Advanced Management of Data

Contact

- Dr. Frank Seifert
- Email: fsei@cs.tu-chemnitz.de
- Office 1/336a (Consultation only by appointment)

Practice Lessons

Practice Lessons

- Monday, 11:30 - 13:00, 1/B006
- Tuesday, 9:15 - 10:45, 1/B006

Start

- next week (21 October 2019)

Contact

- Daniel Richter
- Email: daniel.richter@informatik.tu-chemnitz.de
- Office 1/336d (Consultation only by appointment)

Documents

- You can find the slides of the lecture and practice lessons on the website of the chair Datenverwaltungssysteme

  https://www.tu-chemnitz.de/informatik/DVS/lehre/vorlesungen.php

- slides may be updated during semester

Exam & Study Regulations

Exam

- written, 90 minutes

Study Regulations

- at the moment „Advanced Management of Data“ is only contained in study regulations of course „Automotive Software Engineering 2016“
- for all other students the module „Datenbanken und Objektorientierung“ will be recognized for „Advanced Management of Data“
    - you have to apply individually after the exam by filling in the form „Antrag auf Fachsemestereinstufung/Anrechnung von Prüfungsleistungen“ (Student Service Point): https://www.tu-chemnitz.de/studentenservice/stusek/formulare.php

Requirements

Actually, you should have a background in fundamental database concepts and technology, including

- Semantic data modelling
- the relational data model and languages (relational Algebra, SQL)
    - we offer special slides for self-studying, which cover the most important aspects
- Normalisation
- Concurrency and Transaction Management
- Query Processing
- Physical data modeling and indexing

Importance of Databases

- database industry is valued >€50 billion annually
- databases are basis of information systems
- databases changed the way of operating many businesses fundamentally
- growing importance caused by significant developments in
  - hardware capability
  - the emergence of the Internet and mobile communications
  - e-Commerce
  - Business Intelligence / Big Data

Lecture

Objective

- most commercially relevant database systems are based on the relational paradigm, which does often not adequately address many of these new challenges
- we will name the problems of relational systems
- we will get to know
  - special applications and extensions of relational database systems
  - learn different and new paradigms of data management

Contents

- Repetition of fundamental Database Concepts and Terminology
  - Semantic Data Modeling using the Entity-Relationship Model and UML
  - Logical Data Modeling using the Relational Model (by self-studying)
- Extensions of SQL
- Object-Relational Database Concepts
- Distributed Database Concepts
- NoSQL and Big Data Storage Systems

Literature

- Thomas Connolly, Carolyn Begg: „Database Systems - A Practical Approach to Design, Implementation, and Management“. Sixth Edition, Pearson.
- Ramez Elmasri & Shamkant B. Navathe: „Database Systems“. Seventh Edition, Pearson.

- Further Literature will be added later

Advanced Management of Data

Foundations of Databases

Conceptual Database Design

Why do we use databases?

Limitations of a file-based approach

- separation and isolation of data
- duplication of data
- data dependence
- incompatible file formats
- Inflexible queries

Reason

- The definition of the data is embedded in the application programs, rather than being stored separately and independently.
- There is no control over the access and manipulation of data beyond that imposed by the application programs.

Database

Database

- a shared collection of logically related data, designed to meet the information needs of an organization
- the database holds also a description of this data, which is called either
  - system catalog
  - data dictionary
  - metadata

DBMS

DBMS

The Database Management System (DBMS) is a software system that enables users to define, create, maintain, and control access to the database.

Facilities

- definition of the database through a Data Definition Language (DDL)
- insertion, modification and deletion of data from the database through a Data Manipulation Language (DML)
- retrieval of the data through a Query Language (QL)
- providing controlled access to the database

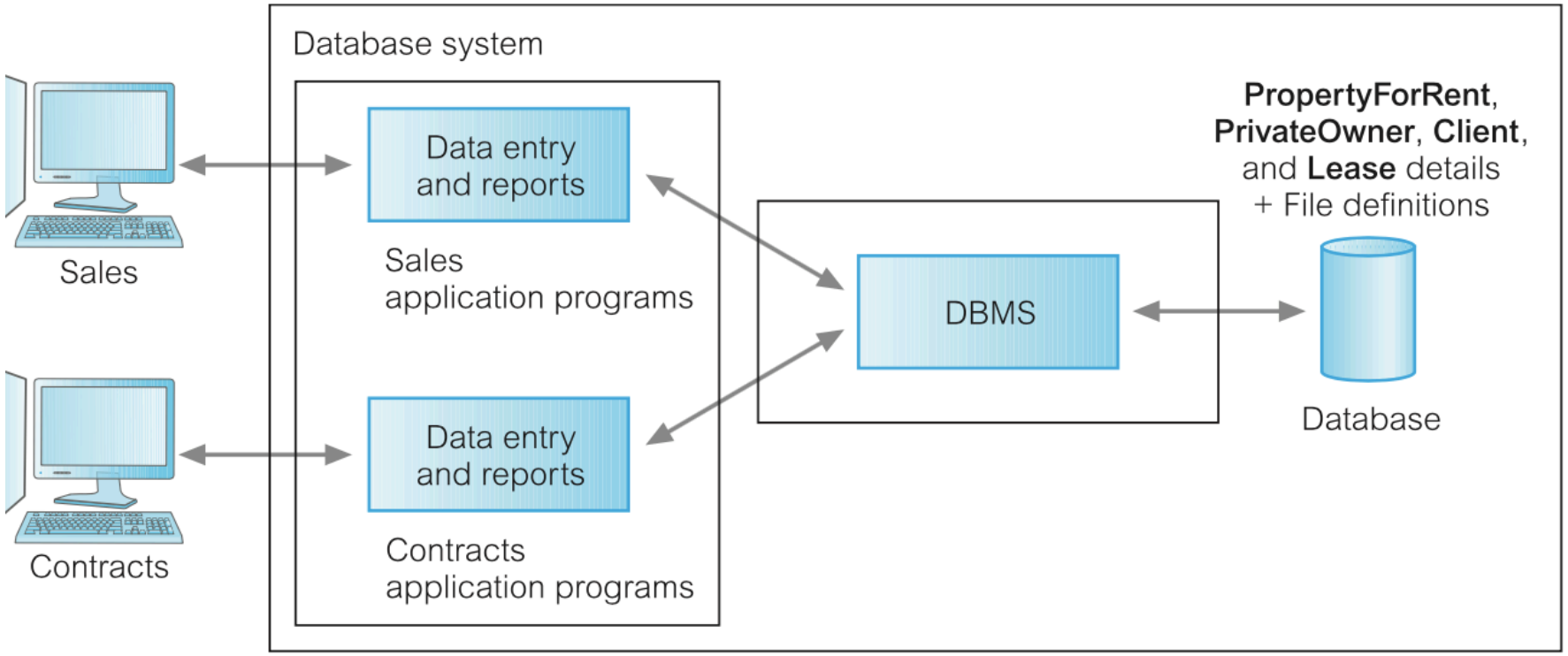
Attention

Usually, the notion „database“ means the whole of database and DBMS.

