

COMP67321

Relational Data Model

Goran Nenadic

School of Computer Science
University of Manchester

1

Aims

- Learn and understand basic concepts of the relational data model
- Understand structure, constraints and operations in relational databases
 - relational schema
 - constraint violations
- Overview mapping from EER to relational model

2

Plan

- Concepts in relational model
 - relations, schemas, constraints
 - update operations
- Mapping ER to relational model
- Mapping EER to relational model

3

Relational model concepts

- Based on the concept of a *relation*
- Formal relational model
 - the strength of the relational approach to data management comes from the formal foundation provided by the theory of relations
 - in *practice*, there is a *standard model* based on SQL – this will be reviewed later
 - there are several important differences between the *formal* model and the *practical* model
- The model was first proposed by Codd in 1970

4

Informal definitions

- A relation looks like a **table** of values.
- A relation typically contains a **set of rows**.
- The data elements in each **row** represent certain facts that correspond to a real-world **entity** or **relationship**
 - in the formal model, rows are called **tuples**
- Each **column** has a column header that gives an indication of the meaning of the data items in that column
 - in the formal model, the column header is called an **attribute name** (or just **attribute**)

5

example

Example of a relation

DEPARTMENT		
DEPTNO	DNAME	LOCATION
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute

6

example

Example of a relation

Relation Name

↓

STUDENT

Attributes

↓

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	26	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Tuples

7

Degree and cardinality

- *Degree* of a relation is the number of its attributes
- *Cardinality* of a relation is the number of its tuples

DEPARTMENT

degree = 3

DEPTNO	DNAME	LOCATION
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

cardinality=4

8

Informal definitions – keys

- Key of a relation:
 - Each row has a value of a data item (or set of items) that uniquely identifies that row in the table
 - this is called the *key*
 - e.g. in the DEPARTMENT table it is DEPTNO
 - e.g. in the STUDENT table, SSN is the key
 - Sometimes row-ids or sequential numbers are assigned as keys to identify the rows in a table
 - called *artificial key* or *surrogate key*

9

Formal definitions - schema

- **Schema** (or description) of a relation:
 - Denoted by $R(A_1, A_2, \dots, A_n)$
 - R is the **name** of the relation
 - A_1, A_2, \dots, A_n are the **attributes** of the relation
- E.g. CUSTOMER (Cust-id, Cust-name, Address, Phone#)
 - CUSTOMER is the relation name
 - Defined over the four attributes:
Cust-id, Cust-name, Address, Phone#
- Each attribute has a **domain** or a set of valid values
 - e.g. the domain of Cust-id is 6 digit numbers

10

Formal definitions - tuple

- A **tuple** is an ordered set of values (enclosed in angled brackets ' $\langle \dots \rangle$ ')
- Each value is derived from an appropriate *domain*
- A row in the CUSTOMER relation is a 4-tuple and would consist of four values, for example:
 $\langle 632895, \text{"John Smith"}, \text{"1 Main St, ..."}, \text{"(404) 894-2000"} \rangle$
- A relation is a **set** of such tuples (rows)

11

Formal definitions - domain

- A domain has a logical definition:
 - e.g.: "USA_phone_numbers" are the set of 10 digit phone numbers valid in the U.S.
- A domain also has a data-type or a format defined for it
 - USA_phone_numbers may have a format: (ddd)ddd-dddd where each d is a decimal digit.
 - dates have various formats such as year, month, date formatted as yyyy-mm-dd, or as dd mm,yyyy etc.

12

Formal definitions – attribute names

- An attribute name typically designates the role played by a domain in a relation:
 - used to interpret the meaning of the data elements corresponding to that attribute
 - e.g.: domain Date may be used to define two attributes named “Invoice-date” and “Payment-date” with different meanings

13

continued

Formal definitions – attribute names

- Example: attribute Cust-name is defined over the domain of character strings of maximum length 25
 - $\text{dom}(\text{Cust-name})$ is `varchar(25)`
- The role these strings play in the CUSTOMER relation is that of the *name of a customer*
- Note: sometimes we also say that Cust-name is `varchar(25)`

14

Basic structure

- Formally, given sets D_1, D_2, \dots, D_n , a **relation** r is a subset of $D_1 \times D_2 \times \dots \times D_n$
- Thus, a relation is a set of tuples (a_1, a_2, \dots, a_n) where each $a_i \in D_i$
 - e.g.: $\text{customer-name} = \{\text{Jones, Smith, Curry, Lindsay}\}$
 $\text{customer-street} = \{\text{Main, North, Park}\}$
 $\text{customer-city} = \{\text{Harrison, Rye, Pittsfield}\}$
 $r = \{ (\text{Jones, Main, Harrison}), (\text{Smith, North, Rye}), (\text{Curry, North, Rye}), (\text{Lindsay, Park, Pittsfield}) \}$
 r is a relation over
 $\text{customer-name} \times \text{customer-street} \times \text{customer-city}$

