Introduction to Operating Systems and SQL for Data Science

Lecture 10 – Normalization

Previous lecture

- SQL

Lecture background

- We saw that relational database give us flexibility for representing relations.
- Question: is there away to determine if a relational scheme is better than others?
- In this lesson, we will learn the process of normalize the dataset and show how to get a better scheme.

https://en.wikipedia.org/wiki/Database_normalization

Normalization

Normalization goal

- Assume we build the relation "courses" in this way:

Course ID	Course name	Department ID	Department name
1	Introduction to programming	202	Computer science
2	Databases	202	Computer science
3	Linear algebra	203	Math

- You can see that in this table we have data redundancy, as the department name is showing twice.

What's wrong with redundancy?

- Trouble while inserting – We can’t add the table new department that for this stage offers no courses.

- Trouble while updating – Because in this table there is data redundancy if we change the department name from “Computer Science” to “Computers” we need to update many rows:
  - Waste of time.
  - Could resolve in non-consist data and mistakes.

- Trouble while deleting – if we deletes the course Linear Algebra which is the only course for Math department, we will lose Math department and its number.

Rules for creating “good” relations

- We need to define “simple”/understandable relations.
- The attributes of the relation need to describe one entity.
- We need to avoid from data redundancy – to avoid update issues.
- We need to avoid from Null values.

Reichman University

“Bad” relation example

A relation has the following attributes:

SSN, student name, address, department code, number of students, course code, course name, credit, grade.

Problems:

- What the relation represent?
- What is the key?
- Updates anomalies
- Many null values

Normalization

Normalize the data is a process which its goal is to create a collection of relations which represent entities in the most accurate and trustworthy, and which is not including unwanted issues that arises from data redundancy (e.g., Null values)

Reichman University

Normalization theory - goals

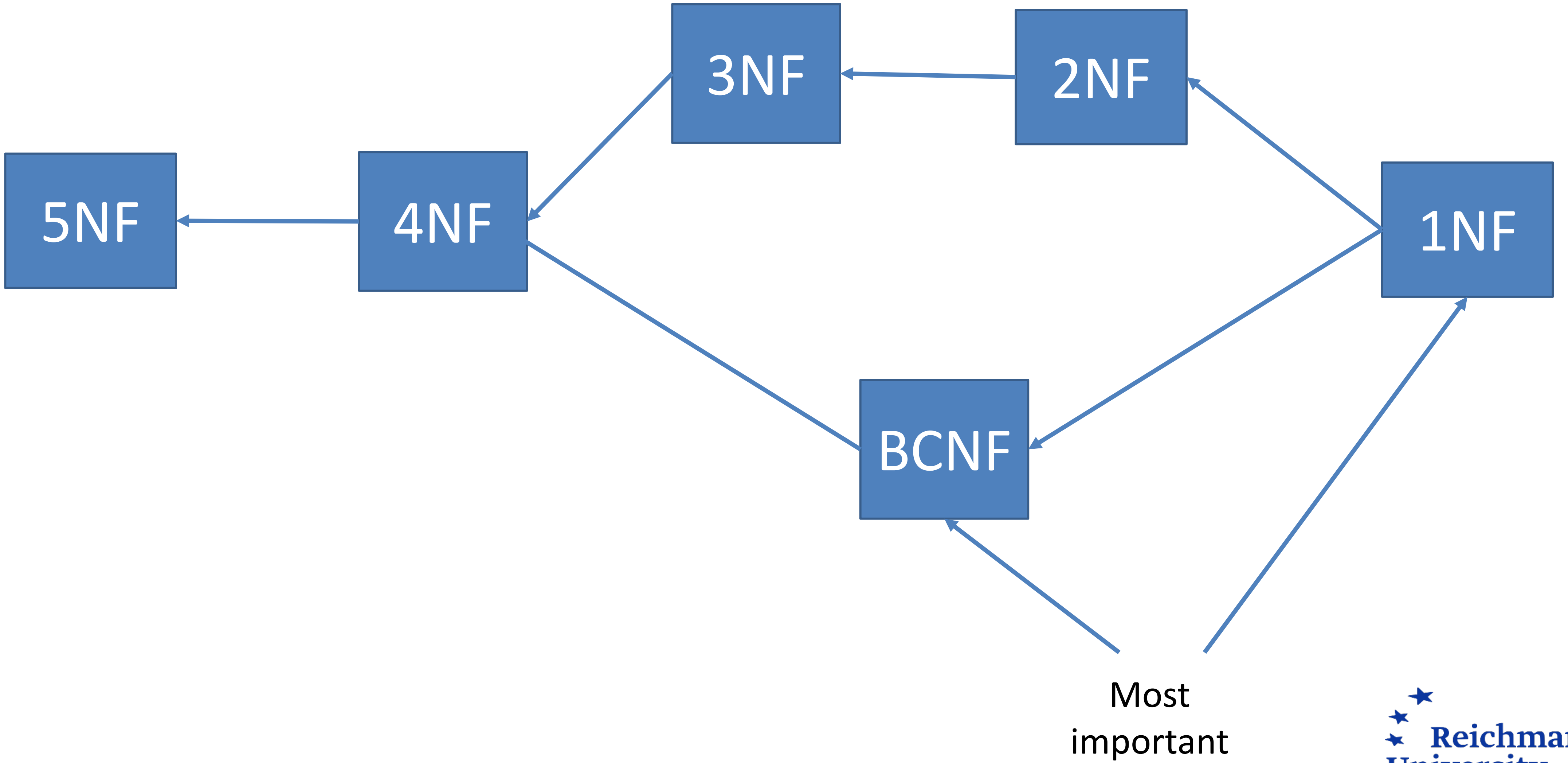
- Define rules of valid relations – normalization rules.

