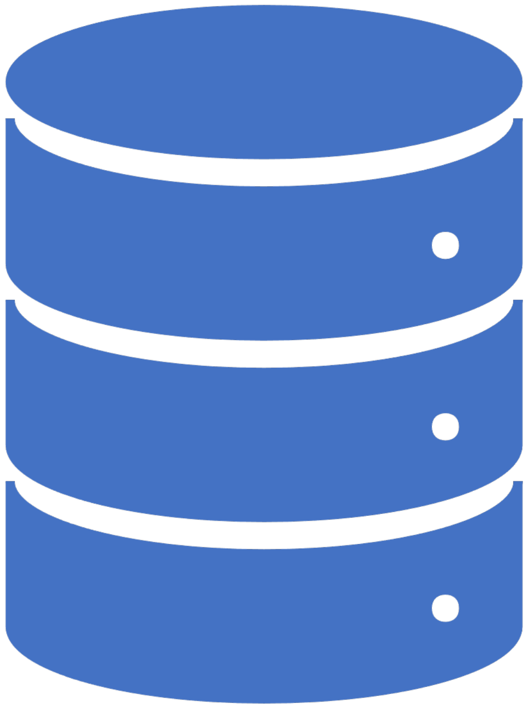


Lecture 5

Functional Dependencies

Functional Dependencies

The structure of a dabatase contains all the relations (tables with relationships) and their integrity constraints

Example 1: **Discussion** table with the following constraints

- Each student has a profesor
- Each professor has an e-mail
- Each discussion has a day, a start and an end hour for the student and for the professor

StudentName	SAge	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	19	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	20	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	21	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	19	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	20	Cristea	cp@nk.ro	Friday	15:30	16:20

Functional Dependencies

The data model is NOT GOOD!

Anomalies on INSERT

Anomalies on UPDATE

Anomalies on DELETE

StudentName	SAge	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	19	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	20	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	21	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	19	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	20	Cristea	cp@nk.ro	Friday	15:30	16:20

Functional Dependencies

So, the data model, should be ***split*** in multiple and ***good*** *structures*

Student

StudentName	Age
Rus	19
Irimie	20
Dan	21
Pavel	19

Professor

ProfessorName	PEmail
Dan	d@sd.com
Cristea	cp@nk.ro

Interaction

StudentName	ProfessorName	Day	StartHour	EndHour
Rus	Mihai	Monday	12:00	12:50
Irimie	Cristea	Tuesday	15:30	16:20
Dan	Mihai	Monday	12:00	12:50
Pavel	Mihai	Monday	13:00	13:30
Irimie	Cristea	Friday	15:30	16:20

Now:
No insert anomalies
No update anomalies
No delete anomalies

Functional Dependencies

Is it a structure good or it is bad? How can it be changed / transformed (from a bad one to a good one)?

- by using ***functional dependencies***

- The theory of functional dependencies was introduced by Edgar Frank Codd, in 1970

$$\boldsymbol{\alpha \rightarrow \beta}$$

α, β are subsets of attributes of a relation R

“α functionally determines β”

or

“β functional dependent on α”

Functional Dependencies

Definition:

Let R[A_1, A_2, …, A_n] be a relation and α, β two subsets of attributes of the relation R. ***The attribute*** $\boldsymbol{\alpha}$ (simple or composite) ***functionally determines attribute*** $\boldsymbol{\beta}$ (simple or composite), notation

$$\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$$

if and only if every value of α in R is associated with a precise and unique value for β (association that holds through the entire existence of relation R)

- If an α values appears in multiple rows, each of these rows will contain the same value for β

$$\prod_{\alpha}(t_1) = \prod_{\alpha}(t_2) \text{ implies } \prod_{\beta}(t_1) = \prod_{\beta}(t_2)$$

where $\prod_{\alpha} t$ is the projection of the $\boldsymbol{\alpha}$ atributes for the tuple t

