1222 · 2022 800 ANNI

UNIVERSITÀ DEGLI STUDI DI PADOVA

The Relational Model

Basi di Dati

Bachelor's Degree in Computer Engineering
Academic Year 2024/2025

Ornella Irrera

Intelligent Interactive Information Access (IIIA) Hub
Department of Information Engineering
University of Padua

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Introduction

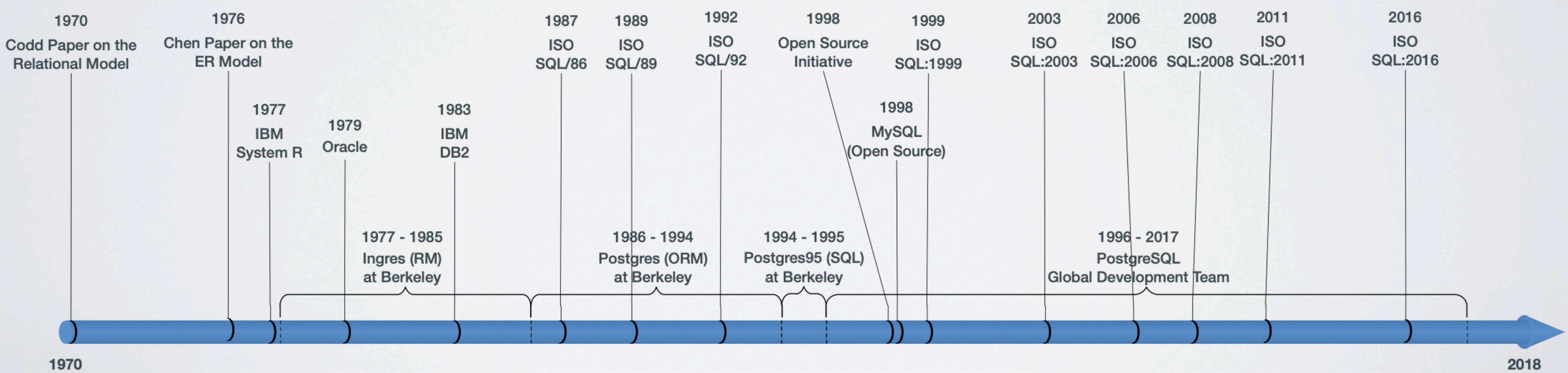
- Ornella Irrera: postdoctoral researcher @ IIIA Hub
- My email: ornella.irrera@unipd.it
- My office: Laboratorio basi di dati 305 DEI/G
- Material: the reference books you used for the first part of the course + slides and notes taken during the course
- Topics: Relational Model, Relational Algebra, Exercises

Outline

- Constructs of the relational model
- Integrity constraints in the relational model
- Mapping from the Entity-Relationship model to the relational model

The Relational Model

- The **relational model** has been proposed by E. F. Codd at IBM in 1970 to promote the **data independence**
- It is based on the notion of (mathematical) **relation** but with important differences
  - Relations are naturally represented by means of **tables**
- The relational models adopts a **value-based approach**: links and references among data in different structures (relations) are realized by means of comparison among values (and not pointers)



Relation

Mathematical Relation: Definition

- Given the sets $D_1, D_2, \ldots, D_n$ the **cartesian product** is the set of the **ordered n-uples**:

$$D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) | d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$$

- A **relation (mathematical)** is a subset of the cartesian product:

$$R \subseteq D_1 \times D_2 \times \ldots \times D_n$$

- $D_1, D_2, \ldots, D_n$ are the **domains** of the relation. A relation over n domains has **degree** n

- The number $|R|$ of n-ples is the **cardinality** of the relation

Mathematical Relation: Properties

- A (mathematical) relation is a **set** and therefore
  - there is **no ordering** among its n-uples
  - the n-uples are all **distinct**

- Each n-uple in a relation is **ordered** and therefore
  - the i-th value is associated to the i-th domain

Mathematical Relation: Example

Game $\subseteq$ String $\times$ String $\times$ Integer $\times$ Integer

Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Milan	Lazio	2	0

- The structure is **positional**
- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Mathematical Relation: Example

$$\text{Game} \subseteq \text{String} \times \text{String} \times \text{Integer} \times \text{Integer}$$

Inter	Juventus	3	2
Milan	Lazio	2	0
Juventus	Milan	3	1
Lazio	Roma	2	3

- The structure is **positional**
- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Mathematical Relation: Example

$$Game \subseteq String \times String \times Integer \times Integer$$

Juventus	3	Milan	1
Lazio	2	Roma	3
Inter	3	Juventus	2
Milan	2	Lazio	0

- The structure is **positional**
- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Mathematical Relation: Example

Game ⊆ String × String × Integer × Integer

Juventus	3	Milan	1
Lazio	2	Roma	3
Inter	3	Juventus	2
Milan	2	Lazio	0

- The structure is **positional**

- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Mathematical Relation: Example

$$Game \subseteq String \times String \times Integer \times Integer$$

Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Lazio	Roma	2	3

- The structure is **positional**
- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Mathematical Relation: Example

Game $\subseteq$ String $\times$ String $\times$ Integer $\times$ Integer

Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Lazio	Roma	2	3

- The structure is **positional**
- The role of a domain is disambiguated by its position: the first and third domains are the name and score of the home team, the second and fourth domains are the name and score of the visitor team

Relation in the Relational Model (1/2)

- A relation in the relational model is similar to a mathematical relation but with some differences:
  - the components of a relation are called **attributes**
  - each attribute is characterized by a **name** and a **set of values**, called **domain of the attribute**

- A relation can be represented as a **table** where **attributes** correspond to **columns** and **attribute names** are used as **column headers**
  - since each component of a relation is univocally identified by an attribute, the **ordering** among **attributes** is **irrelevant**:
  - differently from a mathematical relation, the structure is **not positional**

Relation in the Relational Model (2/2)

