In the Lecture Series Introduction to Database Systems

Functional Dependencies

Presented by Stéphane Bressan

Anomalies: Example

salary = f (position), functional dependency

Assume that the position determines the salary:

position → salary

company

eNumber	firstName	lastName	address	department	Position	salary
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk	2000
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee	1200
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk	2000
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee	1200
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager	2500
755Y	Sylvia	Tok	22 East Coast Lane	Toys	Manager	2600
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager	2200
?	?	?	?	?	Security guard	1500

key   key

Dummy entries very dangerous

Redundant storage → Redundancy

Update anomaly

↳ Manager ⇒ 2500 & 2600 ?!

↳ anomaly

Insertion anomaly

(No security guard now, so no way to (mb salary)

Potential deletion anomaly

↘ If deleted, then the assistant manager position is gone. Loss of info

Normalization: Example

employee

eNumber	firstName	lastName	address	department	Position
1XU3	Dewi	Srijaya	12a Jln Lempeng	Toys	Clerk
4W3E	Izabel	Leong	10 Outram Park	Sports	Trainee
3XXE	John	Smith	107 Clementi Rd	Toys	Clerk
5SD2	Axel	Bayer	55 Cuscaden Rd	Sports	Trainee
6RG5	Winnie	Lee	10 West Coast Rd	Sports	Manager
755Y	Sylvia	Tok	22 East Coast Lane	Toys	Manager
2SD3	Eric	Wei	100 Jurong drive	Toys	Assistant manager

key   key

salary

Position	salary
Clerk	2000
Trainee	1200
Manager	2500
Assistant manager	2200
Security guard	1500

key

- Redundant storage?
  - NO
- Update anomaly?
  - NO
- Deletion anomaly?
  - NO
- Insertion anomaly?
  - NO

Learning Objectives

- Definitions
- Reasoning (Armstrong's axioms)
- Closure and Equivalence
- Minimal Cover

Functional Dependencies

For a relation scheme R, a functional dependency from a set S of attribute of R to a set T of attribute of R exists if and only if:

For every instance of |R| of R, if two t-uples in |R| agree on the values of the attributes in S, then they agree on the values of the attributes in T.

We write: S → T

Functional Dependencies

company(eNumber, firstName, lastName, address, department, position, salary)

{position} → {salary}

If two t-uples in the relation company have the same value for the attribute position then they must have the same value for the salary attribute.

Functional Dependencies

employee(eNumber, firstName, lastName, address, department, position)

{firstName, lastName} $\rightarrow$ {eNumber, address, department, position}

If two t-uples in the relation employee relation have the same first name and last name then they must ~~are~~ be the same t-uple (no duplicate)

One to One function.

Introduction to Database Systems

Strong relationship btw pri keys & functional dependency

Functional Dependencies

company(enumber, firstname, lastname, address, department, position, salary)

{position} $\rightarrow$ {salary}

$\forall$X1 $\forall$X2 $\forall$X3 $\forall$X4 $\forall$X5 $\forall$X6 $\forall$X7 $\forall$X8 $\forall$X9 $\forall$X10 $\forall$P $\forall$S1 $\forall$S2
((company(X1, X2, X3, X4, X5, P, S1)
$\wedge$ company(X6, X7, X8, X9, X10, P, S2))
$\Rightarrow$ (S1 = S2))

↳ in terms of DRC

Introduction to Database Systems

Functional Dependencies

company(enumber, firstname, lastname, address, department, position, salary)

{position} $\rightarrow$ {salary}

```
CHECK ( NOT EXISTS (
SELECT *
FROM company c1, company c2
WHERE c1.position=c2.position AND c1.salary <> c2.salary))
```

Introduction to Database Systems

Functional Dependencies

salary(position, salary)

{position} $\rightarrow$ {salary}

```
CHECK ( NOT EXISTS (
SELECT *
FROM salary s1, salary s2
WHERE s1.position=s2.position AND s1.salary <> s2.salary))
```

PRIMARY KEY (position) ↳ can just simply do this

Introduction to Database Systems

Trivial FDs !!

$X \rightarrow Y$

$Y \subset X$ → PROPER subset

{firstName, address} $\rightarrow$ {firstName}

Introduction to Database Systems

Non-Trivial FDs !

$X \rightarrow Y$

$Y \not\subset X$ → Not a proper subset

{eNumber} $\rightarrow$ {address}

{firstName, lastName} $\rightarrow$ {firstName, address}

Introduction to Database Systems

Completely Non-Trivial FDs

!!

