**ST. XAVIER’S COLLEGE**

**Maitighar, Kathmandu**



**DATABASE MANAGEMENT SYSTEM**

**Theory Assignment #10**

**Submitted by**

Utsav Luitel

013BSCCSIT046

**Submitted to**

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

**Date of Submission:**

**Thursday, October 08, 2015**

**1.1 Functional dependency:**

**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.

**1.2 Closure of a Set of Functional Dependencies**

1. 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*.
2. Suppose we are given a relation scheme *R*=(*A*,*B*,*C*,*G*,*H*,*I*), and the set of functional dependencies:
3. *A tex2html_wrap_inline1090 B*
4. *A tex2html_wrap_inline1090 C*
5. *CG tex2html_wrap_inline1090 H*
6. *CG tex2html_wrap_inline1090 I*
7. *B tex2html_wrap_inline1090 H*

Then the functional dependency tex2html_wrap_inline1194 is logically implied.

1. 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*.

1. The **closure** of a set *F* of functional dependencies is the set of all functional dependencies logically implied by *F*.
2. We denote the closure of *F* by tex2html_wrap_inline1222 .
3. 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.
4. 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 .
5. 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.
6. 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 HI*by the union rule.
   * *AG tex2html_wrap_inline1090 I*by several steps:
     + Note that *A tex2html_wrap_inline1090 C*holds.
     + Then *AG tex2html_wrap_inline1090 CG*, by the augmentation rule.
     + Now by transitivity, *AG tex2html_wrap_inline1090 I*.

(You might notice that this is actually pseudo transivity if done in one step.)

# Functional Dependencies

# 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.

A Functional Dependencies is a relationship between an attribute "Y" and a determinant (1 or more other attributes) "X" such that for a given value of a determinant the value of the attribute is uniquely defined.

* X is a determinant
* X determines Y
* Y is functionally dependent on X
* X → Y
* X →Y is trivial if Y ⊆ X

## Example:

Let R be  NewStudent(*stuId, lastName, major, credits, status, socSecNo*)

FDs in R include

* *{stuId}→{lastName}*, but not the reverse
* *{stuId} →{lastName, major, credits, status, socSecNo, stuId}*
* *{socSecNo} →{stuId, lastName, major, credits, status, socSecNo}*
* *{credits}→{status}*, but not *{status}→{credits}*

## Closure of a Set of Functional Dependencies:

**Armstrong's Axioms**

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.

**Trivial Functional Dependency**

**Trivial**

If a functional dependency (FD) X → Y holds, where Y is a subset of X, then it is called a trivial FD. Trivial FDs always hold.

**Non-trivial**

If an FD X → Y holds, where Y is not a subset of X, then it is called a non-trivial FD.

**Completely non-trivial**

If an FD X → Y holds, where x intersect Y = Φ, it is said to be a completely non-trivial FD.

Decomposition

* Decomposition – the process of breaking down in parts or elements.
* Decomposition in database means breaking tables down into multiple tables
* From Database perspective means going to a higher normal form

Important that decompositions are “good”,

Two Characteristics of Good Decompositions

1. Lossless.

Lossless means functioning without a loss.

In other words, retain everything.

Important for databases to have this feature.

Formal Definition

* Let *R* be a relation schema.
* Let *F* be a set of functional dependencies on *R*.
* Let and 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 *F+*

1) R1 ∩ R2 🡪 R1

2) R1 ∩ R2 🡪 R2