Introduction to Database Systems

Exercises: Logical design and normalization

Functional dependencies

Dependencies between the attributes of the relational scheme define **which values in the relation are possible and which can not exist**.

Functional dependencies represent the relationship between the attributes of the relational schema. They limit the permissible values of the attributes in the relationship tuples.

Let X and Y be nonempty sets of attributes in relation schema R: $\mathrm{X,Y} \subseteq \mathrm{R}$.

The attributes $\mathbf{X}$ **functionally determine** attributes $\mathbf{Y}$ (denoted by $\mathbf{X} \rightarrow \mathbf{Y}$) if there do not exist two tuples in any relation to R schema that would match the values of the X attributes and would not match the values of the Y attributes.

Identification of functional dependencies

Functional dependencies can be identified in two ways:

- based on the **understanding of the relational schema** (common sense, documentation, description)
- on the basis of a **representative set of data** (dependencies must be valid at all times).

In doing so, we want functional dependencies to be complete, which means that all determinants are minimal. For the sake of simplicity, we usually find a **minimum coverage of a set of functional dependencies**.

Exercise 1

Let *EMPLOYEE* be a relational schema of a relation *employed*. Identify functional dependencies based on understanding of the relational schema.

ID	NameSurname	Position	Salary	BranchID	BranchAddress
21	Janez Novak	CEO	4800	P002	Kranjčeva ulica 25, 1000 Ljubljana
37	Zvonko Nered	Programmer	1500	P007	Koprska 10, 1000 Ljubljana

Each employee has a unique employee number. Each branch is located in precisely one location and has at most one branch in each location. The salary of the employee depends on the position of the employee and the branch in which he is employed.

Exercise 2

Let $R=ABCDE$ be a relational schema of relation r.

Identify functional dependencies based on representative set of data.

A	B	C	D	E
a	b	z	w	a
e	b	r	w	b
a	d	z	w	c
e	d	r	w	d
a	f	z	s	e
e	f	r	s	f

Reasoning about functional dependencies

Let X and Y be subsets of attributes of relational schema R: $X,Y \subseteq R$. A set of functional dependencies F_R **logically implies** dependency $X \to Y$, if every relation with schema R, that satisfies all dependencies in F_R, also meets the dependence $X \to Y$.

Denoted,

$$F_R \vDash X \to Y.$$

We use **Armstrong's Axioms** and **inferential rules** to derive functional dependencies.

Armstrong‘s Axioms

- **A1**: reflexivity $\mathbf{Y \subseteq X \Rightarrow X \rightarrow Y}$
- **A2**: augmentation $\mathbf{\{X \rightarrow Y\} \vDash XZ \rightarrow YZ}$
- **A3**: transitivity $\mathbf{\{X \rightarrow Y, Y \rightarrow Z\} \vDash X \rightarrow Z}$

Armstrong's axioms or theorems are **sound** because they generate only valid dependencies and are **complete** because they generate all valid dependencies.

Inferential rules

- **I1**: decomposition: $\mathbf{\{X \to YZ\} \vDash X \to Y}$

  $$A1 : YZ \to Y$$

  $$A3 : \{X \to YZ, YZ \to Y\} \vDash X \to Y$$

- **I2**: union: $\mathbf{\{X \to Y, X \to Z\} \vDash X \to YZ}$

  $$A2 : \{X \to Y\} \vDash X \to YX$$

  $$A2 : \{X \to Z\} \vDash YX \to YZ$$

  $$A3 : \{X \to YX, YX \to YZ\} \vDash X \to YZ$$

- **I3**: pseudotransitivity: $\mathbf{\{X \to Y, WY \to Z\} \vDash WX \to Z}$

  $$A2 : \{X \to Y\} \vDash WX \to WY$$

  $$A3 : \{WX \to WY, WY \to Z\} \vDash WX \to Z$$

Exercise 3

Let $R = ABCDXYZW$ be a relational schema and F_R and E_R corresponding sets of functional dependencies. Does the dependence set F_R logically imply all dependencies in the E_R set?

Justify the answer using Armstrong's axioms.

$$F_R = \{A \rightarrow BC, B \rightarrow D, C \rightarrow D, X \rightarrow ZW, X \rightarrow Y, YW \rightarrow Z\}$$

$$E_R = \{A \rightarrow BCD, B \rightarrow D, AC \rightarrow BD, X \rightarrow W\}$$

A1: reflexivity $Y \subseteq X \Rightarrow X \rightarrow Y$

A2: augmentation $\{X \rightarrow Y\} \vDash XZ \rightarrow YZ$

A3: transitivity $\{X \rightarrow Y, Y \rightarrow Z\} \vDash X \rightarrow Z$

I1: decomposition: $\{X \rightarrow YZ\} \vDash X \rightarrow Y$

$A1 : YZ \rightarrow Y$

$A3 : \{X \rightarrow YZ, YZ \rightarrow Y\} \vDash X \rightarrow Y$

I2: union: $\{X \rightarrow Y, X \rightarrow Z\} \vDash X \rightarrow YZ$

$A2 : \{X \rightarrow Y\} \vDash X \rightarrow YX$

$A2 : \{X \rightarrow Z\} \vDash YX \rightarrow YZ$

$A3 : \{X \rightarrow YX, YX \rightarrow YZ\} \vDash X \rightarrow YZ$

Exercise 4

Let $R = ABCDEFGH$ be a relational schema and F_R corresponding set of functional dependencies. Does the dependency set F_R logically imply dependency f ?

