Functional Dependencies and Normalization

1. Functional Dependencies

A functional dependency (FD) for relation R is a formula of the form

A→ B

where A and B are sets of attributes of R. A is called the LHS (left hand side) and B is called the RHS (right hand side), and we say the LHS (functionally) determines the RHS, or the RHS is (functionally) dependent on the LHS.

The meaning of the functional dependency is that for every value of A, there is a unique value of B. We say the FD holds for R if, for any instance of R, whenever two tuples agree in value for all attributes of A, they also agree in value for all attributes of B.

Example 1: For the relation Student(studentID, name, DateOfBirth, phoneNumber), assuming each student has only one name, then the following functional dependency holds

{studentID} →{name, DateOfBirth}

However, assuming a student may have multiple phone numbers, then the FD

{studentID} → {phoneNumber}

does not hold for the table.

By convention, we often omit the curly braces { } for the set, and write the first functional dependency in Example 1 as

studentID→name, DateOfBirth.

Note that the above FD can also be written equivalently into the two FDs below:

studentID → name

studentID→DateOfBirth

Trivial FD: A trivial FD is one where the RHS is a subset of the LHS. That is, every attribute on the RHS is also on the LHS. For example, the two FDs below are trivial:

name →name,

studentID, name → name

Trivial FDs do not provide real restrictions on the data in the relation, and they are usually ignored.

Functional Dependencies and Normalization

Inference rules for functional dependencies

A set of FDs may logically imply some other FDs. For instance, a→b, and b→c together imply a→c. Similarly, a→b and c →d together imply a,c→b,d.

Given a set F of FDs, we can find all FDs implied by those in F using the following three inference rules.

(1) a→a for every attribute a
(2) If a→b, and b→c, then a→c
(3) If a→b and c→d, then a,c→b,d

Given two sets of FDs, E and F, we say E and F are equivalent, if every FD in E is implied by F, and vice versa.

Closure set of a set of attributes w. r. t a set of FDs

Given a set F of FDs, and a set X of attributes, the closure set of X (w.r.t F), denoted X^+, is the set of all attributes that can be inferred to be functionally dependant on X. Surely X is a subset of X^+. For example, if F={ a→b, b→c }, then $\{a\}^+ =\{a,b,c\}$.

To find the closure set X^+, we can use the simple steps below:

1) Initially $X^+=X$
2) For each FD in F, if the LHS is in X^+, add the RHS to X^+
3) Repeat until no change to X^+ is possible.

Example 2: Suppose F contains a→b,

a,b→d,

c→e

X= {a,c}.

Initially $X^+=\{a,c\}$. Using the first FD, we can add b to X^+. Now using the second FD, we can add d to X^+, and using the last FD we can add e to X^+. Thus the final closure set is $X^+=\{a,b,c,d,e\}$.

2. Relationship between FDs and keys in a relation

A set X of attributes in R is a superkey of R if and only if X^+ contains all attributes of R. In other words, X is a superkey if and only if it determines all other attributes.

X is a candidate key if and only if it is a superkey, but none of its proper subset is a superkey.

In other words, X is a candidate key if and only if $X^+=R$, but for any proper subset Y of X, $Y^+ \neq R$.

The above observation provides a way to find all candidate keys of R using the functional (dependencies. We can simply try all subsets of R, and test whether it is a candidate key.

However, the candidate keys can be found more efficiently, using the observations below:

Observation 1: any candidate key must contain attributes that have not appeared on the RHS of any functional dependency.

Observation 2: if an attributed has occurred on the RHS of some FD, but not on the LHS of any FD, then it cannot be in any candidate key.

Given a set F of functional dependencies that hold on R, we can find all candidate keys of R as follows:

1. Find all attributes that have not appeared on the RHS of any FD. Denote this set by A.
2. Denote the set of attributes that appear on the RHS of some FD, but not on the LHS of any FD by B.
3. Compute the closure set A^+, if $A^+=R$, then A is the only candidate key.
4. If $A^+\neq R$, then for each attribute x in R-B, test whether $A \cup \{x\}$ is a candidate key. If not, try to add another attribute in R-B to A and test whether it is candidate key.
5. Repeat step 4, until all candidate keys have been found.

Example 3.

R =(a,b,c,d,e) F={ a→c; c, d →b}

The set of attribute not occurred on the RHS of any FD is {a, d, e}.

Since the closure set of {a,d,e} contains all attributes of R, it is the only candidate key of R.

Now suppose F={ a→c; c →b,d; d→a}

{e} is the set of attributes that have not appeared on the RHS of any FD. So every candidate key must contain e. b only appeared on RHS, so no candidate key can contain b.

Since $\{e\}^+ = \{e\}$, {e} is not a superkey. We test each of {c,e}, {a,e}, and {d,e} next.

{c,e}+ ={c,e,b,d,a} . Therefore {c,e} is a superkey. {c,e} is also a candidate key since neither {e} nor {c} is a superkey.

{a,e}+ ={a,e,c,b,d}. Similar to the above, {a,e} is a candidate key.

Similarly we can verify {d,e} is a candidate key.

Therefore {c,e},{a,e}, {d,e} are all of the candidate keys.

3. Minimal cover of a set of FDs:

Given a set F of FDs, we say another set E of FDs is a minimal cover of F if

(1) Every FD in E has a single attribute on the RHS.
(2) F and E are equivalent, that is, every FD in E can be inferred from the FDs in F, and every FD in F can be inferred from the FDs in E.
(3) Every FD A→ b in E is minimal in its LHS, that is, there is no proper subset C of A such that C→b .
(4) There is no redundant FD in E. That is, removing any FD from E will result in a set of FD that is not equivalent to F.

