Database Management Systems

Functional Dependencies

M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering
www.memregursoy.com

Good vs Bad Schema

- Our "good" schema:
  - Employee(Ssn, Ename, Bdate, Address, Dnumber)
  - Project(Pnumber, Pname, Plocation, Dnum)
  - Works_On(Ssn, Pnumber, Hours)
- Consider the following "bad" schema:

Redundancy Redundancy

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation
123456789	1	32.5	Smith, John B.	ProductX	Bellaire
123456789	2	7.5	Smith, John B.	ProductY	Sugarland
666884444	3	40.0	Narayan, Ramesh K.	ProductZ	Houston
453453453	1	20.0	English, Joyce A.	ProductX	Bellaire
453453453	2	20.0	English, Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong, Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong, Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong, Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong, Franklin T.	Reorganization	Houston

Anomalies

- **Insertion anomaly**
  - Can't add a project without an employee working on it
  - Can't add an employee unless they're working on at least one project
- **Deletion anomaly**
  - If I delete the project "ProductZ", then I lose the information of "Narayan, Ramesh K."
- **Update anomaly**
  - If a project's name or location is changed, this change must be applied to every employee working on the project, otherwise database becomes logically inconsistent

Find Anomalies

- Consider the following schema – is it good or bad?
- Can you find examples of anomalies?
- What would be a good design?

StudentNum	CourseNum	Student Name	Student City	Course
S21	9201	Jones	Edinburgh	Accounting
S21	9267	Jones	Edinburgh	Physics
S24	9267	Smith	Glasgow	Physics
S30	9201	Richards	Manchester	Accounting
S30	9322	Richards	Manchester	Maths

Schema Refinement

- **Schema refinement** is the process of refining a schema, i.e., turning a bad schema into a good one.

- Existence of anomalies means a schema is bad, and thus the schema should be refined.

- **Functional dependencies (FDs)** enable us to formally reason about anomalies and understand what causes the anomalies in a given schema.

- After we identify FDs, we can perform schema refinement through **normalization** and **decomposition**.

Functional Dependencies

- A **functional dependency X -> Y** holds over relation R if, for every allowable state of R, whenever two tuples have the same value for **X**, they must have the same value for **Y**
  - For tuples t1, t2: If t1[X] = t2[X], then t1[Y] = t2[Y]
  - If they agree on **X**, they must agree on **Y**
  - "**X** functionally determines **Y**"
  - **X** and **Y** can be single attributes or sets of attributes

X	Y	Z
1	a	p
2	b	q
1	a	r
2	b	p

X	Y	Z
1	a	p
2	b	q
1	a	r
2	c	p

X	Y	Z
1	a	p
2	b	q
1	a	r
3	b	p

Does X -> Y hold in any of these relations?

Functional Dependencies

- An FD is a property of the attributes in the schema
  - Must hold in every state (instance) of the relation
- Given a relation state, we can conclude that an FD may hold, but we cannot say it holds for certain.
- However, given a relation state, it is possible to certainly conclude which FDs do not hold.
  - Seeing one violation of that FD is enough

Which FDs may exist in this relation?

Which FDs do not hold for certain in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Anomalies and FDs

- Which FDs seem to hold in this relation?

StudentNum	CourseNum	Student Name	Address	Course
S21	9201	Jones	Edinburgh	Accounting
S21	9267	Jones	Edinburgh	Physics
S24	9267	Smith	Glasgow	Physics
S30	9201	Richards	Manchester	Accounting
S30	9322	Richards	Manchester	Maths

Functional Dependencies

- The LHS or the RHS of an FD does not have to be a single attribute; it can contain multiple attributes.
  - AB -> C
  - X -> YZ
- If an attribute is a **key**, then all other attributes are determined by it
  - If K is a key, then K -> ABCDEF...
  - Equivalently: K -> A, K -> B, K -> C, K -> D, ...
- Notice that the meaning of having multiple attributes on the LHS is different than having multiple attributes on the RHS.
  - You can break up the RHS and get multiple FDs
  - Can you break up the LHS?

