**Functional Dependencies**

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.

**Definition:**

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.

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 by

Supplier.supplier\_id→supplier.sname

Supplier.supplier\_id→supplier.status

Supplier.supplier\_id→supplier.city

Or simply,

supplier\_id→ sname

supplier\_id→ status

supplier\_id→city

Question: is following functional dependency is valid?

sname→status

sname→city

Answer: 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-info

Student-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}

Functional dependency X→Y satisfied on the relation R/ hold on R. FD X→Y is satisfied on relation R if the cardinality of ∏Y(σ x=x(r)) is at most one. That is if, two tuples ti and tj of R have the same X value then the corresponding value of Y must identical.

Let R be a relational schema

α ⊆ R and β ⊆ R

then the functional dependency α → β holds on R iff for any legal relation r(R), whenever any two tuples t1 and t2 of r agree on the attributes α then they also agree on the attributes β .

That is, if t1[α ]=t2[α ] then t1[ β ]=t2[ β ].

**Application of Functional dependencies**

Functional dependencies are applicable

* To test the relation whether they are legal under a given set of functional dependency.
* To specify the constraints for the legal relation
* Let r is a relation and F is a given set of functional dependencies. If r satisfies F, then we determine that r is legal under a given set of functional dependency F
* We say that f holds on R if all legal relations on R satisfy the set of functional dependencies F.

**Types of Functional Dependencies**

**1. Fully functionally dependency**

For a given relation schema R, FD X→Y, Y is said to be fully functionally dependent on X if there is no Z (where Z is a proper subset of X) such that Z→Y.

Example: Let us consider relational schema R=(A,B,C,D,E,H) with the FDs

F={A→BC,CD→E,C→E,CD→E,CD→AH, ABH→BD,DH→BC}

* Here, the FD A→BC is left reduced, so clearly, BC is fully functionally dependent on A (because there is no possible proper subset of only element A)
* Here, the FDs CD→E, C→E where E is functionally dependent on CD and again E is functionally dependent on subset of CD. That is C (i.e. C→E). Hence E is not fully functionally dependent on CD.

Example: Consider a relation sales

Sales (product\_id#,sales\_date#,quantity,product\_name)

With the following functional dependencies

F={product\_id,sales\_date→quantity, product\_id→quantity, product\_id→product\_name}

* Here,. FDs product\_id,sales\_date→quantity, product\_id→quantity, quantity is not fully functional dependent on product\_is,sales\_date.
* Here, functional dependency product\_id→product\_name, product\_name is fully functional dependent on product\_id.

**2. Partial functional dependency**

For a given relation schema R with set of functional dependency F on attribute of R. Let K as a candidiate key in R. if X is a proper subset of K and X and X→A then A is said to be partially dependent on K.

Example: Consider a relation schema ‘student\_course\_info’

student\_course\_info(name#,course#,grade,phone\_no,major,course\_department) with the following FDs

{name→phone\_no,major

course→course\_department,

name,course→grade

}

Here {name,course} is a candidate key. Here grade is fully functionally dependent on {name,course}. If thee is a possible FD name→grade then we can not say grade is fully functionally dependent on {name,course}. Here phone\_no, major and course\_department are partially dependent on {name,course}

**3. Transitive dependency**

For a given relational schema R with set of functional dependency F. Let X and Y be the subset of r anf Let A be the attribute of R s.t. X⊄ Y, A⊄ XY. If the functional dependencies {X→Y, Y→A} implies by F (i.e. X→Y→A) then A is said to be transitively dependent on X.

Example:

Let us consider relational schema ‘prof\_info’

prof\_info=(prof\_name#,department\_name, head\_of\_department)

with the set functional dependency

F={prof\_name→department\_name, department\_name→head\_of\_department}

Here prof\_name→department\_name→head\_of\_department so head\_of\_department is transitively dependent on the key prof\_name.

Example:

Let R=(A,B,C,D,E) and FDs F={AB→C,B→D,C→E}

Here AB act a candidate key and E is transitively dependent on the key AB, ince AB→C→E).

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

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,

AG→I

A→C,CG→H,CG→I,CG→HI,B→H,A→H,

AG→I,CG→Hi

}

Here, first six FDs obtain by reflexive axiom.

We can further simplify the the computation of F+ by using the following addition rule.

(a) if α → β holds and α → γ holds, then α → β γ (Additivity or union rule)

(b) if α → β γ holds then α → β holds and α → γ holds (projectivity/decomposion)

(c) if α → β holds and γ β →δ holds then α γ →δ holds (pseudotransitivity)

Examples: Let R=(A,B,C,D) and F={A→B,A→C,BC→D} then compute F+.

• Since A→B and A→C then by union rule A→BC.

• Since BC →D, then by projective/decomposition B→D, C→D. Again by transitivity A→B & B→D ⇒ A→D and A→C and C→D ⇒ A→D.

• Hence, F+ ={A→A, B→B, C→C, D→D, A→B, A→C, BC→D, B→D, C→D, A→D}

**Attribute Closure**

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**

**1. 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.

**2. Testing functional dependencies**

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

**Decomposition**

The idea of decomposition is break down large and complicated relation in no. of simple and small relations which minimized data redundancy. It can be consider principle to solve the relational model problem.

**Definition**

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= R1U R2. if R ≠ R1U 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+.

R1I R2→R1

R1I R2→R2

Example 1: The problems in the relational schema branch\_loan (illustrated in above example) 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, amoun)

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 Computer Science Smith A

Scott 329 427993 Mathematics James S

John 328 374537 Computer Science Adams A

Allen 432 729312 Physics Blake C

Turner 523 252731 Chemistry Miller B

John 320 374537 Computer Science Martin A

Scott 328 727993 Mathematics Ford B

**Problems:**

**1. Redundancy:**

Data for major and phone\_no of student are stored several times in the database once for each course that is taken by a student.

**2. Complicates updating:**

Multiple copies of some facts may lead to update which leads possibility of inconsistency.

Here, change of phone\_no of John is required we need to update three records (tuples) corresponding to the student John. If one of the three tuples is not changed there will be inconsistency in the date.

**3. Complicate insertion**

If this is only the relation in the database showing the association between a faculty member and the course he / she teaches, then the information that a given professor is teaching a given course cannot be entered in the database unless a student is registered in the course.

**4. Deletion Problem:**

If the only one student is registered in a given course then the information as to which professor is offering the course will be lost if this is only the relation in the database showing the association between the faculty member and the course he / she teaches. If database have another relation that establishes the relationship between a course, the deletion of 4th & 5th tuple in this relation will not cause the information about the councils teach to be lost.