- Examine each given relation with the rules, and if a problem is discovered - solve it by appropriate "disassemble" to valid relations.

- Analyze user requirements, and form proper relations (synthesis).

Reichman University

Data normalization types



1NF
(First normal form)

Relation in the relational model is satisfy 1NF if:

- It have a table form.
- It have atomic values (not list attributes)
- It have Key(s), and the key(s) contain unique values (and not null)

Example for a relation that not satisfy 1NF:

Employee ID	Employee name	Child name
123	Eliezer	Gila, Moshe
124	John	Yehoshua, Zvika, Tamar

Employees

1NF - Solution

Multiple attributes will be broken down into two tables:

Employee ID	Employee name
123	Eliezer
124	John

Employees

Employee ID	Child name
123	Gila
123	Moshe
124	Yehoshua
124	Zvika
124	Tamar

Employees children

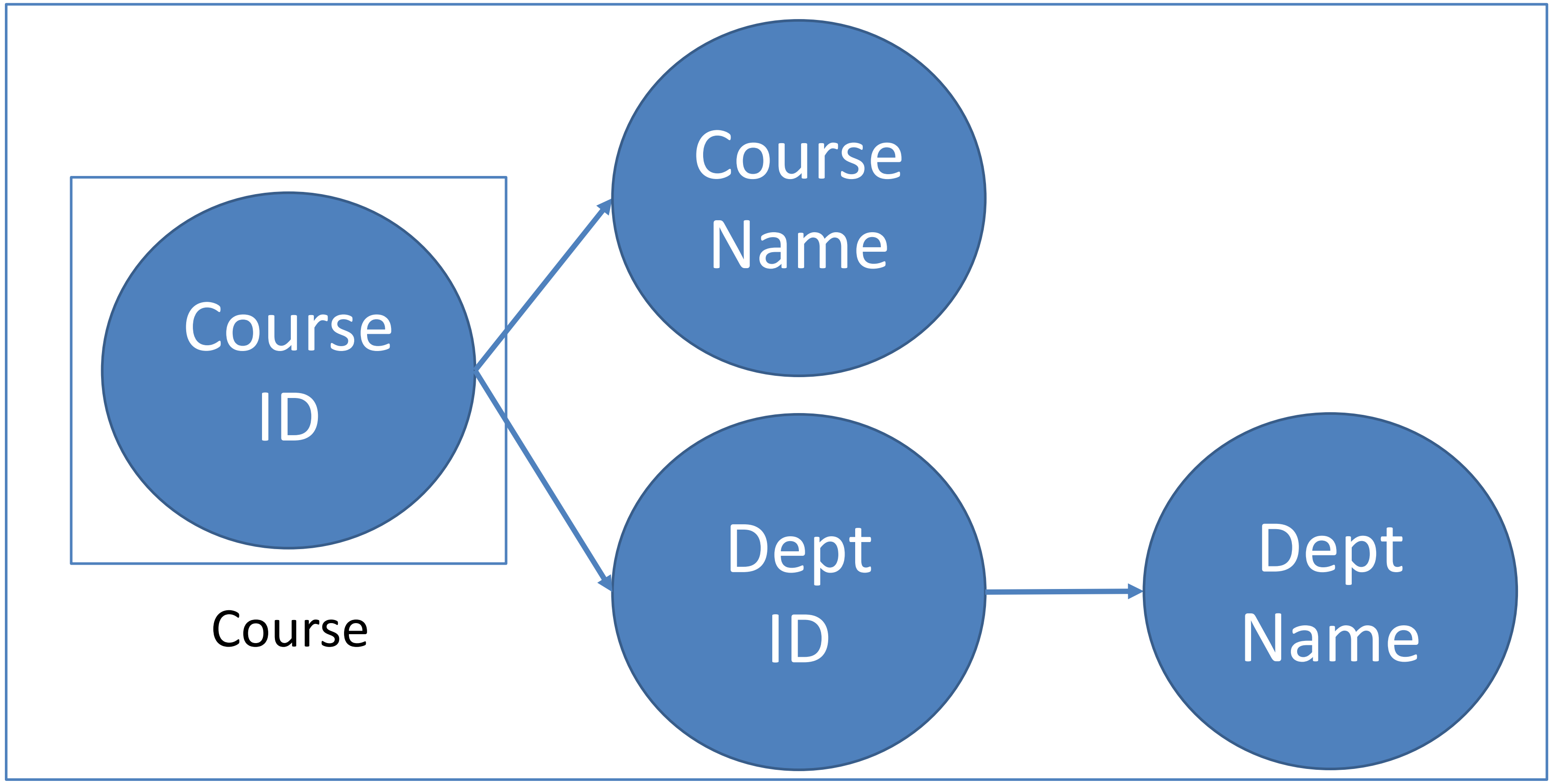
Reichman University

Functional dependency - definition

Given relation R (a_1,…,a_n) and two subgroups X, Y $X, Y \subseteq (a_1, \ldots, a_n)$. *X* is said to **functionally determine** *Y* (written *X* → *Y*) if and only if each *X* value in *R* is associated with precisely one Y value in R; R is then said to satisfy the functional dependency X → Y.

