Functional Dependencies

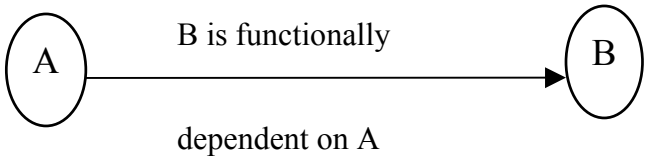
Hi! We are going to discuss functional Dependencies.

For our discussion on functional dependencies assume that a relational schema has attributes (A, B, C... Z) and that the whole database is described by a single universal relation called R = (A, B, C, ..., Z). This assumption means that every attribute in the database has a unique name.

What is functional dependency in a relation?

A ***functional dependency*** is a property of the semantics of the attributes in a relation. The semantics indicate how attributes relate to one another, and specify the functional dependencies between attributes. When a functional dependency is present, the dependency is specified as a constraint between the attributes.

Consider a relation with attributes A and B, where attribute B is functionally dependent on attribute A. If we know the value of A and we examine the relation that holds this dependency, we will find only one value of B in all of the tuples that have a given value of A, at any moment in time. Note however, that for a given value of B there may be several different values of A.



In the figure above, A is the determinant of B and B is the consequent of A.

The determinant of a functional dependency is the attribute or group of attributes on the left-hand side of the arrow in the functional dependency. The consequent of a fd is the attribute or group of attributes on the right-hand side of the arrow.

Identifying Functional Dependencies

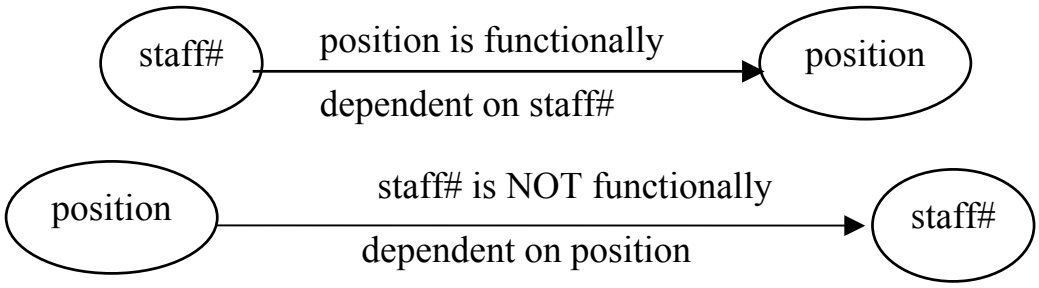
Now let us consider the following Relational schema

STAFFBRANCH

staff#	sname	position	salary	branch#	baddress
SL21	Kristy	manager	30000	B005	22Deer Road
SG37	Debris	assistant	12000	B003	162Main Street
SG14	Alan	supervisor	18000	B003	163Main Street
SA9	Traci	assistant	12000	B007	375Fox Avenue
SG5	David	manager	24000	B003	163Main Street

The functional dependency staff# → position clearly holds on this relation instance. However, the reverse functional dependency position → staff# clearly does not hold.

The relationship between staff# and position is 1:1 – for each staff member there is only one position. On the other hand, the relationship between position and staff# is 1:M – there are several staff numbers associated with a given position.



For the purposes of normalization we are interested in identifying functional dependencies between attributes of a relation that have a 1:1 relationship.

When identifying Fds between attributes in a relation it is important to distinguish clearly between the values held by an attribute at a given point in time and the *set of all possible values* that an attributes may hold at different times.

In other words, ***a functional dependency is a property of a relational schema (its intension) and not a property of a particular instance of the schema (extension).***

The reason that we need to identify Fds that hold for all possible values for attributes of a relation is that these represent the types of integrity constraints that we need to identify. Such constraints indicate the limitations on the values that a relation can legitimately assume. In other words, they identify the legal instances which are possible.

