Relational model

Jaroslav Porubän, Miroslav Biňas,
Milan Nosáľ (c) 2011 - 2016

Relational database model

- Data are represented as a mathematical **relation** (subset of cartesian product) of attribute domains
- **Table** (relation) represents an entity type (e.g.: `Student`)
- **Row** in the table (tuple) represents concrete entity (e.g.: `Student John Doe`)
- **Columns** represent modelled properties (attributes) of the entity type (e.g.: `Name John`)

Example

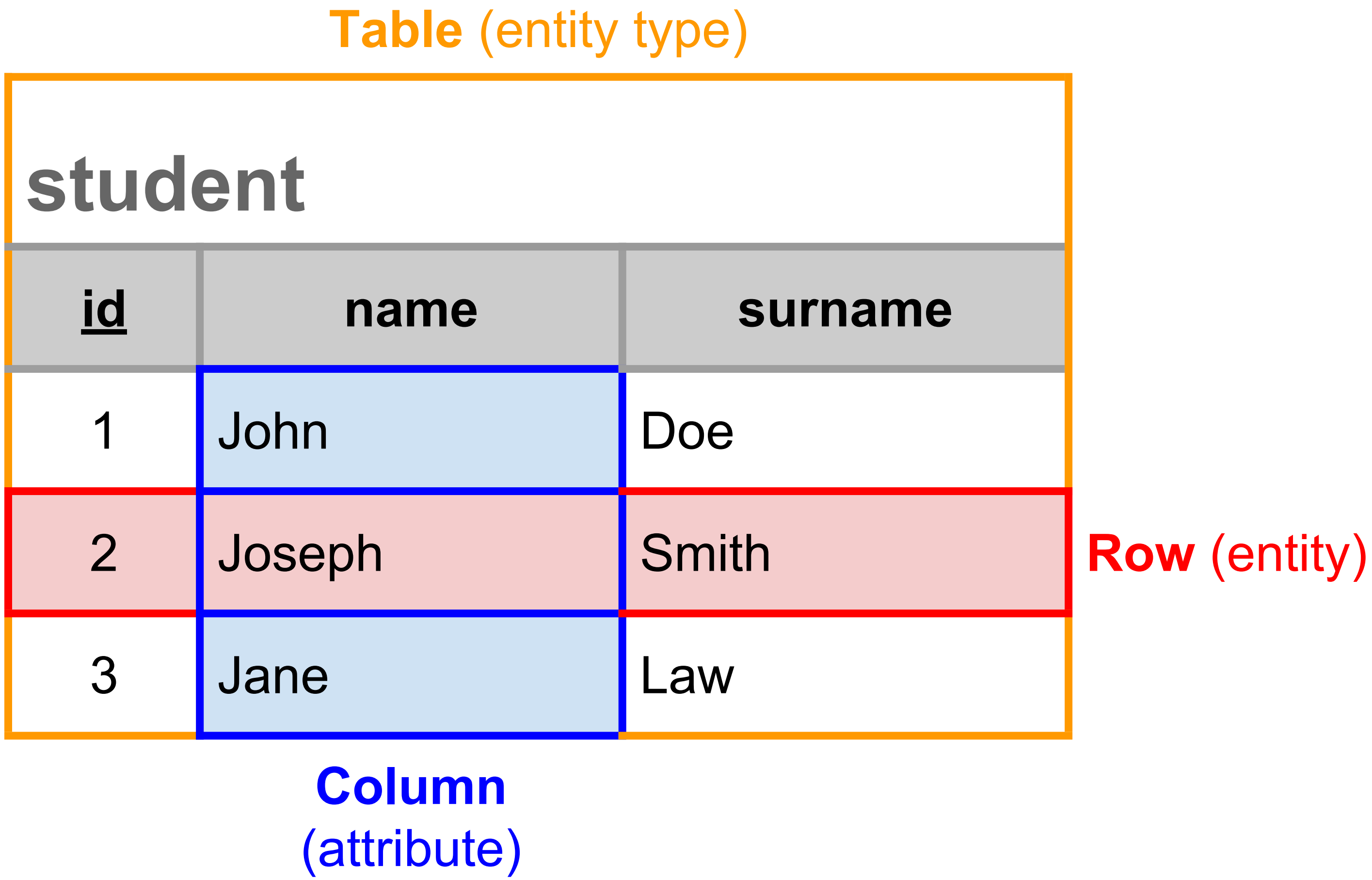


Table (entity type)

