Functional Dependencies

Orit Abitbol 039467113
Shani Goldberg 065961914
Roee Minster 066373770

Functional Dependencies (FD) - Definition

- Let R be a relation scheme and X, Y be sets of attributes in R.
- A functional dependency from X to Y exists if and only if:
  - For every instance |R| of R, if two tuples in |R| agree on the values of the attributes in X, then they agree on the values of the attributes in Y.
- We write $X \to Y$ and say that X determines Y.

- Example on Student (sid, name, supervisor_id, specialization):
  - {supervisor_id} → {specialization} means:
    - If two student records have the same supervisor (e.g., Johann), then their specialization (e.g., Databases) must be the same.
    - On the other hand, if the supervisors of 2 students are different, we do not care about their specializations (they may be the same or different).

Armstrong's Axioms

Be X, Y, Z sets of attributes in a relation scheme of a relation R and F is a set of functional dependencies for R.

Reflexivity:

If $Y \subseteq X$, then $F \vdash X \rightarrow Y$ (trivial FDs).

- Example:

  {name, supervisor_id}$\rightarrow${name}

Augmentation:

If F |– X→Y , then F |– X∪Z→Y∪Z.

- Example:

  if {supervisor_id} →{specialization},

  then {supervisor_id, name}→{specialization, name}.

Transitivity:

If F |– X→Y and F |– Y→Z, then F |– X→Z.

- Example:

  if {supervisor_id} →{specialization} and

  {specialization} →{lab},

  then {supervisor_id}→{lab}.

Properties of Armstrong’s Axioms

- Armstrong’s axioms are sound and complete.
- Sound:
  If $F \vdash f$ then $F \models f$.
- Complete:
  If $F \models f$ then $F \vdash f$.

Where F is a set of FDs and f is a single FD.

Translating Armstrong's Axioms into First Order Logic

Reflexivity:

Be X→Y and

	R		
t_1	u_1	....	u_m
t_2	u_1'	....	u_m'
	. . .		. . .

Then

$$\forall u_1, u_1', u_2, u_2', \ldots, u_m, u_m'$$

$$R(u_1, \ldots, u_m) \cap R(u_1', \ldots, u_m') \cap \left(t_1[X] = t_2[X]\right) \to \left(t_1[Y] = t_2[Y]\right)$$

Where

$$X = u_{x_1}, \ldots, u_{x_k} \;\; and \;\; Y = u_{y_1'}, \ldots, u_{y_k'} \;\; and \;\; Y \subseteq X.$$

Augmentation:

If $\forall u_1, u_1^{'}, u_2, u_2^{'}, \ldots, u_m, u_m^{'}$

$$R(u_1, \ldots, u_m) \cap R(u_1^{'}, \ldots, u_m^{'}) \cap \left(t_1[X] = t_2[X]\right) \rightarrow \left(t_1[Y] = t_2[Y]\right)$$

Then

$$\forall u_1, u_1^{'}, u_2, u_2^{'}, \ldots, u_m, u_m^{'}$$

$$R(u_1, \ldots, u_m) \cap R(u_1^{'}, \ldots, u_m^{'}) \cap \left(t_1[XW] = t_2[XW]\right) \rightarrow \left(t_1[YW] = t_2[YW]\right)$$

Where

$$X = u_{x_1}, \ldots, u_{x_k} \;\; and \;\; Y = u_{y_{1^{'}}}, \ldots, u_{y_{k^{'}}} \;\; and \;\; W = u_{w_{1^{"}}}, \ldots, u_{w_{k^{"}}}.$$

Transitivity:

If

$$\forall u_1, u_1^{'}, u_2, u_2^{'}, \ldots, u_m, u_m^{'}$$

$$R(u_1, \ldots, u_m) \cap R(u_1^{'}, \ldots, u_m^{'}) \cap \left(t_1[X] = t_2[X]\right) \rightarrow \left(t_1[Y] = t_2[Y]\right)$$