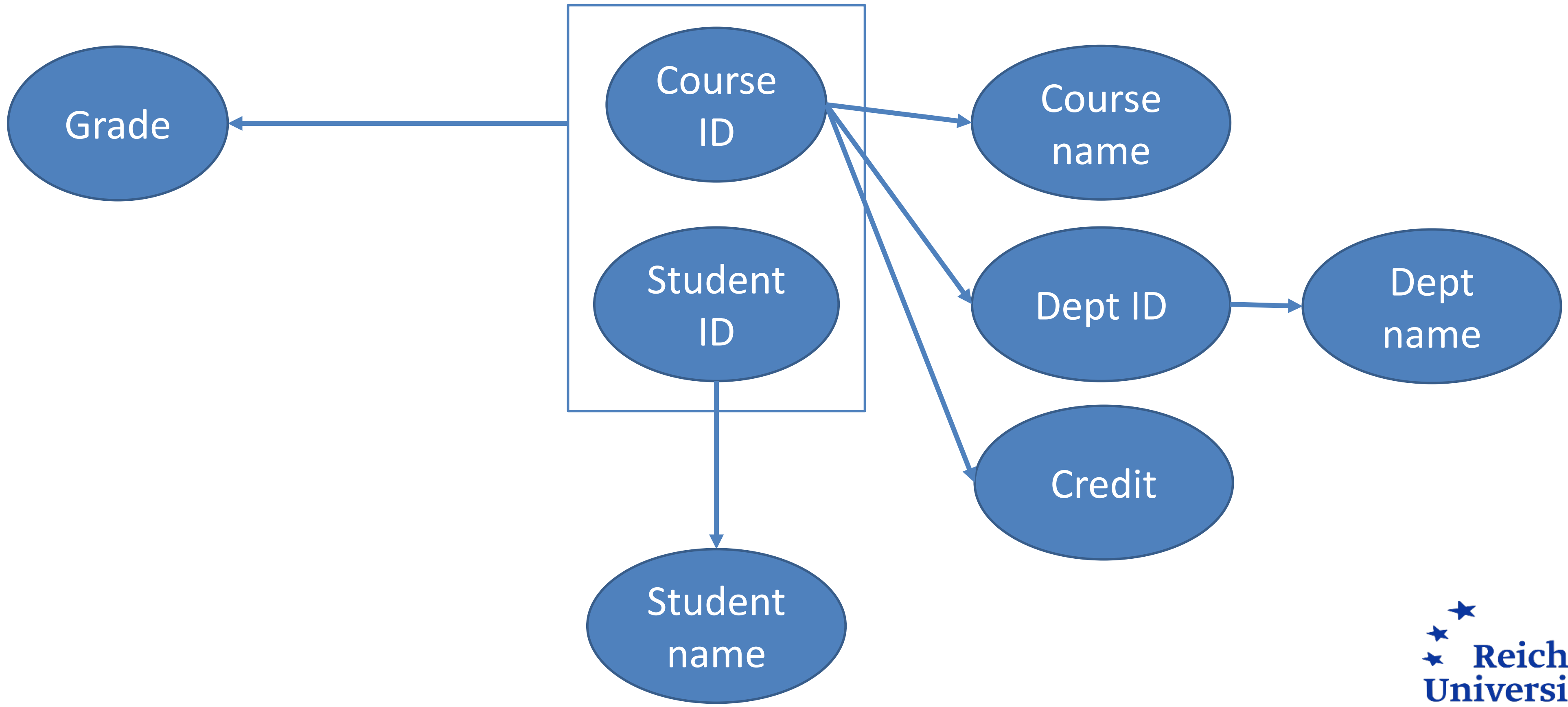
Functional dependency - definition

Customarily, *X* is called the ***determinant*** set and *Y* the ***dependent*** set.



Example B

grades (<u>Course ID</u>, <u>Student ID</u>, Course name, Student name, dept ID, dept name, credit, grade)



Dependencies types

- Trivial dependency, if $Y \subseteq X$:

{Course ID, Student ID} -> {Course ID}

- Real Nontrivial dependency if $X \cap Y = \emptyset$

{Course ID, Student ID} -> {Course name, student name}

- Other nontrivial dependency, any other case

{Course ID, Student ID} -> {Course ID, student name}

Reichman University

Key

Given sub-group $X \subseteq \{a_1, \dots, a_n\}$ said to be the key to relation $R(a_1, \dots, a_n)$ if:

a) $X \rightarrow \{a_1, \dots, a_n\}$

b) There is no $X^* \subset X$ such as, $X^* \rightarrow \{a_1, \dots, a_n\}$ which means the key is minimal.

Super key

Given a sub-group $X \subseteq \{a_1, \dots, a_n\}$ is a super key if exist a $X^* \subseteq X$ so that $X^* \rightarrow \{a_1, \dots, a_n\}$.

2NF (Second normal form)

- Every attribute which is not part of the key is functionally dependent by the (whole) key.

- This rule applies to relationships whose key consists of more than one attribute, and the condition that must exist is that any attribute that does not belong to the key must be functionally dependent on the whole key and not on some of the key fields.

