Lecture 6

Normal Forms

Normal Forms

- A data collection can be represented in a relational model (by using tables) in multiple ways

- The database design should be carried out smoothly taking into account the subsequent queries and operations that will be performed, i.e.
  - no additional tests should be required when the data is changed
  - the operations should be performed through SQL statements alone

- If the relations from the database satisfy some conditions, the database will be managed properly (i.e. the relations should be in a certain normal form)

- **Redundancy** is the main cause of the problems that may arise and have a negative impact in the structure of the relational databases: wasting space, insert anomalies, update anomalies, deletion anomalies

Normal Forms

- A possible solution for redundancy can be the replacement of a relation with a collection of smaller relations (these ones will contain a strict subset of attributes from the original relation)
- Ideally, only redundancy-free schemas should be allowed
- Should be, at least identified the schemas with redundancy, even if such schemas are allowed (e.g. for performance reasons)

- The *functional dependencies* can be used to identify the projection problems and also to suggest improvements
- For example, let R be a relation with 3 attributes A, B and C.
    - **No functional dependencies**: no redundance
    - For the dependency $A \to B$: more records can have the same value for A, case in which there are unique values for B

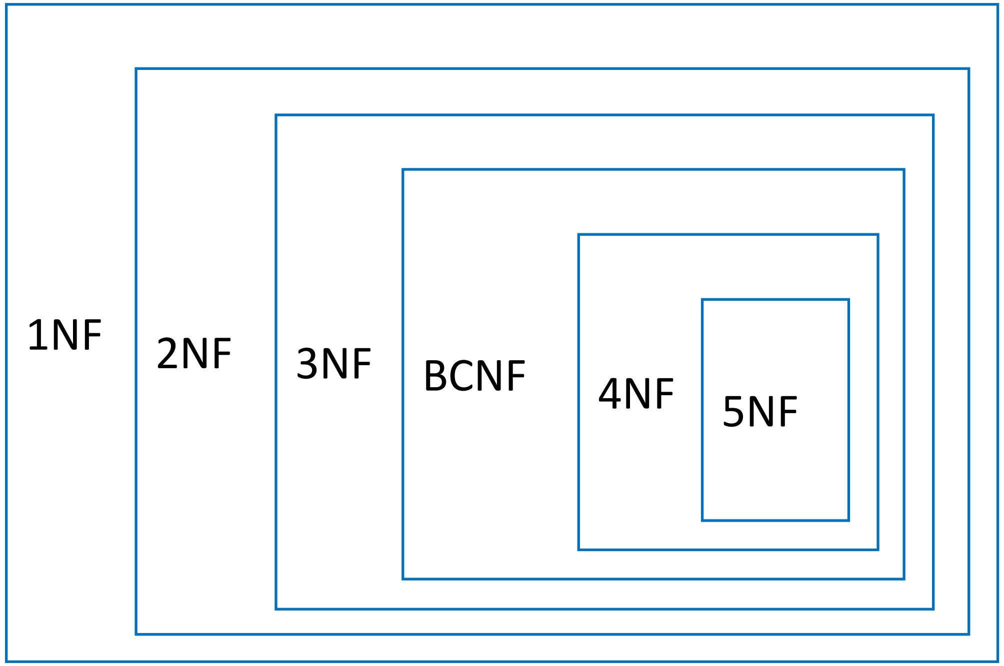
- The decompose should be used only when it is necessarily (e.g. of a problem: a part of the data can be lost)

Normal Forms

- If a relation is in a particular ***normal form*** then some of the problems are eliminated / minimized, and helps in deciding if the decomposing of a relation is necessarily or not
- The ***normal forms*** (based on the functional dependencies) are
    - First normal form (1NF)
    - Second normal form (2NF)
    - Third normal form (3NF)
    - Boyce-Codd normal form (BCNF)
    - Forth normal form (4NF)
    - Fifth normal form (5NF)

{ BCNF ⊆ 3NF, 3NF ⊆ 2NF, 2NF ⊆1NF }

- 1NF, 2NF, 3NF were defined by Codd
- BCNF was defined by Boyce and Codd
- 4NF, 5NF were defined by Fagin