15

State of relation

- If R is the **name** of the relation and A1, A2, ..., An are the **attributes** of the relation, then
- $r(R)$ is a specific **state** (or "value" or "population" or "extension") of relation R
 - this is a *set of tuples* (rows)
 - $r(R) = \{t_1, t_2, \dots, t_k\}$ where each t_i is an n-tuple
 - k is the cardinality of the relation
 - $t_i = \langle v_1, v_2, \dots, v_n \rangle$ where each v_j *element-of* $\text{dom}(A_j)$

16

Populated database state

- Each *relation* will have many tuples in its current relation state
- The *relational database state* is a union of all the individual relation states
- Whenever the database is changed, a new state arises

17

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	658 Wess, Houston, TX	M	40000	888665555	3
Alice	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellare, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	NULL	1

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

Dnumber	Location
1	Houston
4	Stafford
5	Bellare
5	Sugarland
5	Houston

WORKS_ON

Essn	Prcj	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellare	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
333445555	Alice	F	1986-04-05	Daughter
999887777	Theodore	M	1983-10-25	Son

example

18

Definition summary

Informal terms		Formal terms
Table		Relation
Column header		Attribute
All possible (allowed) column values		Domain
Row		Tuple
Table definition		Schema of a relation
Populated table		State of the relation

19

Characteristics of relations

- Ordering of tuples in a relation $r(R)$:
 - tuples are *not considered to be ordered*, even though they appear to be in the tabular form
- Ordering of attributes in a relation schema R (and of values within each tuple):
 - we will consider the attributes in $R(A_1, A_2, \dots, A_n)$ and the values in $t=\langle v_1, v_2, \dots, v_n \rangle$ to be ordered

20

Example

STUDENT

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21

Name	Ssn	Home_phone	Address	Office_phone	Age	Gpa
Benjamin Bayer	305-61-2435	373-1616	2918 Bluebonnet Lane	NULL	19	3.21
Chung-cha Kim	381-62-1245	375-4409	125 Kirby Road	NULL	18	2.89
Dick Davidson	422-11-2320	NULL	3452 Elgin Road	749-1253	25	3.53
Rohan Panchal	489-22-1100	376-9821	265 Lark Lane	749-6492	28	3.93
Barbara Benson	533-69-1238	839-8461	7384 Fontana Lane	NULL	19	3.25

Same states
(but with different order of tuples)

21

continued

Characteristics of relations

- Values in a tuple:
 - all values are considered atomic (indivisible).
 - each value in a tuple must be from the domain of the attribute for that column
 - if tuple $t = \langle v_1, v_2, \dots, v_n \rangle$ is a tuple (row) in the relation state r of $R(A_1, A_2, \dots, A_n)$ then each v_i must be a value from $dom(A_i)$
 - a special **null** value is used to represent values that are unknown or inapplicable to certain tuples.

22

continued

Characteristics of relations

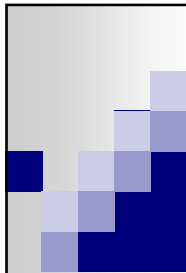
- Notation:
 - We refer to **component values** of a tuple t by:
 - $t[A_i]$ or $t.A_i$
 - this is the value v_i of attribute A_i for tuple t
 - e.g. STUDENT.Name
 - Similarly, $t[A_u, A_v, \dots, A_w]$ refers to the sub-tuple of t containing the values of attributes A_u, A_v, \dots, A_w , respectively in t

23

continued

Characteristics of relations

24



Relational database schema

25

Relational database schema

- Relational database – a set of relations
- Relational database **schema** - a set S of relation schemas that belong to the same database
 - S is the name of the whole **database schema**
 - $S = \{R_1, R_2, \dots, R_n\}$
 - R_1, R_2, \dots, R_n are the names of the individual **relation schemas** within the database S

26

Relational integrity constraints

- Constraints are **conditions** that must hold on **all** valid relation states
- Three *main types* of constraints
 - **key** constraints
 - **entity integrity** constraints
 - **referential integrity** constraints
- An implicit constraint is the **domain** constraint
 - every value in a tuple must be from the *domain of its attribute* (or it could be **null**, if allowed for that attribute)

