ST. XAVIER’S COLLEGE

(Affiliated to Tribhuvan University)

Maitighar, Kathmandu

****

**Database Management System**

Theory Assignment # 10

**SUBMITTED BY:**

Biraj Poudyal

013BSCCSIT015

**SUBMITTED TO:**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **Lecturer** |  |

Submission Date: 8th October 2015

**THEORY ASSIGNMENT**

1. **FUNCTIONAL DEPENDENCIES**
   1. **Basic Concept**

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

Functional Dependency is the starting point for the process of normalization. Functional dependency exists when a relationship between two attributes allows you to uniquely determine the corresponding attribute’s value. If ‘X’ is known, and as a result you are able to uniquely identify ‘Y’, there is functional dependency. Combined with keys, normal forms are defined for relations.

Examples

Bear Number determines Student Name:

BearNum ---> StuName

Department Number and Job Rank determine Security Clearance:

(DeptNum, JRank) --->SecClear

Social Security Number determines Employee Name and Salary:

SSN ---> (EmpName, Salary)

Additionally, the above can be read as:

SSN --->EmpName and SSN Salary

* 1. **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*

*CG tex2html_wrap_inline1090 H*

*CG 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.
* **Pseudotransitivity 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 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 pseudotransivity if done in one step.)

* 1. **Closure of Attribute Sets**

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.

1. **Decomposition**
   1. **Loseless-Join Dependencies**

Can also be called Nonadditive. If you decompose a relation R into relations R_1 and R_2 you will guarantee a Lossless-Join if R_1⋈R_2 = R.

If R is split into R1 and R2, for the decomposition to be lossless then at least one of the two should hold true.

Projecting on R1 and R2, and joining back, results in the relation you started with.[[1]](https://en.wikipedia.org/wiki/Lossless-Join_Decomposition#cite_note-1) Let R be a relation schema.

Let F be a set of [functional dependencies](https://en.wikipedia.org/wiki/Functional_dependency) on R.

Let R_1 and R_2 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+ (where F+ stands for the closure for every attribute or attribute sets in F):[[2]](https://en.wikipedia.org/wiki/Lossless-Join_Decomposition#cite_note-2)

* R_1 ∩ R_2 → R_1
* R_1 ∩ R_2 → R_2
  1. **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 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,

• 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.