Normal Forms

- If a relation is not in normal form X, it can be decomposed into multiple relations that are in formal form X
- There are: ***simple attributes*** and ***composite attributes*** = a set of attributes in a relation (at least 2 attributes included)

Definition:

Repeating attributes = the attributes (simple or composite) that can take multiple values for a record in the relation

- When a relation is defined in the relational model, the attribute values must be scalar, atomic (one value); otherwise, they cannot be further decomposed

Example 6: Consider the relation ***Student[StudentName, Age, Course, Grade]***

StudentName	Age	Course	Grade
Rus	19	Algebra Databases Geometry	8.90 10 7.50
Hora	20	Functional Programming Databases Functional Analysis	8.75 9.75 8

Key: StudentName

Composite repeating attribute: pair (Course, Grade)

Normal Forms

Decompose this relation ***Student[StudentName, Age, Course, Grade]***:

- StudentDetail[StudentName, Age]
- Exam[StudentName, Course, Grade]

StudentDetail

StudentName	Age
Rus	19
Hora	20

Exam

StudentName	Course	Grade
Rus	Algebra	8.90
Rus	Databases	10
Rus	Geometry	7.50
Hora	Functional Programming	8.75
Hora	Databases	9.75
Hora	Functional Analysis	8

Normal Forms

- The repeating attributes cannot be used in the relational model; they should be avoided but without lossing the data

Let *R[A]* a relation and *A* a set of attributes

- α is a repeating attribute in *R* (simple or composite)

R can be **decomposed** into 2 relations, so that α not be any longer a repeating attribute

If *C* is a key in *R* then the relation *R* ca be decomposed in the following two relations:

  - $R'[C \cup \alpha] = \prod_{C \cup \alpha}(R)$
  - $R''[A - \alpha] = \prod_{A - \alpha}(R)$

Normal Forms

First Normal Form (1NF)

- A relation is in the ***first normal form*** (***1NF***) if it does not have repeating attributes

or

- A relation is in the ***first normal form*** (***1NF***) if each attribute of the relation can have only atomic values (the lists and the sets are excluded)

- Due to the definition of the relational model, this condition of 1NF is by default.

Second Normal Form (2NF)

- A relation is in the ***second normal form*** (***2NF***) if
  - it is in the first normal form (1NF)

  and
  - every (simple or composite) non-prime attribute is fully functionally dependent on every key in the relation

or

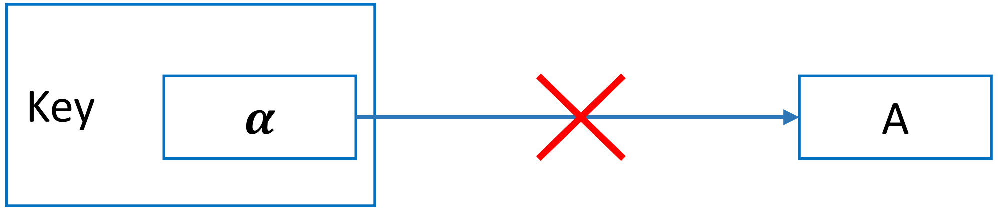
- A relation is in the ***second normal form*** (***2NF***) if it is in the first normal form (1NF) and does not have partial functional dependencies

Normal Forms

Definition:

A relation has a ***partial functional dependency*** when a non-prime attribute is functional dependent with a part of the primary key of the relation (but not with the entire key).

- R is a relation in the 1NF, but not in 2NF. Then R has a composite key (and also a functional dependency $\alpha \rightarrow \beta$, where α (simple or composite) is a proper subset of a key and β is a non-prime attribute



Partial dependencies (A is not in a key)

Decomposition: Let *R[A]* a relation, *A* a set of attributes and *C* a key

- β non-prime, β functionally dependent on α, $\alpha \subset C$ (β is functionally dependent on a proper subset of attributes from a key)
- The dependency $\alpha \rightarrow \beta$ ca be eliminated if *R* is **decomposed** into 2 relations,
  - $R'[\alpha \cup \beta] = \prod_{\alpha \cup \beta}(R)$
  - $R''[A - \beta] = \prod_{A-\beta}(R)$

Normal Forms

Example 2: ***Meeting [StudentName, Course, MeetingDate, Professor]***

- key: {StudentName, Course}
- the functional dependency {Course} → {Professor} holds ⇒ the attribute *Professor* is not fully functionally dependent on a key, and so, relation *Meeting* is ***not in 2NF***

