Relational logical model

Normalization and denormalization

Data Models

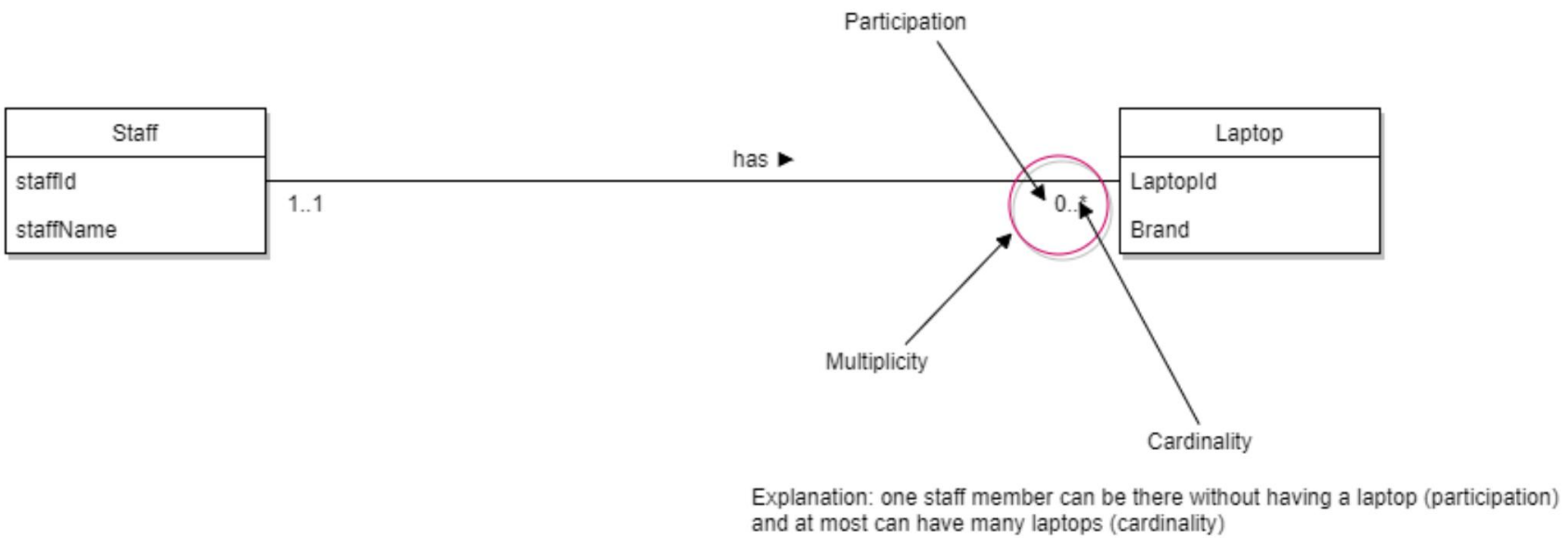
- *Data model* is an abstract, independent, logical definition of data structures, data operators, and other things that together make up *the abstract system* with which the user interacts.

Relational table (relation)

- ***A relation*** is a set of elements called
  in motorcades.
- **Relation—** A flat table consisting of columns and rows
- **Tuple** - A string of a relationship.
- In mathematics, a tuple is a sequence of a finite number of elements. • Many mathematical objects are formally defined as tuples.
  For example, a graph is defined as a tuple (V,E), where V is the set of vertices and E is a subset of V × V denoting edges.
- **Degree.** The degree of a relationship is determined by the number of attributes that it contains.
- **Cardinality** - The number of tuples that a relation contains.
- **The multiplicity** (multiplicity) **of a relationship** is a characteristic that indicates how many attributes of an entity class with a given role can or should participate in each instance of a relationship of any type.
- **Cardinality** of **a relationship** Indicates the maximum number of possible relationships in which a certain entity can participate (no more than)

Multiplicity and Cardinality

- Multiplicity = Cardinality + Participation

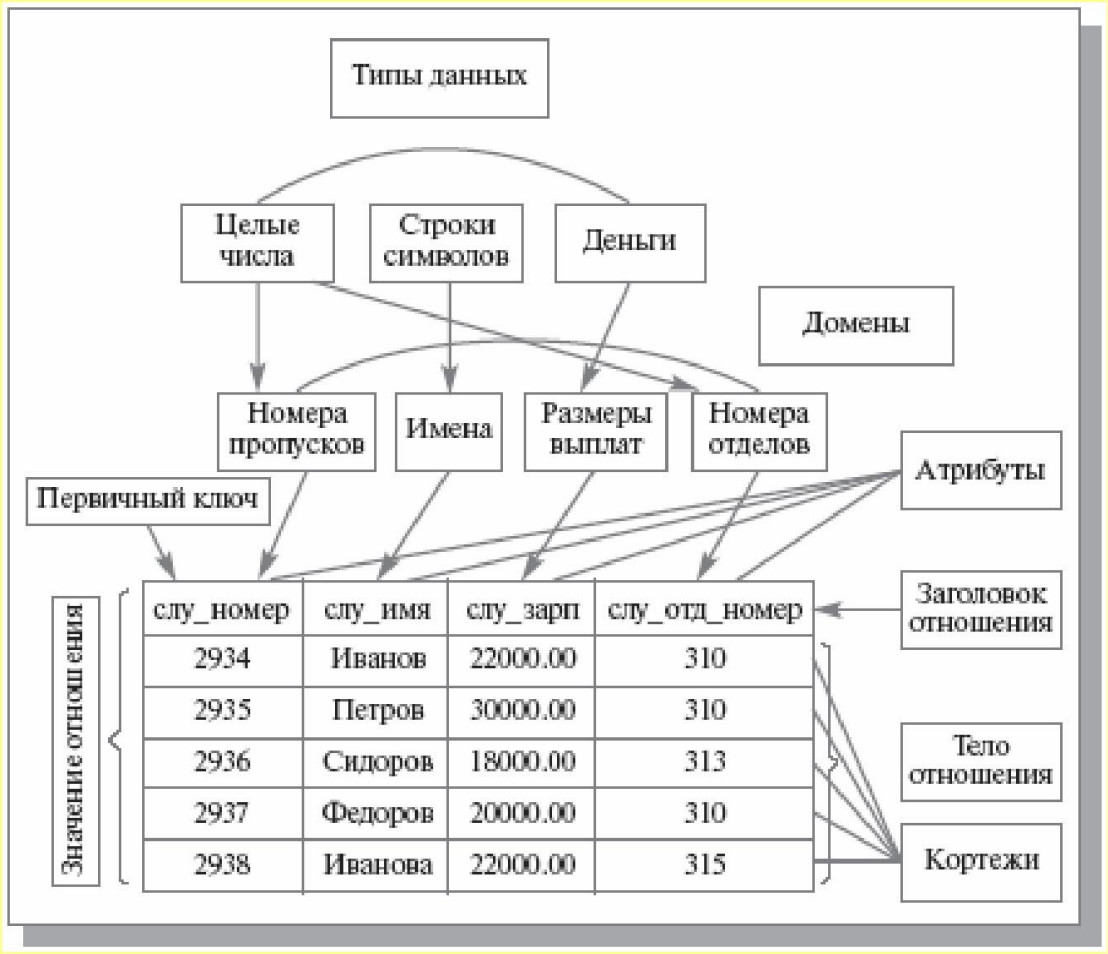


- https://martinfowler.com/bliki/Multiplicity

Alternative names

Official terms Alternative option 1		Alternative Option 2
Relation	**Table**	File
tuple	Row	Record
attribute	Column	field

Relational Table: What is What



Properties of a relational table

1. A table is a two-dimensional structure consisting of rows and columns.

2. Each row of the table (tuple) represents a separate entity within a set of entities
3. Each column of the table represents an attribute, and each the column has its own name
4. At each intersection of a row and a column there is a unique meaning
5. Each table must have an attribute or several attributes that uniquely identify each row.
6. All values in the column must be displayed in the same way. format. For example, if an attribute is assigned an integer format, then all values in the column representing that attribute must be integers.

7. Each column has a specific range of values, called the domain of the attribute
8. The order of rows and columns is not important.

Relational Database Keys

Key type	Definition
Superkey (superkey)	An attribute (or combination of attributes) that uniquely identifies each entity in a table
Candidate key	Minimal superkey. A superkey that does not contain a subset of attributes that is itself a superkey.
Primary key	A candidate key chosen to uniquely identify all other attribute values in any row. Cannot contain null values.
Secondary key	An attribute (or combination of attributes) used solely for the purpose of data retrieval
Foreign key	An attribute (or combination of attributes) in one table whose values must either match the values of the primary key of another table or be empty .

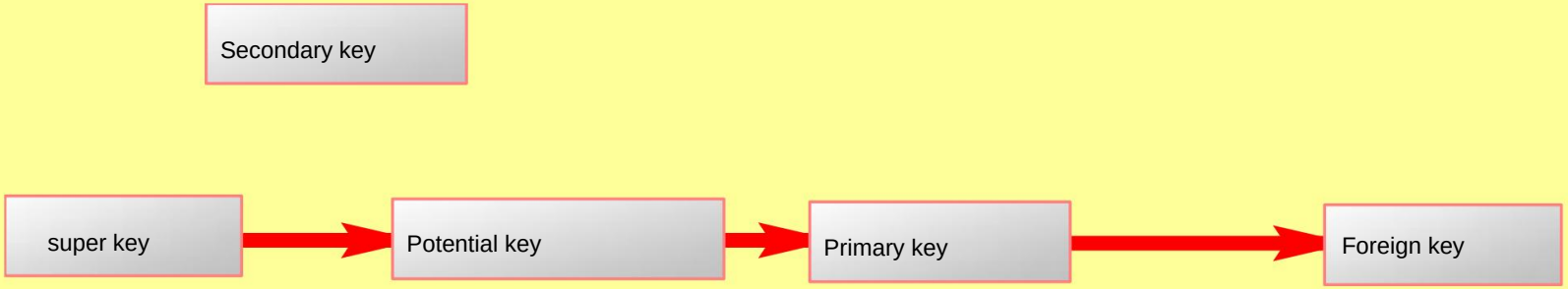
Keys example

Human

id inn last name	Name	surname	passport series	passport number	date of birth

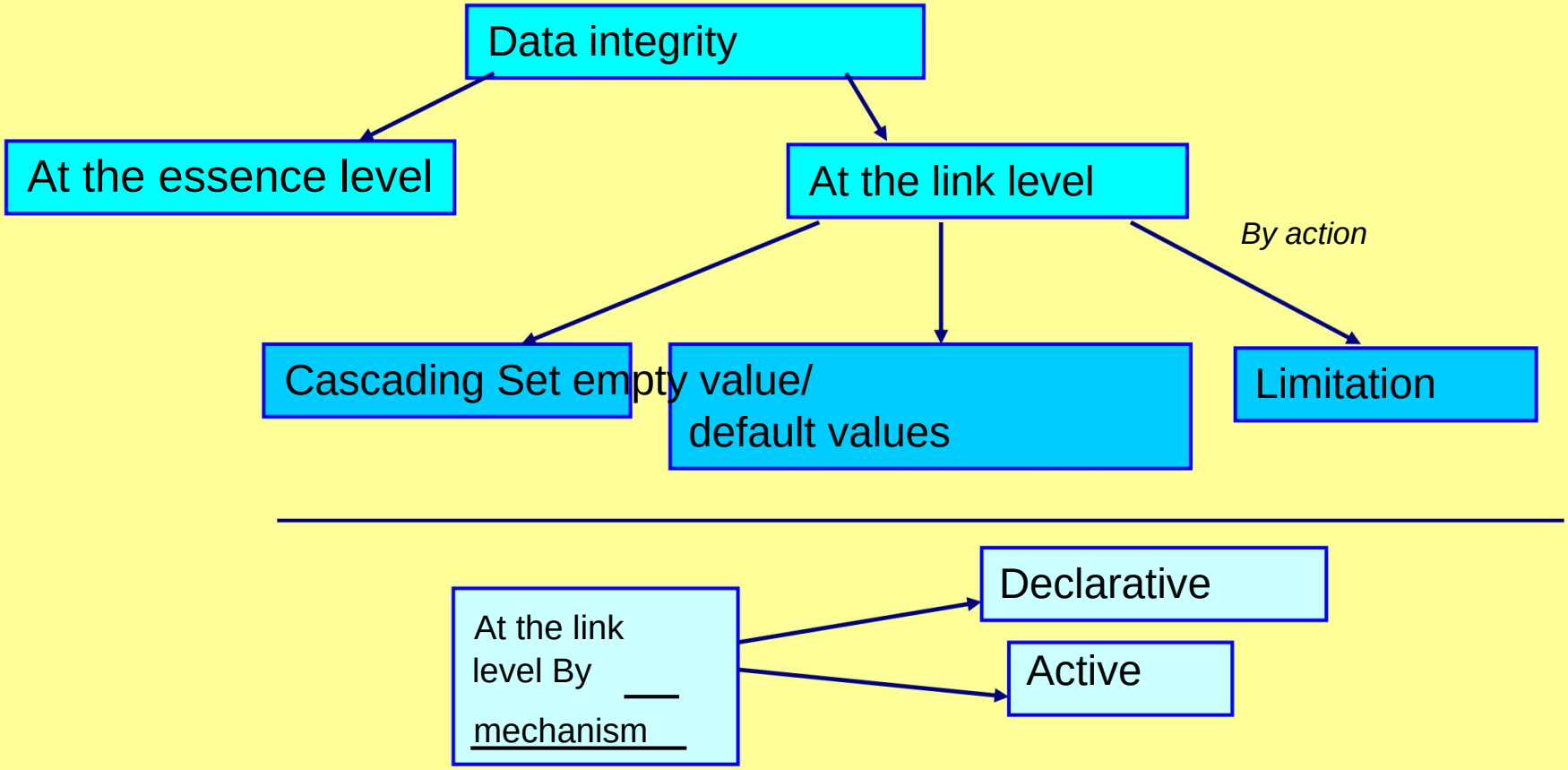
super key	**Potential key**	**Primary ny key**	**out of key**	**secondary ny**
id	id	id	id	f+i+o+ date of birth has
inn	inn			
full name + i + o + date of birth	full name + i + o + date of birth			
passport series + passport number	passport series + passport number			
f+i+o+INN				
f+inn				
...				
id+INN+ f+i+o+series+number+date of birth				

Keys



Data integrity

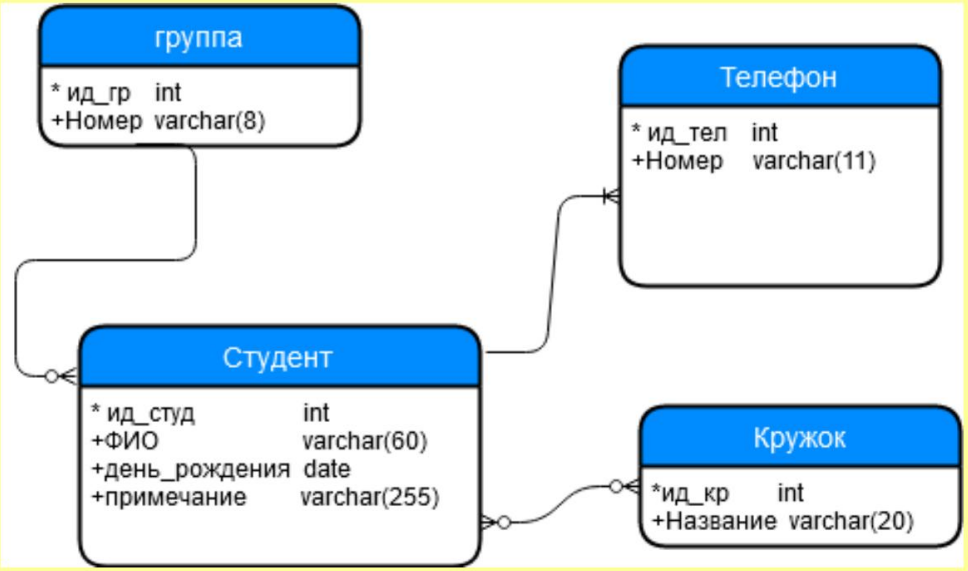
- **Empty value.** Indicates that the attribute value in the present moment is unknown or unsuitable for this tuple



