**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Lab Assignment #10**

**Submitted by:**

Arun Sanjel  
013BSCCSIT010

**Submitted to:**

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

# Date of Submission: Thursday, October 08, 2015

# Functional Dependencies

## Basic Concepts

The concept of functional dependency (also known as normalization was introduced by professor Codd in 1970 when he defined the first three normal forms (first, second and third normal forms). Normalization is used to avoid or eliminate the three types of anomalies (insertion, deletion and update anomalies) which a database may suffer from. These concepts will be clarified soon, but first let us define the first three normal forms.

Functional dependency is a relationship that exists when one attribute uniquely determines another attribute.

If R is a relation with attributes X and Y, a functional dependency between the attributes is represented as X->Y, which specifies Y is functionally dependent on X. Here X is a determinant set and Y is a dependent attribute. Each value of X is associated precisely with one Y value.

## Closure of a Set of Functional Dependencies

* We need to consider all functional dependencies that hold. Given a set *F* of functional dependencies, we can prove that certain other ones also hold. We say these ones are **logically implied** by *F*.
* Suppose we are given a relation scheme *R*=(*A*,*B*,*C*,*G*,*H*,*I*), and the set of functional dependencies:

A *tex2html_wrap_inline1090* B

A *tex2html_wrap_inline1090* C

C *tex2html_wrap_inline1090* H

**C *tex2html_wrap_inline1090* I**

B *tex2html_wrap_inline1090* H

Then the functional dependency tex2html_wrap_inline1194 is logically implied.

* To see why, let tex2html_wrap_inline940 and tex2html_wrap_inline946 be tuples such that

tex2html_wrap_inline1200

As we are given A*tex2html_wrap_inline1090*B, it follows that we must also have

tex2html_wrap_inline1204

Further, since we also have B*tex2html_wrap_inline1090*H, we must also have

tex2html_wrap_inline1208

Thus, whenever two tuples have the same value on *A*, they must also have the same value on *H*, and we can say that A*tex2html_wrap_inline1090*H.

* The **closure** of a set *F* of functional dependencies is the set of all functional dependencies logically implied by *F*.
* We denote the closure of *F* by tex2html_wrap_inline1222 .
* To compute tex2html_wrap_inline1222 , we can use some rules of inference called **Armstrong's Axioms**:
  + - **Reflexivity rule**

if tex2html_wrap_inline958 is a set of attributes and tex2html_wrap_inline1158 , then tex2html_wrap_inline1058 holds.

* + - **Augmentation rule**

if tex2html_wrap_inline1058 holds, and tex2html_wrap_inline1234 is a set of attributes, then tex2html_wrap_inline1236 holds.

* + - **Transitivity rule**

if tex2html_wrap_inline1058 holds, and tex2html_wrap_inline1240 holds, then tex2html_wrap_inline1242 holds.

* These rules are **sound** because they do not generate any incorrect functional dependencies. They are also **complete** as they generate all of tex2html_wrap_inline1222 .
* To make life easier we can use some additional rules, derivable from Armstrong's Axioms:
  + **Union rule**

 if tex2html_wrap_inline1058 and tex2html_wrap_inline1242 , then tex2html_wrap_inline1250 holds.

* + **Decomposition rule**

 if tex2html_wrap_inline1250 holds, then tex2html_wrap_inline1058 and tex2html_wrap_inline1242 both hold.

* + **Pseudo transitivity rule**

if tex2html_wrap_inline1058 holds, and tex2html_wrap_inline1260 holds, then tex2html_wrap_inline1262 holds.

* Applying these rules to the scheme and set *F* mentioned above, we can derive the following:
  + A*tex2html_wrap_inline1090*H, as we saw by the transitivity rule.
  + CG*tex2html_wrap_inline1090*HIby the union rule.
  + AG*tex2html_wrap_inline1090*Iby several steps:
    - Note that A*tex2html_wrap_inline1090*Cholds.
    - Then AG*tex2html_wrap_inline1090*CG, by the augmentation rule.
    - Now by transitivity, AG*tex2html_wrap_inline1090*I.

## Closure of Attribute Sets

After finding a set of functional dependencies that are hold on a relation, the next step is to find the Super key for that relation (table). The set of identified functional dependencies play a vital role in finding the key for the relation. We can decide whether an attribute (or set of attributes) of any table is a key for that table or not by identifying the attribute or set of attributes’ closure. If A is an attribute, (or set of attributes) then its attribute closure is denoted as A+.]

 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 .

 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 .

 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**

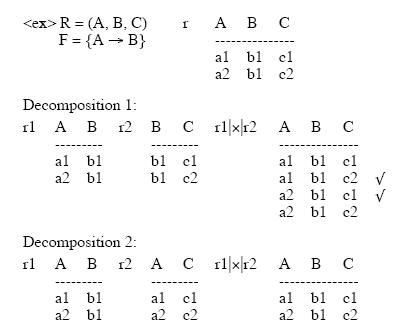
 If we use this algorithm on our example to calculate tex2html_wrap_inline1306 then we find:

* 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.

 This algorithm has worst case behavior quadratic in the size of *F*. There is a linear algorithm that is more complicated.

# Decomposition

Let R be a relation schema. A set of relation schemas is a decomposition of R if. That is, every attribute in R appears in at least one for. Let r be a relation on R, and let. That is, is the database that results from decomposing R into .One of the properties of bad design suggests decomposing a relation into smaller relations.Must achieve lossless-join decomposition.



## Lossless join Decomposition

Let {R1, R2 } be a decomposition of R (R1 U R2 = R); the decomposition is lossless if for every legal instance r of R:

r = ΠR1(r) |X| ΠR2(r)

Lossless join property is necessary if the decomposed relation is to be recovered from its decomposition.

Let R be a schema and F be a set of FD’s on R, and α = (R1, R2) be a decomposition of R. Then α has a lossless join with respect to F iff

R1 ∩ R2 -> R1 (or R1 - R2 ) or

R2 ∩ R1 -> R2 (or R2 - R1 )

where such FD exist in Closure of F. This is a sufficient condition, but not a necessary condition

From the previous example:

R = (ABC) F = {A -> B}

R1 = (AB), R2 = (AC)

R1∩ R2 = A, R1- R2 = B

check A -> B in F ? Yes. Therefore lossless

R1 = (AB), R2 = (BC)

R1∩ R2 = B, R1 - R2 = A , R2 - R1= C

check B -> A in F ? NO

check B -> C in F ? NO

So, this is lossy join.

## Dependency Presentation

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 be a decomposition of R.

•The restriction of F to is the set of all functional dependencies in that include only attributes of .

•Functional dependencies in a restriction can be tested in one relation, as they involve attributes in one relation schema.

•The set of restrictions is the set of dependencies that can be checked efficiently.

•We need to know whether testing only the restrictions is sufficient.

•Let F' is a set of functional dependencies on schema R, but in general, .

•However, it may be that .

•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 is a dependency-preserving decomposition.