- In the dependency $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}, \boldsymbol{\alpha}$ is the $\boldsymbol{determinant}$ and $\boldsymbol{\beta}$ is the ***dependent***

Definition (alternative):

The dependency $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is fulfilled in R if and only if for each instance of R, any two tuples t_1 and t_2 for which the α values are identical, will have also identical values for β

Functional Dependencies

- The functional dependency can be seen as a restriction (property) that has to be satisfied by the database on it's entire existence (values can be added, modified in the relation only if the functional dependency is satisfied)
- A functional dependency $\boldsymbol{\alpha} \rightarrow \boldsymbol{\beta}$ is **trivial** if $\boldsymbol{\alpha} \supseteq \boldsymbol{\beta}$
- If a relation contains a functional dependency, some of the associations among values will be store multiple times (this cause ***data redundancy***)

Example: For the relation *Discussion*, the functional dependencies are:

StudentName	SAge	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	19	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	20	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	21	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	19	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	20	Cristea	cp@nk.ro	Friday	15:30	16:20

StudentName → Age
(*StudentName* functionally determines *Age*
or
(*Age* functional dependent on *StudentName*)

ProfesssorName → PEmail

ProfesorName, Day, StartHour, EndHour → StudentName

Functional Dependencies

Example 2: Meeting [<u>StudentName, Course</u>, MeetingDate, Professor]

- key: {StudentName, Course}
- a course is associated with one professor
- a professor can be associated with multiple courses → follows the functional dependency: {Course} →{Professor}

Meeting	StudentName	Course	MeetingDate	Professor
1	Rus Maria	Databases	01/10/2021	Mihai Horatiu
2	Irimie Dan	Fundamental Algorithms	11/10/2021	Cristea Paul
3	Dan Mihai	Fundamental Algorithms	10/11/2021	Cristea Paul
4	Pavel Traian	Databases	08/10/2021	Mihai Horatiu
5	Irimie Dan	Databases	12/10/2021	Mihai Horatiu

Functional Dependencies

If the relation constains a functional dependency, some problems can arise (some of them has to be solved through programming, not only by executing SQL commands):

- **wasting space**: the same associations are stored multiple times (e.g. Course *Databases* with Professor *Mihai Horatiu* is stored *3 times)*

Meeting	StudentName	Course	MeetingDate	Professor
1	Rus Maria	Databases	01/10/2021	Mihai Horatiu
2	Irimie Dan	Fundamental Algorithms	11/10/2021	Cristea Paul
3	Dan Mihai	Fundamental Algorithms	10/11/2021	Cristea Paul
4	Pavel Traian	Databases	08/10/2021	Mihai Horatiu
5	Irimie Dan	Databases	12/10/2021	Mihai Horatiu

- **insertion anomalies**: cannot be specified a *Professor* for a course *X*, unless there is at least one student having a meeting date to the course *X*

Functional Dependencies

- **update anomalies**: if the *Professor* will be changed for the course X, also, should be performed changes in all the associations with the course *X* (it is not known how many associations exist), otherwise the database will have errors (it will be inconsistent) (e.g. if the *Professor* is changed in the 1st record, but not changed in the 4th and 5th records, the operation will finish with an error in the relation (incorrect data)
- **deletion anomalies**: when some of the records are deleted, also, can be deleted data that is not intended to be removed (e.g. if records 1, 4 and 5 are deleted, the association between the *Course* and the *Professor* is also removed from the database)

Meeting	StudentName	Course	MeetingDate	Professor
1	Rus Maria	Databases	01/10/2021	Mihai Horatiu
2	Irimie Dan	Fundamental Algorithms	11/10/2021	Cristea Paul
3	Dan Mihai	Fundamental Algorithms	10/11/2021	Cristea Paul
4	Pavel Traian	Databases	08/10/2021	Mihai Horatiu
5	Irimie Dan	Databases	12/10/2021	Mihai Horatiu