- Let $X = \{A_1, A_2, \ldots, A_n\}$ be the **set of attributes** and let $\mathcal{D} = \{D_1, D_2, \ldots, D_n\}$ be the **set of attribute domains**, the function

$$\begin{aligned} \text{dom}: \quad & X \rightarrow \mathcal{D} \\ & A_i \mapsto D_i \end{aligned}$$

maps each attribute into its domain

- We call **tuple** over a set of attributes

$$\begin{aligned} t_j: \quad & X \rightarrow D_i \\ & A_i \mapsto v_{i,j} \in D_i \end{aligned}$$

the function which maps each attribute into a value of its domain. We can use the following notation $t_j[A_i]$ or $t_j.A_i$ to indicate the value of an attribute

- A **relation** over a set of attributes X is a set of tuples over X

Relation in the Relational Model: Example

Game

Home	Visitor	Home Score	Visitor Score
Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Milan	Lazio	2	0

- There are 4 attributes in this relation and, in the tabular representation, they correspond to the columns of the table; the **tuples** (n-ples) correspond to the rows.
- The attribute domains are String for **Home** and **Visitor**, and Integer for **Home Score** and **Visitor Score**. Typically, they are not indicated in the tabular representation of a relation

Relation in the Relational Model: Example

Game

Home	Visitor	Home Score	Visitor Score
Inter	Juventus	3	2
Milan	Lazio	2	0
Juventus	Milan	3	1
Lazio	Roma	2	3

- There are 4 attributes in this relation and, in the tabular representation, they correspond to the columns of the table; the **tuples** (n-ples) correspond to the rows.
- The attribute domains are String for **Home** and **Visitor**, and Integer for **Home Score** and **Visitor Score**. Typically, they are not indicated in the tabular representation of a relation

Relation in the Relational Model: Example

Game

Home	Home Score	Visitor	Visitor Score
Juventus	3	Milan	1
Lazio	2	Roma	3
Inter	3	Juventus	2
Milan	2	Lazio	0

- There are 4 attributes in this relation and, in the tabular representation, they correspond to the columns of the table; the **tuples** (n-ples) correspond to the rows.
- The attribute domains are String for **Home** and **Visitor**, and Integer for **Home Score** and **Visitor Score**. Typically, they are not indicated in the tabular representation of a relation

Relation in the Relational Model: Example

Game

Home	Visitor	Home Score	Visitor Score
Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Lazio	Roma	2	3

- There are 4 attributes in this relation and, in the tabular representation, they correspond to the columns of the table; the **tuples** (n-ples) correspond to the rows.
- The attribute domains are String for **Home** and **Visitor**, and Integer for **Home Score** and **Visitor Score**. Typically, they are not indicated in the tabular representation of a relation

Relation in the Relational Model: Example

Game

Home	Visitor	Home Score	Visitor Score
Juventus	Milan	3	1
Lazio	Roma	2	3
Inter	Juventus	3	2
Lazio	Roma	2	3

- There are 4 attributes in this relation and, in the tabular representation, they correspond to the columns of the table; the **tuples** (n-ples) correspond to the rows.
- The attribute domains are String for **Home** and **Visitor**, and Integer for **Home Score** and **Visitor Score**. Typically, they are not indicated in the tabular representation of a relation

Relation Schema

A relation schema $R(T)$ consists of a relation name R and a relation type T denoted by

$$R(A_1 : D_1, A_2 : D_2, \dots, A_n : D_n)$$

where the attributes are $A_1, A_2, \dots, A_n$ and the attribute domains are $D_1, D_2, \dots, D_n$.

- Two relation types are equal if they have the same **degree**, that is the number of attributes n, the attributes themselves and the domains of the attributes with the same name.
- The order of the attributes is irrelevant
- We can also use the compact notation

$$R(A_1, A_2, \dots, A_n)$$

Relation Instance

A **relation instance** r of the relation schema $R(A_1, A_2, \dots, A_n)$, denoted by $r(R)$, is **a set of tuples** $r = \{t_1, t_2, \dots, t_m\}$ where each tuple is a labeled list of values

$$t_j =< A_1 : v_{1,j}, A_2 : v_{2,j}, \dots, A_n : v_{n,j} >$$

- The value of attribute A_i in tuple t_j is indicated as
- $$t_j[A_i] = v_{i,j} \in D_i = \text{dom}(A_i)$$ $$t_j.A_i = v_{i,j} \in D_i = \text{dom}(A_i)$$
- The **cardinality** $|r| = m$ of a relation is the number of its tuples

Incomplete Information: The NULL Value

- It may not be always possible to associate an appropriate value with an attribute of a tuple

- To overcome this issue, we introduce a special value, called **NULL**, which does not belong to any domain, and we extend the

$$t_j[A_i] = v_{i,j} \in D_i \ \vee \ \text{NULL}$$

- **NULL** is not one of the “ordinary” values of a domain **D**
  - “unused values” may not exist
  - “unused values” may change over time
  - you need to treat such “unused values” as special cases in the applications
  - “unused values” would be different from domain to domain

Meaning of the NULL Value

Person

Name	Surname	Age	MsC	HomePhone
Mario	Rossi	32	NULL	0498271234
Giovanni	Verdi	NULL	Law	0498275678
Marta	Bruni	28	Medicine	NULL

- **Value undefined**: an appropriate value for a given instance does not exist
  - Master Degree (MsC) does not apply to those people who have not attended university
- **Value not available**: an appropriate value for a given instance exists but it is not known in a given moment
  - You may not know the Age of a Person
- **Value unknown**: an appropriate value for a given instance may or may not exist
  - A Person may or may not have an home phone number because the contract is still in progress

Relational Model: Value-based Approach

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
8765	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Course
3456	30	03
3456	24	02
9283	28	01
6554	26	01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	Chemistry	Bruni
03	Physics	Rossi