Referential integrity

Designation save type	Code on SQL	What happens when deleting	What happens when you update
Restriction		Prevents deleting data from a parent table if it is linked to any data in a child table (check immediately)	Prevents changing the primary key from the parent table if any data in the child table is associated with it (check immediately)
Limitation	No action	Prevents deleting data from a parent table if it has any data associated with it in a child table (check deferred)	Prevents changing the primary key from a parent table if any data in the child table is associated with it (check deferred)
		When you delete data from a parent table, the related data in Cascading cascade subsidiary	When you change the primary key in the parent table, the foreign keys associated with it in the child table will change to same meaning
Installation	set null	When deleting data from When changing the primary key from the parent table, foreign parent table, foreign keys child table keys related child table related with the deleted data will receive an empty value (null)	the changed keys will receive an empty value (null)
Installation	set default	When deleting data from the parent table, the foreign keys of the child table are related to the parent table, the foreign keys of the child table are related to the with the deleted data will receive default value to be set in the child table	changeable keys will receive default value to be set in the child table

Conceptual model (ER diagram)



Rules for constructing a relational logical model based on a conceptual

Entity/Relationship	Method of transformation
Strong essence	Create relationships that include all simple attributes
Weak essence	Create relationships that include all simple attributes (after transforming the relationship with each owner entity, you must also define a primary key)
Multi-valued attribute	Create a relationship representing a multivalued attribute and pass a copy the primary key of the owner entity to the new one relation to use as external the key

Character data types

- Char(7) •

Logging site → “Lesopov”

- Forest → “Forest_ _ _ _”

- Varchar(7) •

Lesopoval → "Lesopov"

- Forest → “Forest”

Character Large object (Text) •up to 2 GB •No default value •Pattern search (like) not possible •Limited sorting and indexes

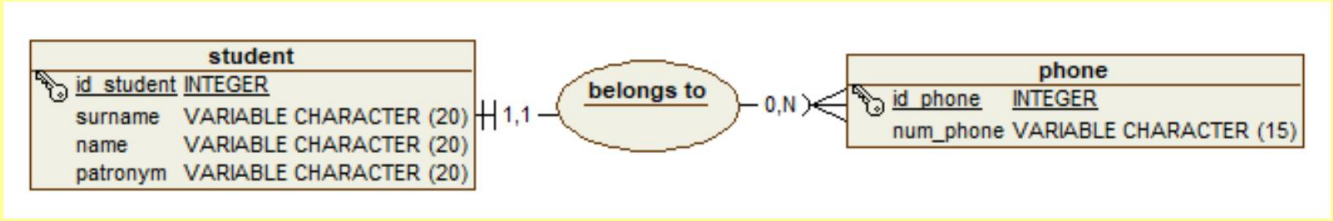
Rules for constructing a relational logical model based on a conceptual

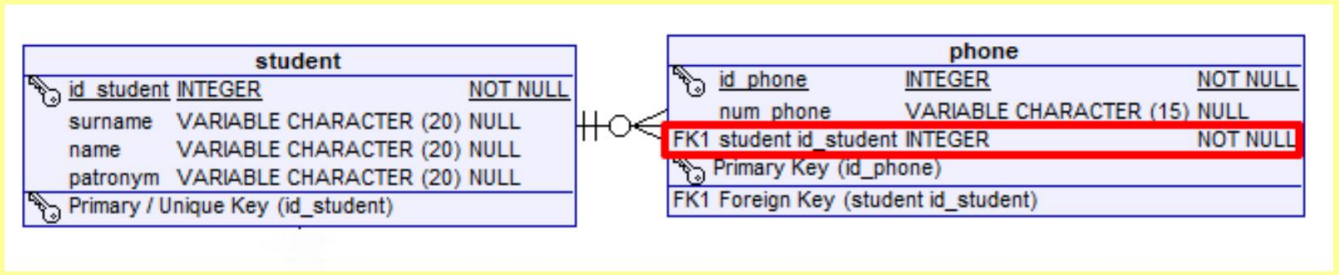
Connection	Method of transformation
Two-way communication type 1:*	Passing the primary key of an entity to the "one" side for used as a primary key in a relationship corresponding to an entity on the many side. All attributes of the relationship are also passed to the many side.
Two-way relationship of type *:*, complex relationship	Create a relation representing the relationship and include all the relationship attributes Pass a copy of the primary key from each owner entity to the new relation for use as foreign keys
Two-way communication type 1:1:	
mandatory participation of both parties	Combining entities into one relationship
mandatory participation of one party	Passing the primary key of an entity to "optional" side for use as a foreign key in a relationship representing an entity on a "required" side
optional participation of both parties	If there is no additional information, then the choice becomes arbitrary

Two-way communication type 1:*

Passing the primary key of an entity to the "one" side for use in

as the primary key in the relationship corresponding to the entity on the many side. All attributes of the relationship are also passed to the many side.



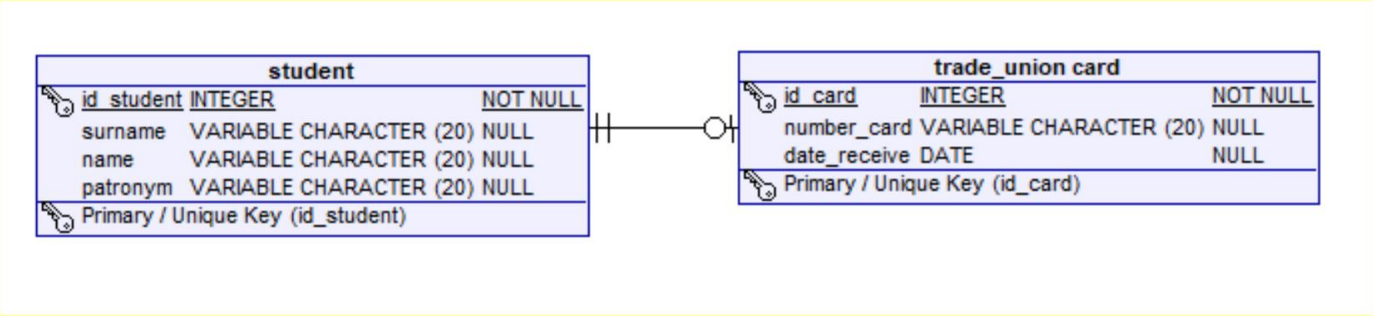


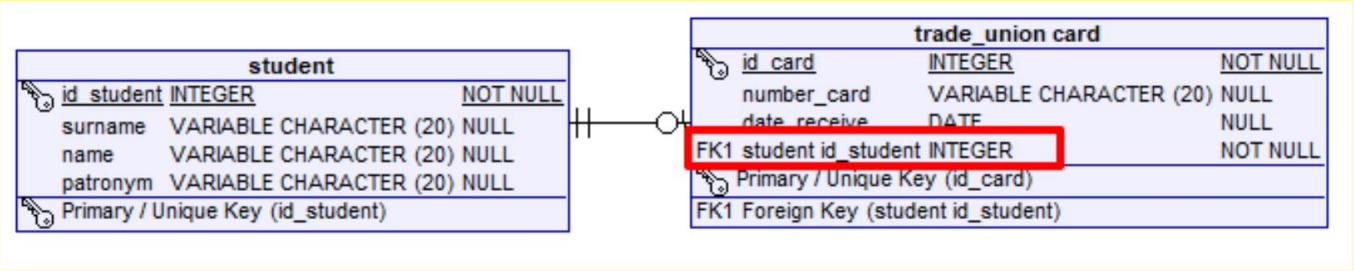
1:1 mandatory participation of one party

Passing the primary key of an entity to "optional"

side for use as a foreign key in a relationship representing an entity on the

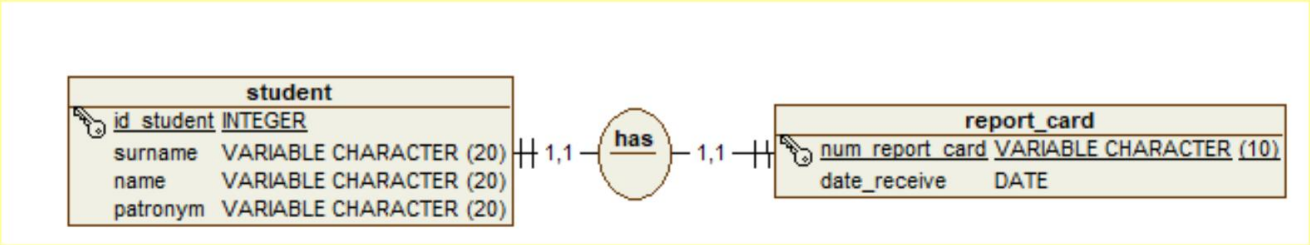
"required" side

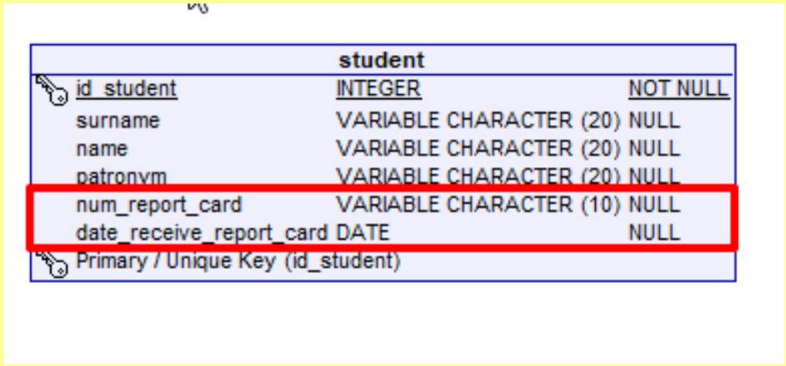




1:1 mandatory participation of both parties

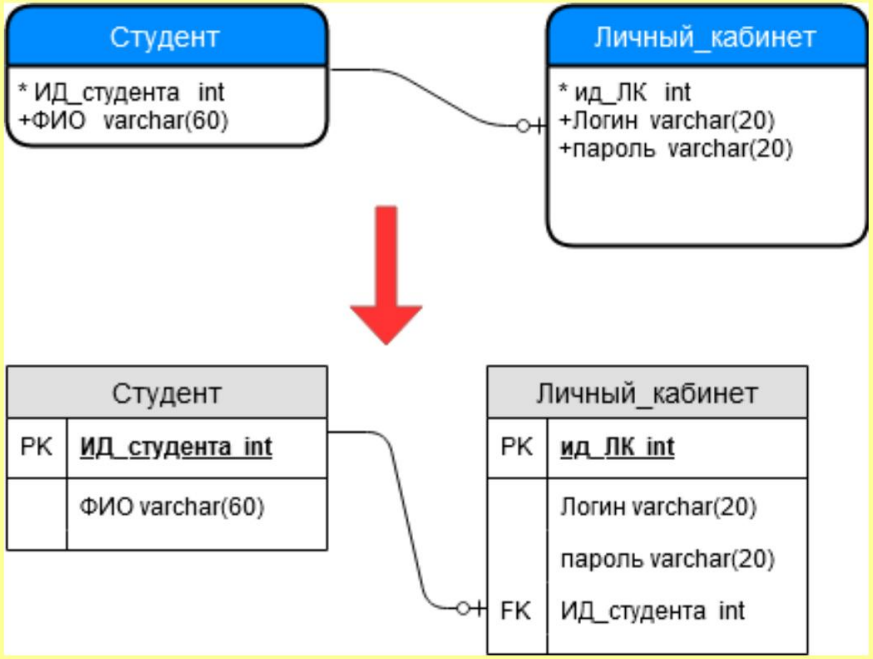
Combining entities into one relationship

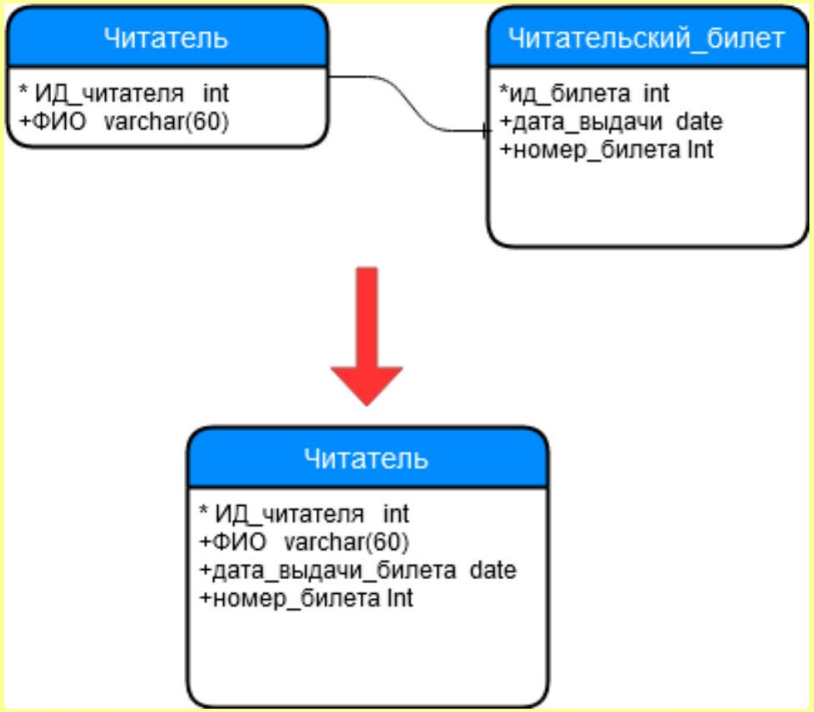




Logical 1:1

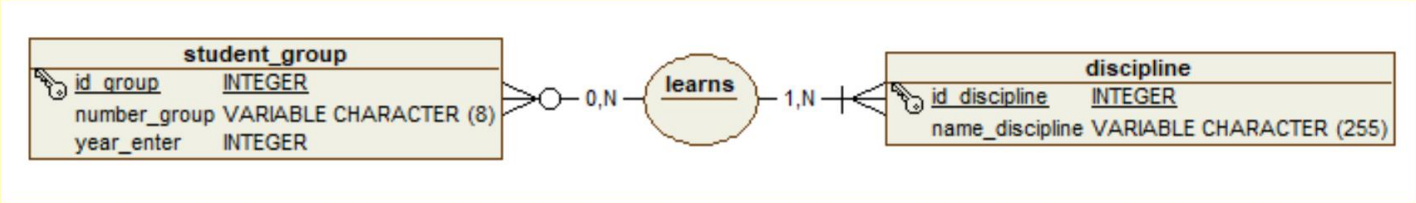
One side is mandatory Both sides are mandatory

