Logical Data Modeling: The Relational Model

ACS 575: Database Systems

Instructor: Dr. Jin Soung Yoo
Department of Computer Science
Purdue University Fort Wayne

References

- W. Lemanhieu, et al., Principles of Database Management, Ch 6

Outline

- Relational Model
- Normalization
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a 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* (relational algebra and relational calculus)
- Unlike the (E)ER, the relational model has no standard 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
- An attribute of the entity is represented as a column (/data elements) of the relation. Each column has a **column header** that gives an indication of the meaning of the data items in that column.
  - The column header is corresponding to an attribute type in EER model

Example of a Relation

Relation name

Column headers (Attribute types)

SUPPLIER

Tuples

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
…				

* *Relations is* a *table*, with rows and columns.

Basic Concepts

- Correspondence between EER model and relational model

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

- Examples

```
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, e.g.,
  - integer domain
  - gender domain (male and female values),
  - time domain (e.g., define time as day, month and year)
- Each attribute type is defined using a corresponding domain
- A domain can be used multiple times in a relation
  - E.g., majorpronr, ninorprodnr, and quality are integer type.

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 $t = \langle v_1, v_2, v_3, \ldots v_n \rangle$ 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
- Example tuples

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

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

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
  - The *Cartesian product* specifies all possible combinations of values form the underlying domains.

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
…		

- Of all these possible combinations, the current relation state represents only the valid tuples that represent a specific state of the real world.

Outline

- Relational Model
  - Basic Concepts
  - Formal Definitions
  - ☞ **Types of Keys**
  - Relational Constraints
- Normalization
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a relational model

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
  - E.g., `(Studentnr, Name, HomePhone)` is a superkey.
  - `Studentnr` is also a superkey. But `Name` and `HomePhone` values are not unique (can have redundant values).

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**)
  - E.g., `Studentnr`

- The key constraint states that every relation must have at least 1 key that allows to uniquely identify its tuples

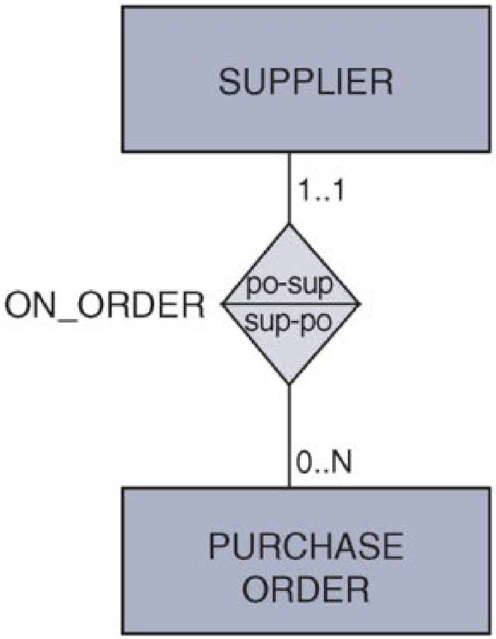
Candidate Keys, Primary Keys and Alternative Keys

- A relation may have more than one key (**candidate keys**)
  - E.g., `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 Key Example 1



SUPPLIER

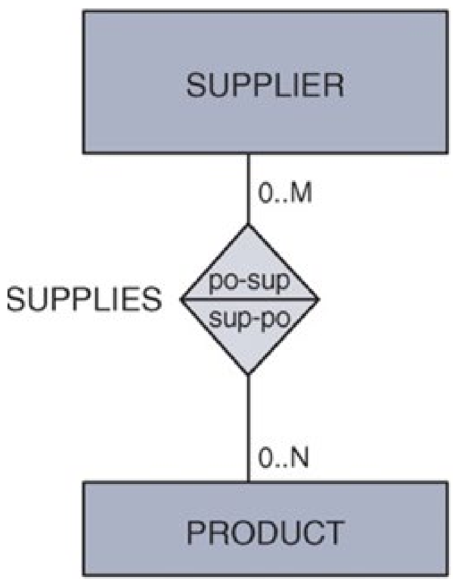
SUPNR	SUPNAME	SUPADDRESS	SUPCITY	SUPSTATUS
...				
37	Ad Fundum	82, Wacker Drive	Chicago	95
94	The Wine Crate	330, McKinney Avenue	Dallas	75
...				

PURCHASE_ORDER

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

A foreign key which references SUPNR, the primary key of SUPPLIER

Foreign Key Example 2



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
…				

FK1

SUPPLIES

SUPNR	PRODNR	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
…			

FK2

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

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 Database State

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
...		

Outline

- Relational Model
- **Normalization**
  - Insertion, Deletion and Update Anomalies
  - Informal Normalization guidelines
  - Functional Dependencies and Prime Attribute Type
  - Normalization forms
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a relational model

Normalization

- **Normalization** of a relational model is a process of analyzing the given relations to ensure they do not contain any redundant data.
- The **goal of normalization** is to ensure that no anomalies can occur during data insertion, deletion, or update.

