomposition D'' which has the LJP then D' has the lossless join property.