Relational Model: Value-based Approach

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
8765	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Course
3456	30	03
3456	24	02
9283	28	01
6554	26	01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	Chemistry	Bruni
03	Physics	Rossi

Advantages of the Value-based Approach

- **Independence** from **physical data structures**, which may change

- Data are more **portable** from one system to another one

- Values allow for **bi-directionality** (while physical pointers are not)

Database Schema

A database schema (relational schema) consists of a set of relation schemas $R_i(T_i)$ with different names and denoted by

$$\mathbf{R} = \{R_1(T_1), R_2(T_2), \ldots, R_n(T_n)\}$$

- The integrity constraints (discussed in the following) are integral part of the database schema

Database Instance

A **database instance** $\mathbf{r}$ of the database schema $\mathbf{R} = \{R_1(T_1), R_2(T_2), \ldots, R_n(T_n)\}$ is a set of relation instances

$$\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$$

Database Schema and Instance: Example

$$\mathbf{R} = \{\text{Student}(\text{BadgeNumber}, \text{Name}, \text{Surname}, \text{BirthDate}),$$
$$\text{Worker}(\text{BadgeNumber})\}$$

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
8765	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Worker

BadgeNumber
6554
8765

Integrity Constraints

Integrity Constraints in the Relational Model

An integrity constraint is a condition expressed on the database schema (intensional level) and it must be complied with by the database instances (extensional level) which represent a correct state for the application

- Each constraint is a boolean predicate which maps each database instance into **true** if the constraint is complied with or **false** otherwise
- We add a set of constraints to the database schema and we consider **correct** only the database instances which comply with **all the constraints**

Integrity Constraints: Classification

- **Intra-relational**, concerning a single relation:
  - domain constraint
  - tuple constraint
  - key constraint
- **Inter-relational**, concerning two or more relations:
  - referential integrity constraint

Integrity Constraints: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Integrity Constraints: Example

Domain Constraints: a Mark must be in the range [0, 30]

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Integrity Constraints: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Domain Constraint: Title and Teacher cannot be NULL

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Integrity Constraints: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Tuple Constraint: Laude is possible if and only if Mark is 30

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Integrity Constraints: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Key Constraint: there do not exist two tuples with the same value for the attribute BadgeNumber

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Integrity Constraints: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
6554	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	32		02
9283	28	TRUE	01
7899	26		01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	NULL	NULL
03	Physics	Rossi

Referential Integrity Constraint: there is not tuple in the Student relation with value 7899 for the attribute BadgeNumber

Tuple Constraints

A tuple constraint defines conditions on the values of each tuple of a relation, independently from all the other tuples. A tuple constraint concerning only one attribute is called a domain constraint

- A tuple constraint on a relation R is a boolean sentence, using AND, OR and NOT to compose atomic boolean predicates comparing values of or expressions on the attributes of the relation R

Tuple Constraints: Example

Student	Mark	Laude	Course
3456	30	TRUE	03
3456	**32**		02
9283	**28**	**TRUE**	01
7899	26		01

- **Domain Constraint**

$$(\text{Mark} \geq 18) \text{ AND } (\text{Mark} \leq 30)$$

- **Tuple Constraint**

$$(\text{Mark} = 30) \text{ OR NOT } (\text{Laude} = \text{TRUE})$$

Mark = 30	Laude = TRUE	(Mark = 30) OR NOT (Laude = TRUE)
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE
TRUE	FALSE	TRUE
FALSE	FALSE	TRUE

Super-key

Let $R(T)$ be a relation schema and let SK be a sub-set of the attributes T. SK is a super-key for R if, for a relation instance $r(R)$, it holds

$$\forall t_1, t_2 \in r, \quad t_1[SK] \neq t_2[SK]$$

- In the relation instance there cannot be two different tuples which have the same values on all the attributes of the super-key
- Since relations are sets, a relation cannot contain the same tuple twice. As a consequence, there always exists at least one super-key for each relation, that is the whole set of attributes T

Key

A super-key SK of a relation schema $R(T)$ is a key K (with $K = SK$) if, for a relation instance $r(R)$ and for each proper subset K' of K, it holds

$$\exists t_1, t_2 \in r \mid \; t_1[K'] = t_2[K']$$

- A key is a **minimal super-key**, i.e. if you remove only one of its attributes, it is no more a key
- The same relation may have more than one key (super-key)

Super-key and Key: Example

Student

Badge	Name	Surname	BirthDate	Course
6554	Mario	Rossi	05/12/1982	Analysis
8765	Mario	Neri	07/04/1979	Physics
3219	Mario	Rossi	07/04/1979	Chemistry
9283	Piero	Neri	22/07/1983	Analysis
3456	Piero	Rossi	05/12/1982	Chemistry

- **Badge** is a key
  - Badge is a super-key
  - It is a minimal super-key since it consists of just one attribute
- (**Name**, **Surname**, **BirthDate**) is a key
  - the set Name, Surname, BirthDate is a super-key
  - It is a minimal super-key since no one of its subset is a super-key
- (**Name**, **Surname**, **BirthDate**, **Course**) is a super-key but not a key

Key Constraint

A **key constraint** is an **assertion** specifying that a set of attributes constitutes a key for a relational and that **all the valid relation instances** must **comply with** that **constraint**

- There is no limit to the number of key constraints on a relation

Importance of the Keys

- Keys guarantee the access to each datum in the database

- Each value is univocally accessible by
  - name of the relation
  - value of the key, which identifies a specific tuple in the relation
  - name of the attribute whose value has to be accessed

- As we will see in the following, keys are the main mean to link data in different relations

Keys and NULL Values

Student

Badge	Name	Surname	BirthDate	Course
NULL	Mario	NULL	05/12/1982	Analysis
8765	Mario	Neri	07/04/1979	Physics
3219	Mario	Rossi	07/04/1979	Chemistry
9283	Piero	Neri	NULL	Analysis
NULL	Piero	Rossi	05/12/1982	NULL