Justify the answer using Armstrong's axioms or inferential rules.

$$F_R = \{AB \to D, AC \to FG, B \to G, D \to B\}$$

1. $f = AC \to C$
2. $f = AC \to G$
3. $f = AB \to DG$
4. $f = AD \to BG$

A1: reflexivity $Y \subseteq X \Rightarrow X \to Y$
A2: augmentation $\{X \to Y\} \vDash XZ \to YZ$
A3: transitivity $\{X \to Y, Y \to Z\} \vDash X \to Z$

I1: decomposition: $\{X \to YZ\} \vDash X \to Y$
$A1 : YZ \to Y$
$A3 : \{X \to YZ, YZ \to Y\} \vDash X \to Y$

I2: union: $\{X \to Y, X \to Z\} \vDash X \to YZ$
$A2 : \{X \to Y\} \vDash X \to YX$
$A2 : \{X \to Z\} \vDash YX \to YZ$
$A3 : \{X \to YX, YX \to YZ\} \vDash X \to YZ$

I3: pseudotransitivity: $\{X \to Y, WY \to Z\} \vDash WX \to Z$
$A2 : \{X \to Y\} \vDash WX \to WY$
$A3 : \{WX \to WY, WY \to Z\} \vDash WX \to Z$

Key identification

Let F_R be a set of functional dependencies of relational schema R and let X be a subset of attributes of schema R: $X \subseteq R$. Attributes X are **key candidates** of schema R or corresponding relations if:

- attributes X functionally determine all the attributes of the R schema,
- There is no subset $X' \subset X$, which would functionally determine all attributes of schema R.

The relational schema may have several key candidates. We select one of them, which we call the **primary key**. A subset of the relational schema attributes, which is the primary key in another schema, is called a **foreign key**.

When identifying key candidates, we must check all possible subsets of the relational schema attributes. The first condition in the definition is checked by closing set of attributes, while the second condition gradually limits the space of possible key candidates.

Normalization

Normalization is the process of transforming relational schemes or corresponding relations into a form in which we cannot get anomalies.

Normalization uses a bottom-up approach. Planning a database thus starts with a single relationship that contains all the important attributes. The relationship is then gradually transformed into the desired normal form.

- **1NF**: first normal form
- **2NF**: second normal form
- **3NF**: third normal form
- **BCNF**: Boyce-Codd normal form
- **4(B)NF**: fourth (business) normal form
- **5(B)NF**: fifth (business) normal form

1NF, 2NF, 3NF and BCNF are based on functional dependencies, 4BNF is based on several value dependencies, 5BNF is based on contact dependencies.

First normal form (1NF)

A relation is in 1NF if it:

- has functional dependencies,
- has a primary key,
- contains only atomic values (no lists or sets).

We eliminate the superseded attributes (non-atomic value attributes) by entering the missing values or transferring the multivalued attributes together with the key to a new relation.

Second normal form (2NF)

The relation is in the second normal form 2NF, if:

- it is in first normal form 1NF and
- it doesn’t have partial dependencies.

Partial dependency means that the attributes, which are not part of the key, are functionally depend only on the part of the key.

We resolve the partial dependencies by breaking the relation to several new relations. The relation is automatically in 2NF if it is in 1NF and its key consists of only one or all of the schema attributes.

Third normal form (3NF)

Relation is in third normal form 3NF, if:

- it is in second normal form 2NF and
- it doesn‘t have transitive dependencies.

Transitive dependence is a functional dependency between attributes that are not part of the key.

We eliminate transitive dependencies by breaking the relationship into several new relations. The relation is automatically in 3NO if it is in 2NO and its key consists of all or all but one schema attribute.

Exercise 5

Let ENROLMENT be a relational schema, which represents enrollment of the students in each year. Gradually normalize the scheme to 3NF.

ENROLMENT(EnrolmentID, Name, Surname, (RegID, ProgrammeID, ProgrammeName, Year, RegistrationFee))

Exercise 6

Let $\mathrm{R{=}ABCD}$ be a relational schema, $\mathrm{F_R}$ corresponding set of functional dependencies and $\mathrm{X{=}AB}$ primary key for R.

In what normal form is the schema R?

Gradually normalize the scheme to 3NF, and for each new scheme, specify the keys and associated functional dependencies.

$$\mathrm{FR} = \{\mathrm{AB} \rightarrow \mathrm{CD}, \mathrm{C} \rightarrow \mathrm{D}\}$$

Exercise 7

Let $R{=}ABCDE$ be a relational schema, F_R corresponding set of functional dependencies.

What normal form is the R scheme in?

Gradually normalize the scheme to 3NF, and for each new scheme, specify the keys and associated functional dependencies.

$$F_R = \{A \rightarrow D, B \rightarrow E, E \rightarrow B\}$$

Introduction to database systems

Exercises: Physical design

Functional dependencies – BCNF (3.5NF)

R (<u>A, B, C</u>, D, E)

FR = {A->D, BD->E}

Can we normalise to BCNF?

- Give an example with dependency preserving decomposition
- Give an example without dependency preserving decomposition

Optimizer

For each of the following queries, identify one possible reason why an optimizer might not find a good plan. Rewrite the query so that a good plan is likely to be found.

1. An index is available on the *age* attribute:

```
SELECT E.dno
FROM Employee E
WHERE E.age=20 OR E.age=10
```

2. A B+ tree index is available on the *age* attribute:

```
SELECT E.dno
FROM Employee E
WHERE E.age<20 AND E.age>10
```

Optimizer

3. An index is available on the age attribute:

```
SELECT E.elno
FROM Enlployee E
WHERE 2*E.age<20
```

4. No index is available:

```
SELECT DISTINCT *
FROM Enlployee E
```

5. No index is available:

```
SELECT AVG (E.sal)
FROM Employee E
GROUP BY E.dno
HAVING E.dno=22
```

6. The sid in Reserves is a foreign key that refers to Sailors:

```
SELECT S.sid
FROM Sailors S, Reserves R
WHERE S.sid=R.sid
```