Functional Dependencies

$A_1 \ldots A_m \rightarrow B_1 \ldots B_n$

"$A_1, \ldots, A_m$ functionally determine $B_1, \ldots, B_n$"

gearhead(person_ID, main_address, make, model, year, color, income, parking_space, age, insurance_policy, tag_number)

1

Keys

Assume a relation with schema $R(A_1, \ldots, A_n)$.

- A subset of $\{A_1, \ldots, A_n\}$ is a superkey of R if it functionally determines all the attributes of R.
- A superkey is a key if no proper subset of it is a superkey.

2

Armstrong's Axioms

- Reflexivity – If $\{B_1, \ldots, B_n\} \subseteq \{A_1, \ldots, A_m\}$ then $A_1 \ldots A_m \rightarrow B_1 \ldots B_n$. (These are the trivial functional dependencies.)
- Augmentation – If $A_1 \ldots A_m \rightarrow B_1 \ldots B_n$, then $A_1 \ldots A_m C_1 \ldots C_k \rightarrow B_1 \ldots B_n C_1 \ldots C_k$.
- Transitivity – If $A_1 \ldots A_m \rightarrow B_1 \ldots B_n$ and $B_1 \ldots B_n \rightarrow C_1 \ldots C_k$, then $A_1 \ldots A_m \rightarrow C_1 \ldots C_k$.
- These are sufficient to infer the consequences of a set of functional dependencies.

3

Splitting/Combining

A useful consequence of Armstrong's axioms is the splitting/combining rule.

$A_1 \ldots A_m \rightarrow B_1 \ldots B_n$

if and only if

for every i from 1 to n, $A_1 \ldots A_m \rightarrow B_i$.

(See Exercise 3.2.2 for some other potentially useful consequences.)

4

Closure of Attributes

Suppose that S is a set of functional dependencies and that $\{A_1, \ldots, A_m\}$ is a set of attributes. The closure of $\{A_1, \ldots, A_m\}$ (with respect to S) is the set of attributes B such that $A_1 \ldots A_m \rightarrow B$ is a consequence of the functional dependencies in S.

The closure is denoted by $\{A_1, \ldots, A_m\}^+$.

5

Computing the Closure

To compute the closure of $\{A_1, \ldots, A_m\}$ with respect to S:

Let $X = \{A_1, \ldots, A_m\}$.

Repeat:

If $B_1 \ldots B_k \rightarrow C_1 \ldots C_n$ is in S and $\{B_1, \ldots, B_k\}$ is a subset of X, then for each i from 1 to n, C_i is added to X if it is not already in X.

Until X cannot be made any larger.

6

Minimal Bases

It is often useful to work with a minimal basis for a set of functional dependencies and their consequences.

A minimal basis for a set S of functional dependencies and their consequences is a set B such that

- The FDs in B all have singleton right sides.
- If any FD is removed from B, the consequences of those that are left do not equal the consequences of S.
- If any attribute is removed from the left side of any FD in B, the consequences of the altered B do not equal the consequences of S.

Computing a minimal basis

To compute a minimal basis from a set S of FDs:

Set S' to the FDs from S split so that all right sides are singletons.

Repeat:

If a FD can be inferred from the other FDs in S', remove it.

If FD1 = $A_1 \ldots A_m \rightarrow B$ and FD2 is the same with A_i removed for some i and FD2 can be inferred from S', remove FD1 from S' and add FD2.

Until S' cannot be changed anymore.

Projecting a set of FDs

Given R($A_1, \ldots, A_n$) and set S of FDs. Suppose that R_1 is the projection π_L(R). What functional dependencies hold on R_1?

Let T be empty.

For each subset X of the attributes of R_1, compute X^+ with respect to S and add $X \rightarrow A$ to T for each attribute A in X^+ that is also an attribute of R_1.

It is recommended that a minimal basis be computed for T.

Note that the proof that a FD belongs to T may refer to attributes of R that are not attributes of R_1.

Anomalies

The main source of maintenance problems is requiring that too much information be stored in each row of a table, usually because each row describes the properties of more than one object.

Redundancy – The same information about one object is stored in more than one row (because the object is related to more than one other object).

Update anomaly- When information about one object is updated in one row, it may not be updated in all the other rows where it exists.

Anomalies(cont.)

Insertion anomaly – When we input information about one object, we are forced to input information about another object as well.

Deletion anomaly – When delete all information about one object, we may delete information about another object as well. (The information about the second object only appears in rows that contain the information about the first object.)

[gearhead]

Normalization

Anomalies are reduced by decomposing a table into two or more tables so that certain conditions are satisfied. A table that satisfies those conditions are said to be in normal form.

