

COMP67321
**Data modelling using
the ER model**

Goran Nenadic

School of Computer Science
University of Manchester

1

Aims

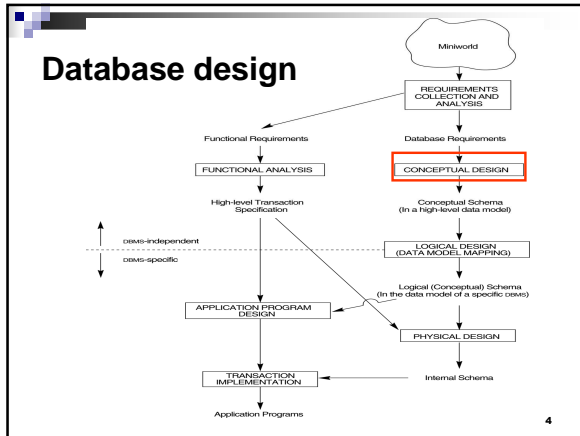
- Understand the data modelling process
- Learn concepts, semantics and notations of the Entity-Relationship (ER) model
- Understand DB design using ER

2

Overview of DB design process

- Two main activities:
 - data (and database) design
 - applications design
- Focus in this lecture on database design
 - designing the conceptual schema for a database application
- Applications design focuses on the programs and interfaces that access the database
 - generally considered part of software engineering

3

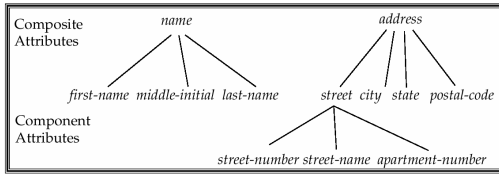


- ## Entity-Relationship (ER) model
- Modelling requirements using non-technical but free of ambiguities approach
 - Top-down approach
 - Identify *entities* and *relationships* between them that must be presented in the DB
 - Different notations
 - various diagrams
 - UML (Unified modelling language)
- 5

- ## ER model: plan
- Entities, entity sets
 - Attributes, keys
 - Relationships, relationship sets
 - Cardinalities and participation
 - ER diagrams, UML
 - Design issues
- 6

Attribute types

- *simple* and *composite* attributes.



- *single-valued* and *multi-valued* attributes
 - e.g. multi-valued attribute: *phone-numbers*
- *derived* attributes, computed from other attributes
 - e.g. *age*, given date of birth

10

Key attributes

- An attribute of an entity type for which each entity must have a unique value is called a key attribute of the entity type.
 - e.g. SSN of EMPLOYEE
 - loan-no for LOAN
 - customer-id for CUSTOMER