Unnormalized Relation Data Model

SUPPLIES

SUPNR	PRODNR	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	
…									

PK is (SUPNR, PRODNR)

PO_LINE

PONR	PRODNR	QUANTITY	PODATE	SUPNR
1511	0212	2	2015-03-24	37
1511	0345	4	2015-03-24	37
…				

PK is (PONR, PRODNR)

Redundant information !!

- The SUPPLIES relation also include all the attribute types for SUPPLIER and all the attribute types for PRODUCT.
- The PO_LINE relation includes purchase order date and supplier number.

Insertion, Deletion and Update Anomalies

- A least three types of anomaly may arise when working with an unnormalized relational mode
  - **Insertion anomaly**
  - **Deletion anomaly**
  - **Update anomaly**

Anomaly Examples

- **Insertion anomaly**
  - When we wish to insert a new tuple in the SUPPLIES relation, must be sure to include the correct supplier and product information
  - It is difficult to insert a new product for which there are no suppliers yet or a new supplier who does not supply anything yet since the primary key is a combination of SUPNR and PRODNR, which can thus both not be NULL
- **Deletion anomaly**
  - If delete a particular supplier from the SUPPLIES relation, consequently, all corresponding product data may get lost as well, which is not desirable.

Anomaly Examples

- **Update anomaly**
  - When we wish to update the supplier address in the SUPPLIES relation. This would necessitate multiple updates with the risk of inconsistency

Normalized Relation 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 Needs

- 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

- **Guide 1**: Design a relational model in such a way that it is easy to explain its meaning
  - E.g., `MYRELATION123(SUPNR, SUPNAME, SUPTWITTER, PRODNR, PRODNAME, …)`

    The relation name is not very meaningful!!

    ➔ `SUPPLIER(SUPNR, SUPNAME, SUPTWITTER, PRODNR, PRODNAME, ……)`
- **Guide 2**: Attribute types from multiple entity types should not be combined in a single relation
- **Guide 3**: Avoid excessive amount of NULL values in a relation
  - E.g., `SUPPLIER(SUPNR, SUPNAME, SUPTWITTER …)`

    Not many suppliers have a Twitter account. Many NULL values in SUPTWITTER

    ➔ `SUPPLIER(SUPNR, SUPNAME, …)`

    `SUPPLIER-TWITTER(SUPNR, SUPTWITTER)`

Functional Dependencies

- 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`
    - The employee name is functionally dependent upon the social security number.
    - A social seucrity number uniquely determines an employee name.
  - `PNUMBER` $\rightarrow$ `{PNAME, PLOCATION}`
  - `{SSN, PNUMBER}` $\rightarrow$ `HOURS`

Prime Attribute Type

- A **prime attribute type** is an attribute type that is part of a candidate key
- E.g., `R1(`<u>`SSN`</u>`, `<u>`PNUMBER`</u>`, PNAME, HOURS)`
  - Prime attribute types: `SSN` and `PNUMBER`
  - Non-prime attribute types: `PNAME` and `HOURS`

Full Functional Dependency and Partial Dependency

- A functional dependency X → Y is a **full functional dependency** if removal of any attribute type A from X means that the dependency does not hold anymore
  - E.g., To know the number of hours an employee worked on a project, we need to know both the SSN of the employee and the project number.

    `SSN, PNUMBER` → `HOURS`

    `HOURS` is fully functionally dependent upon `SSN` and `PNUMBER`.
- A functional dependency X → Y is a **partial dependency** if an attribute type A from X can be removed from X and the dependency still holds
  - E.g., `SSN, PNUMBER` → `PNAME`

    `PNAME` only depend upon `PNUMBER`.

Transitive Dependency and Trivial Functional Dependency

- 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
  - e.g., R1(<u>PROJ_NUM</u>, <u>EMP_NUM</u>, JOB_CLASS, CHG_HOUR, WORK_HOURS)
    - If the charge hour is determined by the job class, JOB_CLASS → CHG_HOUR
- A functional dependency $X \rightarrow Y$ is called a **trivial functional dependency** if Y is a subset of X
  - e.g., SSN, NAME → SSN

Multivalued Dependency

- There is a **multivalued dependency** from X to Y, X →→ Y, if and only if each X value exactly determines a set of Y values, independently of the other attribute types
  - E.g., A relation of university courses, the books recommended for the course, and the lecturers who will be teaching the course.