student

id	name	surname
1	John	Doe
2	Joseph	Smith
3	Jane	Law

Row (entity)

Column
(attribute)

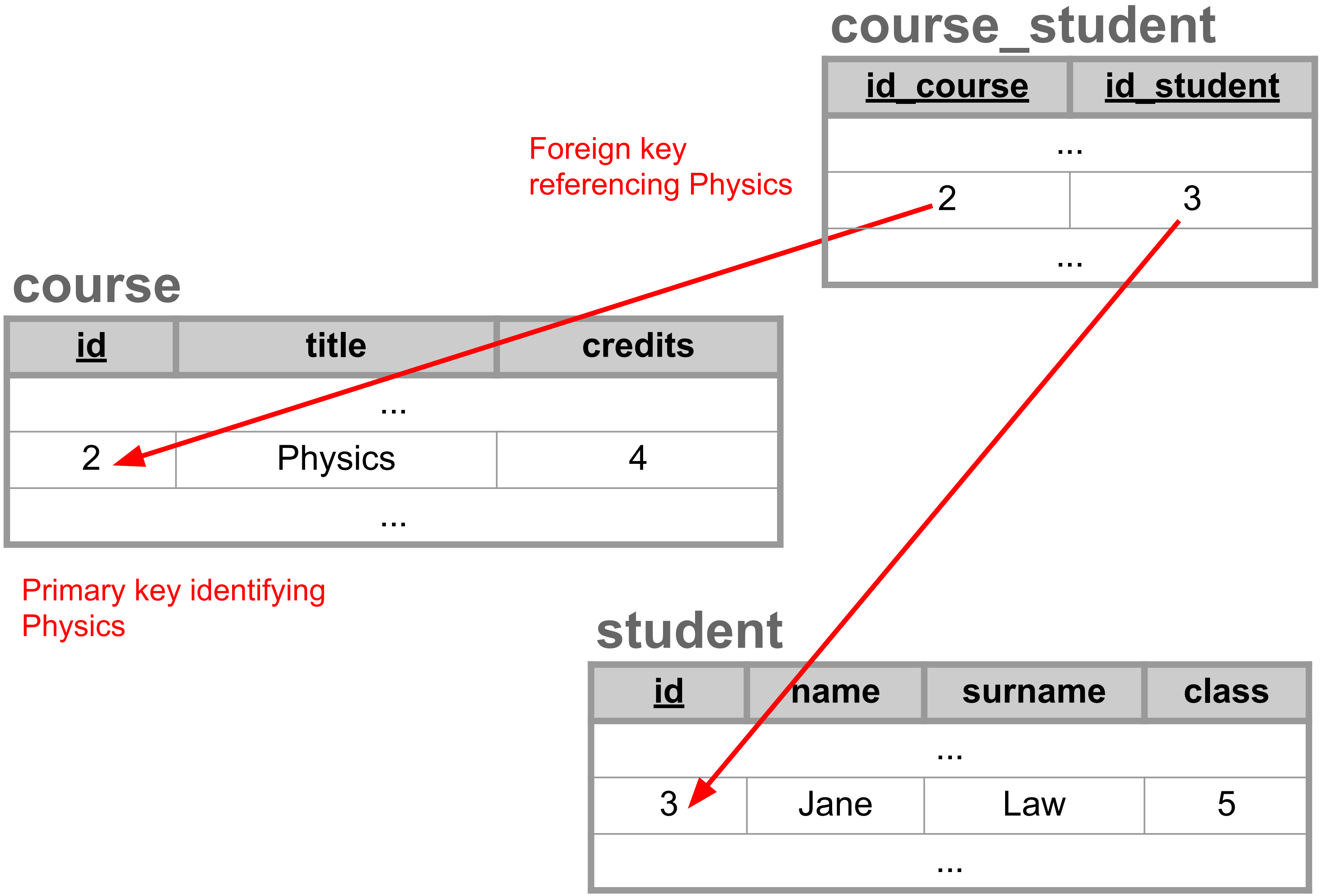
Relational database model

- Every table has a unique name
- Every tuple in table has the same structure
- Every column has its name
- The order of columns is insignificant
- Every column contains values of the same attribute
- Every tuple in the table represents a single entity of the given entity type
- Every tuple is unambiguously identified by its primary key
- Ordering of the rows is insignificant
- All values in a given tuple are unambigously and fully dependent on its primary key

Primary and foreign keys

- A key is an attribute (simple key) or a set of attributes (composed key) that unambiguously identify tuples in the table
- **Primary key**
  - Cannot be NULL
  - Has to be **unique** and **minimal**
  - Can be artificial, created specifically for the purpose of entity identification (column `id`)
- **Foreign key** - unambiguously identifies (references) a tuple in a different table (used to represent relationships)

Primary and foreign keys



course_student

id_course	id_student
...	
2	3
...	

Foreign key referencing Physics

course

id	title	credits
...		
2	Physics	4
...		

Primary key identifying Physics

student

id	name	surname	class
...			
3	Jane	Law	5
...			

Keys

- **Super key**
  - A set of attributes unambiguously identifying every entity (e.g., id, name and surname of a student)
- **Candidate key**
  - Minimal super key (e.g., id of a student)
- **Primary key**
  - A single selected candidate key

Data integrity in DB

- **Data integrity** - database has to guarantee that stored data are the whole time consistent and intact
- DBS pprovides mechanisms (so called **integrity constraints**) to enforce data integrity during manipulation with the data (`INSERT`, `UPDATE`, `DELETE`)
- Integrity constraints:
