COS 597A: Principles of Database and Information Systems

Managing Functional Dependencies and Redundancy

General functional constraints (Review)

General form for relational model:

- Let $\alpha(R)$ denote the set of names of attributes
- in the schema for relation R
- Let X and Y be subsets of $\alpha(R)$

The functional dependency $X \rightarrow Y$ holds if for any instance I of R and for any pair of tuples t_1 and t_2 of R, $\pi_X(t_1) = \pi_X(t_2) \Rightarrow \pi_Y(t_1) = \pi_Y(t_2)$

· special cases: candidate keys, superkeys

Redundancy

- Functional dependencies capture redundancy in a relation e.g. area code → state: why store state?
- Redundancy good for reliability
- Redundancy bad for
 - space to store
 - repetitions
 - must maintain on changes
 - representation of one relationship
 - embedded in another

Example relation for a city elementary school system: school_child: (<u>name, st_addr, apt.</u>, birthday, school) st_addr -> school

consider a large apt. building

Solution: decompose

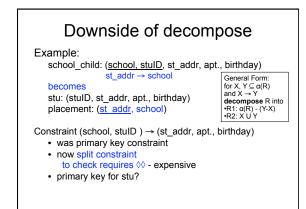
Example:

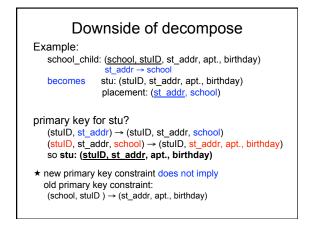
child: (<u>name, st_addr, apt.</u>, birthday) placement: (<u>st_addr</u>, school)

- child ◊◊ placement gives school_child because of functional dependency
- · space gain larger than space cost
- functional dependency now primary key constraint
- st_addr, school correspondence explicitly maintained

General Form:

- for X, Y $\subseteq \alpha(R)$ and X \rightarrow Y
- decompose R into
- R1: α(R) (Y-X) R2: X U Y





Decomposition: Formal Properties

- Let Φ be a set of functional dependencies (FDs) for a relational scheme R with attribute set $\alpha(R)$
- Let $\Phi^{\scriptscriptstyle +}$ denote the set of all FDs implied by Φ the closure of Φ
- Let X, $Y \subseteq \alpha(R)$, where $X \cap Y$ is not necessarily empty
- Decomposition of R into R₁: X and R₂:Y is - lossless if for every instance I of R that satisfies Φ $\pi_{\chi}(I) \otimes \pi_{\gamma}(I) = I$
 - guaranteed to get back R
 - dependency preserving if $(\Phi_{\mathsf{x}} \ \mathsf{U} \ \Phi_{\mathsf{Y}})^{\scriptscriptstyle +}$ = $\Phi^{\scriptscriptstyle +}$
 - can check all FDs for R by checking all for X and all for Y without doing JOIN

Implied functional dependencies

- Definition: a functional dependency X→Y is implied by Φ if X→Y holds whenever all functional dependences in Φ hold
- Armstrong's Axioms for attribute sets X, Y, Z
 - 1. if $X \subseteq Y$ then $Y \to X$ reflexivity
- 2. if $X \rightarrow Y$ then $\forall Z (XZ \rightarrow YZ)$ augmentation
- 3. if $X \to Y$ and $Y \to Z$ then $X \to Z$ transitivity
- Theorem: The set of all functional dependences obtained from Φ by repeated application of Armstrong's Axioms gives Φ⁺

Normal Forms

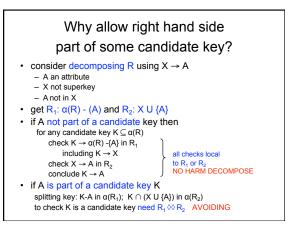
- How do we find "good " ("best"?) decomposition?
- Identify normal forms with desirable properties
- Decompose so resulting relations are in normal form

Boyce-Codd Normal Form (BCNF)

- Let R denote a relational scheme with attribute set $\alpha(R)$
- R is in BCNF with respect to a set Φ of FDs if for all FDs in Φ^+ of the form X \rightarrow Y with X, Y $\subseteq \alpha(R)$, at least one of
 - $\begin{array}{l} \ Y \subseteq \ X \ \ (trivial \ func. \ dep.) \\ \ X \ is \ a \ superkey \ for \ R \end{array}$
- very strong normal form
- can't always get dependency preserving decomposition into set of BCNF relations

Third Normal Form (3NF)

- Let R denote a relational scheme with attribute set $\alpha(R)$
- R is in 3NF with respect to a set Φ of FDs if for all FDs in Φ^+ of the form X \rightarrow Y with X, Y $\subseteq \alpha(R)$, at least one of
 - $Y \subseteq X$ (trivial func. dep.)
 - X is a superkey for R
 - each attribute A in Y-X is contained in a candidate key for R
- can always get lossless, dependency preserving decomposition into 3NF relations
- · cannot always remove all functional dependencies



Revisit example

Lossless-join decomposition? Dependency preserving decomposition? Normal forms?

school_child: (school, stulD, st_addr, apt., birthday)

becomes stu: (stulD, st addr, apt., birthday) placement: (st addr, school)

Constraint (school, stuID) → (st_addr, apt., birthday) • was primary key constraint • now split constraint

to check requires 00 - expensive

Decompositon to achieve 3NF

- Is polynomial-time algorithm for 3NF losslessjoin, dependency-preserving decomposition
- Can require adding "extra" relation.
- · Get at expense of redundancy

Example

R with attributes ABCD; AB primary key; other functional dependencies A→C; B→C decompose R1: ABD; R2: BC lossless? dependency-preserving?

Decompositon to achieve 3NF

- Is polynomial-time algorithm for 3NF lossless dependency-preserving decomposition
- Can require adding "extra" relation.
- Get at expense of redundancy

Example

R with attributes ABCD; AB primary key; $A \rightarrow C$; $B \rightarrow C$ decompose R1: ABD; R2: BC add R3: AC redundant because can get from R1 \otimes R2

Discussion

- Consider normal forms when designing relations.
- Using 3NF minimizes problems of general functional dependencies

 does not eliminate
- Use BCNF if can get it
 _ decomposition algorithm simpler too!