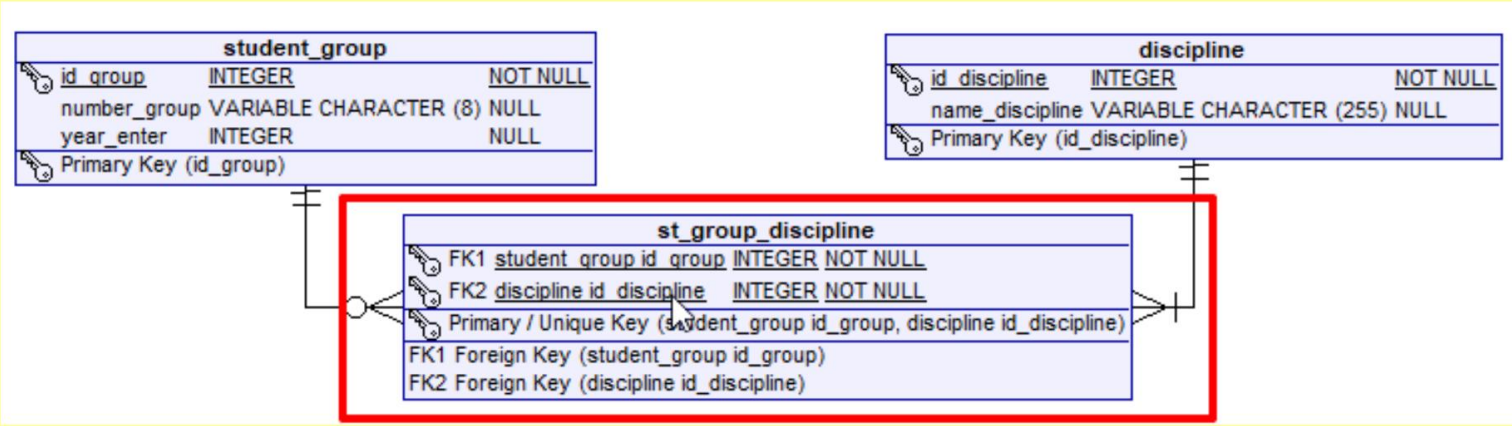




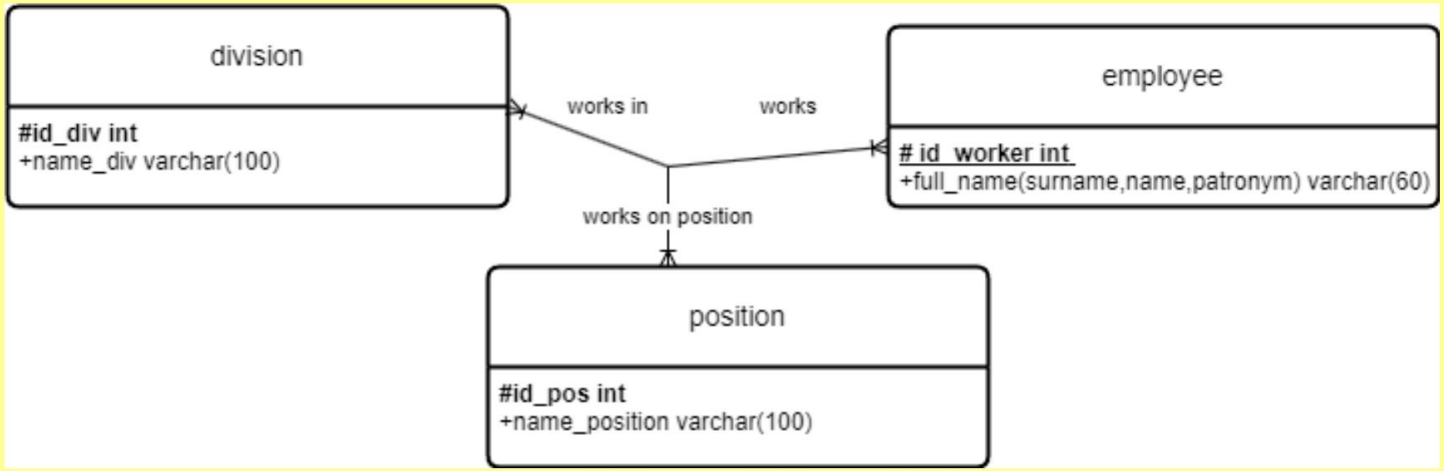
Two-way communication type *:*

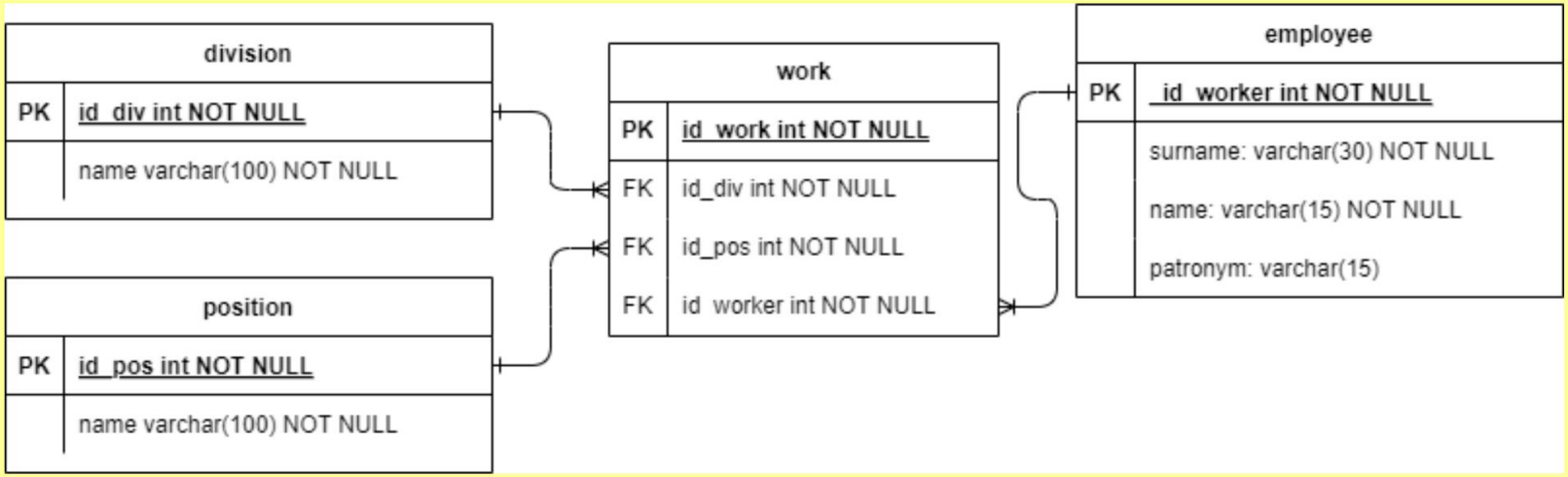
Create a relation representing the relationship and include all the relationship attributes Pass a copy of the primary key from each owner entity to the new relation for use as foreign keys





complex connection (triple)





Keys by origin

- **Natural Key (NK)** – a set of attributes of the entity described by the record that uniquely identifies it (for example, a passport number for a person)
- **Surrogate Key (SK)** – automatically

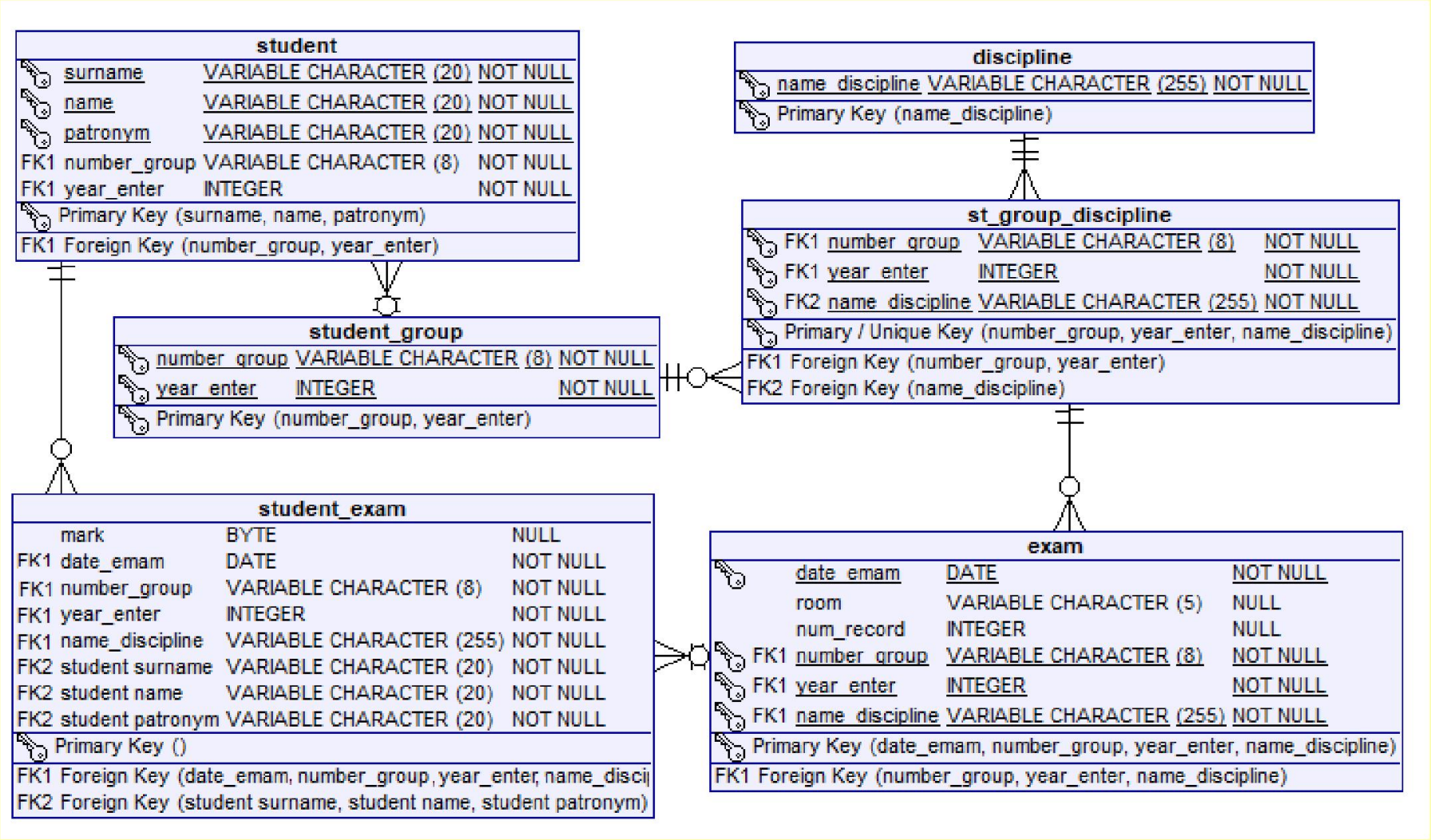
  a generated field that is not related to the information content

  of the record. Usually, the role of the SC is played by an

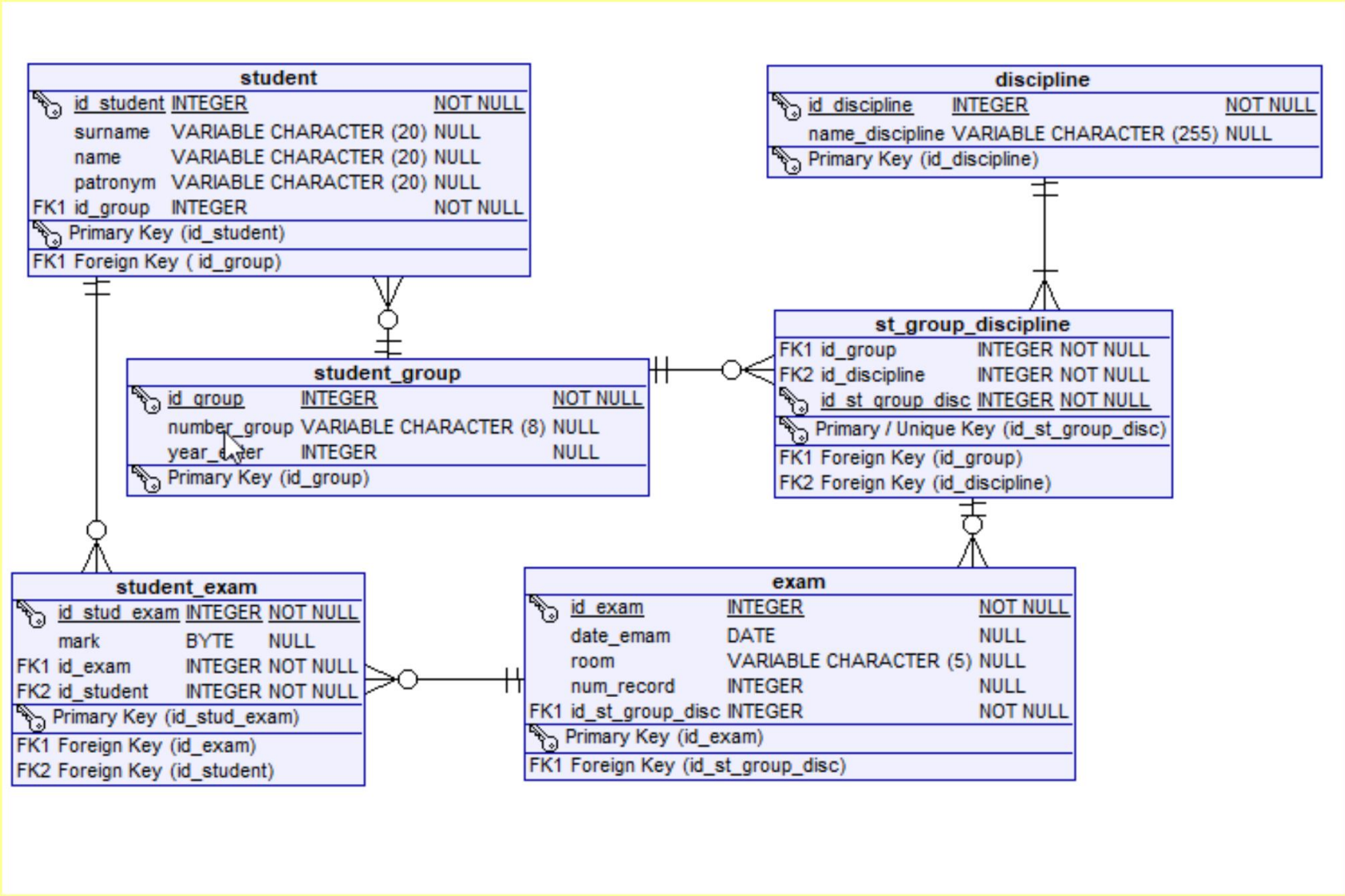
  auto-incrementing field of the INTEGER

  type.

"Attribute Race" Natural Keys



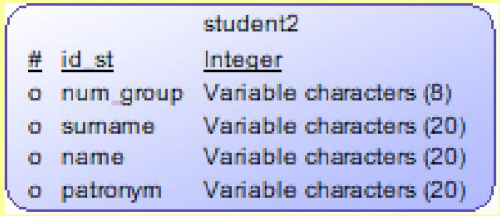
"Attribute Race" Surrogate Keys



Design issues

- Update anomalies
- Removal anomalies
- Insertion anomalies
- Large memory usage

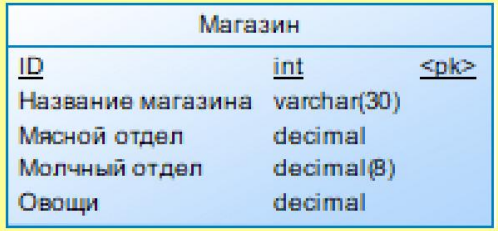
  under data

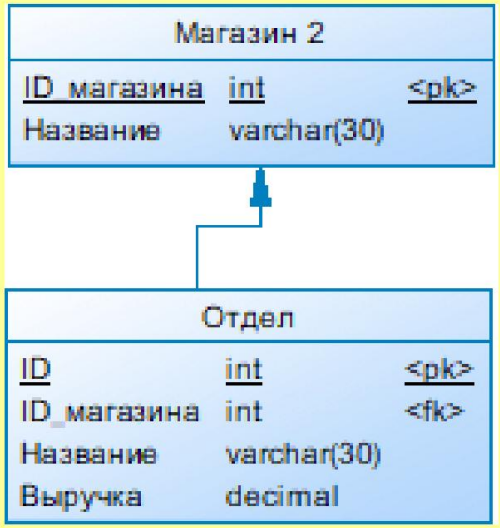


Type	Place (bytes)
TINYINT	1
SMALLINT	2
MEDIUMINT	3
INT	4
BIGINT	8

Line length	Place for one byte ow ow lines (bytes)	Place for two bytes lines (bytes)
		2
1 2	1 2	4
8	8	16
10	10	20
20	20	40

What is the best way to formalize the subject area?