  - **Domain integrity**
  - **Entity integrity**
  - **Referential integrity**

Domain integrity

- Domain defines values that are allowed in a given column of the table
- Every value in the column must
  - Be atomic
  - Belong to the given domain
- Examples:
  - Age is positive integer
  - Sex is ‘male’ or ‘female’
- Domain integrity enforces the same structure of the tuples and in effect makes the easily processable

Entity integrity

- Primary key cannot contain `NULL` value
- In a single table (relation) the tuples are differentiated (unambiguously identified) by primary key
  - 2 different rows have different primary key
- Enforces entity identifiability
  - E.g., there are two students with the same name - we have to be able to tell them apart - one is supposed to get A, the other one E
  - Weak entity is identified by its identifying relationship

Referential integrity

- Each foreign key has to
  - **Unambiguously identify** tuple from another table using its primary key
  - Or it has **`NULL` value**, if the relationship does not exist
- It enforces relationships' consistency (e.g., it prevents nonexistent relationships - student John Doe is studying a nonexistent course)

Normalized schema

- Goal:
  - Store data without duplications
  - Prevent data manipulation anomalies
- **Normal forms**
  - Set of rules describing an “ideal database” (in several levels) from the viewpoint of redundancy
  - Every level requires fulfillment of its rules and fulfillment of each previous normal form

Normalization

- Based on functional dependencies between table columns (attributes)
- Good and thought-through ER model design usually leads to proper fulfillment of normalization requirements necessary for practical application (first three forms)
    - E.g., not using composed attributes
- From the viewpoint of performance is normalization not always desirable