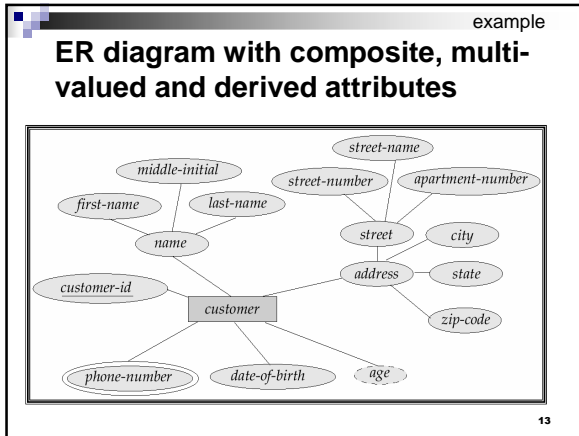
11

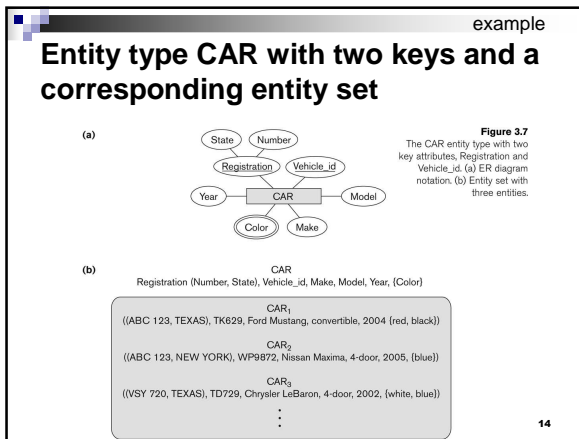
continued

Key attributes

- A key attribute may be composite
 - VehicleTagNumber is a key of the CAR entity type with components (Number, State).
- An entity type may have more than one key.
 - The CAR entity type may have two keys:
 - VehicleIdentificationNumber
 - VehicleTagNumber (license plate number)

12





Relationships

- A relationship relates (or links) two or more distinct entities with a specific meaning.
 - e.g. EMPLOYEE John Smith *works on* the ProductX PROJECT.
 - or EMPLOYEE Franklin Wong *manages* the Research DEPARTMENT.
- Relationships of the same type are grouped or typed into a relationship type.
 - e.g. the WORKS_ON relationship type in which EMPLOYEES and PROJECTS participate,
 - the MANAGES relationship type in which EMPLOYEES and DEPARTMENTS participate.

15

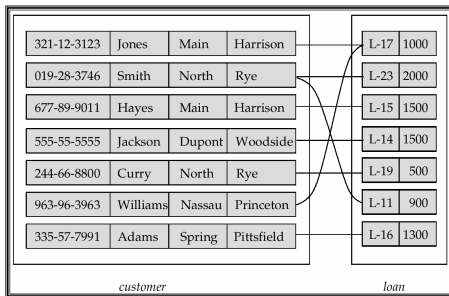
Relationship types and sets

- Relationship type
 - the schema description of a relationship
 - identifies the relationship name and the participating entity types
 - also identifies certain relationship constraints
- Relationship set
 - current set of relationship **instances** represented in the database
 - = current **state** of a relationship type

