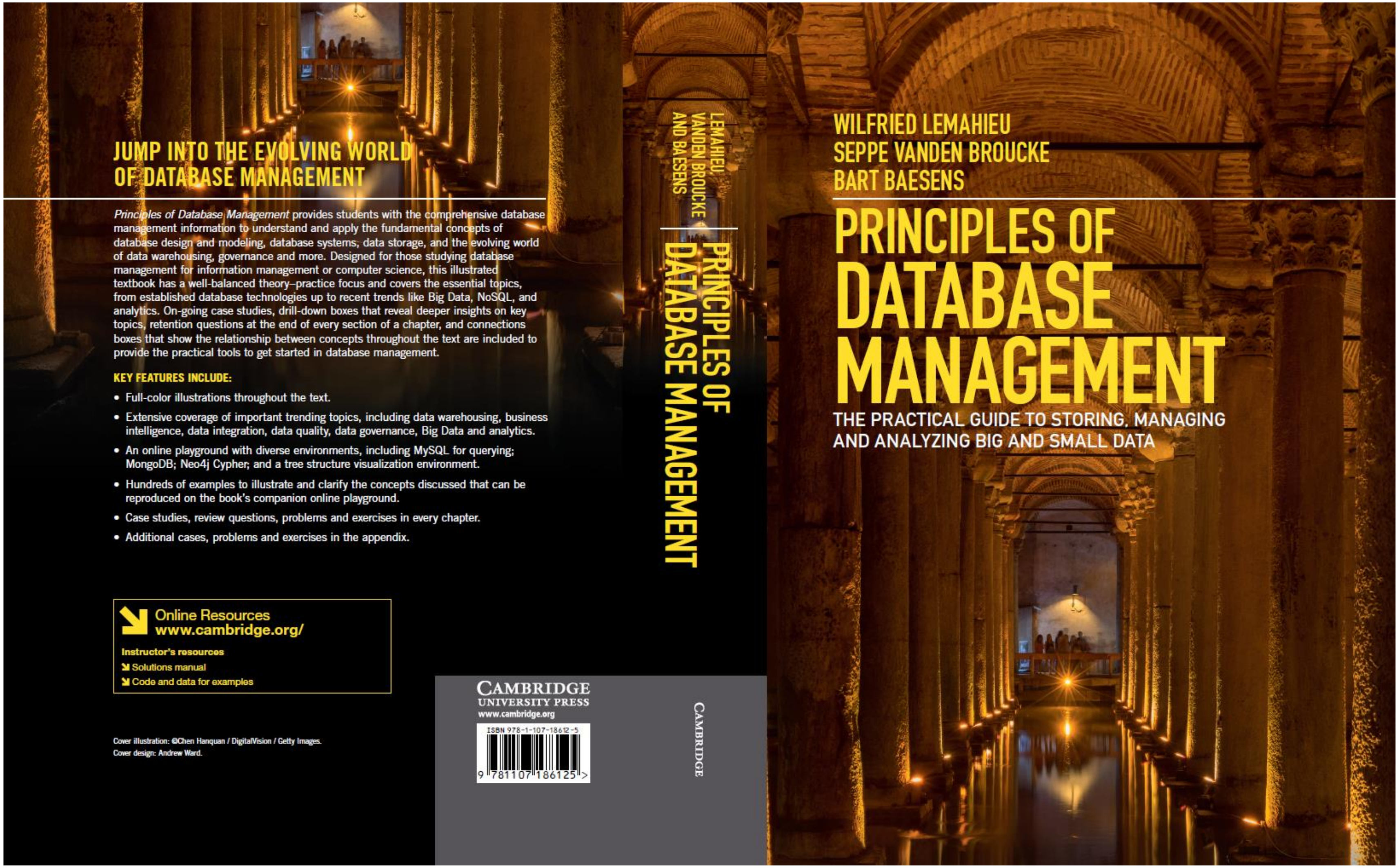
Relational Databases



www.pdbmbook.com

Introduction

- Relational Model
- Normalization
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a relational model

Relational Model

- Basic Concepts
- Formal Definitions
- Types of Keys
- Relational Constraints
- Example Relational Model

Basic Concepts

- Relational model was first formalized by Edgar F. Codd in 1970
- Relational model is a formal data model with a sound mathematical foundation, based on set theory and first order predicate logic
- No graphical representation
- Commonly adopted to build both logical and internal data models
- Microsoft SQL Server, IBM DB2 and Oracle

Basic Concepts

- A database is represented as a collection of relations
- A relation is defined as a set of tuples that each represent a similar real world entity
- A tuple is an ordered list of attribute values that each describe an aspect of an entity

Basic Concepts

SUPPLIER

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
37	Ad Fundum	82, Wacker Drive	Chicago	95
52	Spirits & co.	928, Strip	Las Vegas	NULL
68	The Wine Depot	132, Montgomery Street	San Francisco	10
69	Vinos del Mundo	4, Collins Avenue	Miami	92
...				

Basic Concepts

EER model	**Relational Model**
Entity type	Relation
Entity	Tuple
Attribute type	Column name

Basic Concepts

Student (Studentnr, Name, HomePhone, Address)

Professor (SSN, Name, HomePhone, OfficePhone, E-mail)

Course (CourseNo, CourseName)

Formal Definitions

- A domain specifies the range of admissible values for an attribute type
  - Example: gender domain, time domain
- Each attribute type is defined using a corresponding domain
- A domain can be used multiple times in a relation

BillOfMaterial

MAJORPRODNR	MINORPRODNR	QUANTITY
5	10	2
10	15	30

Formal Definitions

- A *relation* $R(A_1, A_2, A_3, \ldots A_n)$ can now be formally defined as a set of m tuples $r = \{t_1, t_2, t_3, \ldots t_m\}$ whereby each *tuple* t is an ordered list of n values $t = <v_1, v_2, v_3, \ldots v_n>$ corresponding to a particular entity
  - each value v_i is an element of the corresponding domain, $dom(A_i)$, or is a special NULL value
  - NULL value means that the value is missing, irrelevant or not applicable

Formal Definitions

```
Student(100, Michael Johnson, 123 456
789, 532 Seventh Avenue)

Professor(50, Bart Baesens, NULL, 876
543 210, Bart.Baesens@kuleuven.be)

Course(10, Principles of Database
Management)
```

Formal Definitions

- A relation essentially represents a set (no ordering + no duplicates!)
- The domain constraint states that the value of each attribute type A must be an atomic and single value from the domain dom(A)
- Example: `COURSE(coursenr, coursename, study points)`

```
(10, Principles of Database Management, 6)

(10, {Principles of Database Management, Database
Modeling}, 6)  ➔  WRONG!
```

Formal Definitions

- A relation R of degree n on the domains dom(A_1), dom(A_2), dom(A_3), … , dom(A_n) can also be alternatively defined as a subset of the Cartesian product of the domains that define each of the attribute types

Domain Product ID
001
002
003
…

X

Domain Product Color
Blue
Red
Black

X

Domain Product Category
A
B
C

ProductID	Product Color	Product Category
001	Blue	A
001	Blue	B
001	Blue	C
001	Red	A
001	Red	B
001	Red	C
…		

Types of Keys

- Superkeys and Keys
- Candidate Keys, Primary Keys, Alternative Keys
- Foreign Keys

Superkeys and Keys

- A superkey is defined as a subset of attribute types of a relation R with the property that no two tuples in any relation state should have the same combination of values for these attribute types
- A superkey specifies a uniqueness constraint
- A superkey can have redundant attribute types
- Example: `(Studentnr, Name, HomePhone)`

Superkeys and Keys

- A key K of a relation scheme R is a superkey of R with the additional property that removing any attribute type from K leaves a set of attribute types that is no superkey of R
- A key does not have any redundant attribute types (minimal superkey)
- Example: `Studentnr`
- The key constraint states that every relation must have at least 1 key that allows to uniquely identify its tuples

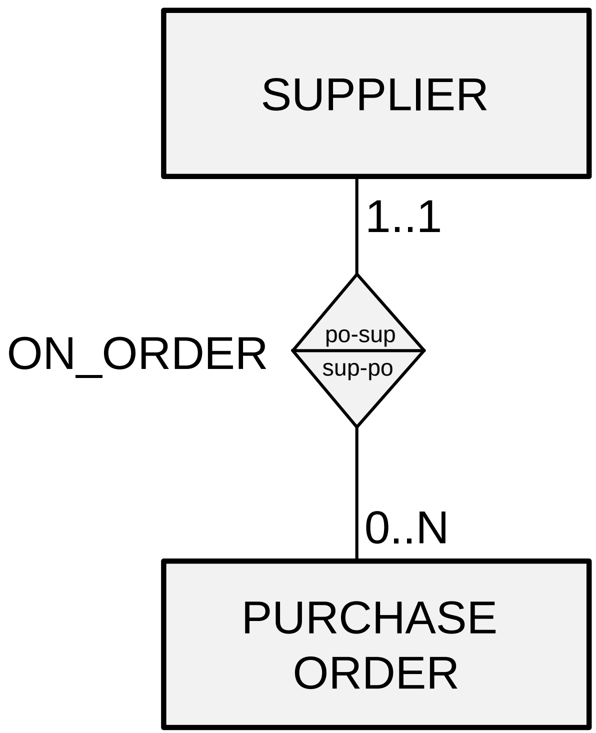
Candidate Keys, Primary Keys and Alternative Keys

- A relation may have more than one key (candidate keys)
  - PRODUCT: `product number` and `product name`
- Primary key is used to identify tuples in the relation, to establish connections to other relations and for storage purposes
  - Entity integrity constraint: attribute types that make up the primary key should always satisfy a NOT NULL constraint
- Other candidate keys are then referred to as alternative keys

Foreign Keys

- A set of attribute types FK in a relation R_1 is a foreign key of R_1 if two conditions are satisfied (referential integrity constraint)
  - the attribute types in FK have the same domains as the primary key attribute types PK of a relation R_2
  - a value FK in a tuple t_1 of the current state r_1 either occurs as a value of PK for some tuple t_2 in the current state r_2 or is NULL