Normal forms

- In practice three normal forms are used
  1. NF - each attribute contains only atomic values
     - E.g., name, surname
  2. NF - each non-key attribute depends on the full primary key
     - Table has single purpose (problem of composed keys)
  3. NF - attributes are not transitively dependent
     - Attribute depends only on primary key
- Using NF results in multiple tables

1. NF violation

- If we have to parse values from an attribute
  - e.g., saving name and surname together
  - Misuse of attribute note
- The violation requires parsing of data in application code

User	
username	user
nezbednik	Milan Nosáľ 01.04.87 (M)
juzek	Jozef Rapavý 04.04.88 (M)

2. NF violation

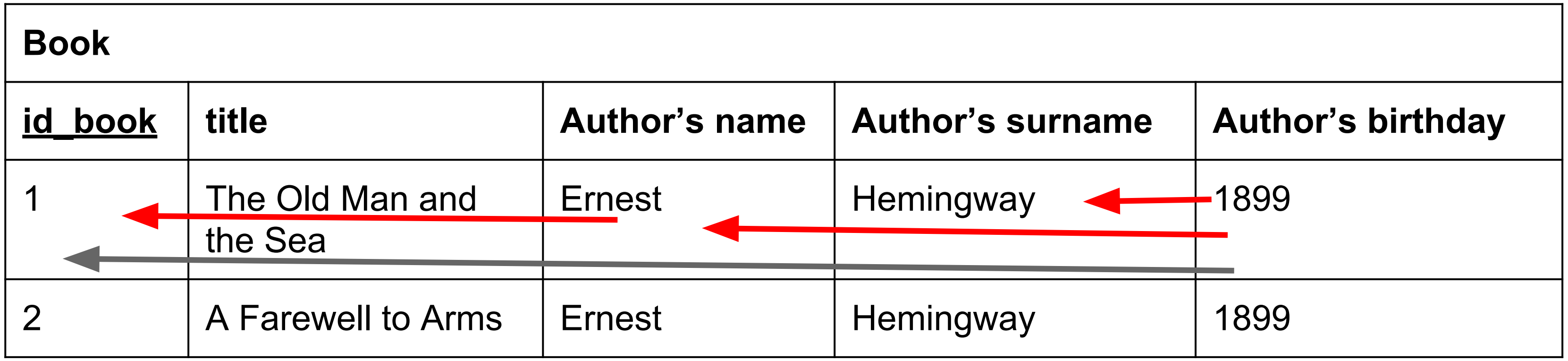
- If an attribute is dependent only on a part of the primary key, the table violates 2. NF

username	name	surname	birthday	sex	date	type	text
nezbednik	Milan	Nosáľ	01.04.87	M	16/02/2016 10:09:00	friends	Ako si...
nezbednik	Milan	Nosáľ	01.04.87	M	21/02/2016 06:00:55	public	Lenivec,...
juzek	Jozef	Rapavý	04.04.88	M	15/02/2016 13:33:00	friends	Dota je...
anezka	Anežka	Pekná	NULL	F	NULL	NULL	NULL

  - These should be two entity types - two different tables (User and Post)

3. NF violation

- Transitive dependency



Book				
id_book	title	Author’s name	Author’s surname	Author’s birthday
1	The Old Man and the Sea	Ernest	Hemingway	1899
2	A Farewell to Arms	Ernest	Hemingway	1899

  - Again those are two entity types - Book and its Author

3. NF violation

- E.g., calculated attributes (derived)

Purchase				
id_customer	**id_article**	**quantity**	**unit price**	**total price**
1	2	10	20	200
1	13	10	30	300

  - Current price can be always calculated from quantity and unit price
  - Violation of this 3. NF is sometimes desirable for performance reasons

What to be careful about

- Use atomic attributes
  - Do not code multiple values in a single attribute
- Substitue multivalued and composed attributes with a new entity type
  - E.g., address of a person will not be transformed into several attributes of the person, but a new table Address should be created
- Do not calculate (derive) attributes

