Design Theory for Relational Databases

Functional Dependencies

Guillaume Raschia — Polytech Nantes; Université de Nantes
Last update: November 11, 2021

Contents

[Source : J. Ullman, Stanford]

Integrity Constraints

Functional Dependencies

$X \to Y$

An FD is an assertion about a relation R that whenever two tuples of R **agree on all the attributes of** X, then they must also **agree on all attributes in set** Y

$$X \to Y \quad := \quad \forall t, u \in R,\ t[X] = u[X] \implies t[Y] = u[Y]$$

- Say "X determines Y" or "X gives Y" and also "$X \to Y$ holds in R"
- Convention: ..., X, Y, Z represent set of attributes; A, B, C, ...represent single attributes
- Convention: no set formers in sets of attributes, just ABC rather than $\{A, B, C\}$

Example FD's

`Drinkers(name, addr, beersLiked, brewery, favBeer)`

Expected FD's to assert:

1. name → addr favBeer
   - Note: this FD is the same as name → addr and name → favBeer
   - No splitting rule for the left-hand side (lhs)
2. beersLiked → brewery

Example Data

name	addr	beersLiked	brewery	favBeer
Alice	Nantes	Trompe Souris	La Divatte	Titan
Alice	Nantes	**Titan**	Bouffay	Titan
Bob	Rennes	**Titan**	Bouffay	Titan

FD's

- **name** → addr implies (**Alice**, Nantes) twice
- **name** → favBeer implies (**Alice**, Titan) twice
- **beersLiked** → brewery implies (**Titan**, Bouffay) twice

Keys of Relations

- K is a **superkey** for relation R if K functionally determines all the attributes of R
  In other words, a set of attributes K is a superkey in R if for any two tuples t, u in R, $t[K] = u[K]$ implies $t = u$. That is, a superkey is a set of attributes that **uniquely identifies** a tuple in a relation
- K is a **key** for R if K is a superkey, but no proper subset of K is a superkey: K is minimal

Among the—**candidate**—**keys**, arbitrarily promote one into the **primary key**

Example: Superkey

`Drinkers(name, addr, beersLiked, brewery, favBeer)`

{name, beersLiked} is a superkey
because together these attributes determine all the other attributes

- name → addr favBeer
- beersLiked → brewery

Example: Key

```
Drinkers(name, addr, beersLiked, brewery, favBeer)
```

{name, beersLiked} is a key

because neither {name} nor {beersLiked} is a superkey

- name doesn't → brewery
- beersLiked doesn't → addr

There are no other keys, but lots of superkeys: any superset of {name, beersLiked}

Where Do Keys Come From?

1. Just assert a—surrogate—key K
   - The only FD's are $K \to A$ for all attributes A
2. Assert FD's and deduce the keys
   - Like we did on the previous Drinkers example

More FD's From "Physics"

FD's are integrity contraints on the database, coming from the real-life problem

Example

"no two courses can meet in the same room at the same time"

- tells us: hour room → course

Short Digression on Inclusion Dependencies

```
Drinkers(name, addr, beersLiked, brewery, favBeer)
Bars(name, addr)
Frequents(drinker, bar)
```

Inclusion Dependencies (IND)

1. Every drinker from the `Frequents` table must be an existing name in the `Drinkers` table
2. Every bar from the `Frequents` table must be an existing name in the `Bars` table

Inclusion Dependencies

IND is a Referential integrity

Attributes of one relation refer to values in another one

Formally, we have an inclusion dependency $S[Y] \subseteq R[X]$ when every value of the set of attributes Y in S also occurs as a value of the set of attributes X in R:

$$\pi_Y(S) \subseteq \pi_X(R)$$

Foreign Keys

- Most often IND's occur as part of a **foreign key**
- Foreign key is a conjunction of a primary key and an IND:

$$S[X] \subseteq R[K] \text{ and } K \text{ is a key in } R$$

Example: Foreign Key

```
Bars(name, addr)
Frequents(drinker, bar)
```

The Bars-Frequents link

- As an IND, we expect `Frequents.bar` from `Frequents` to be found in `Bars.name`
- Since name is a primary key in `Bars`, then `Frequents.bar` is a foreign key in `Frequents`

Inference System

Inferring FD's

We are given a set of FD's $\mathcal{F} = \{f_i\}_{1 \le i \le n}$, and we want to know whether an FD $X \to A$ must hold in any relation that satisfies the given FD's

Example

If $A \to B$ and $B \to C$ hold, surely $A \to C$ holds, even if we don't say so

The inference system is important for the design of good relation schemas

Inference Test

To test if $X \to A$, start by assuming two tuples t and u agree on all attributes of X

R	X	A	the rest
t	00...0	0	00...0
u	00...0	?	??...?