Functional Dependencies

- **wasting space, insert anomalies, update anomalies, deletion anomalies** are caused by the functional dependencies among the sets of attributes
- To avoid these problems, the dependencies among values should be kept in separate relations; the previous relation should be decomposed in a good way, such that the data should not be lost or added
- ***A good decomposition should be done in the database design phase, when the functional dependencies can be identified.***

Student

StudentName
Rus Maria
Irimie Dan
Dan Mihai
Pavel Traian

Course

Course
Databases
Fundamental Algorithms

Professor

Professor
Mihai Horatiu
Cristea Paul

Meeting

Meeting	StudentName	Course	MeetingDate	Professor
1	Rus Maria	Databases	01/10/2021	Mihai Horatiu
2	Irimie Dan	Fundamental Algorithms	11/10/2021	Cristea Paul
3	Dan Mihai	Fundamental Algorithms	10/11/2021	Cristea Paul
4	Pavel Traian	Databases	08/10/2021	Mihai Horatiu
5	Irimie Dan	Databases	12/10/2021	Mihai Horatiu

Functional Dependencies

Let r be the instance of a relation R

- ***r satisfies the functional dependency*** $\boldsymbol{\alpha \rightarrow \beta}$ if for each pair of tuples t_1 and t_2 from r such that $\prod_\alpha t_1 = \prod_\alpha t_2$ is also satisfied $\prod_\beta t_1 = \prod_\beta t_2$

or

$$\forall t_1, t_2 \in r: \prod_\alpha t_1 = \prod_\alpha t_2 \Rightarrow \prod_\beta t_1 = \prod_\beta t_2$$

where $\prod_\alpha t$ is the projection of the $\boldsymbol{\alpha}$ atributes for the tuple t

- A ***functional dependency f is satisfied on R*** if and only if any r instance of R satisfies f
- r ***does not respect*** *a functional dependency f, if r does not satisfies f*
- *r is* **a legal instance of R** *if r satisfies all the functional dependencies defined on R*

Functional Dependencies

Example 3: Student[Hobby, StudentName, StudentSurname]

Hobby	StudentName	StudentSurname
Singing	Rus	Maria
Playing games	Irimie	Dan
Swiming	Mihai	Maria
Running	Rus	Maria
Taking pictures	Irimie	Dan

- The functional dependency StudentSurname → StudentName is not fulfill by the relation ***Student***
- r satisfies the functional dependency StudentName → StudentSurname

This last one, does not means that StudentSurname → StudentName is fulfilled in the relation ***Student***

Functional Dependencies

Is a functional dependency ***f*** fulfilled on R based on a set of functional dependencies ***F***?

Example 1: In ***Discussion*** table there is

- F={ StudentName → Age
    ProfesssorName → PEmail
    ProfesorName, Day, StartHour, EndHour → StudentName }

- o *Day, StartHour, EndHour → Age* is fulfilled?
- o *ProfesorName, Day, StartHour, EndHour → Age* is fulfilled?

F logical implies f (denoted $F \Rightarrow f$) if each r instance of the relation R that satisfies F also satisfies f

F and G are sets of functional dependencies and f is a functional dependency.
F logical implies G (denoted $F \Rightarrow G$) if $F \Rightarrow$g for each $g \in G$

Closure of F (denoted F^+) is the set of all functional dependencies implied by F

$$\mathrm{F}^{+} = \{\mathrm{f} \mid \mathrm{F} \Rightarrow \mathrm{f}\}$$

F and G are ***equivalent*** (denoted $F \equiv G$) if $\mathrm{F}^{+} = G^{+}$ (i.e. $F \Rightarrow G$ and $\mathrm{G} \Rightarrow F$)

Functional Dependencies

Armstrong Axioma's

Let $\alpha, \beta, \gamma \subseteq R$

- ***Reflexivity***: if $\beta \subseteq \alpha$ then $\alpha \rightarrow \beta$
- ***Augmentation***: if $\alpha \rightarrow \beta$ then $\alpha\gamma \rightarrow \beta\gamma$
- ***Transitivity***: if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$ then $\alpha \rightarrow \gamma$