Finding a minimal cover E of F

Initially E=F

Step 1: rewrite each FD that has *m* attributes on the RHS into *m* FDs where the RHS is a single attribute.

Step 2: remove trivial FDs.

Step 3: minimize LHS of each FD

For each FD X→y in E, and for each attribute in x in X, if X-{x} → y is implied by E, then replace X→y with X-{x}→y.

Step 4: remove redundant FDs

For each FD in E, if it is implied by other FDs in E, then remove it from E.

Example 4.

Suppose F={ a,b,c → c,d,e,f ; c→e; a→b; d→f }

Step 1 will rewrite the first FD into

Functional Dependencies and Normalization

a,b,c →c

a,b,c → d

a,b,c→e

a,b,c→f

Step 2 will remove a,b,c →c.

Now F ={ a,b,c → d ; a,b,c→e; a,b,c→f; c→e; a→b; d→f }

Step 3 will change F to

F= {a,c → d; c→ e; a,c→f ; c→e; a→b; d→f}

Step 4 will remove a,c→f (since it is implied by a,c→d and d→f), and one of the c→e, and change F to the minimal cover

F= {a,c → d; c→ e; a→b; d→f}

4. Normal Forms

Boyce-Codd Normal Form (BCNF)

A table R is in BCNF if for every non-trivial FD A→b, A is a superkey.

3rd Normal Form (3NF)

A table R is in 3NF if for every non-trivial FD A→b, either A is a superkey or b is a key attribute.

2nd Normal Form (2NF)

A table R is in 2NF if for every non-trivial FD A→b, either A is not a proper subset of any candidate key, or b is a key attribute.

An attribute that has appeared in some candidate key is called a **key attribute**. Attributes that are not key attributes are called **non-key attributes.** If A is a proper subset of some candidate key, we say A→b is a **partial dependency**, and b is **partially dependent** on the candidate key. If A is not a proper subset of some candidate key, and A is not a superkey, we say A→b is a **transitive dependency,** and b is **transitively dependent** on the candidate keys**.**

With these concepts, 3NF and 2NG can be equivalently defined as follows:

A table R is in 2NF if there are no non-key attributes that are partially dependent on any candidate key.

A table R is in 3NF if it is already in 2NF, and there are no non-key attributes that are transitively dependent on any candidate key.

Clearly, if a table is in BCNF, then it is in 3NF. If a table is in 3NF, then it is in 2NF.

The above definitions also provide a way to test whether a table is in 2NF, 3NF or BCNF. One just needs to check each non-trivial FD to see whether it violates the definition of the normal forms.

5. Normalization

Given a set F of FDs that hold for table R, if R is not in 2NF (or 3NF, BCNF), we can decompose R into smaller tables so that each of the smaller tables are in 2NF (or 3NF, BCNF). This process is called normalization.

The approach is: for each FD A→b that violates the definition of the normal form, we decompose R into R1 = (A, b), and R2=(R-{b}). This process is repeated until all tables are in the normal form.

Example 4

R=(a,b,c,d). F contains a→c and a, b→ d.

The only candidate key is {a,b}. a→c is a partial dependency and c is a non-key attribute. Therefore, a→c violates the definition of 2NF. We can decompose R into (a,c) and (a,b,d), which are both in 2NF. Actually both are in BCNF.

Caution: when decomposing a table using the FDs, we need to make sure the FD is such that the RHS attribute does not appear on the LHS (otherwise it is trivial), and the LHS is minimal, that is, removing at attribute from the LHS the FD will no longer hold.

Lossless and FD-preserving decomposition

Suppose R is decomposed into smaller tables R1,R2,…,Rn. We say that the decomposition is **lossless** if for any instance r of R, r is equivalent to the natural join of its projections onto R1, R2, …., Rn.

If both A and B are attributes in R*i* , and A→B is implied by F, then we A→B is a FD that hold on R*i.*

We say the decomposition is **FD-preserving** if the FDs that hold on R1, R2,…, Rn altogether is equivalent to F.

It is known that any table can be decomposed into 3NF tables with lossless-join property and FD-preserving property, but we may not be able to decompose a table into BCNF with both of these properties.

An algorithm for decomposing a table into 3NF with both properties is as follows:

1. Find the minimal cover of F. Suppose it is E.

2. For each FD A→b in E, if b is a non-key attribute, have a table (A,b), where A is the primary key.
3. (optional but good) Merge tables that have identical primary key.
4. Have a table R-B, where B is the set of non-key attributes that appeared on the RHS of some FD in E.

Example 5

Suppose R= (a, b, c, d, e, f)
The FD set F contains a,b →c,d,e and d→a.
The minimal cover of the functional dependencies is:
a,b-->c
a,b-->d
a,b-->e
d-->a

The candidate keys are
b,f,a
b,f,d

Thus c, e are non-key attributes.

The table is NOT in 2NF because there are partial dependencies .e.g., a.b→c

A FD-preserving decomposition to 3NF works as follows:

Using step 2, we will have two tables (a,b,c) and (a,b,e), where {a,b} is the primary key.
Using step 3, we merge the above table into (a,b,c,e).
Using step 4, we get the table (a,b,d,f).

Notes about the Normalization Tool

1. There may be multiple minimal covers of a set of FDs (For instance, {a→b; b→c; c→a} and {a→c; c→b; b→a} are both minimal covers of {a→b,c; b→c,a; c→a,b}). The minimal cover obtained by this tool is one of them.
2. There may be multiple correct ways for a table to be decomposed into 3NF or BCNF tables. Therefore, if your own decomposition is different from what is obtained by this tool, it does not mean your decomposition is incorrect.
3. The 3NF decomposition obtained by this tool is a dependency-preserving decomposition.