Application Programs

An Application Program (in the context of a database) is a computer program that interacts with the database by issuing an appropriate request to the DBMS.

Database processing example



[Connolly & Begg]

Views

- are subsets of the database
- allow each user to have an individual view of the database and include only relevant information
- provide a mechanism to customize the appearance of the database
- provide a level of security by excluding data that some users should not see
- can present a consistent, unchanging picture of the database, even if the underlying structure is changed

Advantages of DBMSs (1)

- Control of data redundancy
- Data consistency
- More information from the same amount of data
- Sharing of data
- improved data integrity
- improved security
- enforcement of standards

Advantages of DBMSs (2)

- economy of scale
- balance of conflicting requirements
- improved data accessibility and responsiveness
- increased productivity
- improved maintenance through data independence
- increased concurrency
- improved backup and recovery services

Disadvantages of DBMSs

- Complexity
- Size
- Cost
- additional hardware costs
- Cost of conversions
- Performance
- Impact of failures

Three-Level Architecture

External Level

- The users' view of the database
- describes that part of the database that is relevant to each user

Conceptual Level

- The logical view of the database
- describes what data is stored in the database and the relationships among the data

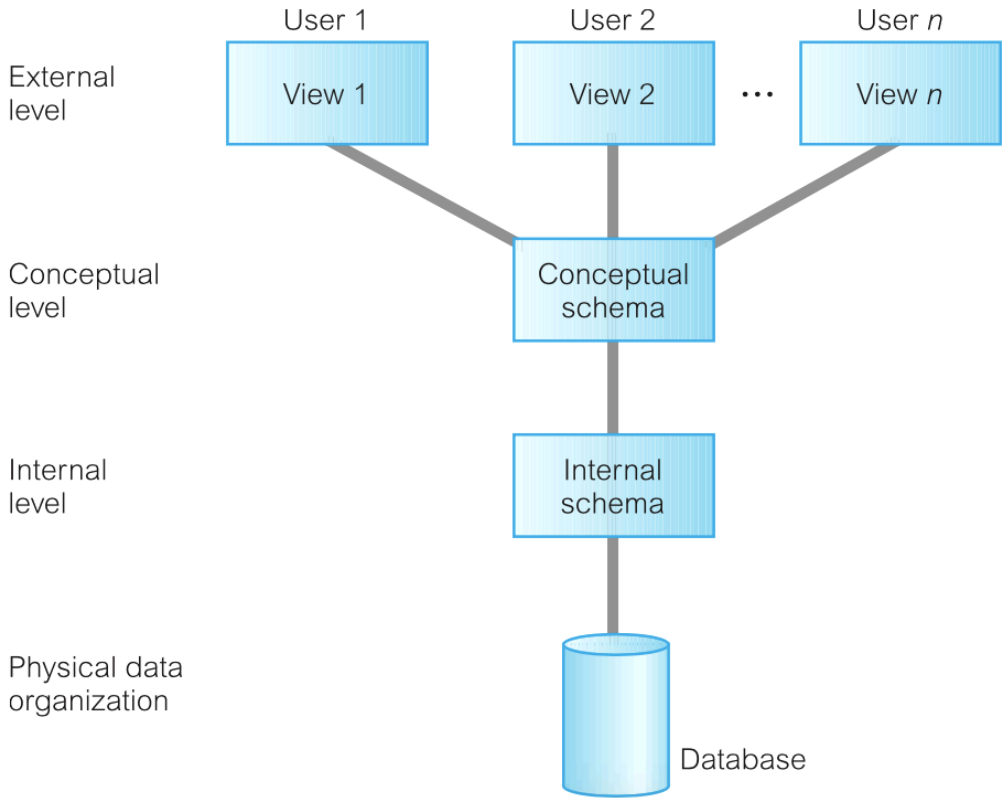
Internal Level

- The physical representation of the database on the computer
- describes how the data is stored in the database

Database Schema

Database Schema

- overall description of the database
- according to the levels of abstraction there are three schemas
  - multiple external schemas
  - one conceptual schema
  - one internal schema
- the DBMS is responsible for mapping between these three types of schema and checks for consistency