The Armstrong Axioma's system is

- ***Correct*** (any derivative functional dependency is implied by F)
- ***Complete*** (all functional dependencies from F^+ can be derived)

Example: Let consider the relation R[A, B, C, D, E] with the set F={A→C; B→C; CD→E}.
Prove that F⇒AD→E.
Solution: A→C (given) becomes AD→CD (from augmentation).
CD→E (given) becomes AD→E (from transitivity with the upper implication)

Functional Dependencies

Properties for functional dependencies:

- If C is a key of R[A_1, A_2, …, A_n] then C→$\boldsymbol{\beta}$, ∀$\boldsymbol{\beta}$ a subset of {A_1, A_2, …, A_n}.
  - such a dependency is always true, hence it will not be eliminated through decomposition
- If $\beta \subseteq \alpha$ then $\alpha \to \beta$ ***trivial functional dependency (reflexivity)***

$$\prod_{\alpha}(t_1) = \prod_{\alpha}(t_2) \underset{\beta\subseteq\alpha}{\Longrightarrow} \prod_{\beta}(t_1) = \prod_{\beta}(t_2) \Rightarrow \alpha \to \beta$$

where $\prod_{\alpha} t$ is the projection of the $\boldsymbol{\alpha}$ atributes for the tuple t

- If $\alpha \to \beta$ and $\beta \to \gamma$ then $\alpha \to \gamma$ ***transitivity***

$$\prod_{\alpha}(t_1) = \prod_{\alpha}(t_2) \underset{\alpha\to\beta}{\Longrightarrow} \prod_{\beta}(t_1) = \prod_{\beta}(t_2) \underset{\beta\to\gamma}{\Longrightarrow} \prod_{\gamma}(t_1) = \prod_{\gamma}(t_2) \Rightarrow \alpha \to \gamma$$

- The ***projection of the set F on*** $\boldsymbol{\alpha}$, denoted F_α, is the set of those dependencies from F^+ that implies only attributes from α, i.e.

$$F_\alpha = \{\beta \to \gamma \in F^+ \mid \beta\gamma \subseteq \alpha\}$$

Functional Dependencies

Additional inference rules

- **Reunion**: *if* $\alpha \rightarrow \beta$ and $\alpha \rightarrow \gamma$ then $\alpha \rightarrow \beta\gamma$
- **Decomposition**: if $\alpha \rightarrow \beta$ then $\alpha \rightarrow \beta'$ for any $\beta' \subseteq \beta$

Example: Show that {A→BCD} ≡ {A→B; A→C; A→D}.
Solution: Let F = {A→BCD} and G = {A→B; A→C; A→D}
From the decomposition rule follows: F⇒ A→B, F⇒A→C and F⇒A→D. So, F⇒G.
From the reunion follows

{A→B; A→C} ⇒ A→BC and {A→BC; A→D} ⇒ A→BCD

So, G⇒F and consequently, F ≡ G

Functional Dependencies

Definition:

Let R[A_1, A_2, …, A_n] be a relation and let **α**, **β** be two subsets of attributes of R. Attribute **β** is ***fully functionally dependent on α if***

- **β** is functionally dependent on **α** (i.e. (that is) $\alpha \to \beta$))

and

- **β** is not functionally dependent on any proper subset of **α** (i.e. (that is) $\forall \gamma \subset \alpha, \gamma \to \beta$ is not true)

Definition:

A set of α attributes is a ***super-key*** of a relation R (having the set of functional dependencies F) if $F \Rightarrow \alpha \to R$

Definition:

A set of α attributes is a ***key*** of a relation R if

- α is a super-key
- no subset of α is a super-key (i.e. $\beta \subset \alpha, \beta \to R \notin F^+$)

Definition:

An attribute A ∈ R (simple or composite) is called ***prime*** if there is a key C (simple or composite) and $A \subseteq C$ (A can itself be a key). If an attribute in not included in any key, it is called ***non-prime***