Course	Book	Lecturer
AHA	Silberschatz	John D
AHA	Nederpelt	John D
AHA	Silberschatz	William M
AHA	Nederpelt	William M
AHA	Silberschatz	Christian G
AHA	Nederpelt	Christian G
OSO	Silberschatz	John D
OSO	Silberschatz	William M

- The lecturers attached to the course and the books attached to the course are independent of each other
  Course →→ Book
  Course →→ Lecturer
  - If we were to add a new book to the AHA course, we would have to add one record for each of the lecturers on that course, and vice versa.

Outline

- Relational Model
- Normalization
  - Insertion, Deletion and Update Anomalies
  - Informal Normalization guidelines
  - Functional Dependencies and Prime Attribute Type
  - **Normalization forms**
- Mapping a conceptual ER model to a relational model
- Mapping a conceptual EER model to a relational model

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
    - Hence, no composite or multivalued attribute types (domain constraint!)
- Example:
  - `SUPPLIER(SUPNR, NAME(FIRST NAME, LAST NAME), SUPSTATUS)`

    This relation is not in 1NF due to the composite attribute type.

  ➔ `SUPPLIER(SUPNR, FIRST NAME, LAST NAME, SUPSTATUS)`

    Now it is at least in 1NF

1 NF Example

- `DEPARTMENT(DNUMBER, {DLOCATION}, DMGRSSN)`
  - Assume a department can have multiple locations and multiple departments are possible at a given location
  - This relation is not in 1NF due to the multi-valued attribute type

➔ `DEPARTMENT(DNUMBER, DMGRSSN)`
`DEP-LOCATON(`*`DNUMBER`*`, DLOCATION)`

- Note that *italic* is used for indicating a foreign key

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

Now, both relations are at least in 1NF

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

1 NF Example

- R1(SSN, ENAME, DNUMBER, DNAME,{PROJECT(PNUMBER, PNAME, HOURS)})
  - Assume an employee can work on multiple projects and multiple employees can work on the same project
  - This relation is not in 1NF due to the multi-valued composite attribute type

→ R11(SSN, ENAME, DNUMBER, DNAME)
R12(*SSN*, PNUMBER, PNAME, HOURS)

Now, both relations are at least in 1NF

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 (i.e., There are no partial dependency)
- 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

2 NF Example

- `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
  - `R1` is not in 2NF because `PNAME` is not fully functionally dependent on the primary key, it only depends on `PNUMBER`.

➔ `R11(SSN, PNUMBER, HOURS)`
`R12(PNUMBER, PNAME)`

Now, both relations are at least in 2NF

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 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

3 NF Example

- 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
  - R1 is not in 3NF because of two transitive dependences
    - DNAME is transitively dependent on SSN via DNUMBER. (DNUMBER is functionally dependent on SSN)
    - DMGRSSN is transitively dependent on SSN via DNUMBER. (DMGRSSN is functionally dependent on DNUMBER)

→ R11(<u>SSN</u>, ENAME, *DNUMBER*)
R12(<u>DNUMBER</u>, DNAME, *DMGRSSN*)

Now, both relations are at least in 3NF

<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 relation R is in the **Boyce-Codd normal form (BCNF)** (also referred to as **3.5 NF**) provided
  - *each* of its non-trivial functional dependencies $X \rightarrow Y$, X is a superkey—that is, X is either a candidate key or a superset thereof
  - In other words, for every one of dependencies, $X \rightarrow Y$, at least one of the following conditions hold
    - $X \rightarrow Y$ is a trivial functional dependency ($Y \subset X$)
    - X is a superkey.
- BCNF normal form is stricter than the third normal form
  - Every relation in BCNF is also in 3 NF (not vice versa)

BCNF (3.5 NF) Example

- 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
  - R1 is not in BCNF because
    - SUPNR and PRODNR are a superkey of the relation
    - A non-trivial functional dependency between SUPNR and SUPNAME

→ R11(*SUPNR*, PRODNR, QUANTITY)
R12(SUPNR, SUPNAME)

Now, both relations are at least in 3.5 NF

Fourth Normal Form (4 NF)

- 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