Foreign Keys



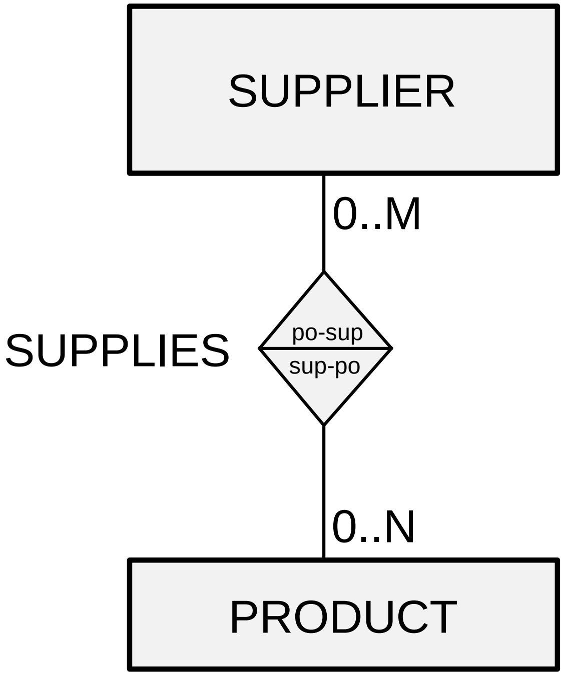
SUPPLIER

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
…				
37	Ad Fundum	82, Wacker Drive	Chicago	95
94	The Wine Crate	330, McKinney Avenue	Dallas	75
…				

PURCHASE_ORDER

PONR	PODATE	*SUPNR*
1511	2015-03-24	37
1512	2015-04-10	94
…		

Foreign Keys



SUPPLIER

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
…				

PRODUCT

PRODNR	PRODNAME	PRODTYPE	AVAILABLE_QUANTITY
0119	Chateau Miraval, Cotes de Provence Rose, 2015	rose	126
0154	Chateau Haut Brion, 2008	red	111
…		red	5

Foreign Keys

SUPPLIES

<u>SUPNR</u>	***<u>PRODNR</u>***	**PURCHASE_PRICE**	**DELIV_PERIOD**
…			
68	0327	56.99	4
…			
21	0289	17.99	1
21	0327	56.00	6
21	0347	16.00	2
…			
69	0347	18.00	4
84	0347	18.00	4
…			

Relational Constraints

Domain constraint	The value of each attribute type A must be an atomic and single value from the domain dom(A).
Key constraint	Every relation has a key that allows to uniquely identify its tuples.
Entity integrity constraint	The attribute types that make up the primary key should always satisfy a NOT NULL constraint.
Referential integrity constraint	A foreign key FK has the same domain as the primary key PK attribute type(s) it refers to and either occurs as a value of PK or NULL.

Example Relational Data Model

SUPPLIER(<u>SUPNR</u>, SUPNAME, SUPADDRESS, SUPCITY, SUPSTATUS)

PRODUCT(<u>PRODNR</u>, PRODNAME, PRODTYPE, AVAILABLE QUANTITY)

SUPPLIES(<u>*SUPNR*</u>, <u>*PRODNR*</u>, PURCHASE_PRICE, DELIV_PERIOD)

PURCHASE_ORDER(<u>PONR</u>, PODATE, *SUPNR*)

PO_LINE(<u>*PONR*</u>, <u>*PRODNR*</u>, QUANTITY)

Example Relational Data Model

Supplier

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
...				

Product

PRODNR	PRODNAME	PRODTYPE	AVAILABLE_QUANTITY
0119	Chateau Miraval, Cotes de Provence Rose, 2015	rose	126
0384	Dominio de Pingus, Ribera del Duero, Tempranillo, 2006	red	38
...			

Supplies

SUPNR	PRODNR	PURCHASE_PRICE	DELIV_PERIOD
21	0119	15.99	1
21	0384	55.00	2
...			

Purchase_Order

PONR	PODATE	SUPNR
1511	2015-03-24	37
1512	2015-04-10	94
...		

PO_Line

PONR	PRODNR	QUANTITY
1511	0212	2
1511	0345	4
...		

Normalization

- Insertion, Deletion and Update Anomalies
- Informal Normalization guidelines
- Functional Dependencies and Prime Attribute Type
- Normalization forms

Insertion, Deletion and Update Anomalies

SUPPLIES

<u>SUPNR</u>	***<u>PRODNR</u>***	**PURCHASE_PRICE**	**DELIV_PERIOD**	***SUPNAME***	***SUPADDRESS***	**…**	***PRODNAME***	***PRODTYPE***	**…**
21	0289	17.99	1	*Deliwines*	*240, Avenue of the Americas*		*Chateau Saint Estève de Neri, 2015*	*Rose*	
21	0327	56.00	6	*Deliwines*	*240, Avenue of the Americas*		*Chateau La Croix Saint-Michel, 2011*	*Red*	
…									

PO_LINE

<u>PONR</u>	***<u>PRODNR</u>***	**QUANTITY**	***PODATE***	***SUPNR***
1511	0212	2	*2015-03-24*	*37*
1511	0345	4	*2015-03-24*	*37*
…				

Insertion, Deletion and Update Anomalies

Supplier

SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
21	Deliwines	240, Avenue of the Americas	New York	20
32	Best Wines	660, Market Street	San Francisco	90
...				

Product

PRODNR	PRODNAME	PRODTYPE	AVAILABLE_QUANTITY
0119	Chateau Miraval, Cotes de Provence Rose, 2015	rose	126
0384	Dominio de Pingus, Ribera del Duero, Tempranillo, 2006	red	38
...			

Supplies

SUPNR	PRODNR	PURCHASE_PRICE	DELIV_PERIOD
21	0119	15.99	1
21	0384	55.00	2
...			

Purchase_Order

PONR	PODATE	SUPNR
1511	2015-03-24	37
1512	2015-04-10	94
...		

PO_Line

PONR	PRODNR	QUANTITY
1511	0212	2
1511	0345	4
...		

Insertion, Deletion and Update Anomalies

- To have a good relational data model, all relations in the model should be normalized
- A formal normalization procedure can be applied to transform an unnormalized relational model into a normalized form.
- The advantages are twofold:
  - At the logical level, the users can easily understand the meaning of the data and formulate correct queries
  - At the implementation level, the storage space is used efficiently and the risk of inconsistent updates is reduced

Informal Normalization Guidelines

- Design a relational model in such a way that it is easy to explain its meaning
  - `MYRELATION123(SUPNR, SUPNAME, SUPTWITTER, PRODNR, PRODNAME, …)` versus `SUPPLIER(SUPNR, SUPNAME, SUPTWITTER, PRODNR, PRODNAME, ……)`
- Attribute types from multiple entity types should not be combined in a single relation
- Avoid excessive amount of NULL values in a relation
  - `SUPPLIER(SUPNR, SUPNAME, …)`
  - `SUPPLIER-TWITTER(SUPNR, SUPTWITTER)`

Functional Dependencies and Prime Attribute Type

- A functional dependency $X \rightarrow Y$, between two sets of attribute types X and Y implies that a value of X uniquely determines a value of Y
  - there is a functional dependency from X to Y or Y is functionally dependent on X
- Examples:
  - `SSN` $\rightarrow$ `ENAME`
  - `PNUMBER` $\rightarrow$ `{PNAME, PLOCATION}`
  - `{SSN, PNUMBER}` $\rightarrow$ `HOURS`

Functional Dependencies and Prime Attribute Type

- A prime attribute type is an attribute type that is part of a candidate key
- Example: `R1(SSN, PNUMBER, PNAME, HOURS)`
  - Prime attribute types: `SSN` and `PNUMBER`
  - Non-prime attribute types: `PNAME` and `HOURS`

Normalization Forms

- First Normal Form (1 NF)
- Second Normal Form (2 NF)
- Third Normal Form (3 NF)
- Boyce-Codd Normal Form (BCNF)
- Fourth Normal Form (4 NF)

First Normal Form (1 NF)

- The first normal form (1 NF) states that every attribute type of a relation must be atomic and single valued
  - no composite or multivalued attribute types (domain constraint!)
- `SUPPLIER(`<u>`SUPNR`</u>`, NAME(FIRST NAME, LAST NAME), SUPSTATUS)`
- `SUPPLIER(`<u>`SUPNR`</u>`, FIRST NAME, LAST NAME, SUPSTATUS)`

First Normal Form (1 NF)

- DEPARTMENT(DNUMBER, DLOCATION, *DMGRSSN*)
  - Assumption: a department can have multiple locations and multiple departments are possible at a given location
- DEPARTMENT(DNUMBER, DMGRSSN)
- DEP-LOCATION(*DNUMBER*, DLOCATION)

First Normal Form (1 NF)

DNUMBER	DLOCATION	DMGRSSN
15	{New York, San Francisco}	110
20	Chicago	150
30	{Chicago, Boston}	100

1 NF

DEPARTMENT

DNUMBER	***DMGRSSN***
15	110
20	150
30	100

DEP-LOCATION

DNUMBER	**DLOCATION**
15	New York
15	San Francisco
20	Chicago
30	Chicago
30	Boston

First Normal Form (1 NF)

- `R1(`<u>`SSN`</u>`, ENAME, DNUMBER, DNAME, PROJECT(`<u>`PNUMBER`</u>`, PNAME, HOURS))`
  - assume an employee can work on multiple projects and multiple employees can work on the same project
- `R11(`<u>`SSN`</u>`, ENAME, DNUMBER, DNAME)`
- `R12(`<u>*`SSN`*</u>`, `<u>`PNUMBER`</u>`, PNAME, HOURS)`

Second Normal Form (2 NF)

- A functional dependency $X \rightarrow Y$ is a full functional dependency if removal of any attribute type A from X means that the dependency does not hold anymore
  - Examples: `SSN, PNUMBER` $\rightarrow$ `HOURS`; `PNUMBER` $\rightarrow$ `PNAME`
- A functional dependency $X \rightarrow Y$ is a partial dependency if an attribute type A from X can be removed from X and the dependency still holds
  - Example: `SSN, PNUMBER` $\rightarrow$ `PNAME`

Second Normal Form (2 NF)