What to be careful about

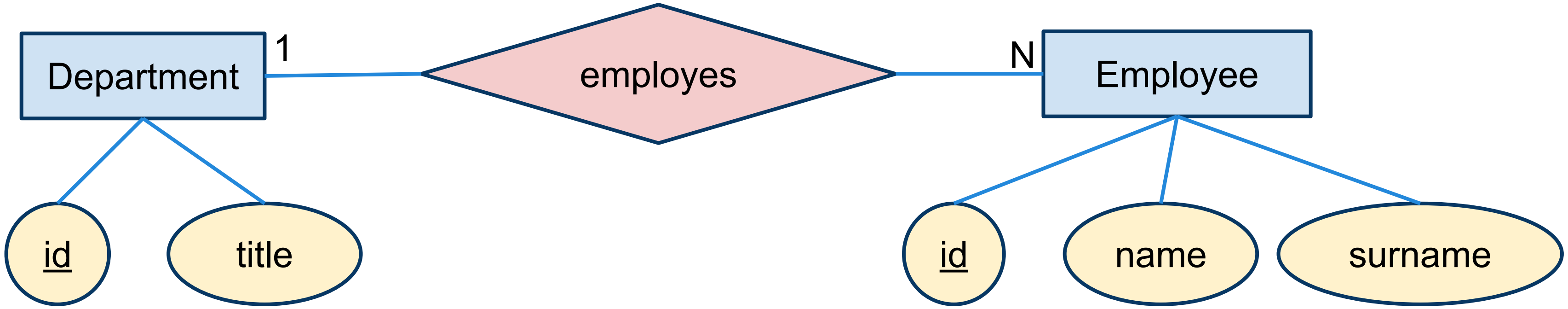
- Define entity types clearly
  - Every table should have a single purpose
  - E.g., table recording users should not record also their posts - that is purpose of another table - table Post
- Watch out for repeating values in multiple related table columns
  - E.g., when the user and post are put together, and there is a user with multiple posts, all the attributes for the given user will be repeated for each his post

Transforming ERM to RM

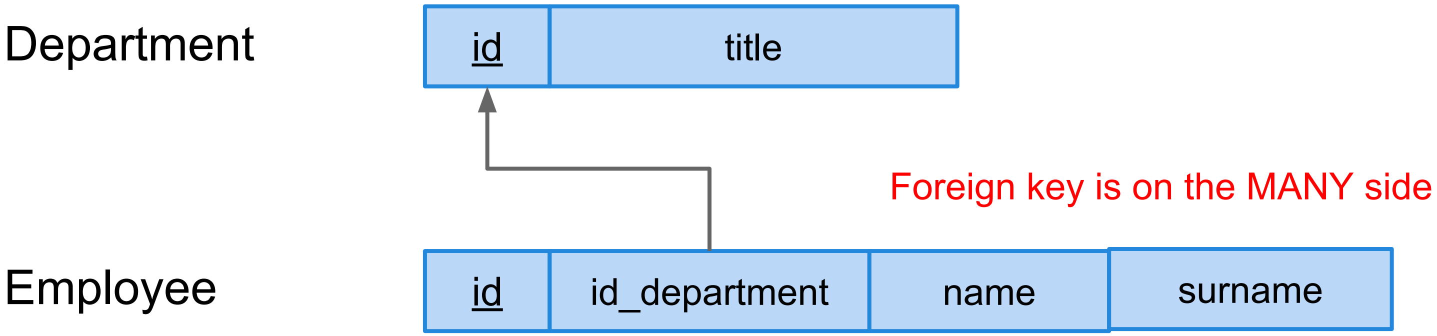
- To implement a database in relational DBMS ERM has to be transformed to RM
- Considering similarity of the models the transformation is pretty straightforward:
    - Entity type = table
    - Entity attribute = table column (unless it is a multivalued or composed attribute)
    - Relationship = combination of primary and foreign key (referential integrity)
    - It is necessary to add constraints for domain integrity (data types, etc.)