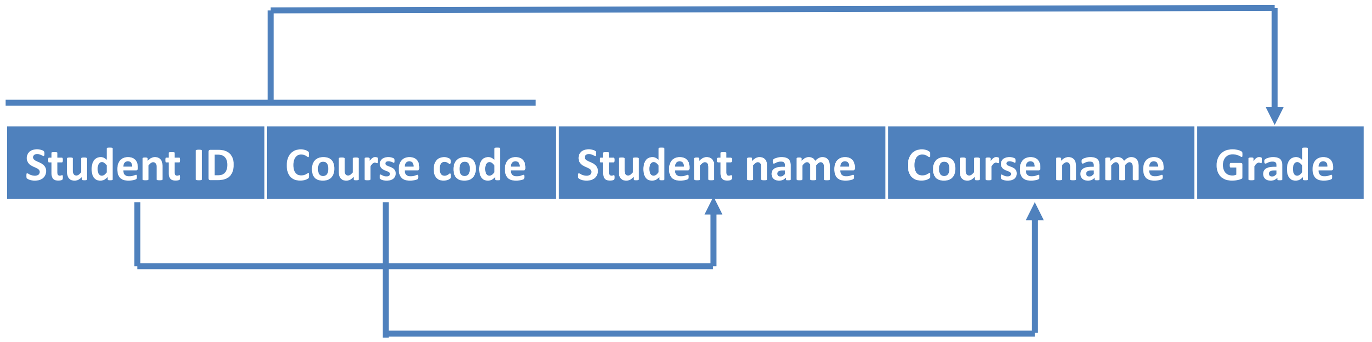
2NF - Continue

- If such a partial dependency is discovered - the functional determine (i.e., the "determining" part of the key and the dependent attribute) must be copied and create a separate relation for them, in which the "determining“ part is the key.

- The dependent attribute is deleted from the original relation - but not the "determinant". As a rule, do not disassemble existing functional dependencies.

2NF - Example

Partiality dependent on the key.



- Student ID and Course code are the relation key.
- They both functionally determine grade – as should be.
- Student name is functionally dependent by Student ID alone.
- Course name is functionally dependent by Course code alone.

2NF - Solution

Courses: (Course code, Course name)

Students: (Student ID, Student name)

Grades: (Student ID, Course code, grade)

3NF
(Three normal form)

Every attribute which is not part of the (whole) key is not functionally dependent by any other attribute but the (whole) key itself.

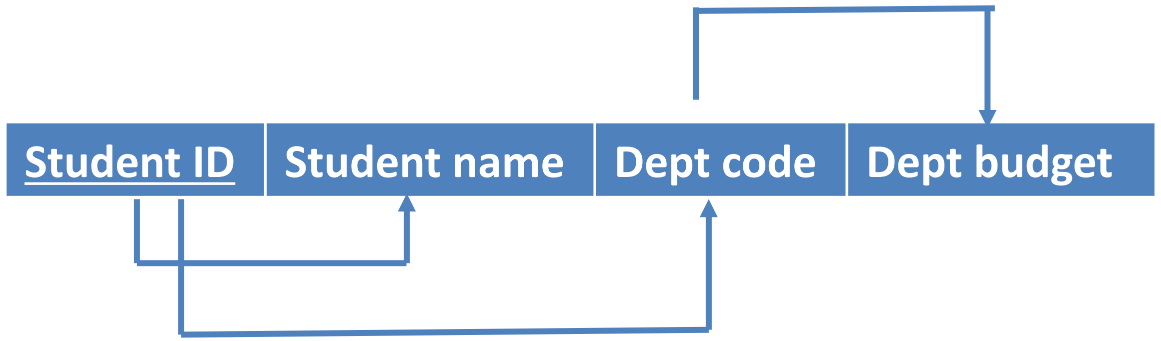
If such rule is discovered, we need to copy the determine and dependent attributes to different relation.

As 2NF, the determine attribute stays in the original relation. (because is part of the key)

3NF - example

In the following relation budget-department is functionally dependent on dept code.

In this example the relation is satisfy 2NF (because every attribute is dependent by the key) but not satisfy 3NF (because budget is dependent not only by the key, it also dependent by dept code which is not part of the key).



3NF - Solution

Solution:

Students (student ID, Student name, course code)

Departments (Dept code, dept budget)

2NF VS. 3NF

Every relation that satisfy 3NF will satisfy 2NF.

The opposite is not true.