- The 2, 3 and BC normal forms are concerned with functional dependencies. 4 normal form is concerned with a more general type of dependency kwon as a multivalued dependency.

4 NF Example

- `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
  - `R1` is not in 4NF due to a multi-valued dependency between course and textbook.
    - Each course exactly determines a set of textbooks, independently of the instructor

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

→ `R11(course, textbook)`
`R12(course, instructor)`

Now, both relations are at least in 4 NF

COURSE	INSTRUCTOR
Database Management	Baesens
Database Management	Lemahieu

COURSE	BOOK
Database Management	Database cookbook
Database Management	Databases for dummies

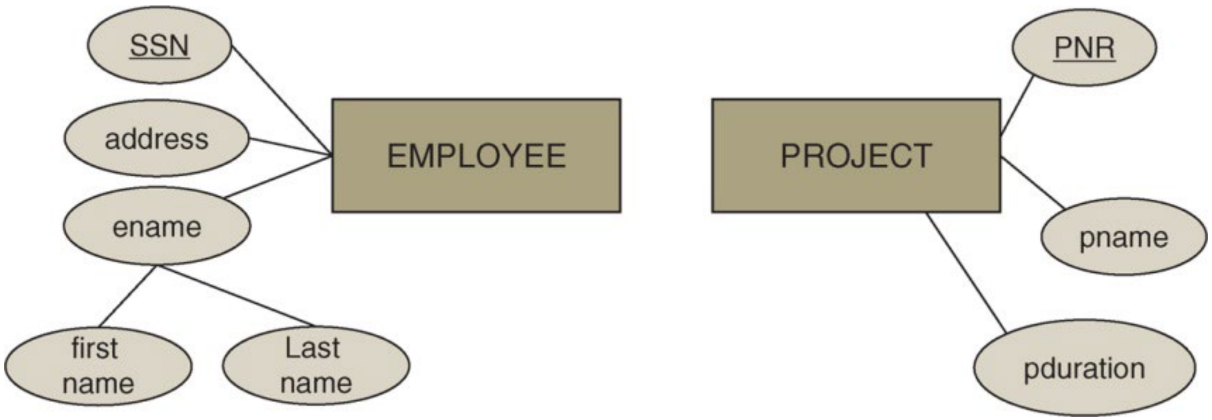
Outline

- Relational Model
- Normalization
- **Mapping a conceptual ER model to a relational model**
  - Mapping Entity Types
  - Mapping Relationship Types
  - Mapping Multivalued Attribute Types
  - Mapping Weak Entity Types
- Mapping a conceptual EER model to a relational model

Mapping Entity Types

- Map each entity type into a relation.
- Simple attribute types can be directly mapped
- A composite attribute type needs to be decomposed into its component attribute types
- One of the key attribute types of the entity type can be set as the primary key of the relation.

Example: Mapping Entity Types



EMPLOYEE(SSN, address, first name, last name)

PROJECT(PNR, pname, pduration)

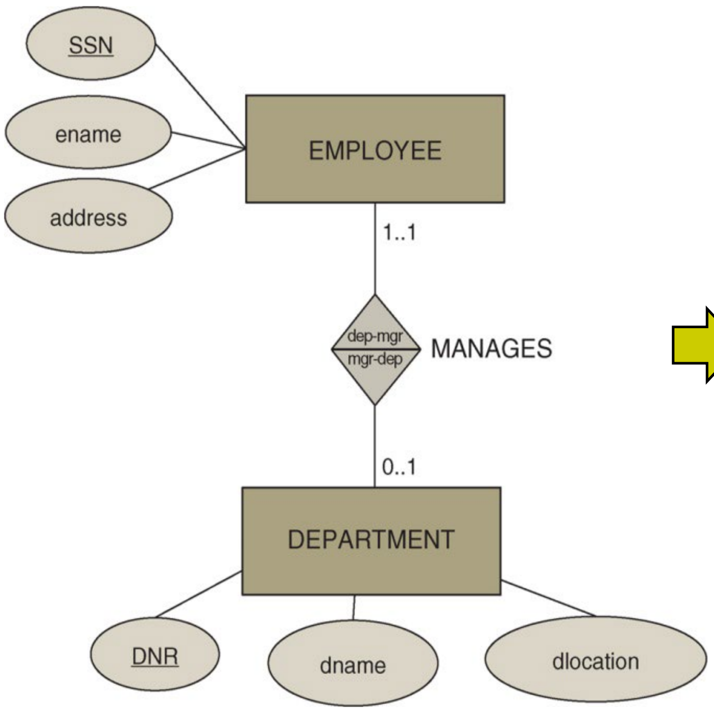
Mapping Relationship Types

- 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

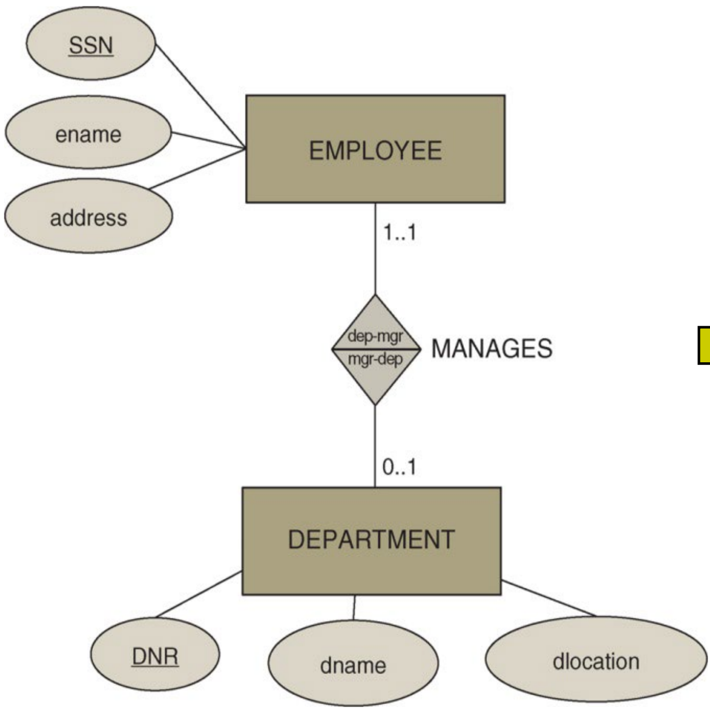
- Create two relations: one for each entity type participating in the relationship type
- The connection can be made by including a foreign key in one of the relations to the primary key of the other
- In case of existence dependency (i.e., total participation, minimum cardinality=1), <u>put the foreign key in the existent dependent relation and declare it as NOT NULL</u>
- The attribute types of the 1:1 relationship type can then be added to the relation with the foreign key

1:1 Relationship Mapping Example (Option 1)



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

1:1 Relationship Mapping (Option 1) - Validation



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

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

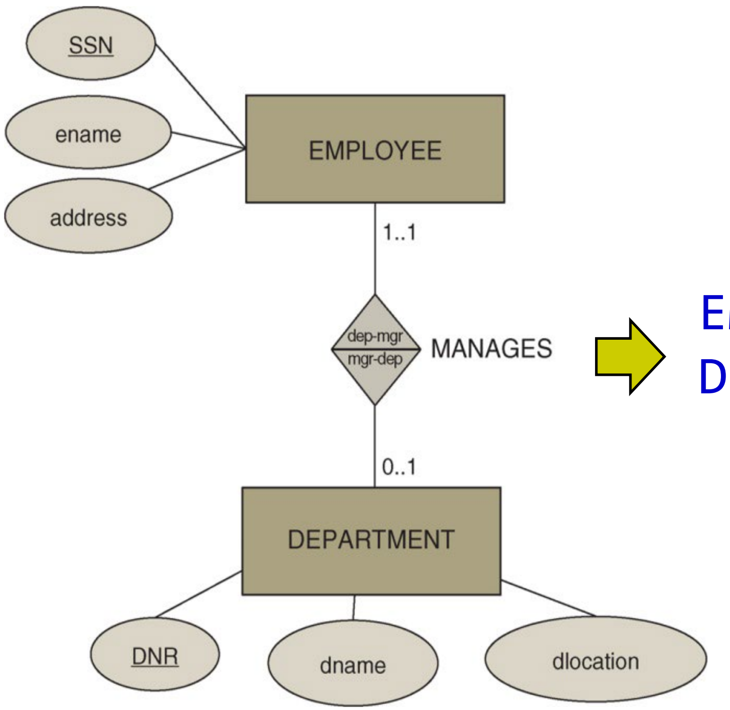
511	John Smith	14 Avenue of the Americas, New York	001
289	Paul Barker	208 Market Street, San Francisco	003
356	Emma Lucas	432 Wacker Drive, Chicago	NULL
412	Michael Johnson	1134 Pennsylvania Avenue, Washington	NULL
564	Sarah Adams	812 Collins Avenue, Miami	001

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

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

- With this relational mode,
  - Can a department have zero managers? Yes
  - Can a department have more than one manager? Yes
  - Can an employee manage zero department? Yes
  - Can an employee manage more than one department? No
  - A lot of NULL values for the DNR foreign key

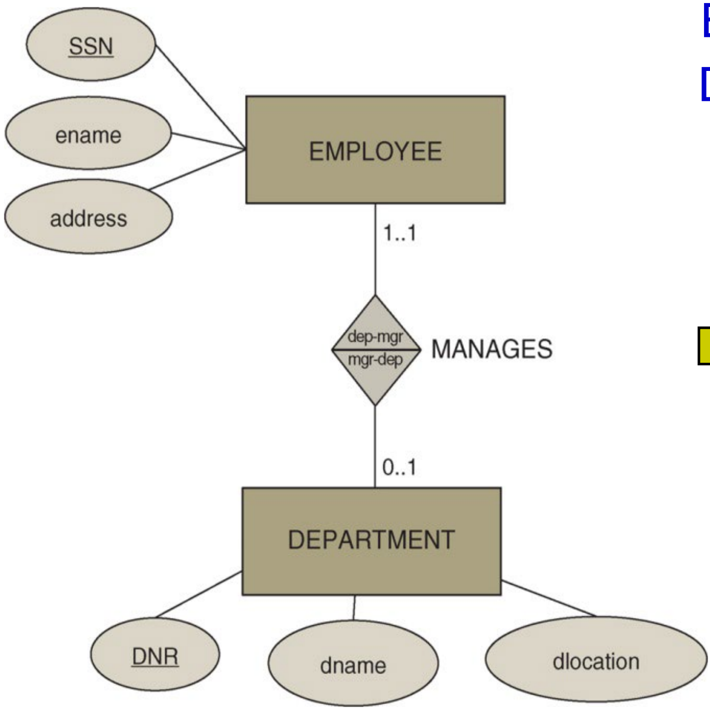
1:1 Relationship Mapping Example (Option 2)



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

* The SSN foreign key is NOT NULL

1:1 Relationship Mapping (Option 2) - Validation



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

* The SSN foreign key is NOT NULL

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

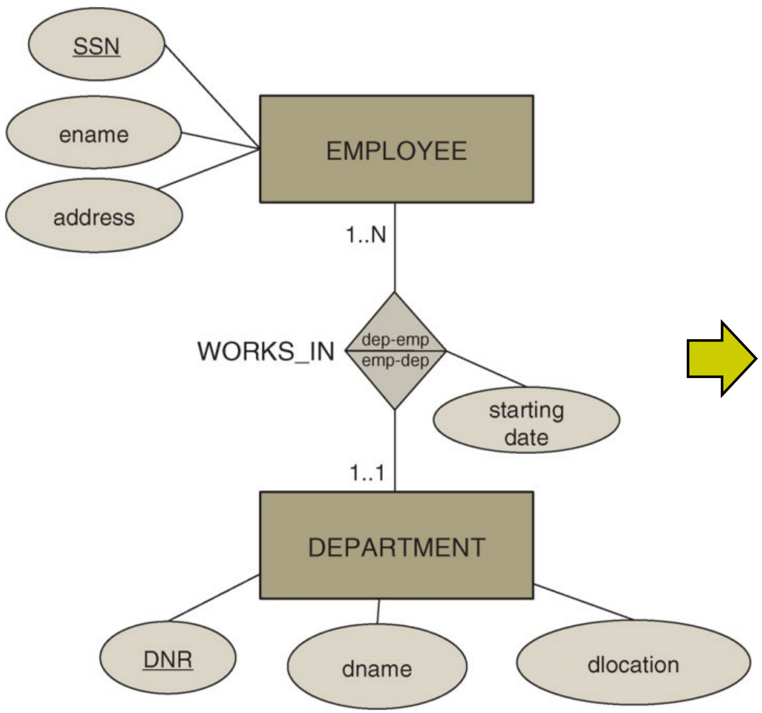
- With this relational mode,
  - Can a department have zero managers? No
  - Can a department have more than one manager? No
  - Can an employee manage zero department? Yes
  - Can an employee manage more than one department? Yes

Option 2 is preferred!!

Mapping a Binary 1:N Relationship Type

- Binary 1:N relationship types can be mapped by <u>including a foreign key in the relation</u> corresponding to the participating entity type <u>at the N-side of the relationship type</u>
- 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

1:N Relationship Mapping Example

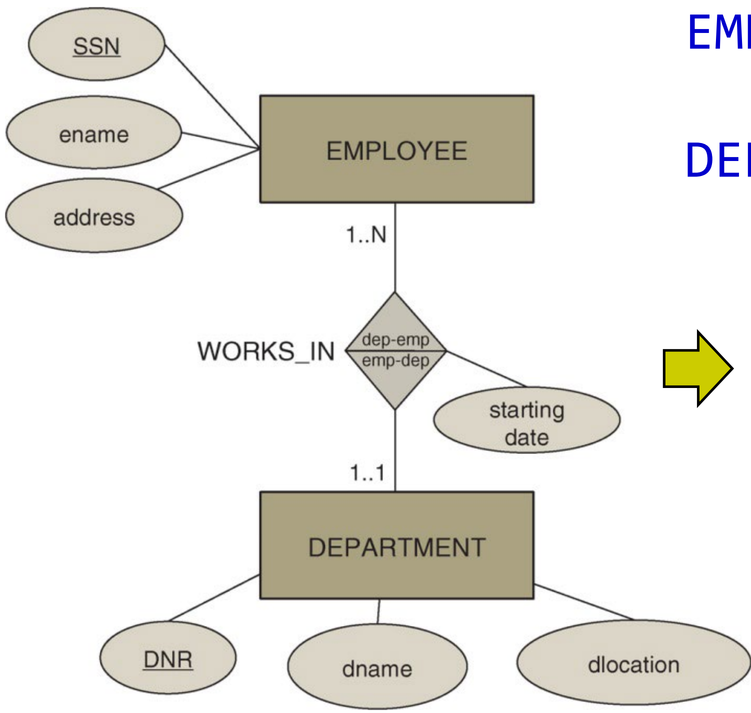


EMPLOYEE(<u>SSN</u>, ename, address,
starting date, *DNR*)

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

* The DNR foreign key is NOT NULL

1:N Relationship Mapping - Validation




