The relational model and syntetic database design

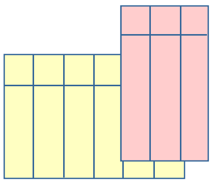
Two themes:

The relational model – the theoretic base for relational data base management systems

From konceptual schema to relational database

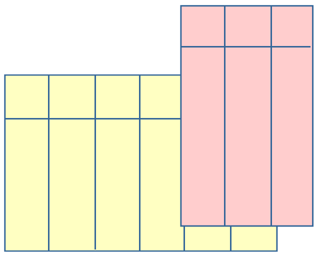
Learning Goals

After this lecture you will be able to:



- **Understand and explain the building blocks of the relational model**
- **Translate a UML class diagram into a relational database schema.**
  - Translate classes
  - Translate attributes
  - Translate associations
  - Translate rules

The relational model



TABLES: TABLEHEADS AND ROWS!

A relational database is a database that is viewed by the owner as a collection of TABLES with ROWS – independently of how the set of data is stored.

The relational model is the structuring mechanism on which relational databases are based.

Ett relationsschema

Table / Relationschema name
Each table is identified by its name!

Relational schema (Table definition)
A list of attributes (columns) that specify what the relation is about

TABLE NAMED

Personno : String	Name: String	Age: Integer	Salary: Integer

Attribute (column)
A label that can be used to describe data in a table

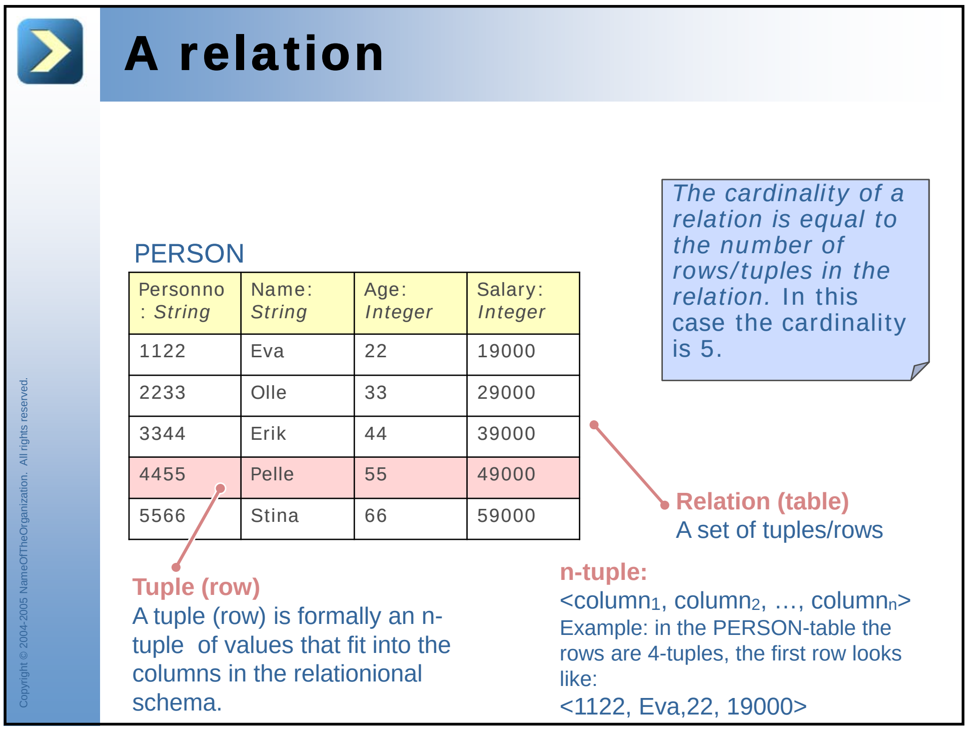
Domain
A set of values used to determine what would be the *allowed* values of a column

The *grade* of a relational schema is equal to the number of columns in the schema.

Here the grade = 4.

A relational schema may be written using the following simple syntax (identifying attribute is usually underlined):

TABLENAME(<u>Personno</u>, Name, Age, Salary)



A relation

PERSON

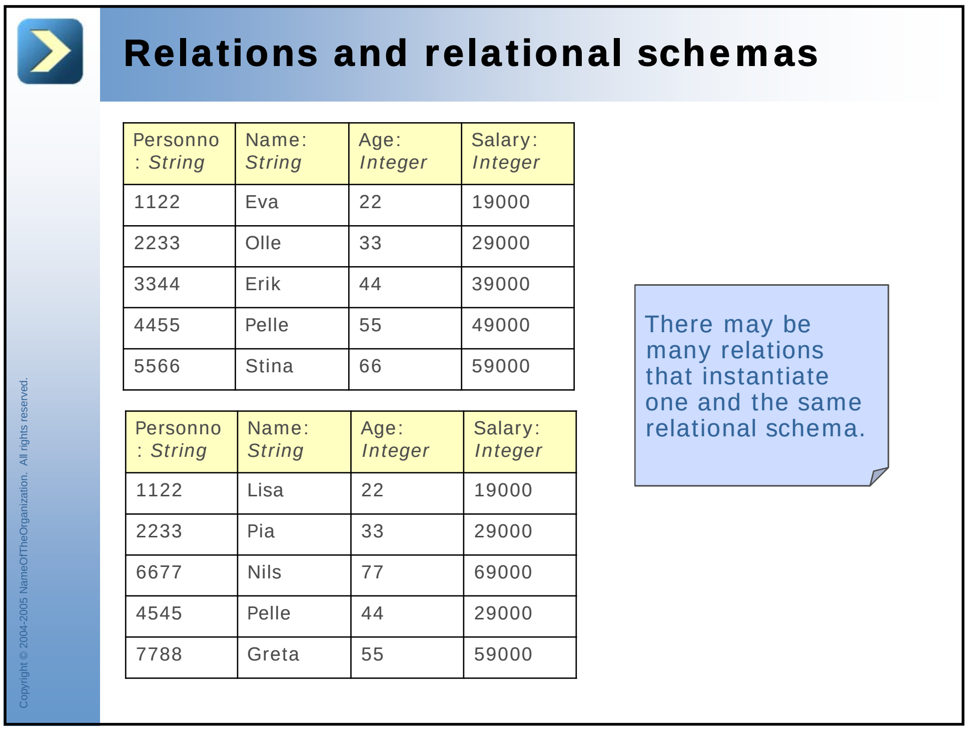
Personno : *String*	Name: *String*	Age: *Integer*	Salary: *Integer*
1122	Eva	22	19000
2233	Olle	33	29000
3344	Erik	44	39000
4455	Pelle	55	49000
5566	Stina	66	59000

The cardinality of a relation is equal to the number of rows/tuples in the relation. In this case the cardinality is 5.

Relation (table)
A set of tuples/rows

Tuple (row)
A tuple (row) is formally an n-tuple of values that fit into the columns in the relationional schema.

n-tuple:
<column$_1$, column$_2$, …, column$_n$>
Example: in the PERSON-table the rows are 4-tuples, the first row looks like:
<1122, Eva,22, 19000>

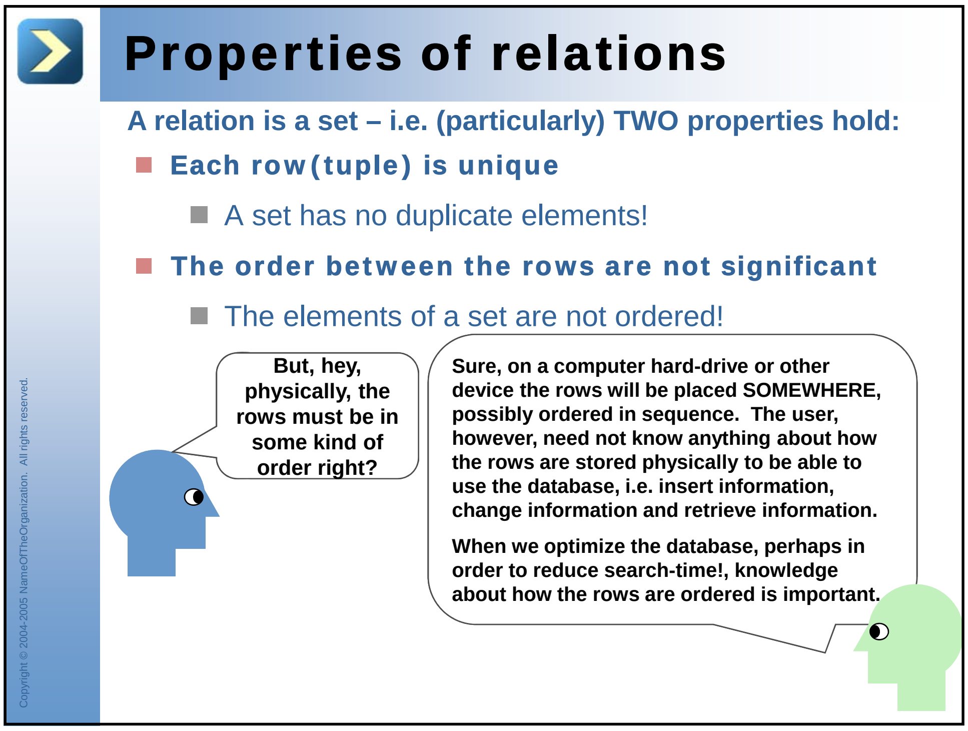


Relations and relational schemas

Personno : *String*	Name: *String*	Age: *Integer*	Salary: *Integer*
1122	Eva	22	19000
2233	Olle	33	29000
3344	Erik	44	39000
4455	Pelle	55	49000
5566	Stina	66	59000

Personno : *String*	Name: *String*	Age: *Integer*	Salary: *Integer*
1122	Lisa	22	19000
2233	Pia	33	29000
6677	Nils	77	69000
4545	Pelle	44	29000
7788	Greta	55	59000

There may be many relations that instantiate one and the same relational schema.