$X \rightarrow Y$

$\boxed{Y \cap X = \emptyset}$!! ** *

{firstName, lastName} → {address}

Superkeys

A set of attributes whose knowledge determines the value of the entire t-uple is a **superkey**

employee(eNumber, firstName, lastName, address, department, position, salary)

smallest size in terms of inclusion

{firstName, lastName} → candidate key

{eNumber} → candidate key

{firsName, lastName, address} → Not. cos contain first 2 → smallest set exist

{eNumber, address} → Not cos contain enumber

Candidate Keys

A minimal (for inclusion) set of attributes whose knowledge determines the value of the entire t-uple is a **candidate key**

employee(eNumber, firstName, lastName, address, department, position, salary)

{firstName, lastName}

{eNumber}

Any candidate key is definitely a superkey

Primary Keys

The designer chooses one candidate key to be the ***primary key***

Reasoning about Functional Dependencies

It is sometimes possible to infer new functional dependencies from a set of given functional dependencies

(independently from any particular instance of the relation scheme or of any additional knowledge)

Reasoning about Functional Dependencies

For example:

From

{eNumber} → {firstName}

and

{eNumber} →{lastName}

We can infer

{eNumber} → {firstName, lastName}

Armstrong's Axioms

- Be X, Y, Z be subsets of the relation scheme of a relation R
- **Reflexivity**: → Every trivial func dependency is true

  If $Y \subset X$, then $X \to Y$
- **Augmentation**:

  If $X \to Y$, then $X \cup Z \to Y \cup Z$
- **Transitivity**:

  If $X \to Y$ and $Y \to Z$, then $X \to Z$

Reduce redundancy by adding on both sides

* Cannot remove common things from both sides

Armstrong's Axioms

employee(eNumber, firstName, lastName, address, department, position, salary)

Reflexivity:

If {firstName} ⊂ {firstName, lastName},

Then {firstName, lastName} → {firstName}

Armstrong's Axioms

employee(eNumber, firstName, lastName, address, department, position, salary)

Augmentation:

If {position} → {salary},

then {position, eNumber} → {salary, eNumber}

Armstrong's Axioms

employee(eNumber, firstName, lastName, address, department, position, salary)

Transitivity:

If {eNumber} → {position}

and {position} → {salary},

Then {eNumber} → {salary}

Armstrong's Axioms

Armstrong's axioms are sound

For example: Transitivity

Let X, Y, Z be subsets of the relation R

If $X \to Y$ and $Y \to Z$, then $X \to Z$

Proof:
1. Let R be a relation scheme.
2. Let $X \to Y$ and $Y \to Z$ be two functional dependencies on R.
3. Let T1 and T2 be two tuples of |R| are such that, for all attributes Ax in X, T1. Ax = T2.Ax.
4. We know that for all Ay in Y, T1. Ay = T2.Ay since $X \to Y$
5. We know that for all Ay in Y, T1. Ay = T2.Ay since $Y \to Z$
6. Therefore for all Az in Z, T1. Az = T2.Az
7. Therefore $X \to Z$

Q.E.D

Armstrong's Axioms

Armstrong's axioms are sound

For example: Consider the scheme {name, room, tel} with the set of functional dependencies:

{{room} → {tel}, {tel} → {name}}

We can deduce that the following functional dependency holds:

{room} → {name}

Proof:
1. Let R= {name, room, tel}
2. Let {room} → {tel} be a functional dependency on R
3. Let {tel} → {name} be a functional dependency on R
4. Therefore {room} → {name} holds on R by Transitivity of (2) and (3)

Q.E.D.

Armstrong's Axioms

Armstrong's axioms are sound

For example: Weak-Augmentation
Let X, Y, Z be subsets of the relation R

If $X \rightarrow Y$, then $X \cup Z \rightarrow Y$

Proof

1. Let R be a relation scheme
2. Let $X \rightarrow Y$ be a functional dependency on R
3. Therefore $X \cup Z \rightarrow Y \cup Z$ by Augmentation of (2) with Z
4. We know that $Y \cup Z \rightarrow Y$ by Reflexivity because $Y \subset Y \cup Z$
5. Therefore $X \cup Z \rightarrow Y$ by Transitivity of (3) and (4)

Q.E.D.

Closure of a Set of Functional Dependencies

For a set F of functional dependencies, we call the closure of F, noted F+, the set of all the functional dependencies that F entails

Armstrong's Axioms

Armstrong's axioms are complete

F+ can be computed by applying the Armstrong Axioms in all possible ways

↳ All possible Armstrong Axioms

Closure of a Set of Functional Dependencies

Consider the relation scheme R(A,B,C,D)

- F = {{A} →{B},{B,C} →{D}}
- F+ = {{A} →{A}, {B}→{B}, {C}→{C}, {D}→{D}, […], {A}→{B}, {A,B}→{B}, {A,D}→{B,D}, {A,C}→{B,C}, {A,C,D}→{B,C,D}, {{A} →{A,B}, {A,B}→{A,B}, {A,D}→{A,B,D}, {A,C}→{A,B,C}, {A,C,D}→{A,B,C,D}, {B,C} →{D}, […], {A,C} →{D}, […]}

Equivalence of Sets of Functional Dependencies

Two sets of functional dependencies F and G are equivalent if and only if

F+ = G+

Finding Keys: Example

Example: Consider the relation scheme R(A,B,C,D)

WTS : {A, B} → {A,B,C,D}

with functional dependencies:

{A}→{C} and {B}→{D}.