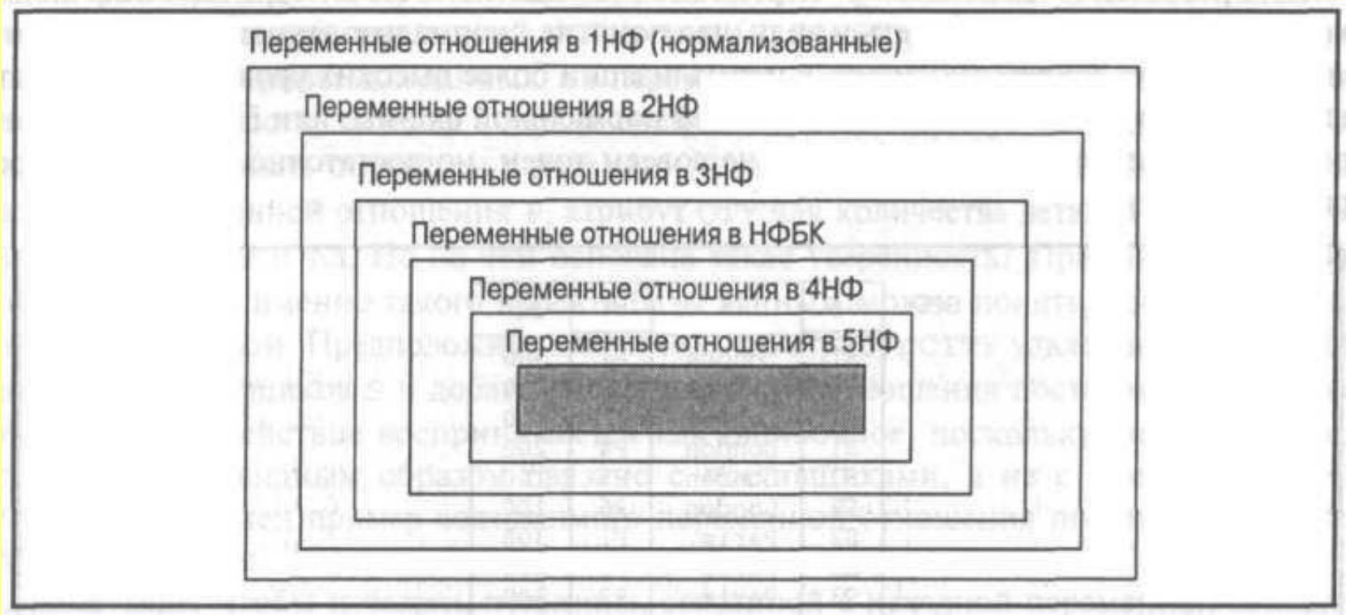
Normalization

- **Data normalization** is the process of bringing a model to a form that allows one to obtain a database structure in which storage redundancy is eliminated and anomalies when adding, deleting, and changing data are minimized .

- The normalization process is carried out in stages.
  At each stage, a certain restriction is imposed on the base structure and a certain defect is corrected in the base structure. A base with the corresponding restrictions is said to be in one of **the normal forms.**

Normal forms

1. First Normal Form (1NF) 2. Second Normal Form (2NF) 3. Third Normal Form (3NF) 4. Boyce-Codd Normal Form (BCNF) 5. Fourth Normal Form (4NF) 6. Fifth Normal Form (5NF) 7. Sixth Normal Form (6NF) 8. Domain-Key Normal Form (DKNF)



Unnormalized relation

- **Unnormalized form (UNF).**

  A table containing one or more

  repeating groups of data.

First normal form

- **First normal form (1NF).**

  A relationship in which at the intersection of each row and each column contains one and only one atomic meaning.

Possible violations of 1NF:

- The value can be composite (full name) •

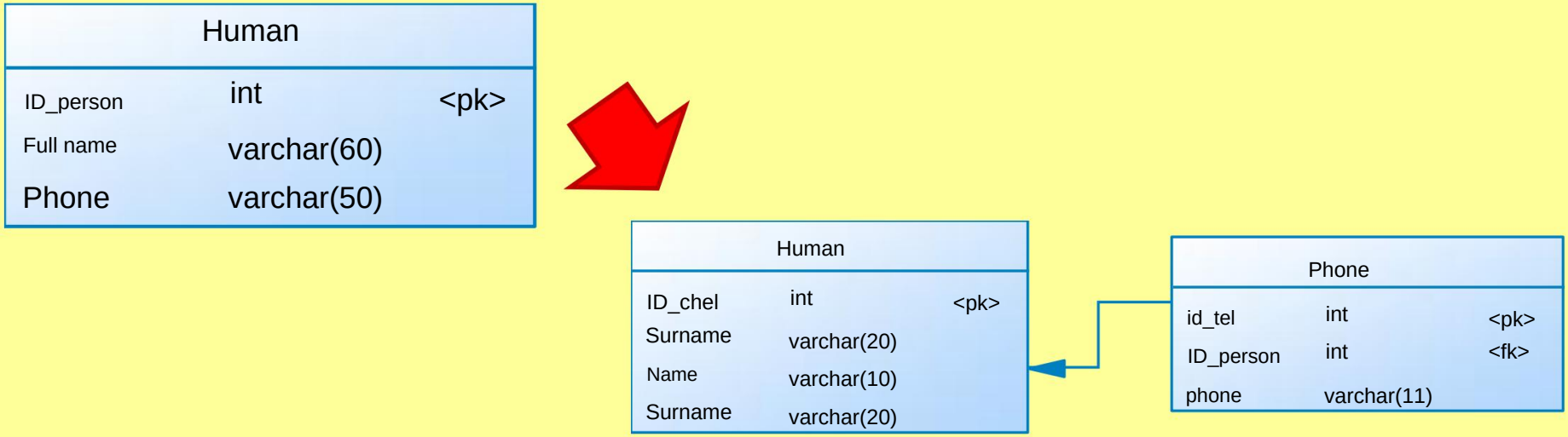
multi-valued (list of phone numbers)

How to reduce to 1 NF?

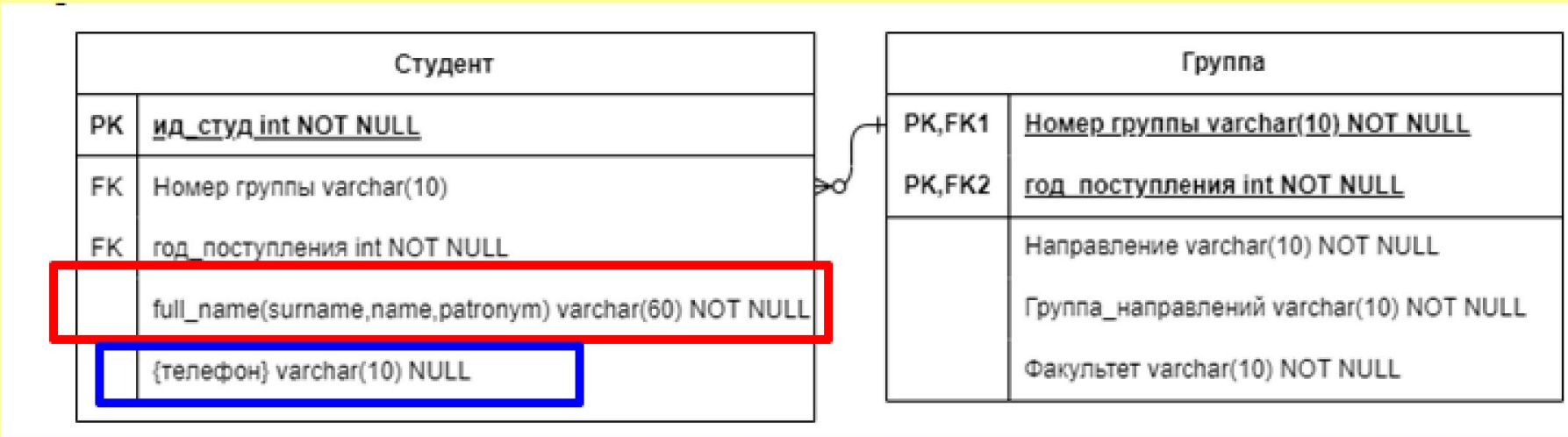
- Composite field - split into elements • Multi-valued attribute -

create a relation (table) representing the multi-valued attribute and pass a copy of the primary key of the owner entity to the new relation for