And

$$\forall u_1, u_1^{'}, u_2, u_2^{'}, \ldots, u_m, u_m^{'}$$

$$R(u_1, \ldots, u_m) \cap R(u_1^{'}, \ldots, u_m^{'}) \cap \left(t_1[Y] = t_2[Y]\right) \rightarrow \left(t_1[W] = t_2[W]\right)$$

Then

$$\forall u_1, u_1^{'}, u_2, u_2^{'}, \ldots, u_m, u_m^{'}$$

$$R(u_1, \ldots, u_m) \cap R(u_1^{'}, \ldots, u_m^{'}) \cap \left(t_1[X] = t_2[X]\right) \rightarrow \left(t_1[W] = t_2[W]\right)$$

Where

$$X = u_{x_1}, \ldots, u_{x_k} \;\; and \;\; Y = u_{y_1^{'}}, \ldots, u_{y_k^{'}} \;\; and \;\; W = u_{w_1^{''}}, \ldots, u_{w_k^{''}}.$$

Projection of Functional Dependencies

- Given a set of FDs F over R, we want to know which set of FDs is satisfied in a smaller relation scheme, S where S is a subset of R.

- Definition:

  The projection of a set of FDs F over R onto a relation scheme S, where S is a subset of R, is given by:

  $$F[S] = \{X \rightarrow Y \mid X \rightarrow Y \in F \ \ and \ \ XY \subseteq S\}$$

  The FDs in F[S] are said to be *embedded* in S.

- If $F^{+}[S] = \{X \rightarrow Y \mid X \rightarrow Y \in F^{+} \ \mathit{and} \ XY \subseteq S\}$ then S is said to *preserve* the set of FDs F over R.

- Theorem:
  - There exist a relation r over R and a relation s over S where S $\subset$ R and a set of FDs F over R s.t s|=F^{+}[S]. But, there does not exist a relation r over R where r|= F and s $= \pi_S(r)$ s.t s|=F^{+}[S].

- Proof:
  - Let R be a relation scheme R={A,B,C,D,E,H,I} and let S be a relation scheme S={A,B,C,D,E} .
    F={A→I, B→I, C→H, D→H, IH→E}.
    Let G= {AC→E, AD→E, BC→E, BD→E}, G is a cover of $F^+[S]$.
    Let s={t_1, t_2, t_3, t_4} be the relation over S where S is:

A	B	C	D	E
a_1	b_1	c_1	d_1	e_1
a_2	b_2	c_2	d_2	e_2
a_3	b_3	c_1	d_2	e_3
a_1	b_2	c_3	d_3	e_4

Can you complete the columns H and I?

It easy to see that s|=G and s|= $F^+[S]$. Suppose that there exist a relation r over R s.t r|=F and s= $\pi_S(r)$.
We can conclude that $\exists u_1,u_2,u_3, u_4 \in r$ s.t for $i \in \{1,2,3,4\}$, $t_i=u_i[S]$.
Let u_1 [HI] =< h_1, i_1 >:

A	B	C	D	E	H	I
a_1	b_1	c_1	d_1	e_1	h_1	i_1
a_2	b_2	c_2	d_2	e_2	h_1	i_1
a_3	b_3	c_1	d_2	e_3	h_1	
a_1	b_2	c_3	d_3	e_4		i_1

Then the following equalities can be deduced from F:

1. $u_1[A]=u_4[A]$ and $A \rightarrow I \in F \Rightarrow u_1[I]=u_4[I]$
2. $u_2[B]=u_4[B]$ and $B \rightarrow I \in F \Rightarrow u_2[I]=u_4[I]$
3. (1) + (2) $\Rightarrow u_1[I]= u_2[I]$.
4. $u_1[C]=u_3[C]$ and $C \rightarrow H \in F \Rightarrow u_1[H]=u_3[H]$
5. $u_2[D]= u_3[D]$ and $D \rightarrow H \in F \Rightarrow u_2[H]=u_3[H]$
6. (4) + (5) $\Rightarrow u_1[H]=u_2[H]$.
7. (3) + (6) $\Rightarrow u_1[IH]=u_2[IH]$.