Functional Dependencies

Example 1: In the relation ***Discussion***[StudentName, Age, ProfessorName, PEmail, Day, StartHour, EndHour] with the functional dependencies

- StudentName → Age
- ProfessorName → PEmail
- ProfessorName, Day, StartHour, EndHour → StudentName

- {ProfessorName, Day, StartHour, EndHour} is the only **key** of the relation ***Discussion***
- ProfessorName, Day, StartHour and EndHour are the only **prime** attributes of the relation ***Discussion***
- Any set that include {ProfessorName, Day, StartHour, EndHour} is **super-key** of the relation **Discussion**

The closing of the attributes

Let $\alpha \subseteq R$ and F a set of (fulfilled) functional dependencies on R

The ***closing of*** $\boldsymbol{\alpha}$ (with respect to a set F of functional dependencies), denoted α^+, is the set of attributes that are functional dependent from α on the base of the functional dependencies from F, i.e.

$$\alpha^+ = \{A \in R \mid F \Rightarrow \alpha \rightarrow A\}$$

It can be seen like $F \Rightarrow \alpha \rightarrow \beta$ if and only if $\beta \subseteq \alpha^+$ (by respecting the functional dependencies from F)

Functional Dependencies

Decomposition of the relations

The ***decomposition of a relation R*** is a set of (sub)relations {R_1, R_2, …, R_n} such that each $R_i \subseteq R$ and $R = \cup R_i$

If r is an instance from R then r can be decomposed in {r_1, r_2, …, r_n}, where each $r_i = \prod_{R_i}(r)$

Example 1: The relation ***Discussion***[StudentName, Age, ProfessorName, PEmail, Day, StartHour, EndHour] can be decomposed

{ R_1= (StudentName, Age),
R_2= (ProfessorName, PEmail)
R_3= (ProfessorName, Day, StartHour, EndHour)
R_4= (Day, StartHour, EndHour, StudentName) }

Decomposition of the relations - Properties

- The decomposition must keep ***the information***
  - The data from the initial relation = The data from all decomposed relations
  - Mandatory for keeping the consistency of the data
- The decomposition must respect ***all the functional dependencies***
  - The functional dependencies from the initial relation = The reunion of the functional dependencies from the decomposed relations
  - Facilitate the check of violations of the functional dependencies

Functional Dependencies

Decomposition of the relations

- The decomposing must ***keep the information*** ⟺ can be reconstruct r from linking up al it's projections $\{r_1, r_2, \ldots, r_n\}$
- If $\{R_1, R_2, \ldots, R_n\}$ is a decomposing of R then for any r instance from R, there is

$$r \subseteq \prod_{R_1}(r) * \prod_{R_2}(r) * \ldots * \prod_{R_n}(r)$$

Example 1: ***Discussion***[StudentName, Age, ProfessorName, PEmail, Day, StartHour, EndHour] – the initial table

R_1

StudentName	Age
Rus	19
Irimie	20
Dan	21
Pavel	19

R_2

ProfessorName	PEmail
Mihai	d@sd.com
Cristea	cp@nk.ro

R_3

Professor Name	Day	StartHour	EndHour
Mihai	Monday	12:00	12:50
Cristea	Tuesday	15:30	16:20
Mihai	Monday	13:00	13:30
Cristea	Friday	15:30	16:20

R_4

Day	StartHour	EndHour	StudentName
Monday	12:00	12:50	Rus
Tuesday	15:30	16:20	Irimie
Monday	12:00	12:50	Dan
Monday	13:00	13:30	Pavel
Friday	15:30	16:20	Irimie

Functional Dependencies

Decomposition of the relations

$$R_1 * R_2 * R_3 * R_4$$

(from R_2 *to* R_3 *and to* R_4)