27

Key constraints

- **Superkey** of R: a set of attributes SK of R so that no two tuples in any valid relation state $r(R)$ will have the same value for SK
 - i.e. for any distinct tuples t_1 and t_2 in $r(R)$
 $t_1[SK] \neq t_2[SK]$
 - this condition must hold in *any valid state* $r(R)$
- **Key** of R is a "minimal" superkey
 - a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

28

continued

Key constraints

- Example: consider the CAR relation schema:
CAR(State, Reg#, SerialNo, Make, Model, Year)
 - CAR has two keys:
 - Key1 = {State, Reg#}
 - Key2 = {SerialNo}
 - Both are also superkeys of CAR
 - {SerialNo, Make} is a superkey but *not* a key.
- In general:
 - any *key* is a *superkey* (but not vice versa)
 - any set of attributes that *includes a key* is a *superkey*
 - a *minimal* superkey is a *key*

29

continued

Key constraints

- If a relation has several **candidate keys**, one is chosen arbitrarily to be the **primary key**
 - primary key attributes are underlined
 - Example: CAR relation schema
CAR(State, Reg#, SerialNo, Make, Model, Year)
we chose SerialNo as the primary key
- The primary key value is used to *uniquely identify* each tuple in a relation
 - provides the tuple identity
 - can be used to *reference* the tuple from another tuple

30

Key constraints

- Selecting primary key:
 - choose as primary key the smallest of the candidate keys (in terms of size)
 - not always applicable – choice is sometimes subjective
- Displaying relational schema
 - each relation schema can be displayed as a row of attribute names
 - relation name is written above the attribute names
 - the primary key attribute(s) will be underlined

COMPANY database schema

EMPLOYEE

Fname	Minit	Lname	Ssn	Bdate	Address	Sex	Salary	Super_ssn	Dno
-------	-------	-------	-----	-------	---------	-----	--------	-----------	-----

DEPARTMENT

Dname	Dnumber	Mgr_ssn	Mgr_start_date
-------	---------	---------	----------------

DEPT_LOCATIONS

Dnumber	Dlocation
---------	-----------

PROJECT

Pname	Pnumber	Plocation	Dnum
-------	---------	-----------	------

WORKS_ON

Essn	Pno	Hours
------	-----	-------

DEPENDENT

Essn	Dependent_name	Sex	Bdate	Relationship
------	----------------	-----	-------	--------------

Entity integrity

- **Primary key attributes** PK of each relation schema R cannot have null values:
 - $t[PK] \neq \text{null}$ for any tuple t in $r(R)$
 - because primary key values are used to *identify* the individual tuples
 - If PK has several attributes, null is not allowed in any of these attributes
- Note: Other attributes of R may be constrained to disallow null values, even though they are not members of the primary key.

Referential integrity

- A constraint involving **two** relations, used to specify a **relationship** among tuples in two relations (**referencing** and **referenced** relation)
- Tuples in the **referencing** relation R1 have attributes FK (called **foreign key** attributes) that reference the primary key attributes PK of the **referenced** relation R2.
 - a tuple t1 in R1 is said to **reference** a tuple t2 in R2 if $t1[FK] = t2[PK]$

34

continued

Referential integrity

- Value in the foreign key column (or columns) FK of the **referencing relation R1** can be **either**:
 - (1) a value of an existing primary key value of a corresponding primary key PK in the **referenced relation R2**, or
 - (2) a **null** (in this case, the FK in R1 should **not** be a part of its own primary key)
- Informally, the referential integrity constraint states that a tuple in one relation that refers to another relation must relate to an existing tuple in that relation

35

Example of a foreign key

EMPLOYEE

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800	20
7521	ALLEN	SALESMAN	7698	20-FEB-81	1600	30
7566	WARD	SALESMAN	7698	22-FEB-81	1250	30
7654	JONES	MANAGER	7839	02-APR-81	2975	20
7787	MARTIN	SALESMAN	7696	28-SEP-8	1250	30