Properties of relations

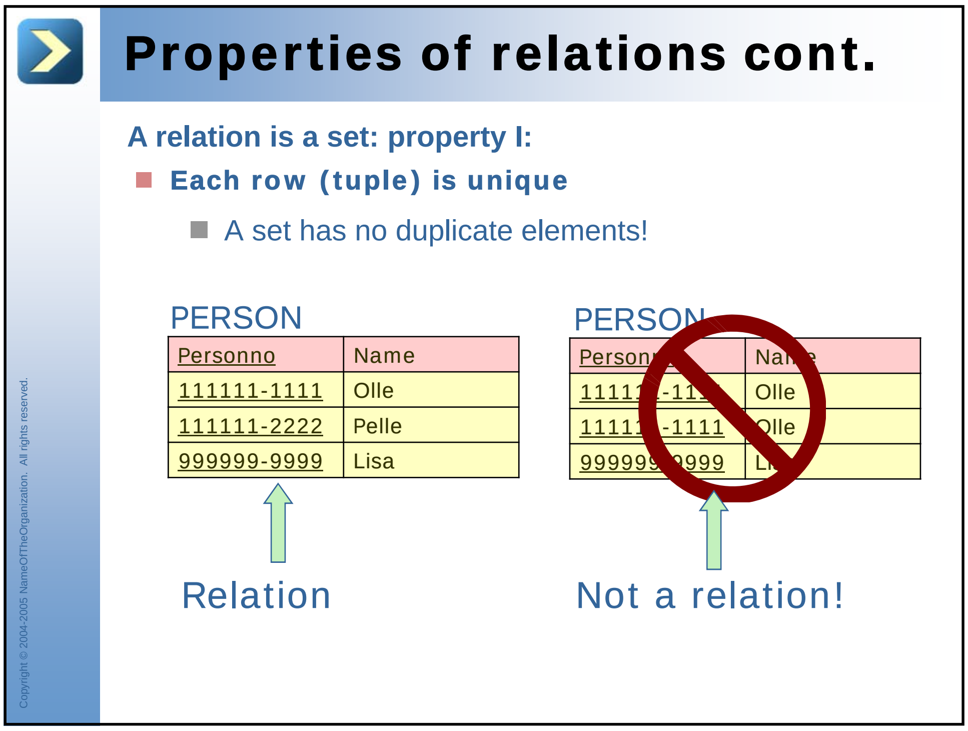
A relation is a set – i.e. (particularly) TWO properties hold:

- **Each row(tuple) is unique**
  - A set has no duplicate elements!
- **The order between the rows are not significant**
  - The elements of a set are not ordered!

But, hey, physically, the rows must be in some kind of order right?

Sure, on a computer hard-drive or other device the rows will be placed SOMEWHERE, possibly ordered in sequence. The user, however, need not know anything about how the rows are stored physically to be able to use the database, i.e. insert information, change information and retrieve information.

When we optimize the database, perhaps in order to reduce search-time!, knowledge about how the rows are ordered is important.



Properties of relations cont.

A relation is a set: property I:

- **Each row (tuple) is unique**
  - A set has no duplicate elements!

PERSON

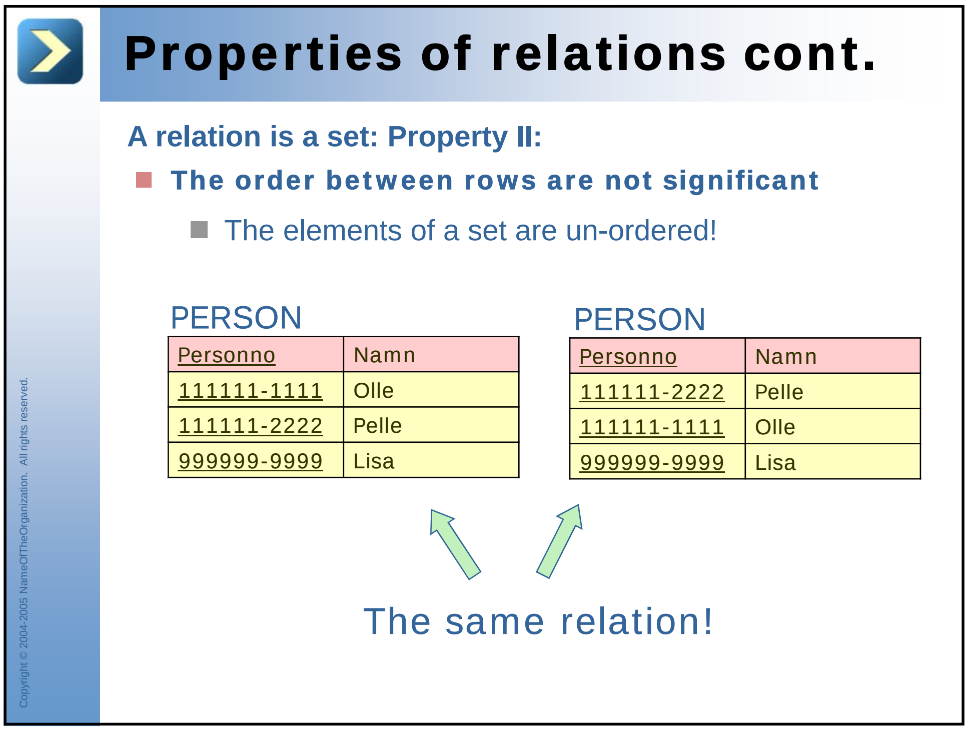
Personno	Name
111111-1111	Olle
111111-2222	Pelle
999999-9999	Lisa

Relation

PERSON

Person	Na e
1111 -11	Olle
1111 -1111	Olle
99999 9999	Li

Not a relation!



Properties of relations cont.

A relation is a set: Property II:

- **The order between rows are not significant**
  - The elements of a set are un-ordered!

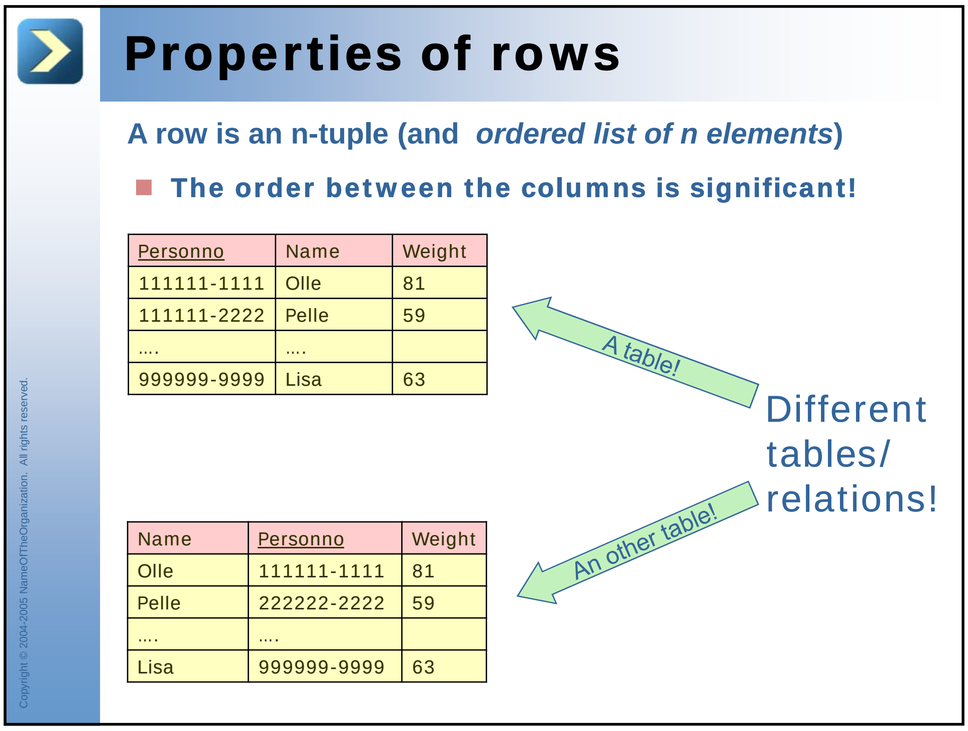
PERSON

Personno	Namn
111111-1111	Olle
111111-2222	Pelle
999999-9999	Lisa

PERSON

Personno	Namn
111111-2222	Pelle
111111-1111	Olle
999999-9999	Lisa

The same relation!



Properties of rows

A row is an n-tuple (and *ordered list of n elements*)

- **The order between the columns is significant!**

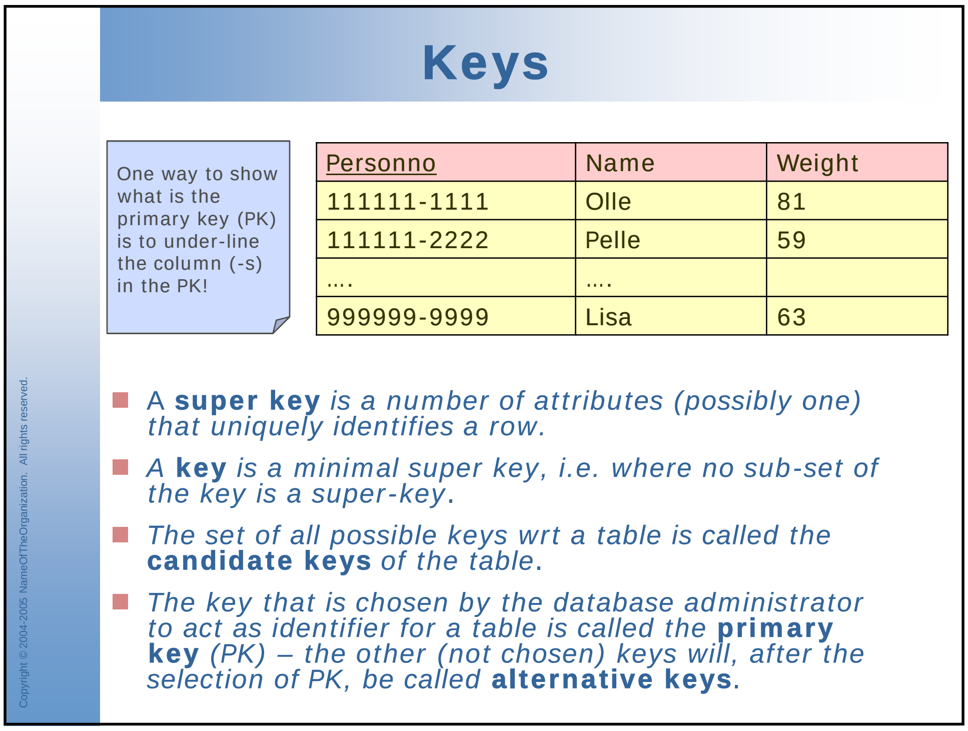
Personno	Name	Weight
111111-1111	Olle	81
111111-2222	Pelle	59
….	….	
999999-9999	Lisa	63

A table!