ProfessorName	PEmail	StudentName	SAge	Day	StartHour	EndHour
Mihai	d@sd.com	Rus	19	Monday	12:00	12:50
Mihai	d@sd.com	Dan	21	Monday	12:00	12:50
Cristea	cp@nk.ro	Irimie	20	Tuesday	15:30	16:20
Mihai	d@sd.com	Pavel	19	Monday	13:00	13:30
Cristea	cp@nk.ro	Irimie	20	Friday	15:30	16:20

When the projection is computed, may ***arise new rows***, rows that have not existed in the initial relation. But here, is not the case. ***So, this decomposition is good.***

The decomposition $\{R_1, R_2, \ldots, R_n\}$ of the relation R is with ***keep the dependencies*** if
$\left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right) \Rightarrow F$ and $F \Rightarrow \left(F_{R_1} \cup F_{R_2} \cup \cdots \cup F_{R_n}\right)$

Functional Dependencies

Decomposition of the relations: not a good example:

Example 1prime: ***Discussion***[StudentName, Age, ProfessorName, PEmail, StartHour]

StudentName	SAge	ProfessorName	PEmail	StartHour
Rus	19	Mihai	d@sd.com	12:00
Irimie	20	Cristea	cp@nk.ro	15:30
Dan	21	Mihai	d@sd.com	16:00
Pavel	19	Mihai	d@sd.com	13:00
Irimie	20	Cristea	cp@nk.ro	12:00

Decomposed as
{ R_1= (StudentName, Age),
R_2= (ProfessorName, PEmail)
R_3= (ProfessorName, StartHour)
R_4= (StartHour, StudentName) }

R_1

StudentName	Age
Rus	19
Irimie	20
Dan	21
Pavel	19

R_2

ProfessorName	PEmail
MIhai	d@sd.com
Cristea	cp@nk.ro

R_3

Professor Name	StartHour
Mihai	12:00
Cristea	15:30
Mihai	16:00
Mihai	13:00
Cristea	12:00

R_4

StartHour	StudentName
12:00	Rus
15:30	Irimie
16:00	Dan
13:00	Pavel
12:00	Irimie

Functional Dependencies

Decomposition of the relations: not a good example

$$R_1 * R_2 * R_3 * R_4$$

(from R_2 to R_3 and to R_4)

ProfessorName	PEmail	StudentName	SAge	StartHour
Mihai	d@sd.com	Rus	19	12:00
Mihai	d@sd.com	Irimie	20	12:00
Cristea	cp@nk.ro	Irimie	20	15:30
Mihai	d@sd.com	Dan	21	16:00
Mihai	d@sd.com	Pavel	19	13:00
Cristea	cp@nk.ro	Rus	19	12:00
Cristea	cp@nk.ro	Irimie	20	12:00

So, there are 2 records that have not been in the initial instance.
The projection of the decomposition is not generating the same tuples that have been before in the instance
So, this decomposition is not good.

Functional Dependencies

Decomposition of the relations

Definition:

- The ***projection operator*** is used to decompose a relation

Let [R_1, R_2, ..., R_n] be a relation and $\alpha = \{A_{i_1}, A_{i_2}, \dots, A_{i_m}\}$ a subset of attributes, $\alpha \subset \{A_{i_1}, A_{i_2}, \dots, A_{i_m}\}$.

- The ***projection of relation R on α*** is

$$R'[A_{i_1}, A_{i_2}, \dots, A_{i_m}] = \prod\nolimits_{\alpha}(R) = \prod\nolimits_{\{A_{i_1}, A_{i_2}, \dots, A_{i_m}\}}(R) = \{r[a] \mid r \in R\}$$

where $\forall r = (a_1, a_2, \dots, a_n) \in R \;\Rightarrow \prod_{\alpha}(R) = r[a] = (a_{i_1}, a_{i_2}, \dots, a_{i_m}) \in R'$ and all elements in R' are distinct

Definition:

The ***natural join operator*** is used to compose relations.

Let $R[\alpha, \beta]$, $S[\beta, \gamma]$ be two relations over the specified sets of attributes, $\alpha \cap \gamma = \emptyset$.