Transforming 1:N (1:1) relationship

ERD



RM

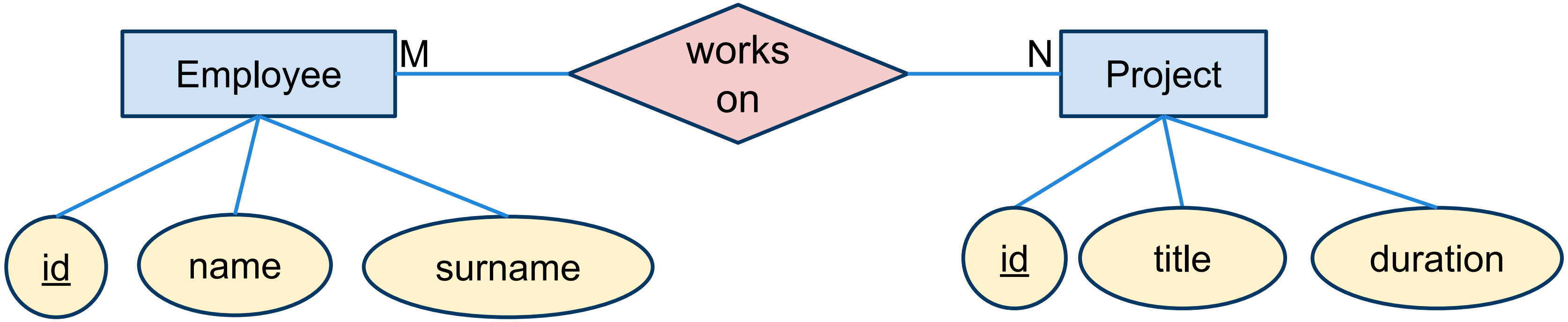


Department	id	title		
Employee	id	id_department	name	surname

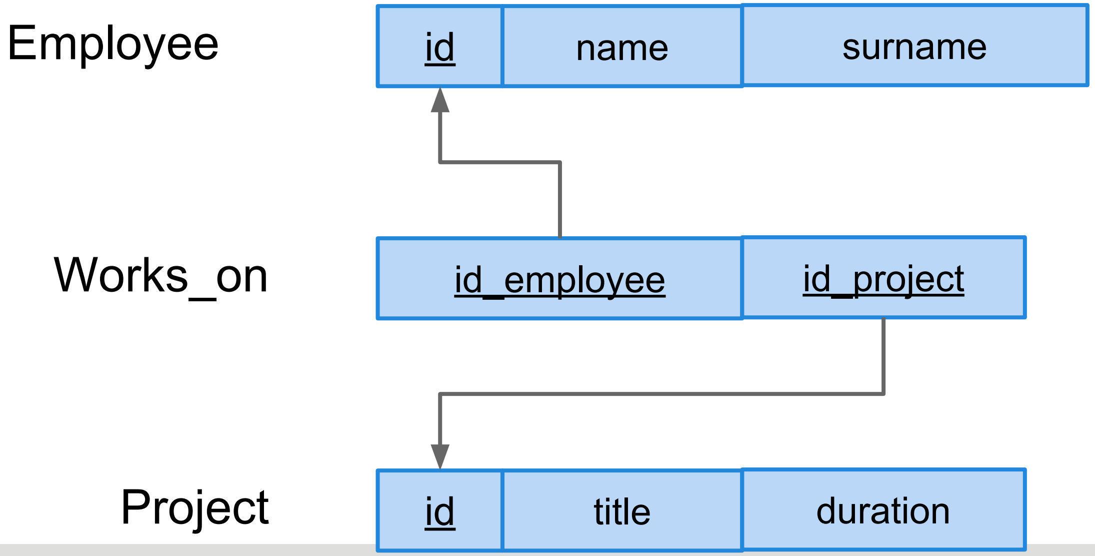
Foreign key is on the MANY side