Different tables/ relations!

An other table!

Name	Personno	Weight
Olle	111111-1111	81
Pelle	222222-2222	59
….	….	
Lisa	999999-9999	63

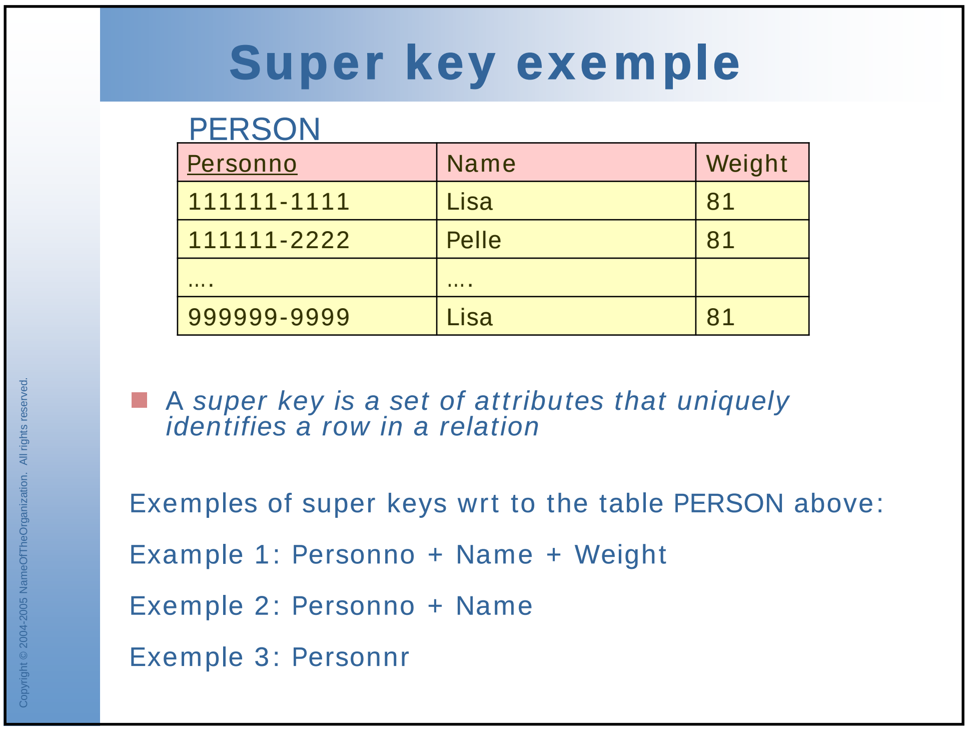


Keys

One way to show what is the primary key (PK) is to under-line the column (-s) in the PK!

Personno	Name	Weight
111111-1111	Olle	81
111111-2222	Pelle	59
….	….	
999999-9999	Lisa	63

- A **super key** *is a number of attributes (possibly one) that uniquely identifies a row.*
- *A* **key** *is a minimal super key, i.e. where no sub-set of the key is a super-key.*
- *The set of all possible keys wrt a table is called the* **candidate keys** *of the table.*
- *The key that is chosen by the database administrator to act as identifier for a table is called the* **primary key** *(PK) – the other (not chosen) keys will, after the selection of PK, be called* **alternative keys**.



Super key exemple

PERSON

Personno	Name	Weight
111111-1111	Lisa	81
111111-2222	Pelle	81
….	….	
999999-9999	Lisa	81

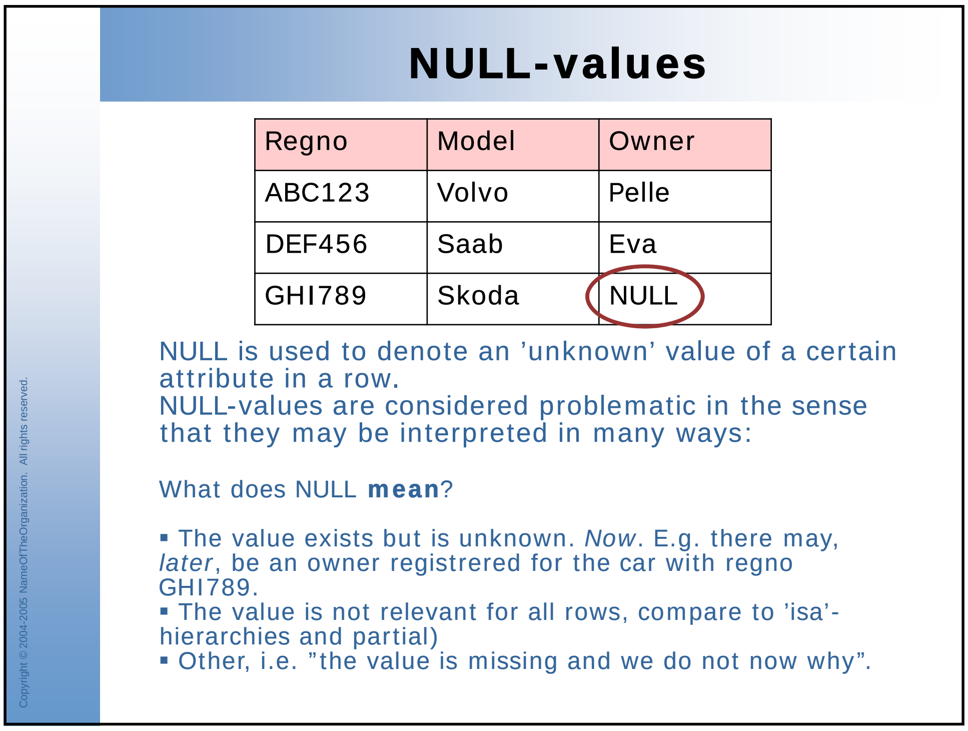
- A *super key is a set of attributes that uniquely identifies a row in a relation*

Exemples of super keys wrt to the table PERSON above:

Example 1: Personno + Name + Weight

Exemple 2: Personno + Name

Exemple 3: Personnr



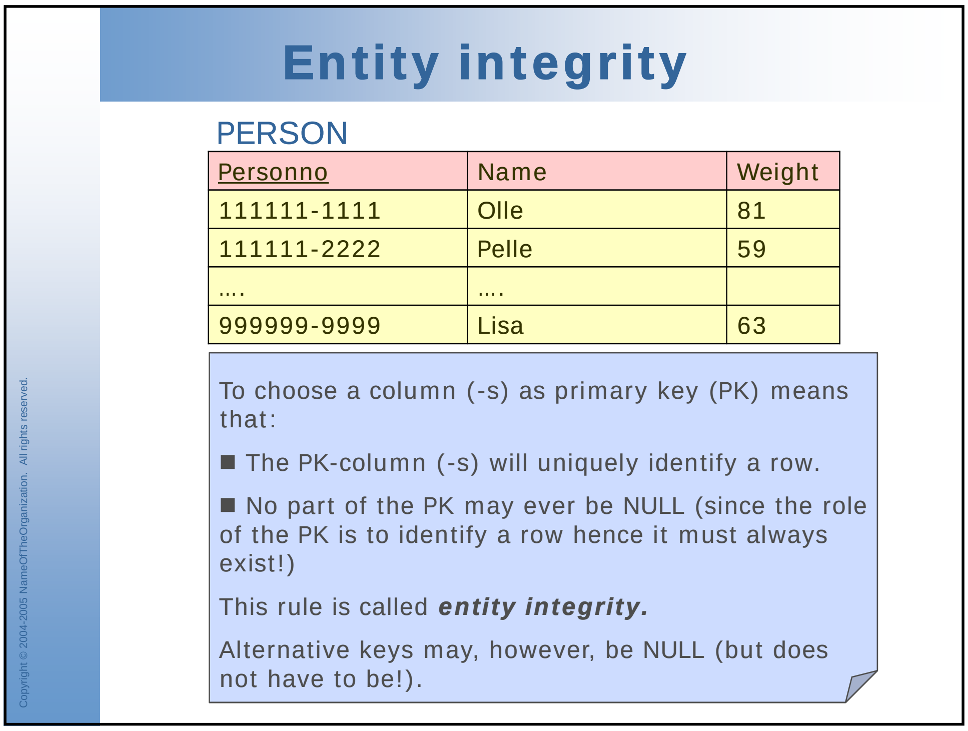
NULL-values

Regno	Model	Owner
ABC123	Volvo	Pelle
DEF456	Saab	Eva
GHI789	Skoda	NULL

NULL is used to denote an ’unknown’ value of a certain attribute in a row.
NULL-values are considered problematic in the sense that they may be interpreted in many ways:

What does NULL **mean**?

- The value exists but is unknown. *Now*. E.g. there may, *later*, be an owner registrered for the car with regno GHI789.
- The value is not relevant for all rows, compare to ’isa’-hierarchies and partial)
- Other, i.e. ”the value is missing and we do not now why”.



Entity integrity

PERSON

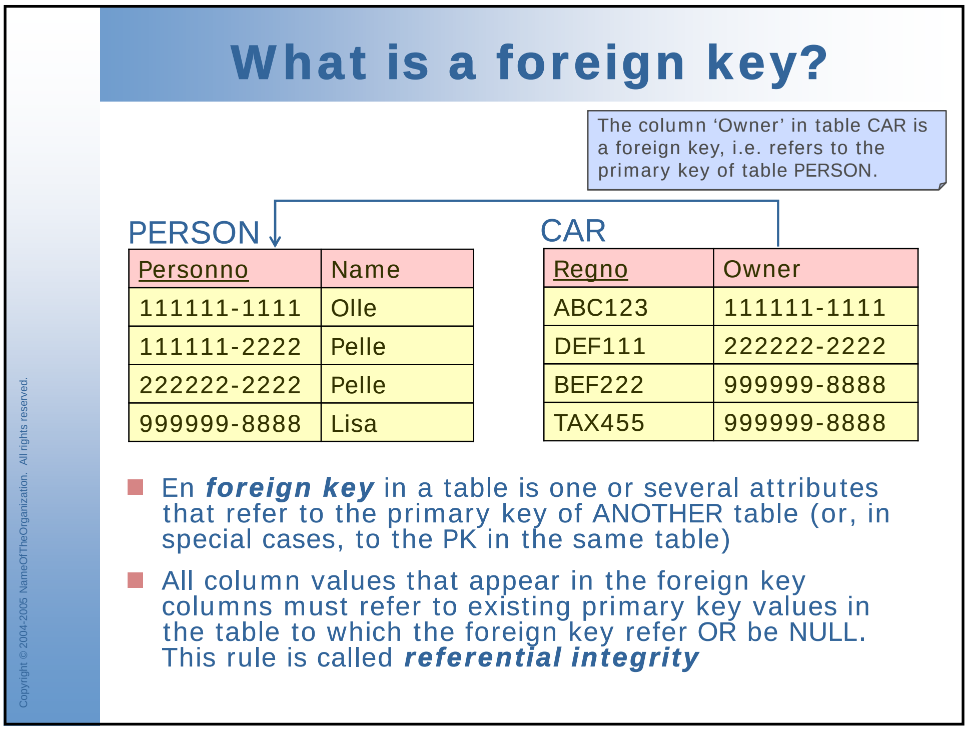
Personno	Name	Weight
111111-1111	Olle	81
111111-2222	Pelle	59
….	….	
999999-9999	Lisa	63

To choose a column (-s) as primary key (PK) means that:

- The PK-column (-s) will uniquely identify a row.
- No part of the PK may ever be NULL (since the role of the PK is to identify a row hence it must always exist!)

