

Opracowano na podstawie:
Thomas Connolly, Carolyn Begg, Systemy baz danych, Wydawnictwo RM, 2004

1 Relational data model

Relational data model [3] is based on mathematical concept of relation. In the model data and relationships are represented as tables, where every of them contains columns with unique identifiers.

In set theory [PWN Encyclopedia] the (binary) relation is defined as a subset of the Cartesian product of the sets $A \times B$ consisting of these pairs $\langle x, y \rangle$, in which x and y are related to considered relation R .

Cartesian product of sets [PWN Encyclopedia], one of the concepts of set theory; the Cartesian product of the sets X and Y is the set of all ordered pairs $\langle x, y \rangle$, in which the first element x belongs to the set X and the second y to the set Y .

2 Logical Database Design for the Relational Model

Logical database design [3] - a process of creating a model for information used in an company, based on a specific data model, but independent of the specific DBMS and other physical implementation details.

Step 2

Creation and verification of a local logical model for a whole perspective

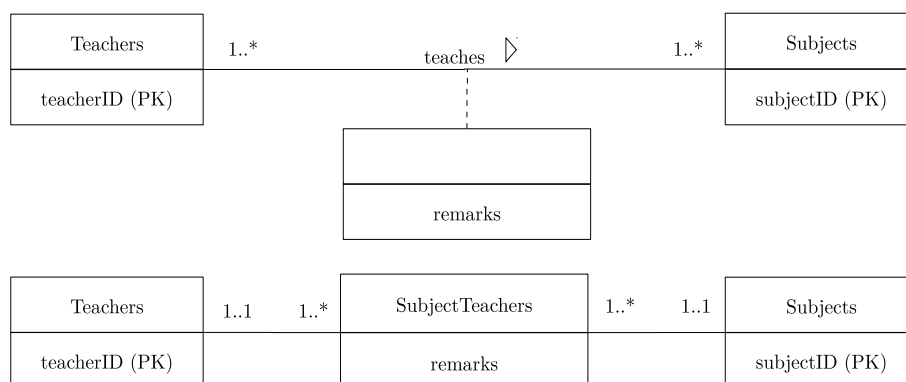
Local logical model consists of: [3]: ER diagram, relations schema and documentation.

Step 2.1

Remove properties incompatible with a relational model (optional step)

1. Remove binary relationships "many to many" (*:*)

Binary relationships "many to many" are replaced by creating an intermediate entity set and replacing "many to many" relationships for two "one to many" relationships, in which the new entity set takes part.

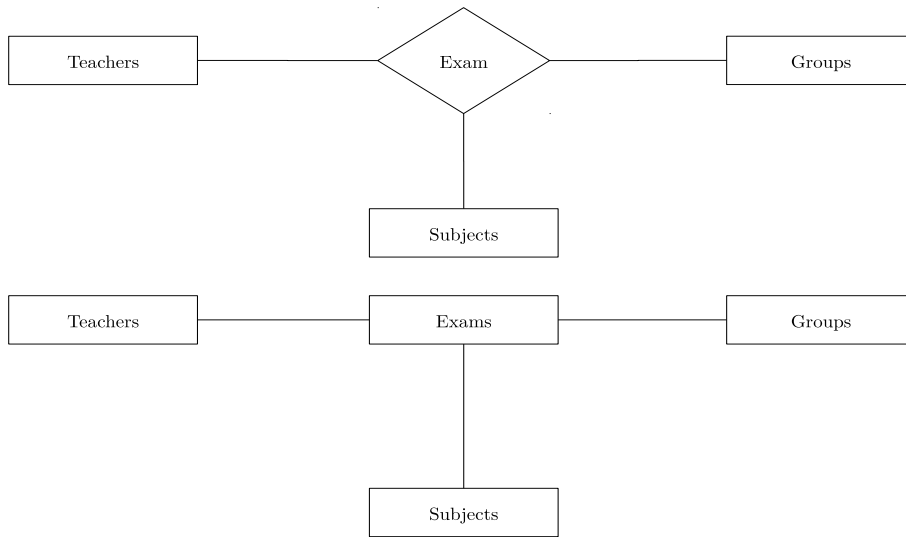


2. Remove recursive, binary relationships "many to many" (*:*)

Recursive relationships "many to many" are removed in similar way as binary relationships "many to many". The difference is that in recursive relationships, one entity set occurs twice (on both sides of the relationship).