Example

- Hourly_Emps(Ssn, Name, Lot, Rating, WageHrly, HrsWorked)
  - We'll use the notation {S,N,L,R,W,H} or SNLRWH
  - Each letter refers to an actual attribute
- Some FDs on Hourly_Emps:
  - Ssn is the key: S -> SNLRWH
  - Rating determines hourly wages: R -> W

S	N	L	R	W	H
			1	100	
			2	200	
			3	250	
			2	300	

What is wrong with this instance (state)?

Inference Rules for FDs

- Given a set of FDs, we can infer additional FDs that hold
  - Let ***F*** be a set of FDs for some relation
  - ***F*** = {A -> B, B -> C, AB -> E, D -> E, ...}

- Armstrong's rules of inference
  - Reflexivity: If Y is a subset of X, then X -> Y
  - Augmentation: If X -> Y, then XZ -> YZ
  - Transitivity: If X -> Y and Y -> Z, then X -> Z

- These form a sound and complete set of inference rules
  - All other inference rules can be derived from these

Inference Rules for FDs

- <u>Decomposition:</u> If X -> YZ, then X -> Y and X -> Z

- Proof:
  - X -> YZ is given
  - We know YZ -> Y from reflexivity
  - Since X -> YZ and YZ -> Y, from transitivity, X -> Y
  - (You can do the same to arrive at X -> Z)

Inference Rules for FDs

- Union: If X -> Y and X -> Z, then X -> YZ

- Proof:
  - From X -> Z, we use augmentation to obtain XX -> XZ
  - XX is the same as X, so: X -> XZ
  - From X -> Y, we use augmentation to obtain XZ -> YZ
  - Take X -> XZ and XZ -> YZ, then use transitivity to obtain X -> YZ

Inference Rules for FDs

- Pseudo-transitivity: If X -> Y and YZ -> W, then XZ -> W

- Proof:
  - From X -> Y, we use augmentation to obtain XZ -> YZ
  - Take XZ -> YZ and YZ -> W, then use transitivity to obtain XZ -> W

Inference Rules for FDs

- <u>Composition:</u> If X -> Y and A -> B, then XA -> YB

- Proof:
  - From X -> Y, use augmentation to obtain XA -> YA
  - From XA -> YA, use decomposition to obtain XA -> Y
  - From A -> B, use augmentation to obtain XA -> XB
  - From XA -> XB, use decomposition to obtain XA -> B
  - Take XA -> Y and XA -> B, use union to obtain XA -> YB

Exercises

- Say we are given a relation R(C,S,J,D,P,Q,V)
  - C is the key, which means: C -> CSJDPQV
  - In addition, we have FDs: JP -> C and SD -> P

- What can we infer?
  - JP -> C, C -> CSJDPQV imply: JP -> CSJDPQV
    - Thus, JP qualifies as a potential key for the relation
  - SD -> P implies: SDJ -> JP
  - SDJ -> JP, JP -> CSJDPQV imply: SDJ -> CSJDPQV
    - Thus, SDJ also qualifies as a potential key for the relation

Exercises

- Say we are given a relation R(A,B,C,G,H,I) with FDs:
  - ***F*** = {A -> B, A -> C, CG -> HI, B -> H}

1) Is A -> H implied by ***F*** ?
2) Is AG -> I implied by ***F*** ?
3) Is AC -> G implied by ***F*** ?
4) Is G -> I implied by ***F*** ?
5) Is AB -> C implied by ***F*** ?

Closure

- We have two types of **closure**:
  - Closure of a set of FDs
  - Closure of one or more attributes **(attribute closure)**
- Let ***F*** denote a set of FDs. The closure of ***F***, denoted $\boldsymbol{F^+}$, is the set of all FDs that can be inferred from ***F***.
  - Size of $\boldsymbol{F^+}$ can be quite large! (exponential in # of attrs)
- Closure of a set of attributes X with respect to ***F***, denoted X^+, is the set of all attributes that are functionally determined by X.
  - Say X = {A, B}, then X^+ could be = {A, B, C, E, ...}.
  - How to build attribute closure X^+?
    - Start with the original set of attributes (X), i.e., $X^+ = X$
    - Keep adding attributes to X^+ as long as given FDs allow inference

Exercises

- Consider the relation CLASS(Classid, Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity).
- The set of FDs ***F*** consists of:
  - Classid -> Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity
  - Course# -> Credit_hrs
  - Course#, Instr_name -> Text, Classroom
  - Text -> Publisher
  - Classroom -> Capacity

1) What is the attribute closure of Classid?
2) What is the attribute closure of Course#?
3) What is the attribute closure of {Course#, Instr_name}?