```
EMPLOYEE(SSN, ename, address,
         starting date, DNR)
DEPARTMENT(DNR, dname, dlocation)
```

* The DNR foreign key is NOT NULL

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)

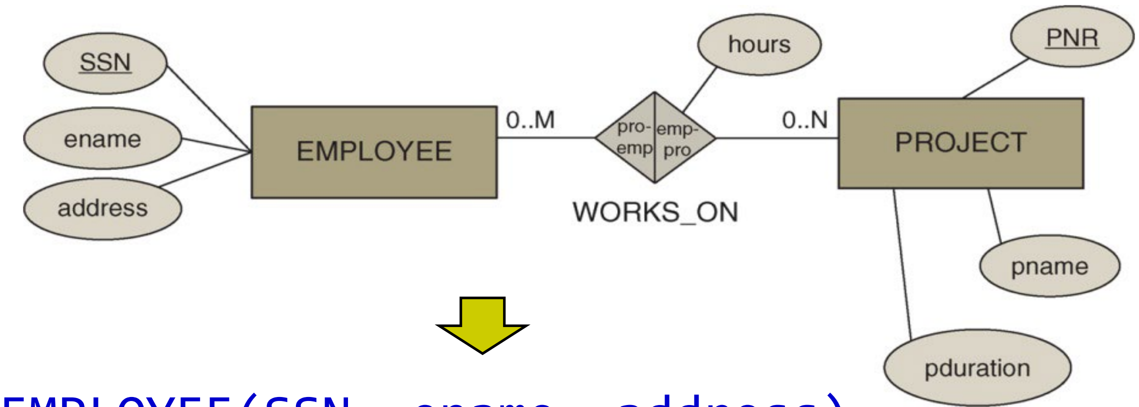
001	Marketing	3th floor
002	Call center	2nd floor
003	Finance	basement
004	ICT	1st floor

- With this relational mode,
  - Can we ensure that an employee works in exactly one department ? No
  - Can a department have more than one employee? Yes
  - Can we guarantee that every department has at least one employee? No
- Any semantics lost in the mapping should be documented and followed up using application code !!

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

N:M Relationship Mapping



EMPLOYEE(<u>SSN</u>, ename, address)
PROJECT(<u>PNR</u>, pname, pduration)
WORKS_ON(<u>*SSN*</u>, <u>*PNR*</u>, hours)

- Validation
  - This relational mode satisfies all four cardinalities !!

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

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

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

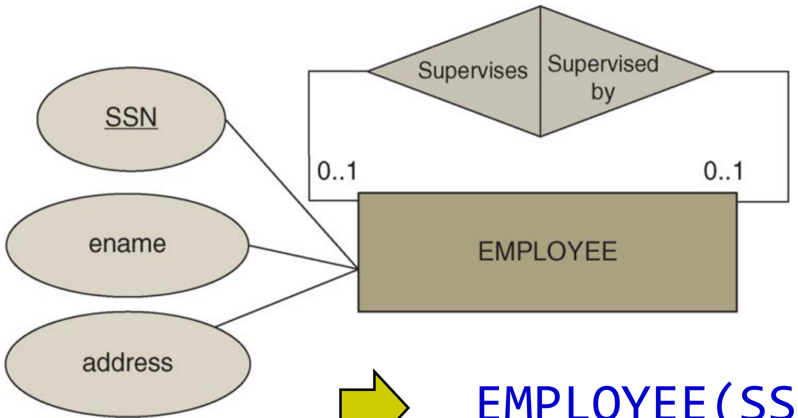
WORKS_ON(<u>*SSN*, *PNR*</u>, hours)

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 <u>adding a foreign key referring to the primary key of the same relation</u>
- 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

Unary Relationship Mapping Example



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

* The DNR foreign key is NOT NULL

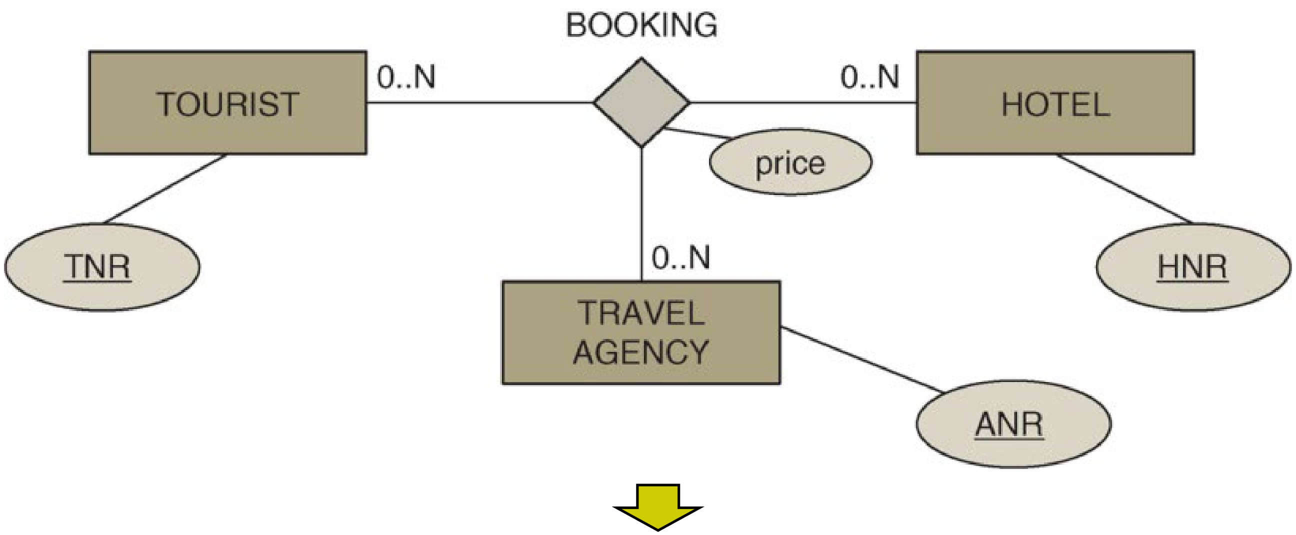
EMPLOYEE(<u>SSN</u>, ename, address, *supervisor*)			
511	John Smith	14 Avenue of the Americas, New York	289
289	Paul Barker	208 Market Street, San Francisco	412
356	Emma Lucas	432 Wacker Drive, Chicago	289
412	Dan Kelly	668 Strip, Las Vegas	NULL

- Validation
  - Out of the four cardinalities, three are supported by this relational mode.
    - Some employees supervise more than one other employee.
  - Again, any semantics lost in the mapping should be documented and followed up using application code !!

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

n-ary Relationship Mapping Example 1



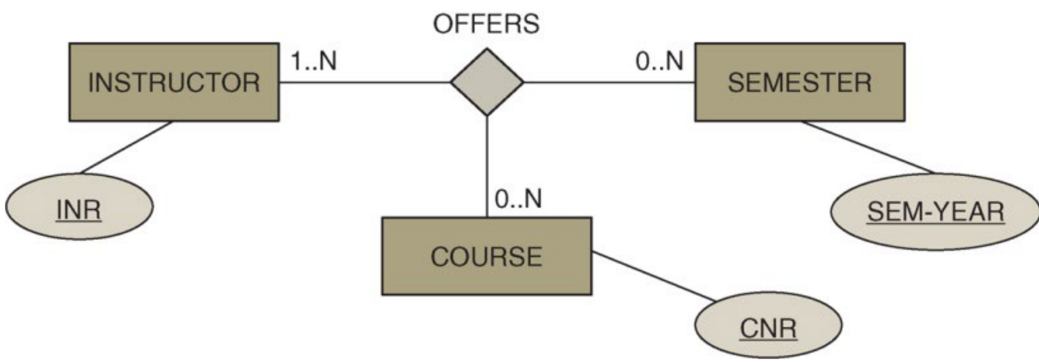
TOURIST(<u>TNR</u>, …)

TRAV_AGENCY(<u>ANR</u>, …)

HOTEL(<u>HNR</u>, …)