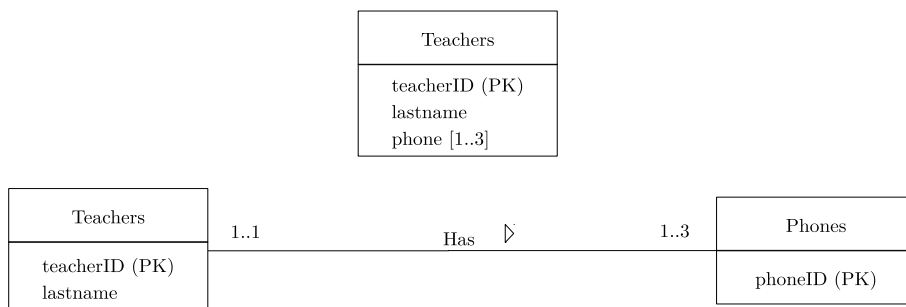
3. Remove complex relationships.

Remove complex relationships by adding intermediate entity sets. Complex relationships is replaced by a number of binary entity sets "one to many", which binding newly created entity set to entity sets in the original relationship.



4. Remove multi-value attributes

Remove multi-value attributes by adding new entity set.



Information about phones: landline phone, mobile phone i work phone.

Step 2.2

Derive relations for logical data model A definition of a relation representing entities, relationships, and attributes in Database Defition Language (DBDL) consists of relation name and a list of simple relation attributes in round brackets (parentheses). In the following lines, the primary key, alternative and foreign keys are written, for which a relation containing the pointed key should be given. In addition, a list of derived attributes along with calculation method is provided.

Relationships between entities are represented by primary keys and foreign keys. A child relation contains a foreign key, which is the primary key of a parent relationship.

1. Strong entity types

For a strong entity set, create a relation and add all simple attributes. For composite attributes, add to the relation simple attributes that they consists of.

Students
studentID (PK) lastname firstname address street home_number postoffice postcode

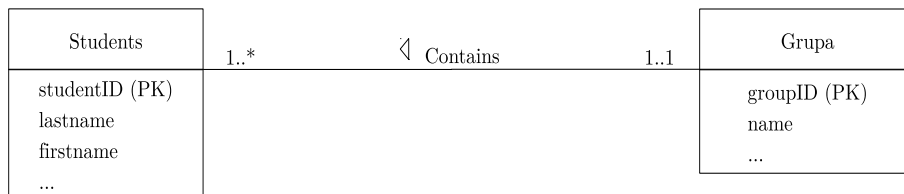
```
Students (studentID, lastname, firstname, street, ..., postcode)
Primary Key studentID;
```

Listing 1: {sql/010.sql}

- Weak entity sets For a weak entity set, create a relation and add all simple attributes. Identification of a primary key is only possible after creating relation representing all relationships with parent entity sets, because a primary key of a weak entity is partly or completely derived from parent entities.

3. Binary relationship "one to many"

Binary relationship "one to many" is represented by placing the primary key attributes of the parent entity ("one side") in the relation representing the child entity ("many side"). These attributes form a foreign key of the child relation.



```
Students (studentID, lastname, firstname, ..., groupID)
PrimaryKey studentID
Foreign Key grouID references Groups(groupID);

Groups(groupID, name, ...)
Primary Key groupID;
```

Listing 2: {sql/020.sql}

4. Binary relationship "one to one"

The representation of a binary relationship "one to one" depends on a type of participation constraints:

- mandatory on both sides:
Combine two relations and choose a primary key from primary keys of the relations. The remaining key becomes the alternative key of the relation.
- mandatory on one side:
In a child entity (mandatory participation) copy of a key from parent entity (optional participation) should be placed.
- optional on both sides:
If we do not have any additional information, then any entity can parent entity.