16

example

Relationship set *borrower*



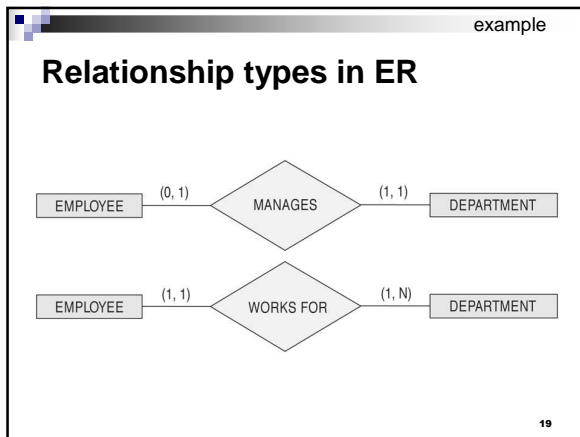
17

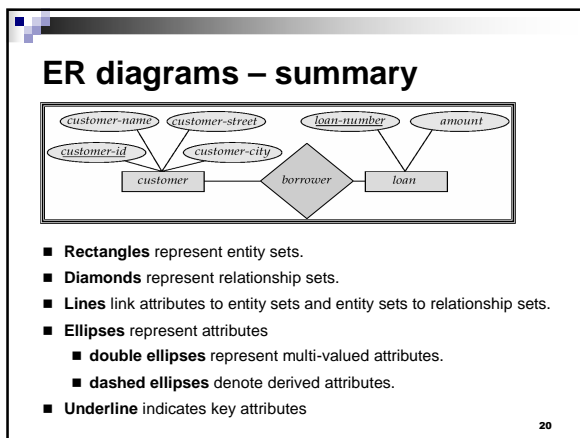
continued

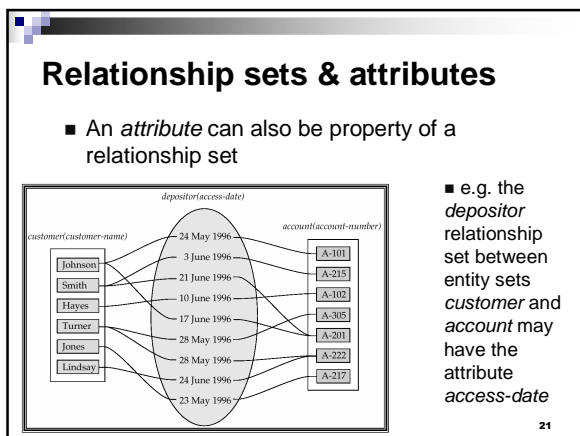
Relationship type and sets

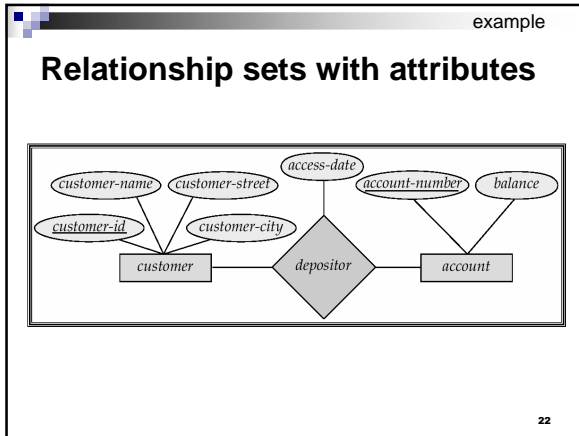
- Each instance in the set relates individual participating entities – one from each participating entity type
- In ER diagrams, we represent the *relationship type* as follows:
 - diamond-shaped box is used to display a relationship type
 - connected to the participating entity types via straight lines

18









Degree of a relationship set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are *binary* (or degree two).
 - most relationship sets in a DB system are binary
- Relationship sets may involve more than two entity sets.
 - e.g. ternary relations

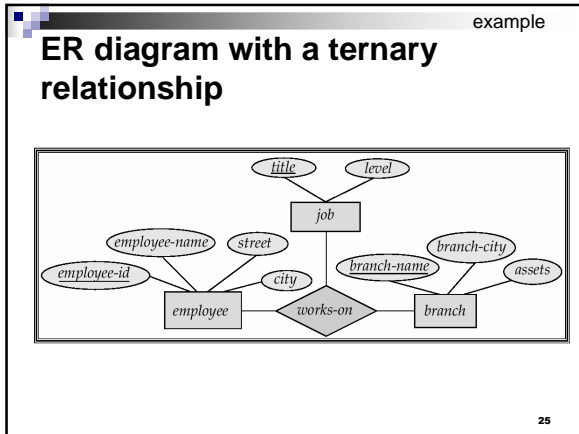
23

example

Degree of a relationship set

- Example of ternary relationship
 - ☞ Employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. So, there is a ternary relationship set between entity sets *employee*, *job* and *branch*

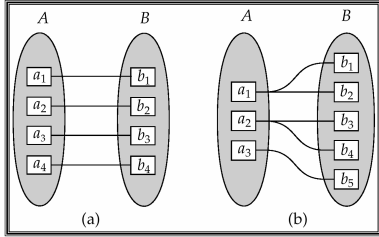
24



- ## Constraints on relationships
- Cardinality (specifies *maximum* participation)
 - One-to-one (1:1)
 - One-to-many (1:N) or Many-to-one (N:1)
 - Many-to-many (M:N)
 - Participation (specifies *minimum* participation)
 - aka: Existence Dependency Constraint
 - partial (optional participation, not existence-dependent)
 - total (mandatory participation, existence-dependent)
- 26

- ## Mapping cardinalities
- Express the number of entities to which another entity can be associated via a relationship set.
 - Most useful in describing binary relationship sets.
 - For a binary relationship set, the mapping cardinality must be one of the following:
 - one to one (1:1), one to many (1:*),**
 - many to one (*:1), many to many (*:*)**
- 27