We have assumed that $r|=F$ therefore, $u_1[E]= u_2[E]$ since $IH\rightarrow E \in F$.
However, this leads to a contradiction since $t_1[E]\neq t_2[E] \Rightarrow u_1[E] \neq u_2[E]$.

A	B	C	D	E	H	I
a_1	b_1	c_1	d_1	e_1	h1	i1
a_2	b_2	c_2	d_2	e_2	h1	i1
a_3	b_3	c_1	d_2	e_3	h1	
a_1	b_2	c_3	d_3	e_4		i1

We can conclude that there does not exist a relation r over R s.t $r|=F$ and $s= \pi_S(r)$.

This proof was taken from “A Guided Tour of Relational Databases and Beyond”, Mark Levene and George Loizou, Springer Publishing Company.

Inclusion Dependencies

- Definition:
  - An Inclusion Dependency (IND) over a DB scheme R is a statement of the form $R_1[X] \subseteq R_2[Y]$ where R1,R2∈R and X,Y are sequences of attributes s.t $X \subseteq R_1$, $Y \subseteq R_2$ and $|X|=|Y|$.
- Example: $STUDENTS[Department] \subseteq DEANS[Department]$

STUDENTS	
Student	**Department**
Roee	CS
Shani	CS
Orit	Math
Ira	Math
Yossi	Biology

DEANS	
Dean	**Department**
Eyal	CS
Chris	Biology
John	Math
Mark	Physics

Multivalued Dependencies

- Definition:
  - An MVD $X \rightarrow\rightarrow Y(R)$ is satisfied in a relation r over R, denoted by $r \models X \rightarrow\rightarrow Y(R)$, if $\forall\, t_1, t_2 \in r$, if $t_1[X]=t_2[X]$, then $\exists t_3 \in r$ s.t:
    1. $t_1[X]=t_2[X]=t_3[X]$.
    2. $t_3[Y]= t_1[Y]$, $t_3[Z]= t_2[Z]$.

- Example:

Furniture	Color	Num_of_Legs
Table	Brown	4
Table	Brown	6
Table	Black	4
Table	Black	6
Chair	Black	3
Chair	Black	1

- In this table we have the MVD
  Furniture→→Num_of_Legs.

Chase FDs - Test for Looseness Join

Chase FDs - Test for Lossless Join

- Why do we need it for?

  To know when natural join of two or more relations is meaningful, which means that the join operation does not cause any loss of information.

  The Chase test will allow us to conclude whether or not a natural join of a given decomposition is a lossless join.

  Clarification: Loss in lossless refers to the loss of information and not the loss of rows. Actually the loss of information occurs because of added rows in the joins.

The chase algorithm is composed of two parts:

1. the pre-processing of the input.
2. the execution of the algorithm itself using the processed input from part 1.

Chase FDs - Test for Lossless Join

- Input for Part 1:
  - Relation scheme $R = (A_1, A_2, \ldots, A_n)$.
  - Decomposition $D = (R_1, R_2, \ldots, R_k)$.
- Input for Part 2:
  - Set of FDs F.

- Output:
- **YES** if the decomposition has the lossless join property.
- **NO** otherwise.

Part 1:

1. Build an empty table of size k×n, where k is the number of the decompositions and n is the number of the attributes s.t:

	A_1	A_2	…	A_n
R_1				
R_2				
⋮				
R_k				

Part 1 (continue):

2. Fill each column A_i as follow:

```
for (i=1, i<=n, i++)
{       index=1
        for (j=1,  j<=k, j++)
        {           if (Ai is in Rj)
                        write the lower case letter of the attribute
                    else
                      write the lower case letter of the attribute
                      with the index++.
        }
}
```