DEPARTMENT

DEPTNO	DNAME	LOCATION
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

Foreign Key

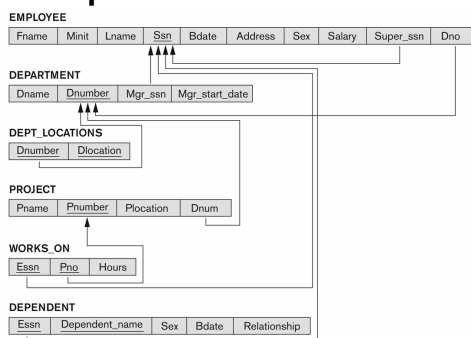
36

Displaying a relational database schema and its constraints

- Each relation schema can be displayed as a row of attribute names
- The name of the relation is written above the attribute names
- The primary key attribute (or attributes) will be underlined
- A foreign key constraints are displayed as a directed arc from the foreign key attributes to the referenced table
 - can also point the primary key of the referenced relation for clarity

37

Example



38

Other types of constraints

- Semantic integrity constraints:
 - based on application semantics and cannot be expressed by the model per se
 - e.g.: "the max. no. of hours per employee for all projects he or she works on is 56 hrs per week"
- A **constraint specification** language may have to be used to express these
 - SQL-99 allows triggers and **ASSERTIONS** to express for some of these

39

Update operations on relations

- Basic operations for changing the database:
 - INSERT a new tuple in a relation
 - DELETE an existing tuple from a relation
 - MODIFY an attribute of an existing tuple
- Integrity constraints should not be violated by the update operations
- Updates may **propagate** to cause other updates automatically. This may be necessary to maintain integrity constraints.

40

continued

Update operations on relations

- In case of integrity violation, several actions or options can be taken:
 - cancel the operation that causes the violation (RESTRICT or REJECT)
 - perform the operation but inform the user of the violation
 - trigger additional updates so the violation is corrected (CASCADE option, SET NULL option)
 - execute a user-specified error-correction routine

41

Possible violations: INSERT

- INSERT may violate any of the constraints:
 - Domain** constraint: if one of the attribute values provided for the new tuple is not of the specified attribute domain
 - Key** constraint: if the value of a key attribute in the new tuple already exists in another tuple in the relation
 - Entity** integrity: if the primary key value is null in the new tuple
 - Referential** integrity: if a foreign key value in the new tuple references a primary key value that does not exist in the referenced relation

42

Possible violations: DELETE

- DELETE may violate only referential integrity:
 - If the primary key value of the tuple being deleted is referenced from other tuples in the database
 - can be remedied by several actions:
 - RESTRICT option: reject the deletion
 - CASCADE option: propagate the new primary key value into the foreign keys of the referencing tuples
 - SET NULL option: set the foreign keys of the referencing tuples to NULL
 - one of the above options must be specified during database design for each foreign key constraint

43

Possible violations: MODIFY

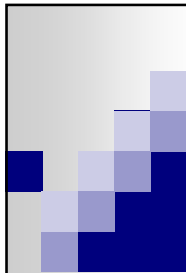
- Depends on the type of attribute being MODIFIED (UPDATED)
 - updating the primary key (PK)
 - similar to a DELETE followed by an INSERT
 - need to specify similar options to DELETE
 - updating a foreign key (FK):
 - may violate referential integrity
 - updating an ordinary attribute (neither PK nor FK):
 - can only violate domain constraints and NOT NULL constraint on an attribute being modified

44

Summary so far

- Relational model concepts
 - definitions and characteristics of relations
- Relational Database Schemas
 - relational model constraints
 - domain constraints
 - key constraints
 - entity integrity
 - referential integrity
- Relational update operations
 - dealing with constraint violations

45



Mapping ER to relational model

46

From ER to relational - overview

- A conceptual schema, expressed as an ER diagram, can be transformed into a set of relational schemas that can be stored by a DBMS
- This is the **logical design** step of **database design**
- Transforming ER schemas to relational schemas is part of the functionality of many tools based on the ER model

47

Algorithm

- **ER-to-Relational mapping**
 - Step 1: Mapping of Regular Entity Types
 - Step 2: Mapping of Weak Entity Types
 - Step 3: Mapping of Binary 1:1 Relation Types
 - Step 4: Mapping of Binary 1:N Relationship Types.
 - Step 5: Mapping of Binary M:N Relationship Types.
 - Step 6: Mapping of Multivalued attributes.
 - Step 7: Mapping of N-ary Relationship Types.
- **Mapping EER model constructs to relations**
 - Step 8: Options for Mapping Specialization or Generalization.
 - Step 9: Mapping of Union Types (Categories).