Use the given FD's to infer that these tuples must also agree in certain other attributes

- If A is one—subset—of these attributes, then $X \to A$ is true
- Otherwise, the two tuples, with any forced equalities, form a two-tuple relation that proves $X \to A$ does not follow from the given FD's

Example: Inference Test

Question

Does $A \to C$ holds in $R(A, B, C, D)$ with $\mathcal{F} = \{A \to B, B \to C\}$?

R	A	B	C	D
t	0	0	0	0
u	0	?	?	?

$\Longrightarrow$ by $A \to B$

R	A	B	C	D
t	0	0	0	0
u	0	0	?	?

$\Longrightarrow$ by $B \to C$

R	A	B	C	D
t	0	0	0	0
u	0	0	0	?

- Then, if any t and u agree on A, they agree on C
- $A \to C$ follows from $\mathcal{F}$, also denoted $\mathcal{F} \models A \to C$

Closure Test

An easier way to test is to compute the **closure** of X, denoted X^+

1. Basis: $X^+ = X$
2. Induction: look for an FD's lhs Y that is a subset of the current X^+. If the FD is $Y \to Z$, add Z to X^+
3. Stop when a fixpoint is reached

Example: Closure Test

$$\mathcal{F} = \{AB \to CD, C \to A, B \to DE, A \to E, DE \to F\}$$

CD^0 = {CD}	init. step
CD^1 = $CD^0 \cup$ {A} = {CDA}	by firing C→ A, C in CD^0
CD^2 = $CD^1 \cup$ {E} = {CDAE}	by firing A → E, A in CD^1
CD^3 = $CD^2 \cup$ {F} = {CDAEF}	by firing DE → F, DE in CD^2
CD^4 = CD^3 = CD^+	

Side note: CDA, CDE, CDF, CDAE, CDAF, CDEF, CDAEF all have closure =CDAEF

Closure Test and Inference

Definition (Attribute Closure)

$$X^+ = \{A \mid \mathcal{F} \models X \to A\}$$

Does $X \to A$ follows from $\mathcal{F}$?

$\Longleftrightarrow$ Membership test: Does $A \in X^+$?

Back to Key Finding

Remember: $K \to$ all attributes and K is minimal

1. For each subset of attributes X, compute X^+
2. Add X as a new key if $X^+ =$ all attributes
3. However, drop XY whenever we add X
   - Because XY is a non-minimal superkey

A Few Tricks

- No need to compute the closure of the empty set or of the set of all the attributes
- If we find $X^+ =$ all attributes, so is the closure of any superset of X
  - Then, it's worth considering X by increasing cardinalities
- If an attribute is not in any rhs of FD, then it MUST be part of every key
  - Step 1 is then: Find non-rhs attributes Z then for each subset ZX...

Example: Key Finding

ABCD with $\mathcal{F} = \{A \to B, AC \to D, D \to C\}$

1. Only A is non-rhs attribute
2. $A^+ = AB$; A is not superkey
3. $AB^+ = AB$
   - Since AB is already a—subset of a—closure (of A), then $AB^+ = A^+$
4. $AC^+ = ACDB$; AC is a (super)key
5. $AD^+ = ADCB$; AD is a (super)key
6. ABC, ABD, ACD may be skipped as obvious superkeys
7. Any other subset does not contain A

Keys are AC, AD

Projecting FD's

Finding All Implied FD's

Motivation

normalization: the process where we break a relation schema into two or more schemas

Example

ABCD with FD's $AB \to C$, $C \to D$, and $D \to A$

- Decompose into ABC, AD: What FD's hold in ABC?
- Not only $AB \to C$, but also $C \to A$!