5. Recursive relationship "one to one"

Representation depends on the type of participation constraints, as with non-recursive relationships.

- mandatory on both sides:
The relationship is represented as one relation with two copies of the primary key, but one of the copies is a foreign key and has a name corresponding to the relationship represented.

(b) mandatory on one side:

There are two ways to represent this type of relationship: as one relation with two copies of the primary key (similar to the above) or by creating a new relation representing the relationship. The new relation contains two copies of the primary key that are the foreign key and have the appropriate names.

(c) optional on both sides:

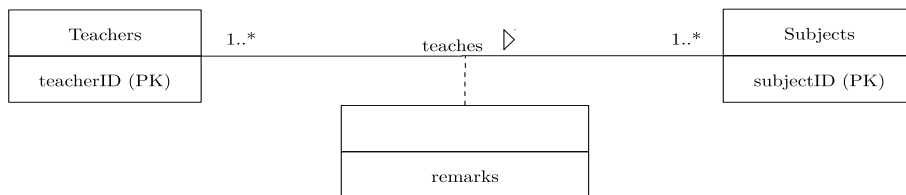
The relationship is represented by creating a new relationship (similar to above).

6. Superclass-subclass relationship

It is assumed that a superclass is a parent and a subclass is a child. There are many ways to represent this relationship.

7. Binary "many to many" relationship

For each binary "many to many" relationship, create a relation that contains the attributes assigned to the relationship and a copy of the primary key attributes of all entities in the relationship.



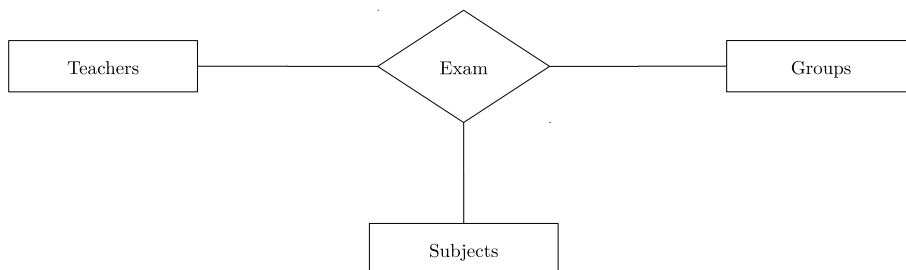
```

Teachers (teacherID, ...)
Primary Key teacherID;
Subjects (subjectID, ...)
Primary Key subjectID;
TeachersSubjects (teacherID, subjectID, remarks)
Primary Key teacherID, subjectID
Foreign Key teacherID references Teachers(teacherID)
Foreign Key subjectID references Subjects(subjectID);
  
```

Listing 3: {sql/050.sql}

8. Complex relationship

For each complex relationship, create a relation containing attributes assigned to the relationship and copies of primary key attributes of all entities in the relationship.



```

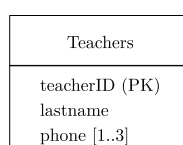
Exams (teachersID, subjectID, groupID)
Primary Key teachersID, subjectID, groupID
Foreign Key teachersID references Teachers(teachersID)
Foreign Key groupID references Groups(groupID)
Foreign Key subjectID references Subjects(subjectID);
  
```

Listing 4: {sql/060.sql}

Assumption: each exam can be approached only once. There are relations: Teachers, Groups, Subjects.

9. Multi-valued attributes

For each multi-valued attribute, create a new relation that represents it and insert in the relation as foreign key the primary key of the entity that contains the attribute.



```

Teachers(teacherNr, ...)
Primary Key teacherNr;
Phones(phoneID, numer, teacherNr)
Primary Key phoneID
Foreign Key teacherNr references Teachers(teacherNr);

```

Listing 5: {sql/070.sql}

Step 2.3

Validate relations using normalization

1. The first normal form [2] simply means that each element in every tuple has an atomic value.
The first normal form [3] - is a relation in which each intersection of a row and a column contains one and only one value.