Let's identify the functional dependencies that hold using the relation schema STAFFBRANCH

In order to identify the time invariant Fds, we need to clearly understand the semantics of the various attributes in each of the relation schemas in question.

For example, if we know that a staff member's position and the branch at which they are located determines their salary. There is no way of knowing this constraint unless you are familiar with the enterprise, but this is what the requirements analysis phase and the conceptual design phase are all about!

staff# $\rightarrow$ sname, position, salary, branch#, baddress

branch# $\rightarrow$ baddress

baddress $\rightarrow$ branch#

branch#, position $\rightarrow$ salary

baddress, position $\rightarrow$ salary

Trivial Functional Dependencies

As well as identifying Fds which hold for all possible values of the attributes involved in the fd, we also want to ignore trivial functional dependencies. A functional dependency is trivial if, the consequent is a subset of the determinant. In other words, it is impossible for it *not* to be satisfied.

Example: Using the relation instances on page 6, the trivial dependencies include:

{ staff#, sname} $\rightarrow$ sname

{ staff#, sname} $\rightarrow$ staff#

Although trivial Fds are valid, they offer no additional information about integrity constraints for the relation. As far as normalization is concerned, trivial Fds are ignored.

Inference Rules for Functional Dependencies

We'll denote as F, the set of functional dependencies that are specified on a relational schema R. Typically, the schema designer specifies the Fds that are *semantically obvious*; usually however, numerous other Fds hold in all legal relation instances that satisfy the dependencies in F.
These additional Fds that hold are those Fds which can be *inferred* or *deduced* from the Fds in F.
The set of all functional dependencies implied by a set of functional dependencies F is called the closure of F and is denoted F+.
The notation: F □ X → Y denotes that the functional dependency X → Y is implied by the set of Fds F.
Formally, F+ ≡ {X → Y | F □ X → Y}
A set of inference rules is required to infer the set of Fds in F+.

For example, if I tell you that Kristi is older than Debi and that Debi is older than Traci, you are able to infer that Kristi is older than Traci. How did you make this inference? Without thinking about it or maybe knowing about it, you utilized a transitivity rule to allow you to make this inference. The set of all Fds that are implied by a given set S of Fds is called the closure of S, written S+. Clearly we need an algorithm that will allow us to compute S+ from S. You know the first attack on this problem appeared in a paper by Armstrong which gives a set of inference rules. The following are the six well-known inference rules that apply to functional dependencies.

IR1: reflexive rule – if $X \supseteq Y$, then $X \rightarrow Y$

IR2: augmentation rule – if $X \rightarrow Y$, then $XZ \rightarrow YZ$

IR3: transitive rule – if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

IR4: projection rule – if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$

IR5: additive rule – if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$

IR6: pseudo transitive rule – if $X \rightarrow Y$ and $YZ \rightarrow W$, then $XZ \rightarrow W$

The first three of these rules (IR1-IR3) are known as Armstrong's Axioms and constitute a necessary and sufficient set of inference rules for generating the closure of a set of functional dependencies.

These rules can be stated in a variety of equivalent ways. Each of these rules can be directly proved from the definition of functional dependency. Moreover the rules are complete, in the sense that, given a set S of Fds, all Fds implied by S can be derived from S using the rules. The other rules are derived from these three rules.

Given R = (A,B,C,D,E,F,G,H, I, J) and

F = {AB → E, AG → J, BE → I, E → G, GI → H}

Does F □ AB → GH?

Proof

1. AB → E, given in F
2. AB → AB, reflexive rule IR1
3. AB → B, projective rule IR4 from step 2
4. AB → BE, additive rule IR5 from steps 1 and 3
5. BE → I, given in F
6. AB → I, transitive rule IR3 from steps 4 and 5
7. E → G, given in F
8. AB → G, transitive rule IR3 from steps 1 and 7
9. AB → GI, additive rule IR5 from steps 6 and 8
10. GI → H, given in F
11. AB → H, transitive rule IR3 from steps 9 and 10
12. AB → GH, additive rule IR5 from steps 8 and 11 - proven

Irreducible sets of dependencies

Let S1 and S2 be two sets of Fds, if every FD implied by S1 is implied by S2- i.e.; if S1+ is a subset of S2+-we say that S2 is a cover for S1+(Cover here means equivalent set). What this means that if the DBMS enforces the Fds in S2, then it will automatically be enforcing the Fds in S1.