This rule is called ***entity integrity.***

Alternative keys may, however, be NULL (but does not have to be!).



What is a foreign key?

The column 'Owner' in table CAR is a foreign key, i.e. refers to the primary key of table PERSON.

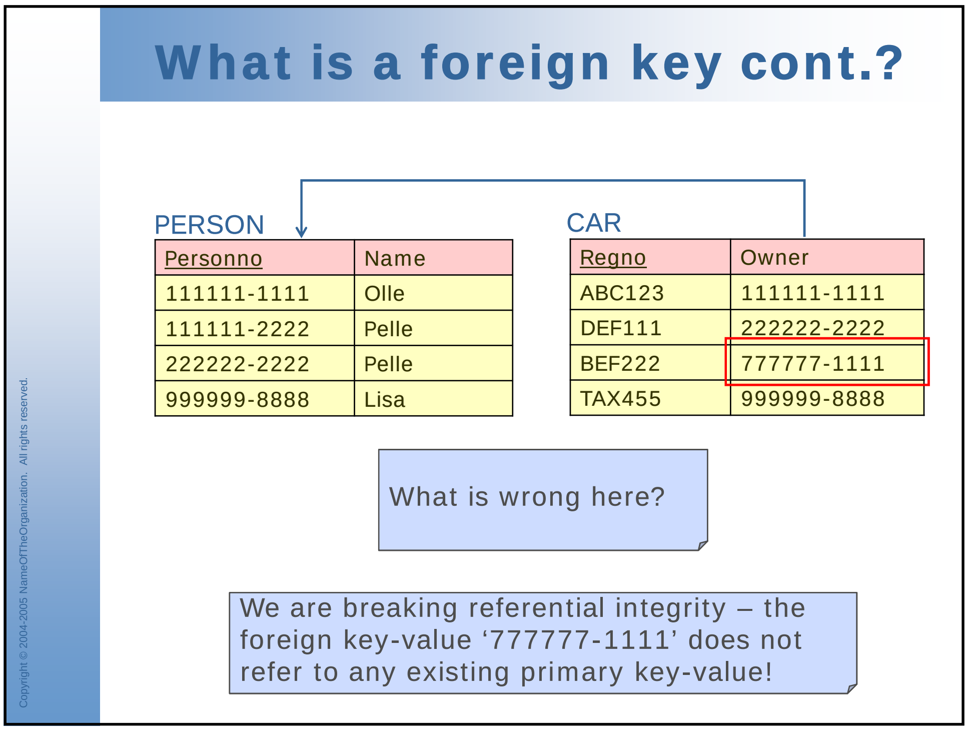
PERSON

<u>Personno</u>	Name
111111-1111	Olle
111111-2222	Pelle
222222-2222	Pelle
999999-8888	Lisa

CAR

<u>Regno</u>	Owner
ABC123	111111-1111
DEF111	222222-2222
BEF222	999999-8888
TAX455	999999-8888

- En ***foreign key*** in a table is one or several attributes that refer to the primary key of ANOTHER table (or, in special cases, to the PK in the same table)
- All column values that appear in the foreign key columns must refer to existing primary key values in the table to which the foreign key refer OR be NULL. This rule is called ***referential integrity***



What is a foreign key cont.?

PERSON

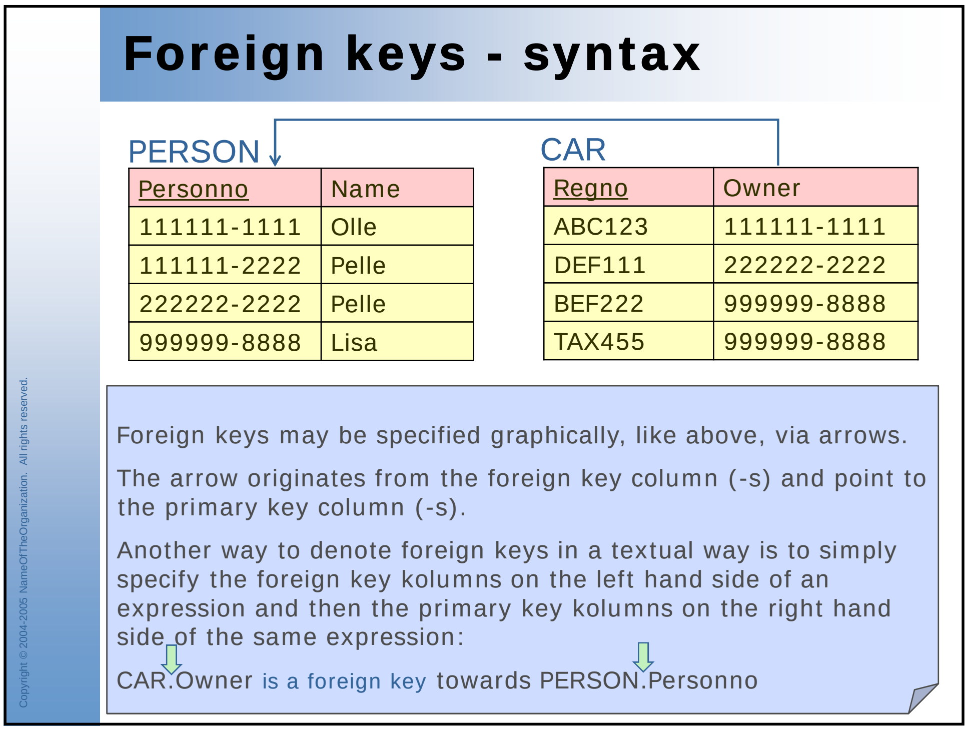
<u>Personno</u>	Name
111111-1111	Olle
111111-2222	Pelle
222222-2222	Pelle
999999-8888	Lisa

CAR

<u>Regno</u>	Owner
ABC123	111111-1111
DEF111	222222-2222
BEF222	777777-1111
TAX455	999999-8888

What is wrong here?

We are breaking referential integrity – the foreign key-value '777777-1111' does not refer to any existing primary key-value!



Foreign keys - syntax

PERSON

<u>Personno</u>	Name
111111-1111	Olle
111111-2222	Pelle
222222-2222	Pelle
999999-8888	Lisa

CAR

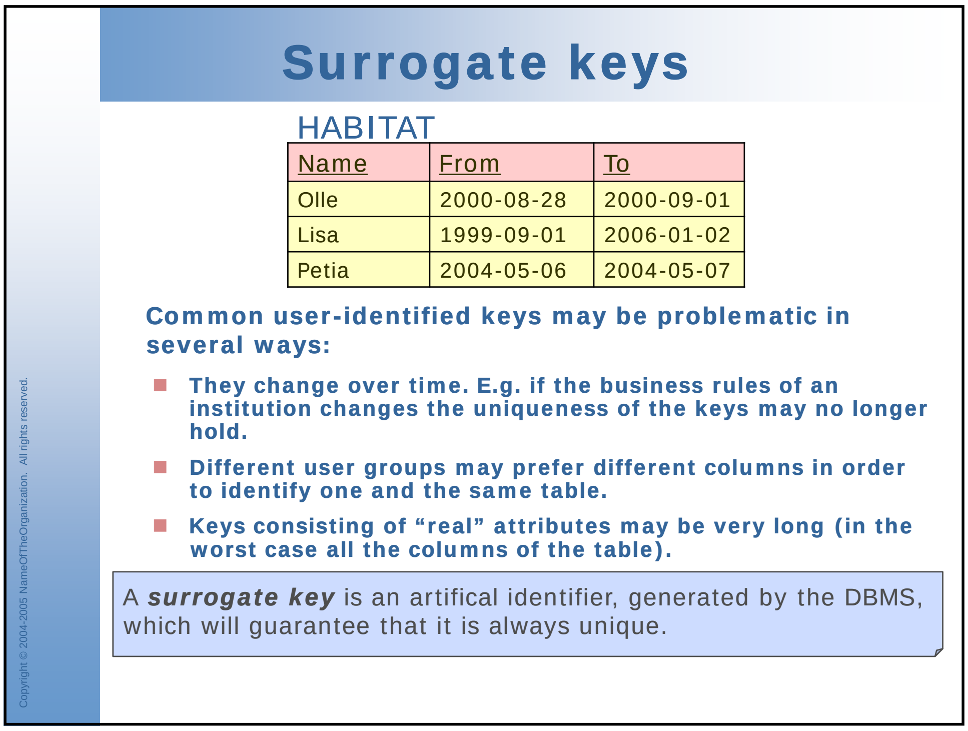
<u>Regno</u>	Owner
ABC123	111111-1111
DEF111	222222-2222
BEF222	999999-8888
TAX455	999999-8888

Foreign keys may be specified graphically, like above, via arrows.

The arrow originates from the foreign key column (-s) and point to the primary key column (-s).