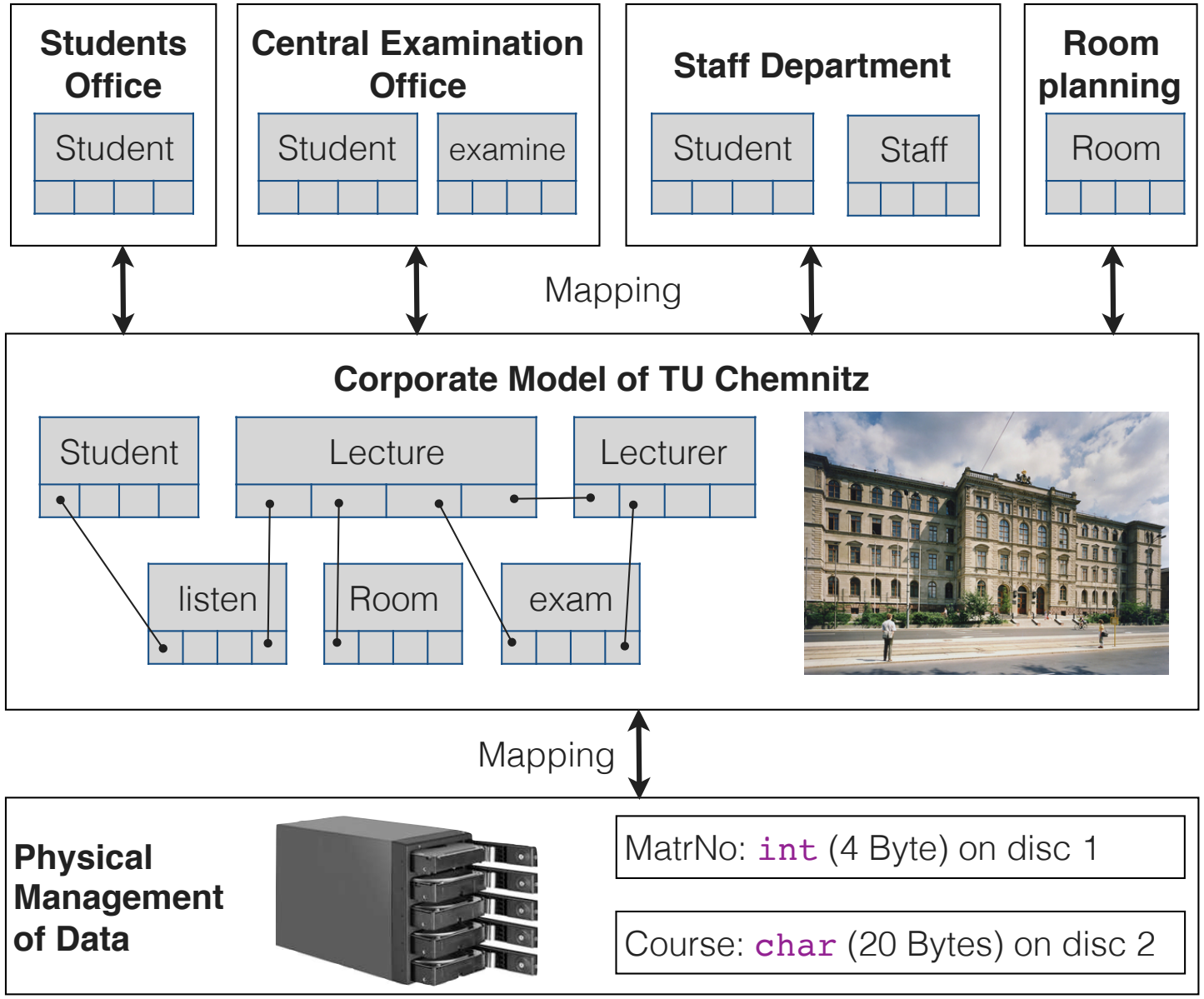
[Connolly & Begg]

Example

external level
(external schemas)

conceptual level
(conceptual schema)

internal level
(internal schema)



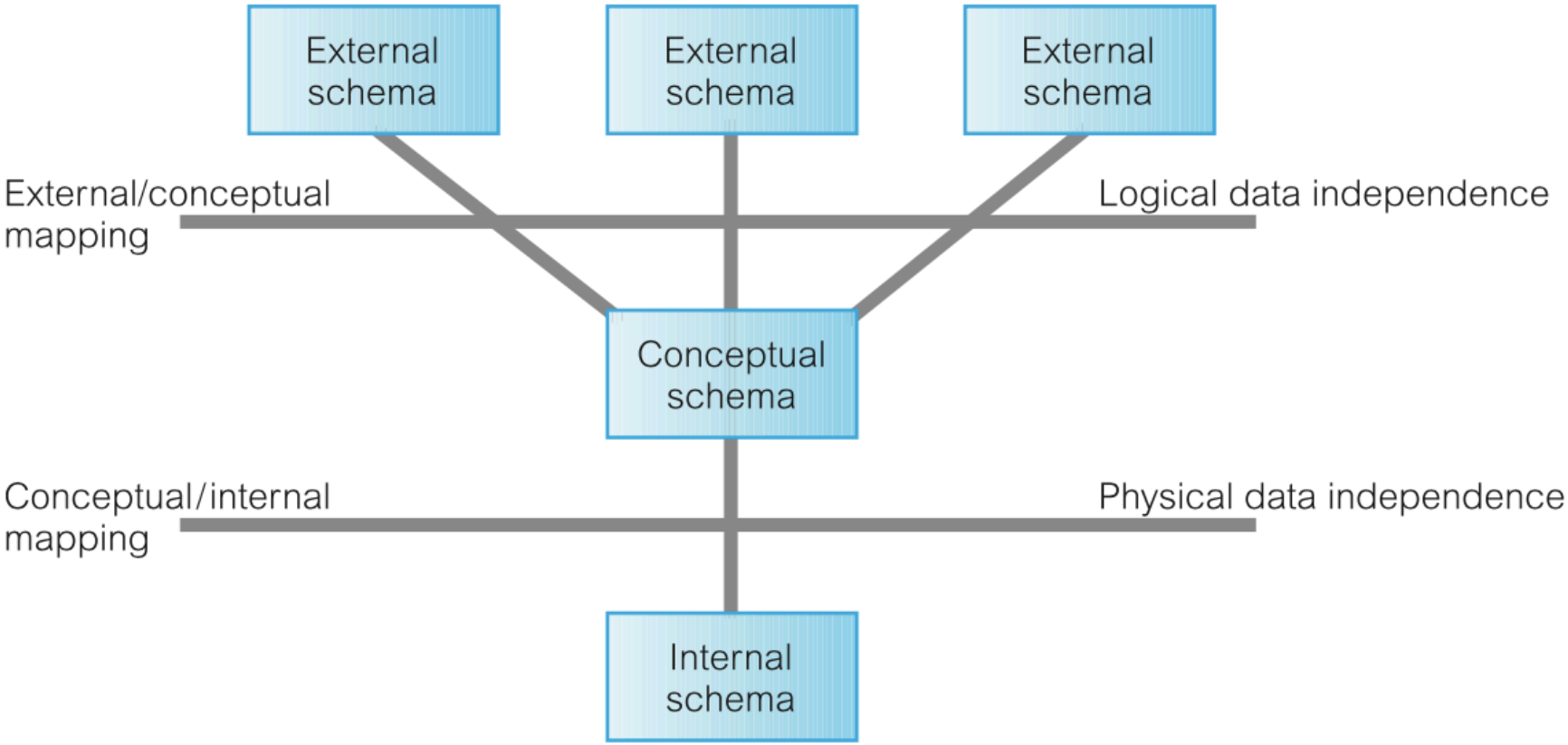
Data Independence

A major objective for the three-level architecture is to provide data independence, which means that upper levels are unaffected by changes to lower levels.