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

- The dependency is eliminated - if *Meeting* relation is decomposed into the following two relations
  - Interaction[StudentName, Course, MeetingDate]
  - Course[Course, Professor]

Normal Forms

Example 7: ***Teaching[<u>ProfessorId, CourseId</u>, ProfessorName, Title, CourseName]***

- key: {ProfessorId, CourseId}
- functional dependencies {ProfessorId} → {ProfessorName, Title}, {CourseId} → {CourseName}

- By eliminating these two dependencies, the relation is decomposed into the following three relations
  - Professor[<u>ProfessorId</u>, ProfessorName, Title]
  - Course[<u>CourseId</u>, CourseName]
  - ProfessorCourses[<u>ProfessorId, CourseId</u>]

Normal Forms

The *transitive dependency* is need for the 3NF

Definition:

An attribute Z is ***transitively dependent*** on an attribute X if $\exists Y$ such that $X \rightarrow Y,\ Y \rightarrow Z, Y \rightarrow X$ does not hold (and Z is not in X or Y).

Third Normal Form (3NF)

- A relation is in the ***third normal form*** (***3NF***) if it is in the second normal form (2NF) and none non-prime attribute is transitively dependent on any key in the relation

or

- A relation is in the ***third normal form*** (***3NF***) if for every non-trivial functional dependency $X \rightarrow A$ that holds over R
  - *X* is a super-key

  or
  - *A is a prime attribute*

or

Normal Forms

- A relation *R* that satisfies the functional dependencies *F* is in the ***third normal form*** (***3NF***) if for all $\alpha \rightarrow A$ from F^+
  - $A \in \alpha$ (trivial functional dependency)

  or
  - α contains a key of *R*

  or
  - *A* is a prime attribute

A is a key

Normal Forms

Boyce-Codd Normal Form (BCNF)

- A relation is in the ***Boyce-Codd normal form*** (***BCNF***) if every determinant (for a functional dependency) is a key (informal definition - simplifying assumption: determinants are not too big, only the non-trivial functional dependencies are considered).

or

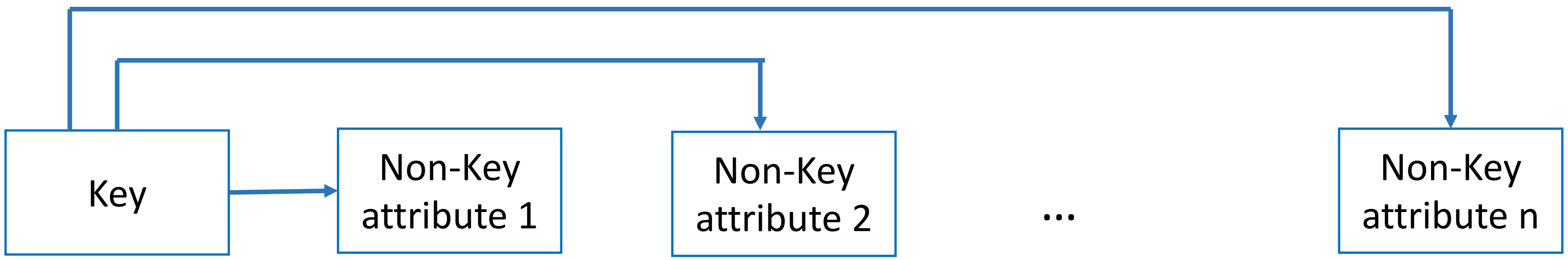
- A relation that satisfies the functional dependencies F is in the ***Boyce-Codd normal form*** (***BCNF***) if for all $\alpha \rightarrow A$ from F^+
  - $A \in \alpha$ (the trivial functional dependency)
  - α contains a key of the relation R

A is not a key

Normal Forms

BCNF and 3NF

- If relation R is in ***BCNF*** obviously it is also in **3NF**
- If relation *R* is in ***3NF*** it is possible to appear some redundancies (it is a compromise, and it is used when BCNF cannot be fulfilled)



Example 3NF, no BCNF: Let the relation Meeting[Course, Professor, MeetTime] with the functional dependencies {Course→Professor; Professor, MeetTime→Course}