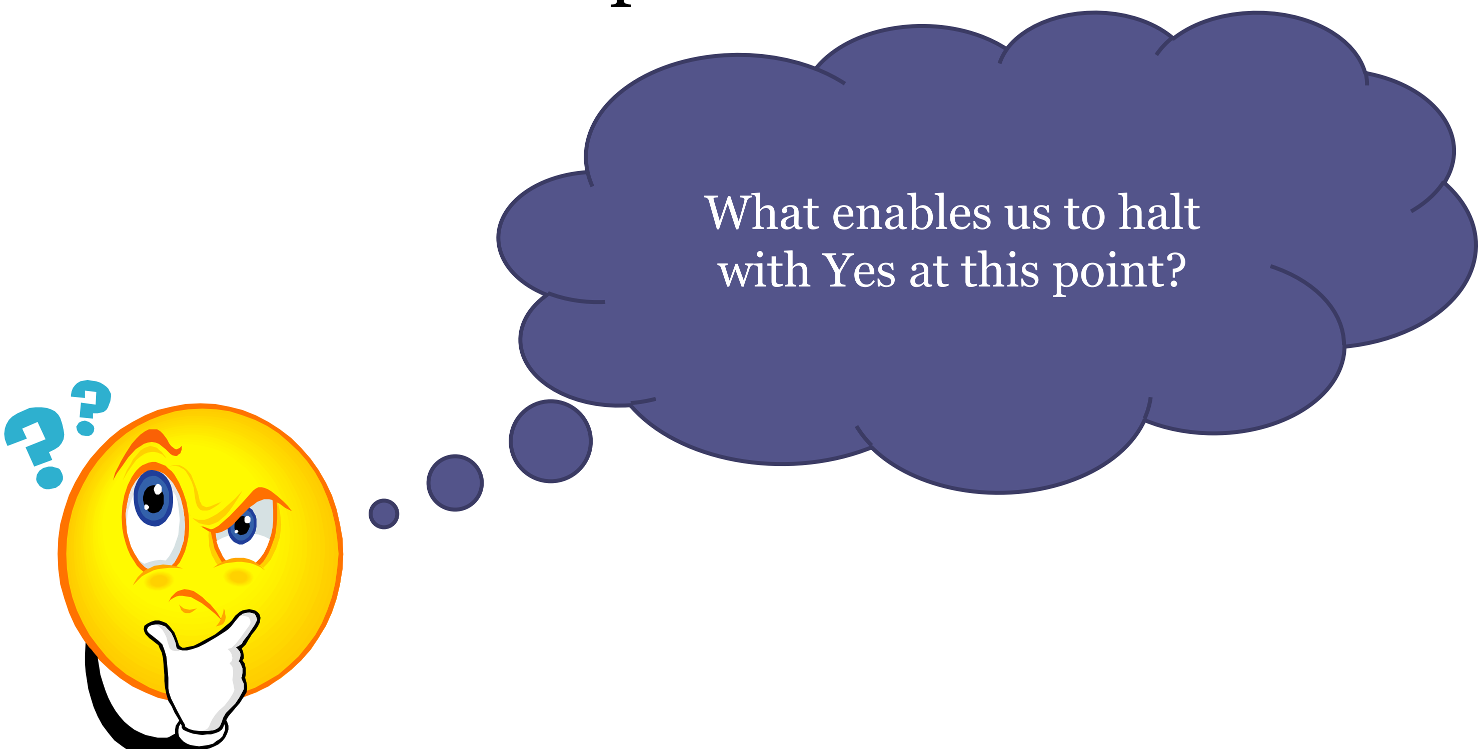
Example: given R={A,B,C,D}, D={AB,BC,CD}

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a_1	b	c	d_2
R_3	a_2	b_1	c	d

Part 1 (continue):

Halt with **YES** if entire row is without indexes,

otherwise continue to part 2.