There are two kinds of data independence:

- Logical Data Independence: The immunity of the external schemas to changes in the conceptual schema
- Physical Data Independence: The immunity of the conceptual schema to changes in the internal schema

Data Independence



[Connolly & Begg]

Database Languages

Data Sublanguages

Languages that do not include constructs for all computing needs, such as conditional or iterative statements, which are in contrast provided by high-level programming languages:

- *Data Definition Language (DDL)* is used to specify the database schema
- *Data Manipulation Language* is used to both read and update the database

Host Language

Many DBMSs have a facility for *embedding* the sublanguage in a high-level programming language such as C, C#, or Java, which is sometimes referred to as the *host language.*

Data Model

Definition

A Data Model is an integrated collection of concepts for describing and manipulating data, relationships between data, and constraints on the data in an organization. Usually, it consists of three components:

- a structural part, consisting of a set of rules according to which databases can be constructed
- a manipulative part, defining the types of operation that are allowed on the data
- a set of integrity constraints, which ensures that the data is accurate

Categories of data models

- object-based (e.g. Entity-Relationship, Semantic, Object-Oriented)
- record-based (e.g. relational, network, hierarchical)
- physical (e.g. record structures and orderings)

Phases of Database Design

Database design is made up of three main phases:

Conceptual Design

The process of constructing a model of the data used in an enterprise, independent of all physical considerations.

Logical Design

The process of constructing a model of the data used in an enterprise based on a specific data model, but independent of a particular DBMS and other physical considerations.

Physical Design

The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

Conceptual Data Modeling

To get a precise understanding of the nature of the data and how it is used by the enterprise, we need a model for communication that is nontechnical and free of ambiguities.

Entity-Relationship (ER) Model

- is the most widely used in the conceptual database design process
- although there is general agreement about what each ER-concept means, there are a number of different notations that can be used to represent these concepts diagrammatically
- we use the Unified Modeling Language (UML) to visualize the concepts of ER. UML is currently recognized as the de facto industry standard modeling language for object-oriented software engineering projects
- we use the UML notation for drawing ER models, but continue to describe the concepts of ER models using traditional database terminology

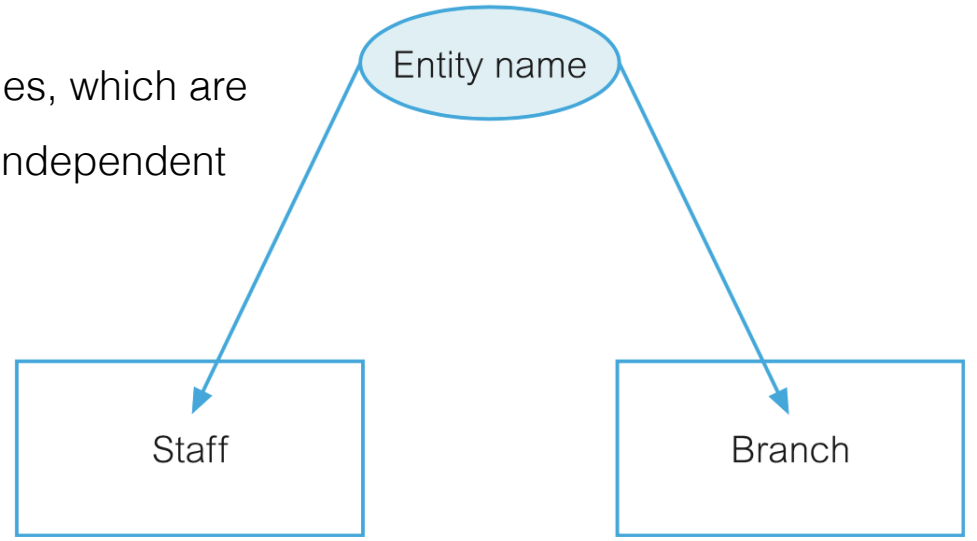
The ER Model (1)

Entity Type

A group of objects with the same properties, which are identified by the enterprise as having an independent existence (physically or conceptually).

Entity Occurence

A uniquely identifiable object of an entity type.

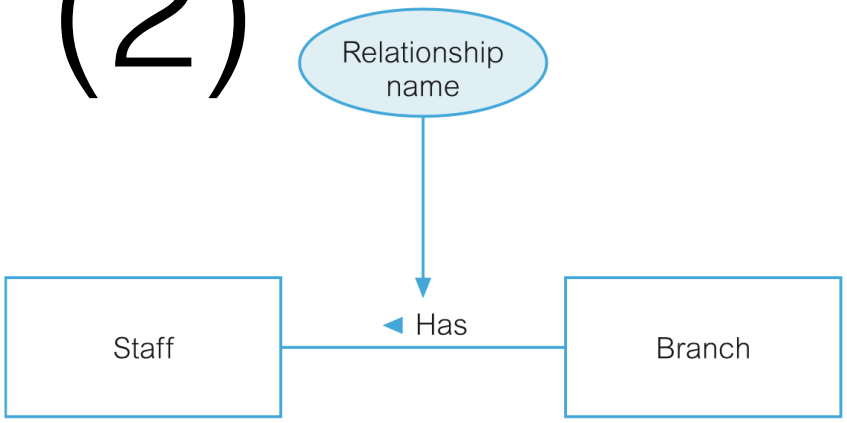


[Connolly & Begg]

Diagrammatic representation of Entity Types

Each entity type is shown as a rectangle, labeled with the name of the entity, which is normally a singular noun. In UML, the first letter of each word in the entity name is uppercase.

The ER Model (2)



[Connolly & Begg]

Relationship Type

A set of meaningful associations among entity types.

Relationship Occurrence

A uniquely identifiable association that includes one occurrence from each participating entity type.

Diagrammatic representation of relationship types

- Each relationship type is shown as a line connecting the associated entity types and labeled with the name of the relationship.
- A relationship is named using a verb or a short phrase including a verb. The first letter of each word in the relationship name is shown in uppercase. Whenever possible, a relationship name should be unique for a given ER model.
- A relationship is only labeled in one direction by placing an arrow symbol beside the name indicating the correct direction for a reader to interpret the relationship name.

The ER Model (3)

Degree of a relationship type

Describes the number of participating entity types in a relationship. A relationship of degree two is called binary.

