Organization of Data bases

semester 4

lector: Lyudmyla Rozova, associate professor, phd

email: lyudmyla.rozova@khpi.edu.ua

Lecture 5

1. Normal Forms of Relations in DataBase
2. Functional dependency
3. Decomposition
4. First, second ,third normal forms

1.Normal Forms of Relations in DataBase

The starting point of any database design is the representation of the subject area in the form of one or more relations, and at each design step, a certain set of relation schemes with "improved" properties is produced.

To remove the anomalies in the relations’ design, the method of normalizing of relations schemes is used, and each next normal form has properties, that are better than the previous one. Each normal form corresponds to a certain set of **constraints**.

Relation is in some **normal form** if it satisfies **set of constraints** of this normal form.

An example is ***the first normal form constraint*** — the values of all the attributes of a relation are **atomic.**

In the theory of relational databases, the following sequence of **normal forms** is usually presented:
1) first normal form (1 NF);
2) second normal form (2 NF);
3) third normal form (3 NF);
4) normal form of Boyce-Codd (BCNF);
5) fourth normal form (4 NF);
6) fifth normal form, or the normal form of the projection-join (5 NF or PJ / NF).

Whether it is possible to continue normalization further, to get 6, 7, etc. NF?
Indeed, there are additional NFs, but 5 NFs are considered to be final.
And for practical design, 3 NF is considered sufficient.

The <u>**main properties of normal forms**</u> are:

1) each next normal form is in some sense better than the previous normal form;

2) the properties of the previous normal forms are preserved when passing to the next normal form.

The database design process is based on the normalization method. It is the decomposition of a relation in the previous normal form into two or more relations that satisfy the requirements of the next normal form.

The normalization process is based on the concept of normal forms (NF).

2. Functional dependency (FD)

Normalization is based on the concept of **functional dependency** of the attributes of relation.

Functional dependency is monosemantic dependency tabulated in database management systems.

Let's give the definition of **functional dependency**

Let **R** be a relation. The set of attributes **Y** is functionally dependent on the set of attributes **X** (X functionally determines Y) only if, when for any state of the relation **R** and for any corteges $r_1, r_2 \in R$, from the fact that $r_1X = r_2X$ follows that $r_1Y = r_2Y$ (so in all corteges that have the same **X** attribute values, the **Y** attribute values are also the same in any state of the **R** relation).

Functional dependency designation: **$X \rightarrow Y$.**

The set of **X** attributes is **the determinant of functional dependency,**

the set of **Y** attributes is **the dependent part.**

3.Decomposition without data losses

Normalization uses a decomposition operation. The normalization procedure provides for the division of relation into others - decomposition. Moreover, this decomposition should be without loss of information from the original relation. We can say that the decomposition should be reversible.

For example, let's consider the relation

Students (*GradebookNo., Surname, Name, GroupCode, Address, Phone*).

Let's decompose it into two relations:

Students1 (*GradebookNo., Surname, Name*) and

Students2 (*GroupCode, Address, Phone*)

In this decomposition, information is lost. -, the address and phone number only for the first student in the list will be displayed for each group. The joining of the obtained relations makes it impossible to rebuilt completely the original relation.

Let's make the decomposition in another way:
Students3 (*GradebookNo., Surname, Name*) and
Students4 (*GradebookNo., Address, Phone*)
This is lossless decomposition. You can join these two relations, and get the original.
At its core, **decomposition** is an operation of **projection** of relational algebra, therefore, each relation obtained during decomposition is called a **projection of the original one**.

So the question is: what conditions must be for the projection of the original relation to guarantee that by the joining of obtained relations we will get the original relation?

4. The first normal form (NF)

This question is answered by **Hez's theorem**:
Let **R (A, B, C)** be a relation, where **A, B, C** are subsets of the set of its attributes.
If **R** satisfies **FD**: **A → B**,
then **R** is equal to the joining of its projections **{A, B}** and **{A, C}.**

First normal form (NF)

Relation is in **1st NF** if and only if all domains used in it contain only **scalar values**.
In other words: The relation is in **1st NF** if and only if the values of all fields are indivisible. All relations are automatically in **1st NF**.
For example, if the relation has a full-name field, the relation is not in **1st NF**, since the values of this field can be divided into last name and first name.