Conclusion

Normalization rule	requirement	Normalization action
1NF: the key	A key is defined, and the key contains unique (not null) values that unambiguously identify each value.	Create new relation for each non-atomic value and nested relations.
2NF: the whole key	Any attribute other than a key attributes depends functionally on the entire key.	Split into relation according to a partial key and the fields that depend on it. Leave a relation with the original key and any field depending on it.
3NF: Nothing but the key	Any field other than a key field does not functionally depend on any field other than the key.	Split and create a relation that contains the non-key fields that are functionally dependent on other non-key fields.

Reichman University

Armstrong's axioms for functionality decencies

- Reflexivity:

$$IF\ Y \subseteq X\ Then\ X \rightarrow Y$$

- Expansion:

$$IF\ X \rightarrow Y\ Then\ X \cup Z \rightarrow Y \cup Z$$

- Transitivity:

$$IF\ X \rightarrow Y\ and\ Y \rightarrow Z\ Then\ X \rightarrow Z$$

Reichman University

Rules

- Union rule:

$$X \rightarrow Y; X \rightarrow Z \; Then \; X \rightarrow Y \cup Z$$

Prove:

$$X \rightarrow Y$$
$$X \rightarrow Z$$
$$X \cup X \rightarrow Y \cup X$$
$$X \cup Y \rightarrow Z \cup Y$$
$$X \rightarrow Y \cup Z$$

Reichman University

Rules

- The partiality transitive rule:

$$WY \rightarrow Z, X \rightarrow Y \; Then \; WX \rightarrow Z$$

- Disassembly rule:

$$X \rightarrow Y, Z \subseteq Y \; Then \; X \rightarrow Z$$

Reichman University

Closure of Dependency Sets

Given F dependent groups in relation R.

We mark F^+ all the dependencies that resulting from the dependencies of F that are get from the mentioned rules:

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

$F+ = \{\emptyset \rightarrow \emptyset,\ A \rightarrow \emptyset,\ A \rightarrow A,\ A \rightarrow B,\ A \rightarrow C,\ A \rightarrow AB,\ A \rightarrow AC,\ A \rightarrow BC,$
$A \rightarrow ABC,\ B \rightarrow \emptyset,\ B \rightarrow B,\ B \rightarrow C,\ B \rightarrow BC,\ C \rightarrow \emptyset,\ C \rightarrow C,\ AB \rightarrow \emptyset,\ AB \rightarrow A,$
$AB \rightarrow B,\ AB \rightarrow C,\ AB \rightarrow AB,\ AB \rightarrow AC,\ AB \rightarrow BC,\ AB \rightarrow ABC,\ AC \rightarrow \emptyset,$
$AC \rightarrow A,\ AC \rightarrow B,\ AC \rightarrow C,\ AC \rightarrow AB,\ AC \rightarrow AC,\ AC \rightarrow BC,\ AC \rightarrow ABC,$
$BC \rightarrow \emptyset,\ BC \rightarrow B,\ BC \rightarrow C,\ BC \rightarrow BC,\ ABC \rightarrow \emptyset,\ ABC \rightarrow A,\ ABC \rightarrow B,$
$ABC \rightarrow C,\ ABC \rightarrow AB,\ ABC \rightarrow AC,\ ABC \rightarrow BC,\ ABC \rightarrow ABC\}$

Reichman University

Closure of Dependency Sets regarding the dependent groups

The closure of the group attribute X in the dependent group F is the group $X^{+} \subseteq \{a_1, \dots, a_n\}$ that exist $X \rightarrow X^{+}$ under F.

In simple words, which attributes are dependent in the group of attributes while using collection of dependencies.

Algorithm for finding X_F^+

```
$F_{new} \leftarrow F$
$X^+ \leftarrow X$
$changed \leftarrow True$
$WHILE\,(changed)$
$\{$
    $changed \leftarrow False$
    $\forall Y \rightarrow Z \in F_{new}$
    $\{$
        $if\ Y \subseteq X^+\ then$
        $\{$
            $X^+ \leftarrow X^+ \cup Z;$
            $F_{new} \leftarrow F_{new} - \{Y \rightarrow Z\}$
            $changed \leftarrow True$
        $\}$
    $\}$
$\}$
```

Reichman University

Example

$$A \rightarrow B, BC \rightarrow D.$$

- $A^+ = AB$
- $C^+ = C$
- $(AC)^+ = ABCD$

BCNF
(Boyce Codd normal form)

Definition: Every relation satisfy BCNF if every time that exist a non-trivial functional dependency X->Y, then X is a super key.