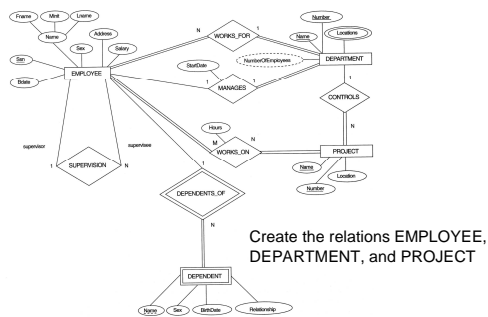
48

1. Mapping of regular entity types

- For each regular (strong) entity type E in the ER schema, create a relation R that includes all the simple attributes of E.
- Choose one of the key attributes of E as the primary key for R.
- If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.
- Note: we allow the names of attributes for relations to be different than the names of the corresponding attributes for entities

49

Example



50

Result of step 1

EMPLOYEE (FNAME, MINIT, LNAME, SSN,
BDATE, ADDRESS, SEX, SALARY)

DEPARTMENT (DNAME, DNUMBER)

PROJECT(PNAME, PNUMBER, PLOCATION)

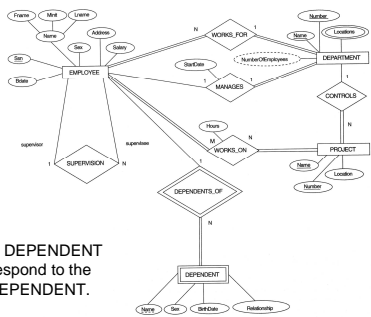
51

2. Mapping of weak entity types

- For each weak entity type W in the ER schema with owner entity type E , create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R
- Also, include as foreign key attributes of R the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s).
- The primary key of R is the *combination* of the primary key(s) of the owner(s) and the partial key of the weak entity type W , if any.

52

Example



Create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT.

53

Result of step 2

- For the weak entity type DEPENDENT, create the relation DEPENDENT

DEPENDENT (ESSN, DEPENDENT_NAME,
SEX, BDATE, RELATIONSHIP)

The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT

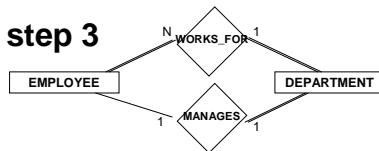
54

3. Mapping of binary 1:1 relations

- For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.
- **Foreign key approach:** Choose one of the relations, say S, and add as a foreign key in S the primary key of T
 - It is better to choose an entity type with total participation in R in the role of S.
- Also include in S all the simple attributes (or simple components of composite attributes) of R

55

Result of step 3



- For relationship type MANAGES, we choose the relation DEPARTMENT

EMPLOYEE (FNAME, MINIT, LNAME, SSN,
BDATE, ADDRESS, SEX, SALARY)

DEPARTMENT (DNAME, DNUMBER,
MGRSSN, MGRSTARTDATE)

56

continued

3. Mapping of binary 1:1 relations

- Alternative approaches
 - **Merged relation option:** merge the two entity types and the relationship into a single relation. This may be appropriate when both participations are total.
 - **Cross-reference or relationship relation option:** set up a third relation R for the purpose of cross-referencing the primary keys of the two relations S and T representing the entity types.

57

4. Mapping of binary 1:N relations

- For each regular binary 1:N relationship type R, identify the relation S that represent the participating entity type at the N-side of the relationship type.
- Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R.
- Include any simple attributes of the 1:N relation type as attributes of S.

58

example

Result of step 4



- e.g. for WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation (and call it DNO)

EMPLOYEE (FNAME, MINIT, LNAME, SSN, BDATE, ADDRESS, SEX, SALARY, SUPERSSN, DNO) ← from SUPERVISION

DEPARTMENT (DNAME, DNUMBER, MGRSSN, MGRSTARTDATE)

PROJECT (PNAME, PNUMBER, PLOCATION, DNUM) ← from CONTROLS

59

5. Mapping of binary M:N relations

- For each regular binary M:N relationship type R, create a new relation S to represent R
 - Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S.
 - Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S.

60

5. Mapping of binary M:N relations

- M:N relationship type WORKS_ON from the ER diagram is mapped by creating a relation WORKS_ON in the relational database schema
 - The primary keys of the PROJECT and EMPLOYEE relations are included as foreign keys in WORKS_ON and renamed PNO and ESSN, respectively.
 - Attribute HOURS in WORKS_ON represents the HOURS attribute of the relation type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes (ESSN, PNO)

WORKS_ON(ESSN, PNO, HOURS)