When there are NULL values, the values of the attributes constituting a key:

- do not allow us to identify tuples
- do not allow us to link to/from other relations

Primary Key Constraint

A primary key constraint is a key constraint also requesting that NULL values are not allowed for the attributes constituting the key

Student

<u>BadgeNumber</u>	Name	Surname	BirthDate	Course
6554	Mario	Rossi	05/12/1982	Chemistry
8765	Mario	Neri	07/04/1979	Analysis
3456	Piero	Rossi	05/12/1982	Physics

- The attributes constituting the primary key are <u>underlined</u>
- We can define **only one** primary key for each relation
- We typically chose as primary key the key with the **minimal number of attributes**

Foreign Key

A set of attributes $FK = (A_1, A_2, \ldots, A_n)$ of a relation schema R_1 is a foreign key of R_1 referring to a relation R_2 with primary key $PK = (B_1, B_2, \ldots, B_n)$ if

$$\mathrm{dom}(A_i) = \mathrm{dom}(B_i), \quad 1 \leq i \leq n$$

$$\forall t_1 \in r(R_1),\ \ t_1[FK] = \mathrm{NULL} \vee \exists t_2 \in r(R_2) \mid t_1[FK] = t_2[PK]$$

- The foreign key allows us to define the **referential integrity constraint** which requires that the property above is complied with by all the database instance
- Note the referential integrity constraint is enforced by **comparisons based on the values**

Referential Integrity Constraint: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
8765	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Course
3456	30	03
3456	24	02
9283	28	01
6554	26	01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	Chemistry	Bruni
03	Physics	Rossi

Referential Integrity Constraint: Example

Student

BadgeNumber	Name	Surname	BirthDate
6554	Mario	Rossi	05/12/1982
8765	Paolo	Neri	07/04/1979
9283	Luisa	Verdi	22/07/1983
3456	Marta	Rossi	14/02/1981

Exam

Student	Mark	Course
3456	30	03
3456	24	02
9283	28	01
6554	26	01

Course

Code	Title	Teacher
01	Analysis	Bianchi
02	Chemistry	Bruni
03	Physics	Rossi

Mapping from ER Model to Relational Model

Mapping of the (transformed) ER Schema into a Relational Schema

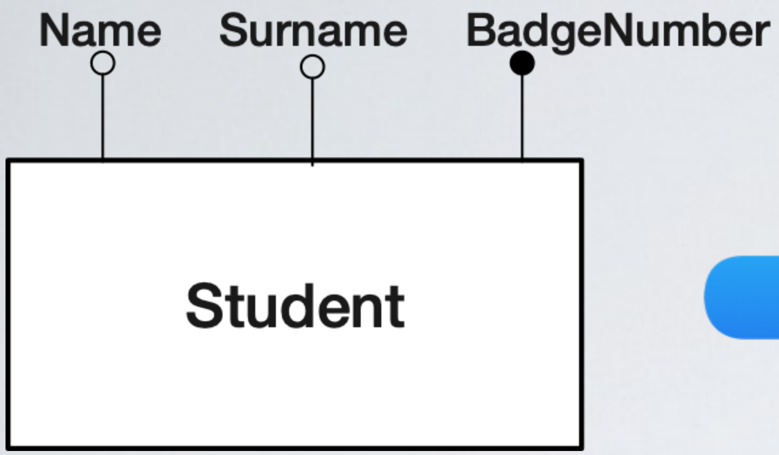
- The goal of the mapping is to pass from a (transformed) ER schema plus constraints to a relational schema plus constraints, representing the same information
- It is a semi-automatic process in the sense that most of the choices have been already made while transforming the ER schema
- The produced relational schema is typically a good starting point but we need to check its quality (normalization) and we may perform further transformations for performance reasons
- It consists of the following steps:
  - mapping of the entities into relations together with the respective constraints
  - mapping of the relationships into relations together with the respective constraints
  - mapping of external constraints

Mapping of Entities: General Rules

- Each entity **E** of the ER schema is mapped into a relation R_E of the relational schema

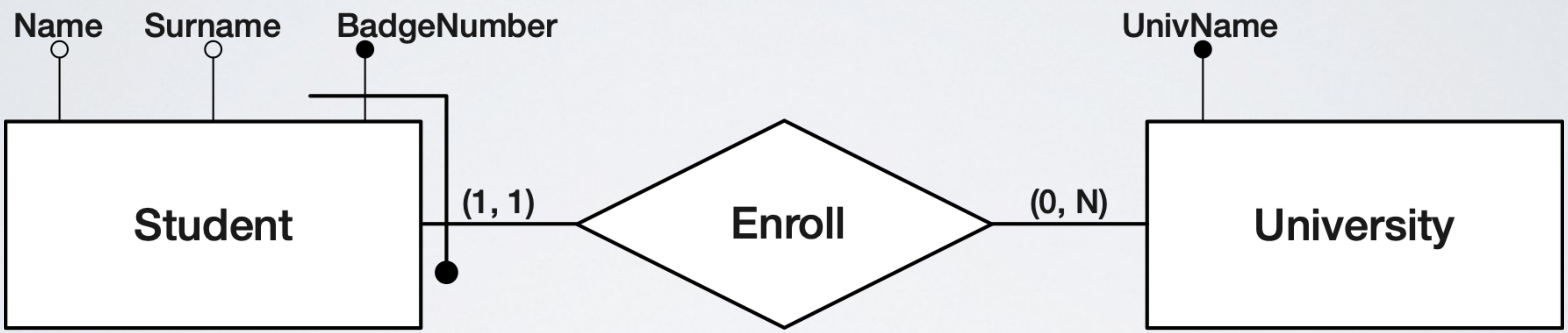
- The attributes of the relation R_E are
  - all the attributes of the entity **E**
  - the attributes deriving from the merging of relationships in R_E – for each merged relationship **Q** we add
    - the attributes of the relationships **Q**
    - the identifiers of all the other entities participating in **Q**