Alternative definition for BCNF

Relation R with dependency group F is a BCNF if for each functional dependency X->Y in F^+ exist at least one of the following conditions:

- X->Y is a trivial dependency.
- X is the super key of the relation

Lossless-join decomposition

- In order to normalize a relation which not satisfy BCNF we must do a lossless-join decomposition.

- Decomposition of a relation R is a collection of sub-relations $R_1,\dots,R_n$.

- A decomposition $R_1,\dots,R_n$ is a lossless-join decomposition of R given functionally dependent groups F if every relation r over R existing $r = \pi_{R1}r \bowtie \dots \bowtie \pi_{Rn}r$

Example

City	AreaCode	PhoneNum	Name
Tel Aviv	03	123456	Cohen Moshe
Tel Aviv	03	123457	Levin vered
Haifa	04	123456	Perez haim

CityCodePhone

City	AreaCode
Tel Aviv	03
Haifa	04

CityCode

~~City~~	PhoneNum	Name
~~Tel Aviv~~	123456	Cohen Moshe
~~Tel Aviv~~	123457	Levin vered
~~Haifa~~	123456	Perez haim

CityPhone

Preserving data: essential, allows to restore the data.

Algorithm to normalize BCNF

- The algorithm input is a relation R
- Step 1: find a connection X->Y that violate the BCNF situation. If there is no such connection, stop.
- Step 2: compute X^+.
- Step 3:Decompose relation R into X^+ and (R-X+) U X.
- Step 4: Return the process for all every table.

Example

- R:

Course code, Student ID, semester, course name, student name, department number, department name, credit, grade, exam date.

The key is – Course code, Student ID and semester.

Example - continue

- R:

Course code, Student ID, semester, course name, student name, department number, department name, credit, grade, exam date.

functional dependency: {
Course code -> Course name, credit, department number.
Student ID -> student name.
Department number -> department name.
Course code, semester -> exam date.
Course code, student ID, semester -> grade.
}

Example - continue

- Examine the connection Course code -> Course name. Thus, the table doesn’t satisfy BCNF. Why?
- (Course code is not R super key)

- Let’s compute the closure of Course code:

(Course code)$^+$: [Course code, Course name, credit, department number, department name]

Example - continue

R: Course code, Student ID, semester, course name, student name, department number, department name, credit, grade, exam date.

(Course code)$^+$: [Course code, Course name, credit, department number, department name]

So, if we made a lossless join decomposition(step 3), we get the following two tables:

R2=(Course code, Course name, credit, department number, department name)

R3=(Course code, Student ID, semester, student name, grade, exam date)

Example - continue

R2=(Course code, Course name, credit, department number, department name)

Now, examine the connection department code -> department name. Thus, the table doesn’t satisfy BCNF. Why? (department code is not R super key) we will break into two relations in the following way:

R4=(Course code, Course name, credit, department number)

R5=(department number, department name)

Example - continue

R3=(Course code, Student ID, semester, student name, grade, exam date)

Now, examine the connection Student ID-> student name. Thus, the table doesn’t satisfy BCNF. Why? (student ID is not R super key) we will break into two relations in the following way:

R6=(Course code, Student ID, semester, grade, exam date)

R7=(Student ID, student name)

Example - continue

R6=(Course code, Student ID, semester, grade, exam date)

Now, examine the connection Course code, semester-> exam date. Thus, the table doesn’t satisfy BCNF. Why? (Course code, semester is not R super key) we will break into two relations in the following way:

R8=(Course code, Student ID, semester, grade)

R9=(Course code, semester, exam date)

Final solution

All relation now satisfy BCNF hence we stop.

R4=(Course code, Course name, credit, department number)

R5=(department number, department name)

R7=(Student ID, student name)

R8=(Course code, Student ID, semester, grade)

R9=(Course code, semester, exam date)

BCNF vs. 3NF

- Statement: if a relation satisfy BCNF it also satisfy 3NF.