Commonly used normal forms are:

Boyce-Codd Normal Form (BCNF)

Third Normal Form

Fourth Normal Form

(Others exist: Fifth Normal Form, Domain Key Normal Form, Project-Join Normal Form.)

First & Second Normal Forms

A relation is in First Normal Form if its attribute domains contain only atomic values (the values have no internal structure of interest to us).

(It is easy to set up a database so that all relations are in First Normal Form from the beginning.)

A relation is in Second Normal Form if each attribute is either in a key for the relation or is functionally determined only by one or more keys for the relation. (No attribute not in a key is functionally determined by a proper subset of a key.)

(More advanced normal forms will imply that a relation is also in Second Normal Form.)

Decomposition

If a relation is not in a desired advanced normal form, it is decomposed into two or more relations, each of which is in the desired advanced normal form.

A set of relations $\{R_1, ..., R_n\}$ is a decomposition of a relation R if each R_i is a projection from R and the union of the attributes for all the relations in the set equals the set of attributes for R.

[gearhead]

Boyce-Codd Normal Form

A relation R is in Boyce-Codd Normal Form if the left side of every nontrivial functional dependency for R is a superkey for R.

This avoids redundancy: suppose AB ➔ C, B ➔ D. The relation R(A,B,C,D) is not in Boyce-Codd Normal Form.

A	B	C	D
1	1	2	4
1	2	3	6
2	1	1	4
2	2	4	6

de-com-pose:

A	B	C
1	1	2
1	2	3
2	1	1
2	2	4

B	D
1	4
2	6

Computing BCNF

Given a relation $R(A_1, . . ., A_n)$ and set S of its functional dependencies.

Test the FDs in S until one is found, say X ➔ Y, where X is not a superkey for R. (If none found, stop.)

Replace R and S with $R_1(X^+)$ and the projection of S down to R_1, and $R_2(((A_1, . . ., A_n) - (X^+ - X))$ and the projection of S down to R_2.

Repeat the above on R_1 and R_2 with their sets of functional dependencies.

Testing X ➔ Y

To test whether X is a superkey for R, compute X^+. If X^+ does not contain all the attributes of R, X is not a superkey and the FD X ➔ Y violates the conditions for BCNF.

[gearhead]

Lossless Joins

The decomposition produced by the previous algorithm has the property that it has a lossless join, that is, the natural join of the relations in the decomposition reproduces the original relation. (No information is lost.)

Example where join is not lossless:

A	B	C
1	2	3
4	2	6

A	B
1	2
4	2

B	C
2	3
2	6

A	B	C
1	2	3
1	2	6
4	2	3
4	2	6

Chase Test

Given R(L) decomposed into $R_i(L_i)$, i = 1,...,k. For each i, make a tuple t_i for R such that for each attribute A in L_i, the value of A in t_i is *a*, and the other values in t_i are distinct variables.

Use the FDs for R to infer what values the variables have to be. If the tuple <*a*,...,*a*> can be derived, the decomposition has the lossless join property. If it can't be derived, the variables can be given non-*a* values so that the original t_is form an instance of R that is not reproduced by the natural join of the R_is.

Chase Test Example

Given R(A,B,C,D), FDs A → B, B → C, CD → A. R is decomposed into R_1(A,D), R_2(A,C), R_3(B,C,D).

t_1 = <*a*, X1, X2, *a*>

t_2 = <*a*, X3, *a*, X4>

t_3 = <X5, *a*, *a*, *a*>

Tuple <*a*, *a*, *a*, *a*> can be deduced.

FDs Can Be Lost

Given R(A,B,C) with FDs AB → C, C → A. Decompose R into Boyce-Codd Normal Form relations.

R_1(C, A) with FD C → A

R_2(B,C) with no FDs

FD AB → C has been lost.

Third Normal Form

A relation R is in Third Normal Form if the left side of every nontrivial functional dependency for R is a superkey for R or if every attribute in the right side of the functional dependency that is not in the left side is in a key for R.

Example on previous slide is in 3NF but not in BCNF.

A decomposition into 3NF relations is dependency preserving but need not have the lossless join property.

However, there are algorithms that do produce decompositions into 3NF relations that have the lossless join property.

3NF Synthesis Algorithm

Given relation R and a minimal basis S of the FDs for R.

For each FD X → A in S make a relation with XA as the attributes of the relation.

If none of the attribute lists for the new relations is a superkey for R, make another relation whose attribute list is a key for R.

In practice, all the relations made for FDs having the same left side are combined into one relation.

