Introduction to Operating Systems and SQL for Data Science

Practice 10 - Normalization

Rules for defining “good” relations

- We should define simple relations
- Set of attributes in a relation should describe one entity
- We should avoid data duplication (preventing errors in updates)
- We should avoid many null values

Rules for defining “good” relations

Look at the following relation:

(id, student_name, address, department_code, number_of students,
course_id, course_name, grade, credit_poimts)

Rules for defining “good” relations

Look at the following relation:

(id, student_name, address, department_code, number_of students,
course_id, course_name, grade, credit_poimts)

<u>Identify problems with the above relation:</u>

Rules for defining “good” relations

Look at the following relation:

(id, student_name, address, department_code, number_of students, course_id, course_name, grade, credit_poimts)

<u>Identify problems with the above relation:</u>

- Not clear what the relation describes
- What is the key

Normalization

Goals:

- Define rules for relation validation – normalization rules
- For each relation- check rules, if the rules are not satisfied, we should fix it by dividing the relation to valid sub relations

Dependency types

<u>Functional dependency:</u>

- Dependency between A and B marked A→B
- A defining B (knowing A => knowing B)
- N:1 relation (can be many a's with same b value)

An example:

Id→Name

- 1234 → Jhon snow
- 3646 → Jhon snow
- 5678 → Cersei Lannister

Reichman University

Dependency types

<u>Injective dependency:</u>

- Injective dependency between A and B marked A←→B
- knowing A => knowing B and vice versa
- 1:1 relation

examples:

- ID ←→ Student_id
- Department_id ←→ Head_of_dep_id (assuming a department has one head and one can be head of one department at most)

Dependency types

<u>Complex dependency:</u>

- Value of A can be connected to many B values and vice versa
- Marked A<←→> B
- N:M relation

An example:

Course_id <←→> Student_id

Normalization rules -1NF

<u>1NF – first normal form:</u>

- Each attribute contain atomic values – meaning an attribute does not contain some list or inner data structure
- There must be a unique key (not null)

In the relational model, each relation should be at least in 1NF.
An example for “bad” relation:

Students(<u>id,</u> name, department, taken_course_ids)

The field taken_courses_ids is not atomic therefore the relation is NOT 1NF

Normalization rules -2NF

2NF – second normal form:

- Each attribute which is not the key has a functional dependency to the entire key.

For each functional dependency (X→Y) we well copy X U Y to a new relation, where X's attributes define a key. And delete Y's attributes from the original relation.

Normalization rules -2NF

An example:

(student_id, course_id, student_name, id, course_name, grade)

Where:

(student_id, course_id) → grade

Course_id→course_name

Student_id→student_name

We will have 3 relations:

Students(student_id,student_name)

Courses(course_id,course_name)

Grades(student_id,course_id,grade)

Reichman University

Normalization rules -3NF

Super key: we say that A is a super key of R

3NF – third normal form:

- R is in 3NF if for every FD $X \rightarrow A \in F+$ such that $A \notin X$
  - X is a superkey of R or
  - A is contained in a key of R

For each "bad" functional dependency ($X \rightarrow Y$) we well copy X U Y to a new relation,
where X's attributes define a key. And delete Y's attributes from the original relation.

Reichman University

Normalization rules -3NF

An example:

(student_id,department_id, student_name, id, department_name)

Where:

department id→ department_name

Student_id→student_name,department_id

We will have 2 relations:

Students(student_id,student_name, department_id)

Departments(department _id, department _name)

Reichman University

Normalization rules -BCNF

<u>BCNF – Boyce Codd Normal Form:</u>

- For each functional dependency (X→Y) X should be a super key

An example:

Orders(supplier_id, supplier_code, item-code, amoune)

With the following dependencies:

Supplier_id + item code are a key

Supplier_code + item_code are a key

Supplier_id → supplier_code

the third dependency is a problem since supplier id is not a super key.
We will have 2 relations:

Suppliers(<u>supplier_id,</u>supplier_code)

Orders(<u>supplier_id,Item_code</u>,amount)

Reichman University

How to check whether a schema is BCNF?

By definition:

- Compute F^+
- For every FD $X \rightarrow Y \in F^+$ check whether X is a superkey.

Problem: the size of F^+ is exponential in R

<u>Theorem</u>: If R is not BCNF with respect to F (there exists an FD $X \rightarrow Y \in F^+$ that violates the conditions), then there exists an FD $Z \rightarrow W \in F$ that violates the conditions.