Transforming M:N relationship

ERD



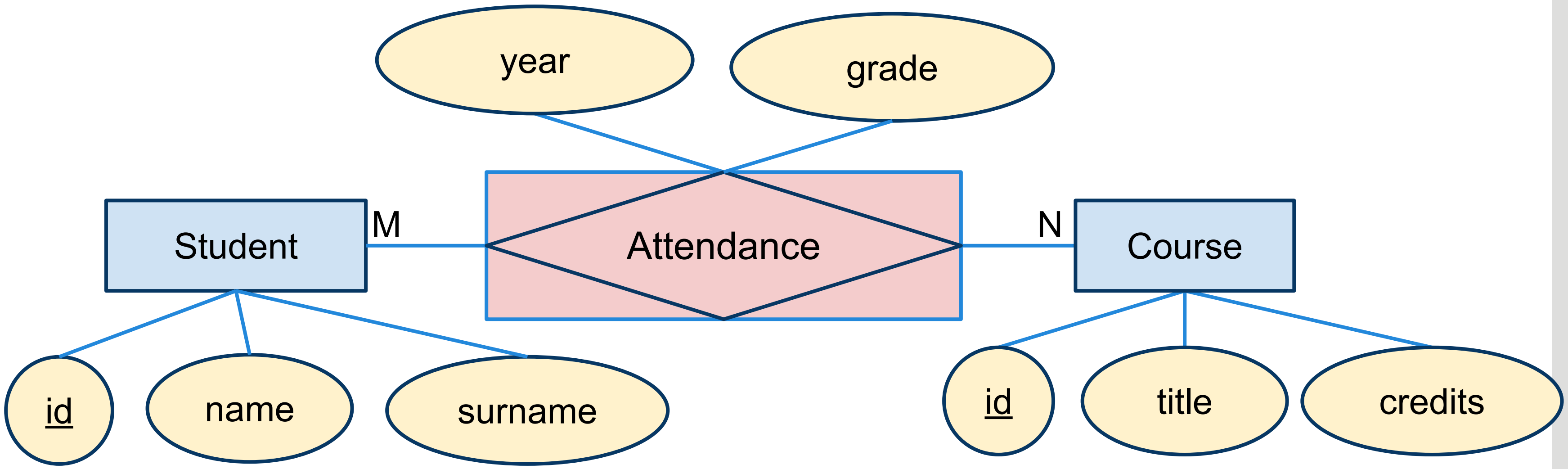
RM



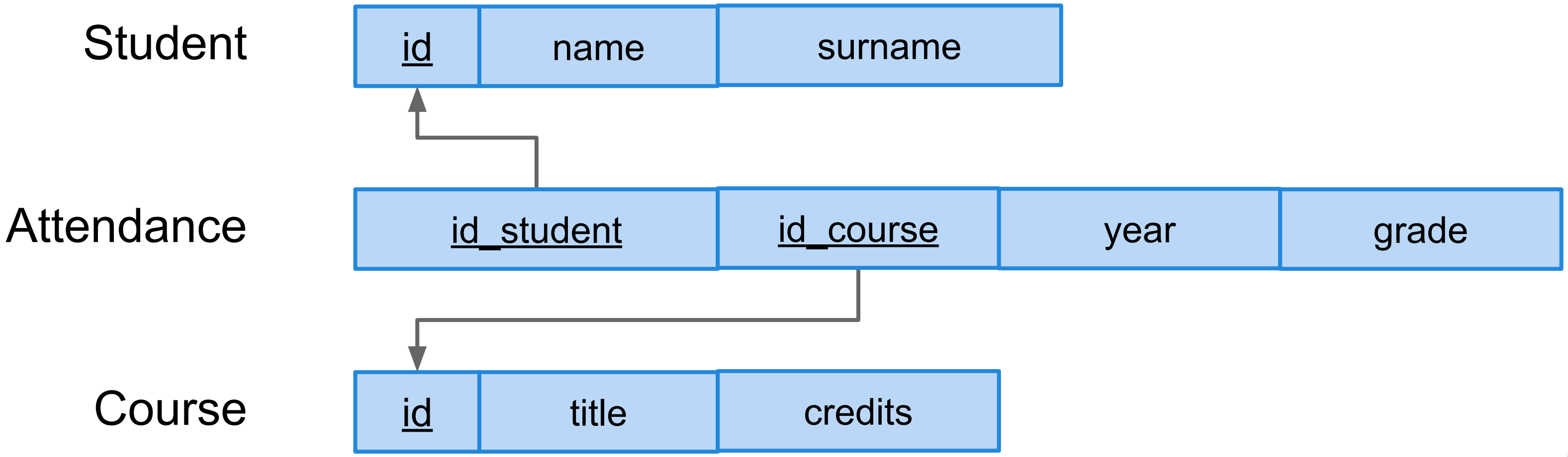
Employee	id	name	surname
Works_on	id_employee	id_project	
Project	id	title	duration