- Key: {Course, MeetTime} and {Professor, MeetTime}
- Meeting is in 3NF, but not BCNF

Normal Forms

Example 7: Consider the relation ***Communication[<u>StudentName</u>, MeetingDate, Professor, Department]***, in which is stored the professor with the department in which he / she works

- key: {StudentName} (because the relation contains data about the students (i.e. one row per student))
- the functional dependency {Professor} → {Department} holds ⇒ the relation is ***not in 3NF***
- The dependency is eliminated - if *Communication* relation is decomposed into the following two relations
  - Talk[<u>StudentName</u>, MeetingDate, Professor]
  - ProfessorAsigned[<u>Professor</u>, Department]

Example 8: Consider the relation ***Apartment[<u>StudentId</u>, Name, ZipCode, City, Street, Number]***, in which is stored the address for a group of students

- key: {StudentId}
- the functional dependency {ZipCode} → {City, Street} holds ⇒ the relation is ***not in 3NF***. Any other functional dependencies?
- The dependency is eliminated - if *Appartment* relation is decomposed into the following two relations
  - Address[<u>StudentId</u>, Name, ZipCode, Number]
  - CityZipCode[<u>ZipCode</u>, City, Street]

Normal Forms

Example 9: Consider the relation ***Consultancy[CDate, CHour, Professor, Class, Group]***, in which are stored the meetings between professors and students

- The restrictions given will provide the key and the functional dependencies:
  - a consultancy between a professor and a group of students is at most one per day ⇒ {CDate, Group} key
  - on a certain date and hour, a professor has at most one consultancy ⇒ {Professor, CDate, CHour} key
  - on a certain date and hour, there is at most one consultancy in the class ⇒ {Class, CDate, CHour} key
  - a professor does not change a class in a day ⇒ {Professor, CDate} → {Class} functional dependency
- All the attributes appear in at least one key, and so, there are no non-prime attributes
- So, the **relation is in 3NF** (from the previous given definitions)

- Objective: to eliminate the functional dependency {Professor, CDate} → {Class}
- By eliminating the functional dependency {Professor, CDate} → {Class}, the initial relation is decomposed in
  - Meeting[CDate, CHour, Professor, Group]
  - ProfessorClass[Professor, CDate, Class]
- These 2 relations does not contain other functional dependencies, so, they **are in BCNF**
- On the other hand, the key {Class, CDate, CHour} does not appear anywhere, so it has to be checked in other manners (e.g. through program / code)

Normal Forms

Normal forms based on functional dependencies

- 1NF – all the values of the attributes are atomic (have one value)
- 2NF – all the non-key attributes depend on the entire key (no partial dependencies)
- 3NF – the tables in 2NF and all the non-prime attributes depend only on the key (no transitive dependencies)
- BCNF – all dependencies are data keys

- Each attribute depends
  - on the ***key*** (→ key definition)
  - on the ***entire key*** (→ 2NF)
  - and nothing else, ***except the key*** (→ BCNF)

- Each ***non-prime*** attribute depends
  - on the ***key*** (→ key definition)
  - on the ***entire key*** (→ 2NF)
  - and nothing else, ***except the key*** (→ 3NF)

Normal Forms

Examples in which the normal forms are **not** respected:

- ***2NF*** – all the non-prime attributes must depend on the ***entire*** key
  - Meeting[StudentName, Course, Professor, Grade]
- ***3NF*** – all the non-prime attributes must depend ***only*** on the key
  - Consultancy[StudentName, Title, Professor, Department]
- ***BCNF*** – all the functional dependencies are implied by the candidate key
  - Discussion[Professor, Day, StartHour, EndHour, StudentName]

Decomposition in BCNF

Let R be a relation with the functional dependencies F

- If $\alpha \rightarrow \beta$ does not respect BCNF then R can be decomposed in R – β and $\alpha\beta$

Normal Forms

Decomposition in BCNF Example:

Let R[C, B, A, D, P, F, M] be a relation with key C and {AP→C; BD→P; A→B}

- For BD→P can be performed the decomposition
  - [B, D, P]
  - [C, B, A, D, F, M]
- Then, for A→B can be performed the decomposition for [C, B, A, D, F, M], obtaining
  - [A, B]
  - [C, A, D, F, M]