- A relation R is in the second normal form (2 NF) if it satisfies 1 NF and every non-prime attribute type A in R is fully functional dependent on any key of R
- If the relation is not in second normal form, we must:
  - decompose it and set up a new relation for each partial key together with its dependent attribute types
  - keep a relation with the original primary key and any attribute types that are fully functional dependent on it

Second Normal Form (2 NF)

- `R1(SSN, PNUMBER, PNAME, HOURS)`
  - assume an employee can work on multiple projects; multiple employees can work on the same project and a project has a unique name
- R11(<u>SSN</u>, <u>*PNUMBER*</u>, HOURS)
- R12(<u>PNUMBER</u>, PNAME)

Second Normal Form (2 NF)

SSN	PNUMBER	PNAME	HOURS
100	1000	Hadoop	50
220	1200	CRM	200
280	1000	Hadoop	40
300	1500	Java	100
120	1000	Hadoop	120

2 NF

PNUMBER	PNAME
1000	Hadoop
1200	CRM
1500	Java

SSN	PNUMBER	HOURS
100	1000	50
220	1200	200
280	1000	40
300	1500	100
120	1000	120

Third Normal Form (3 NF)

- A functional dependency $X \rightarrow Y$ in a relation R is a transitive dependency if there is a set of attribute types Z that is neither a candidate key nor a subset of any key of R, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold
- A relation is in the third normal form (3 NF) if it satisfies 2 NF and no non-prime attribute type of R is transitively dependent on the primary key
- If the relation is not in third normal form, we need to decompose the relation R and set up a relation that includes the non-key attribute types that functionally determine the other non-key attribute types

Third Normal Form (3 NF)

- R1(<u>SSN</u>, ENAME, DNUMBER, DNAME, DMGRSSN)
  - Assume an employee works in one department, a department can have multiple employees and a department has one manager
- R11(<u>SSN</u>, ENAME, *DNUMBER*)
- R12(<u>DNUMBER</u>, DNAME, *DMGRSSN*)

Third Normal Form (3 NF)

<u>SSN</u>	NAME	DNUMBER	DNAME	*DMGRSSN*
10	O'Reilly	10	Marketing	210
22	Donovan	30	Logistics	150
28	Bush	10	Marketing	210
30	Jackson	20	Finance	180
12	Thompson	10	Marketing	210

3 NF

<u>SSNR</u>	NAME	*DNUMBER*
10	O'Reilly	10
22	Donovan	30
28	Bush	10
30	Jackson	20
12	Thompson	10

<u>DNUMBER</u>	DNAME	*DMGRSSN*
10	Marketing	210
30	Logistics	150
20	Finance	180

Boyce-Codd Normal Form (BCNF)

- A functional dependency X → Y is called a trivial functional dependency if Y is a subset of X
  - Example: SSN, NAME → SSN
- A relation R is in the Boyce-Codd normal form (BCNF) provided each of its non-trivial functional dependencies X → Y, X is a superkey—that is, X is either a candidate key or a superset thereof
- BCNF normal form is stricter than the third normal form
  - Every relation in BCNF is also in 3 NF (not vice versa)

Boyce-Codd Normal Form (BCNF)

- `R1(SUPNR, SUPNAME, PRODNR, QUANTITY)`
  - Assume a supplier can supply multiple products; a product can be supplied by multiple suppliers and a supplier has a unique name
- R11(<u>*SUPNR*</u>, <u>PRODNR</u>, QUANTITY)
- R12(<u>SUPNR</u>, SUPNAME)

Fourth Normal Form (4 NF)

- There is a multivalued dependency from X to Y, X $\rightarrow\rightarrow$ Y, if and only if each X value exactly determines a set of Y values, independently of the other attribute types
- A relation is in the fourth normal form (4 NF) if it is in Boyce-Codd normal form and for every one of its non-trivial multivalued dependencies X $\rightarrow\rightarrow$ Y, X is a superkey—that is, X is either a candidate key or a superset thereof

Fourth Normal Form (4 NF)

- `R1(course, instructor, textbook)`
  - Assume a course can be taught by different instructors, and a course uses the same set of textbooks for each instructor
- R11(<u>course</u>, <u>textbook</u>)
- R12(<u>course</u>, <u>instructor</u>)

Fourth Normal Form (4 NF)

COURSE	**INSTRUCTOR**	**BOOK**
Database Management	Baesens	Database cookbook
Database Management	Lemahieu	Database cookbook
Database Management	Baesens	Databases for dummies
Database Management	Lemahieu	Databases for dummies

4 NF

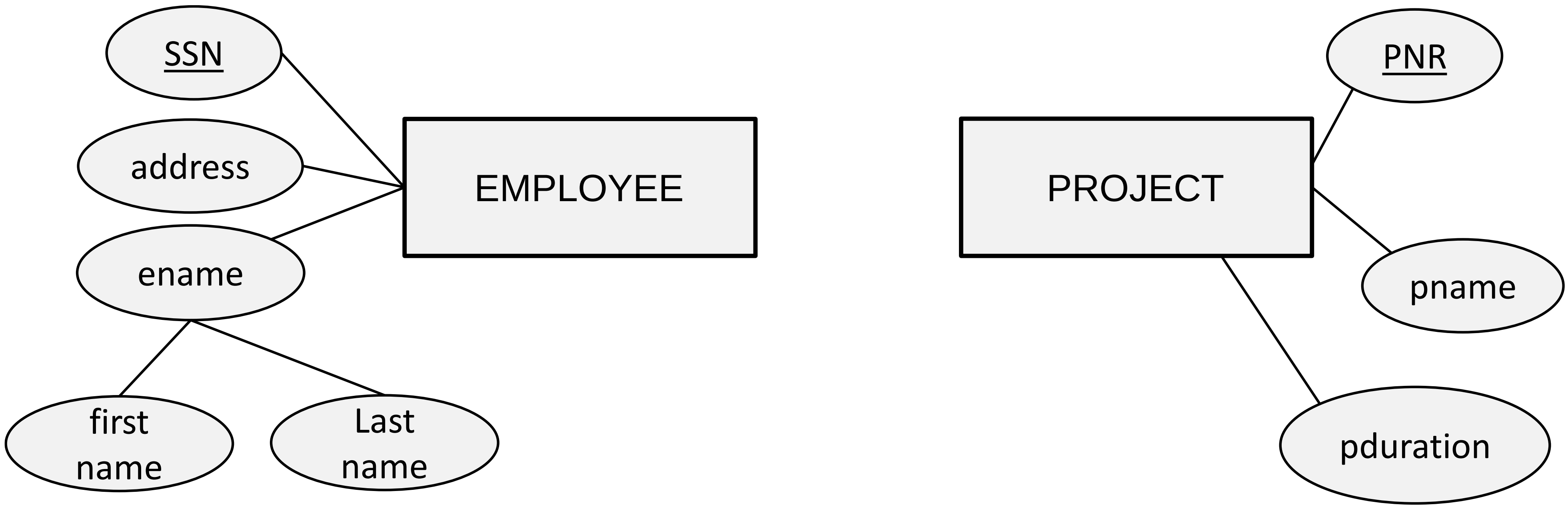
COURSE	**INSTRUCTOR**
Database Management	Baesens
Database Management	Lemahieu

COURSE	**BOOK**
Database Management	Database cookbook
Database Management	Databases for dummies

Mapping a Conceptual ER Model to a Relational Model

- Mapping Entity Types
- Mapping Relationship Types
- Mapping Multivalued Attribute Types
- Mapping Weak Entity Types
- Putting it All Together

Mapping Entity Types



EMPLOYEE(<u>SSN</u>, address, first name, last name)

PROJECT(<u>PNR</u>, pname, pduration)