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

n-ary Relationship Mapping Example 2



INSTRUCTOR(INR, …)

COURSE(CNR, …)

SEMESTER(SEM-YEAR, …)

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

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*)

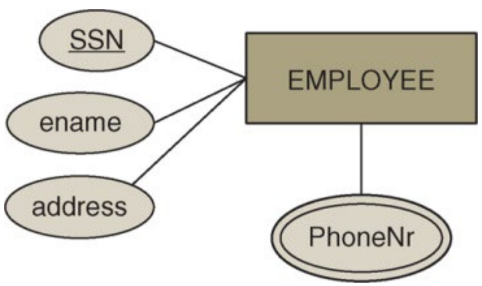
10	100	1-2015
12	100	1-2016
10	120	1-2015
14	120	1-2015

- Validation
  - The minimum cardinality of 1, starting that during a semester a course should be offered by at least one instructor, cannot be guaranteed by the relational model.

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

Multivalued Mapping Example



EMPLOYEE(SSN, ename, address)
EMP-PHONE(PhoneNr, *SSN*)
* Assume that each phone number is assigned to only one employee

EMPLOYEE(SSN, ename, address)
EMP-PHONE(PhoneNr, *SSN*)
* Assume that a phone number can be shared by multiple employees

- Validation

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

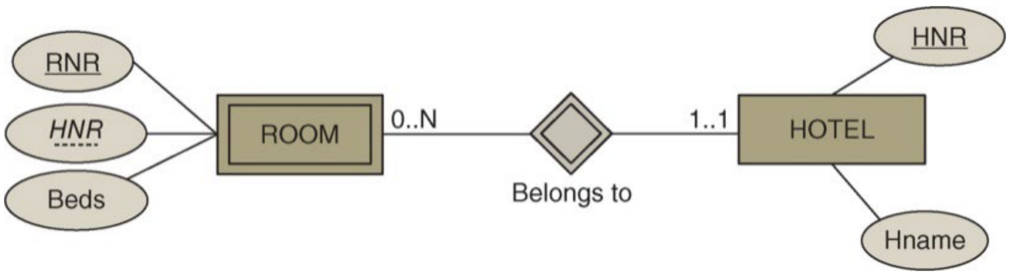
EMP-PHONE(PhoneNr, *SSN*)

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

Weak Entity Type Mapping Example



Hotel (HNR, Hname)

Room (RNR, *HNR*, beds)

- Validation

ROOM (RNR, *HNR*, Beds)

2	101	2
6	101	4
8	102	2

HOTEL (HNR, 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
Composite attribute type	Component attribute type
Multivalued attribute type	Relation and foreign key
Key attribute type	Primary or alternative key

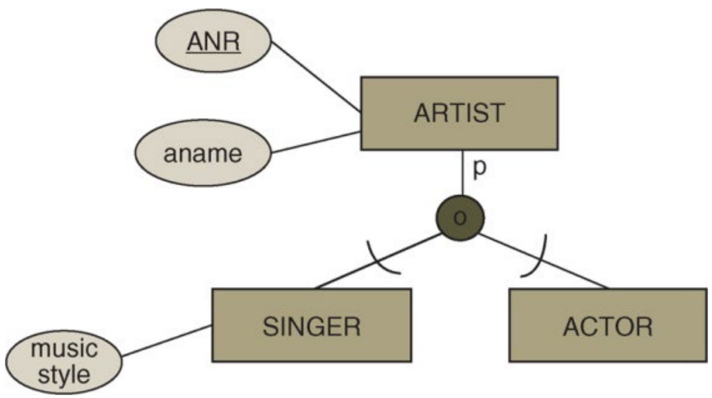
Outline

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

EER Specialization: Example 1



ARTIST(ANR, aname, …)
SINGER(*ANR*, music style, …)
ACTOR(*ANR*, …)

- Validation

ARTIST(ANR, aname, ...)

2	Madonna	...
6	Tom Cruise	
8	Claude Monet	
12	Andrea Bocelli	

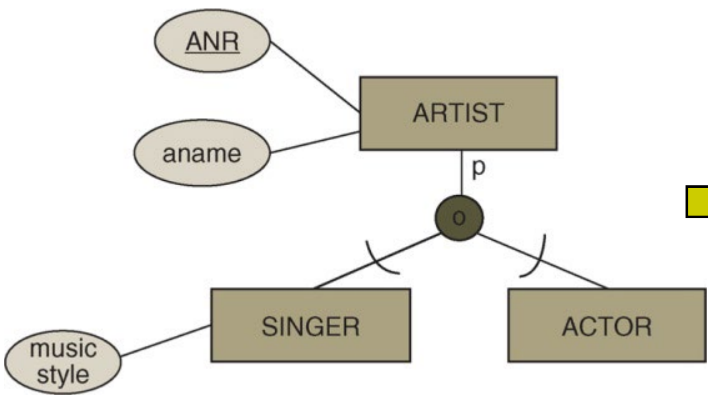
SINGER(*ANR*, music style, ...)

2	Pop music	...
12	Classical music	

ACTOR(*ANR*, ...)

6	...
2	

Example 2



SINGER(ANR, aname, music style, …)
ACTOR(ANR, aname, …)

- Validation

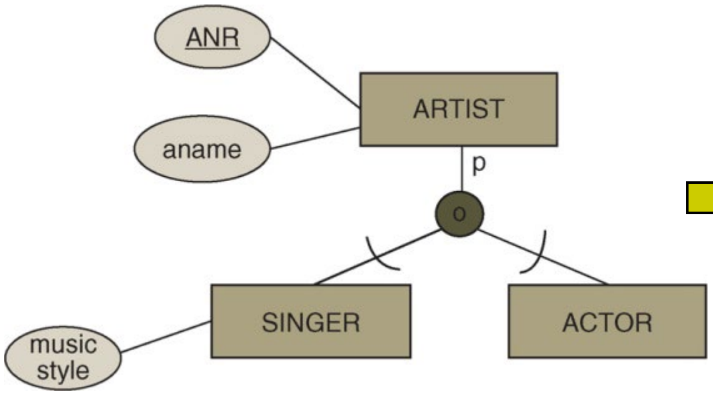
SINGER(ANR, aname, music style, ...)

2	Madonna	Pop music	...
12	Andrea Bocelli	Classical music	

ACTOR(ANR, aname, ...)

6	Tom Cruise	...
2	Madonna	

Example 3

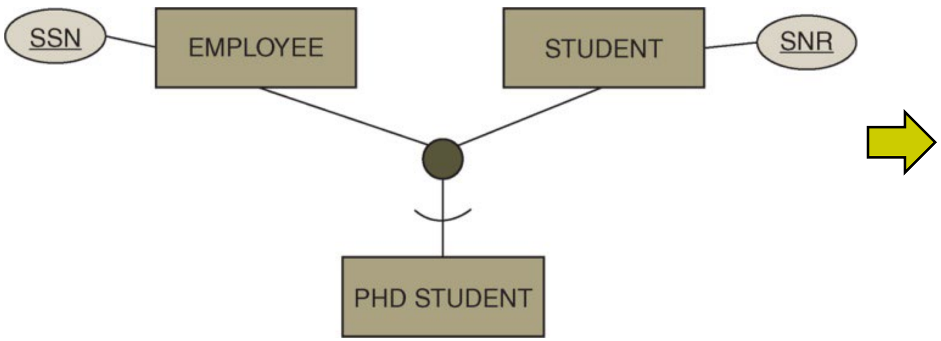


ARTIST(<u>ANR</u>, aname, music style, …, discipline)

- Validation