Reichman University

BCNF and 3NF

- Every BCNF schema is also in 3NF
  - The opposite does not necessarily hold
- BCNF prevents more duplications than 3NF
- There always exists a 3NF decomposition that preserves dependencies and information
  - This is not true in BCNF and therefore we will sometimes prefer 3NF even though it is less efficient in sense of duplications

Normalization rules -Summary

NF	
1NF	Non null key Atomic Values
2NF	Each attribute which is not the key has a functional dependency to the entire key.
3NF	Each attribute which is not the key doesn't have a functional dependency to a different attribute which is not in some key.
BCNF	For each functional dependency (X→Y) X should be a candidate key

Reichman University

Normal Forms - Questions

In a cellular company, it was decided to set up an information system that would help manage the company data. The system will store customer information - ID number, name, address and phone number as well as bank account information and credit card for debit. Each customer selects a route from a list of possible routes. Each route, identified by a unique name, has a number of commitment months, a monthly fee and a description of the track.

The system holds information about the catalog of devices and accompanying equipment - catalog number, device's name and the accompanying equipment and its price. For mobile devices, we also have the type of device (second, third generation, smartphone, etc.).

The system holds information about the device and accompanying equipment owned by the customers. The system saves the date the device and equipment were sold to the customer. For some mobile devices, a warranty is provided. The system retains the types of warranty (each type of warranty is given a serial number) including the duration of the warranty and its contents.

Normal Forms - Questions

Describe the relations the system should have:

Reichman University

Normal Forms - Questions

Describe the relations the system should have:

Customer (ID, name, address, bank account details, credit card number, route name)

Route (route name, description, commitment months, monthly payment)

Devices(catalog_number, name, price)

Mobile device (catalog_number, type)

Ownership (catalog_number,customer_id, date of sale)

Warranty (id, duration, type, contents)

Device warranty (warranty_id, catalog_number)

Reichman University

Decomposition

- A decomposition of R is a set $\{R_1,\ldots,R_n\}$ such that $U_{i=1,\ldots,n} R_i = R$

- Motivation:
  - Allows a better modeling of the database
- Charechteristics of a good modeling:
  - Preserves information (necessary)
  - Preserves dependencies (not necessary, desired)

Reichman University

Preserving Information

- R - a relational schema
- F – a set of FD's
- $P=\{R_1,\ldots,R_n\}$ decomposition

- P preserves information with respect to F if for every relation r over R such that $r \vDash F$ the following holds:

$$\bowtie_{i=1..n} \pi_{R_i}(r)=r$$

Reichman University

Preserving Information - example

- R={ID, Name, Address}
- F={ID→Name, ID→Address}
- Does the following decomposition preserve data?
  - P={R_1(ID, Name), R_2(Name, Address)

- No!

Reichman University

Preserving Information – example

r =

ID	NAME	ADDR
1	Alice	CA
2	Alice	TX

$\pi_{R_1}(r)$

ID	NAME
1	Alice
2	Alice

$\pi_{R_2}(r)$

NAME	ADDR
Alice	CA
Alice	TX

$\pi_{R_1}(r) \bowtie \pi_{R_2}(r)$

ID	NAME	ADDR
1	Alice	CA
1	Alice	TX
2	Alice	CA
2	Alice	TX

- Note that the relation obtained by joining the two projections is different than the original relation.

Reichman University

Preserving Information – example

- However, the decomposition

$$\rho' = \{R'_1(ID, NAME), R'_2(ID, ADDR)\}$$

  preserves information.

- Indeed, $\pi_{R'_1}(r) \bowtie \pi_{R'_2}(r) = r$

r =

ID	NAME	ADDR
1	Alice	CA
2	Alice	TX

$\pi_{R'_1}(r)$

ID	NAME
1	Alice
2	Alice

$\pi_{R'_2}(r)$

ID	ADDR
1	CA
2	TX

Algorithm for information preserving

- R - a relational schema and P={R_1,…,R_n} a decomposition
- Does P preserves information?

- We show the algorithm's run on the following example R(A,B,C,D,E,H)
- P={R1(A, B), R2(A, C, D), R3(B, E, H)}

Reichman University

Algorithm for information preserving

- 1st step - Initialization:
  - Create a relation r over R such that:
    - Every schema R_i has its own tuple t_i
    - For every attribute A:
      - If $A \in R_i$ then t[A]=a
      - Else $t[A]=a_i$
      - Similarly with b for B, c for C, etc.