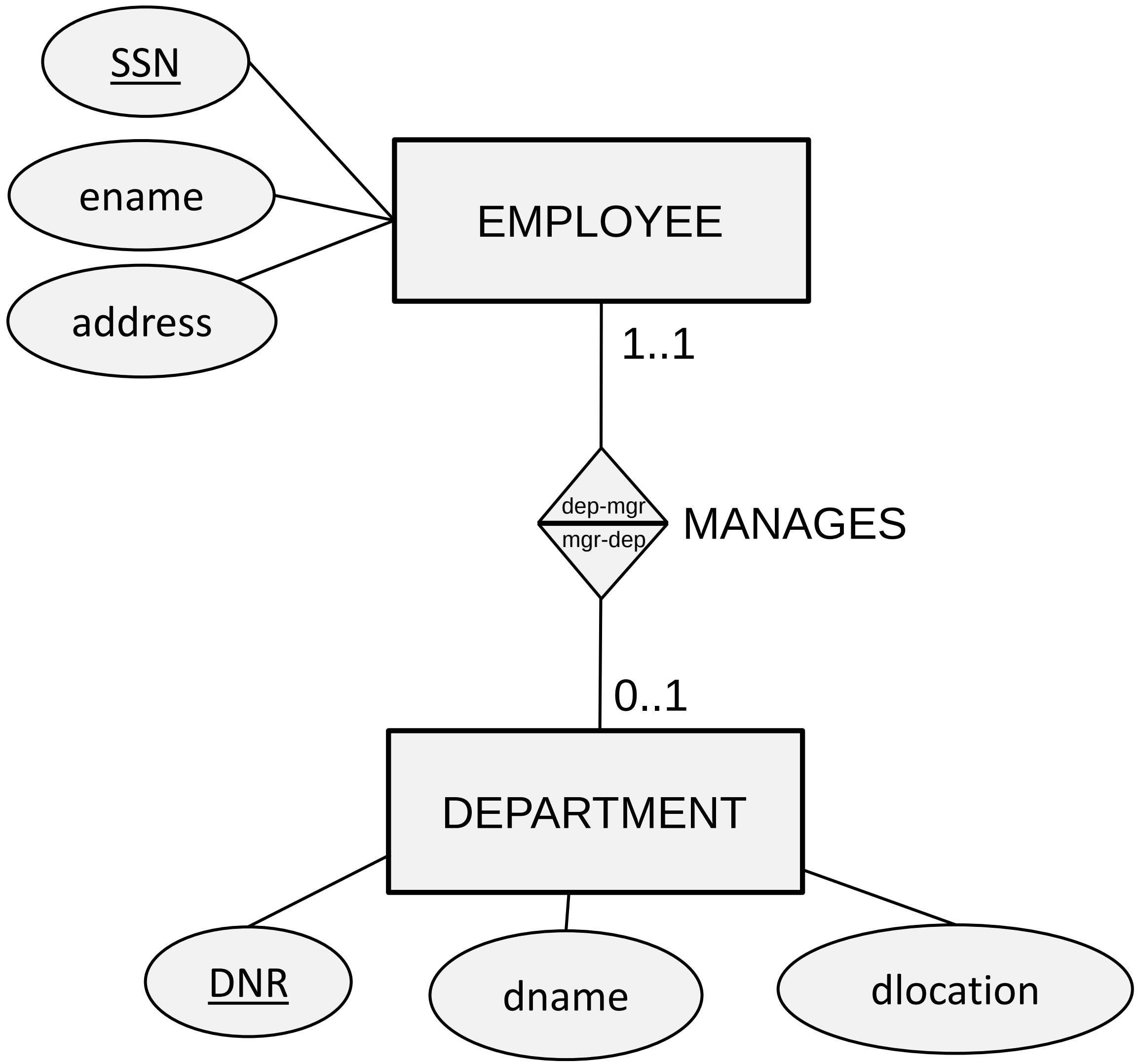
Mapping Relationship Types

- Mapping a binary 1:1 relationship type
- Mapping a binary 1:N relationship type
- Mapping a binary M:N relationship type
- Mapping unary relationship types
- Mapping n-ary relationship types

Mapping a Binary 1:1 Relationship Type

- Create two relations: one for each entity type participating in the relationship type
- The connection can be made by including a foreign key in one of the relations to the primary key of the other
- In case of existence dependency, put the foreign key in the existent dependent relation and declare it as NOT NULL
- The attribute types of the 1:1 relationship type can then be added to the relation with the foreign key

Mapping a Binary 1:1 Relationship Type



Mapping a Binary 1:1 Relationship Type

EMPLOYEE(SSN, ename, address, *DNR*)

DEPARTMENT(DNR, dname, dlocation)

EMPLOYEE(SSN, ename, address, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	003
356	Emma Lucas	432 Wacker Drive, Chicago	NULL
412	Michael Johnson	1134 Pennsylvania Avenue, Washington	NULL
564	Sarah Adams	812 Collins Avenue, Miami	001

DEPARTMENT(DNR, dname, dlocation)

001	Marketing	3th floor
002	Call center	2nd floor
003	Finance	basement
004	ICT	1st floor

Mapping a Binary 1:1 Relationship Type

EMPLOYEE(<u>SSN</u>, ename, address)

DEPARTMENT(<u>DNR</u>, dname, dlocation, *SSN*)

EMPLOYEE(<u>SSN</u>, ename, address)

511	John Smith	14 Avenue of the Americas, New York
289	Paul Barker	208 Market Street, San Francisco
356	Emma Lucas	432 Wacker Drive, Chicago

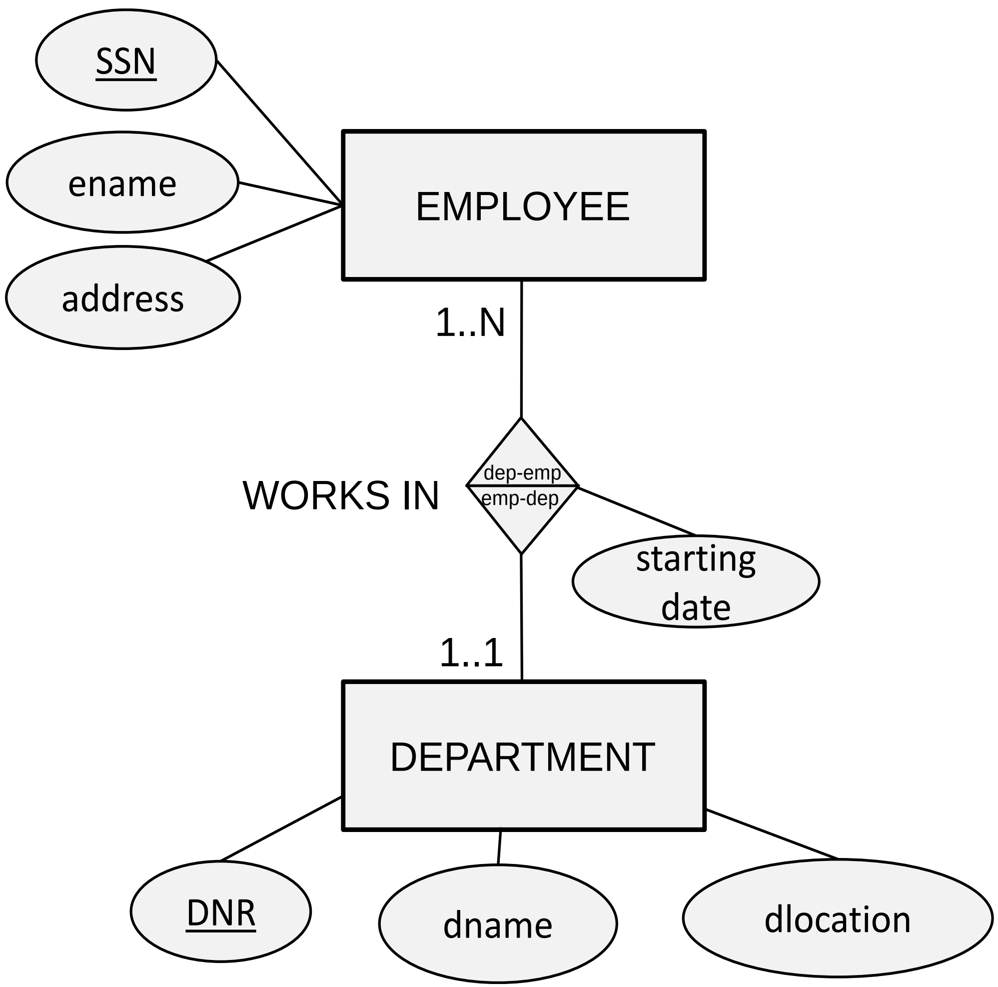
DEPARTMENT(<u>DNR</u>, dname, dlocation, *SSN*)

001	Marketing	3th floor	511
002	Call center	2nd floor	511
003	Finance	basement	289
004	ICT	1st floor	511

Mapping a Binary 1:N Relationship Type

- Binary 1:N relationship types can be mapped by including a foreign key in the relation corresponding to the participating entity type at the N-side of the relationship type
- The foreign key refers to the primary key of the relation corresponding to the entity type at the 1-side of the relationship type
- Depending upon the minimum cardinality, the foreign key can be declared as NOT NULL or NULL ALLOWED
- The attribute types of the 1:N relationship type can be added to the relation corresponding to the participating entity type

Mapping a Binary 1:N Relationship Type



EMPLOYEE(<u>SSN</u>, ename, address, starting date, *DNR*)

DEPARTMENT(<u>DNR</u>, dname, dlocation)

Mapping a Binary 1:N Relationship Type

EMPLOYEE(<u>SSN</u>, ename, address, starting date, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	01/01/2000	001
289	Paul Barker	208 Market Street, San Francisco	01/01/1998	001
356	Emma Lucas	432 Wacker Drive, Chicago	01/01/2010	002

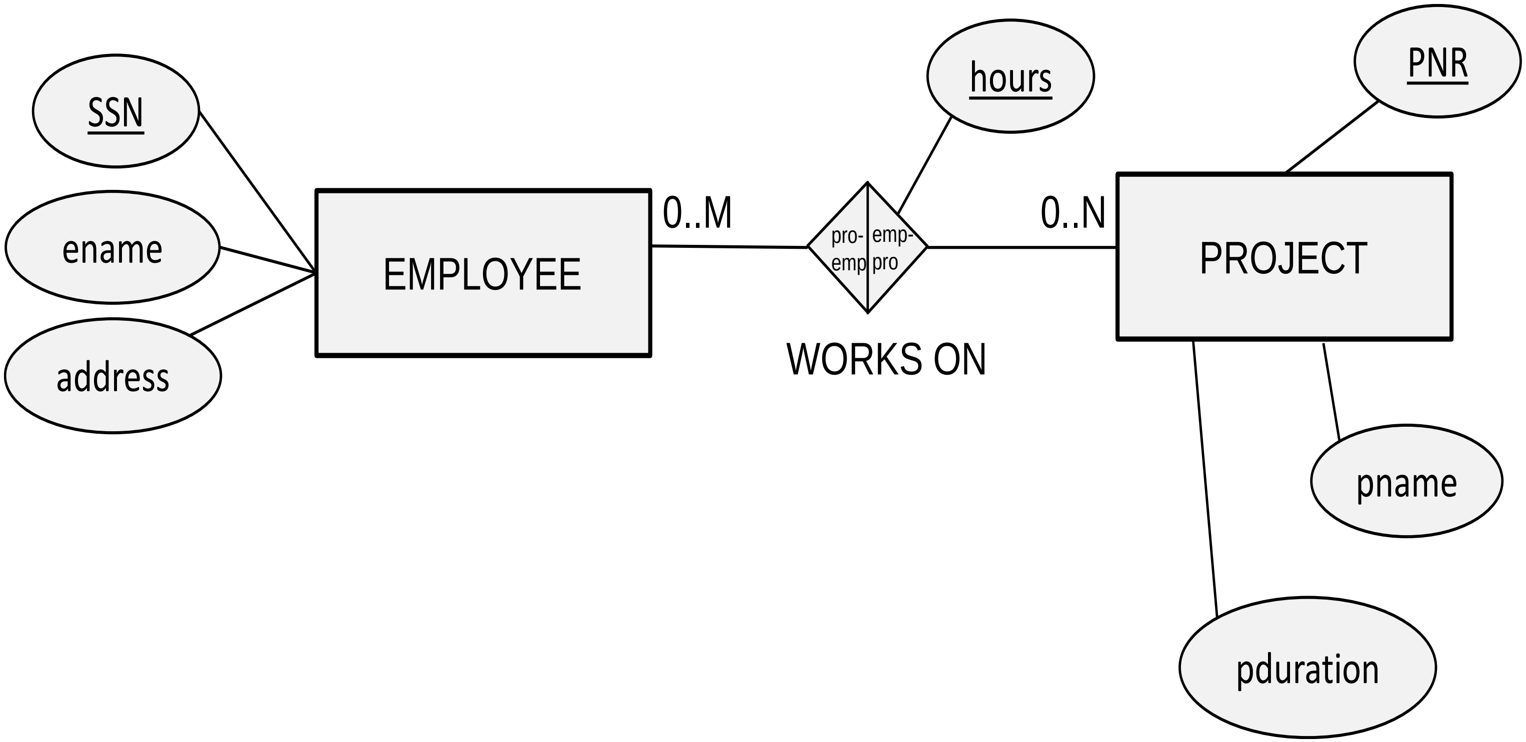
DEPARTMENT(<u>DNR</u>, dname, dlocation)