- The primary key of R_E is given by the principal identifier of **E**, that is attributes R_E and/or deriving from external identification



Student

BadgeNumber	Name	Surname



Student

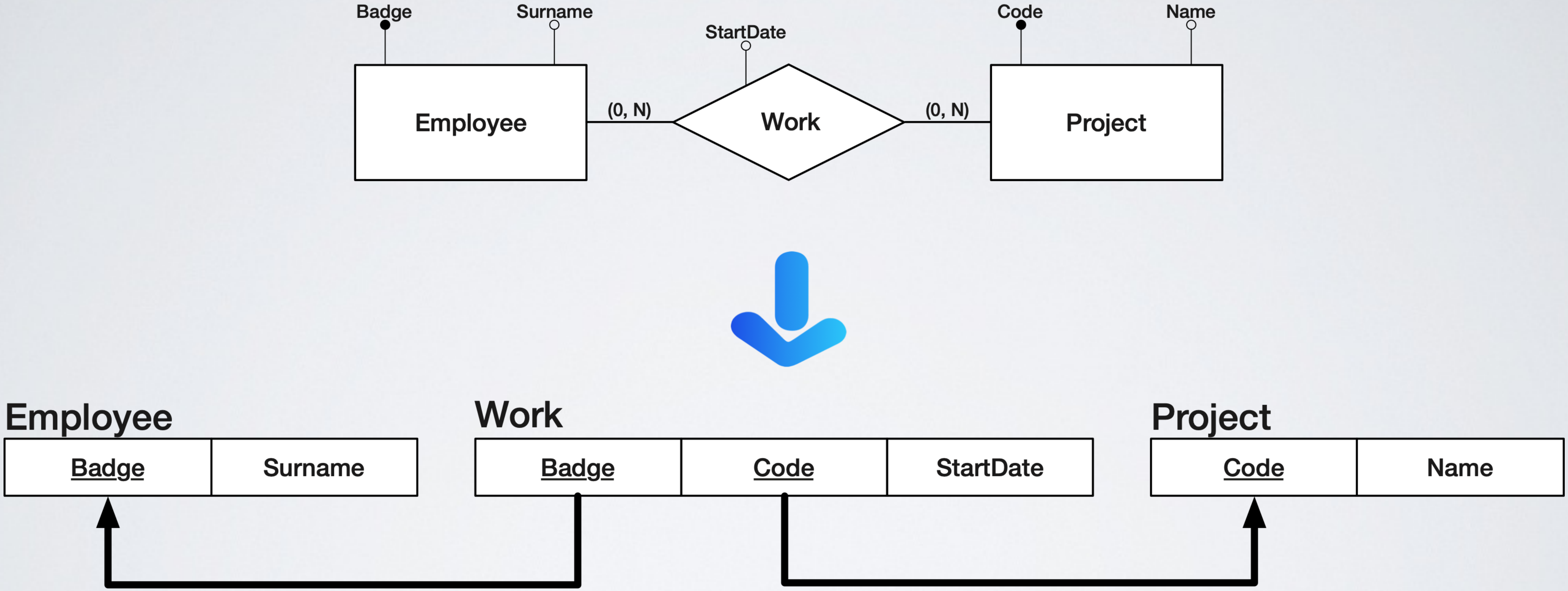
BadgeNumber	UnivName	Name	Surname

Mapping of Relationships: General Rules

- Each relationship **Q** of the ER schema which has not been merged in the previous step is mapped to a relation $R_{\mathbf{Q}}$

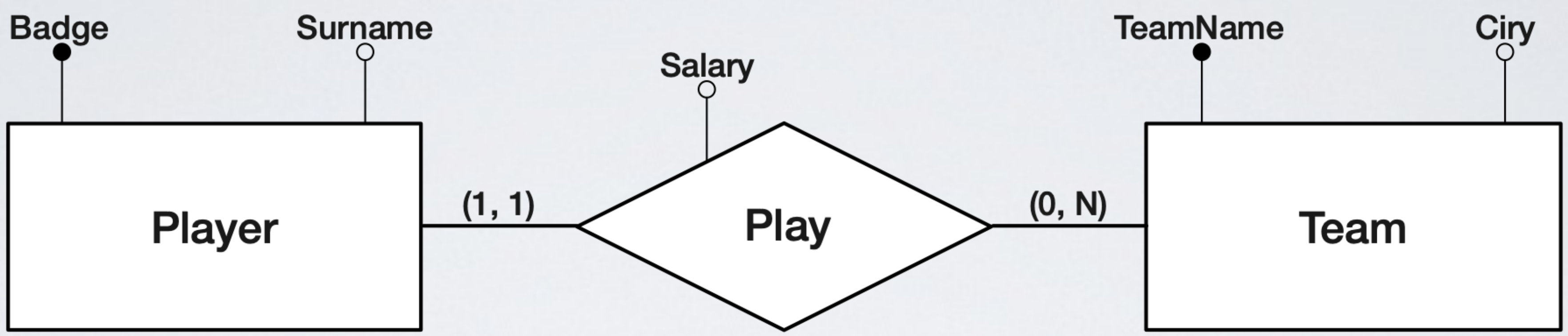
- The attributes of the relation $R_{\mathbf{Q}}$ are
  - all the attributes of the relationship **Q**
  - the identifier of the entities (primary keys of the relations) that participate to **Q**

- Choice of the primary key of $R_{\mathbf{Q}}$
  - If no entity participates to **Q** with maximum cardinality 1, then the primary key consists of all the primary keys of the participating entities
  - Otherwise, the primary key of each entity participating to **Q** with maximum cardinality 1 is a key and we can choose one of these candidate keys as the primary key