5. Second normal form

5. The second normal form

Relation is in 2nd NF if and only if it is in 1st NF and each non-key attribute is **irreducibly dependent on a primary key**.

A non-key attribute is an attribute that is not part of any potencial key.

A functional dependence is called **irreducible** if no attribute can be omitted from its determinant without breaking the dependence.

For example, consider the relation

The Academic progress (*GradebookNo, Surname, Name, Subject, Mark*).

If the *GradebookNo*. is assigned here as the **primary key**, then the attribute *Subject* will not depend on this key. Those, in this case, the relation is not in 2nd NF.

You can then take the set of attributes {*GradebookNo.,Subject*} as the **primary key**. In this case, all attributes depend on this key.

But the attributes ***Surname, Name*** depend reducibly on a primary key.

So the attribute ***Subject*** can be removed from the determinant of the following FD without breaking the dependency:

{*GradebookNo.,Subject*}→ { *Surname, Name* }

The relation is not in 2 NF with this primary key.

To get a relation in 2 NF, let's perform a decomposition.

This can be done by using **Heath's theorem**.

The **Academic progress** relation satisfies the FD:

{GradebookNo.} → {Surname, Name } (A → B)

Outside of this FD, the following set of attributes are:

{*Subject, Mark*} (C)

Then, according to ***Hez's theorem,***

The **Academic progress** relation is equal to the joining of its projections with the following sets of attributes:

{*GradebookNo, Surname, Name* } ({A, B})

{*GradebookNo, Subject, Mark* } ({A, C})

So the lossless decomposition is possible into relations:

Students (*GradebookNo, Surname, Name*)

The Academic progress 1 (*GradebookNo, Subject, Mark*).

6. Third normal form

Relation is in **3rd NF** if and only if it is in **2nd NF** and each non-key attribute is **nontransitively dependent on the primary key**.
In other words: relation **R** is in third normal form (3NF) if and only if the relation is in 2NF and all non-key attributes are **mutually independent** (non-transitively dependent on the primary key)

Attributes are said to be ***mutually independent*** if none of them is functionally dependent on the other.

For example, add the ***GroupLeader*** attribute to the ***Students*** relation. Then the following FD will be in relation:
{*Gradebook No.*} → {GroupCode}, {GroupCode} → {*GroupLeader*}
That is, the ***GroupLeader*** attribute depends on the primary key (**GradebookNo**) transitively through the ***GroupCode*** attribute, and not directly.
So this relation is not in 3 NF.

Let's make the decomposition. In this case, according to Hez's theorem, there can be two variants of decomposition.

Variant 1 is based on FD **{ *Gradebook No* } → {*Surname, Name, GroupCode*}.**

As a result, we get the following projections:

{ *Gradebook No, Surname, Name, GroupCode*} and

{ *Gradebook No, GroupLeader* }

Variant 2 is based on FD

{*GroupCode*} → {*GroupLeader*}.

As a result, we get the following projections:

{*GroupCode, GroupLeader*}

{GroupCode, *Gradebook No, Surname, Name* }

The second variant is suitable, because in the first variant, errors are possible when updating the data

As a result we have two projections

Groups {*GroupCode, GroupLeader*} and

Students1 {*GradebookNo., Surname, Name, GroupCode*}.