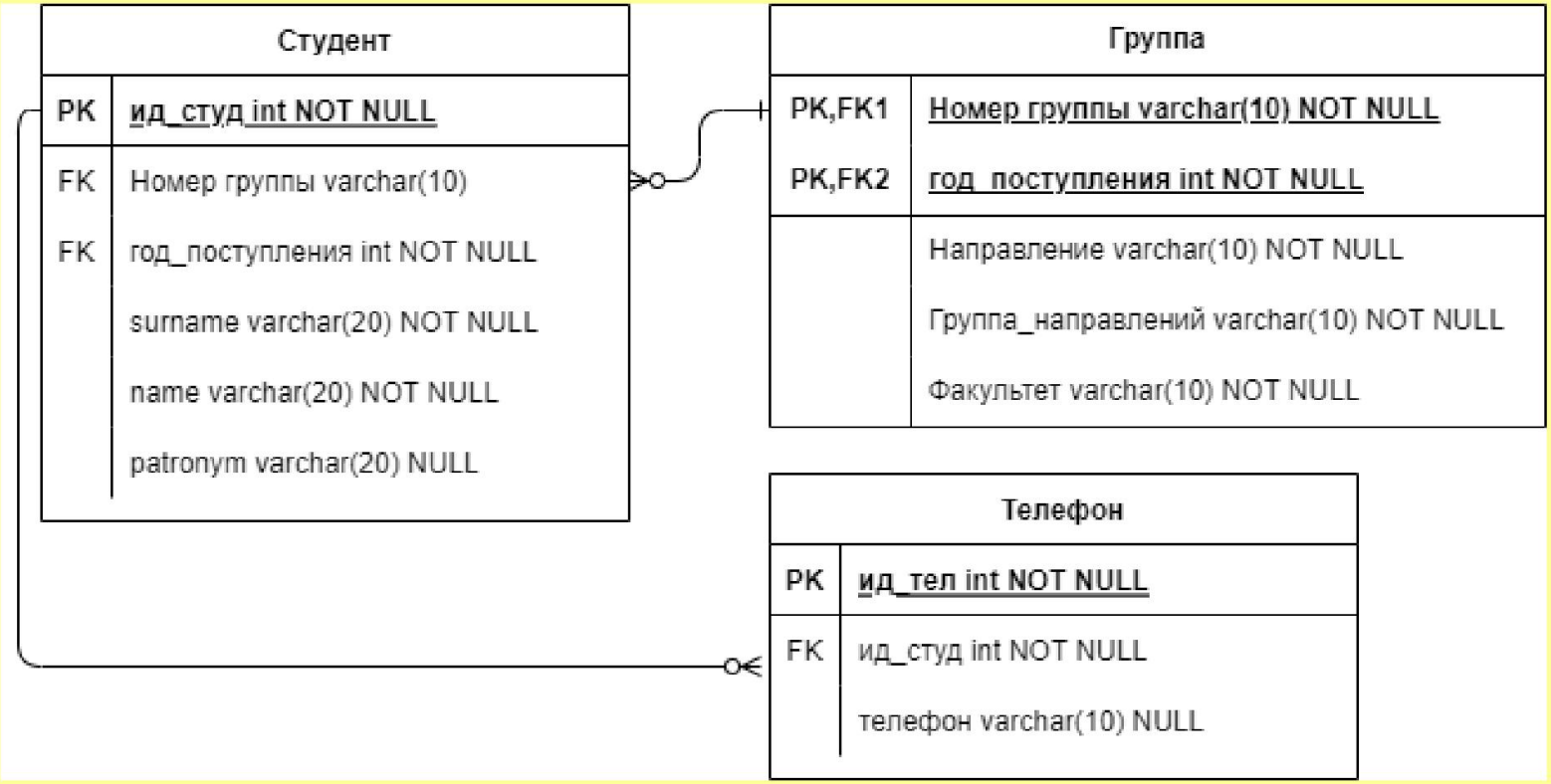
use as a foreign key



Unnormalized base data

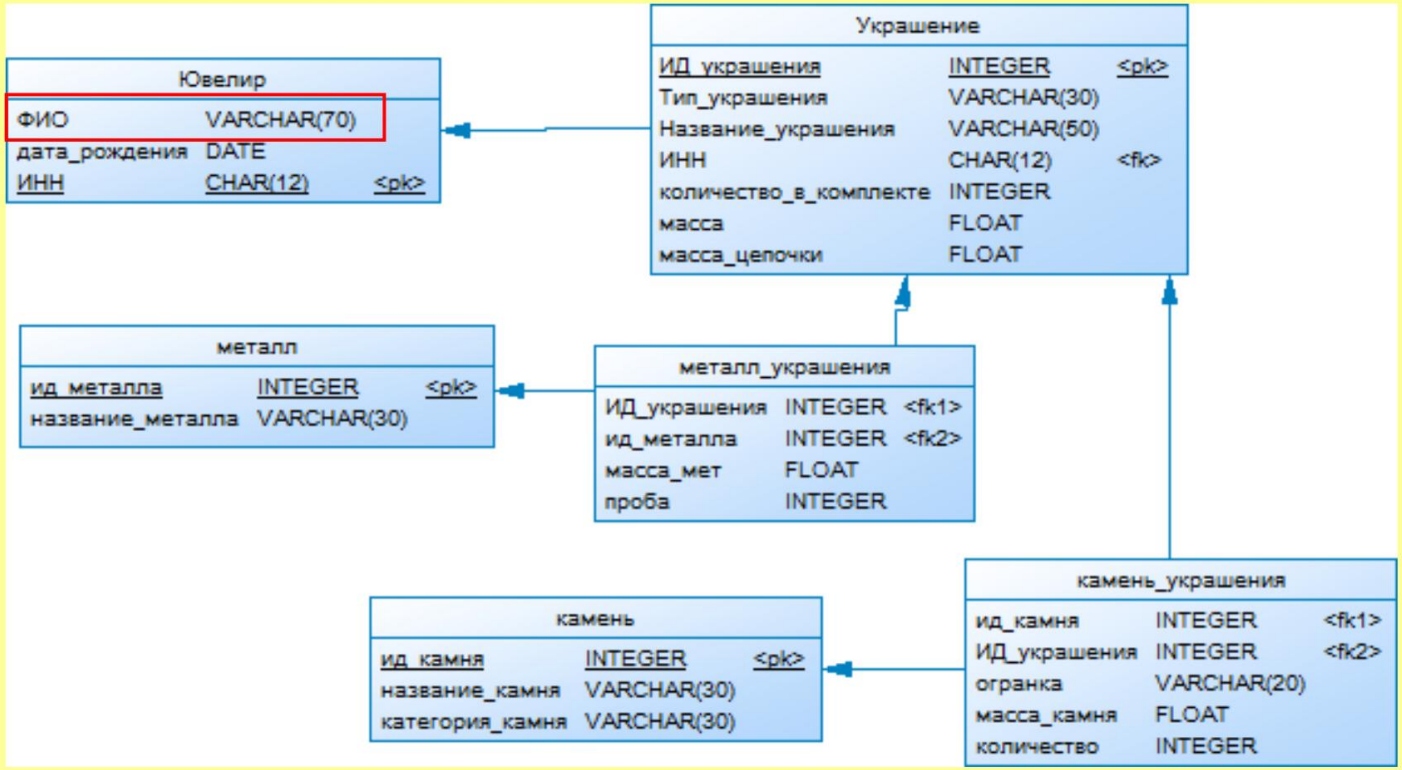


Database in 1NF

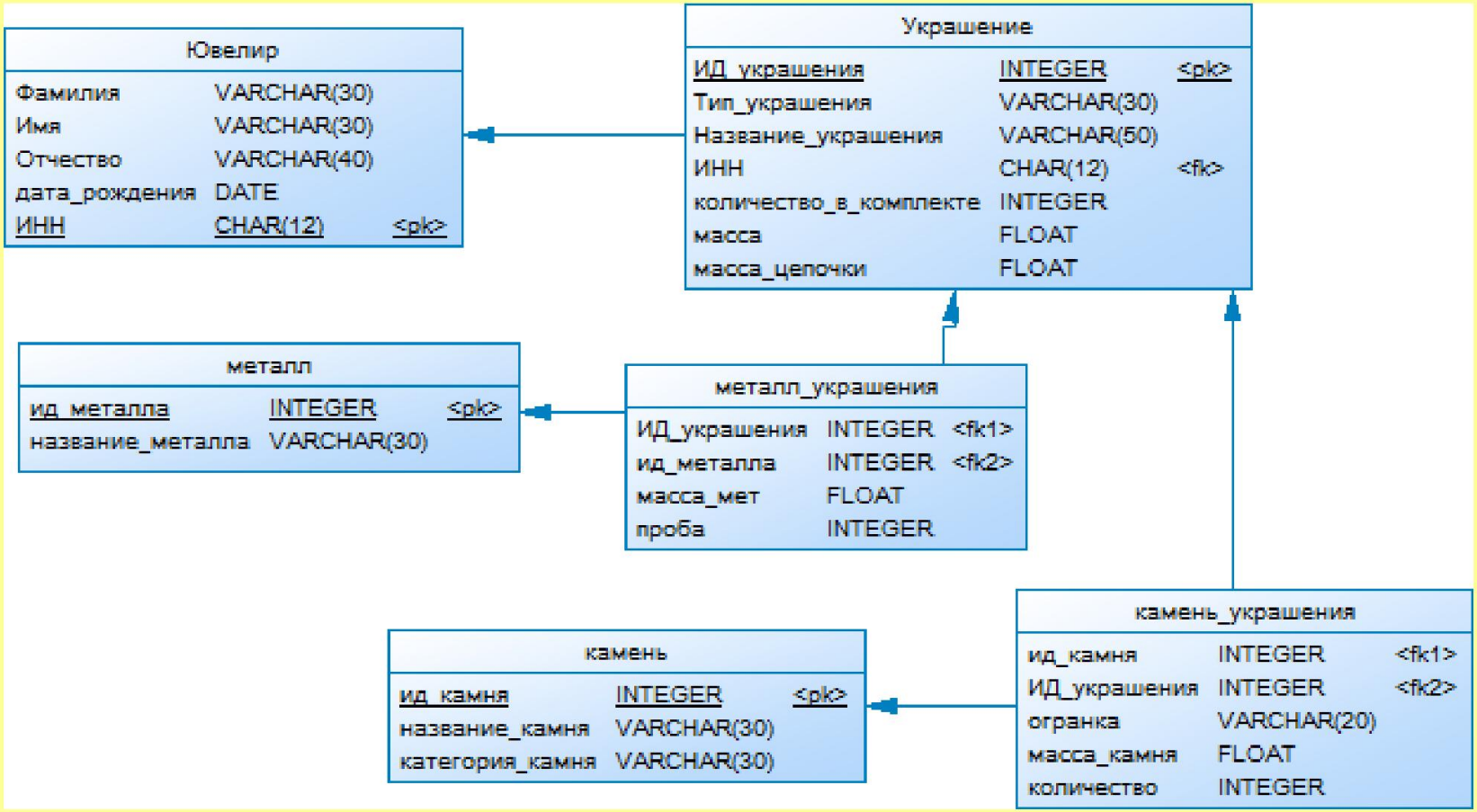


Unnormalized base

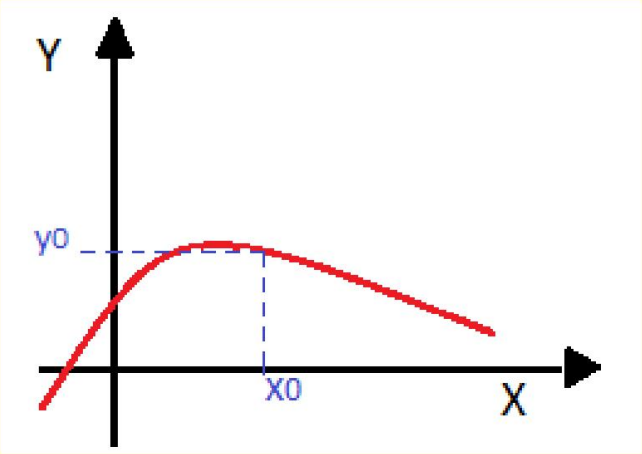
data



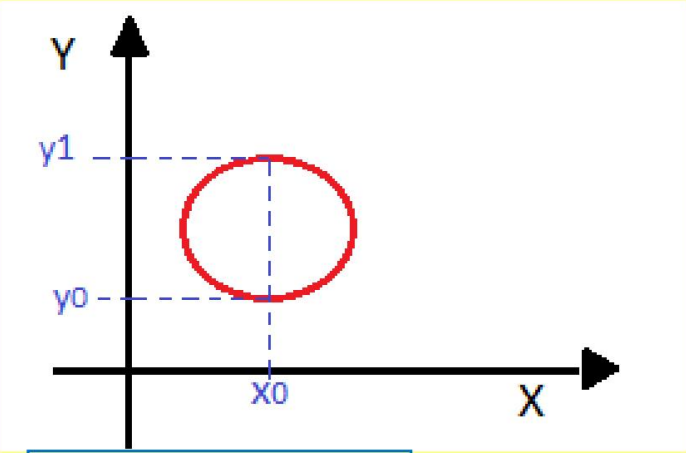
Database in 1NF



Functional dependency

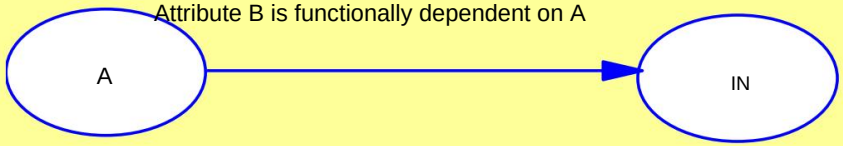


Y-function (Y depends on x)



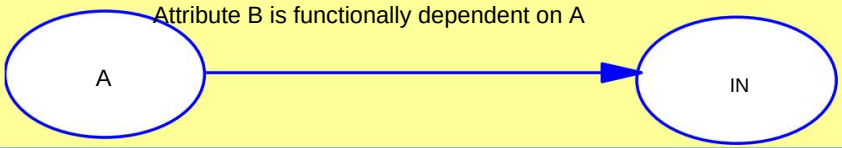
Y- NOT function

- **Functional dependency.** Describes the connection between the attributes of a relationship. For example, if in a relationship R containing attributes A and B, attribute B is functionally dependent on attribute A (which is denoted as A(B)), then each value of attribute A is associated with only one value of attribute B. (Moreover, attributes A and B can consist of one or more attributes.)



Functional dependency diagram

Functional dependency



Functional dependency diagram
(attribute A is a determinant of attribute B)

- **Determinant.** The determinant of a functional dependency is an attribute or group of attributes located on the functional dependency diagram to the left of the arrow.
- A functional dependency is called ***trivial*** if it remains true under all conditions. • the dependence is trivial

if and only if the right-hand side of the expression defining the dependence contains a subset

(but not necessarily a proper subset) of the set that is specified on the left-hand side (determinant) of the expression

- The functional dependence A ÿ B is ***complete*** functional dependency if the removal of any attribute from A results in the loss of this dependency. A functional dependency A ÿ B is called ***partial*** if A contains some attribute such that, when removed, this dependency is preserved.

Armstrong's Axioms

- A set of inference rules, called *Armstrong's axioms,* show ways to derive new functional dependencies from given ones.
- Assume that A, B and C are subsets of attributes R relations.

Armstrong's axioms:

- 1. Reflexivity. If B is a subset of A, then AÿB.
- 2. Complement. If A ÿ B, then A, C ÿ B, C.
- 3. Transitivity. If A ÿ B and B ÿ C, then A ÿ C.
- 4. Self-determination. A ÿ A.
- 5. Decomposition. If A ÿ B, C, then A ÿ B and A ÿ C.
- 6. Union. If A ÿ B and A ÿ C, then A ÿ B, C.
- 7. Composition. If A ÿ B and C ÿ D, then A, C ÿ B, D.

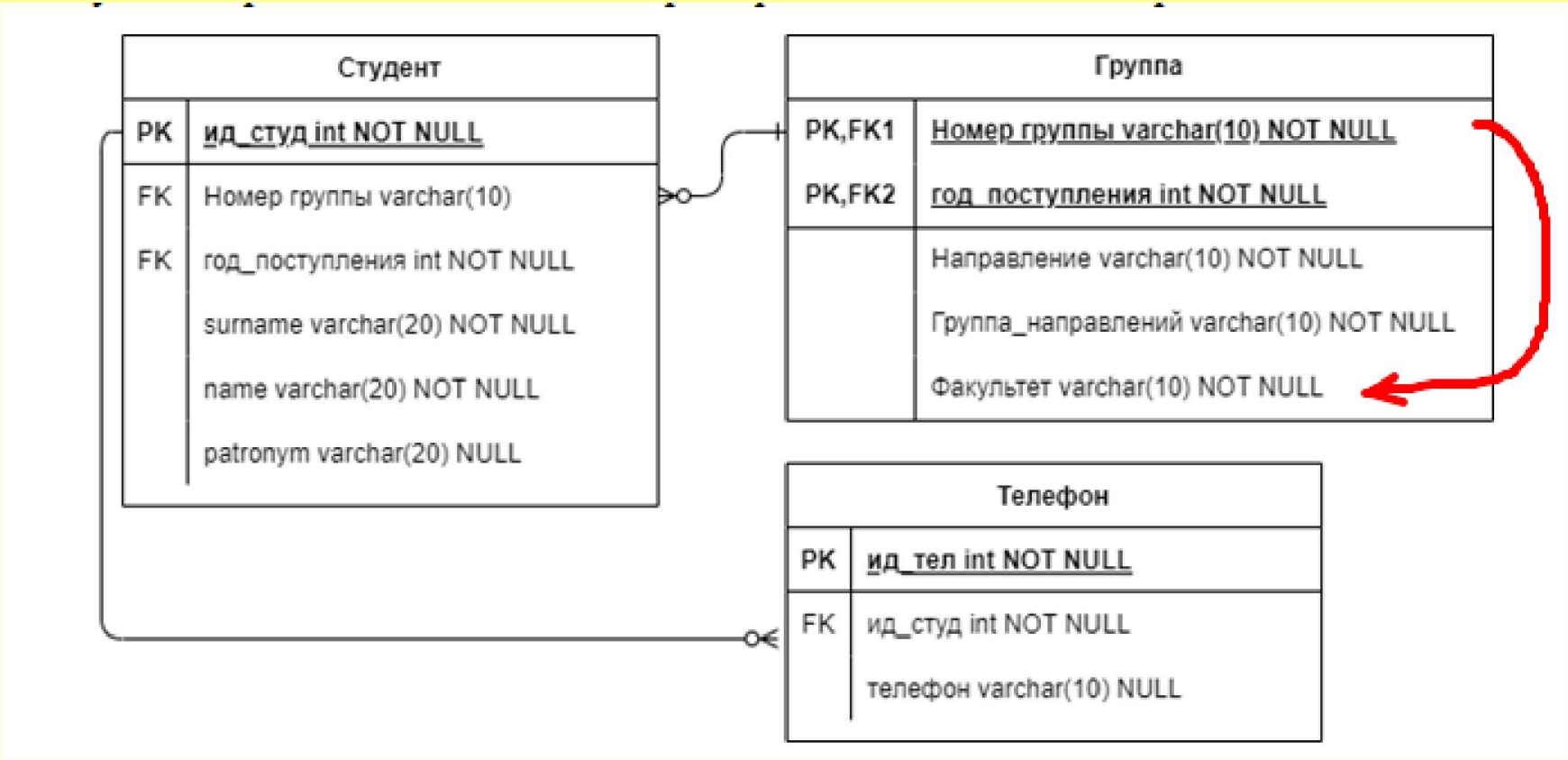
Second normal form (2NF)

- **Second Normal Form (2NF) -** Relation,
  in first normal form, in which every attribute other than the primary key
  attribute is completely functionally independent of any candidate key.

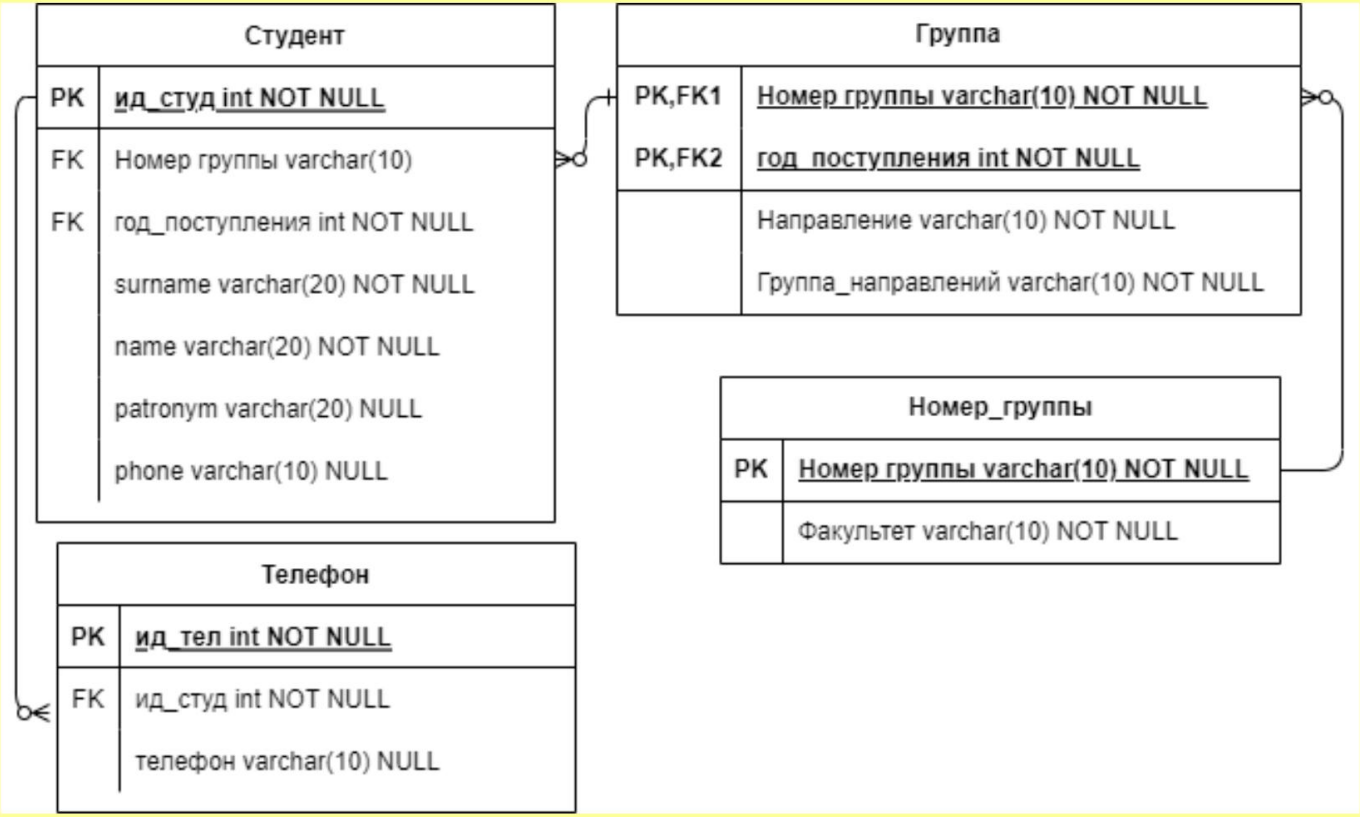
- Second normal form (2NF) - A relation that is in first normal form and every
  attribute of which that is not part of the primary key is characterized
  by a complete functional dependence on that primary key.