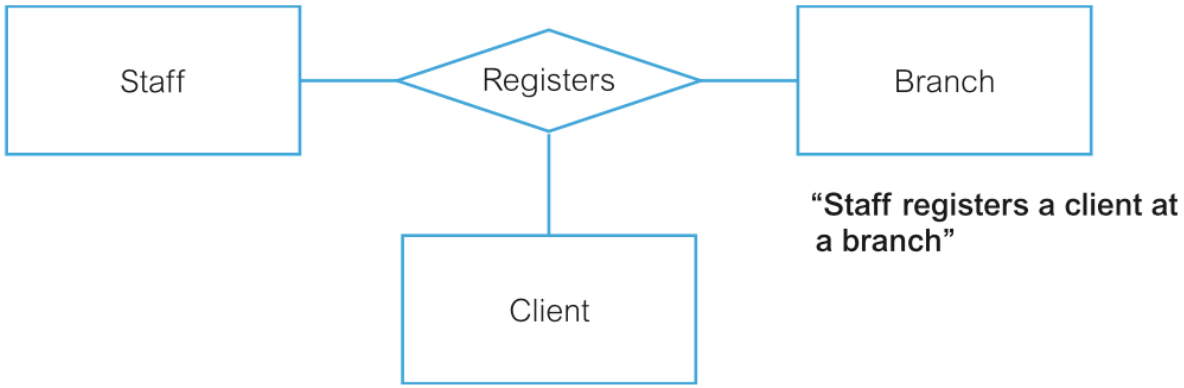
Complex Relationships and diagrammatic representation

A relationship with degree higher than two is called complex and represented by a diamond.

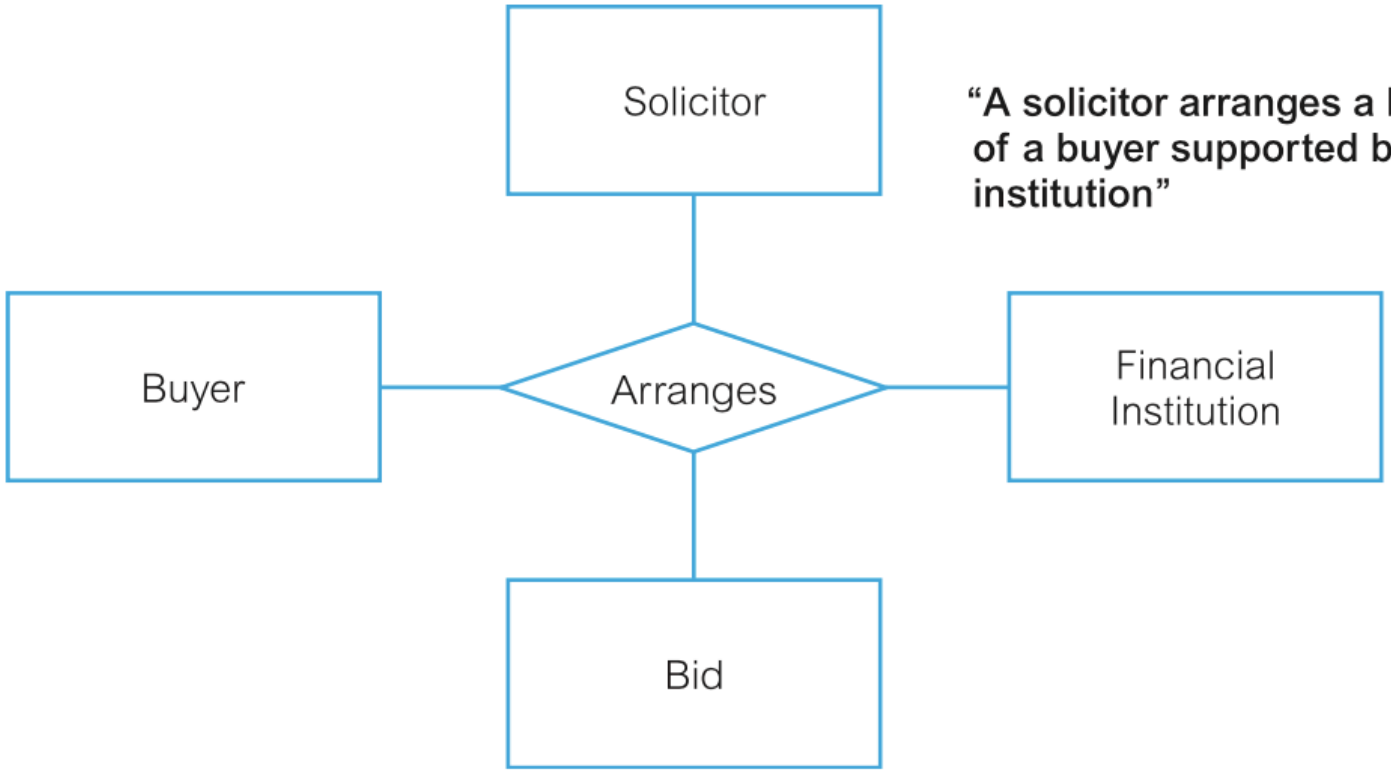
The name of the relationship is displayed inside the diamond, and in this case, the directional arrow normally associated with the name is omitted.

Example:

Ternary Relationship *Registers*



Example - Quarternary Relationship *Arranges*



"A solicitor arranges a bid on behalf of a buyer supported by a financial institution"

[Connolly & Begg]

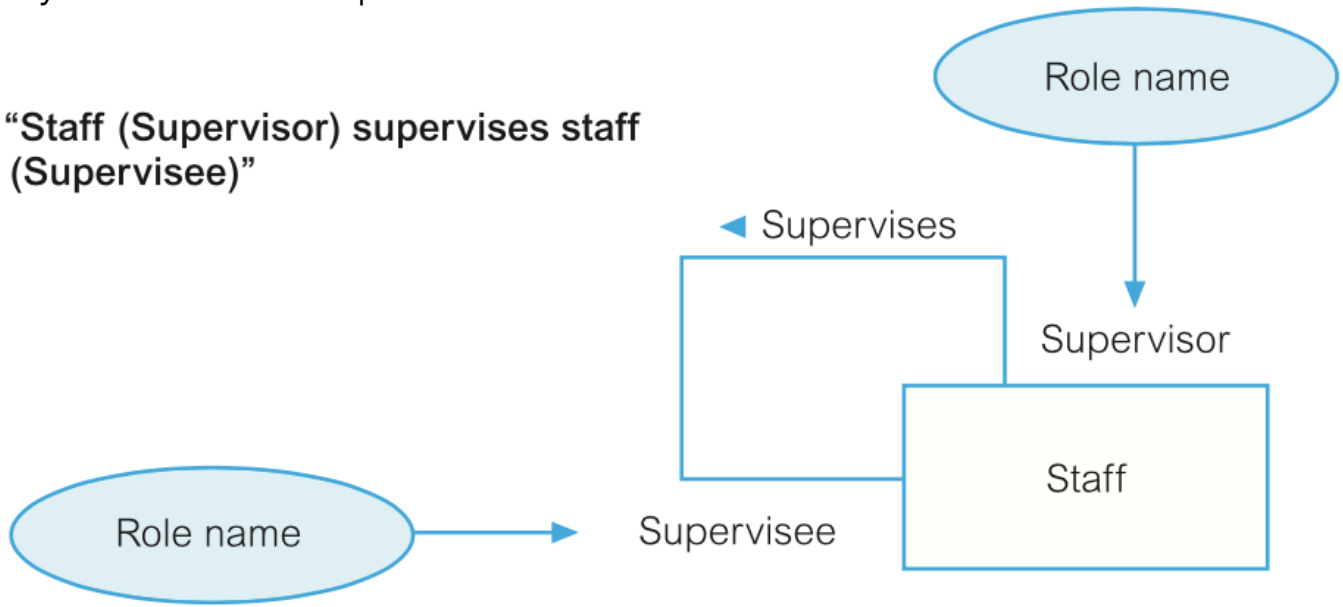
The ER Model (4)