- In general, multiple dependencies may cause that BCNF will not be fulfilled
- The order in which it is treat it may arise different relations in the decomposition

- In general, the decomposition in BCNF does not keep the dependencies

Example: R[C, B, A, D, P, F, M] in [B, D, P], [A, B] and [C, B, A, D, F, M] does not keep the initial dependencies {AP→C; BD→P; A→B}

- On BCNF can appear also redundancy

Normal Forms

Example:

- Let $\alpha \rightarrow A$ a functional dependency from F that not respect BCNF
- The decomposition of R in $R_1 = \alpha A$ and $R_2 = R - A$
- If R_1 or R_2 are not in BCNF, the decomposition is going to be continued

[StudentName, SAge, ProfessorName, PEmail, Day, StartHour, EndHour]

[ProfessorName, PEmail, StudentName, Day, StartHour, EndHour]   [StudentName, SAge]

[ProfessorName, PEmail]   [ProfessorName, StudentName, Day, StartHour, EndHour]

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

Normal Forms

Decomposition in 3NF

- The same strategy as for BCNF
- Can be kept the dependencies?
  - If $X \to Y$ cannot be kept then *XY* is added
  - The problem with *XY* is the fact that not all the time respect 3NF
    - e.g. Let CAP be added to keep $AP \to C$, but if also $A \to C$ is fulfilled, then is not correct
- Solution: instead of using the initial set F, can be used a ***minimal cover of F***

Definition:

- An ***attribute*** $\boldsymbol{A} \in \boldsymbol{\alpha}$ ***is redundant in the functional dependency*** $\boldsymbol{\alpha} \to \boldsymbol{\beta}$ if

$$(F - \{\alpha \to \beta\}) \cup \{\alpha - \{A \to \beta\}\} \equiv F$$

  - To check if $A \in \alpha$ is redundant in $\alpha \to \beta$ calculate $(\alpha - A)^+$
  - Then $A \in \alpha$ is redundant in $\alpha \to \beta$ if $B \in (\alpha - A)^+$

Definition:

- A functional dependency $\boldsymbol{f} \in \boldsymbol{F}$ ***is redundant*** if $F - \{f\}$ is equivalent with F
  - To check if $\alpha \to A$ is redundant in F calculate α^+ with respect to $F - \{\alpha \to A\}$
  - Then $\alpha \to A$ is redundant in F if $A \in \alpha^+$

Normal Forms

Minimal cover

Definition:

- A ***minimal cover*** for the set ***F*** of functional dependencies is a set ***G*** of functional dependencies for which
  - 1. Each functional dependency from G has the form $\alpha \rightarrow A$
  - 2. For each functional dependency $\alpha \rightarrow A$ from G, α does not have redundant attributes
  - 3. There are no redundancy functional dependencies in G
  - 4. G and F are equivalent

or

  - 1. The right side of every dependency in G has a single attribute
  - 2. The left side of every dependency in G is irreducible (i.e. no attributes can be removed from the determinant of a dependency in G without changing G's closure)
  - 3. No dependency f in G is redundant (no dependency can be discarded without changing G's closure)
  - 4. F≡G

- Each set of functional dependencies has at least one minimal cover.
- The minimal covers are not unique (depend on the order in which the functional dependencies / redundant attributes are chosen)

Normal Forms

Decomposition in 3NF example:

Let R[A, B, C, D, E] with the set of functional dependencies F={ABCD→E, E→D, A→B, AC→D}

- The attributes BD from ABCD→E are redundant ⇒ F={AC→E, E→D, A→B, AC→D}
- AC→D is redundant ⇒ F={AC→E, E→D, A→B} and this is a minimal cover

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

- The minimal cover: F={AC→E, E→D, A→B}
- Key: AC
- R is not in 3NF because A→B does not respect 3NF
- Decomposition 3NF of R
  - Relations for each functional dependencies: $R_1(A, C, E)$, $R_2(E, D)$, $R_3(A, B)$
  - Relation for the key of R: $R_4(A, C)$
  - Remove the redundant relation: R_4 ($R_4 \subset R_1$)
  - ⇒ 3NF decomposition is $\{R_1(A, C, E), R_2(E, D), R_3(A, B)\}$

