Functional Dependencies

Anne Denton

Department of Computer Science
North Dakota State University

Outline

1. Informal guidelines for relational schemas
   - 1. Semantics and 2. Anomalies
   - 3. Null values and 4. Spurious tuples
2. Functional Dependencies (FD)
   - Definition
   - Inference rules

Table of Contents

Informal guidelines for relational schemas

- In practice, people are often asked to work with existing databases
- The databases may have evolved over time and no longer satisfy design expectations
- Some potential problems can be identified using informal guidelines

Guideline 1: Clear semantics of attributes

- Make sure that you can easily explain the meaning of attributes.
  - Example: If you create a table of orders that has an order id, and then a customer id and name of the customer who placed the order
    ... that is not very clear
- Normalization principles provide formal arguments against this
- Unclear semantics can signify problems that may not even be eliminated through normalization

Guideline 2: Prevent anomalies

- Strange behavior or anomalies can indicated problems
  - Example: Take the below example of orders, and id and last name of the customer who placed it (primary key: oid)

Order

order_id	customer_id	customer_name
55	11	Smith
56	11	Smith
58	3	Miller
61	7	Smith

Brainstorming question

Think about what can go wrong with the below table

Order

order_id	customer_id	customer_name
55	11	Smith
56	11	Smith
58	3	Miller
61	7	Smith

Anomalies

Insertion anomaly 1: If you insert ⟨'63','11','Smiht'⟩ the name is clearly inconsistent.

Insertion anomaly 2: We would not be able to insert customer id - name information of a customer who has not order listed in the table
Because order_id is primary key ⟨null,17,'Butcher'⟩ cannot be inserted

Deletion anomaly: Consider what happens to the customer information if, e.g. their only order is cancelled
We loose more information than we mean to.

Modification anomaly: Consider what happens if the customer changes his/her name?
All tuples have to be updated.

Anomalies continued

- Try to make sure that no update anomalies can happen
- Applying normalization principle allows preventing most of them
- If anomalies cannot be prevented, note them clearly and enforce constraints through programs operating on the database

Table of Contents

Guideline 3 : Avoid null values in attributes

- Problems with null
  - Waste of storage space
  - Different possible meanings (not applicable, non-existent, etc.)
  - Problems with aggregate functions
  - Problems with join (inner join <=> outer join)

Guideline 4: Prevent spurious tuples

- Imagine splitting the table Order into two tables, where customer_name in both tables is again the last name of the customer:

Order_Name

order_id	customer_name
55	Smith
56	Smith
58	Miller
61	Smith

Customer

customer_id	customer_name
3	Miller
7	Smith
11	Smith

- Consider what happens upon joining both tables

Spurious tuples

- Joining these two tables on customer_name gives:

order_id	customer_name	customer_id
55	Smith	7
55	Smith	11
56	Smith	7
56	Smith	11
58	Miller	3
61	Smith	7
61	Smith	11

- The three red tuples where not in the original relation, and don't represent correct information
- They are called spurious tuples

Reasons for spurious tuples

- Join attribute should be a key to one of the relations
- Ideally join on foreign key-primary key combinations
- The importance of selecting a join attribute that is a key to at least one of the relations will also follow from normalization considerations

Table of Contents

Formal approaches

- Most formal methods to improve database design are based on the concept of functional dependencies
- More general concept of multi-valued dependencies has functional dependencies as a special case
- Formal methods start by considering database as a single relation, called the "universal relation"
- It is always possible to find a universal relation that contains all information
- The universal relation is a theoretical concept, it's not useful as a practical representation

Definition of a functional dependency

- Functional dependencies are defined between sets of attributes X and Y
- Y is functionally dependent on X if knowing X leaves no ambiguity as to what the value of Y is
- The notation is: X $\rightarrow$ Y
- Formally: For any two tuples t_1 and t_2, if the value of X is the same for both, $t_1[X] = t_2[X]$, it follows that the value of Y is the same, $t_1[Y] = t_2[Y]$

Question 1 (Multiple answers can be correct)

Which of the following statements are correct about a possible functional dependency $X \rightarrow Y$ with X and Y being sets of attributes of relation R

1. It can be inferred that $Y \rightarrow X$
2. If the value of X is known there is no ambiguity left as to what value Y has
3. In a relation that only has attributes X and Y, X would be a superkey
4. In a relation that only has attributes X and Y, Y would be a superkey

Some conclusions

- By the definition of a key, any attribute in a relation (Y) is always functionally dependent on any candidate key of the relation (X)
- A key of a relation is not dependent on an attribute that is not a key
- If you were to project a relation to $X \cup Y$, with duplicates eliminated, and $X \rightarrow Y$ then X would be a superkey of the relation

Question 2 (Multiple answers can be correct)

Which of the following statements are correct about a functional dependency $X \rightarrow Y$ with X and Y being sets of attributes of relation R

1. If it is ensured that the values of X in the relation will be unique, X will functionally determine Y
2. If X only has a single value across all tuples in a database instance, it cannot functionally determine Y unless Y also only has a single value
3. If attribute X does not functionally determine Y, then X together with Z will not functionally determine Y either
4. If X functionally determines Y, then a subset of X will also functionally determine Y

Notes

- Functional dependencies have to hold for all legal database states
- You cannot conclude on the existence of a functional dependency by looking at one database state
- If you are sure that a database state is legal, you can conclude on the absence of some functional dependencies
- To be sure that a functional dependency holds, you have to argue within the miniworld requirements

Question 3 (Multiple answers can be correct)

Which of the following functional dependencies are expected to hold at NDSU?