Part 2:

- Use the given FD's to force indexed letters to become non-indexed letters (meaning if A→B and you have a and b_i in the same row, then b_i becomes b).
- Example: lets continue the previous example using the processed table from part 1. Given F={B→A}

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a_1	b	c	d_2
R_3	a_2	b_1	c	d

B→A

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a	b	c	d_2
R_3	a_2	b_1	c	d

Part 2 (continue):

- Halt with **YES** if entire row is without indexes, otherwise continue going over all the FD’s until halting with **YES** or if no more changes can be done then halt with **NO**.

- <u>Part 2 (continue)</u>:
- <u>Example</u>: In our last example we don't have a row without indexes and there are no more functional dependencies we can use, so we will halt with **NO**.
- But, if we also had the functional dependency C→D in F we could continue with the algorithm:

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a_1	b	c	d_2
R_3	a_2	b_1	c	d

C→D

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a	b	c	d
R_3	a_2	b_1	c	d

Part 2 (continue):

In this case we halt with **YES** since there is an entire row without indexes.

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a	b	c	d
R_3	a_2	b_1	c	d

- Let’s go back to the question that was raised in the end of part 1 of the Chase test:
  - After we built the first Chase table at the end of part one what enables us to halt with Yes even before executing the second part of the Chase test?

- Lemma 1:
  - At the end of part one of the Chase test the first Chase table contains a row with no indexes if the given decomposition D contains the scheme R itself (meaning $R=\{A_1,A_2,\ldots,A_n\}$).

- Example:
  - R={A,B,C}
  - D={AB,BC,ABC}

	A	B	C
R_1	a	b	c_1
R_2	a_1	b	c
R_3	a	b	c

- Proof:
  - Since the decomposition D contains R then at the end of part one of the Chase test, the first Chase table contains a row without indexes and we can halt with **YES**. This decomposition is indeed lossless since the natural join $r = \rhd\lhd_{i=1}^{k} \pi_{R_i}(r)$ will give us the original table r over R because there exist i s.t Ri=R.

- **Lemma 2**:

  If the Chase test halts with YES then r|=F.

  - **Example**:

Lets go back to our main example where R={A,B,C,D}, D={AB,BC,CD}, F={B→A, C→D}. The last table we got is (lets mark this table r_{last}):

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a	b	c	d
R_3	a_2	b_1	c	d

You can see that there is an indexless row, so the Chase test halts with YES and also r_{last} |=F.

- Lemma 3 :
  If r|=F then the Chase test doesn't necessarily halts with YES.

  - Example:
  Back to our main example where R={A,B,C,D}, D={AB,BC,CD}, F={B→A}. The last table we got is (lets mark this table r_{last}):

	A	B	C	D
R_1	a	b	c_1	d_1
R_2	a	b	c	d_2
R_3	a_2	b_1	c	d

You can see that r_{last} |=F, but in this case the Chase test halts with NO.

- Now you try it…

1. Given R={A,B,C} and D={AB,BC} execute the first part of the Chase test.
2. Lets mark the table you got as r_1 . Write all the FD's s.t $r_1 |= F$ but the Chase test halts with NO.
3. Write the FD's that are necessary for the Chase test to halt with Yes.

The answers are in the next slide…

1. Given R={A,B,C} and D={AB,BC} the first part of the Chase test produce this table:

	A	B	C
R_1	a	b	c_1
R_2	a_1	b	c

2. The FD's s.t r_1|=F but the Chase test halts with NO are: A→B, C→B, A→C, C→A. These FD's are vacuous truth.
3. The FD's that are necessary for the Chase test to halt with Yes are:
   - B→C: force the first row of r_1 to become indexless.
   - B→A : force the second row of r_1 to become indexless.

   For each one of these FD's the Chase test will halt with YES.

References

- "Foundation of Databases", Serge Abiteboul, Richard Hull and Victor Vianu, Addison-Wesley Publishing Company.
- "A Guided Tour of Relational Databases and Beyond", Mark Levene and George Loizou, Springer Publishing Company.
- "Functional Dependencies", Lecture 5, Database Systems Course 236363.
- "Normal Forms", Lecture 6, Database Systems Course 236363.