Second normal form applies to relations with composite keys, i.e., relations whose primary key consists of two or more attributes. A 1NF relation with a primary key based on a single
attribute is always at least in 2NF.

Database in 1NF

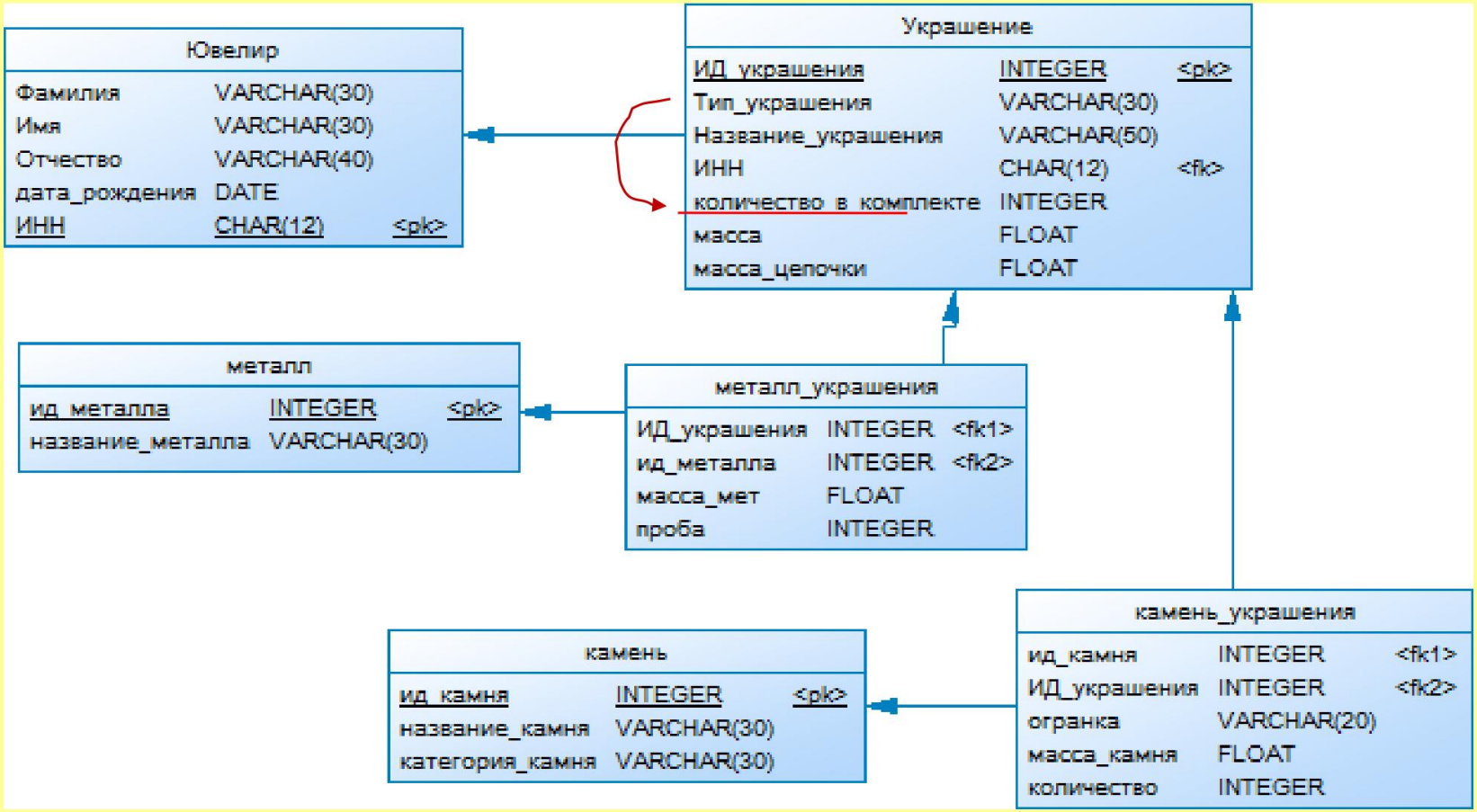


Database in 2NF

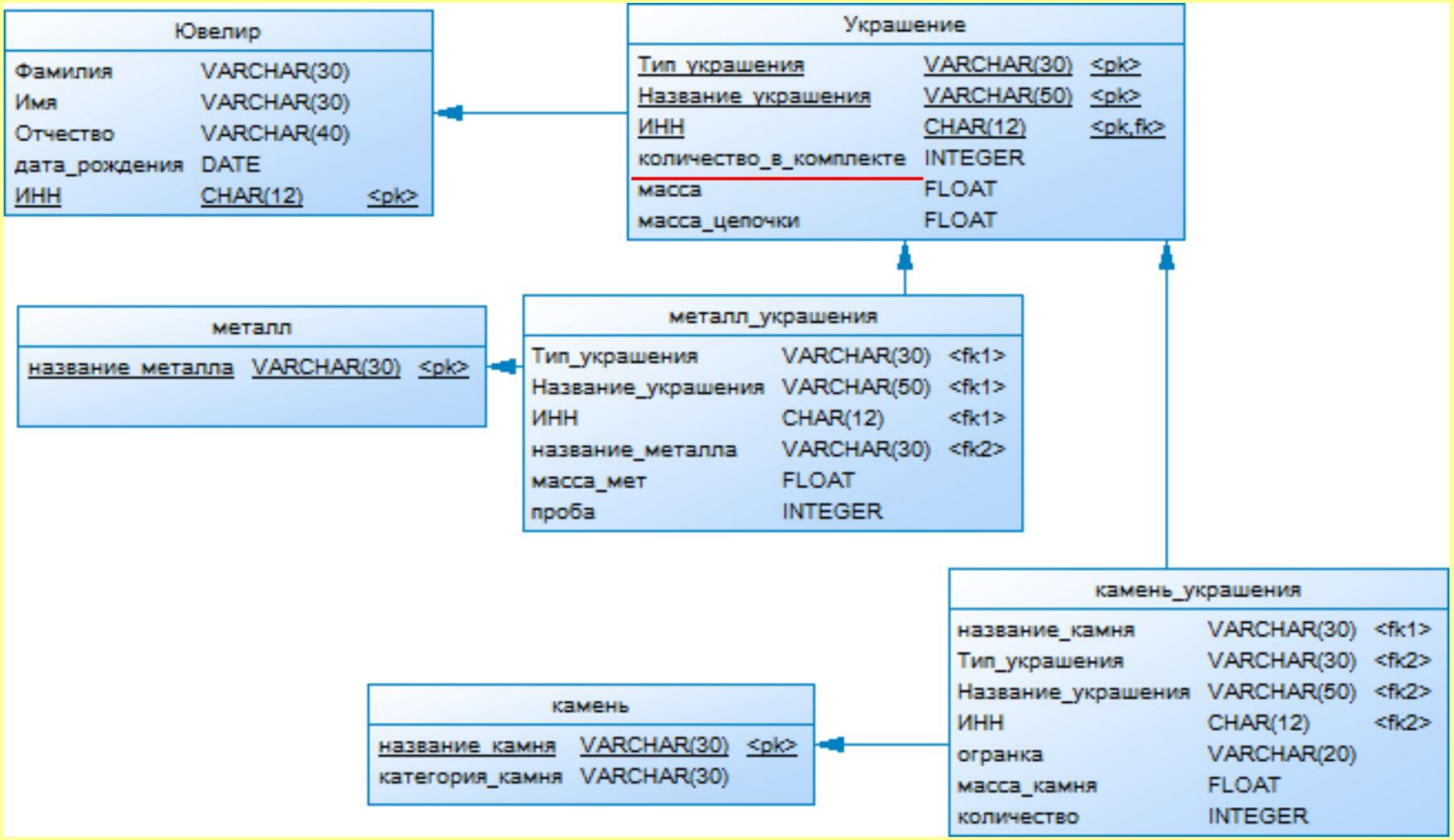


Reduction to 2NF

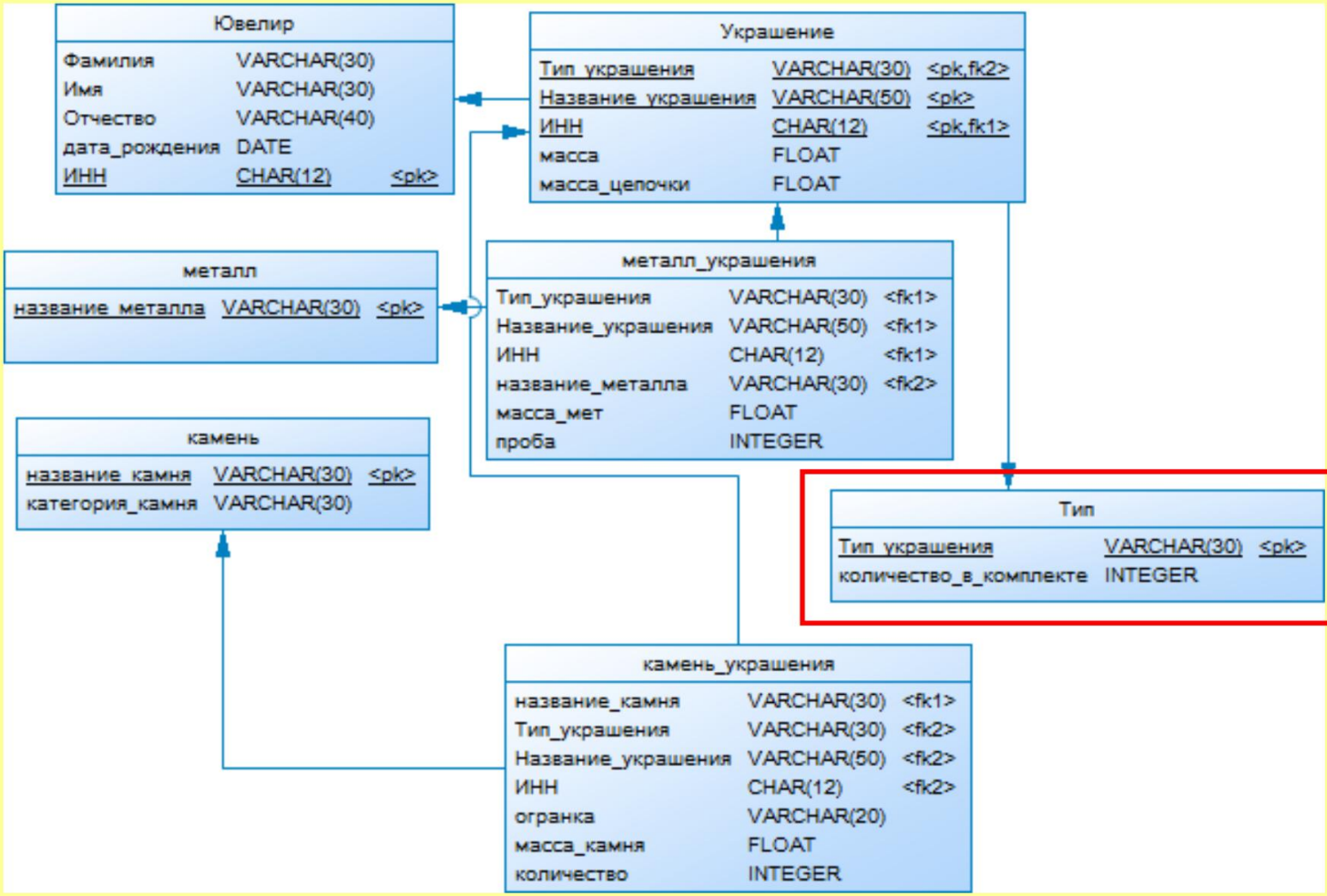
DB in 1NF



Database without surrogate keys



Database in 2NF without surrogate keys



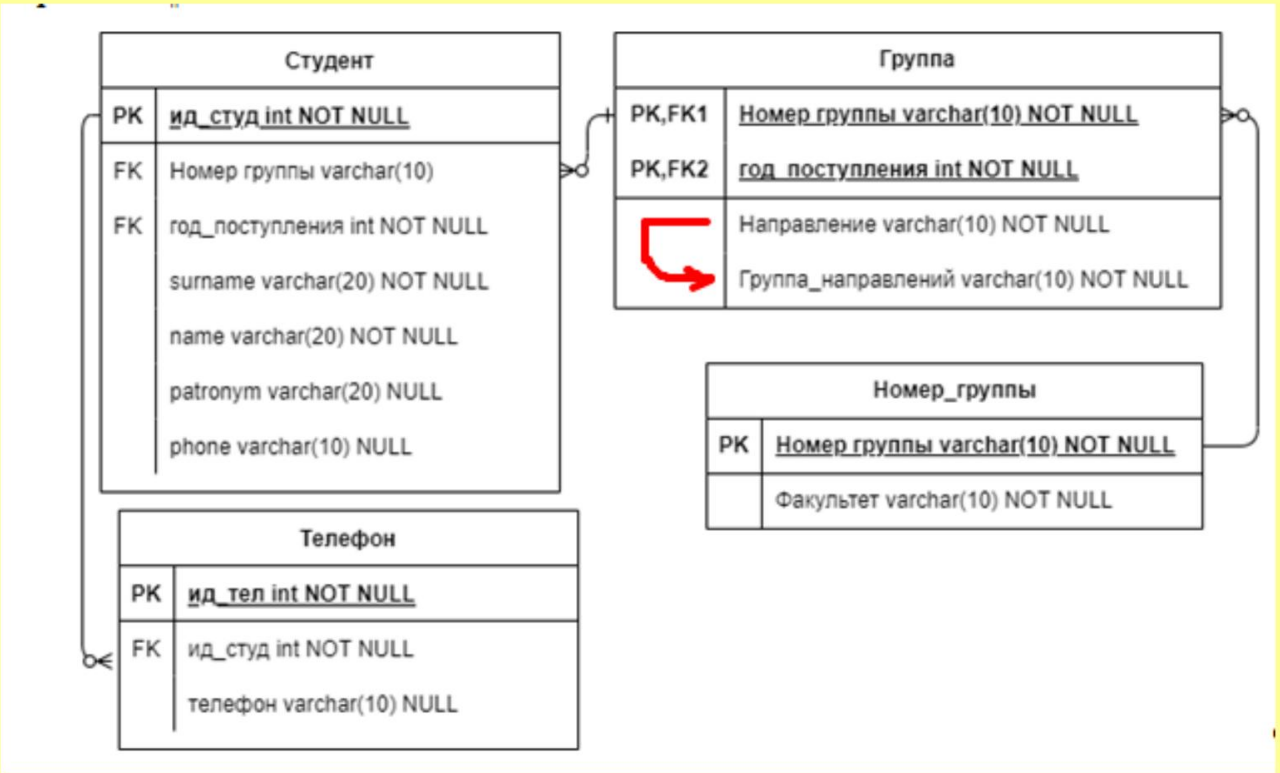
Third Normal Form (2NF)

- **Transitive dependency.** If for attributes A, B and C of some relation there are dependencies of the form A ÿ B and B ÿ C, this means that attribute C transitively depends on attribute A through attribute B (provided that attribute A does not functionally depend on either attribute B or attribute C).
- **Third normal form (TNF) -** A relation that is in the first and second normal forms and has no attributes that are not part of the primary key that are in transitive functional dependence on this primary key.