Recursive Relationship

A relationship type in which the *same* entity type participates more than once in different roles.

Role name

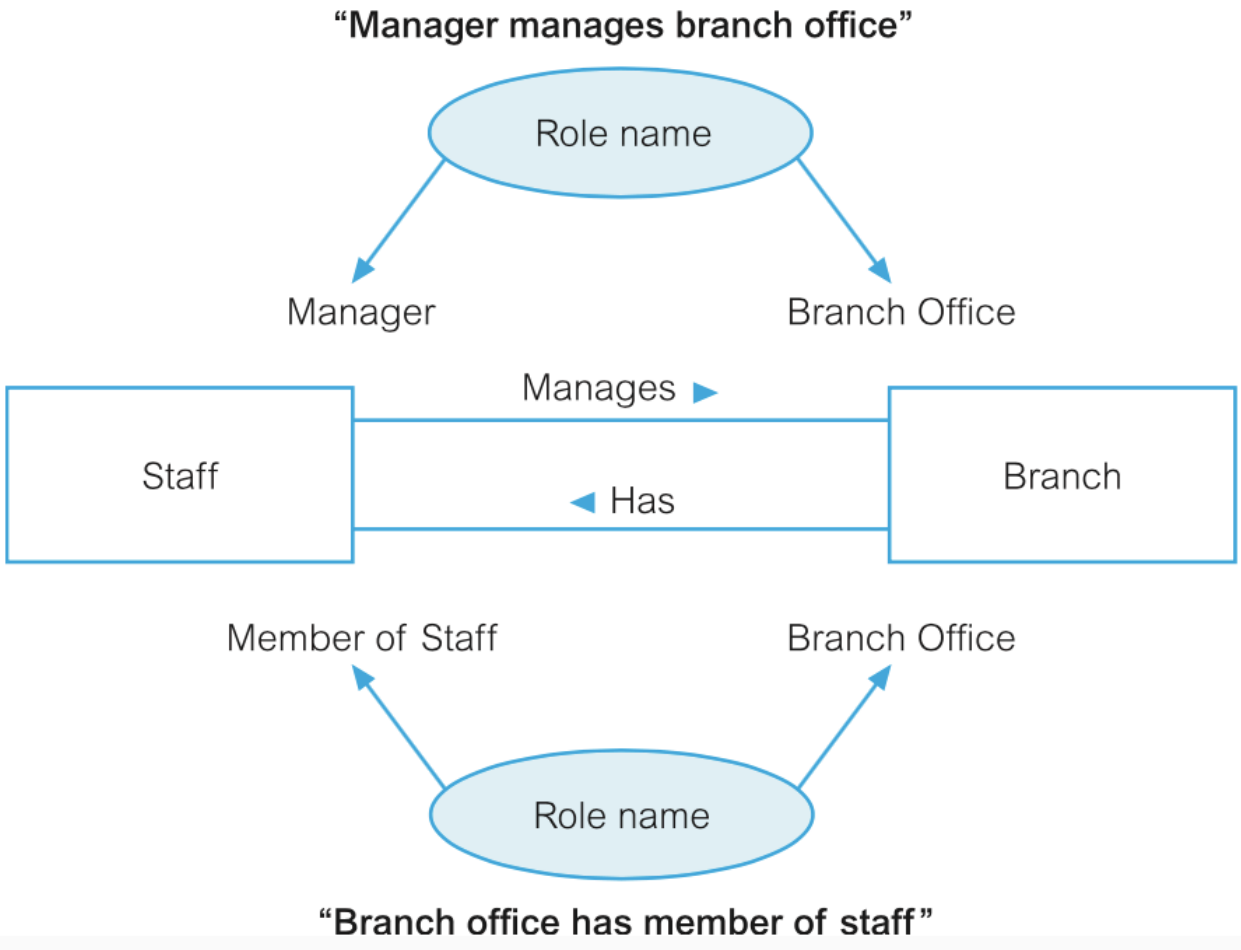
Relationships may be given role names to indicate the purpose that each participating entity type plays in a relationship.



[Connolly & Begg]

Role Names

Role names may also be used when two entities are associated through more than one relationship.



[Connolly & Begg]

The ER Model (5)

Attribute

A property of an entity or a relationship type.

Attribute Domain

The set of allowable values for one or more attributes.

Classification of attributes

- simple or composite
- single-valued or multi-valued
- derived

Classification of Attributes

Simple Attribute

- composed of a single component with an independent existence

Composite Attribute

- composed of multiple components, each with an independent existence

Single-Valued-Attribute

- holds a single value for each occurrence of an entity type

Multi-Valued Attribute

- holds multiple values for each occurrence of an entity type

Derived Attribute

- represents a value that is derivable from the value of a related attribute or set of attributes, not necessarily in the same entity type

The ER-Model (6)

Candidate Key

- The minimal set of attributes that uniquely identifies each occurrence of an entity type.

Primary Key

- The candidate key that is selected to uniquely identify each occurrence of an entity type.
- set of allowable values for one or more attributes

Composite Key

- a candidate key that consists of two or more attributes

The ER Model (7)

Diagrammatic representation of attributes

If an entity type is to be displayed with its attributes, we divide the rectangle representing the entity in two. The upper part of the rectangle displays the name of the entity and the lower part lists the names of the attributes.

