Lecture 5

Chapter 3: Functional Dependencies, Section 3.1 — 3.2.5

John Connor

February 20, 2018

Instance (2.2.6)

Recall that in Lecture 2 we defined *instance*:

Instance (2.2.6)

Recall that in Lecture 2 we defined *instance*:

A set of tuples for a given relation is called an instance of the relation.

Instance (2.2.6)

Recall that in Lecture 2 we defined *instance*:

A set of tuples for a given relation is called an instance of the relation.

Inserts, updates, and deletes can all transform instances of a relation to new instances.

Key

Recall that in Lecture 2 we defined key:

Key

Recall that in Lecture 2 we defined key:

A set of attributes is called a *key* if no two tuples in the relation instance can have the same values for all elements of the key.

Key

Recall that in Lecture 2 we defined key:

A set of attributes is called a *key* if no two tuples in the relation instance can have the same values for all elements of the key.

In our example movies schema, the key was (`title, year`).

Functional Dependency

A *functional dependency* f on a relation R is a statement of the form

$$f = A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m.$$

which is read "$A_1 A_2 \cdots A_n$ functionally determine $B_1 B_2 \cdots B_n$."

Functional Dependency

A *functional dependency* f on a relation R is a statement of the form

$$f = A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m.$$

which is read "$A_1 A_2 \cdots A_n$ functionally determine $B_1 B_2 \cdots B_n$."

This statement means that if two tuples of R have the same values for the attributes $A_1, A_2, \cdots, A_n$ then they must also have the same values for the attributes $B_1, B_2, \cdots, B_m$.

Functional Dependency

A *functional dependency* f on a relation R is a statement of the form

$$f = A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m.$$

which is read "$A_1 A_2 \cdots A_n$ functionally determine $B_1 B_2 \cdots B_n$."

This statement means that if two tuples of R have the same values for the attributes $A_1, A_2, \cdots, A_n$ then they must also have the same values for the attributes $B_1, B_2, \cdots, B_m$.

If f is true of every pair of tuples of **every instance** of R then R is said to satisfy f.

Key: A New Definition

A set of attributes $A = \{A_1, A_2, \cdots, A_n\}$ is a *key* for a relation if

Key: A New Definition

A set of attributes $A = \{A_1, A_2, \cdots, A_n\}$ is a *key* for a relation if

1. the set functionally determines all other attributes of the relation;

Key: A New Definition

A set of attributes $A = \{A_1, A_2, \cdots, A_n\}$ is a *key* for a relation if

1. the set functionally determines all other attributes of the relation;
2. A is minimal. That is, no proper subset of A functionally determines all of the other attributes of the relation.

Key: Question

Can a relation have more than one key?

Key: Question

Can a relation have more than one key?

Yes!

Key: Question

Can a relation have more than one key?

Yes!

As an example, just take any relation R with a key k, and then create a new relation R' by adding an attribute k' to the relation and assign to each tuple of R' a unique value for k'. Now k and k' are keys of R'.

Super Keys

A set of attributes that contains a key is called a *superkey*.

Super Keys

A set of attributes that contains a key is called a *superkey*.
Note that a key is also a superkey.

Super Key: Problems (Exercise 3.1.3)

Suppose R is a relation with attributes $A_1, A_2, \cdots, A_n$. As a function of n, tell how many superkeys R has

1. The only key is A_1.

Super Key: Problems (Exercise 3.1.3)

Suppose R is a relation with attributes $A_1, A_2, \cdots, A_n$. As a function of n, tell how many superkeys R has

1. The only key is A_1.
2. The only keys are A_1 and A_2.

Super Key: Problems (Exercise 3.1.3)

Suppose R is a relation with attributes $A_1, A_2, \cdots, A_n$. As a function of n, tell how many superkeys R has

1. The only key is A_1.
2. The only keys are A_1 and A_2.
3. The only keys are $\{A_1, A_2\}$ and $\{A_3, A_4\}$.

Super Key: Problems (Exercise 3.1.3)

Suppose R is a relation with attributes $A_1, A_2, \cdots, A_n$. As a function of n, tell how many superkeys R has

1. The only key is A_1.
2. The only keys are A_1 and A_2.
3. The only keys are $\{A_1, A_2\}$ and $\{A_3, A_4\}$.
4. The only keys are $\{A_1, A_2\}$ and $\{A_1, A_3\}$.

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \to B$ and the functional dependency $B \to C$ then does R satisfy $A \to C$?

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \rightarrow B$ and the functional dependency $B \rightarrow C$ then does R satisfy $A \rightarrow C$?

Yes!

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \rightarrow B$ and the functional dependency $B \rightarrow C$ then does R satisfy $A \rightarrow C$?

Yes!

Given two tuples $(a_1, b_1, c_1), (a_2, b_2, c_2)$

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \to B$ and the functional dependency $B \to C$ then does R satisfy $A \to C$?

Yes!

Given two tuples $(a_1, b_1, c_1), (a_2, b_2, c_2)$
if $a_1 = a_2$ then $b_1 = b_2$, since R satisfies $A \to B$;

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \to B$ and the functional dependency $B \to C$ then does R satisfy $A \to C$?

Yes!

Given two tuples $(a_1, b_1, c_1), (a_2, b_2, c_2)$
if $a_1 = a_2$ then $b_1 = b_2$, since R satisfies $A \to B$;
but if $b_1 = b_2$ then $c_1 = c_2$, since R satisfies $B \to C$.

Functional Dependencies : Reasoning (3.2.1)

If the relation $R(A, B, C)$ satisfies the functional dependency $A \rightarrow B$ and the functional dependency $B \rightarrow C$ then does R satisfy $A \rightarrow C$?

Yes!