Badge
Surname
StartDate
Code
Name
Employee
(0, N)
Work
(0, N)
Project
Employee
Badge
Surname
Work
Badge
Code
StartDate
Project
Code
Name

Mapping of One-to-Many Relationships



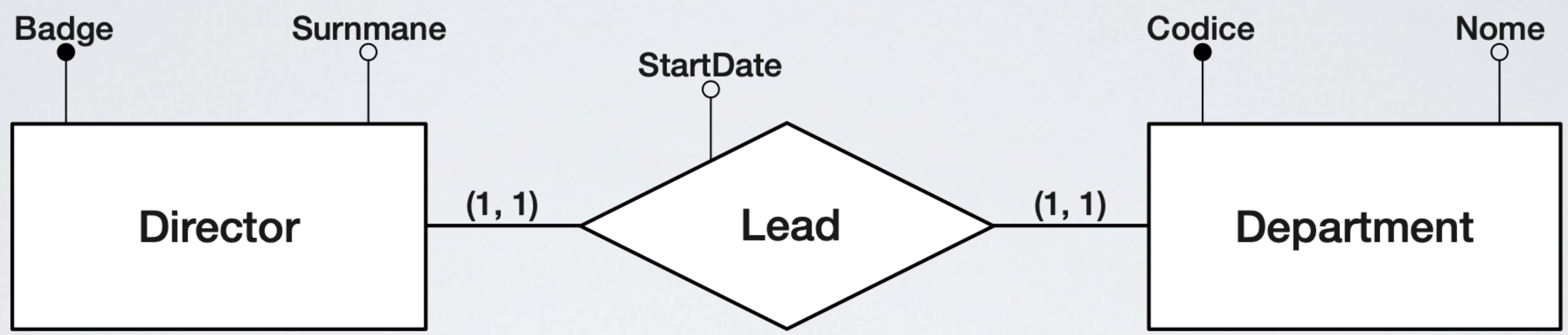
Player

<u>Badge</u>	Surname	TeamName	Salary

Team

<u>TeamName</u>	City

Mapping of One-to-One Relationships



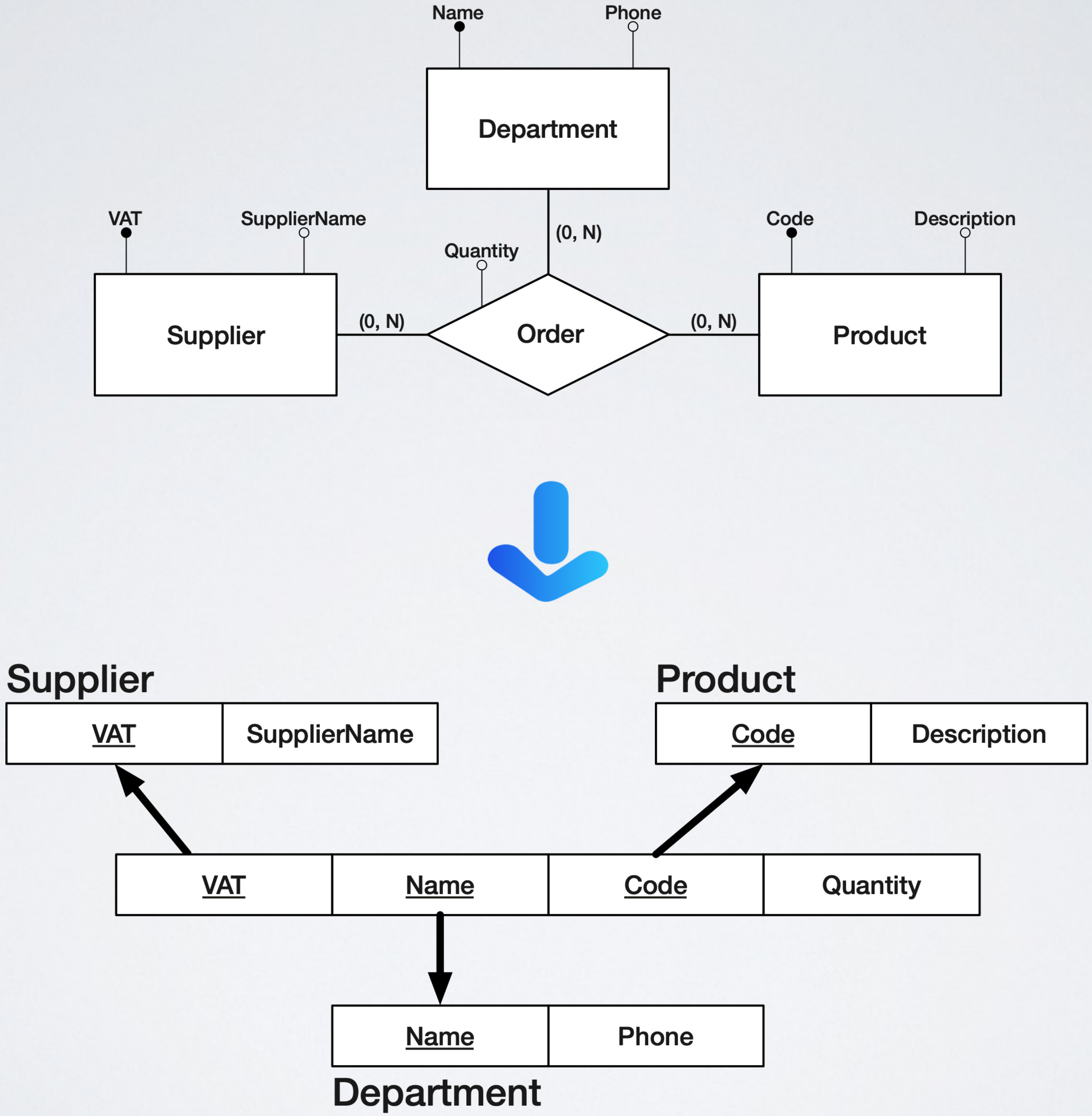
Director

<u>Badge</u>	Surname	Code	StartDate

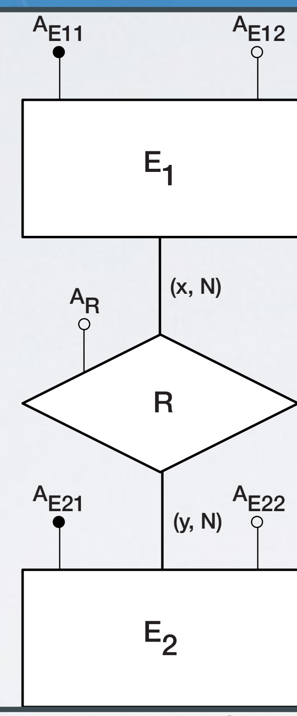
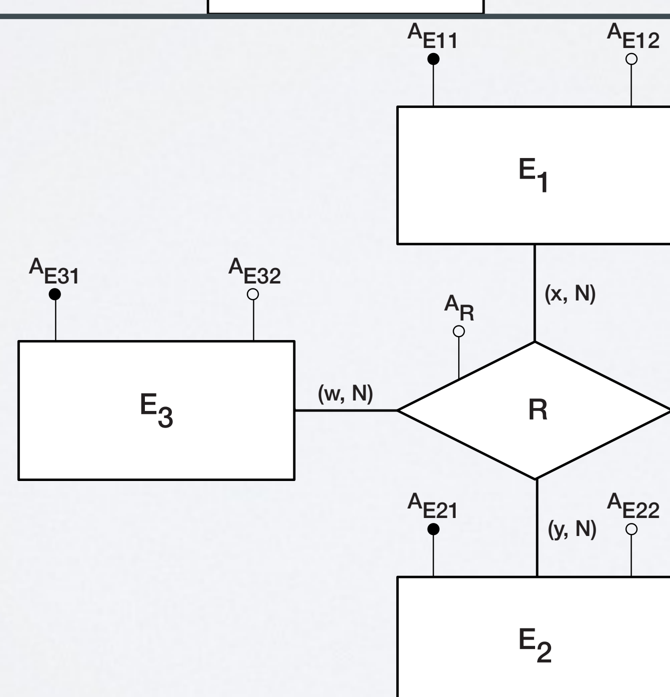
Department

<u>Code</u>	Name