- The ***natural join of relations R and S*** is the relation

$$R * S[\alpha, \beta, \gamma] = \left\{\left(\prod_{\alpha}(r), \prod_{\beta}(r), \prod_{\gamma}(r)\right) \mid r \in R, s \in S \; and \prod_{\beta}(r) = \prod_{\beta}(s)\right\}$$

A relation R can be decomposed into multiple new relations $R_1, R_2, \dots, R_m$

- The ***decomposition is good*** if $R = R_1 * R_2 * \dots * R_m$

i.e. R' s data can be obtained from the data stored in relations $R_1, R_2, \dots, R_m$ (no data is added / lost through decomposition / composition

Functional Dependencies

Lossless – Join Decomposition

Definition:

- A decomposition of R (that have the functional dependencies F) in $R_1, R_2, \ldots, R_n$ is a ***lossless – join decomposition with respect to the set F*** if

$$\prod_{R_1}(r) * \prod_{R_2}(r) * \ldots * \prod_{R_n}(r) = r$$

for any instance r from R that satisfies F

Example: Decomposition of R[A, B, C] in $\{R_1(AC), R_2(BC)\}$

r

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c

r_1

A	C
a_1	c
a_2	c

r_2

B	C
b_1	c
b_2	c

$r_1 * r_2$

A	B	C
a_1	b_1	c
a_1	b_2	c
a_2	b_1	c
a_2	b_2	c

- Because $r \subset r_1 * r_2$ this decomposition is **not lossy - join decomposition**

Functional Dependencies

Lossless – Join Decomposition

Theorem:

The decomposition of R (with the set F of functional dependencies) in $\{R_1, R_2\}$ is lossless – join decomposition with respect with the set F if and only if

- $F \Rightarrow R_1 \cap R_2 \rightarrow R_1$ or
- $F \Rightarrow R_1 \cap R_2 \rightarrow R_2$

Corollary:

If $\alpha \rightarrow \beta$ is fulfilled on R and $\alpha \cap \beta = \emptyset$ then the decomposition of R in $\{R - \beta, \alpha\beta\}$ is a lossless – join decomposition

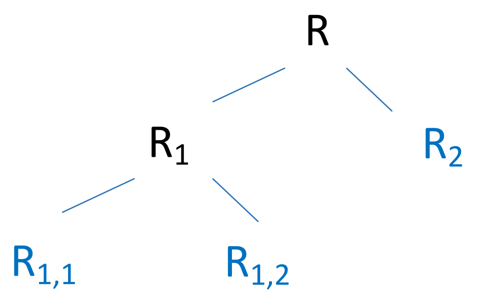
Theorem:

If

$\{R_1, R_2\}$ is a lossless – join decomposition of R, and

$\{R_{1,1}, R_{1,2}\}$ is a lossless – join decomposition of R_1, then

$\{R_{1,1}, R_{1,2}, R_2\}$ is a lossless – join decomposition of R:



Functional Dependencies

Lossless – Join Decomposition example

Let R[A, B, C] with the functional dependencies F = {A→B}

- {AB, AC} is a lossless – join decomposition because AB∩AC=A and A→AB

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

r_1

A	B
a_1	b_1
a_2	b_1

r_2

A	C
a_1	c_1
a_2	c_2

$r_1 * r_2$

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

- {AB, BC} is NOT a lossless – join decomposition because AB∩BC=B and none of the following dependencies B→AB or B→BC are fulfilled by R

r

A	B	C
a_1	b_1	c_1
a_2	b_1	c_2

r_1

A	B
a_1	b_1
a_2	b_1

r_2

B	C
b_1	c_1
b_1	c_2

$r_1 * r_2$

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_2

Functional Dependencies

Lossless – Join Decomposition – example 1

Discussion

StudentName	SAge	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	19	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	20	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	21	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	19	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	20	Cristea	cp@nk.ro	Friday	15:30	16:20