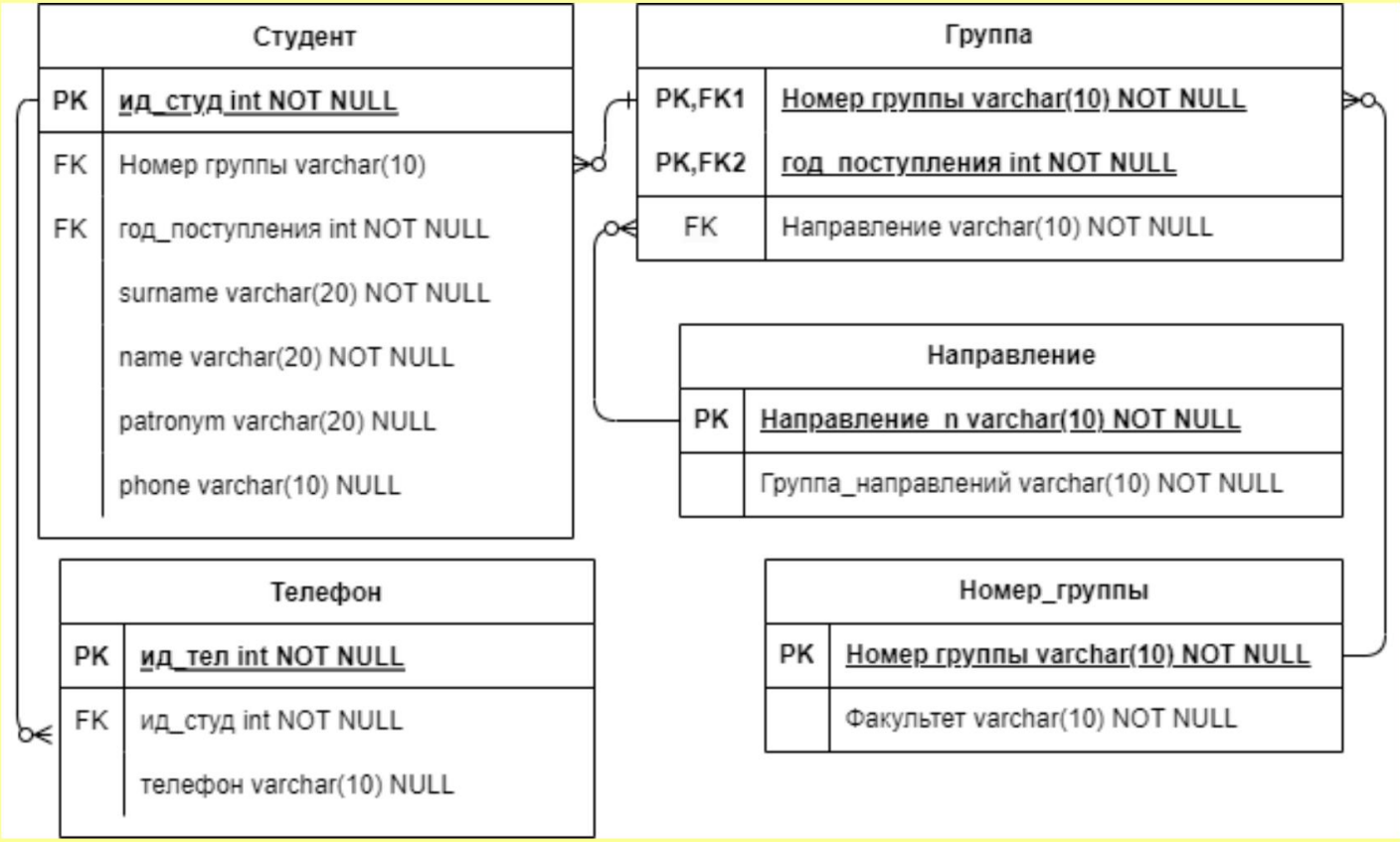
Or if more strictly

- **third normal form (TNF).** A relation in first or second normal form in which no attribute other than the primary key attribute is transitively dependent on any candidate key.

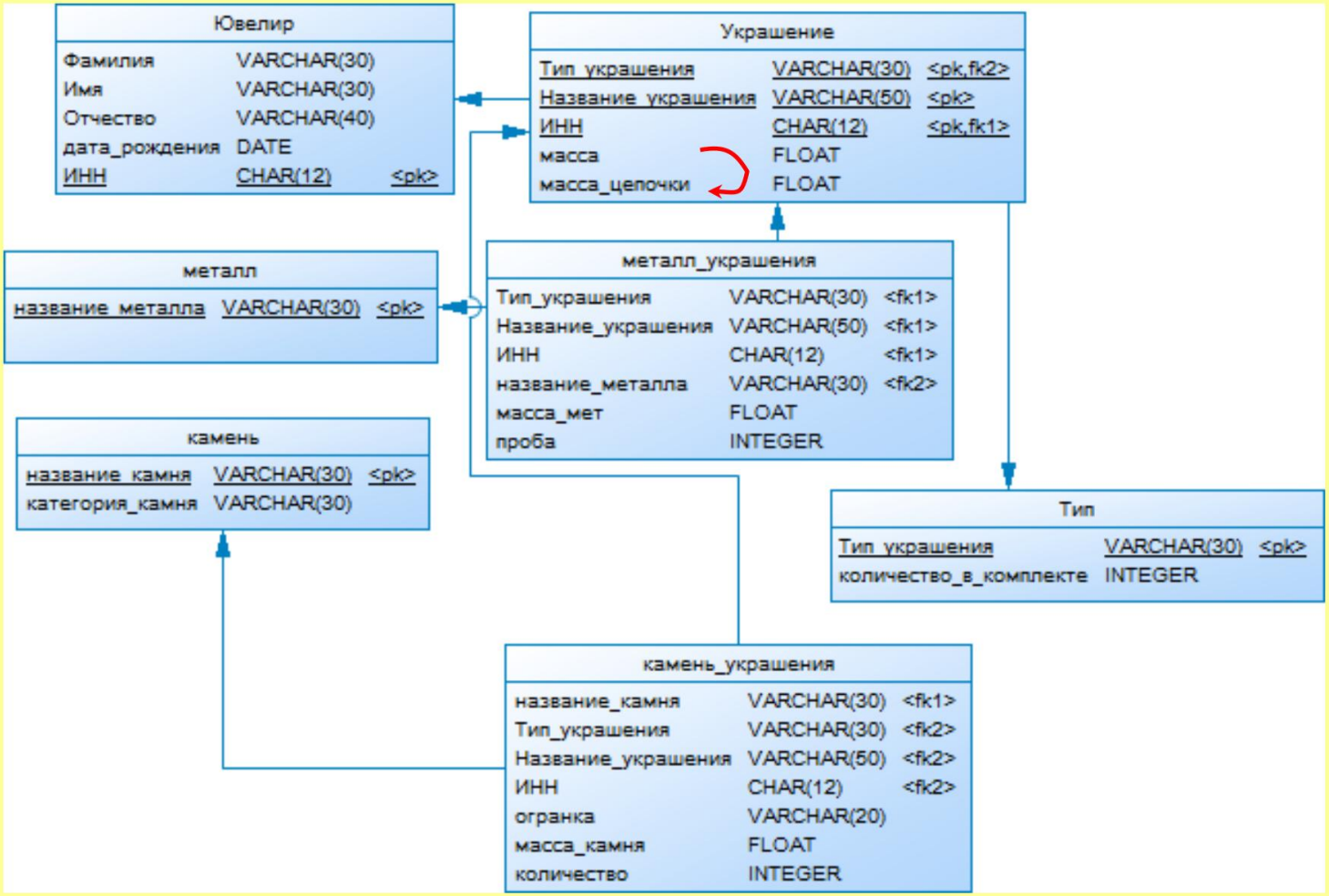
Database in 2NF



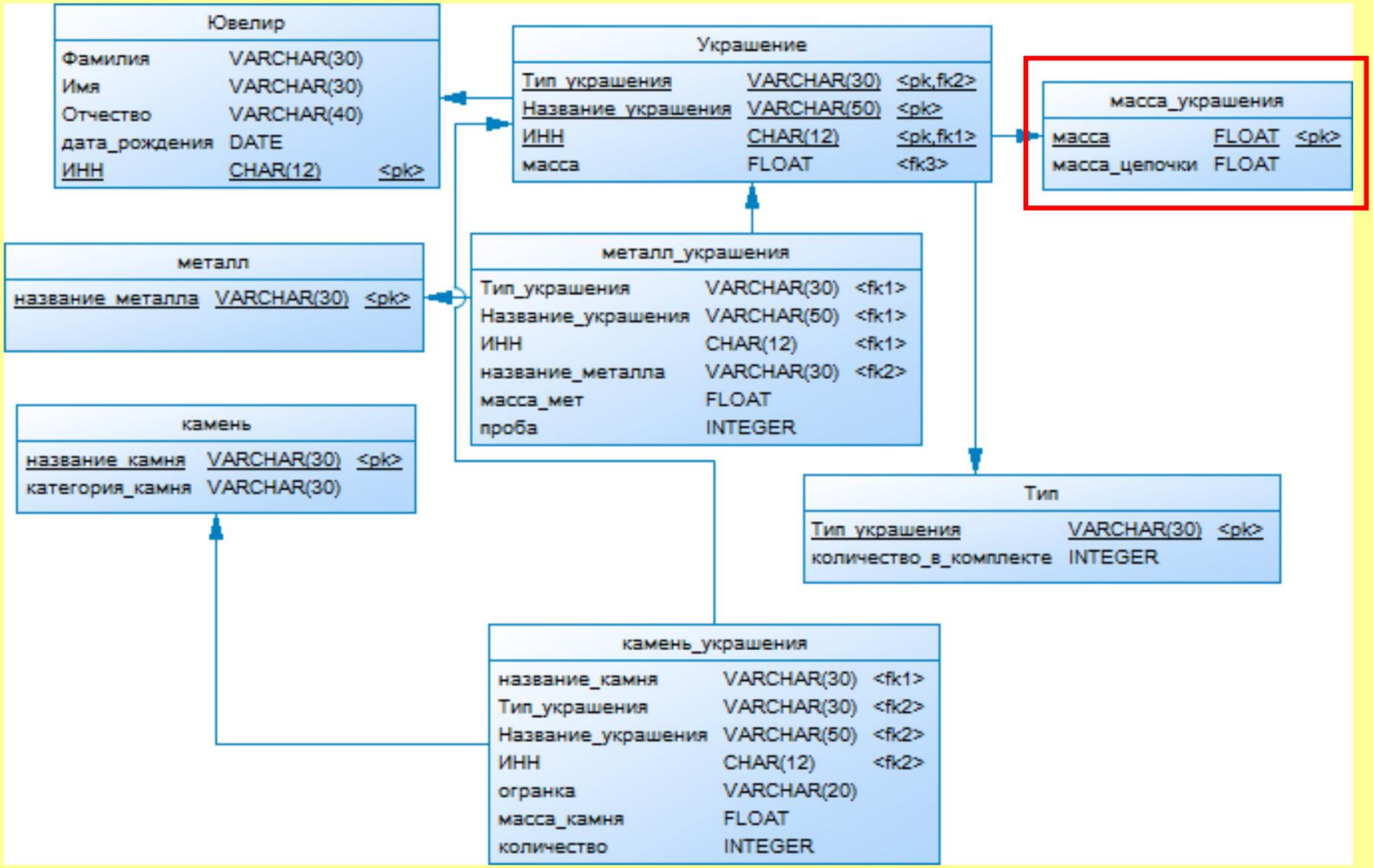
Database in 3NF



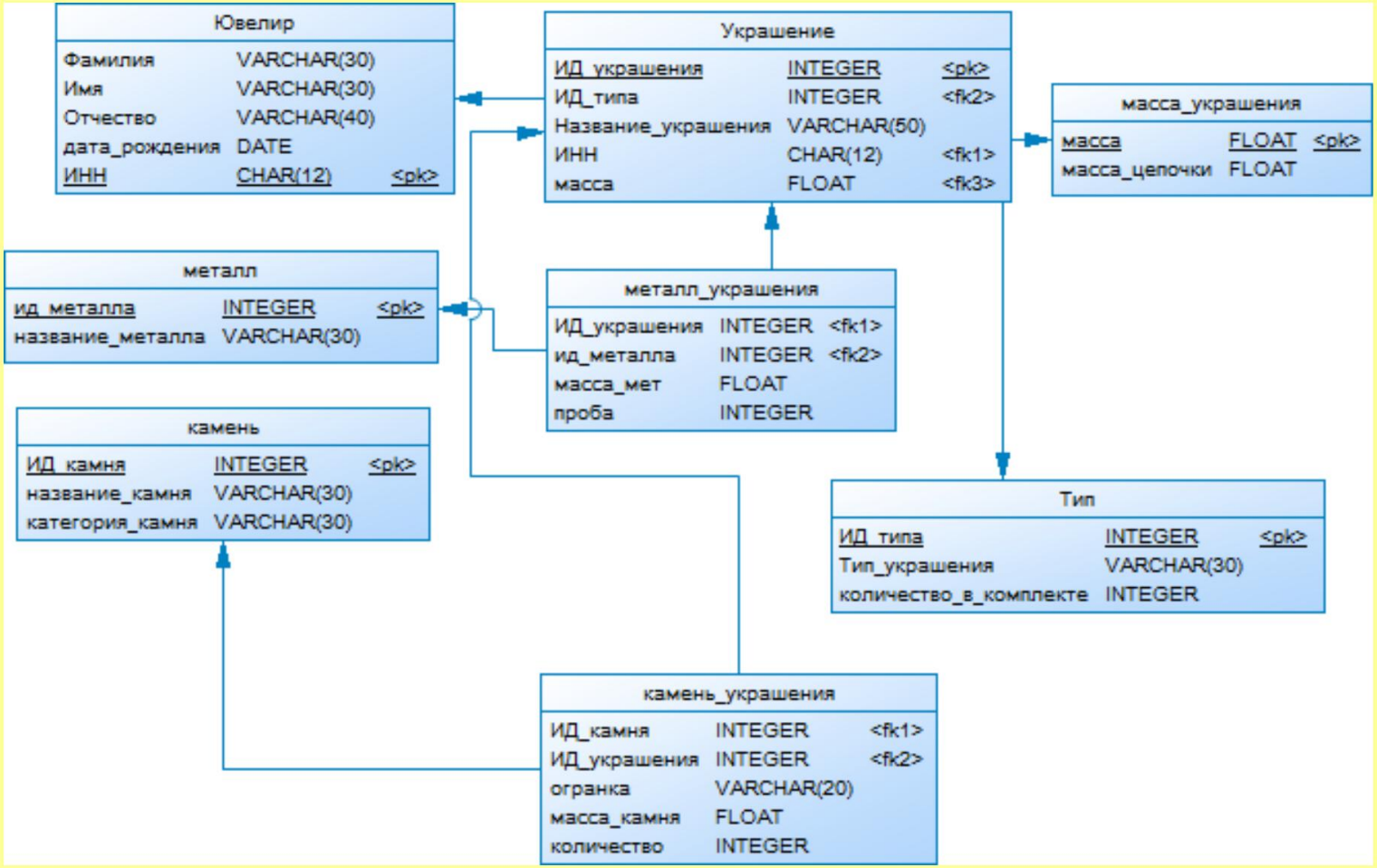
Database in 2NF without surrogate keys



Database in 3NF without surrogate keys



DB in 3NF



Boyce-Codd Normal Form (BCNF)

- **Boyce-Codd Normal Form (BCNF).**
  A relation is in BCNF if and only if each of its determinants is a candidate key.