Mapping cardinalities

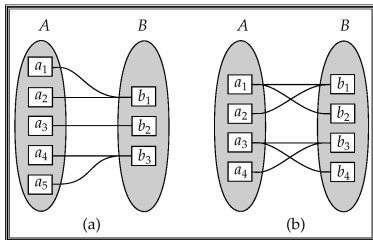


One to one

One to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping cardinalities



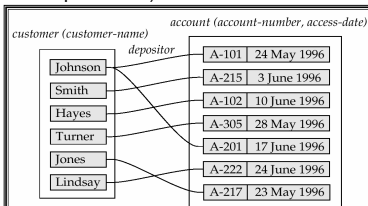
Many to one

Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set

Mapping cardinalities

- Relationship from account to customer is many to one, or equivalently, customer to account is one to many
- if each account can have only one customer, we can make *access-date* an attribute of account (instead of a relationship attribute)



ER diagrams and cardinality

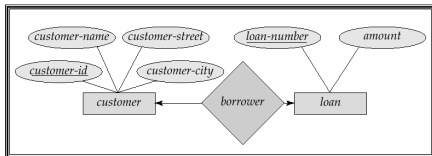
- We express cardinality constraints by drawing either a directed line (\rightarrow), signifying "one," or an undirected line (—), signifying "many," between the relationship set and the entity set.
- Alternatively, we can draw a line and represent cardinality constraints using numbers (1:1, 1:*, etc).

31

example

Example: one-to-one

- A customer is associated with at most one loan via the relationship *borrower*, and a loan is associated with at most one customer via *borrower*

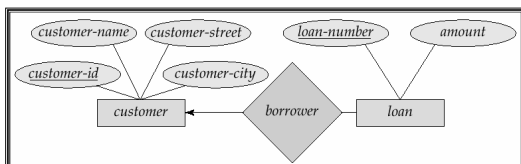


32

example

Example: one-to-many

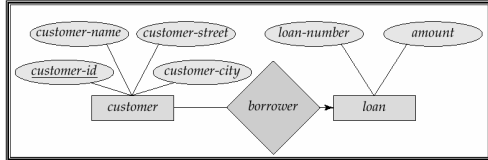
- A loan is associated with at most one customer via *borrower* relationship, a customer is associated with several (including 0) loans via *borrower*



33

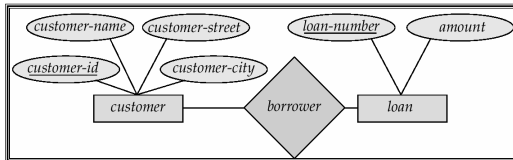
Example: many-to-one

- A loan is associated with several (including 0) customers (e.g. joint loans) via *borrower*, a customer is associated with at most one loan via *borrower*



