ST. XAVIER’S COLLEGE

**(Affiliated to Tribhuvan University)**

Maitighar, Kathmandu



**Database Management System Assignment #10**

**Submitted by:**

Bikash Paneru  
013BSCCSIT012

**Submitted to:**

|  |  |
| --- | --- |
| Er. Sanjay Kumar Yadav Lecturer, St. Xavier’s College |  |

**Date of Submission: October 8, 2015**

**1. FUNCTIONAL DEPENDENCIES**

**1.1. Basic Concept**

A functional dependency occurs when one attribute in a relation uniquely determines another attribute. This can be written A -> B which would be the same as stating "B is functionally dependent upon A."

Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A1, A2,..., An, then those two tuples must have to have same values for attributes B1, B2, ..., Bn.

Functional dependency is represented by an arrow sign (→) that is, X→Y, where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side.

**1.2. Closure of a set of Functional Dependencies**

If F is a set of functional dependencies then the closure of F, denoted as F+, is the set of all functional dependencies logically implied by F. Armstrong's Axioms are a set of rules, that when applied repeatedly, generates a closure of functional dependencies.

**Reflexive rule** − If alpha is a set of attributes and beta is\_subset\_of alpha, then alpha holds beta.

**Augmentation rule** − If a → b holds and y is attribute set, then ay → by also holds. That is adding attributes in dependencies, does not change the basic dependencies.

**Transitivity rule** − Same as transitive rule in algebra, if a → b holds and b → c holds, then a → c also holds. a → b is called as a functionally that determines b.

**1.3. Closure of Attribute Sets**

1. To test whether a set of attributes tex2html_wrap_inline958 is a superkey, we need to find the set of attributes functionally determined by tex2html_wrap_inline958 .
2. Let tex2html_wrap_inline958 be a set of attributes. We call the set of attributes determined by tex2html_wrap_inline958 under a set *F* of functional dependencies the **closure** of tex2html_wrap_inline958 under *F*, denoted tex2html_wrap_inline1292 .
3. The following algorithm computes tex2html_wrap_inline1292 :

*result* := tex2html_wrap_inline958

**while** (changes to *result*) **do**

**for each** functional dependency tex2html_wrap_inline1240

**in**  *F* **do**

**begin**

**if**  tex2html_wrap_inline1302 *result*

**then**  *result* := *result* tex2html_wrap_inline1304 ;

**end**

1. If we use this algorithm on our example to calculate tex2html_wrap_inline1306 then we find:
   * We start with *result* = AG.
   * *A tex2html_wrap_inline1090 B*causes us to include B in *result*.
   * *A tex2html_wrap_inline1090 C*causes *result* to become ABCG.
   * *CG tex2html_wrap_inline1090 H*causes *result* to become ABCGH.
   * *CG tex2html_wrap_inline1090 I*causes *result* to become ABCGHI.
   * The next time we execute the while loop, no new attributes are added, and the algorithm terminates.
2. This algorithm has worst case behavior quadratic in the size of *F*. There is a linear algorithm that is more complicated.

**2. DECOMPOSITION**

Decomposition means replacing a relation with a set of smaller relations.

**2.1. Lossless – Join Decomposition**

Let *R* be a relation schema.

Let *F* be a set of functional dependencies on *R*.

Let tex2html_wrap_inline1620 and tex2html_wrap_inline1622 form a decomposition of *R*.

The decomposition is a lossless-join decomposition of *R* if at least one of the following functional dependencies are in tex2html_wrap_inline1628 :

* 1. tex2html_wrap_inline1630
  2. tex2html_wrap_inline1632

Why is this true? Simply put, it ensures that the attributes involved in the natural join ( tex2html_wrap_inline1634 ) are a candidate key for at least one of the two relations.

This ensures that we can never get the situation where spurious tuples are generated, as for any value on the join attributes there will be a unique tuple in **one** of the relations.

**2.2. Dependency Preservation**

A desirable property in database design is **dependency preservation**. We would like to check easily that updates to the database do not result in illegal relations being created. It would be nice if our design allowed us to check updates without having to compute natural joins. To know whether joins must be computed, we need to determine what functional dependencies may be tested by checking each relation individually.

* Let *F* be a set of functional dependencies on schema *R*.
* Let tex2html_wrap_inline1550 be a decomposition of *R*.
* The **restriction** of *F* to tex2html_wrap_inline1556 is the set of all functional dependencies in tex2html_wrap_inline1628 that include only attributes of tex2html_wrap_inline1556 .
* Functional dependencies in a restriction can be tested in one relation, as they involve attributes in one relation schema.
* The set of restrictions tex2html_wrap_inline1660 is the set of dependencies that can be checked efficiently.
* We need to know whether testing only the restrictions is sufficient.
* Let tex2html_wrap_inline1662 .
* *F*' is a set of functional dependencies on schema *R*, but in general, tex2html_wrap_inline1668 .
* However, it may be that tex2html_wrap_inline1670 .
* If this is so, then every functional dependency in *F* is implied by *F*', and if *F*' is satisfied, then *F* must also be satisfied.

A decomposition having the property that tex2html_wrap_inline1670 is a **dependency-preserving** decomposition.