Another way to denote foreign keys in a textual way is to simply specify the foreign key kolumns on the left hand side of an expression and then the primary key kolumns on the right hand side of the same expression:

CAR.Owner is a foreign key towards PERSON.Personno



Surrogate keys

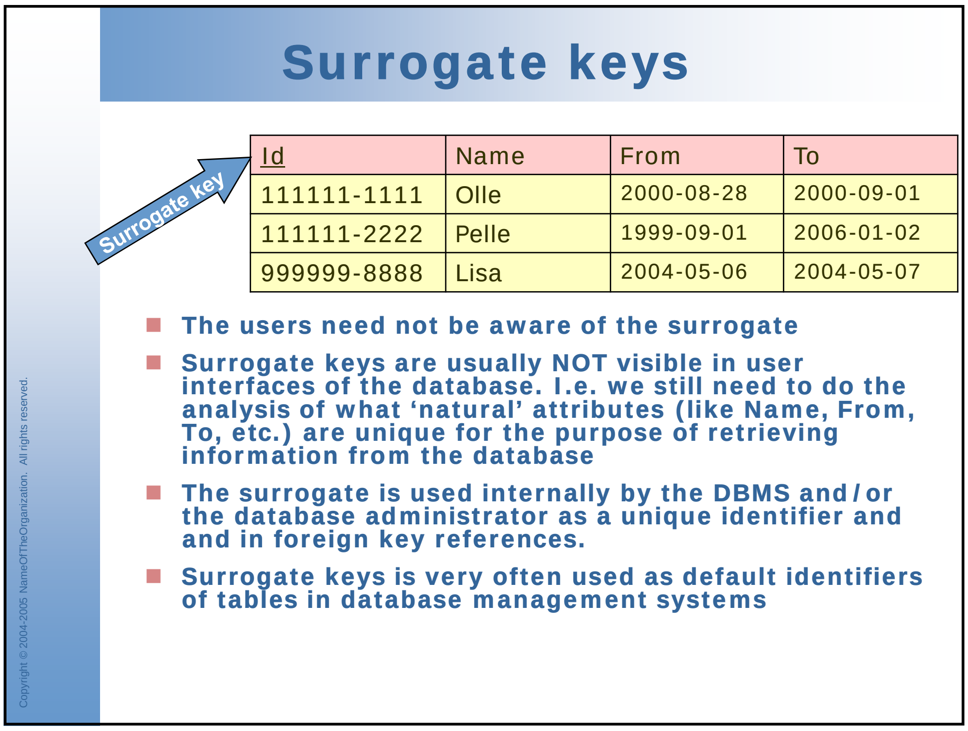
HABITAT

<u>Name</u>	<u>From</u>	<u>To</u>
Olle	2000-08-28	2000-09-01
Lisa	1999-09-01	2006-01-02
Petia	2004-05-06	2004-05-07

Common user-identified keys may be problematic in several ways:

- **They change over time. E.g. if the business rules of an institution changes the uniqueness of the keys may no longer hold.**
- **Different user groups may prefer different columns in order to identify one and the same table.**
- **Keys consisting of “real” attributes may be very long (in the worst case all the columns of the table).**

A ***surrogate key*** is an artifical identifier, generated by the DBMS, which will guarantee that it is always unique.

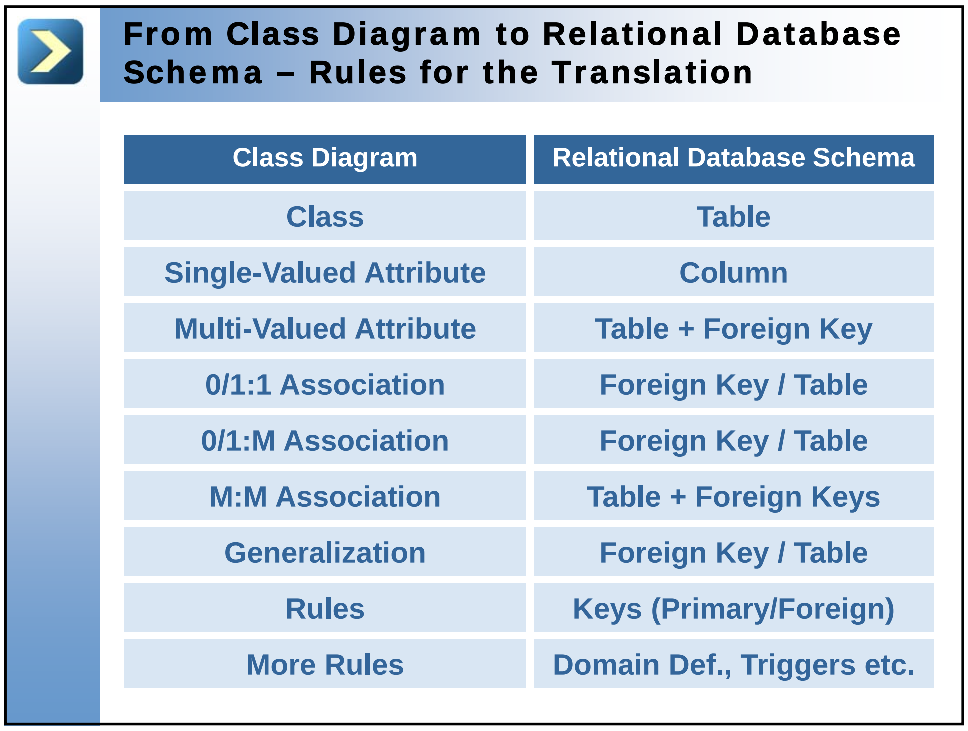


Surrogate keys

Surrogate key

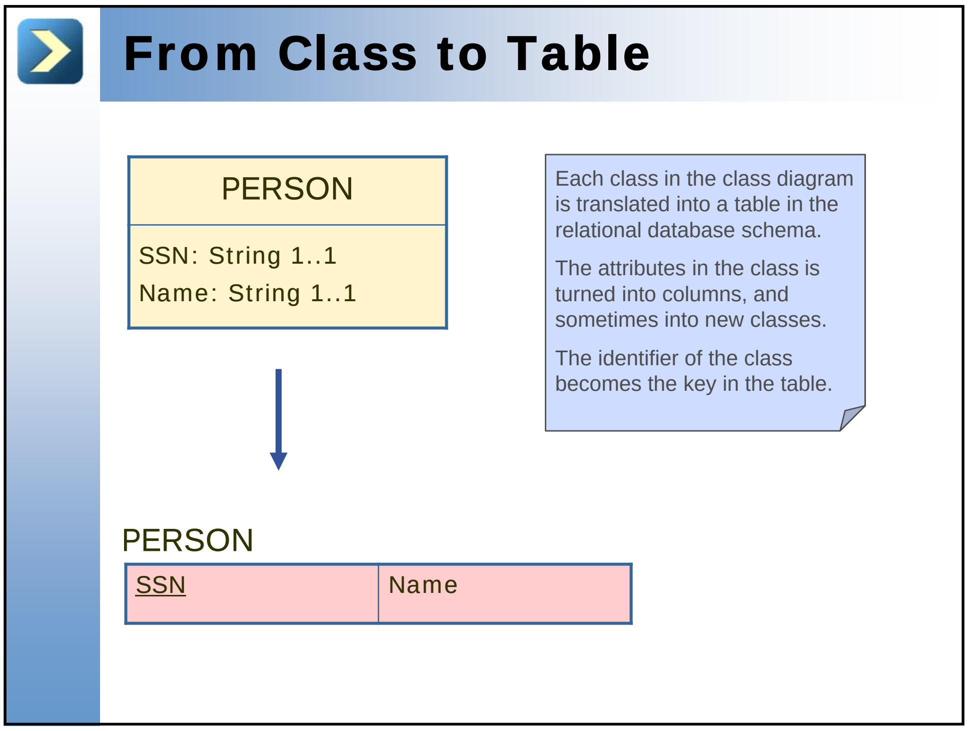
Id	Name	From	To
111111-1111	Olle	2000-08-28	2000-09-01
111111-2222	Pelle	1999-09-01	2006-01-02
999999-8888	Lisa	2004-05-06	2004-05-07

- The users need not be aware of the surrogate
- Surrogate keys are usually NOT visible in user interfaces of the database. I.e. we still need to do the analysis of what ‘natural’ attributes (like Name, From, To, etc.) are unique for the purpose of retrieving information from the database
- The surrogate is used internally by the DBMS and / or the database administrator as a unique identifier and and in foreign key references.
- Surrogate keys is very often used as default identifiers of tables in database management systems



From Class Diagram to Relational Database Schema – Rules for the Translation

Class Diagram	Relational Database Schema
Class	Table
Single-Valued Attribute	Column
Multi-Valued Attribute	Table + Foreign Key
0/1:1 Association	Foreign Key / Table
0/1:M Association	Foreign Key / Table
M:M Association	Table + Foreign Keys
Generalization	Foreign Key / Table
Rules	Keys (Primary/Foreign)
More Rules	Domain Def., Triggers etc.



From Class to Table

PERSON

SSN: String 1..1
Name: String 1..1

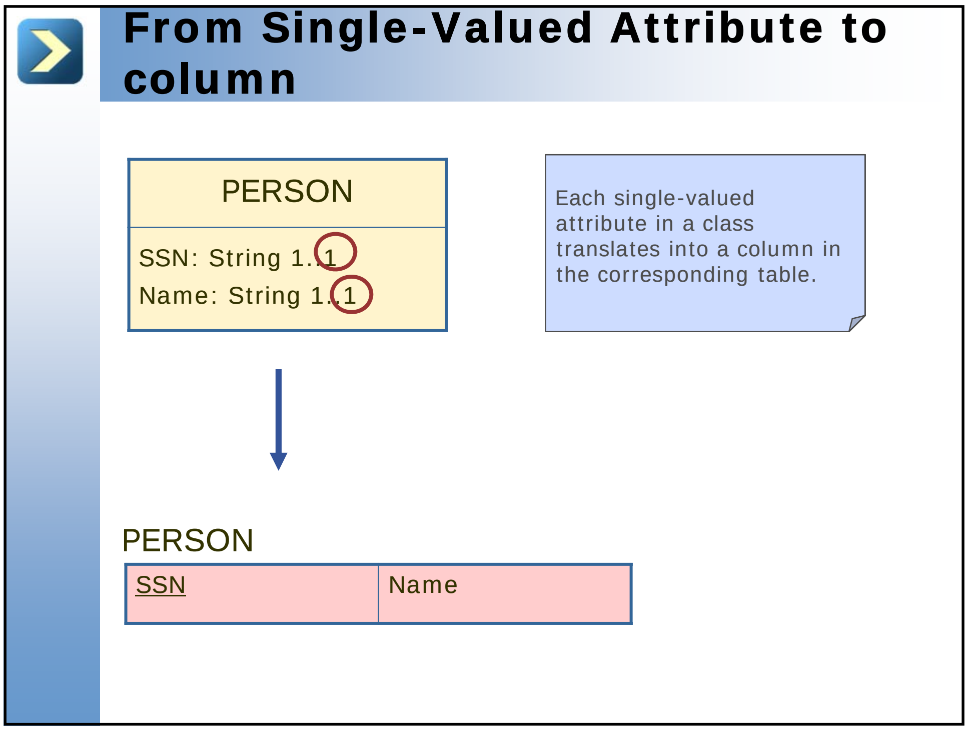
Each class in the class diagram is translated into a table in the relational database schema.