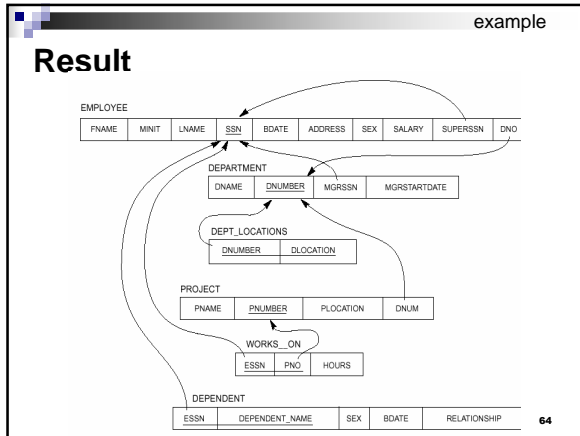
6: Mapping of multi-valued attributes

- For each multi-valued attribute A, create a new relation R.
- This relation R will include an attribute corresponding to A, plus the primary key attribute K (as a foreign key in R) of the relation that represents the entity type that has A as an attribute.
- Primary key of R is the combination of A and K.
- If the multi-valued attribute is composite, we include its simple components.

6. Mapping of multi-valued attributes

- The relation DEPT_LOCATIONS is created.
 - The attribute DLOCATION represents the multi-valued attribute LOCATIONS of DEPARTMENT,
 - DNUMBER represents the primary key of the DEPARTMENT relation.

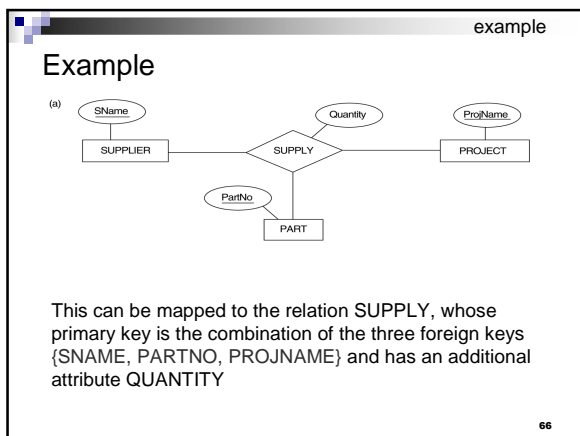
DEPT_LOCATIONS(DNUMBER, DLOCATION)



7. Mapping of n-ary relations

- For each n-ary relationship type R, where $n > 2$, create a new relationship S to represent R.
- Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types.
- Also include any simple attributes of the n-ary relationship type (or simple components of composite attributes) as attributes of S.

65



Summary of mapping constructs

ER Model

Entity type
1:1 or 1:N relationship type
M:N relationship type
n-ary relationship type
Simple attribute
Composite attribute
Multivalued attribute
Value set
Key attribute

Relational Model

"Entity" relation
Foreign key (or "relationship" relation)
"Relationship" relation and 2 FKs
"Relationship" relation and *n* FKs
Attribute
Set of simple component attributes
Relation and foreign key
Domain
Primary (or secondary) key

67

Mapping EER to relations

- Mapping specialisations (generalisations)
- Mapping union types

68

8. Mapping specialisations

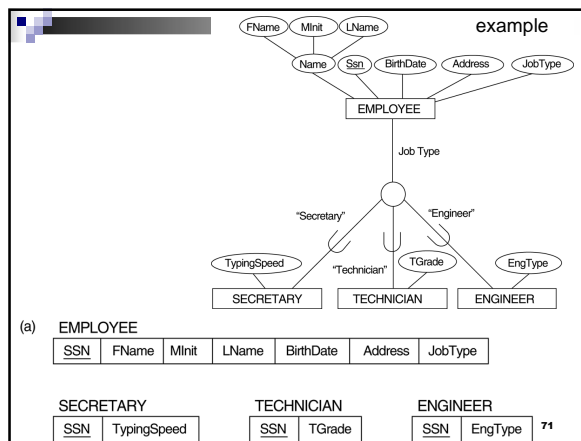
- Convert each specialisation with *m* subclasses $\{S_1, S_2, \dots, S_m\}$ and generalized superclass *C*, where the attributes of *C* are $\{k, a_1, \dots, a_n\}$ and *k* is the (primary) key, into relational schemas using one of the four following options:
 - 8A: Multiple relations – superclass and subclasses
 - 8B: Multiple relations – subclass relations only
 - 8C: Single relation with one type attribute
 - 8D: Single relation with multiple type attributes

69

8. Mapping specialisation (A)

- 8A: Multiple relations - superclass and subclasses
 - create a relation L for C with attributes $\text{Attr}(L) = \{k, a_1, \dots, a_n\}$ and $\text{PK}(L) = k$.
 - create a relation L_i for each subclass S_i , $1 < i < m$, with the attributes $\text{Attr}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$ and $\text{PK}(L_i) = k$.
 - this option works for any specialisation (total or partial, disjoint or over-lapping).