Is {A,B} a candidate key?

Augmentation : {A,B} → {B,C} → Add B on both sides
{A,B} → {A,B,C} → Add A on both sides
{A,B,C} → {A,B,C,D} → Add A,B,C on both sides

Transitivity : {A,B} → {A,B,C,D}

↳ cos : {A,B} → {A,B,C} → {A,B,C,D}

∴ Superkey

Finding Keys: Example

Proper presentation

Example: {A,B} is a superkey.

Proof

1. We know that $\{A\} \rightarrow \{C\}$
2. Therefore $\{A,B\} \rightarrow \{A,B,C\}$, by augmentation of (1) with{A,B}
3. We know that $\{B\} \rightarrow \{D\}$
4. Therefore $\{A,B,C\} \rightarrow \{A,B,C,D\}$, by augmentation of (3) with {A, B, C}
5. Therefore $\boxed{\{A,B\} \rightarrow \{A,B,C,D\}}$ by transitivity of (2) and (4)

Q.E.D

Finding Keys: Example

Example: {A,B} is a candidate key (minimal)

We must show that neither {A} nor {B} alone are candidate keys

This can be done by producing counter example relation instance verifying the functional dependencies given but neither $\{A\}\rightarrow\{A,B,C,D\}$ nor $\{B\}\rightarrow\{A,B,C,D\}$

We will however learn an algorithm to do otherwise

Closure of a Set of Attributes

For a set A of attributes, we call the **closure of A** *(with respect to a set of functional dependencies F)*, noted A+, the maximum set of attributes such that $A\rightarrow A+$ *(as a consequence of F)*

Closure of a Set of Attributes: Example

Consider the relation scheme R(A,B,C,D) with functional dependencies

$\{A\}\rightarrow\{C\}$ and $\{B\}\rightarrow\{D\}$.

- {A}+ = {A,C}
- {B}+ = {B,D}
- {A,B}+ = {A,B,C,D}

$\{C\}^+ = \{C\}$
$\{D\}^+ = \{D\}$
$\{AC\}^+ = \{AC\}$
$\{AD\}^+ = \{A,C,D\}$
$\{BC\}^+ = \{B,C,D\}$
$\{BD\}^+ = \{B,D\}$

Closure of a Set of Attributes: Algorithm 1

- Input:
  - R a relation scheme
  - F a set of functional dependencies
  - $X \subset R$
- Output:
  - X+ the closure of X w.r.t. F

Closure of a Set of Attributes: Algorithm 1

- $X^{(0)} := X$
- Repeat
  - $X^{(i+1)} := X^{(i)} \cup A$, where A is the union of the sets Z of attributes such that there exist $Y \rightarrow Z$ in F, and $Y \subset X^{(i)}$
- Until $\boxed{X^{(i+1)} := X^{(i)}}$!!
- Return $X^{(i+1)}$

Closure of a Set of Attributes: Example

R = {A,B,C,D,E,G}

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, {D}→{E,G}, {B,E}→{C}, {C,G}→{B,D}, {C,E}→{A,G}}

X = {B,D}

↓

∴ Super key

$X^{0} = \{B,D\}$

$X^{(1)} = \{B,D,E,G\}$ $(\{D\} \rightarrow \{E,G\})$

$X^{(2)} = \{B,C,D,E,G\}$ $(\{B,E\} \rightarrow \{C\})$

$X^{(3)} = \{A,B,C,D,E,G\}$ $(\{C\} \rightarrow \{A\})$

$X^{(4)} = \{A,B,C,D,E,G\}$

Closure of a Set of Attributes: Example

R = {A,B,C,D,E,G}
F = { {A,B}→{C}, {C}→{A}, {B,C}→{D}, {A,C,D}→{B}, {D}→{E,G}, {B,E}→{C}, {C,G}→{B,D}, {C,E}→{A,G}}
X = {B,D}

- $X^{(0)}$ = {B,D}
- {D}→{E,G}
- $X^{(1)}$ = {B,D,E,G}
- {B,E}→{C},
- $X^{(2)}$ = {B,C,D,E,G}
- {C,E}→{A,G}
- $X^{(3)}$ = $X^{(4)}$ = X^+ = {A,B,C,D,E,G}

Equivalence of Sets of Functional Dependencies

Every set F of functional dependencies is equivalent to a set of functional dependencies Y→Z such that Z is a singleton, i.e. every right-hand side has a single attribute

Minimal Set of Dependencies

A set of dependencies F is minimal if and only if:

1. Every right-hand side is a single attribute
2. For no functional dependency X→A in F and proper subset Z of X is F – {X→A} ∪ {Z→A} equivalent to F
3. For no functional dependency X →A in F is the set F – {X→A} equivalent to F

Minimal Cover

A set of functional dependencies F is a **minimal cover** of a set of functional dependencies G if and only if

- F is minimal
- F is equivalent to G

!!