The attributes in the class is turned into columns, and sometimes into new classes.

The identifier of the class becomes the key in the table.

PERSON

<u>SSN</u>	Name



From Single-Valued Attribute to column

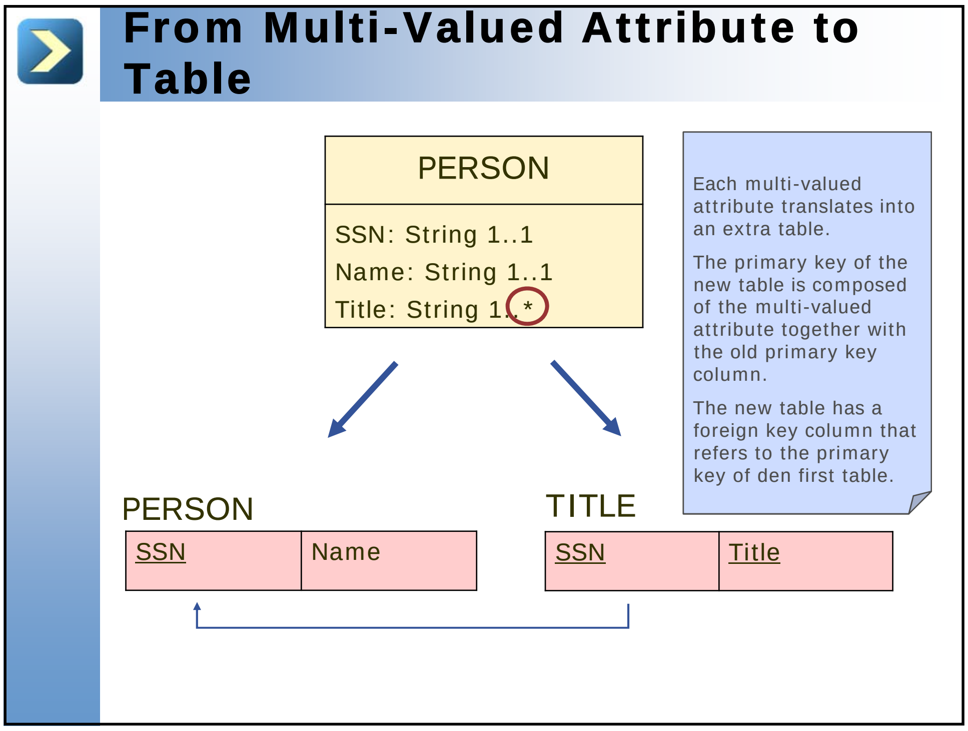
PERSON

SSN: String 1..1
Name: String 1..1

Each single-valued attribute in a class translates into a column in the corresponding table.

PERSON

<u>SSN</u>	Name

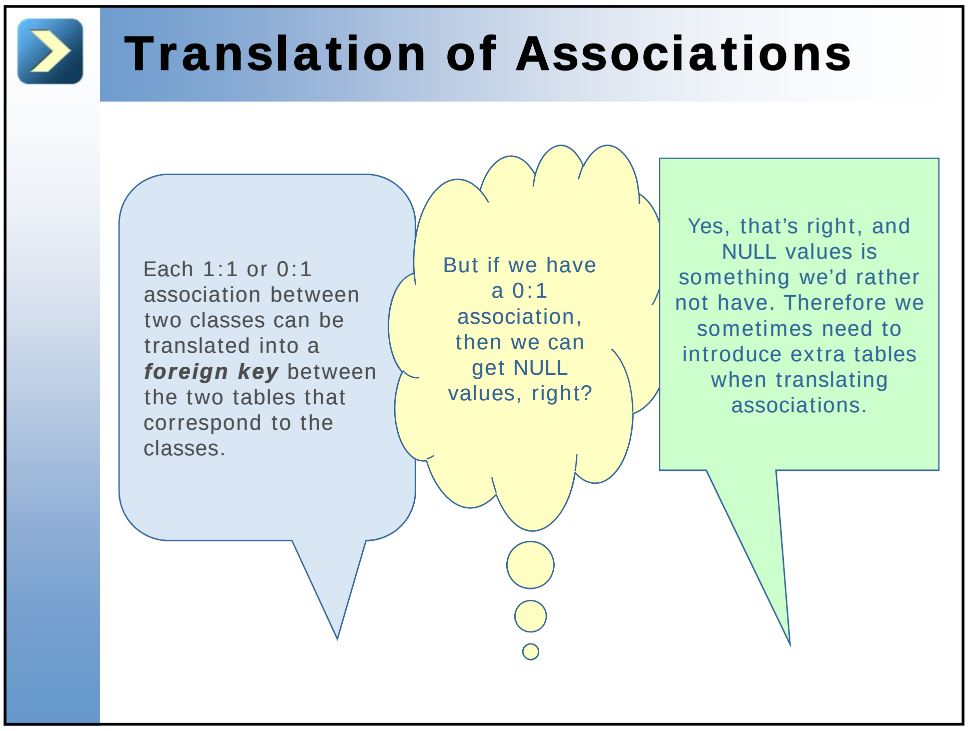


From Multi-Valued Attribute to Table

Each multi-valued attribute translates into an extra table.

The primary key of the new table is composed of the multi-valued attribute together with the old primary key column.

The new table has a foreign key column that refers to the primary key of den first table.

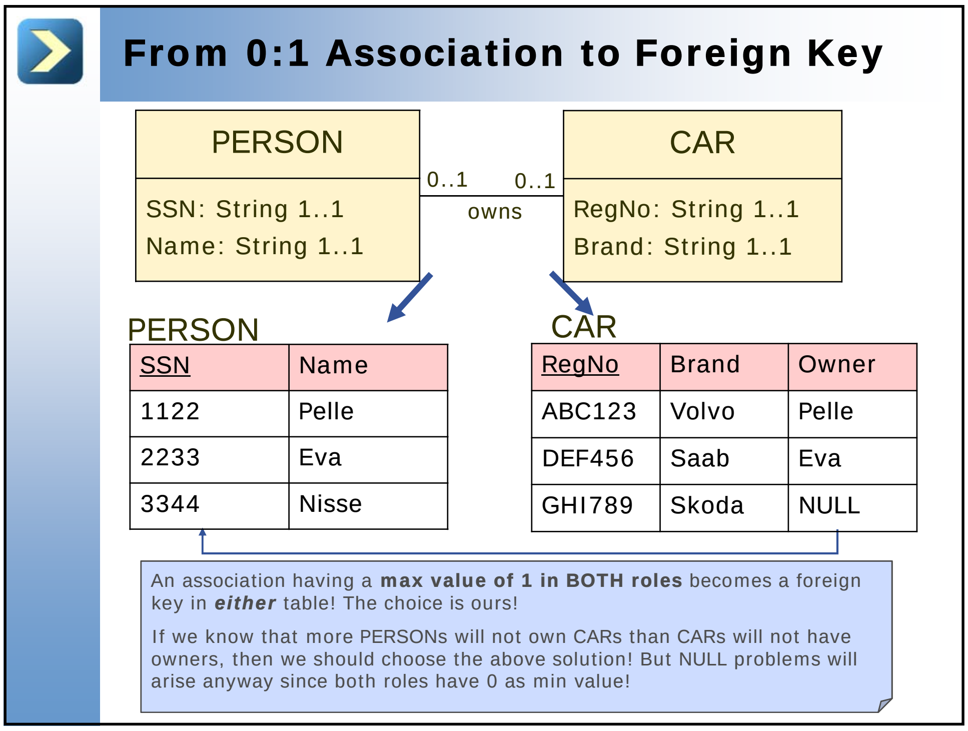


Translation of Associations

Each 1:1 or 0:1 association between two classes can be translated into a ***foreign key*** between the two tables that correspond to the classes.

But if we have a 0:1 association, then we can get NULL values, right?

Yes, that's right, and NULL values is something we'd rather not have. Therefore we sometimes need to introduce extra tables when translating associations.



From 0:1 Association to Foreign Key

PERSON
SSN: String 1..1
Name: String 1..1
0..1
0..1
owns
CAR
RegNo: String 1..1
Brand: String 1..1

PERSON

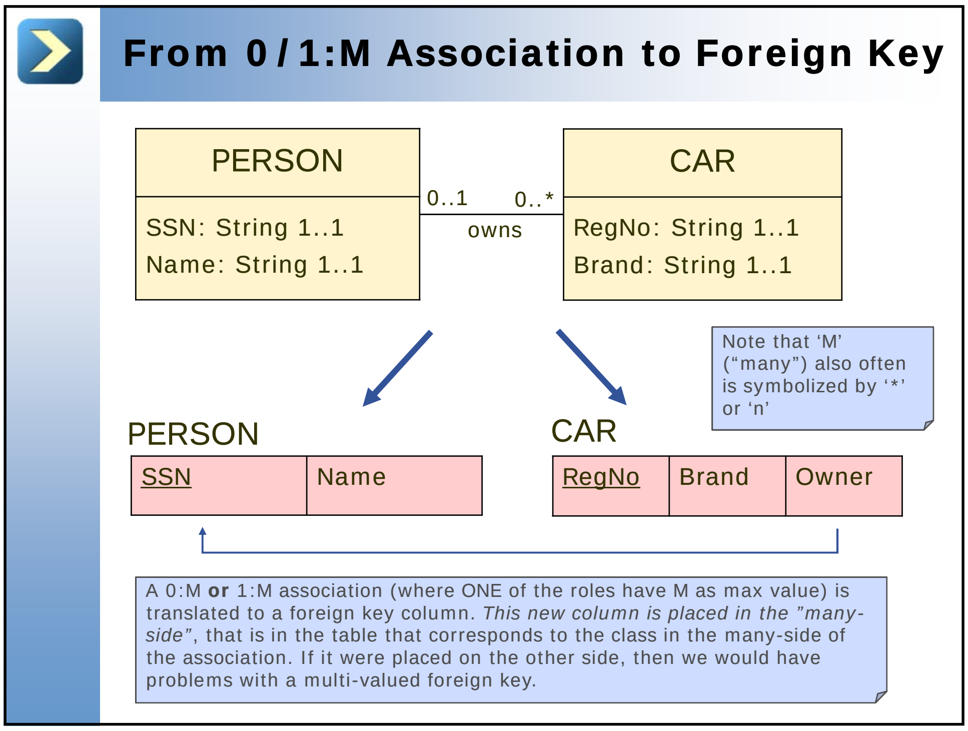
SSN	Name
1122	Pelle
2233	Eva
3344	Nisse

CAR

RegNo	Brand	Owner
ABC123	Volvo	Pelle
DEF456	Saab	Eva
GHI789	Skoda	NULL

An association having a **max value of 1 in BOTH roles** becomes a foreign key in ***either*** table! The choice is ours!