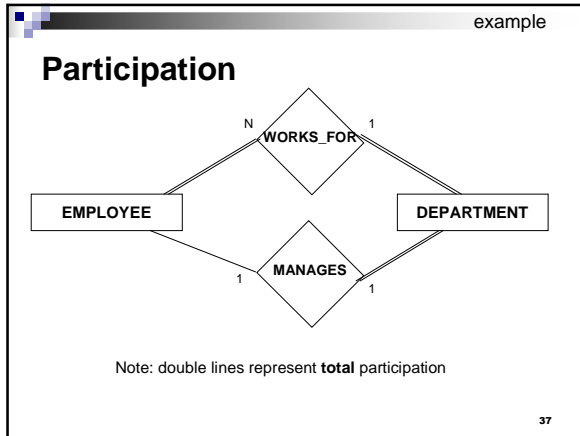
Example: many-to-many

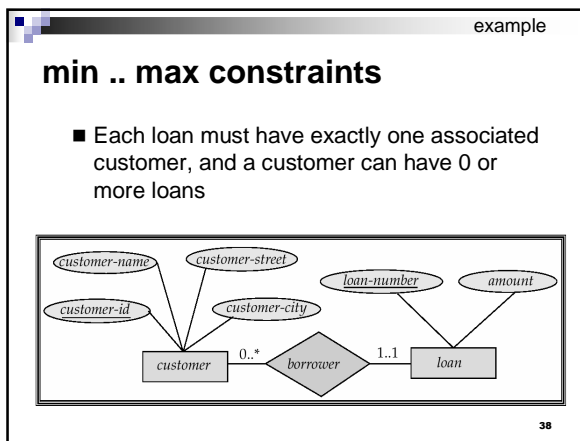
- A customer is associated with several (possibly 0) loans via *borrower*, and a loan is associated with several (possibly 0) customers via *borrower*

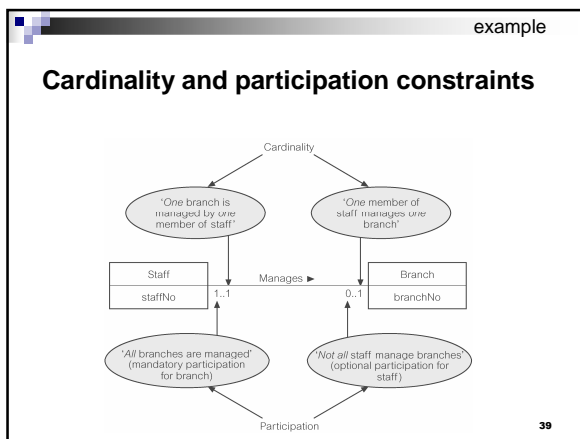


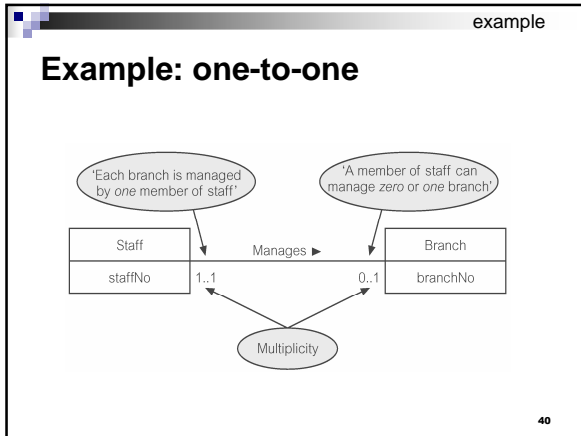
Participation constraints

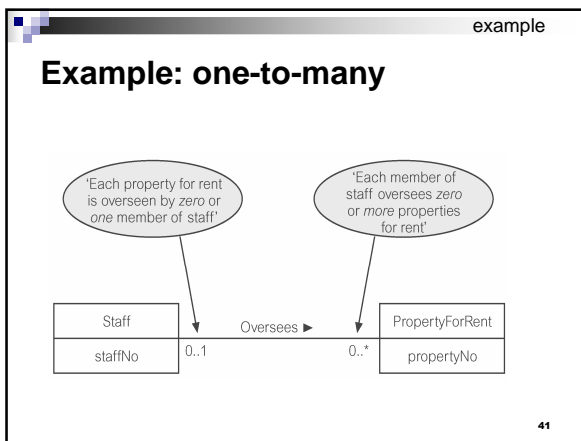
- Cardinality
 - describes maximum number of possible relationship occurrences for an entity participating in a given relationship type.
- Participation
 - determines whether all or only some entity occurrences participate in a relationship
 - total = every entity needs to be 'related' (e.g. EMPLOYEE works_in DEPARTMENT)
 - partial = some entities are involved (e.g. EMPLOYEE manages DEPARTMENT)