Exercises

- Consider the set of FDs ***F*** = {A->B, B->C, CD->E}. Does ***F*** imply A->E?
  - Equivalent: Is A->E in the closure F^+?
  - Equivalent: Is E in attribute closure A^+?

- Let's compute A^+:
  - Initialize A^+ = {A}
  - A->B, so add B to A^+: A^+ = {A,B}
  - B->C, so add C to A^+: A^+ = {A,B,C}
  - Can we add any more attributes to A^+ with what we currently have in A^+? No.
  - So, A^+ doesn't contain E, thus A->E doesn't hold.

Minimal Cover

- To find closure, we expanded on a set of given FDs using inference rules.
- Now, let's think in the opposite direction: Given a set of FDs, shrink/reduce it to find the minimal set that is still equivalent to the original set of FDs. **(Min Cover)**
- A set of FDs ***F*** is **minimal** if:
  1. Every FD in ***F*** has a single attribute on its RHS.
  2. We cannot remove any FD from ***F*** and have a set of FDs that is equivalent to ***F***.
  3. We cannot replace any FD X->A in ***F*** with an FD Y->A where Y is a proper subset of X and still have a set of FDs that is equivalent to ***F***.

Min Cover Algorithm

Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

Note: Explanatory comments are given at the end of some of the steps. They follow the format: (*comment*).

Set $F := E$.

1. Replace each functional dependency $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ in F by the n functional dependencies $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$. (*This places the FDs in a canonical form for subsequent testing*)
2. For each functional dependency $X \rightarrow A$ in F

   for each attribute B that is an element of X

   if $\{ \{F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F

   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.

   (*This constitutes removal of an extraneous attribute B contained in the left-hand side X of a functional dependency $X \rightarrow A$ when possible*)
3. For each remaining functional dependency $X \rightarrow A$ in F

   if $\{F - \{X \rightarrow A\} \}$ is equivalent to F,

   then remove $X \rightarrow A$ from F. (*This constitutes removal of a redundant functional dependency $X \rightarrow A$ from F when possible*)

Min Cover Examples

- Let the given set of FDs be: ***E*** = {B->A, D->A, AB->D}. Find the min cover of ***E***.

- Step 1: All FDs are in canonical form. Nothing to do in this step.
- Step 2: Only FD that has more than one attribute on the LHS is AB->D. Need to check if A or B is redundant.
  - Can AB->D be replaced by A->D or B->D?
  - Answer: Yes. Since B->A, we have B->AB. By transitivity, we get B->AB->D, so B->D. Hence, we can replace AB->D by B->D.
  - Now we have: {B->A, D->A, B->D}.
- Step 3: Are there any redundant FDs in {B->A, D->A, B->D}?
  - Yes, B->D and D->A, so we can derive B->D->A. Hence, B->A is redundant. It should be eliminated.
- Finally, min cover of ***E*** is: {B->D, D->A}.

Min Cover Examples

- Let the given set of FDs be: ***G*** = {A->BCDE, CD->E}. Find the min cover of ***G***.

- Step 1: Convert FDs into canonical form:
  - {A->B, A->C, A->D, A->E, CD->E}
- Step 2: Only FD that has more than one attribute on the LHS is CD->E. Need to check if C or D is redundant.
  - We cannot derive C->E or D->E, hence they are not redundant.
  - Step 2 takes no action.
- Step 3: Are there any redundant FDs?
  - Since A->C and A->D, we have A->CD. By transitivity, A->CD->E, we can derive A->E. Thus, A->E is redundant. It should be removed.
- Finally, min cover of ***G*** is: {A->B, A->C, A->D, CD->E}.