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	Classical music	Singer	

EER Specialization: Example

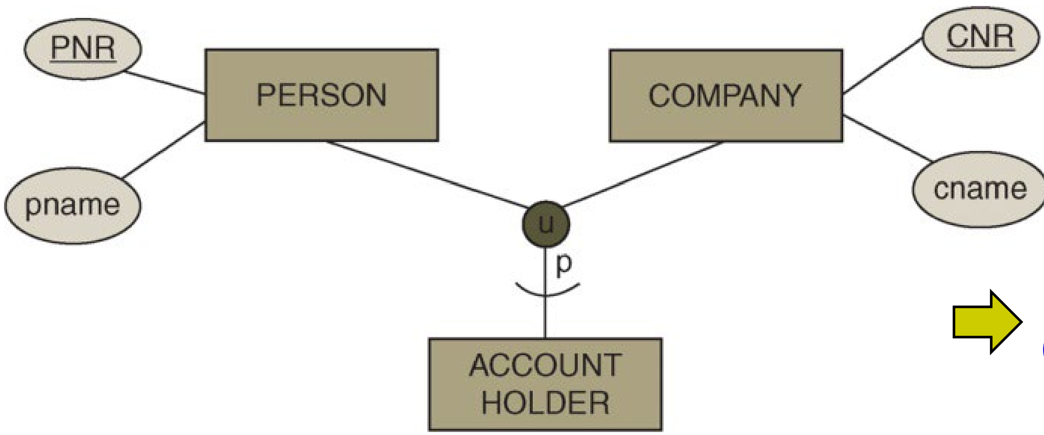


EMPLOYEE(<u>SSN</u>, …)

STUDENT(<u>SNR</u>, …)

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

Mapping an EER Categorization



PERSON(PNR, …, *CustNo*)
COMPANY(CNR, …, *CustNo*)
ACCOUNT-HOLDER(CustNo, …)

- Validation

PERSON(PNR, pname, ... , *CustNo*)

122	Bart	...	6
124	Seppe		8
126	Wilfried		NULL

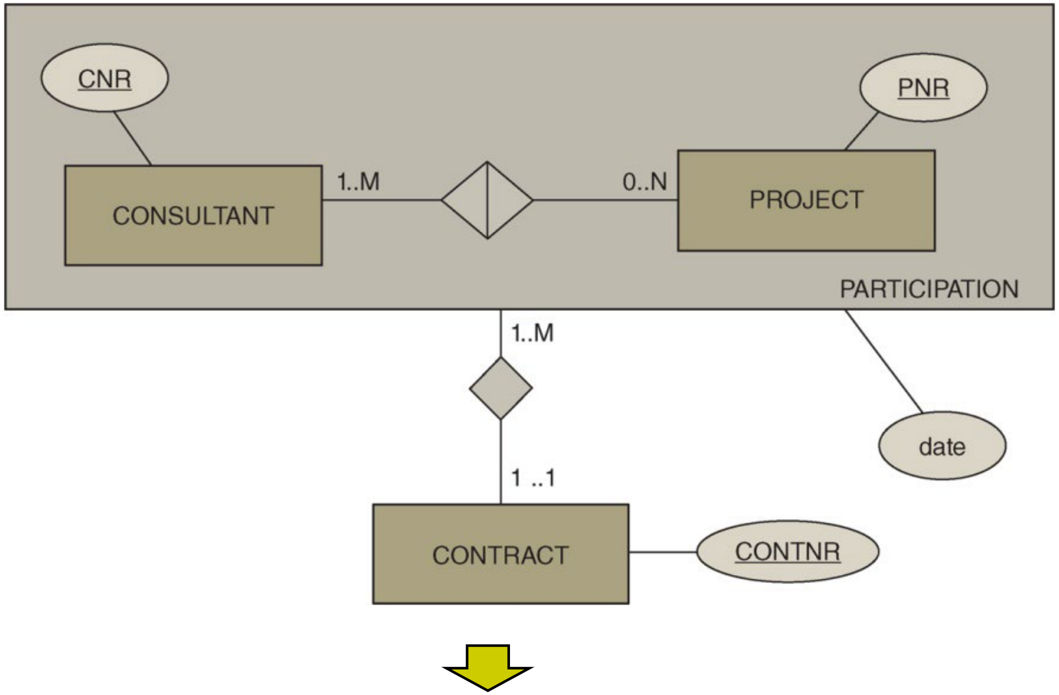
COMPANY(CNR, cname, ... , *CustNo*)

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

APPENDIX :

Mapping UML Class Diagram to Relational Model

Reference: https://web.fe.up.pt/~ssn/2010/lbaw/slides/lbaw-uml2rel.eng.pdf

Mapping Rule from UML Class Diagram to Relational Model

Rule 1	Classes are mapped into relation schemas
Rule 2	Class attributes are mapped to attributes of relations.
Rule 3	Operations of classes are generally not mapped. They can nevertheless be mapped to *stored procedures,* stored and executed in the global context of the database involved.
Rule 4	Objects are mapped into tuples of one or more relations.
Rule 5	Each object is uniquely identified. If the identification of an object is defined **explicitly** by the OID *(object identifier*) stereotype, associated with one or more attributes, this attribute is mapped to primary key in the relation schema. Otherwise, we assume **implicitly** that the corresponding primary key is derived from a new attribute with the name of the relation and common suffix (e.g. "PK", "ID").
Rule 6:	The mapping of many-to-many associations involves the creation of a new relation schema, with attributes acting together as primary key, and individually as foreign key for each of the schemas derived from the classes involved.
Rule 7:	The mapping of one-to-many associations involves the introduction, in the relation schema corresponding to the class that has the constraint "many", of a foreign key attribute for the other schema.
Rule 8:	The mapping of one-to-one associations has in general two solutions. The first corresponds to the fusion of the attributes of the classes involved in one common schema. The second solution is to map each of the classes in the corresponding schema and choose one of the schemas as the most suitable for the introduction of a foreign key attribute for the other schema. This attribute should also be defined as unique within that schema.