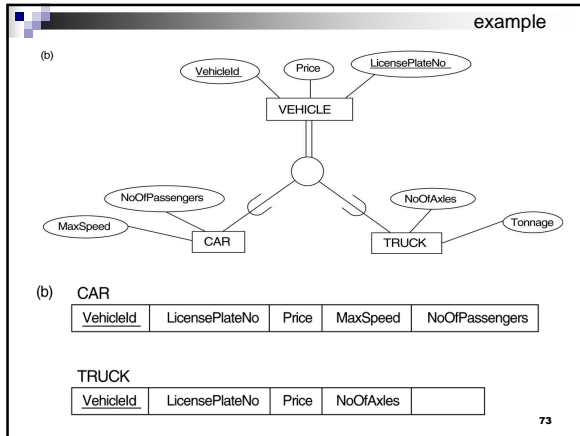
70



8. Mapping specialization (B)

- 8B: Multiple relations - subclass relations only
 - create a relation L_i for each subclass S_i , $1 < i < m$, with the attributes $\text{Attr}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \dots, a_n\}$ and $\text{PK}(L_i) = k$.
 - this option only works for a specialization whose subclasses are total (every entity in the superclass must belong to (at least) one of the subclasses).

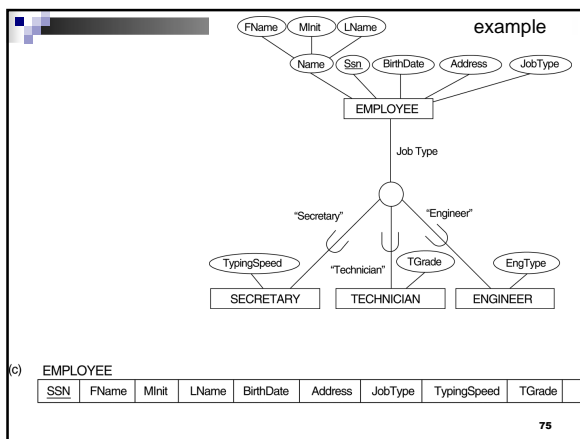
72



8. Mapping specialization (C)

- 8C: Single relation with one type attribute
 - create a single relation L with attributes $Attr(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t\}$ and $PK(L) = k$.
 - the attribute t is called a type (or discriminating) attribute that indicates the subclass to which each tuple belongs

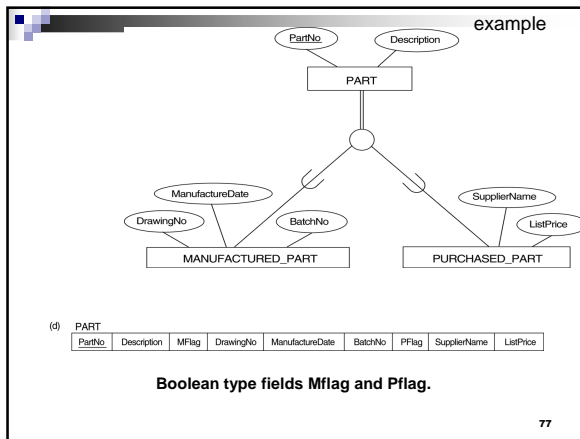
74



8. Mapping specialization (D)

- **8D: Single relation with multiple type attributes**
 - Create a single relation schema L with attributes $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t_1, t_2, \dots, t_m\}$ and $\text{PK}(L) = k$.
 - Each $t_i, 1 < i < m$, is a Boolean type attribute indicating whether a tuple belongs to the subclass S_i .
 - Good for an overlapping (non-disjoint) specialization