Assumption: the reader can borrow every book only once.

reader_id	reader_lastname	address	book_id	title	borrowing_date	returning_date	author_id	author_lastname
1	Kowalski	Mała 1, 25-900 Kielce	1	Potop	1.01.2001	30.01.2001	1	Sienkiewicz
1	Kowalski	Mała 1, 25-900 Kielce	2	Lalka	2.02.2002	30.04.2002	2	Prus
2	Malinowski	Duża 2, 25-900 Kielce	3	Dziady	3.03.2003	30.06.2003	4	Mickiewicz
2	Malinowski	Duża 2, 25-900 Kielce	4	Kordian	4.04.2004	30.08.2004	3	Słowacki

Most of the attributes have been removed to improve readability.

reader_id	reader_lastname	street	house_numer	...	book_id	title	borrowing_date	returning_date	author_id	author_lastname
1	Kowalski	Mała	1	...	1	Potop	1.01.2001	30.01.2001	1	Sienkiewicz
1	Kowalski	Mała	1	...	2	Lalka	2.02.2002	30.04.2002	2	Prus
2	Malinowski	Duża	2	...	3	Dziady	3.03.2003	30.06.2003	4	Mickiewicz
2	Malinowski	Duża	2	...	4	Kordian	4.04.2004	30.08.2004	3	Słowacki

2. Second normal form (2NF) cite Connolly2004 - means the relation is in the first normal form, and every attribute outside the primary key is fully functionally dependent on it.

Functional dependency

Assumptions:

- relational schema has attributes $(A, B, C, \dots Z)$,
- the database is described by a single universal relation $R = (A, B, C, \dots Z)$.

Functional dependency - "Describes the relationship between attributes in a relation. For example, if A and B are attributes of relation R , B is functionally dependent on A (denotes $A \rightarrow B$), if each value of A is associated with exactly one value of B , (A and B may each consist of one or more attributes)."

Full functional dependency

Attributes B are fully functionally dependent on the attributes A if B is functionally dependent on A , but is not functionally dependent on any valid subset of A .

reader_id	reader_lastname	street	home_number	...
1	Kowalski	Mała	1	...
2	Malinowski	Duża	2	...

book_id	title	author_id	author_lastname
1	Potop	1	Sienkiewicz
2	Lalka	2	Prus
3	Dziady	4	Mickiewicz
4	Balladyna	3	Słowacki

reader_id	book_id	borrowing_id	returning_id
1	1	1.01.2001	30.01.2001
1	2	2.02.2002	30.04.2002
2	3	3.03.2003	30.06.2003
2	4	4.04.2004	30.08.2004

- Third normal form (3NF) - relation in the first and second normal forms, and no attribute outside the primary key is transitively dependent on it.

Transitive dependency occurs when A , B i C are relationship attributes such that $A \rightarrow B$ and $B \rightarrow C$. Then C is transitively dependent on A through B (provided that A is not functionally dependent on B or C).

author_id	author_lastname
1	Sienkiewicz
2	Prus
4	Mickiewicz
3	Słowacki

reader_id	reader_lastname	street	home_number	...
1	Kowalski	Mała	1	...
2	Malinowski	Duża	2	...

book_id	title	author_id
1	Potop	1
2	Lalka	2
3	Dziady	4
4	Balladyna	3

- Boyce-Codd normal form BCNF - means a relation in which each determinant is a candidate key. Determinant [3] - "Refers to attribute, or group of attributes, on the left-hand side of the arrow of functional dependency."

Non-compliance with BCNF occurs quite rarely, for example when:

reader_id	book_id	borrowing_date	returning_date
1	1	1.01.2001	30.01.2001
1	2	2.02.2002	30.04.2002
2	3	3.03.2003	30.06.2003
2	4	4.04.2004	30.08.2004

- the relation contains two (or more) complex candidate keys ,
 - the candidate keys overlap, for example they have at least one mutual attribute.
5. fourth (4NF) normal form,
 6. fifth (5NF) normal form,
 7. domain-key normal form (DKNF),
 8. sixth (6NF) normal form.