001	Marketing	3th floor
002	Call center	2nd floor
003	Finance	basement
004	ICT	1st floor

Mapping a Binary N:M Relationship Type

- M:N relationship types are mapped by introducing a new relation R
- The primary key of R is a combination of foreign keys referring to the primary keys of the relations corresponding to the participating entity types
- The attribute types of the M:N relationship type can also be added to R

Mapping a Binary M:N Relationship Type



EMPLOYEE(<u>SSN</u>, ename, address)

PROJECT(<u>PNR</u>, pname, pduration)

WORKS_ON(<u>*SSN*</u>, <u>*PNR*</u>, hours)

Mapping a Binary M:N Relationship Type

EMPLOYEE(SSN, ename, address, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	001
356	Emma Lucas	432 Wacker Drive, Chicago	002

PROJECT(PNR, pname, pduration)

1001	B2B	100
1002	Analytics	660
1003	Web site	52
1004	Hadoop	826

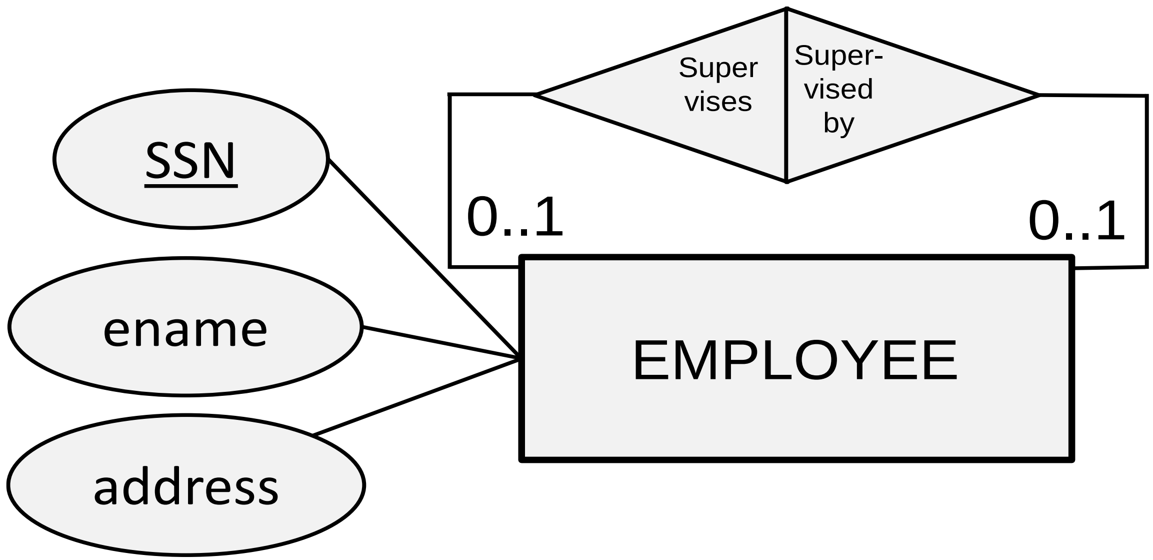
WORKS_ON(*SSN*, *PNR*, hours)

511	1001	10
289	1001	80
289	1003	50

Mapping Unary Relationship Types

- A recursive 1:1 or 1:N relationship type can be implemented by adding a foreign key referring to the primary key of the same relation
- For a N:M recursive relationship type, a new relation R needs to be created with two NOT NULL foreign keys referring to the original relation

Mapping Unary Relationship Types



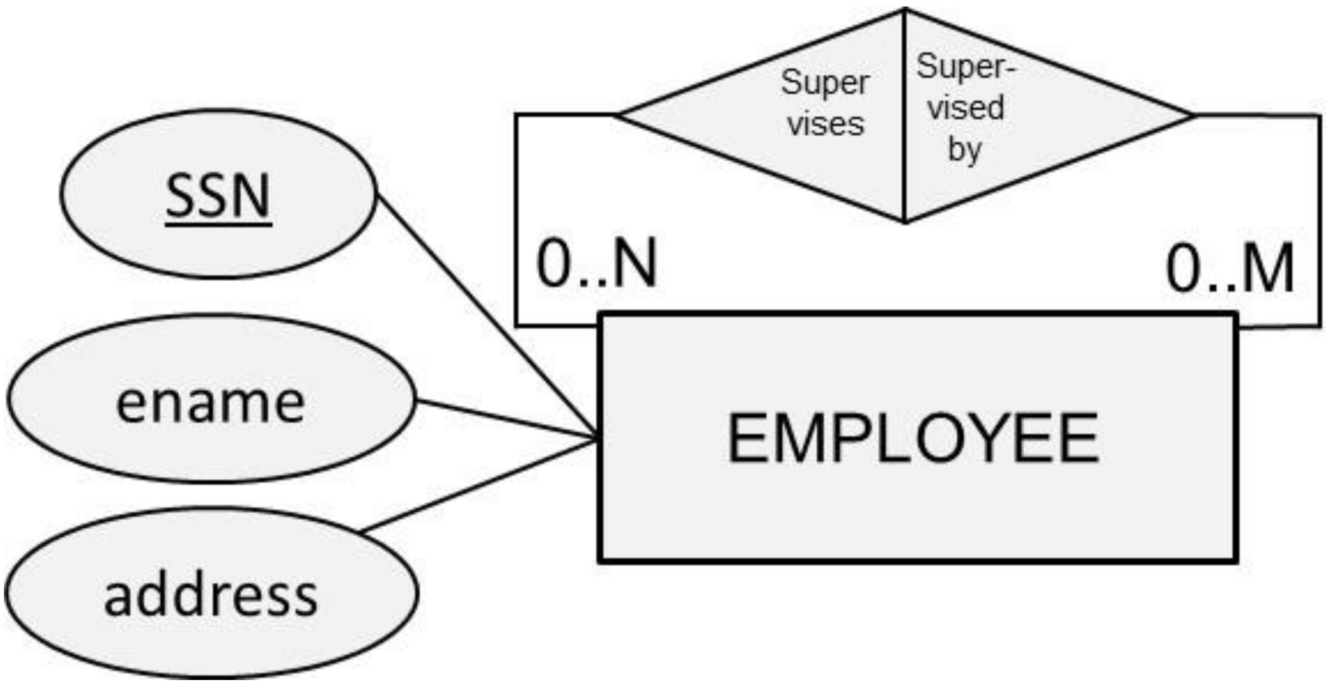
EMPLOYEE(<u>SSN</u>, ename, address, *supervisor*)

Mapping Unary Relationship Types

EMPLOYEE(<u>SSN</u>, ename, address, *supervisor*)

511	John Smith	14 Avenue of the Americas, New York	289
289	Paul Barker	208 Market Street, San Francisco	412
356	Emma Lucas	432 Wacker Drive, Chicago	289
412	Dan Kelly	668 Strip, Las Vegas	NULL

Mapping Unary Relationship Types



EMPLOYEE(<u>SSN</u>, ename, address)

SUPERVISION(<u>*Supervisor*</u>, <u>*Supervisee*</u>)

Mapping Unary Relationship Types

EMPLOYEE(SSN, ename, address)

SUPERVISION(*Supervisor, Supervisee*)

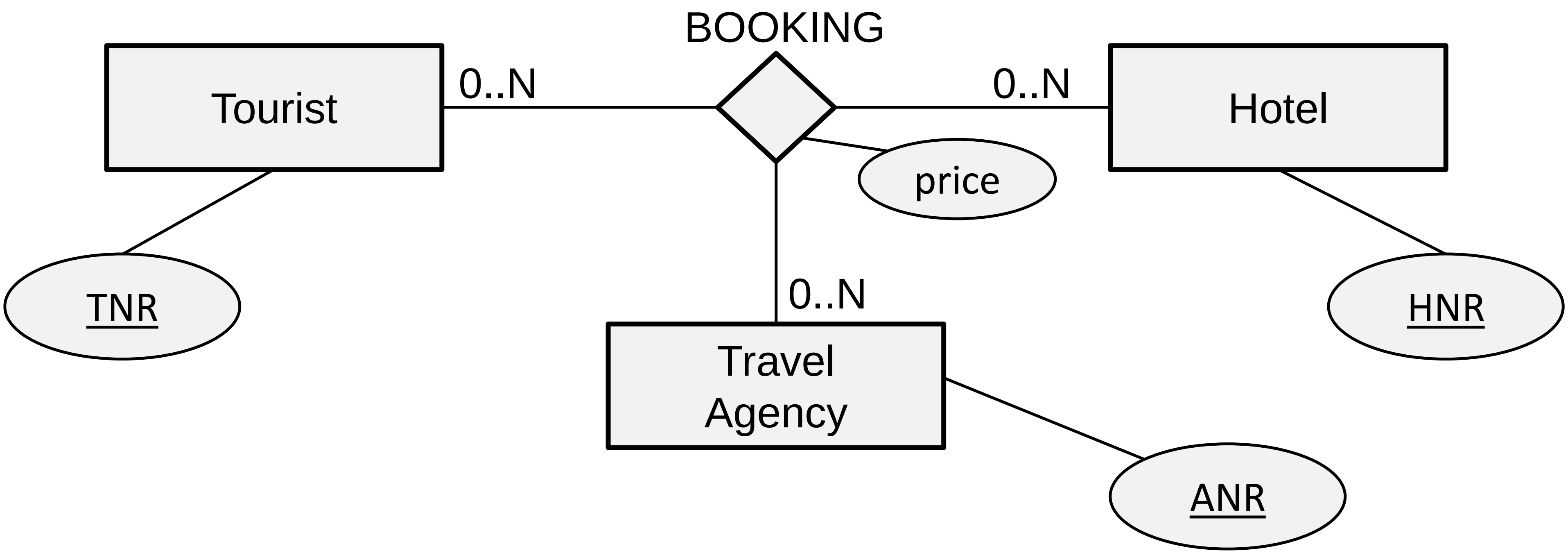
511	John Smith	14 Avenue of the Americas, New York
289	Paul Barker	208 Market Street, San Francisco
356	Emma Lucas	432 Wacker Drive, Chicago
412	Dan Kelly	668 Strip, Las Vegas