1. NDSU email address → Student first and last name
2. Student first and last name → NDSU email address
3. NDSU email address → 7-digit number on NDSU ID card
4. Dept name and course number (e.g., CSci 765) → Final Exam date

Question 4 (Multiple answers can be correct)

1. A legal database instance can unambiguously establish that a functional dependency holds
2. A legal database instance can unambiguously establish that a functional dependency does not hold

Example Schema

- **Order**

order_id	customer_id	customer_name

- Functional dependencies that have to hold for this table
  - order_id $\rightarrow$ customer_id
  - customer_id $\rightarrow$ customer_name
- Further functional dependencies
  - order_id $\rightarrow$ customer_name
  - order_id customer_id $\rightarrow$ customer_name
  - order_id $\rightarrow$ customer_id customer_name
  - order_id customer_id $\rightarrow$ customer_id

Trivial functional dependencies

- Some functional dependencies hold regardless of database state
  - An attribute always functionally determines itself
  - An attribute, together with any number of others, also functionally determines itself
- These observations hold trivially, and the corresponding FDs are called “trivial functional dependencies"
- Formally: If $X \supset Y$, then $X \rightarrow Y$

Question 5 (Multiple answers can be correct)

Which of the following statements are correct

1. Whether a trivial functional dependency holds can be established without seeing any database instance
2. Seeing a database instance can allow you to identify that a trivial functional dependency does not hold

Table of Contents

Inference rules

- Typically a database designer only specifies FDs that are obvious from the semantics of the attributes
- If needed, other functional dependencies are derived by inference rules
- Reasons for importance of inference rules
  - Allow testing if two sets of functional dependencies are equivalent by inferring all others and seeing if the result is identical
  - Can be used for formal normalization algorithms

Union and decomposition rules

- Union rule: If X $\rightarrow$ Y and X $\rightarrow$ Z then X $\rightarrow$ YZ
  - Can be shown from definition of a functional dependency
  - Applying it can be used to show that all relations with the same key can be combined
- Decomposition rule: If X $\rightarrow$ YZ then X $\rightarrow$ Y and X $\rightarrow$ Z
  - Separate relations could be created for the primary key together with each of the other attributes
  - Would require many joins
  - Useful for discussion of normalization

Question 6 (Multiple answers can be correct)

1. If X $\rightarrow$ Y and X $\rightarrow$ Z then X $\rightarrow$ Y, Z
2. If X $\rightarrow$ Y, Z then X $\rightarrow$ Y and X $\rightarrow$ Z
3. If X$\rightarrow$ Y and X $\rightarrow$ Z then Y $\rightarrow$ Z
4. If X $\rightarrow$ Z and Y $\rightarrow$ Z then X, Y $\rightarrow$ Z

Transitive rule

- If X $\rightarrow$ Y and Y $\rightarrow$ Z then X $\rightarrow$ Z
- This rule will be important in the normalization process
- Some transitive functional dependencies are not desirable
  - Details will be discussed as part of formal normalization process
  - Problems arise if Y $\rightarrow$ Z holds, but Y is not a key of the relation
  - Note that since X $\rightarrow$ Z in the above scenario, X can be a key of the relation

Question 7 (Multiple answers can be correct)

If $X \rightarrow Y$ and $Y \rightarrow Z$ then

1. $X \rightarrow Z$
2. $X, Y \rightarrow Z$
3. $X \rightarrow Y, Z$
4. $Z \rightarrow X$

Armstrong's Inference Rules

Set of inference rules that can be used to deduce all others:

Reflexive rule (IR 1): If X $\supset$Y, then X $\rightarrow$ Y

Augmentation rule (IR 2): If X$\rightarrow$ Y, then XZ$\rightarrow$ YZ

Transitive rule (IR 3): If X $\rightarrow$ Y and Y $\rightarrow$ Z then X $\rightarrow$ Z

Challenge Question

Use Armstrong's Inference Rules to prove the Union Rule:
If X $\rightarrow$ Y and X $\rightarrow$ Z then X $\rightarrow$ YZ

Question 8

Consider the following legal instance of a database. Which of the following functional dependencies **can** hold? Which of the following functional dependencies **is guaranteed to** hold?

A	B	C	D	E
1	'gh'	4	'xy'	35
1	'ij'	7	'xy'	76
1	'kl'	11	'xy'	92
4	'mn'	17	'xy'	92

$B \rightarrow A$
$A \rightarrow B$
$A \rightarrow D$
$C \rightarrow A\ B$
$A\ C \rightarrow A$
$A\ E \rightarrow C$