If we know that more PERSONs will not own CARs than CARs will not have owners, then we should choose the above solution! But NULL problems will arise anyway since both roles have 0 as min value!



From 0 / 1:M Association to Foreign Key

PERSON
SSN: String 1..1
Name: String 1..1
0..1
0..*
owns
CAR
RegNo: String 1..1
Brand: String 1..1

Note that ‘M’ (“many”) also often is symbolized by ‘*’ or ‘n’

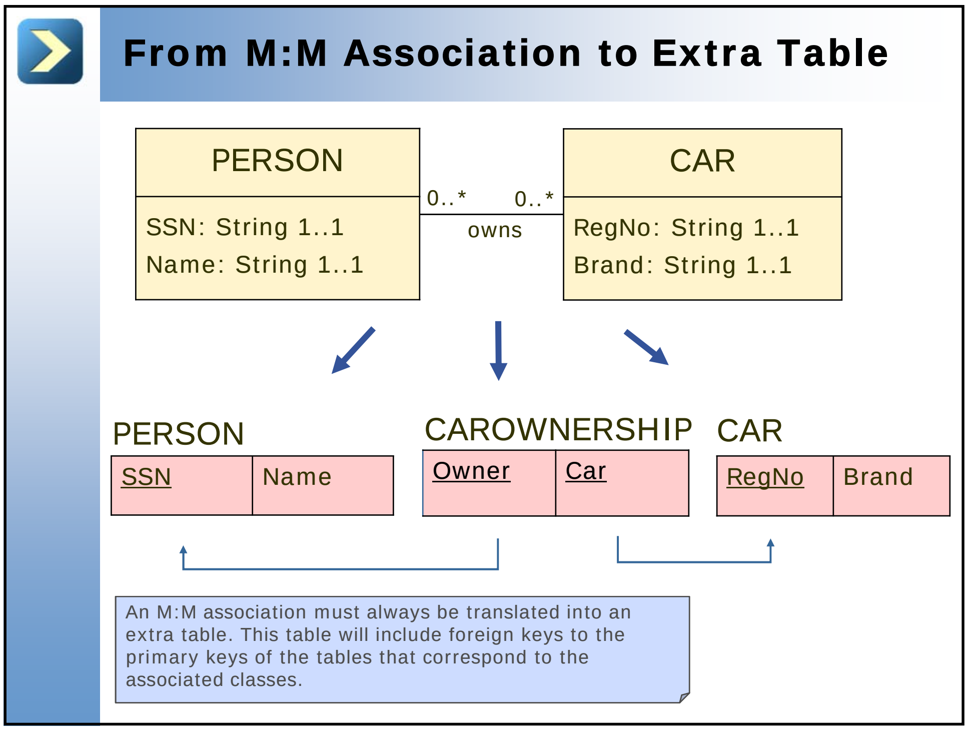
PERSON

SSN	Name

CAR

RegNo	Brand	Owner

A 0:M **or** 1:M association (where ONE of the roles have M as max value) is translated to a foreign key column. *This new column is placed in the ”many-side”*, that is in the table that corresponds to the class in the many-side of the association. If it were placed on the other side, then we would have problems with a multi-valued foreign key.



From M:M Association to Extra Table

PERSON

<u>SSN</u>	Name

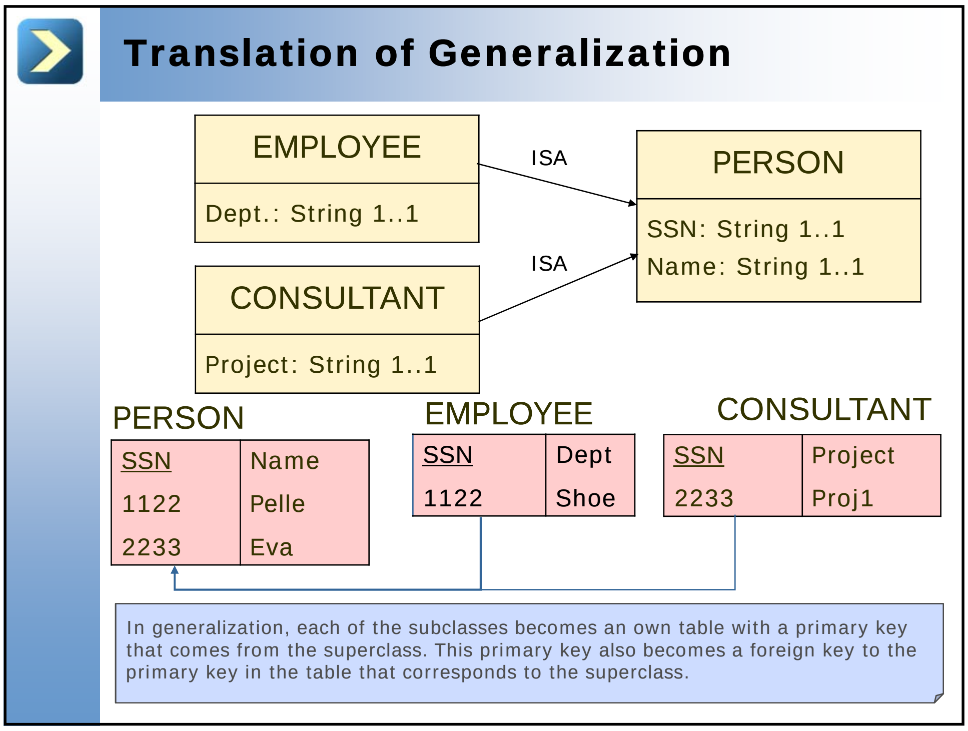
CAROWNERSHIP

<u>Owner</u>	<u>Car</u>

CAR

<u>RegNo</u>	Brand

An M:M association must always be translated into an extra table. This table will include foreign keys to the primary keys of the tables that correspond to the associated classes.



Translation of Generalization

PERSON

<u>SSN</u>	Name
1122	Pelle
2233	Eva

EMPLOYEE

<u>SSN</u>	Dept
1122	Shoe

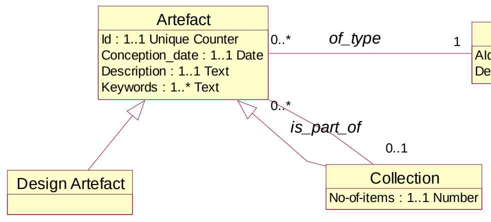
CONSULTANT

<u>SSN</u>	Project
2233	Proj1

In generalization, each of the subclasses becomes an own table with a primary key that comes from the superclass. This primary key also becomes a foreign key to the primary key in the table that corresponds to the superclass.

Exercise

Example UML class schema:



Artefact(<u>Id</u>, Conception_date, Description, myCollection,Aid),
Artefact.myCollection is FK towards Collection.Id
Artefact.Aid is FK towards Artefact_type(Aid)

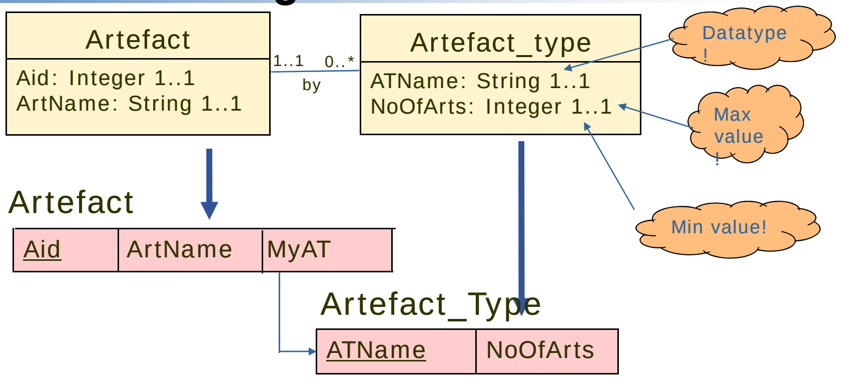
Keywords(<u>Id, keyword</u>), Keywords.Id is FK towards Artefact.Id

Design_artefact(<u>Id</u>), Design_artefact.Id is FK towards Artefact.id

Collection(<u>Id</u>, no_of_items), Collection.id is FK towards Artefact.id

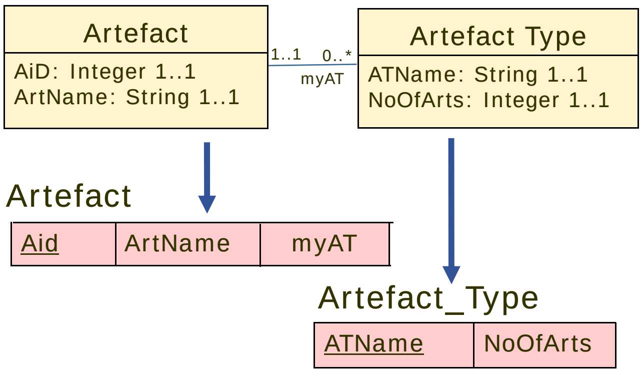
Artefact_type(<u>Aid</u>, Description)