- The decomposition in 3NF is not unique. It depends on
  - The minimal cover chosen
  - The redundant relation chosen to be removed
- Decomposition is the last solution of the problems generated by redundances and anomalies

Normal Forms

Multi-valued dependencies

Example: Consider the relation R[Specialization, Student, MeetingDate] with the repeating attributes *Students* and *MeetingDate*

Specialization	Student	MeetingDate
Computer Science	SCS1 SCS2 … SCSm	MCS1 MCS1 … MCSn
Mathematics	SM1 SM2 … SMp	MM1 MM2 … MMq

The repeating attributes should be eliminatted (obtain 1NF)

- Relation R becomes R'
- *Student* and *MeetingDate* become scalar attribute →

- Here, each student has the same meeting date
- When add / change / remove rocords, additional checks

should be handle

Specialization	Student	MeetingDate
Computer Science	SCS1	MCS1
Computer Science	SCS1	MCS2
Computer Science	…	…
Computer Science	SCS1	MCSn
Computer Science	SCS2	MCS1
Computer Science	…	…
Computer Science	SCSm	MCSn
Mathematics	SM1	MM1
Mathematics	SM1	MM2
Mathematics	…	…
Mathematics	SM1	MMq
Mathematics	SM2	MM1
Mathematics	…	…
Mathematics	SMp	MMq

Normal Forms

Multi-valued dependencies

The ***functional dependency*** $\boldsymbol{\alpha \rightarrow \beta}$ means that ***every value x of α is associated with a unique value y from β***

Definition:

Let R[A] be a relation, with $A = \alpha \cup \beta \cup \gamma$ a set of attributes. The ***multi-valued dependency*** α ⇉ β (should be read ***α multi-determines β***) holds over R if each value *x* of α is associated with a set of values *y* for β: $\beta(x) = \{y_1,\ y_2,\ \dots,\ y_n\}$ and this association holds regardless of the values of γ

Let R[A] be a relation, α ⇉ β a multi-valued dependency, $A = \alpha \cup \beta \cup \gamma$ a set of attributes with γ a non-empty set

○ The association among the values in $\beta(x)$ for β and the value *x* of α holds regardless of the values of γ

(i.e. these associations (between *x* and an element in $\beta(x)$) exist for any value *z* in γ)

Example: if α ⇉ β and there are the rows, there also are the next ones

α	β	γ
x_1	y_1	z_1
x_2	y_2	z_2

α	β	γ
x_1	y_1	z_2
x_2	y_2	z_1

Property 1: Let R[A] be a relation and $A = \alpha \cup \beta \cup \gamma$ a set of attributes. If α ⇉ β then α ⇉ γ

Example: {Specialization} ⇉ {Student}, {Specialization} ⇉ {MeetingDate}

Normal Forms

Multi-valued dependencies

Consider the following relation (it is in BCNF)

Course	Professor	Book
Fundamental Algorithms	Hora	AlgolrithmF I
Fundamental Algorithms	Hora	AlgortihmF II
Fundamental Algorithms	Kyle	AlgolrithmF I
Fundamental Algorithms	Kyle	AlgortihmF II
Databases	Hora	DB I
Databases	Hora	DBMS II
Databases	Hora	DB II

α	β	γ	
a	b_1	c_1	← t_1
a	b_2	c_2	← t_2
a	b_1	c_2	← t_3
a	b_2	c_1	← t_4

$\forall t_1, t_2 \in r$ and $\prod_\alpha(t_1) = \prod_\alpha(t_2) \;\Rightarrow\; \exists\, t_3 \in r$ such that
$\prod_{\alpha\beta}(t_1) = \prod_{\alpha\beta}(t_3)$ and
$\prod_\gamma(t_2) = \prod_\gamma(t_3)$

Aditional rules:

- Complementary: $\boldsymbol{\alpha \rightrightarrows \beta \Rightarrow \alpha \rightrightarrows}$ R-$\boldsymbol{\alpha\beta}$
- Augmentation: $\boldsymbol{\alpha \rightrightarrows \beta, \gamma \subseteq \delta \Rightarrow \delta\,\alpha \rightrightarrows \beta\gamma}$
- Transitivity: $\boldsymbol{\alpha \rightrightarrows \beta, \beta \rightrightarrows \gamma \Rightarrow \alpha \rightrightarrows \gamma - \beta}$
- Replication: $\boldsymbol{\alpha \rightarrow \beta \Rightarrow \alpha \rightrightarrows \beta}$
- Fusion: $\boldsymbol{\alpha \rightrightarrows \beta, \delta \cap \beta = \emptyset, \delta \rightarrow \gamma, \gamma \subseteq \beta \Rightarrow \alpha \rightarrow \beta}$

Normal Forms

Fourth Normal Form (4NF)

Let R a relational schema and F a set of functional dependencies and multi-valued on R

- The relation R is in the ***fourth normal form*** (***4NF***) if for every multi-valued dependency $\alpha \rightrightarrows \beta$ that holds over R, there is
  - $\beta \subseteq \alpha$

  or
  - $\alpha \cup \beta$ =R

  or
  - α is super-key

Trivial multi-valued dependency $\boldsymbol{\alpha} \rightrightarrows \boldsymbol{\beta}$ in relation R: $\beta \subseteq \alpha$ or $\alpha \cup \beta$ =R

- If R[α, β, γ] and $\alpha \rightrightarrows \beta$ non-trivial and $\boldsymbol{\alpha}$ not a super-key then R can be decomposed in
  - $R_1[\alpha, \beta] = \prod_{\alpha \cup \beta}(R)$
  - $R_2[\alpha, \gamma] = \prod_{\alpha \cup \gamma}(R)$

Example: Relation R[Specialization, Student, MeetingDate] becomes
  - R1[Specialization, Student]
  - R2[Specialization, MeetingDate]

Normal Forms

Fourth Normal Form (4NF)

Example: Consider the following relation

Course	Professor	Book
Fundamental Algorithms	Hora	AlgolrithmF I
Fundamental Algorithms	Hora	AlgortihmF II
Fundamental Algorithms	Kyle	AlgolrithmF I
Fundamental Algorithms	Kyle	AlgortihmF II
Databases	Hora	DB I
Databases	Hora	DBMS II
Databases	Hora	DB II

Course $\rightrightarrows$ Professor

Course	Book
Fundamental Algorithms	AlgolrithmF I
Fundamental Algorithms	AlgortihmF II
Databases	DB I
Databases	DBMS II
Databases	DB II

The relation can be decomposed in [Course, Professor] and [Course, Book]

Course	Professor
Fundamental Algorithms	Hora
Fundamental Algorithms	Kyle
Databases	Hora

Normal Forms

- A dependency (simple, multi-valued) in a relation can be eliminated via decompositions (the original relation is decomposed into a collection of new relations)
- There are relations without such dependencies that can still contain redundant information, which can be a source of errors in the database

Example: Consider the relation R[Specialization, Course, Schedule] that store the schedules of the courses per specialization; the relation has no functional dependencies and the key is {Specialization, Course, Schedule}

Specialization	Course	Schedule
Mathematics	Analysis	8:00 – 10:00
Mathematics	Geometry	10:00 – 12:00
Computer Science	Analysis	10:00 – 12:00
Mathematics	Analysis	10:00 – 12:00

- there are redundant data
  - Specialization *Mathematics* has the course *Analysis*
  - Specialization *Mathematics* has the schedule *10:00 – 12:00*
  - Schedule *10:00 – 12:00* is in course *Analysis*

Specialization	Course	Schedule
Mathematics	Analysis	8:00 – 10:00
Mathematics	Geometry	10:00 – 12:00
Computer Science	Analysis	10:00 – 12:00
Mathematics	Analysis	10:00 – 12:00

Normal Forms

Specialization	Course	Schedule
Mathematics	Analysis	8:00 – 10:00
Mathematics	Geometry	10:00 – 12:00
Computer Science	Analysis	10:00 – 12:00
Mathematics	Analysis	10:00 – 12:00

- If some values are changed (e.g. *Mathematics* will have the schedule *12:00 – 14:00* instead of *10:00 – 12:00*, several updates should be performed; rows 2 and 4 will be affected)
- This relation cannot be decomposed into 2 relations, via projection, because new data would be introduced through join; only 3 possible projections on two attributes can be considered