Thanks to the normalization [3], the model better reflects the information of the modeled enterprise, becomes consistent, ensures minimum redundancy and maximum stability. (...) It is sometimes argued that a normalized database does not provide maximum performance.

Relations containing redundant data [3] can cause problems called modification anomalies, which can be divided into insertion, deletion, and update anomalies.

GID	GROUP_NAME	SID	LNAME	FNAME
1	101	7	Kowal	Jan
2	201	3	Malinowski	Henryk
2	202	2	Nowak	Jan
3	301	5	Smith	John
3	301	6	Kowalski	Adam
3	301	4	Polak	Adam
4	401	???	???	???

Insertion anomalies:

- during insertion of a student you need to provide all the information about the group, according to values from other lines, which can lead to inconsistencies,
- insertion of a group that does not contain a student requires null values to the student attributes.

Deletion anomalies:

- during deletion of the a student from a group information about that group is lost.

Update anomalies:

- during a group name update, the update sometimes must be made on multiple lines, what may lead to inconsistencies.

SID	LNAME	FNAME
2	Nowak	Jan
3	Malinowski	Henryk
4	Polak	Adam
5	Smith	John
6	Kowalski	Adam

GID	GNAME
2	201
3	301
4	401

Step 2.4

Validate relations against user transactions

Check whether local logical data model supports all required transactions (similar to step 1.8 for the conceptual data model).

Step 2.5

Check integrity constraints

Integrity constraints are additional conditions that must be met to prevent database inconsistencies.

1. Data presence requirement (NOT NULL)
2. Attribute domains constraints (determined with the attribute domains in step 1.4)
3. Entity integrity - primary key cannot be null and must be unique for each entity.
4. Reference integrity - means that if a foreign key value is specified, it must match the existing tuple in the parent relation. The following issues must be taken into consideration: Can foreign key be null? Which conditions must be met when inserting, modifying or deleting a candidate or foreign key (NO ACTION, CASCADE, SET NULL, SET DEFAULT, NO CHECK).

Students (studentID, lastname, firstname, ..., groupID)

PrimaryKey studentID

Foreign Key groupID references Groups(groupID) ON UPDATE CASCADE, ON DELETE SET NULL;

5. General constraints - they usually follow the rules of the real world, for example group can contain from 12 to 32 students.

Step 2.6

Review logical data model with user

The model and documentation should be reviewed together with the user.

Step 3

Merge and verify logical data models into global model

Step 3.1

Merge logical data models into global model

Step 3.2

Validate the global logical data model

Step 3.3

Check for future growth

Step 3.4

Review global, logical data model with user

References

- [1] Jeffrey D. Ullman, Jennifer Widom, "Podstawowy wykład z systemów baz danych", WNT, 2001
- [2] Jeffrey D. Ullman, Jennifer Widom, "Systemy baz danych, Pełny wykład", WNT, 2006
- [3] Thomas Connolly, Carolyn Begg, "Systemy baz danych", Wydawnictwo RM, 2004
- [4] Kevin Loney, Bob Bryla, Oracle Database 10g, Podręcznik administratora baz danych. Helion 2008
- [5] Scott Urman, Ron Hardman, Michael McLaughlin, "Oracle Database 10g, Programowanie w języku PL/SQL", Helion, 2008

- [6] Price J, "Oracle Database 11g i SQL Programowanie", Helion, 2009
- [7] Ramez Elmasri, Shamkant B. Navathe, "Wprowadzeni do systemów baz danych", Helion, 2005
- [8] Sharon Allen, "Modelowanie danych", Helion, 2006
- [9] Jose A. Ramalho, Oracle 8i, Mikom, 2001
- [10] Dokumentacja do bazy danych Oracle dostępna na stronie www.oracle.com
- [11] Sadalage P. J., Fowler M., "NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence", Addison-Wesley Professional, 2012
- [12] Michael Stonebraker, "New Sql: An Alternative to Nosql and Old Sql For New Oltp Apps", June 16, 2011

Materials for the subject are available at: <http://weaii-moodle.tu.kielce.pl/>