Domain Rules and Foreign Key Rules Using DDL

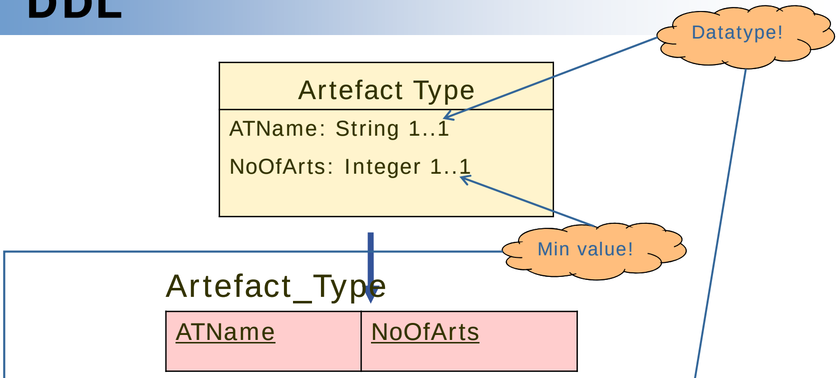


- **The tables above, including datatypes and foreign key (and other rules) can be defined using SQL DDL**
- **SQL has a DDL part (less well-known than the DML part)**
- **DDL – Data Definition Language**
  DML – Data Manipulation Language
- **Using DDL we *define* tables, rules etc.**
  Using DML we can then write queries against the tables we created

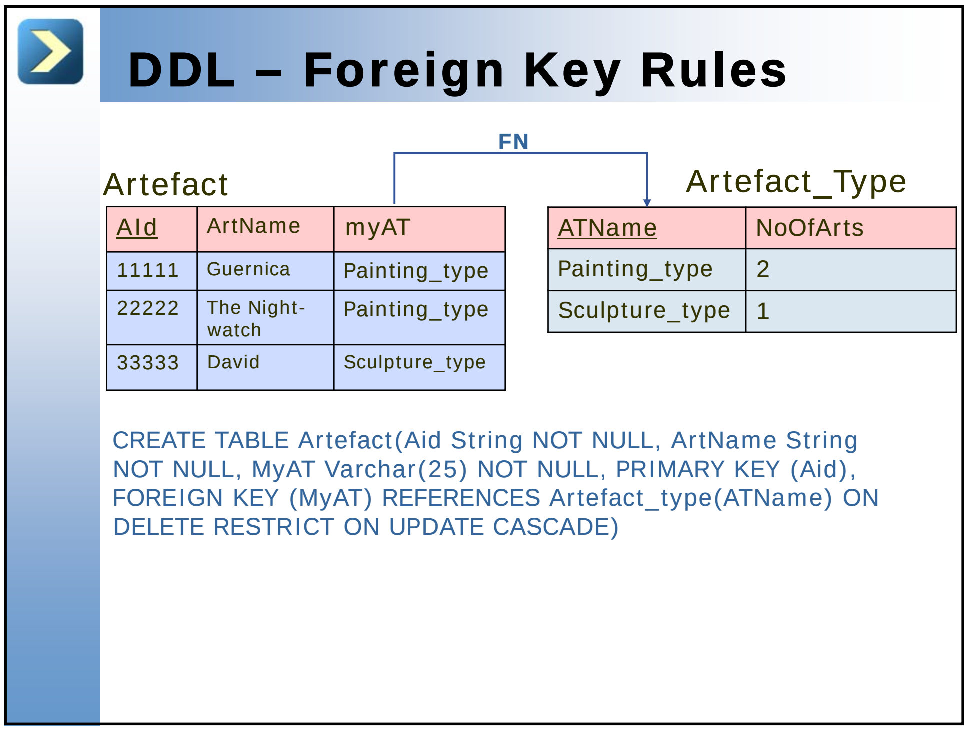
DDL



DDL



CREATE TABLE Artefact_Type (ATName Varchar(25) NOT NULL, NoOfArts Integer NOT NULL, PRIMARY KEY (ATName))



DDL – Foreign Key Rules

FN

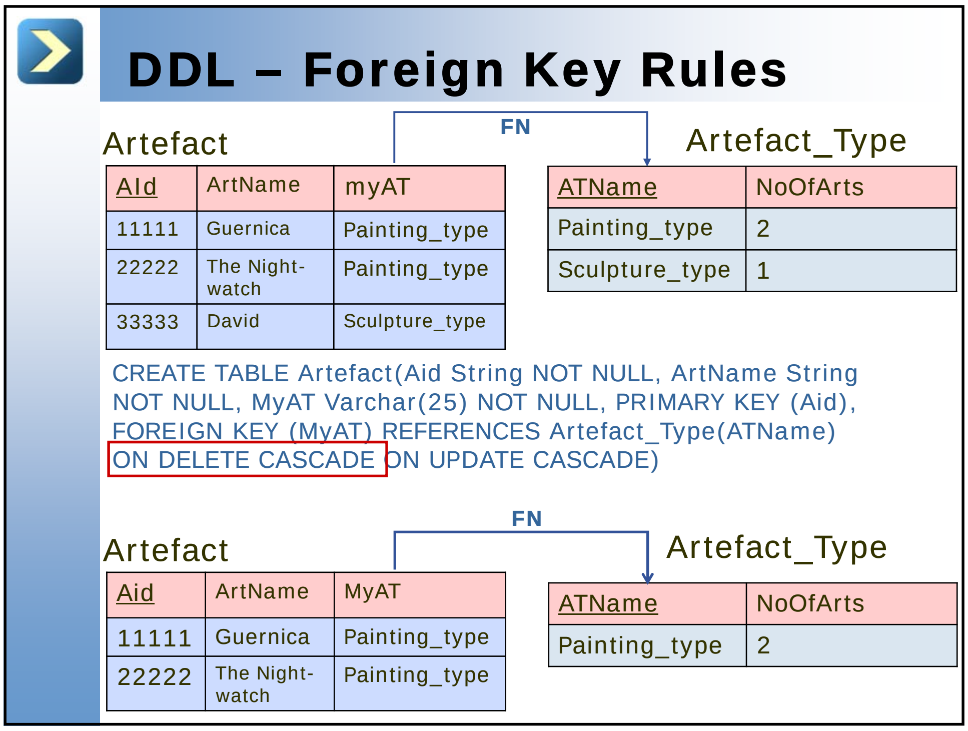
Artefact

AId	ArtName	myAT
11111	Guernica	Painting_type
22222	The Night-watch	Painting_type
33333	David	Sculpture_type

Artefact_Type

ATName	NoOfArts
Painting_type	2
Sculpture_type	1

CREATE TABLE Artefact(Aid String NOT NULL, ArtName String NOT NULL, MyAT Varchar(25) NOT NULL, PRIMARY KEY (Aid), FOREIGN KEY (MyAT) REFERENCES Artefact_type(ATName) ON DELETE RESTRICT ON UPDATE CASCADE)



DDL – Foreign Key Rules

FN

Artefact

AId	ArtName	myAT
11111	Guernica	Painting_type
22222	The Night-watch	Painting_type
33333	David	Sculpture_type

Artefact_Type

ATName	NoOfArts
Painting_type	2
Sculpture_type	1

CREATE TABLE Artefact(Aid String NOT NULL, ArtName String NOT NULL, MyAT Varchar(25) NOT NULL, PRIMARY KEY (Aid), FOREIGN KEY (MyAT) REFERENCES Artefact_Type(ATName) $\boxed{\text{ON DELETE CASCADE}}$ ON UPDATE CASCADE)

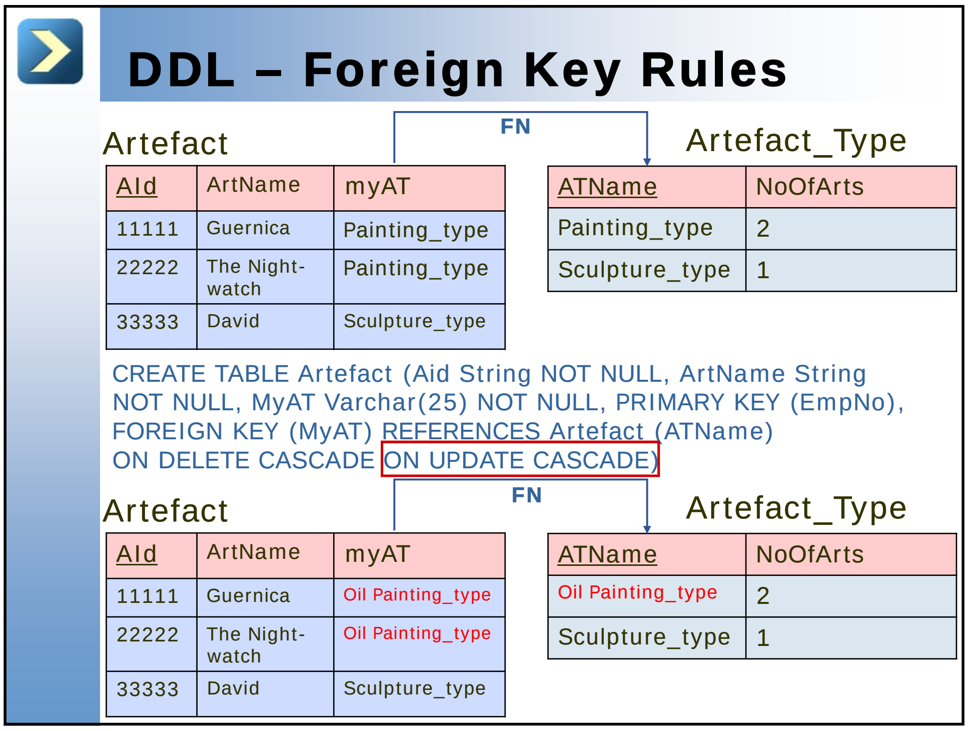
FN

Artefact

Aid	ArtName	MyAT
11111	Guernica	Painting_type
22222	The Night-watch	Painting_type

Artefact_Type

ATName	NoOfArts
Painting_type	2



DDL – Foreign Key Rules

FN

Artefact

AId	ArtName	myAT
11111	Guernica	Painting_type
22222	The Night-watch	Painting_type
33333	David	Sculpture_type

Artefact_Type

ATName	NoOfArts
Painting_type	2
Sculpture_type	1

CREATE TABLE Artefact (Aid String NOT NULL, ArtName String NOT NULL, MyAT Varchar(25) NOT NULL, PRIMARY KEY (EmpNo), FOREIGN KEY (MyAT) REFERENCES Artefact (ATName) ON DELETE CASCADE $\boxed{\text{ON UPDATE CASCADE)}}$

FN

Artefact

AId	ArtName	myAT
11111	Guernica	Oil Painting_type
22222	The Night-watch	Oil Painting_type
33333	David	Sculpture_type

Artefact_Type

ATName	NoOfArts
Oil Painting_type	2
Sculpture_type	1