76

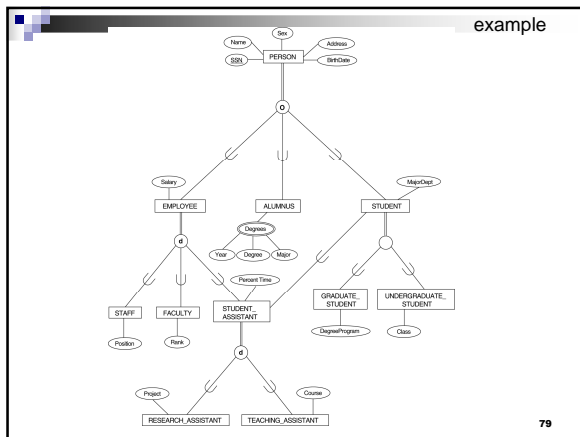


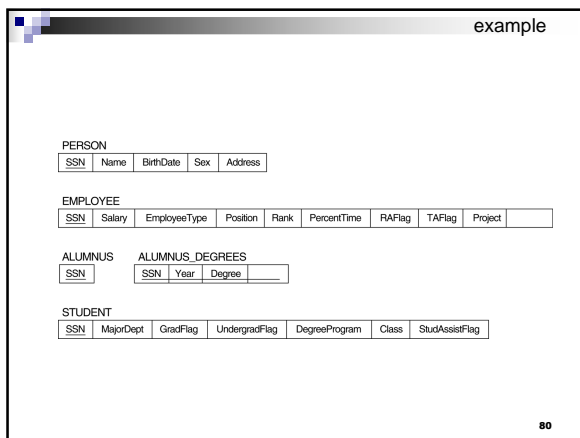
77

Mapping of shared subclasses

- A shared subclass, such as STUDENT_ASSISTANT, is a subclass of several classes, indicating multiple inheritance. These classes must all have the same key attribute; otherwise, the shared subclass would be modeled as a union type (step 9).
- We can apply any of the options discussed in Step 8 to a shared subclass, subject to the restriction discussed in Step 8 of the mapping algorithm..

78

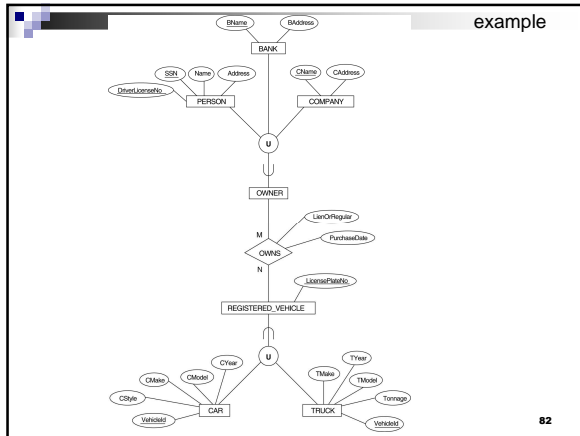


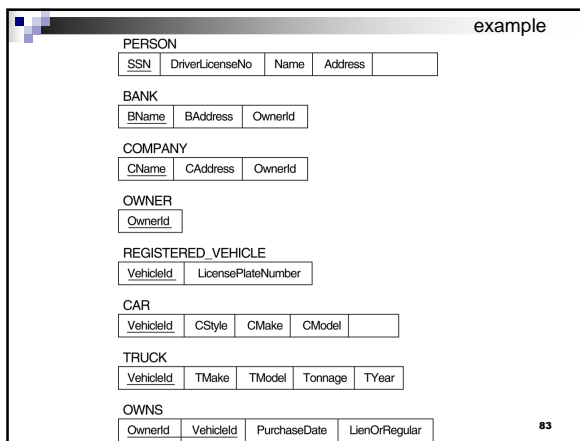


9. Mapping union types

- For mapping a union type whose defining superclass have different keys, it is customary to specify a new key attribute, called a surrogate key, when creating a relation to correspond to the category.
- E.g. we can create a relation OWNER to correspond to the OWNER category and include any attributes of the union in this relation. The primary key of the OWNER relation is the surrogate key, which we called OwnerId.

81







Tools for data modelling

- Oracle's Designer
- Rational Rose (IBM)
- ERwin (Computer Associates)
- DataArchitect (Kompany)
- MySQL Workbench 5.0 (DBDesigner 4) (SUN, open source)
- see http://www.databaseanswers.com/modelling_tools.htm

85

Summary

■ ER-to-Relational model mapping

- Step 1: Mapping of Regular Entity Types
- Step 2: Mapping of Weak Entity Types
- Step 3: Mapping of Binary 1:1 Relation Types
- Step 4: Mapping of Binary 1:N Relationship Types.
- Step 5: Mapping of Binary M:N Relationship Types.
- Step 6: Mapping of Multivalued attributes.
- Step 7: Mapping of N-ary Relationship Types.

■ EER to Relations

- Step 8: Options for Mapping Specialization or Generalization
- Step 9: Mapping of Union Types

86

Reading for this lecture

- Main text: Chapters 5 and 7 in [Elmasri & Navathe]
- Also, Chapter 3 in [Connolly & Begg]
- Solve tutorial exercises

87