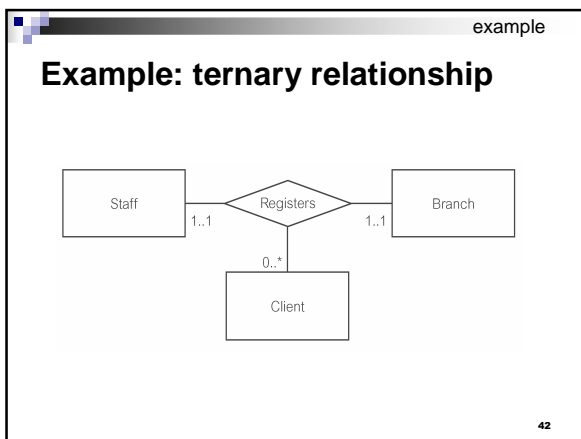












Attributes of relationship types

- A relationship type can have attributes:
 - e.g. HoursPerWeek of WORKS_ON
 - its value for each relationship instance describes the number of hours per week that an EMPLOYEE works on a PROJECT.
 - a value of HoursPerWeek depends on a particular (employee, project) combination
 - Most relationship attributes are used with M:N relationships
 - in 1:N relationships, they can be transferred to the entity type on the N-side of the relationship

43

Recursive relationship type

- An relationship type with the same participating entity type in distinct roles
- Example: the SUPERVISION relationship
- EMPLOYEE participates twice in two distinct roles:
 - supervisor (or boss) role
 - supervisee (or subordinate) role
- Each relationship instance relates two distinct EMPLOYEE entities:
 - One employee in *supervisor* role
 - One employee in *supervisee* role

44

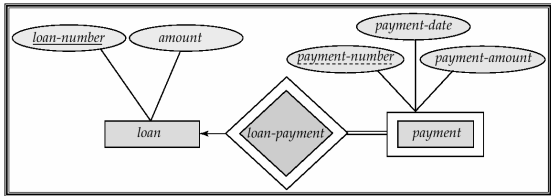
Weak entity sets

- An entity set that does not have a key is referred to as a *weak entity set*.
- The existence of a weak entity set depends on the existence of a *identifying entity set*
 - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
 - identifying relationship is depicted using a double diamond

45

Weak entity sets

- E.g. *payment* cannot exist without a *loan*
- Notations:
 - double rectangles for weak entity sets
 - underline the *discriminator* of a weak entity set with a dashed line.



Weak entity sets

- The *discriminator* (or *partial key*) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity set's discriminator.
- If the existence of entity *x* depends on the existence of entity *y*, then *x* is said to be *existence dependent* on *y*.

Weak entity sets

- Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.
- If *loan-number* were explicitly stored, *payment* could be made a strong entity, but then the relationship between *payment* and *loan* would be duplicated by an implicit relationship defined by the attribute *loan-number* common to *payment* and *loan*

Weak entity types: examples

- We want to model a DEPENDENT entity and relate it to EMPLOYEES in a company
- A dependant is identified by his/her name, *and* the specific EMPLOYEE with whom the dependent is related
 - name of DEPENDENT is the *partial key*
- DEPENDENT is a *weak entity type*
- EMPLOYEE is its identifying entity type via the identifying relationship type DEPENDENT_OF

Weak entity types: examples

- In a university, a *course* is a strong entity and a *course-offering* can be modeled as a *weak entity*
- The discriminator of *course-offering* would be *semester* (including year) and *section-number* (if there is more than one section)
- If we model *course-offering* as a strong entity we would model *course-number* as an attribute. Then the relationship with *course* would be implicit in the *course-number* attribute

Some design issues

- Use of entity sets vs. attributes
Choice mainly depends on the structure of the enterprise being modeled, and on the semantics associated with the attribute in question.
- Use of entity sets vs. relationship sets
Possible guideline is to designate a relationship set to describe an action that occurs between entities

continued

Some design issues

- Attributes or relationships:
 - Manager of DEPARTMENT or relationship MANAGES
 - Works_on of EMPLOYEE or relationship WORKS_ON
 - Department of EMPLOYEE or relationship WORKS_FOR
- In general, more than one relationship type can exist between the same participating entity types
 - MANAGES and WORKS_FOR are distinct relationship types between EMPLOYEE and DEPARTMENT
 - different meanings and different relationship instances.