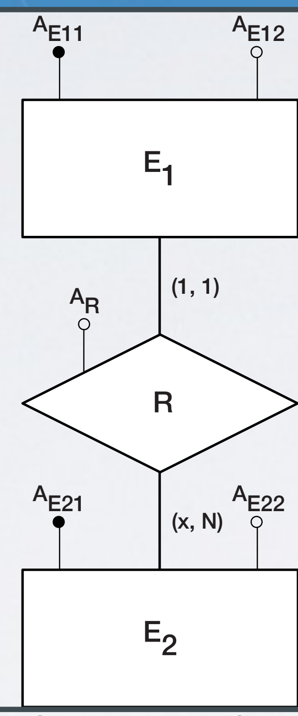
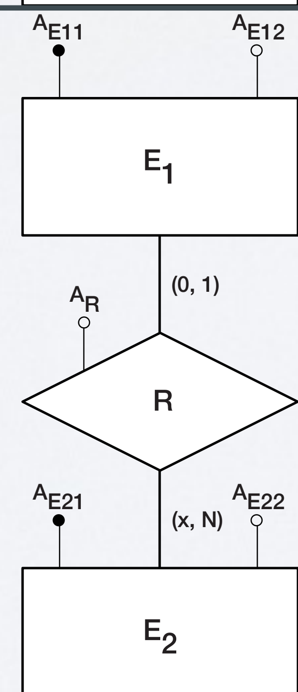
Mapping of Ternary Relationships



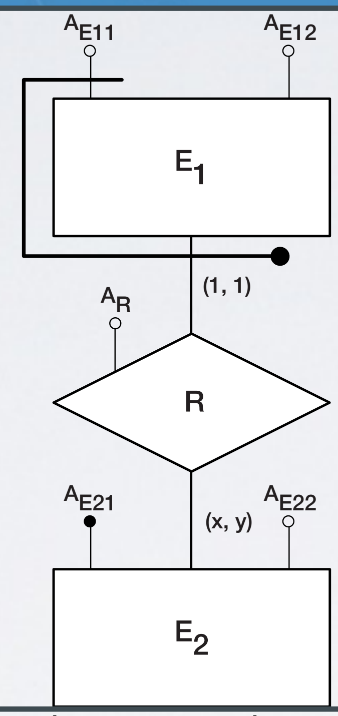
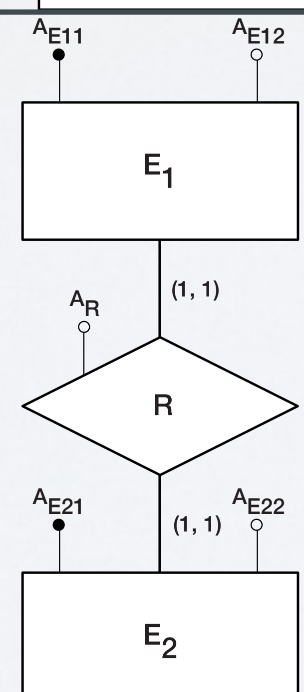
Mapping Summary (1/4)

Type	ER Model	Relational Model
Binary Many-to-Many Relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$
Ternary Relationship		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $E_3(\underline{A_{E31}}, A_{E32})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, \underline{A_{E31}}, A_R)$