Reichman University

Algorithm for information preserving

- In our example:

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b_2	c	d	e_2	h_2
t_3	a_3	b	c_3	d_3	e	h

R(A, B, C, D, E, H)

$F = \{A \rightarrow B, C \rightarrow D, B \rightarrow EH\}$

$\rho = \{R_1(A, B), R_2(A, C, D), R_3(B, E, H)\}$

Reichman University

Algorithm for information preserving

- 2nd step - chase
  - While the table changes do:
    - Look for an FD violation and equate the conclusions
    - "Equate" = change every occurrence of one to the other
      - When equating a_j with a, change a_j with a.
- 3nd step:
  - Return true if and only if there is a row without indexes.

Reichman University

Algorithm for information preserving

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b_2	c	d	e_2	h_2
t_3	a_3	b	c_3	d_3	e	h

F = {A → B, C → D, B → EH}

$$A \rightarrow B\ (t_1,t_2)$$

1

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b	c	d	e_2	h_2
t_3	a_3	b	c_3	d_3	e	h

Reichman University

Algorithm for information preserving

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b_2	c	d	e_2	h_2
t_3	a_3	b	c_3	d_3	e	h

F = {A → B, C → D, B → EH}

$A \rightarrow B(t_1,t_2)$

1

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b	c	d	e_2	h_2
t_3	a_3	b	c_3	d_3	e	h

2

$B \rightarrow EH(t_2,t_3)$

	A	B	C	D	E	H
t_1	a	b	c_1	d_1	e_1	h_1
t_2	a	b	c	d	e	h
t_3	a_3	b	c_3	d_3	e	h

- Note that we have a tuple without subscripts and thus the decomposition preserves information

Reichman University

Information Preserving - example

- R(A,B,C,D,E)
- F = {A→B, B→C, C→D , DE→BC}
- ρ = {R1(A,D), R2(A,E), R3(B,C,D,E)}

	A	B	C	D	E
t_1	a	b_1	c_1	d	e_1
t_2	a	b_2	c_2	d_2	e
t_3	a_3	b	c	d	e

A→B
t_1,t_2

	A	B	C	D	E
t_1	a	b_2	c_1	d	e_1
t_2	a	b_2	c_2	d_2	e
t_3	a_3	b	c	d	e

B→C
t_1,t_2

	A	B	C	D	E
t_1	a	b_2	c_1	d	e_1
t_2	a	b_2	c_1	d_2	e
t_3	a_3	b	c	d	e

C→D
t_1,t_2

	A	B	C	D	E
t_1	a	b_2	c_1	d	e_1
t_2	a	b_2	c_1	d	e
t_3	a_3	b	c	d	e

DE→BC
t_3,t_2

	A	B	C	D	E
t_1	a	b	c	d	e_1
t_2	a	b	c	d	e
t_3	a_3	b	c	d	e

No subscripts

Reminder: Minimal Cover

- Let F be a set of FDs
- A *minimal cover* of F is a set G of FDs such that $G^+=F^+$ with the following properties:
  - FDs in G have a single attribute on the right hand side; that is, they have the form $X \longrightarrow A$
  - All FDs are required: no FD $X \longrightarrow A$ in G is such that $G \setminus \{X \longrightarrow A\} \vDash X \longrightarrow A$
  - All attributes are required: no FD $XB \longrightarrow A$ in G is such that $G \vDash X \longrightarrow A$

Reichman University

Producing A Minimal Cover

- Transform the FDs in F - a single attribute on the r.h.s., let G be the result

Do

1. If there is a FD X→A in G such that A in CLOSURE(X, G \ {X→A}) delete X→A from G.
2. If there is XB→A in G such that A in CLOSURE(X, G), replace XB→A with X→A in G.

Until there are no more changes to G

- G is a minimal cover for the original F.