52

continued

Some design issues

- Binary versus n -ary relationship sets:
Although it is possible to replace any nonbinary (n -ary, for $n > 2$) relationship set by a number of distinct binary relationship sets, a n -ary relationship set shows more clearly that several entities participate in a single relationship.
- Placement of relationship attributes

53

ER notations

- Different notations used
 - check in the textbooks
- UML diagrams are popular
 - used in database design and object-oriented software design
 - UML has many other types of diagrams for software design

54

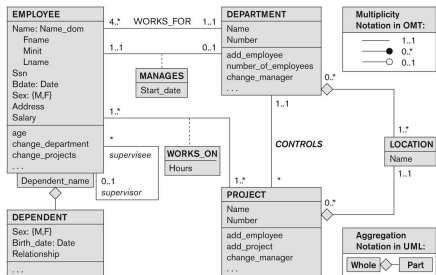
ER models in UML

- Represent classes (similar to entity types) as large rounded boxes with three sections:
 - Top section includes entity type (class) name
 - Second section includes attributes
 - Third section includes class operations (operations are not in basic ER model)
- Relationships (called associations) represented as lines connecting the classes
 - Other UML terminology also differs from ER terminology

55

UML class diagram for COMPANY database schema

Figure 3.16 The COMPANY conceptual schema in UML class diagram notation.



56

Data modeling tools

- A number of popular tools that cover conceptual modeling and mapping into relational schema design.
 - Examples: ERWin, S- Designer (Enterprise Application Suite), ER- Studio, DBDesigner (free), DIA (free)
- POSITIVES:
 - Serves as documentation of application requirements, easy user interface - mostly graphics editor support
- NEGATIVES:
 - Most tools lack a proper distinct notation for relationships with relationship attributes
 - Mostly represent a relational design in a diagrammatic form rather than a conceptual ER-based design

57

example

Data modeling tools

COMPANY	TOOL	FUNCTIONALITY
Embarcadero Technologies	ER Studio	Database Modeling in ER and IDEF1X
	DB Artisan	Database administration, space and security management
Oracle	Developer 2000/Designer 2000	Database modeling, application development
Popkin Software	System Architect 2001	Data modeling, object modeling, process modeling, structured analysis/design
Platinum (Computer Associates)	Enterprise Modeling Suite: Erwin, BPWin, Paradigm Plus	Data, process, and business component modeling
Persistence Inc.	Pwrtier	Mapping from C-O to relational model
Rational (IBM)	Rational Rose	UML Modeling & application generation in C++/JAVA
Resolution Ltd.	Xcase	Conceptual modeling up to code maintenance
Sybase	Enterprise Application Suite	Data modeling, business logic modeling
Visio	Visio Enterprise	Data modeling, design/reengineering Visual Basic/C++

58

Summary

- ER model concepts: entities, attributes, relationships
- Constraints in the ER model
 - cardinality and participation
- ER diagrams
 - various notations
 - UML class diagrams

59

Example: COMPANY database

- We need to create a database schema design based on the following (simplified) requirements:
 - The company is organized into DEPARTMENTS. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
 - Each department *controls* a number of PROJECTS. Each project has a unique name, unique number and is located at a single location.

60

Example: COMPANY database

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
 - Each employee *works for* one department but may *work on* several projects.
 - We keep track of the number of hours per week that an employee currently works on each project.
 - We also keep track of the *direct supervisor* of each employee.
- Each employee may *have* a number of DEPENDENTS.
 - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Reading for this lecture

- Main text: Chapter 3 in [Elmasri & Navathe]
- Also, chapters 9, 10 and 11 in [Connolly & Begg]
- See also Chapter 12 in [Elmasri & Navathe]
- Homework: install and try DIA