289	511
289	356
412	289
412	511

Mapping n-ary Relationship Types

- To map an n-ary relationship type, we first create relations for each participating entity type
- We then also define one additional relation R to represent the n-ary relationship type and add foreign keys referring to the primary keys of each of the relations corresponding to the participating entity types
- The primary key of R is the combination of all foreign keys which are all NOT NULL
- Any attribute type of the n-ary relationship can also be added to R

Mapping n-ary Relationship Types



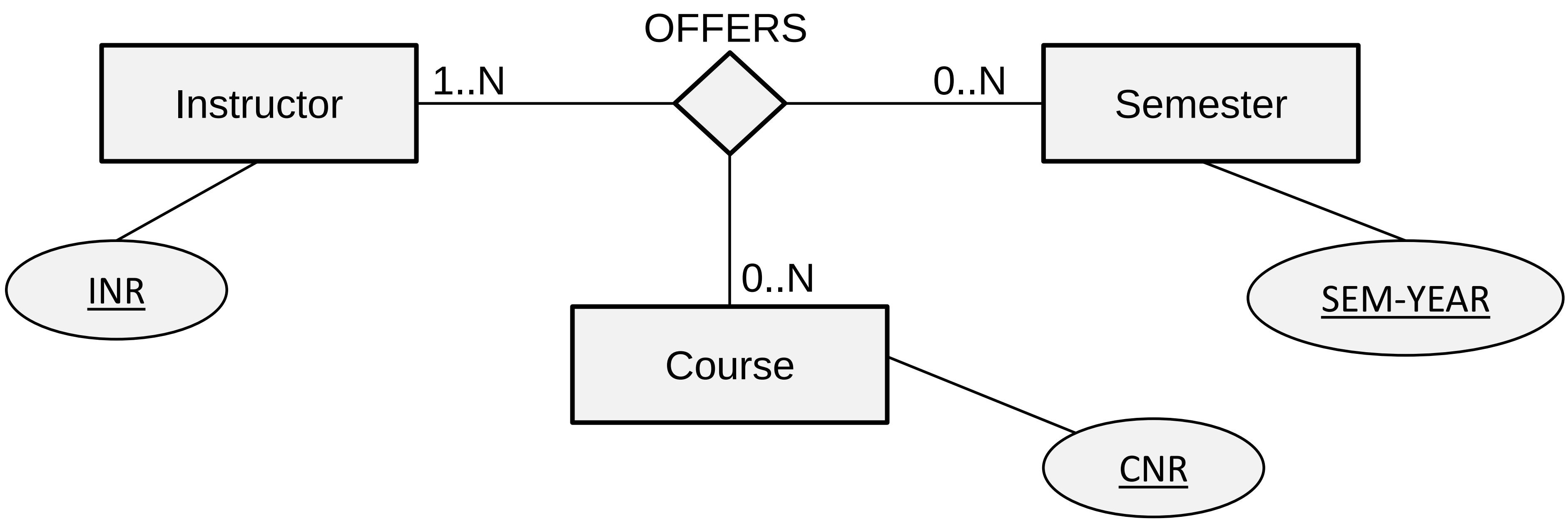
TOURIST(<u>TNR</u>, …)

TRAV_AGENCY(<u>ANR</u>, …)

HOTEL(<u>HNR</u>, …)

BOOKING(<u>*TNR*</u>, <u>*ANR*</u>, <u>*HNR*</u>, price)

Mapping n-ary Relationship Types



INSTRUCTOR(<u>INR</u>, …)

COURSE(<u>CNR</u>, …)

SEMESTER(<u>SEM-YEAR</u>, …)

OFFERS(<u>*INR*</u>,<u>*CNR*</u>,<u>*SEM-YEAR*</u>)

Mapping n-ary Relationship Types

INSTRUCTOR(INR, iname, ….)

10	Bart
12	Wilfried
14	Seppe

COURSE(CNR, cname, ….)

100	Database Management
110	Analytics
120	Java Programming

SEMESTER(SEM-YEAR, ….)

1-2015
2-2015
1-2016

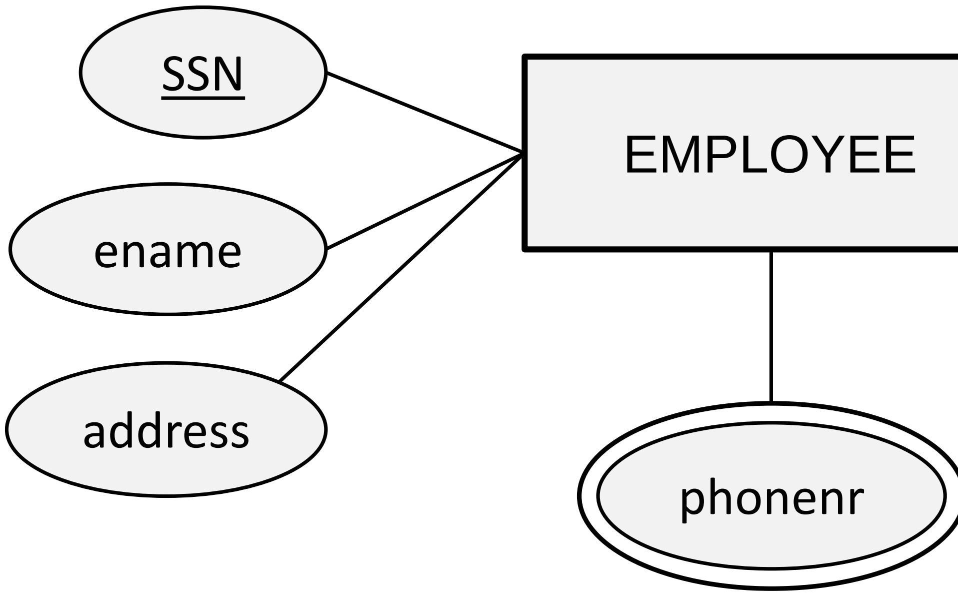
OFFERS(*INR, CNR, SEM-YEAR*)

10	100	1-2015
12	100	1-2016
10	120	1-2015
14	120	1-2015

Mapping Multivalued Attribute Types

- For each multivalued attribute type, we create a new relation R
- We put the multivalued attribute type in R together with a foreign key referring to the primary key of the original relation
- Multivalued composite attribute types are again decomposed into their components
- The primary key can then be set based upon the assumptions

Mapping Multivalued Attribute Types



EMPLOYEE(<u>SSN</u>, ename, address)

EMP-PHONE(<u>PhoneNr</u>, <u>*SSN*</u>)

Mapping Multivalued Attribute Types

EMPLOYEE(<u>SSN</u>, ename, address, *DNR*)

511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	001
356	Emma Lucas	432 Wacker Drive, Chicago	002

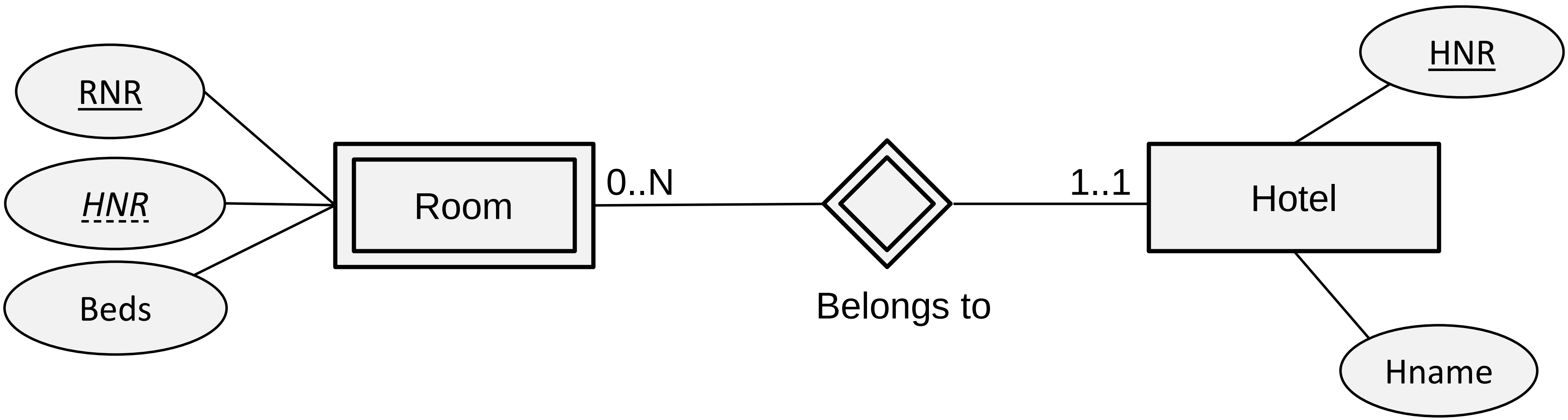
EMP-PHONE(<u>PhoneNR</u>, <u>*SSN*</u>)