Student

StudentName	SAge
Rus	19
Irimie	20
Dan	21
Pavel	19

ProfessorDiscussion

StudentName	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	Cristea	cp@nk.ro	Friday	15:30	16:20

Functional Dependencies

Lossless – Join Decomposition – example 1

Student

StudentName	SAge
Rus	19
Irimie	20
Dan	21
Pavel	19

ProfessorDiscussion

StudentName	ProfessorName	PEmail	Day	StartHour	EndHour
Rus	Mihai	d@sd.com	Monday	12:00	12:50
Irimie	Cristea	cp@nk.ro	Tuesday	15:30	16:20
Dan	Mihai	d@sd.com	Monday	12:00	12:50
Pavel	Mihai	d@sd.com	Monday	13:00	13:30
Irimie	Cristea	cp@nk.ro	Friday	15:30	16:20

Professor

ProfessorName	PEmail
Mihai	d@sd.com
Cristea	cp@nk.ro

PSDiscussion

StudentName	ProfessorName	Day	StartHour	EndHour
Rus	Mihai	Monday	12:00	12:50
Irimie	Cristea	Tuesday	15:30	16:20
Dan	Mihai	Monday	12:00	12:50
Pavel	Mihai	Monday	13:00	13:30
Irimie	Cristea	Friday	15:30	16:20

Functional Dependencies

Example 4 for functional dependencies

Consider the relation ***Student[StudentId, Name, ExamType]*** having the ***StudentId*** as ***key***

For the following instance

StudentId	Name	ExamType
12	Popescu Dan	Written
14	Irimie Raul	Practical
23	Jurca Bogdan	Practical
45	Hora Mihaela	Written
56	Florea Paula	Practical
79	Damian Crina	Practical

- the functional dependency {Name} → {StudentId} is fulfilled (because, there are no 2 different tuples with the same value in *Name* and different values in *StudentId*)
- the functional dependency {StudentId} → {ExamType} is fulfilled (because, there are no 2 different tuples with the same value in *StudentId* and different values in *ExamType*)

Functional Dependencies

Example 5 for functional dependencies

Consider the relation ***Exam[StudentId, ProfessorId, Grade, ExamType]*** having the (***StudentId, ProfessorId)*** as ***key***

For the following instance

StudentId	ProfessorId	Grade	ExamType
12	13	10	Written
14	13	9.7	Practical
23	14	8.6	Quiz
45	23	9.2	Combination
56	5	7.25	Written
79	14	8.75	Presentation

- the functional dependency {StudentId} → {Grade} is fulfilled (because, there are no 2 different tuples with the same value in *StudentId* and different values in *Grade*)
- the functional dependency {ProfessortId} → {ExamType} is not fulfilled (because of pair (1,2) or (3,6))
- the functional dependency {StudentId, Grade} → {ExamType} is fulfilled (because, there are no 2 different tuples with the same value in (*StudentId, Grade)* and different values in *ExamType*)

References:

- C.J. Date, *An Introduction to Database Systems (8th Edition)*, Addison-Wesley, 2003.
- H. Garcia-Molina, J. Ullman, J. Widom, *Database Systems: The Complete Book, Prentice Hall Press*, 2008.
- G. Hansen, J. Hansen, *Database Management And Design (2nd Edition),* Prentice Hall, 1996.
- R. Ramakrishnan, J. Gehrke, *Database Management Systems*, McGraw- Hill, 2007. http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/slides/slides3ed.html
- R. Ramakrishnan, J. Gehrke, *Database Management Systems (2nd Edition),* McGraw-Hill, 2000.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill, 2010. http://codex.cs.yale.edu/avi/db-book/
- L. Țâmbulea, *Curs Baze de date*, Facultatea de Matematică și Informatică, UBB, 2013-2014.
- J. Ullman, J. Widom, *A First Course in Database Systems*, http://infolab.stanford.edu/~ullman/fcdb.html