**ST. XAVIER’S COLLEGE**

**(Affiliated to Tribhuvan University)**

**Maitighar, Kathmandu**

****

**Database Management System**

**Theory Assignment #10**

**SUBMITTED BY:**

**Siddhant Rimal**

**013BSCCSIT039**

**SUBMITTED TO**

|  |  |
| --- | --- |
| **Er. Sanjay Kr. Yadav**  **( Lecturer )** |  |
| **Department of Computer Science** | |

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

10.1. Functional Dependencies

### Basic Concepts

### For a given relation R with attribute X and Y, Y is said to be functionally dependent on X, if given value for each X uniquely determines the value of the attribute in Y. X is called determinant of the functional dependency (FD) and functional dependency denoted by X→Y.

### Functional dependencies are constraints on the set of legal relations. It defines attributes of relation, how they are related to each other.

### It determines unique value for a certain set of attributes to the value for another set of attributes that is functional dependency is a generalization of the notation of key.

### Functional dependencies are interrelationship among attributes of a relation.

|  |
| --- |
| Example 1: |
| consider a relation supplier  Supplier(supplier\_id#,sname,status,city)Here, sname, status and city are functionally dependent on supplier\_id. Meaning is that each supplier id uniquely determines the value of attributes supplier name,supplier status and city This can be express bySupplier.supplier\_id→supplier.sname Supplier.supplier\_id→supplier.status Supplier.supplier\_id→supplier.cityOr simply, supplier\_id→ sname supplier\_id→ status supplier\_id→cityQuestion: is following functional dependency is valid ? sname→status sname→cityAnswer: it is true only if sname is unique, otherwise false.  |  |  | | --- | --- | | Valid case | | | Sname | status | | X | Good | | Y | Good |  |  |  | | --- | --- | | Invalid case | | | Sname | status | | X | Good | | Y | Good | | X | Bad | |

|  |
| --- |
| Example 2: |
| Consider a relation student-infoStudent-info(name#,course#,phone\_no,major,prof,grade) That is, {name,course} is composite primary key This relation has the following functional dependencies {name→phone\_no, name→major, name,course→grage, course→prof} |

### Closure of a set of functional dependencies

### For a given set of functional dependencies F, there are certain other functional dependencies that are logically implies by F. (i.e. if A→B and B→C, then we can write A→C). the set of all functional dependencies logically implies F is the closure of F. Closure of F is denoted by F+.

### We can find all of F+ by applying Armstrong’s Axioms:

### if β ⊆ α then α → β or α →α (reflexive)

### if α → β then γ α →γ β (augmentation)

### if α → β and β →γ then α →γ (transitivity)

### Example: Let R=(A,B,C,G,H,I) F={A→B, A→C,CG→H,CG→I,B→H} Compute closure of F+

### Closure of F+ computed as follow:

### A→H

### by transitivity A→B and B→H

### AG→I

### By augmenting A→C with G we get AG→CG and then by transitivity with CG→I we get AG→I

### CG→HI

### From CG →H and CG→I “union rule” can be inferred from definition of functional dependency

### or,

### Augmentation of CG→I to infer CG→CGI, argumentation of CG→H to infer CGI→HI, and then transitivity.

### Hence, F+={ A→A, B→B, C→C, H→H, G→G, I→I, A→B, A→C, CG→H, CG→I, CG→HI, B→H, A→H, AG→I, G→Hi}

### Closure of attribute sets

### The closure of X under a set of functional dependencies F, written as X+, is the set of attributes {A1,A2, . . Am} such that the FD X→Ai for Ai∈X+ follows from F by the inference axioms for functional dependencies.

### Example: Let X=BCD and F={A→BC,CD→E,E→C,D→AEH,ABH→BD,DH→BC}. Compute the closure X+ of X under F.

### initialize X+:=BCD.

### Since left hand side of the FD CD→E is a subset of X+ (i.e CD⊆ X+), X+ is augmented by the right hand side of the FD (i.e. E) thus now X+:=BCDE.

### Similarly, D⊆ X+, the right hand side of the FD D→AEH is added to X+. Hence now X+:=ABCDEH.

### Now X+ can not be augmented any further because no FDs left hand side is subset of X+.

**Application of Attribute Closure**

* **Testing superkey**

To test α is a superkey we compute α + and check whether α + contains all attributes of R. if so α is a superkey, otherwise not.

* **Testing functional dependencies**  
  To check a functional dependency α → β holds check whether β ⊆ α +. If so α → β ;  
  otherwise not.