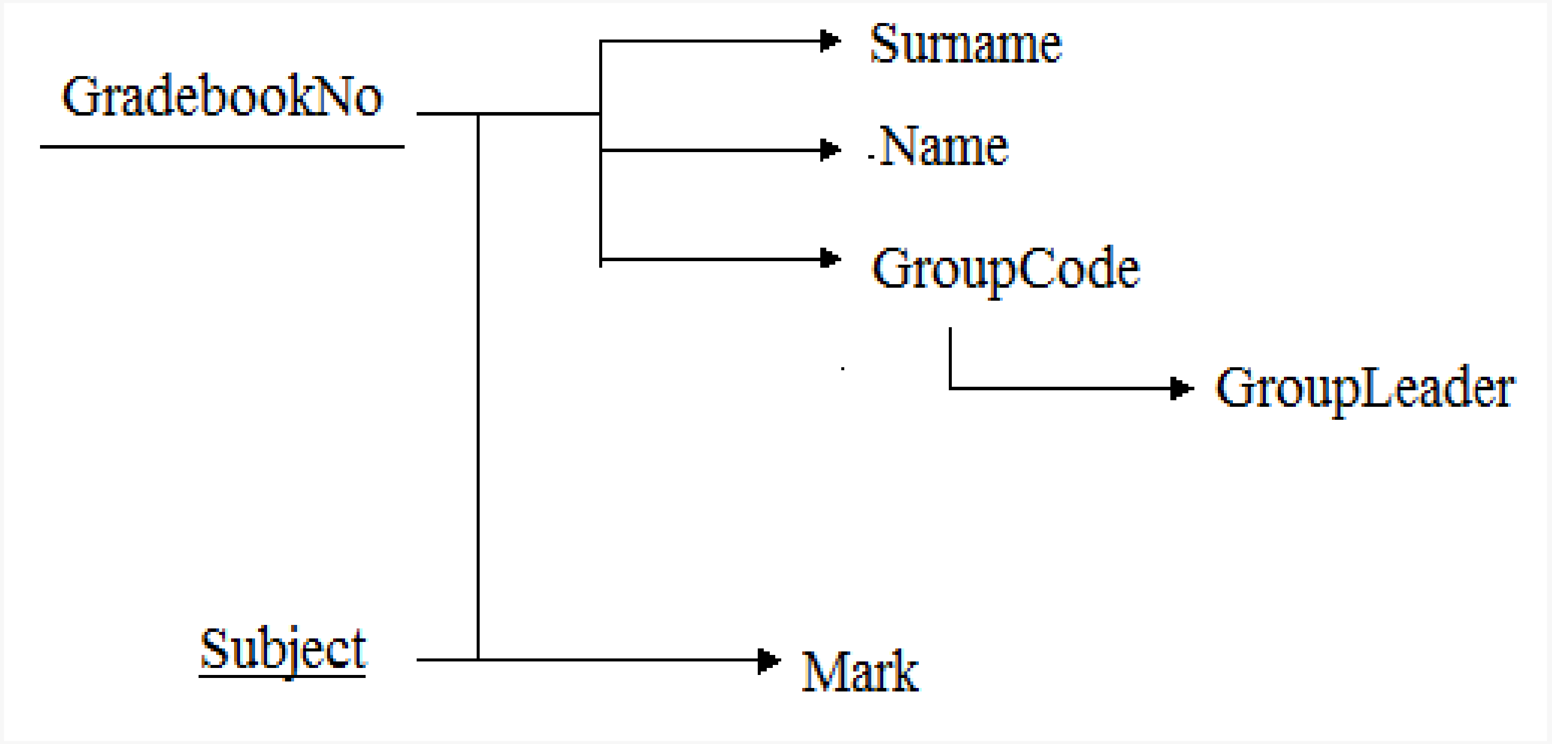
And we get a relation in 3 NF.

So, if the relation is neither in **2 NF** nor in 3 NF, there is **redundancy** that leads to so-called **update anomalies**. It leads to the changes in integrity when ***inserting, deleting or modifying data***.

For example, consider a relation that is neither in 2 NF nor in 3 NF.

The Academic progress (*GradebookNo, Surname, Name, GroupCode, GroupLeader, Subject, Mark*).