Given two tuples $(a_1, b_1, c_1), (a_2, b_2, c_2)$
if $a_1 = a_2$ then $b_1 = b_2$, since R satisfies $A \rightarrow B$;
but if $b_1 = b_2$ then $c_1 = c_2$, since R satisfies $B \rightarrow C$.

From the assumption that $a_1 = a_2$ we derived $c_1 = c_2$, therefore R satisfies $A \rightarrow C$.

Functional Dependency: Question

If R satisfies the functional dependency

$$A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m.$$

is this equivalent to satisfying the set of functional dependencies

$$\begin{aligned} A_1 A_2 \cdots A_n &\to B_1 \\ A_1 A_2 \cdots A_n &\to B_2 \\ &\vdots \\ A_1 A_2 \cdots A_n &\to B_m? \end{aligned}$$

Functional Dependency: Question

If R satisfies the functional dependency

$$A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m.$$

is this equivalent to satisfying the set of functional dependencies

$$\begin{gathered} A_1 A_2 \cdots A_n \to B_1 \\ A_1 A_2 \cdots A_n \to B_2 \\ \vdots \\ A_1 A_2 \cdots A_n \to B_m? \end{gathered}$$

Yes!

Functional Dependency: Splitting Rule

If R satisfies the functional dependency

$$A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_i \cdots B_m.$$

It satisfies

$$A_1 A_2 \cdots A_n \to B_1 B_2 \cdots B_m$$
$$A_1 A_2 \cdots A_n \to B_i$$

Functional Dependency: Combining Rule

If R satisfies the functional dependencies

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_m$$
$$A_1 A_2 \cdots A_n \rightarrow B_i$$

It satisfies

$$A_1 A_2 \cdots A_n \rightarrow B_1 B_2 \cdots B_i \cdots B_m.$$

Is There A Splitting Rule for the Left Hand Side?

If R satisfies the functional dependency

$$A_1 A_2 \rightarrow B_1$$

does it satisfy

$$A_1 \rightarrow B_1$$
$$A_2 \rightarrow B_1?$$

Is There A Splitting Rule for the Left Hand Side?

If R satisfies the functional dependency

$$A_1 A_2 \rightarrow B_1$$

does it satisfy

$$A_1 \rightarrow B_1$$
$$A_2 \rightarrow B_1?$$

No!

Is There A Splitting Rule for the Left Hand Side?

If R satisfies the functional dependency

$$A_1 A_2 \to B_1$$

does it satisfy

$$A_1 \to B_1$$
$$A_2 \to B_1?$$

No!
Consider $\texttt{title}, \texttt{year} \to \texttt{length}$.

Is There A Splitting Rule for the Left Hand Side?

If R satisfies the functional dependency

$$A_1 A_2 \rightarrow B_1$$

does it satisfy

$$A_1 \rightarrow B_1$$
$$A_2 \rightarrow B_1?$$

No!
Consider $\texttt{title}, \texttt{year} \rightarrow \texttt{length}$. Does $\texttt{year} \rightarrow \texttt{length}$ seem likely?

Functional Dependency: Closure

Given a set $A = \{A_1, \cdots, A_n\}$ of attributes and a set S of functional dependencies, the closure of A under S is the set of attributes $B = \{B_1, \cdots, B_m\}$ such that every relation that satisfies S satisfies $A \to B$.

Functional Dependency: Closure

Given a set $A = \{A_1, \cdots, A_n\}$ of attributes and a set S of functional dependencies, the closure of A under S is the set of attributes $B = \{B_1, \cdots, B_m\}$ such that every relation that satisfies S satisfies $A \rightarrow B$.

We denote the closure of A as A^+.

Functional Dependency: Closure Algorithm (3.7)

`INPUT`: A set of attributes $A = \{A_1, \cdots, A_n\}$ and a set S of functional dependencies.

`OUTPUT`: The closure of A under S.

1. Split the functional dependencies in S so each functional dependency has a single attribute on the right.
2. Let X be the set of attributes which will eventually become the closure. Initialize $X = A$.
3. Repeatedly search for some functional dependency

$$B_1 B_2 \cdots B_m \rightarrow C$$

such that all $B_i \in X$ but $C \notin X$. If such an attribute C is found, add C to X and repeate step 3. Otherwise return X.

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S.

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D,$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B,$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E,$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F,$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$
6. $E \to F \in S'$ and $E \in X$ so add F to X. $(X = \{B, C, D, E, F\})$

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$
6. $E \to F \in S'$ and $E \in X$ so add F to X. $(X = \{B, C, D, E, F\})$
7. Are we done yet?

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$
6. $E \to F \in S'$ and $E \in X$ so add F to X. $(X = \{B, C, D, E, F\})$
7. Are we done yet? Yes!

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$
6. $E \to F \in S'$ and $E \in X$ so add F to X. $(X = \{B, C, D, E, F\})$
7. Are we done yet? Yes! Are there any functional dependencies in S that we did not use?

Closure Algorithm : Example 1 (3.8)

Let

$$A = \{B, C\}$$
$$S = \{BC \to D, CD \to BE, E \to F, DG \to C\}$$

1. Split S. Let $S' = \{BC \to D, CD \to B, CD \to E, E \to F, DG \to C\}$
2. $X = A = \{B, C\}$
3. $BC \to D \in S'$ and $B, C \in X$ so add D to X. $(X = \{B, C, D\})$
4. $CD \to B \in S'$ and $C, D \in X$ so add B to X. $(X = \{B, C, D\})$
5. $CD \to E \in S'$ and $C, D \in X$ so add E to X. $(X = \{B, C, D, E\})$
6. $E \to F \in S'$ and $E \in X$ so add F to X. $(X = \{B, C, D, E, F\})$
7. Are we done yet? Yes! Are there any functional dependencies in S that we did not use? Yes! We did not use $DG \to C$.