Transforming relationship with associative entity

ERD



RM



Student: id, name, surname

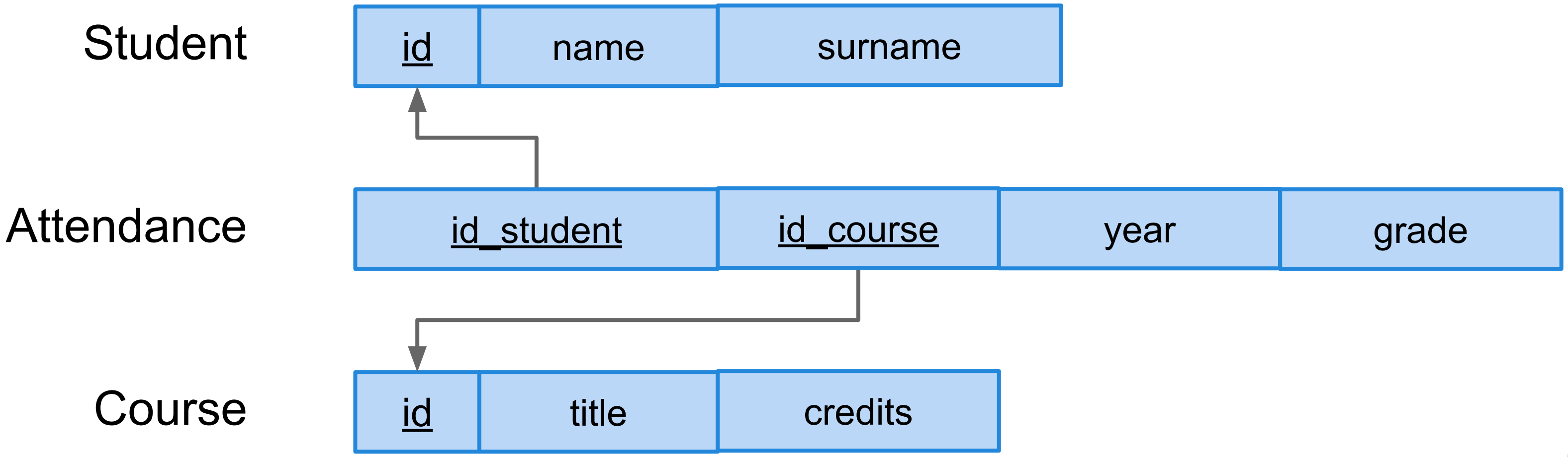
Attendance: id_student, id_course, year, grade

Course: id, title, credits

Primary keys of associative entities

- Minimal key of an associative entity is usually combination of both (or all in general) foreign keys
- Proper primary key depends on semantics
  - id_student, id_predmet
  - id_student, id_predmet, year
  - ...

RM



Student: id, name, surname

Attendance: id_student, id_course, year, grade

Course: id, title, credits

Questions?