900-244-8000	511
900-244-8000	289
900-244-8002	289
900-246-6006	356

Mapping Weak Entity Types

- A weak entity type should be mapped into a relation R with all its corresponding attribute types
- A foreign key must be added referring to the primary key of the relation corresponding to the owner entity type
- Because of the existence dependency, the foreign key is declared as NOT NULL
- The primary key of R is then the combination of the partial key and the foreign key

Mapping Weak Entity Types



Hotel (HNR, Hname)

Room (RNR, *HNR*, beds)

Mapping Weak Entity Types

ROOM (<u>RNR, *HNR*</u>, Beds)

2	101	2
6	101	4
8	102	2

HOTEL(<u>HNR</u>, Hname)

100	Holiday Inn New York
101	Holiday Inn Chicago
102	Holiday Inn San Francisco

Putting it All Together

ER Model	Relational model
Entity type	Relation
Weak entity type	Foreign key
1:1 or 1:N relationship type	Foreign key
M:N relationship type	New relation with two foreign keys
N-ary relationship type	New relation with N foreign keys
Simple attribute type	Attribute type

Putting it All Together

- EMPLOYEE(<u>SSN</u>, ename, streetaddress, city, sex, dateofbirth, *MNR*, *DNR*)
  - MNR foreign key refers to SSN in EMPLOYEE, NULL ALLOWED
  - DNR foreign key refers to DNR in DEPARTMENT, NOT NULL
- DEPARTMENT (<u>DNR</u>, dname, dlocation, *MGNR*)
  - MGNR: foreign key refers to SSN in EMPLOYEE, NOT NULL
- PROJECT (<u>PNR</u>, pname, pduration, *DNR*)
  - DNR: foreign key refers to DNR in DEPARTMENT, NOT NULL
- WORKS-ON (<u>*SSN*</u>, <u>*PNR*</u>, HOURS)
  - SSN foreign key refers to SSN in EMPLOYEE, NOT NULL
  - PNR foreign key refers to PNR in PROJECT, NOT NULL

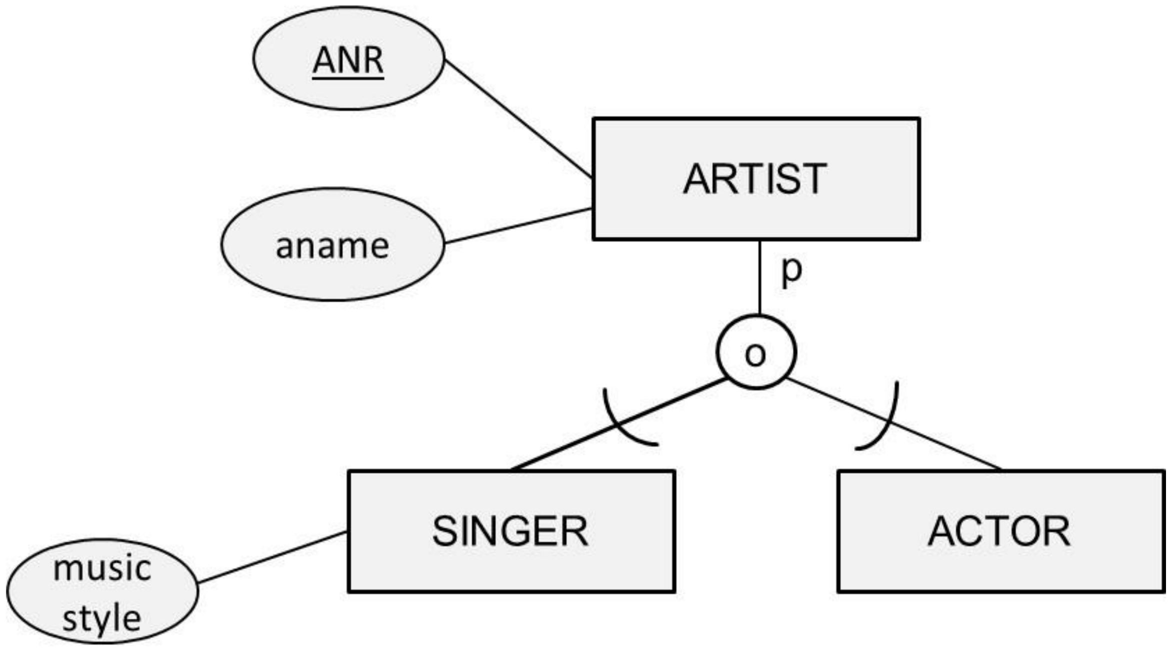
Mapping a Conceptual EER Model to a Relational Model

- Mapping an EER specialization
- Mapping an EER categorization
- Mapping an EER aggregation

Mapping an EER Specialization

- 3 options:
  - Create a relation for the superclass and each subclass and link them with foreign keys
  - Create a relation for each subclass and none for the superclass
  - Create one relation with all attribute types of the superclass and subclasses and add a special attribute type

Mapping an EER Specialization



ARTIST(<u>ANR</u>, aname, …)

SINGER(<u>*ANR*</u>, music style, …)

ACTOR(<u>*ANR*</u>, …)

Mapping an EER Specialization

ARTIST(<u>ANR</u>, aname)

2	Madonna
6	Tom Cruise
8	Claude Monet
12	Andrea Bocelli

Singer(<u>*ANR*</u>, music style)

2	Pop music
12	Classic music

Actor(<u>*ANR*</u>)

6
2

Mapping an EER Specialization

`SINGER(ANR, aname, music style, …)`

`ACTOR(ANR, aname, …)`

Singer(*ANR*, aname, music style)

2	Madonna	Pop music
12	Andrea Bocelli	Classic music

Actor(*ANR*, *aname*)

6	Tom Cruise
2	Madonna

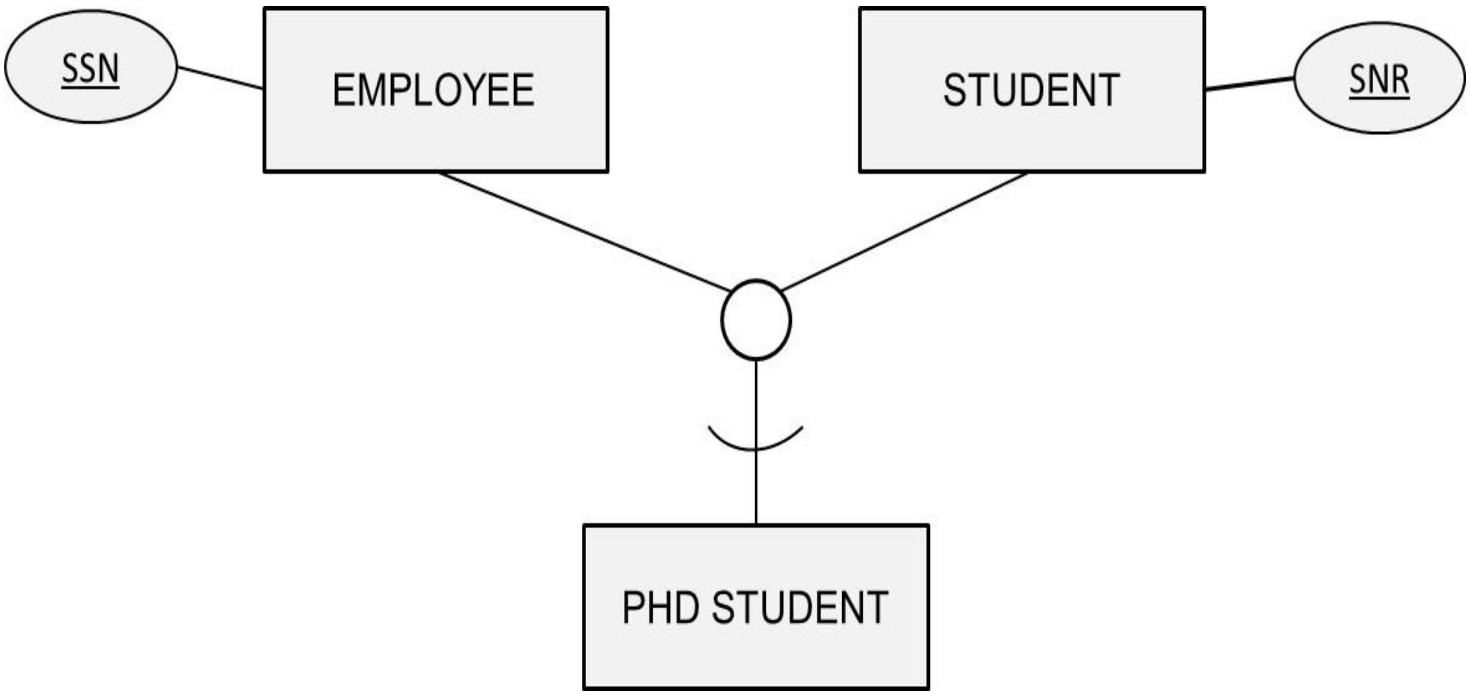
Mapping an EER Specialization

ARTIST(<u>ANR</u>, aname, music style, …, discipline)

Artist(<u>*ANR*</u>, aname, music style, discipline)

2	Madonna	Pop music	Singer/Actor
6	Tom Cruise	NULL	Actor
8	Claude Monet	NULL	Painter
12	Andrea Bocelli	Classic music	Singer