Mapping Rule from UML Class Diagram to Relational Model (cont.)

Rule 8:	The mapping of one-to-one associations has in general two solutions. The first corresponds to the fusion of the attributes of the classes involved in one common schema. The second solution is to map each of the classes in the corresponding schema and choose one of the schemas as the most suitable for the introduction of a foreign key attribute for the other schema. This attribute should also be defined as unique within that schema.
Rule 9:	Association navigability in general has no impact on the mapping process. The exception lies in one-to-one associations, when they are complemented with navigation cues it helps in the selection of the schema that should include the foreign key attribute.
Rule 10:	Aggregation and composition associations have a minimal impact on the mapping process, which may correspond to the definition of constraints cascade ("CASCADE") in changing operations and/or removal of tuples.
Rule 11:	The mapping of generalization associations in general presents three solutions. The first solution consists in crushing the hierarchy of classes in a single schema corresponding to the original superclass. This solution is appropriate when there is a significant distinction in the structure of sub-classes and/or when the semantics of their identification is not strong. The second solution is to consider only schemas corresponding to the sub-classes and duplicate the attributes of the super-class in these schemas; in particular it works if the super-class is defined as abstract. The third solution is to consider all the schemas corresponding to all classes of the hierarchy, resulting in a mesh of connected schemas and maintained at the expense of referential integrity rules. This solution has the advantage of avoiding duplication of information among different schemas, but suggests a dispersion of information by various schemas, and might involve a performance penalty in query operations or updating of data by requiring the execution of various join operations (i.e. "JOIN") and/or validation of referential integrity.