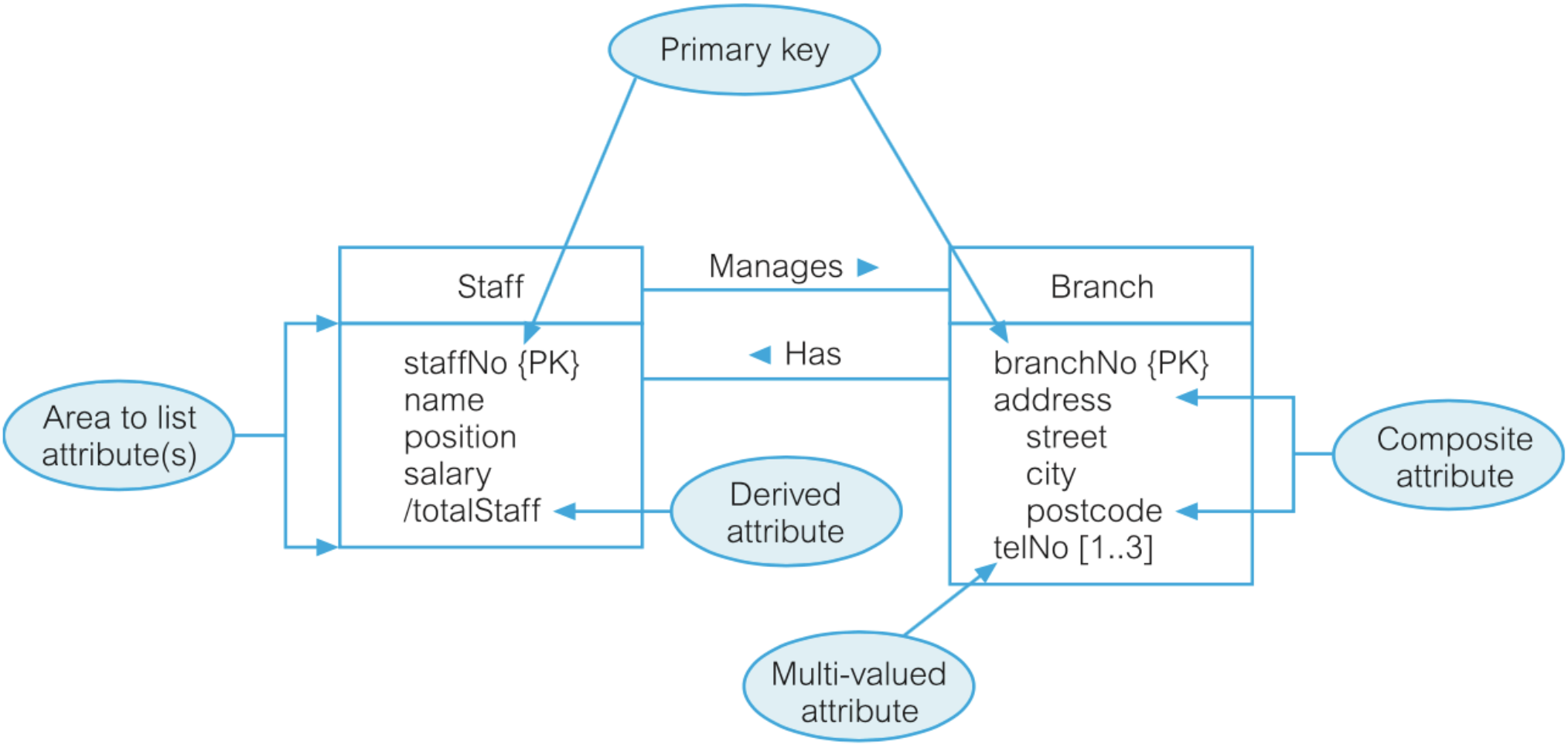
The first attribute(s) to be listed is the primary key for the entity type (if known).

The name(s) of the primary key attribute(s) can be labeled with the tag `{PK}`.

In UML, the name of an attribute is displayed with the first letter in lowercase. If the name has more than one word, the first letter of each subsequent word has to be in uppercase.

Additional tags that can be used include partial primary key `{PPK}` when an attribute forms part of a composite primary key, and alternate key `{AK}`.

Example - Attributes



[Connolly & Begg]

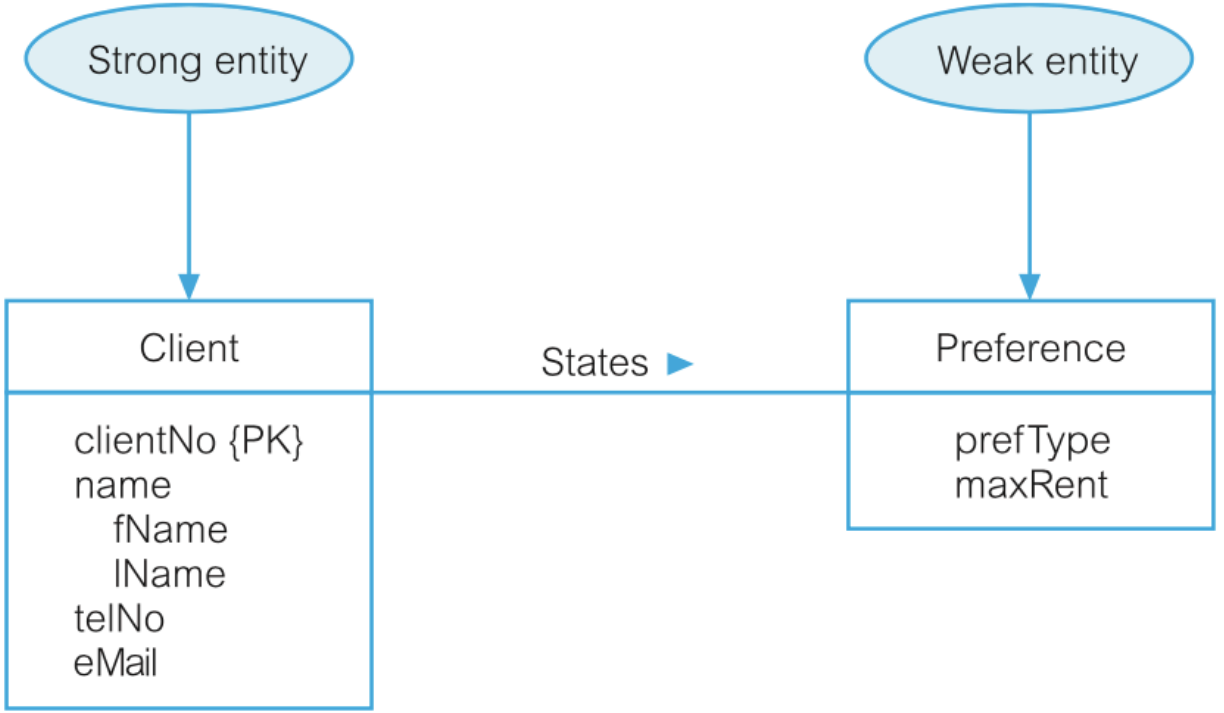
The ER Model (8)

Strong Entity Type

An entity type that is *not* existence-dependent on some other entity type.