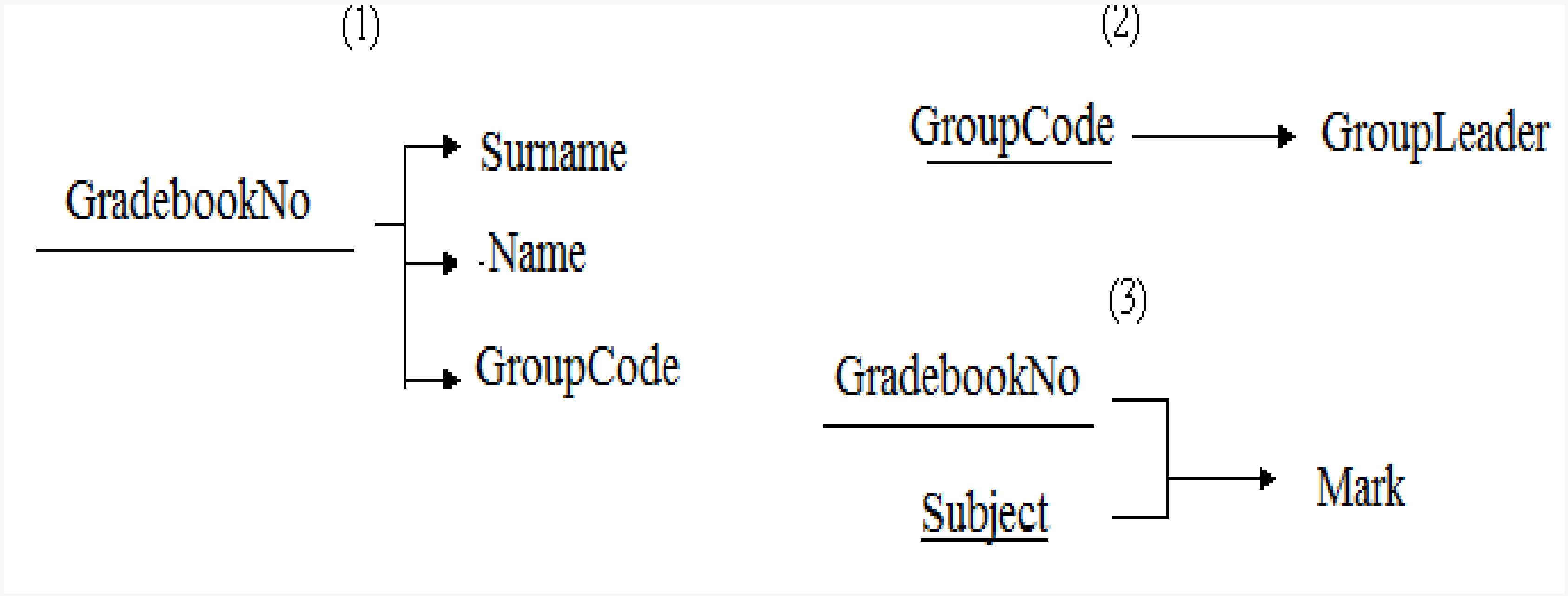
Let's draw a diagram of the FD of this relation:



What anomalies can occur when updating data in this relation?
For example, such as:

- When adding a new student, if the group in which he is studying is correctly specified, the leader of the group may be mistakenly indicated.
- When you delete one of the records, not only the information about the mark for the corresponding student will be deleted, but also the information about which group he is studying in.
- If you change the leader of a group, you should manually change it for all records of students in this group - you can make a mistake.

For this, the relation can be decomposed. It is convenient to do this following the diagram. For example, the following three projections:



All obtained relations are in 2NF and in 3NF.

But is this decomposition was without losses?
Yes, because after joining these three relations, we get the original one.

The resulting relations are independent projections. The projections R1 and R2 of the relation R are independent if and only if two conditions are met:
1) each FD in relation R is a logical consequence of the FD in the projections R1 and R2;
2) common attributes of the projections R1 and R2 form a potential key for at least one of them.
For our three projections, the first condition is met, since all of their FD (indicated by arrows) provide the FD of the original relation. The second condition is also met: for projections (1) and (2) the common attribute is GroupCode, it is the primary key for (2); for (1) and (3)- the common