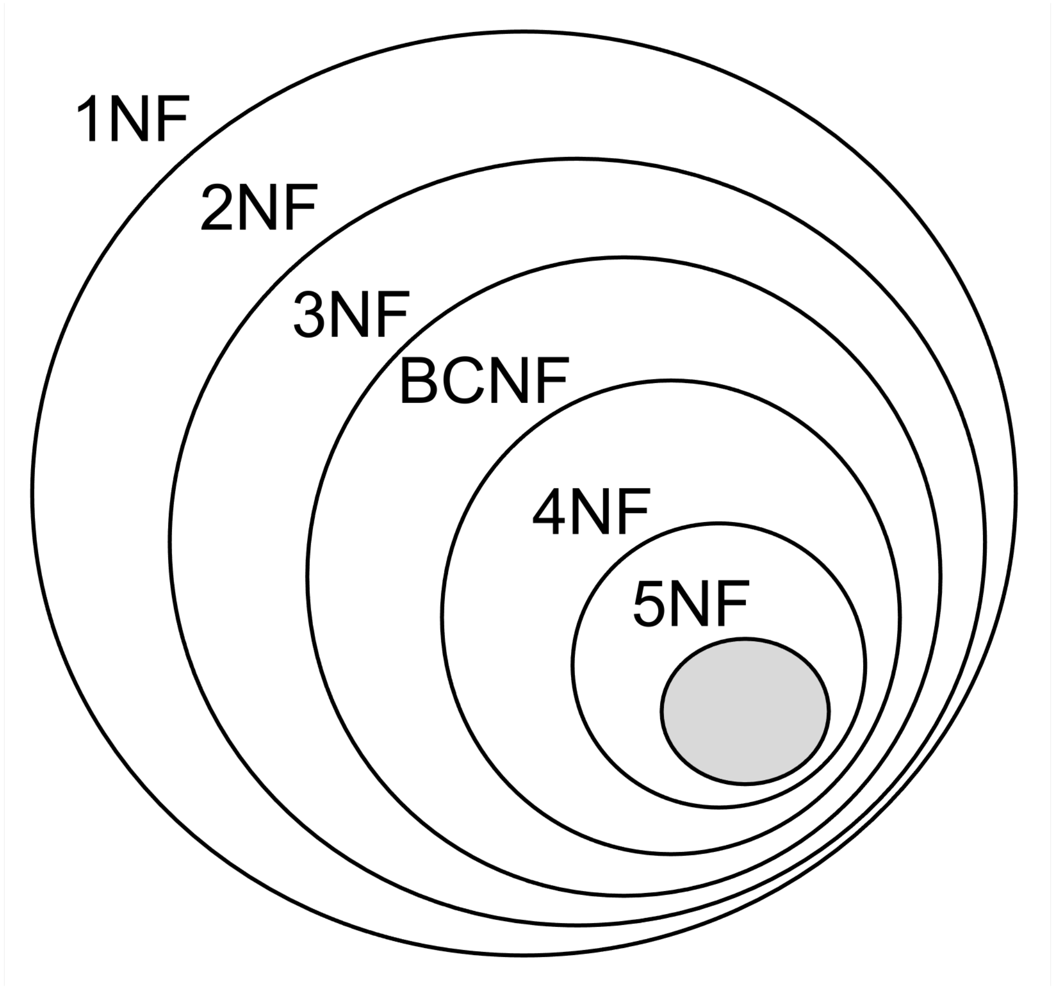
- However, there are tables that satisfy 3NF but not BCNF.

Formal definition for 3NF

- Relation R with functional dependencies group F satisfy 3NF if for each functional dependency X->Y in F^+ is existing at least one of the following rules:
  - X -> Y is a trivial dependency
  - X is a super key of R
  - Every dependency in the expression (Y-X) is equally contain in one of the valid keys of R.

- From the formal definition, it is oblivious that BCNF is stricter than 3NF because its do not allow the third rule. But, which ever table that satisfy BCNF will necessarily satisfy 3NF.

There are even more stricter rules....



https://aksakalli.github.io/2012/03/12/database-normalization-and-normal-forms-with-an-example.html