Losseless Join Decompositions - 3NF

July 18, 2003

1 Introduction

The purpose of this tutorial is to understand why we sometimes need to decompose schemas and how to do it appropriately.

Having redundant information not only is a waste of space but causes anomalies. The anomalies caused by redundancy are: update anomalies (potential inconsistencies), insertion anomalies and deletion anomalies.

To avoid these anomalies, we need to reduce redundancy by decomposing bad schemas. If the database designer is not careful when he decomposes a schema, he may lose information. A decomposition that doesn't lose information is called a Lossless-Join Decomposition. To be able to perform such a decomposition, the database designer needs to have more information about the schema. In practice he is given constraints like each supplier has a unique address, each person has a unique social insurance number, for each supplier and item there is a unique price etc... These constraints are called functional dependencies since the value of some attributes depend funcionally on the values of certain other attributes.

There are other kinds of constraints like a telephone number has 10 digits, the age of a person is less than 150 etc... These constraints are important for guarting against errors in the data, but they do not affect the design of the schema, we will not consider them here. They are important when dealing with data integrity.

2 Decomposing Schemas

- We define a relation scheme to be a set of attributes, $R = \{A1, A2, ..., Ak\}$. A decomposition is a set of (smaller) schemas $\rho = \{R1, R2, ..., Rn\}$ where $R = R1 \cup R2 \cup ... \cup Rn$. The Ri need not be disjoint.
- Relation = Schema + Data
- If r has schema R and $\{R1, R2, ..., Rn\}$ is a decomposition or R, then $\{\prod_{R1} r, ..., \prod_{Rn} r\}$ is the corresponding decomposition of r.

- Decomposing a relation may result in information loss because we may not be able to reconstruct the original relation.
- Let $\{R_1, ..., R_k\}$ be a decomposition of R and let $\mathcal{F}$ be a set of functional dependencies over R. The decomposition has a lossless join with resect to F iff for every relation r of R satisfying $\mathcal{F}$, $r = \prod_{R_1} r \bowtie ... \bowtie \prod_{R_n} r$
- Decomposing a relation may also not preserve the set of functional dependencies. It is important to keep these functional dependencies when we decompose so that the DBMS can enforce them in each of the relations that we will create. A decomposition $\rho = \{R1, R2, ..., Rn\}$ preserves the fd's of $\mathcal{F}$ if we can find every fd of $\mathcal{F}^+$ from the union of all fd's projected from $\mathcal{F}$: $\{\prod_{R1} \mathcal{F}, ..., \prod_{Rn} \mathcal{F}\}$
- Functional dependencies can guarantee that a decomposition does not lose information, but they do not guarantee that all decompositions are lossless.
- A lossless-join decomposition does not necessarily preserve functional dependencies.
- A losslses-join decomposition does not necessarily produce 3NF relations.

3 3NF Decomposition

3.1 Definition and Theorem

- A schema R is in 3NF iff $\forall X \rightarrow A \in \mathcal{F} \Longleftrightarrow \begin{cases} \text{X}\rightarrow A \text{ is trivial} \\ X \text{ is a superkey} \\ A \text{ is contained in a key} \end{cases}$
- Every 1NF relation has a decomposition in 3NF relations which are lossless-join and preserve the functional dependencies.

3.2 Algorithm for 3NF Decomposition:

We assume that $\mathcal{F}$ is a minimal closure.

1. For each $X \rightarrow A \in \mathcal{F}$ create a relation of schema (XA).
2. If no key is contained in one of the schemas created in the first step, create a relation of schema Y where Y is a key.
3. If after the first step there exists a relation R_1 with a schema X_1A_1 contained in the schema X_2A_2 of a relation R_2, delete R_1.

4 Exercices

1. Are these schemas in 3NF?
2. Decompose the relations, as necessary, into collections of relations that are in 3NF.
   (a) $R = \{city, street, zip\}$
   $\mathcal{F} = \{city, street \rightarrow zip,\ zip \rightarrow city\}$
   (b) $R = ABC$
   $\mathcal{F} = \{A \rightarrow B,\ B \rightarrow C\}$
   (c) $R = ABCD$
   $\mathcal{F} = \{AB \rightarrow C,\ C \rightarrow D,\ D \rightarrow A\}$
   (d) $R = ABCD$
   $\mathcal{F} = \{B \rightarrow C,\ B \rightarrow D\}$
   (e) $R = ABCD$
   $\mathcal{F} = \{AB \rightarrow C,\ BC \rightarrow D,\ CD \rightarrow A,\ AD \rightarrow B\}$