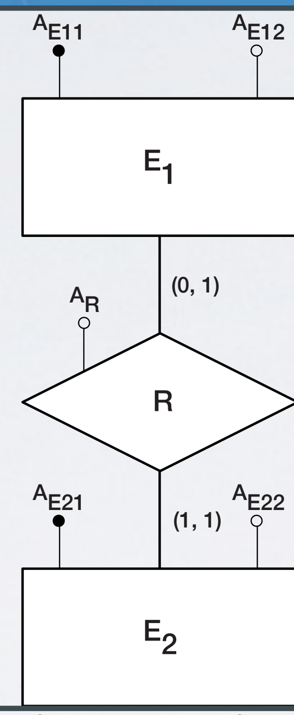
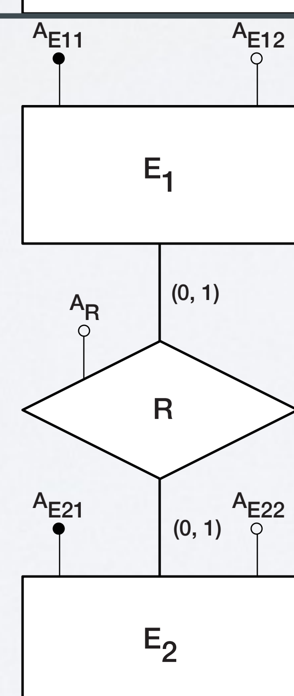
Mapping Summary (2/4)

Type	ER Model	Relational Model
One-to-Many Relationship with Mandatory Participation		$E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
One-to-Many Relationship with Optional Participation		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, A_{E21}, A_R)$ or $E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

Mapping Summary (3/4)

Type	ER Model	Relational Model
Entity with External Identifier		$E_1(\underline{A_{E11}}, \underline{A_{E21}}, A_{E12}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
One-to-One Relationship with Mandatory Participation for both Entities		$E_1(\underline{A_{E11}}, A_{E12}, \mathit{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ or $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22}, \mathit{A_{E11}}, A_R)$

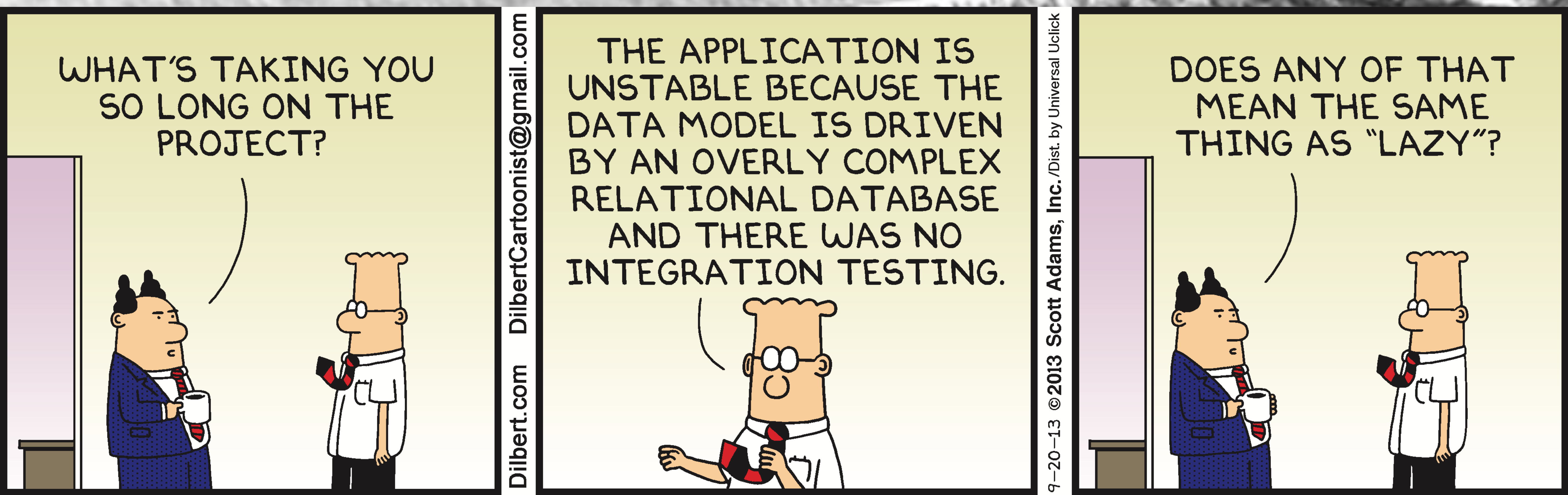
Mapping Summary (4/4)

Type	ER Model	Relational Model
One-to-One Relationship with Optional Participation for One Entity		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22}, \mathit{A_{E11}}, A_R)$
One-to-One Relationship with Optional Participation for both Entities		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22}, A_{E11}, A_R)$ or $E_1(\underline{A_{E11}}, A_{E12}, A_{E21}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ oppure $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \mathit{A_{E21}}, A_R)$

References

- Albano, A., Ghelli, G., and Orsini, R. (1997). ***Basi di dati relazionali e a oggetti***. Zanichelli, Bologna, Italia.
- Atzeni, P., Batini, C., and De Antonellis, V. (1985). ***La teoria relazionale dei dati***. Boringhieri, Torino, Italia.
- Codd, E. F. (1970). A Relational Model of Data for Large Shared Data Banks. ***Communications of the ACM***, 12(6):377–387.
- Codd, E. F. (1979). Extending the Database Relational Model to Capture More Meaning. ***ACM Transactions on Database Systems (TODS)***, 4(4):397–434.


Questions?


WHAT'S TAKING YOU SO LONG ON THE PROJECT?
Dilbert.com DilbertCartoonist@gmail.com
THE APPLICATION IS UNSTABLE BECAUSE THE DATA MODEL IS DRIVEN BY AN OVERLY COMPLEX RELATIONAL DATABASE AND THERE WAS NO INTEGRATION TESTING.
9-20-13 ©2013 Scott Adams, Inc./Dist. by Universal Uclick
DOES ANY OF THAT MEAN THE SAME THING AS "LAZY"?