Next

Next if S2 is a cover for S1 and S1 is a cover for S2- i.e.; if S1+=S2+ -we say that S1 and S2 are equivalent, clearly, if s1 and S2 are equivalent, then if the DBMS enforces the Fds in S2 it will automatically be enforcing the Fds in S1, And vice versa.

Now we define a set of Fds to be irreducible(Usually called minimal in the literature) if and only if it satisfies the following three properties

1. The right hand side (the dependent) of every Fds in S involves just one attribute (that is, it is singleton set)
2. The left hand side (determinant) of every in S is irreducible in turn-meaning that no attribute can be discarded from the determinant without changing the closure S+(that is, with out converting S into some set not equivalent to S). We will say that such an Fd is ***left irreducible.***
3. No Fd in S can be discarded from S without changing the closure S+(That is, without converting s into some set not equivalent to S)

Now we will work out the things in detail.

Relation R {A,B,C,D,E,F} satisfies the following Fds

AB → C

C → A

BC → D

ACD → B

BE → C

CE → FA

CF → VD

D → EF

Find an irreducible equivalent for this set of Fds?

Puzzled! The solution is simple. Let us find the solution for the above.

1. AB → C
2. C → A
3. BC → D
4. ACD → B
5. BE → C
6. CE → A
7. CE → F
8. CF → B

9. CF → D
10. D → E
11. D → F

Now:

- 2 implies 6, so we can drop 6
- 8 implies CF → BC (By augmentation), by which 3 implies CF → D (By Transitivity), so we can drop 10.
- 8 implies ACF → AB (By augmentation), and 11 implies ACD → ACF (By augmentation), and so ACD → AB (By Transitivity), and so ACD → B(By Decomposition), so we can drop 4

No further reductions are possible, and so we are left with the following irreducible set:

AB → C

C → A

BC → D

BE → C

CE → F

CF → B

D → E

D → F

Alternatively:

- 2 implies CD → ACD (By Composition), which with 4 implies CD → BE (By Transitivity), so we can replace 4 CD → B
- 2 implies 6, so we can drop 6(as before)
- 2 and 10 implies CF → AD (By composition), which implies CF → ADC (By Augmentation), which with (the original) 4 implies CF → B(By Transitivity), So we can drop 8.

No further reductions are possible, and so we are left with following irreducible set:

AB → C

C → A

BC → D

CD → B

BE → C

CE → F

CF → D

D → E

D → F

Observe, therefore, that there are two distinct irreducible equivalence for the original set of Fds.

Review Questions

1. Define functional dependencies?
2. Explain the inference rules?
3. What does it mean to say that Armstrong's inference rules are sound? Complete?
4. Prove the reflexity, augmentation, transitivity rules, assuming only the basic definition of functional dependence?
5. List all the Fds satisfied by the STAFFBRANCH relation?

Activities

Find out the relation of Darwen's "General Unification Theorem" with the inference rules

Relation R{A,B,C,D,E,F,G,H,I,J) satisfies the following Fds:

ABD → E

AB → G

B → F

C → J

CJ → I

G → H

Is this an irreducible set? What are the candidate keys?

References

1. Date, C.J., Introduction to Database Systems (7th Edition) Addison Wesley, 2000
2. Elamasri R . and Navathe, S., Fundamentals of Database Systems (3rd Edition), Pearsson Education, 2000.
3. http://www.cs.ucf.edu/courses/cop4710/spr2004