10.2. Decomposition

### Lossless Join Decomposition

### The decomposition of relation schema R= (A1, A2, . ,An) is a set of relation schema { R1, R2, ----- Rm}, such that Ri ⊆ R ∀ 1 ≤ i≤ m and R1 R U 2 U . . U Rm = R. That is all attributes of an original schema (R) must appear in the decomposition (R1, R2). That is, R= R1 U R2. if R ≠ R1 U R2 then such decomposition called lossey join decomposition. That is, R ≠ ∏R1(R) ∏R2(R). Decomposition should *lossless join decomposition.*

### A decomposition of relation schema R into R1 and R2 is lossless join iff at least one of the following dependencies is in F+. R1 ∩ R2→R1 R1 ∩ R2→R2

|  |
| --- |
| Example 1 |
| Consider the relational schemaBranch\_loan = ( branch\_name, branch\_city, assets, customer\_name, loan\_no, amount)  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | branch\_name | branch city | assets | customer name | loan no | amount | | kathmandu | baneshwor | 25000 | rohan | L-15 | 3000 | | Lalitpur | patan | 19000 | mohan | L-17 | 5000 | | Kathmandu | baneshwor | 25000 | raju | L-19 | 10000 | | Pokhara | Prithibinagar | 17000 | manoj | L-10 | 7000 | | Lalitpur | patan | 19000 | swikar | L-30 | 9000 |  The problems in the relational schema branch\_loan can be resolved if we replace it with the following relation schemas. Branch (# branch\_name, branch\_city, assets) Loan (customer\_name, loan\_number, branch\_name, amount) |

|  |
| --- |
| Example 2 |
| Consider the relation schema to store the information a student maintain by the university. Student\_info (#name, course, phone\_no, major, prof, grade)  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | name | course | phone\_no | major | prof | grade | | John | 353 | 374537 | ComputerScience | Smith | A | | Scott | 329 | 427993 | Mathematics | James | S | | John | 328 | 374537 | ComputerScience | Adams | A | | Allen | 432 | 729312 | Physics | Blake | C | | Turner | 523 | 252731 | Chemistry | Miller | B | | John | 320 | 374537 | ComputerScience | Martin | A | | Scott | 328 | 727993 | Mathematics | Ford | B | |

### Dependency Preservation

### Getting lossless decomposition is necessary. But of course, we also want to keep dependencies, since losing a dependency means, that the corresponding constraint can be check only through natural join of the appropriate resultant relation in the decomposition. This would be very expensive, so, our aim is to get a lossless dependency preserving decomposition.

|  |
| --- |
| Example: |
| R=(A, B, C), F={A🡪B, B🡪C} Decomposition of R: R1=(A, C) R2=(B, C) Does this decomposition preserve the given dependencies?Solution: In R1 the following dependencies hold: F1’={A🡪A, C🡪C, A🡪C, AC🡪AC} In R2 the following dependencies hold: F2’= {B🡪B, C🡪C, B🡪C, BC🡪BC} The set of nontrivial dependencies hold on R1 and R2: F':= {B🡪C, A🡪C} A🡪B can not be derived from F’, so this decomposition is NOT dependency preserving.Example: R=(A, B, C), F={A🡪B, B🡪C} Decomposition of R: R1=(A, B) R2=(B, C) Does this decomposition preserve the given dependencies?Solution: In R1 the following dependencies hold: F1={A🡪B, A🡪A, B🡪B, AB🡪AB} In R1 the following dependencies hold: F2= {B🡪B, C🡪C, B🡪C, BC🡪BC} F’= F1’ U F2’ = {A🡪B, B🡪C, trivial dependencies} In F’ all the original dependencies occur, so this decomposition preserves dependencies |

### Definition of Dependency preservation: A decomposition D = {R1, …, Rm} of R is dependency-preserving wrt a set F of FDs if

### (F1 ∪ … ∪ Fm)+ = F+,

### Where Fi means the projection of the dependency set F onto Ri. Fi =Π Ri(F+) denotes a set of FDs X → Y in F+ such that all attributes in X ∪ Y are contained in Ri:

### Fi=Π Ri(F+) ={ X→Y| {X,Y}⊆ Ri and X→Y ∈ F+ }

### Reference:

### [1] TL Paneru, Integrity constraints, PDF, Available under DBMS section: <https://drive.google.com/file/d/0B6sGyhABc5XENGFxeERwWlVqRXc/view>

### [2] Relational Database Design, “Decomposition”,PDF, Available: <https://drive.google.com/file/d/0B6sGyhABc5XEeVNXTWNhQV9JMG8/view>

### [3] SZTakiWeb, PDF, Available: <http://www.sztaki.hu/~fodroczi/dbs/dep-pres-own.pdf>