Reichman University

Question #1

R(A,B,C,D,E,H)

F={AB →H, E →BC, D →H, A →DE, C →E, D →BH}

Find a minimal cover to F

F is minimal if for every X→Y in F the following hold:

1. $|Y| = 1$
2. $F^+ \neq (F \setminus \{X \rightarrow Y\})^+$
3. For every $Z \subset X$ it holds that $F^+ \neq ((F \setminus \{X \rightarrow Y\}) \cup \{Z \rightarrow Y\})^+$

$G \leftarrow \{(X \rightarrow A) \mid \exists Y\ ((X \rightarrow Y) \in F \wedge A \in Y)\}$;

Repeat

1. **For each $f = X \rightarrow A \in G$ do**

   if $A \in X^+_{G \setminus \{f\}}$ then $G \leftarrow G \setminus \{f\}$;

2. **For each $f = X \rightarrow A \in G$ and $B \in X$ do**
   if $A \in (X \setminus \{B\})^+_G$ then $G \leftarrow (G \setminus \{X \rightarrow A\}) \cup \{X \setminus \{B\} \rightarrow A\}$;

until no more changes to G

Reichman University

Question #1

- Initialization

  G={AB →H, E →B, E →C, A →D, A →E, C →E, D →B, D →H}

- Step 1:
  - We omit the FD AB → H since H $\in G\backslash\{AB \rightarrow H\}^+$
  - G={E →B, E →C D →H, A →D, A →E, C →E, D →B, D →H}
- Step 2: no change
- Step 1: no change
- G={E →B, E →C, D →H, A →D, A →E, C →E, D →B}

Question #2

$G=\{E \rightarrow B, E \rightarrow C, D \rightarrow H, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B\}$

$\rho = \{R_1(A, B, D), R_2(A, C), R_3(C, D, E, H)\}$

Is ρ in BCNF?

R_1 is not BCNF

$\pi_{R1}G= \{A \rightarrow B, A \rightarrow D, D \rightarrow B\}$

D is not a superkey!

Reichman University

Question #3

$\rho = \{R_1(A, B, D), R_2(A, C), R_3(C, D, E, H)\}$

$,G=\{E \rightarrow B, E \rightarrow C, D \rightarrow H, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B\}$

Does this decomposition preserve Information?

A	B	C	D	E	H
a	b	c_1	d	e_1	h_1
a	b_2	c	d_2	e_2	h_2
a_3	b_3	c	d	e	h

$D \rightarrow H$
t_1, t_3

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b_2	c	d_2	e_2	h_2
a_3	b_3	c	d	e	h

$A \rightarrow D$
t_1, t_2

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b_2	c	d	e_2	h_2
a_3	b_3	c	d	e	h

$C \rightarrow E$
t_2, t_3

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b_2	c	d	e	h_2
a_3	b_3	c	d	e	h

$D \rightarrow B$
t_1, t_3

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b	c	d	e	h_2
a_3	b_3	c	d	e	h

Question #3

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

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b	c	d	e	h_2
a_3	b_3	c	d	e	h

E→B
t_2,t_3

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b	c	d	e	h_2
a_3	b	c	d	e	h

D→H
t_1,t_2

A	B	C	D	E	H
a	b	c_1	d	e_1	h
a	b	c	d	e	h
a_3	b	c	d	e	h

Yes!

Question #4

$\rho = \{R_1(A, B, D), R_2(A, C), R_3(C, D, E, H)\}$

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

Does this decomposition preserve dependencies?

for each $f = (X \rightarrow Y) \in F$ do begin

$Z_f \leftarrow X$;

repeat

for $i = 1$ to n do

$Z_f \leftarrow Z_f \cup ((Z_f \cap R_i)_F^+ \cap R_i)$

until no more change to Z_f

$X \rightarrow Y$ is preserved iff $Y \subseteq Z_f$

end;

ρ is dependecy preserving iff all $(X \rightarrow Y) \in F$ are preserved

Reichman University

Question #4

$\rho = \{R_1(A, B, D), R_2(A, C), R_3(C, D, E, H)\}$

$G=\{E \rightarrow B, E \rightarrow C, D \rightarrow H, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B\}$

$E \rightarrow B$ is not contained in any schema

- $Z_f = \{E\}$
- $Z_f \cap R_1=\{\}$ no change to Z_f
- $Z_f \cap R_2 =\{\}$ no change to Z_f
- $Z_f \cap R_3 = \{E\}$

  $\{E\}^+_F \cap R_3= \{C,E\}$

  ➔ $Z_f= \{C,E\}$

- $Z_f \cap R_1=\{\}$ no change to Z_f
- $Z_f \cap R_2 =\{C\}$ $\{C\}^+_F \cap R_3= \{C,E\}$ no change to Z_f
- $Z_f \cap R_3 =\{C,E\}$ $\{C,E\}^+_F \cap R_3= \{C,E\}$ no change to Z_f

$E \rightarrow B$ is not preserved!

Reichman University