- (an **extended minimal cover** is obtained by undoing step 1)

Minimal Cover

- Every set of functional dependencies has a minimal cover
- There might be several different minimal cover of the same set

Eg: {D} → {E,G} ⟺ {D} → {E}, {D} → {G}
{D} → {E} ⟹ {D,G} → {E,G}
{D} → {G} ⟹ {D} → {D,G}
⟹ {D} → {E,G}

{D} → {E,G}: {E,G} → {E} : Reflexivity
{E,G} → {G} : Reflexivity
⟹ {D} → {E}, {D} → {G} } Transitivity

Minimal Cover: Example

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D},
{A,C,D}→{B}, {D}→{E,G}, {B,E}→{C},
{C,G}→{B,D}, {C,E}→{A,G}}

① Simplify all RHS!! : {D} → {E,G} : {D} → {E}, {D} → {G}
{C,G} → {B,D} : {C,G} → {B}, {C,G} → {D}
{C,E} → {A,G} : {C,E} → {A}, {C,E} → {G}

Minimal Cover: Example (1)

F = { {A,B}→{C}, {C}→{A}, {B,C}→{D},
{A,C,D}→{B}, {D}→{E,G}, {B,E}→{C},
{C,G}→{B,D}, {C,E}→{A,G}}

F' = { {A,B}→{C}, {C}→{A}, {B,C}→{D},
{A,C,D}→{B}, *{D}→{G}, {D}→{E}*,
{B,E}→{C}, *{C,G}→{B}, {C,G}→{D}*,
{C,E}→{A}, {C,E}→{G}}

{C,E} → {C} → {A}
can be compressed.

{D} → {G} ⟹ {C,D} → {C,G} → {B} > {C,D} → {B}
AND: {A,C,D} → {C,D} (Reflexivity)

Minimal Cover: Example (2)

Hence can be compressed

② Simplify LHS

F' = {{C}→{A}, {C,E}→{A}, {A,C,D}→{B},
{C,G}→{B}, {A,B}→{C}, {B,E}→{C},
{B,C}→{D}, {C,G}→{D}, {D}→{E},
{C,E}→{G}, {D}→{G}}

{C} → {A}
⟹ {C,E} → {A}
{A,C,D} → {B}
can derive
{C,D} → {B}
from the other dependencies

F'' = {*{C}→{A}, {C,D}→{B}*, {C,G}→{B},
{A,B}→{C}, {B,E}→{C}, {B,C}→{D},
{C,G}→{D}, {D}→{E}, {C,E}→{G},
{D}→{G}}

Minimal Cover: Example (3)

③ Remove things you can safely remove

F'' = {{C}→{A}, {C,D}→{B}, {C,G}→{B},
{A,B}→{C}, {B,E}→{C}, {B,C}→{D},
* {C,G}→{D}, {D}→{E}, {C,E}→{G},
{D}→{G}}

can be obtained from other functional dependencies

{C,G} → {C,D} → {B}

(*) {C,G} → {D}
⟹ {C,G} → {C,D}

F''' = {{C}→{A}, {C,D}→{B}, {A,B}→{C},
{B,E}→{C}, {B,C}→{D}, {C,G}→{D},
{D}→{E}, {C,E}→{G}, {D}→{G}}

Extended Minimal Cover: Example (4)

F'''' = {{C}→{A}, {C,D}→{B}, {A,B}→{C},
{B,E}→{C}, {B,C}→{D}, {C,G}→{D},
{D}→{E,G}, {C,E}→{G}}

Combine {D} → {E}
{D} → {G}

iff proof of 1st step!!

Minimal Cover: Algorithm

We can apply steps (1), (2), (3) iteratively in various orders

However only $\boxed{(1) + (2) + (3)}$ is guaranteed to lead to a minimal cover!:

- Put functional dependencies in single attribute rhs form
- Minimize left side of each functional dependency
- Delete redundant functional dependencies

Credits

The content of this lecture is based on chapter 8 of the book "Introduction to database Systems"
By
S. Bressan and B. Catania,
McGraw Hill publisher

Clipart and media are licensed from Microsoft Office Online Clipart and Media