R1	Specialization	Course
	Mathematics	Analysis
	Mathematics	Geometry
	Computer Science	Analysis

R2	Course	Schedule
	Analysis	8:00 – 10:00
	Geometry	10:00 – 12:00
	Analysis	10:00 – 12:00

R3	Specialization	Schedule
	Mathematics	8:00 – 10:00
	Mathematics	10:00 – 12:00
	Computer Science	10:00 – 12:00

Normal Forms

R1*R2	Specialization	Course	Schedule
	Mathematics	Analysis	8:00 – 10:00
	Mathematics	Geometry	10:00 – 12:00
	Computer Science	Analysis	8:00 – 10:00
	Mathematics	Analysis	10:00 – 12:00
	Computer Science	Analysis	10:00 – 12:00

- Evaluation R1*R2 contains an extra tuple, which did not exist in the initial relation
- Evaluation R2*R3 and R1*R3 will also have extra records that are not in the initial relation
- There can be found a relation such that the composition R'*R3 to give the initial relation
- So, R cannot be decomposed in 2 projections, but it can be decomposed in 3 projections
  - e.g. R1R2R3=R1*R2*R3, or R1R2R3=*(R1, R2, R3)

Join dependency

Definition:

The relation R satisfies the ***join*** dependency $* \{R_1, \ldots, R_n\}$ if $R_1, \ldots, R_n$ is a lossless - join decomposition of R.

or

Let R[A] be a relation and $R_i[\alpha_i], i = 1, \ldots n$ the projections of R on α_i. R satisfies the ***join*** dependency $*\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ if $\mathrm{R} = R_1 * \ \ldots * R_n$

A multi-values dependency $\alpha \rightrightarrows \beta$ can be expressed through a join dependency $* \{\alpha\beta, \alpha(R-\beta)\}$

- The previous example has a join dependency (R1R2R3)

Normal Forms

Fifth Normal Form (5NF)

The relation R is in the ***fifth normal form*** (***5NF***) if and only if for every join dependency of R

- R_i=R for any i

  or

- the dependency is involved by a set of functional dependencies from R in which the left side is a key for R

or

The relation R is in the ***fifth normal form*** (***5NF***) if every non-trivial join dependency is implied by the candidate keys in R

- join dependendy $*\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ on R is trivial if at least one α_i is the set of all attributes of R
- join dependency $*\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ on R is implied by the candidate key of R if each α_i is a super-key in R

Example: R1R2R3 is not in 5NF

- Decomposition: projections on R1, R2, R3

Normal Forms

Normal forms examples (homework) ☺

e.g. 1: Consider the relational schema R[<u>Rid</u>, A, B, C, D, E, F, G] with none repeatable attributes. {Rid} is the key and {E, F} is the only candidate key. The set of the functional dependencies is {A→C, B→D, E→G}

Is R in 1NF? What about 2NF, 3NF and BCNF?

Solution: 1NF: yes

2NF: no, because the non-prime attribute G is not completely functional dependent on the key {E, F}

3NF,: no, because it is not in 2NF

BCNF: no, because it is not in 2NF and also 3NF

e.g. 2: Consider the relational schema R[<u>X</u>, Y, Z, V, W] with none repeatable attributes and none candidate keys. {X} is the key. The set of the functional dependencies is {X→V, Z→X, V→W}

Is R in 1NF? What about 2NF and 3NF?

Solution: 1NF: yes; 2NF and 3NF: no

Normal Forms

Normal forms examples (homework) ☺

e.g. 3: Consider the relational schema R[<u>A, B</u>, C, D, E, F] with none repeatable attributes and none candidate keys. {A, B} is the key. The set of the functional dependencies is {C→D, DE→F}
Is R in 3NF?
Solution: 3NF: no

e.g. 4: Consider the relational schema R[<u>A</u>, B, C, D] with none repeatable attributes and none candidate keys. {A, B} is the key. The set of the functional dependencies is {A→D, B→C, D →B}
Is R in 3NF / BCNF?
Solution:

- 1NF: yes
- 2NF: no, because the primary key contains only one attribute and there are no partial dependencies on the key
- 3NF: no. From A→D and D →B follows that exist transitive dependencies on the key (or, from B→C, where D is not super-key and B is a non-prime attribute)
- BCNF: No. It is not in 3NF

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