All Implied FD's

Definition (Closure of $\mathcal{F}$)

$$\mathcal{F}^+ = \{X \to Y \mid \mathcal{F} \models X \to Y\}$$

Example: ABCD with $\mathcal{F} = \{AB \to C, C \to D, D \to A\}$

In $\mathcal{F}^+$, one can find:

- all the FD's from $\mathcal{F}$
- **trivial FD's**: $A \to A$, $AB \to A$, ..., $B \to B$, ...
- $ABD \to CD$, $CA \to DA$, $CB \to DB$, ...
- $AB \to D$, $C \to A$

How to be sure not to forget any FD?

Reasoning with FD's

Armstrong's axioms

1. **Reflexivity** (trivial FD): if $X \supseteq Y$, then $X \rightarrow Y$
2. **Augmentation**: if $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
3. **Transitivity**: if $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- These are sound and complete inference rules for FD's!
  - $\mathcal{F}^+$ is the result of applying these 3 rules
  - syntactic $\vdash$ and semantic $\models$ are mainly the same
- Usually, we are only concerned with **nontrivial** FD's: rhs not contained in lhs

Reasoning with FD's (cont'd)

Commonly derived rules

4. **Union**: if $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
5. **Decomposition**: if $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
6. **Pseudo-transitivity**: if $X \rightarrow Y$ and $YZ \rightarrow T$, then $XZ \rightarrow T$

Project FD's onto Attributes

Given ABC with FD's $\mathcal{F} = \{A \rightarrow B, B \rightarrow C\}$

Problem: project onto AC

Basic Idea

1. Start with given FD's in $\mathcal{F}$ and find all nontrivial FD's that follow from $\mathcal{F}$ w.r.t. the Armstrong's axioms
2. Restrict to those FD's that involve only attributes of the projected schema

Simple Yet Exponential Algorithm

1. For each subset of attributes X in the projected schema, compute X^+
2. Add $X \rightarrow A$ for all A in $X^+ - X$ only if A is a projected attribute
3. However, drop $XY \rightarrow A$ whenever we discover $X \rightarrow A$
   - Because $XY \rightarrow A$ follows from $X \rightarrow A$ in any projection

A Few Tricks

- No need to compute the closure of the empty set or of the set of all the projected attributes
- If we find $X^+ =$ all attributes, so is the closure of any superset of X

Example: Projecting FD's

Given ABC with FD's $\mathcal{F} = \{A \to B, B \to C\}$

Problem: project onto AC

- $A^+ = ABC$ yields $A \to C$
  - We do not need to compute AC^+
- $C^+ = C$ yields nothing

Projection of $\mathcal{F}$ onto AC is $\mathcal{F}_{\text{AC}} = \{A \to C\}$

Equivalence Test

Given $\mathcal{F} = \{A \to B, B \to C\}$ and $\mathcal{G} = \{A \to B, B \to C, A \to C\}$

How to check $\mathcal{F}$ and $\mathcal{G}$ are the same?

- $\mathcal{F}$ not equal to $\mathcal{G}$ but $\mathcal{F}^+$ equal to $\mathcal{G}^+$
- A dead end: compute $\mathcal{F}^+$ and $\mathcal{G}^+$?!
- Solution: check both $\mathcal{F}$ implies $\mathcal{G}$ and $\mathcal{G}$ implies $\mathcal{F}$

Equivalence of FD's

$$\mathcal{F} \equiv \mathcal{G} \iff \mathcal{F}^+ = \mathcal{G}^+ \iff \mathcal{F} \models \mathcal{G} \text{ and } \mathcal{G} \models \mathcal{F}$$

Is $\mathcal{F}$ the same than $\mathcal{G}$?

$\mathcal{F} = \{A \to B, B \to C\}$ and $\mathcal{G} = \{A \to B, B \to C, A \to C\}$

Show $\mathcal{F} \models \mathcal{G}$ and $\mathcal{G} \models \mathcal{F}$

1. $\mathcal{G} \models \mathcal{F}$:
   - Each FD in $\mathcal{F}$ follows from $\mathcal{G}$: trivial
2. $\mathcal{F} \models \mathcal{G}$:
   - $A \to B$ and $B \to C$ in $\mathcal{G}$ both follows from $\mathcal{F}$: trivial
   - Does $A \to C$ follows from $\mathcal{F}$? Answer yes, by closure test

Conclusion

Minutes

- Functional Dependencies are integrity constraints in Databases
- Keys and Foreign Keys are specific forms of FD's
- One can reason with FD's thx to Armstrong's axioms
- The closure test is a simple yet powerful tool for inference
- FD's projection requires closure computation