Mapping an EER Specialization

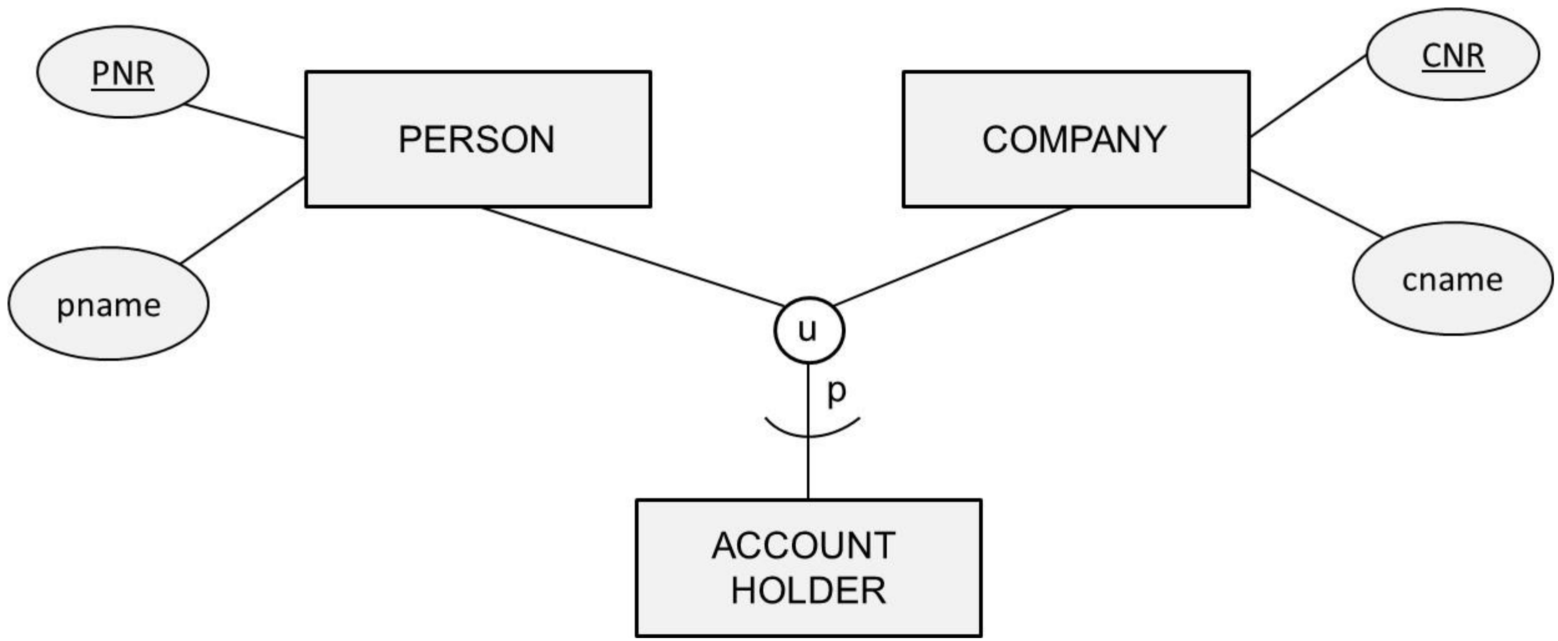


EMPLOYEE(<u>SSN</u>, …)

STUDENT(<u>SNR</u>, …)

PHD-STUDENT(<u>*SSN*</u>, <u>*SNR*</u>, …)

Mapping an EER Categorization



PERSON(<u>PNR</u>, …, *CustNo*)

COMPANY(<u>CNR</u>, …, *CustNo*)

ACCOUNT-HOLDER(<u>CustNo</u>, …)

Mapping an EER Categorization

Person(PNR, pname, … ,*CustNo*)

122	Bart	6
124	Seppe	8
126	Wilfried	NULL

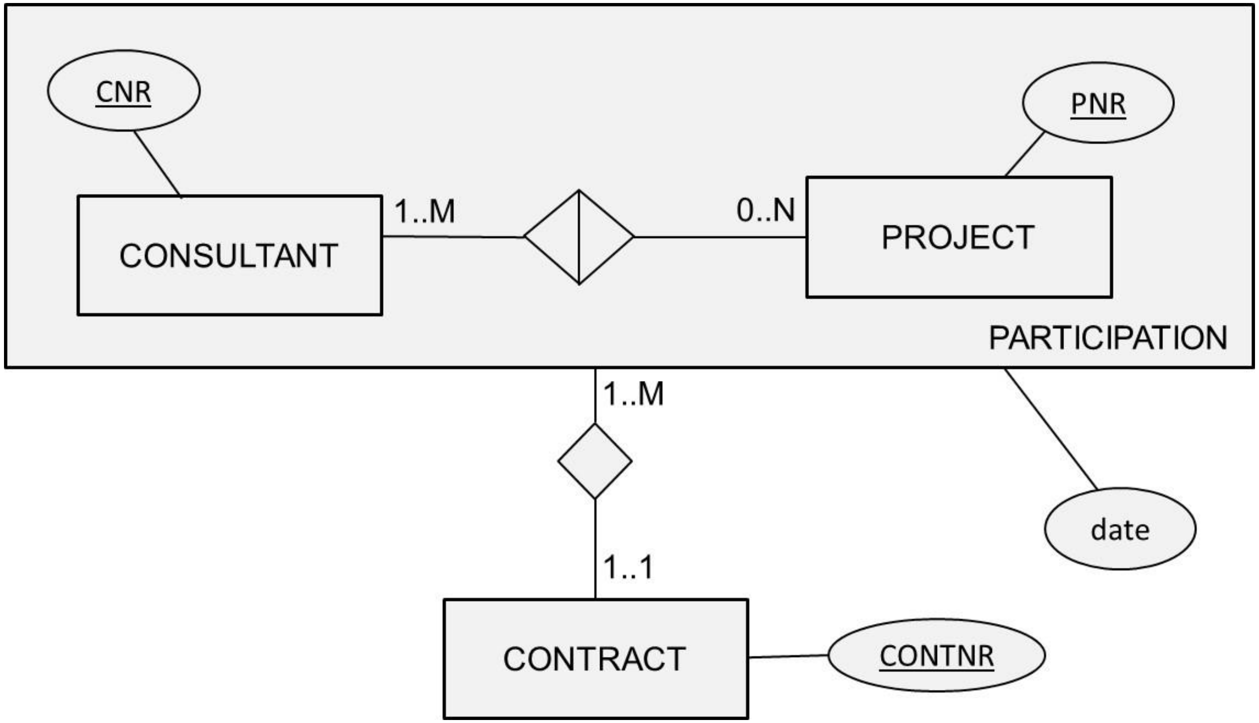
Company(CNR, cname, … ,*CustNo*)

1006	Microsoft	NULL
1008	SAS	10

ACCOUNT-HOLDER(CustNo, …)

6
8
10
12

Mapping an EER Aggregation



CONSULTANT(<u>CNR</u>, …)

PROJECT(<u>PNR</u>, …)

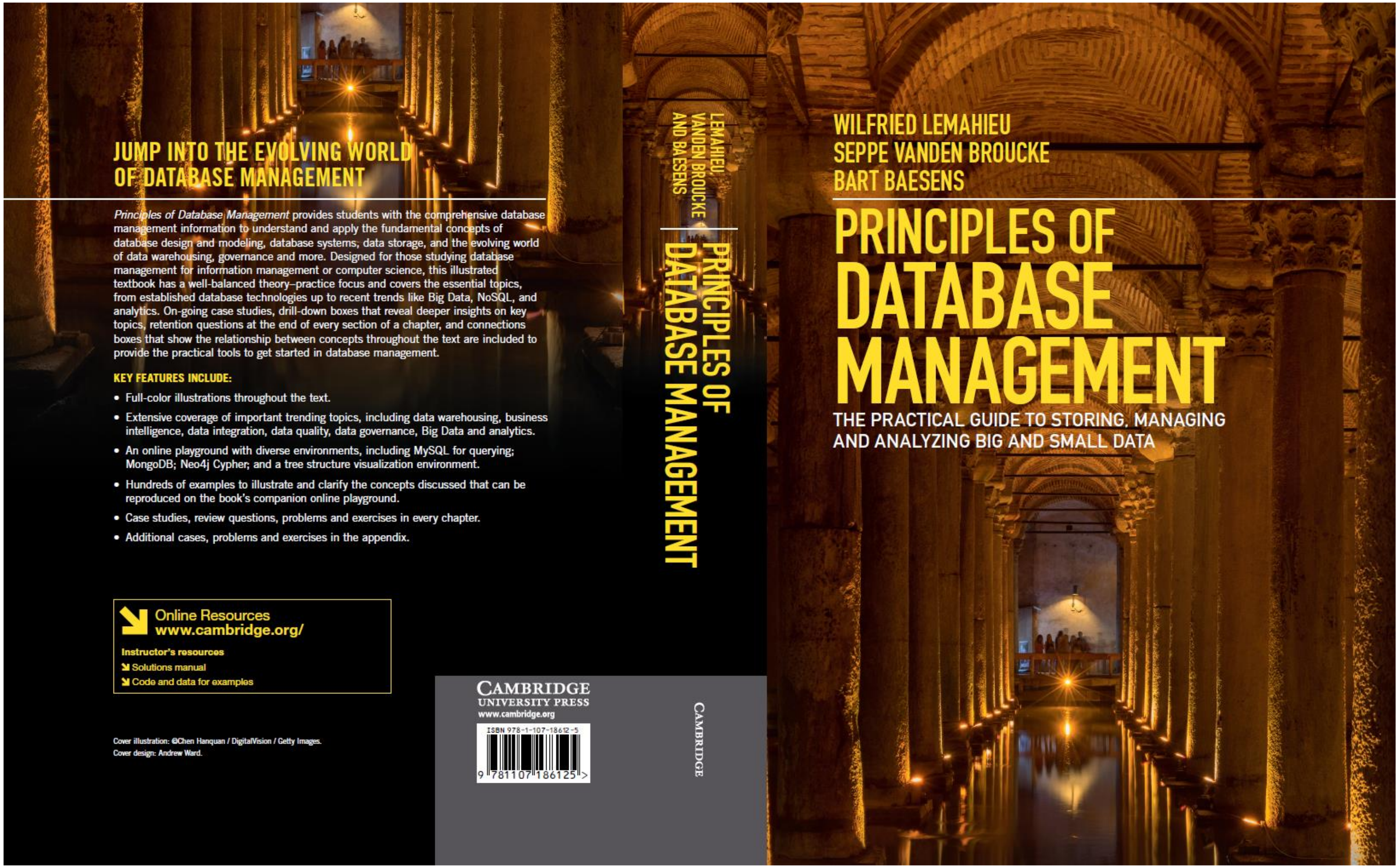
PARTICIPATION(<u>*CNR*</u>, <u>*PNR*</u>, *CONTNR*, date)

CONTRACT(<u>CONTNR</u>, …)

Conclusions

- Relational Model
- Normalization
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a relational model

More information?



www.pdbmbook.com