The resulting set of new relations is a decomposition of R that has the lossless join property and that preserves functional dependencies and the new relations are in 3NF.

Multivalued Dependencies

Sometimes a set of attributes determines a set of values for other attributes instead of a single value for other attributes.

Course	Instructor	Textbook
101	Smith	book1
101	Smith	book2
101	Jones	book1
101	Jones	book2
102	Smith	book3
102	Avery	book3

MVD Definition

Given R(XYZ) where XY and Z are disjoint lists of attributes and X and Y are non-empty. The multivalued dependency (MVD)

$$X \twoheadrightarrow Y$$

holds if whenever tuples xy_1z_1 and xy_2z_2 are in R, the tuple xy_1z_2 is also in R.

(The attributes of R can occur in any order in the schema for R. They don't have to group as shown here, with the X attributes appearing first, followed by the Y attributes and then the Z attributes. Also, X and Y are allowed to overlap.)

Basic MVD Facts

Trivial MVD – $X \twoheadrightarrow Y$ if $Y \subseteq X$.

Transitivity – If $X \twoheadrightarrow Y$ and $Y \twoheadrightarrow Z$, then $X \twoheadrightarrow Z$.

Note: the splitting rule does not apply to MVDs.

FB promotion – If $X \rightarrow Y$ then $X \twoheadrightarrow Y$.

Complementation – If XYZ encompasses all the attributes of the relation (as in the definition of MVD) and $X \twoheadrightarrow Y$, then $X \twoheadrightarrow Z$.

Another trivial MVD – If XYZ encompasses all the attributes of the relation (as in the definition of MVD) and Z is empty, $X \twoheadrightarrow Y$.

Fourth Normal Form

- A relation R is in Fourth Normal Form if for every non-trivial MVD $X \twoheadrightarrow Y$ for R, X is a superkey for R.

A relation is decomposed in to a set of Fourth Normal Form relations by the same algorithm that decomposes it into a set of BCNF relations except that MVDs are used instead of FDs.

Relations Between Normal Forms

- 4NF $\subseteq$ BCNF $\subseteq$ 3NF

FDs and MVDs are generally not preserved by decomposition into 4NF. Only decomposition into 3NF is guaranteed to preserve FDs, but some redundancy may remain.

Chasing FDs and MVDs

Given a set S of FDs and/or MVDs for a relation R(XYZ) where X and Y are nonempty, here is a general approach to deciding whether $X \rightarrow Y$ or $X \twoheadrightarrow Y$ holds.

(As before, the actual order of the attributes for R can be in any order, not just the order suggested by XYZ, and X and Y can overlap.)

Does $X \rightarrow Y$ Hold?

Make two tuples such all the X attributes have the value of *a* and all the attributes in YZ that are not in X are distinct variables. The procedure is to apply the FDs and MVDs in S to prove that for each attribute in Y, the variables or constant assigned to that attribute in the two tuples must be the same. If this goal can't be reached, a counter-example can be made by assigning distinct constants to the variables in the two tuples, subject to the equality constraints that were produced by the inference procedure.

Does X $\rightarrow\rightarrow$ Y Hold?

Make two tuples by assigning *a* to all the attributes in X and Y in one tuple and to all the attributes in X and Z in the other tuple. Assign distinct variables to all the other attributes in the two tuples. The procedure is to apply the the FDs and MVDs in S to see whether the tuple consisting of all *a*'s can be produced. If it can, X $\rightarrow\rightarrow$ Y holds. If it can't, the set of tuples that has been produced is a counter-example when all the variables are replaced by distinct new constants in a way that is consistent with any equality constraints that were produced by the inference procedure.

Examples

Given R(A,B,C,D,E) and dependencies A $\rightarrow\rightarrow$ BC, CD $\rightarrow$ E, B $\rightarrow$ D, E $\rightarrow$ A.

A $\rightarrow$ E?

A $\rightarrow\rightarrow$ E?

CD $\rightarrow$ A?

CD $\rightarrow\rightarrow$ B?

CD $\rightarrow$ B? [The last two are false]

B $\rightarrow\rightarrow$ E? Project onto {B,C,D} (see next slide)

Projecting Dependencies

Given R(L) and set S of FDs and MVDs. Let R'(L') be such that L' $\subset$ L. We want to project S down to S', the set of FDs and MVDs for R'. The FDs can be projected as before, but using the FD test on slide 30. To find the MVDs for S', consider disjoint subsets X and Y of L' and test to see if X $\rightarrow\rightarrow$ Y by Chase test. It is only necessary to derive a tuple with *a* values for all the attributes that are in L', not for all the attributes in L as before.