- The difference between ZNF and BCNF is that the functional dependency A ÿ B is allowed in ZNF if attribute B is a primary key and attribute A is not necessarily a candidate key. Whereas in BCNF this dependency is allowed *only* when attribute A is a candidate key.

Boyce-Codd Normal Form (BCNF)

Attitude ClientInterview

clientNo	interviewDate	interviewTime	staffNo	roomNo
CR76	13-Mau-19	10:30	SG5	G101
CR56	13-Mau-19	12:00	SG5	G101
CR74	13-Mau-19	12:00	SG37	G102
CR56	l-Jul-19	10:30	SG5	G102

three potential keys:

- (clientNo, interviewDate) ,
- (staffNo, interviewDate, interviewTime)
- (roomNo, interviewDate, interviewTime).

Primary key

(clientNo, interviewDate).

Functional dependencies

clientNo	interviewDate interviewTime	staffNo	roomNo
CR76	13-Mau-19 10:30	SG5	G101
CR56	13-Mau-19 12:00	SG5	G101
CR74	13-Mau-19 l- 12:00	SG37	G102
CR56	Jul-19 10:30	SG5	G102

Designation	Dependency	Description
FZ1	clientNo, interviewDate ÿ interviewTime, staffNo, roomNo staffNo,	Primary key
FZ2	interviewDate, interviewTime ÿ clientNo	Potential key
FZ3	roomNo, interviewDate, interviewTime ÿ staffNo,clientNo	Potential key
FZ4	staffNo, interviewDate ÿ roomNo	

Bringing to NFBC

clientNo interviewDate interviewTime staffNo			
CR76 13-Mau-19		10:30	SG5
CR56 13-Mau-19		12:00	SG5
CR74 13-Mau-19		12:00	SG37
CR56 l-Jul-19		10:30	SG5

interviewDate staffNo roomNo		
13-Mau-19	SG5	G101
13-Mau-19	SG5	G101
13-Mau-19	SG37 G102	
l-Jul-19	SG5	G102

- Note that in the example, when creating two new BCNF relationships based on the original ClientInterview relationship, the following functional dependency is "lost": roomNo, interviewDate, interviewTime ÿ staffNo, clientNo (FZ3 dependency), since the determinant of this dependency will no longer be in the same relationship as the attributes it defines.

Fourth normal form (4NF)

- However, as a result of theoretical research, another type of dependence was identified - multi-valued *dependence* (MVD)
- **Multivalued dependency** - Represents a dependency between attributes of a relationship (e.g. A, B, and C) such that each value of A represents the set of values for B and the set of values for C. However, the sets of values for b and c are independent of each other.

A ÿ B

A ÿ C

- A multivalued dependency can be further defined as *trivial* or *non-trivial.* For example, the multi-valued dependence A ÿ In some relation R is defined as trivial if the attribute B is a subset of attribute A or A and B = R. Conversely, a multivalued dependency is defined as nontrivial if neither condition is satisfied. A trivial multivalued dependency (MVD) does not impose any restrictions on the relationship, while a nontrivial one does

  imposes.

- **Fourth normal form (4NF)** A relation in Boyce- Codd normal form that contains no nontrivial multivalued dependencies.

Fourth normal form (4NF)

branchNo	sName	oName
transporter	Ann Beech	Carol Farrel
transporter	David Ford	Carol Farrel
transporter	Ann Beech	Tina Murphy
transporter	David Ford	Tina Murphy

- the BranchStaffOwner relation presented in the table contains the names of employees (sName) and property owners (oName) of a specific branch of the company (branchNo).

- in this relation there is a multivalued dependency, since it contains two independent connections of type 1:*
- branchNo —> sName
- branchNo —> oName

Reduction to 4NF

branchNo	sName	oName
transporter	Ann Beech	Carol Farrel
transporter	David Ford	Carol Farrel
transporter	Ann Beech	Tina Murphy
transporter	David Ford	Tina Murphy

branchNo	sName
transporter	Ann Beech
transporter	David Ford

branchNo	oName
transporter	Carol Farrell
transporter	Tina Murphy

Fifth normal form (5NF)

- **Lossless join dependency:** A decomposition property that ensures that there are no dummy rows when the original relation is reconstructed using a natural join operation.

- **Fifth normal form (5NF).** The relation without connection dependencies.
- Fifth normal form (5NF), also called *project-join normal form,* or PJNF

Form -PJNF) means that the relation in this form has no join dependencies

Fifth normal form (5NF)

КлассПредметУчитель

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров

Недопустимое состояние

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров
9Б	Алгебра	Петров

Допустимое состояние

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров
9Б	Алгебра	Петров
9А	Алгебра	Петров

- This relationship describes school classes (Class) that will have certain subjects (Subject) taught by teachers (Teacher) for these classes. In addition, if a certain class (C) requires a certain subject (D), a certain teacher (T) is engaged in teaching such a subject (D), and this teacher (T) has already taught at least one subject for class (C), then this same teacher (T) will also be assigned to teach the required subject (D) for class (C).
- If a class 9A property requires the subject "Algebra" (according to the data in line 1), 9A is taught by teacher Petrov (according to the data in line 2), teacher Petrov is taught the subject "Algebra" (according to the data in line 3), then teacher Petrov must also teach the subject "Algebra" for class 9A. This example clearly illustrates the cyclic nature of the constraint that applies to the ClassSubjectTeacher relationship.

Reduction to 5NF

КлассПредметУчитель

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров

Недопустимое состояние

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров
9Б	Алгебра	Петров

Допустимое состояние

Класс	Предмет	Учитель
9А	Алгебра	Иванов
9А	Физика	Петров
9Б	Алгебра	Петров
9А	Алгебра	Петров

Class	Subject
9A	Algebra
9A	Physics
9B	Algebra

Class	Teacher
9A	Ivanov
9A	Petrov
9B	Petrov

Subject	Teacher
Algebra	Ivanov
Physics	Petrov
Algebra	Petrov

Domain Key Normal Form (DKNF)

- Each constraint in the relationships between tables should depend only on key constraints and domain constraints, where the domain represents the set of valid values for a column.

- This form prevents the addition of invalid data by setting the constraint at the relationship level between tables, but not at the table or column level

Restrictions

- **A domain constraint** is a constraint that requires that a particular attribute use values only from a certain specified domain (set of values).

- **A key constraint** is a constraint that states that some attribute or combination of attributes is a candidate key.

- Thus, **the requirement for domain-key** normal form is that every constraint imposed on a table is a logical consequence of the domain constraints and key constraints that are imposed on that table.

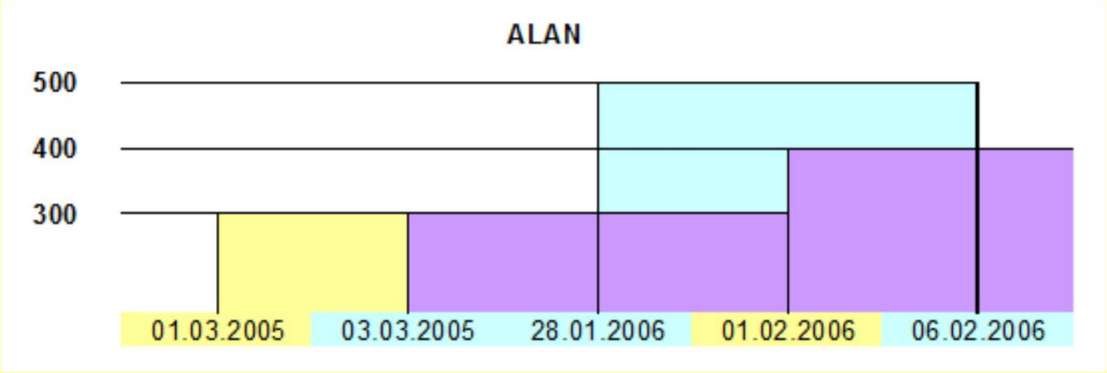
- Table located in the domain-key normal form, must be in 5NF

Temporal data

- In temporal databases, each tuple contains information about the state of the modeled object, as well as the time when this information was recorded in the database. Such a blurring of information about one logical object across several tuples was called *a vertical temporal anomaly*
- **Actual (model) time** shows a period in the past, present, or future when a fact was true in the simulated world
- **Transaction time** indicates the period in the past or present when a given record was present in the database.

Timelines

Табельный номер	Зарплата	действительное время	транзакционное время
ALAN	500	с 1 января 2006	с 20 декабря 2005
BOB	300	с 1 марта 2005 по 31 января 2006	с 3 марта 2005 по 27 января 2006
BOB	500	с 1 февраля 2006	с 28 января 2006 по 5 февраля 2006
BOB	400	с 1 февраля 2006	с 6 февраля 2006



Problems of developing a system with temporal data

- Temporal data is needed for analysis situations
- Up-to-date data must be available at all times without any degradation in performance
- Storing temporal data increases query complexity, reduces their performance and requires more space
- Operational management requires a lot of data manipulation speeds

Temporal data models

- Model of R. Snodgrass
  R = (A1, ..., An, Ts, Te, Vs,

Ve) • Model of K. Jensen
  R = (A1, ..., An, Vs, Ve, T, Op). Op::=I|D

- Model of J. Ben-Zvi *R =*
  (A1, ..., An, Tes, Trs, Tee, Tre, Td) •

Model of C. Hadiya *R*
  *= {([Ts, Te+*Vs , Ve+A1)-, …, ,(*Ts, Te+*Vs, Ve+An)-*

- Model E. McKenzie
  is a sequence of states in model time indexed by transaction time
  R = (VR, T), VR = (A1V1, ..., AnVn)

R. Snodgrass Model

- *R(A1, ..., An, T), where A1, ..., An is a set of attributes, T is a bitemporal* attribute.
- Then *R can be represented as R = (A1, ..., An, Ts, Te, Vs, Ve),*
- where *Ts, Te, Vs, Ve are atomic temporal attributes containing*
- start and end dates of transaction and model time.

Model K. Jensen

- *R = (ÿ1, ..., ÿn, Vs, Ve, T, Op). Op::=I|D*
- tuples are read-only.
- Bitemporal relation *R with a set of attributes A1, ..., An can be* represented as follows: *R = (A1, ..., An, Vs, Ve, T, Op).* • *As in* the

Snodgrass model, the attributes • *Vs and*

Ve store the start and end dates of the fact's relevance in simulated reality accordingly,

- attribute *T — information about the time of entering the tuple into* change *log .*
- Requests to create and delete tuples are denoted by attribute *Op with the corresponding symbols I (insert) and D (delete).* Data modifications are a pair of requests (delete and create a record) with the same attribute *T.*

Model J. Ben-Zvi

- *R = (ÿ1, ..., ÿn, Tes, Trs, Tee, Tre, Td)*
- Let the bitemporal relation *R consist of* set of attributes *A1, ..., An, T,* •

where T is a temporal attribute defined on a set of bitemporal elements. • Then *R can be* represented as follows:

R = (A1, ..., ÿn, Tes, Trs, Tee, Tre, Td),

- where *Tes is the time attribute when the value of the tuple attribute* becomes relevant;
- *Trs is an attribute that stores information about* when the Tes attribute *was saved in the database;* •

Tre is an attribute that stores information about when the fact ceases to exist. be relevant in the simulated reality; • *Tee is the time*

attribute when Tre was recorded in the database; • *Td is the attribute indicating*

the time when the record was logically deleted from the database.

Model C. Gadiya

- *R = {([Ts, Te+*Vs, Ve+A1)-, …, ,(*Ts, Te+*Vs, Ve+An)-*
- This approach assumes the presence of bitemporal labels for each of the attributes of the tuple, which provides the possibility of more flexible modeling of reality. • Given a bitemporal relation *R(A1, ..., An, T), where T is*

  attribute defined on a set of bitemporal elements. Then the bitemporal relation *R can be represented as relations* where each attribute has its own temporal label: *R = {([Ts, Te+*Vs, Ve+A1)-, …, ,(*Ts, Te+*Vs, Ve+An)-.*

- A tuple consists of *n elements. Each element*

  represents a triple of values: transaction time *[Ts, Te+, model* time *Vs, Ve+ and the value of the attribute Ai.*

Model E. MacKenzie

- *R = (VR, T), VR = (A1V1, ..., AnVn)*
- In this model, a bitemporal relation is

  a sequence of states in model time indexed by transaction time. In tuples with model time, attributes have their own temporal labels.

- Bitemporal relation *R with set of attributes A1, …, An*

  can be represented as a relation in which each attribute is labeled with a timestamp: *R = (VR, T),*

- *where VR is the relation in model* time; • *T is the transaction time.*
- *The state diagram of the relation* of model

time has the form • *VR = (A1V1, ..., AnVn),* • *where A1, ..., An is* a set of attributes; • *V1, Vn*

is an attribute of model time, each

of which corresponds to the attributes *A1, ..., An*

and denotes the time of relevance of its value in the modeled reality.

6 normal form

- A relation variable is in sixth normal form if and only if it satisfies all non-trivial join dependencies. It follows from the definition that a variable is in 6NF if and only if it is irreducible, i.e., cannot be further decomposed without loss.

Клиент

ID	Интервал действия	Улица	Город	Индекс	Область	Телефон

Улицы

ID	Интервал действия	Улица

Индексы

ID	Интервал действия	Индекс

Телефоны

ID	Интервал действия	Телефон

Города

ID	Интервал действия	Город

Области

ID	Интервал действия	Область

Denormalization

• **normalization** of the relation variable R means replacing it with a set of such projections Rl, R2, ..., Rn that the result of the inverse connection Rn will necessarily be the value R. **The ultimate goal** of the **redundancy** by bringing projections Rl, R2, ..., **normalization is to reduce the degree of data** the projections Rl, R2, ..., Rn to the highest possible level

normalization.

• Now we can move on to defining the concept **of denormalization.** Let R1, R2, Rn is a set of relation variables. Then

denormalization of these relation variables is called such a replacement of the relation variables by their combination R, that for all possible values of i (where i = 1, ..., n) the projection of R on the attributes Ri necessarily leads again to the creation of values Ri. The ultimate goal of denormalization is to increase the degree of data redundancy by bringing the relation variable R to a lower level of normalization compared to the original relation variables Rl, R2, ..., Rn.

• **Objective:** To determine the need to introduce controlled redundancy by relaxing normalization conditions to improve performance

systems.

Denormalization issues

- Once you start denormalization, it is difficult to say when to stop it.

- problems of redundancy and update anomalies,

  which arise from having to deal with relational variables that are not fully normalized.

- When denormalization is said to "promote high performance," what is really meant is that it promotes high *performance for some specific applications.* Any physical structure chosen that is great for some applications in terms of performance may be completely unsuitable for others.

Possibility of using denormalization

1. Joining tables with one-to-one relationships (1:1).
2. Duplicate non-key attributes in one-to-many (1:*) relationships to reduce the number of joins.
3. Duplicate foreign key attributes in one-to-one relationships to many" (1:*) to reduce the number of connections.
4. Duplicate attributes in many-to-many relationships (1:*) to reduce the number of connections.
5. Introducing repeating field groups.
6. Merging reference tables with base tables.
7. Creating tables from data contained in other tables.

Features of denormalization

- denormalization can complicate the physical implementation of the system;
- denormalization often leads to decreased flexibility;
- denormalization can speed up reading data, but at the same time slow down the updating of records.