Weak Entity Type

An entity type that is existence-dependent on some other entity type.



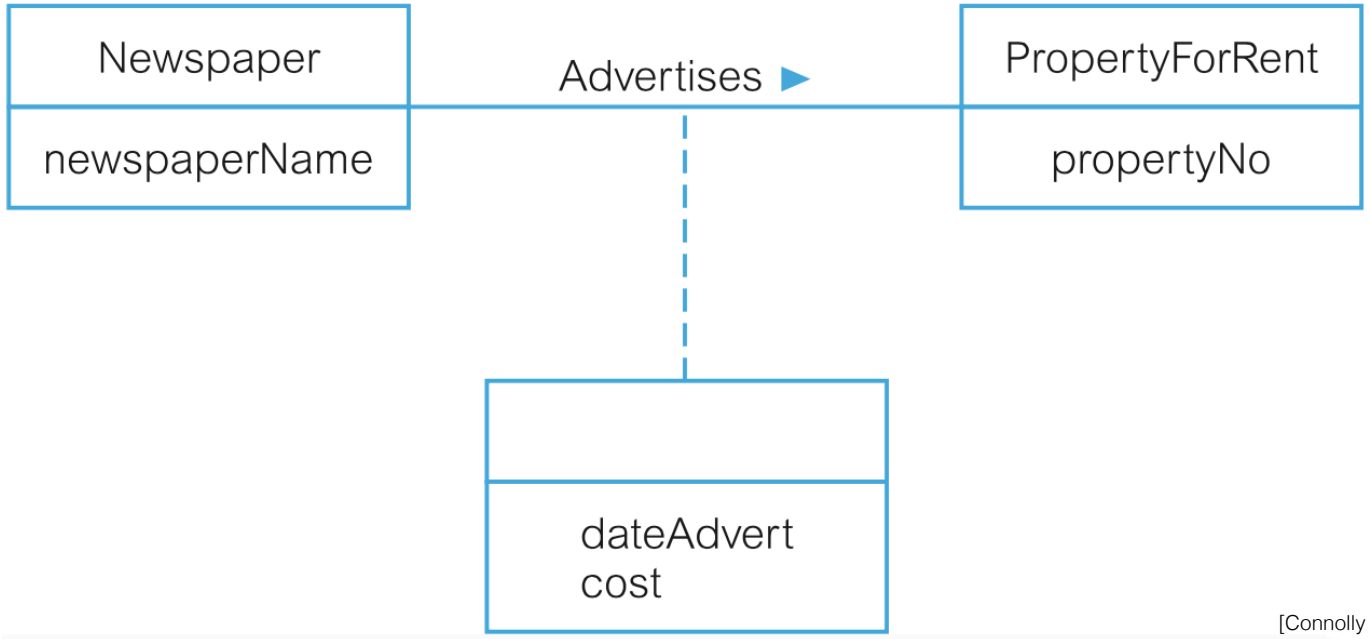
[Connolly & Begg]

The ER Model (9)

Diagrammatic representation of attributes on relationships

Attributes associated with a relationship type are represented using the same symbol as an entity type.

To distinguish between a relationship with an attribute and an entity, the rectangle representing the attribute(s) is associated with the relationship using a dashed line.



[Connolly & Begg]

The ER Model (10)

Multiplicity

The number (or range) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.

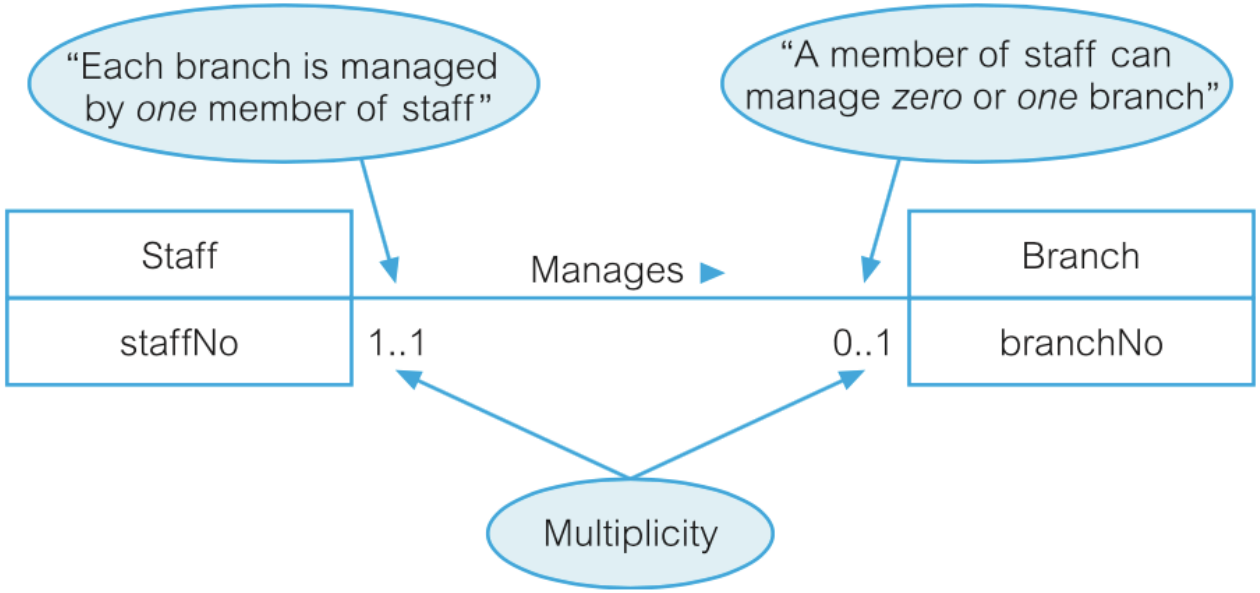
The most common degree for relationships is binary. Binary relationships are generally referred to as being

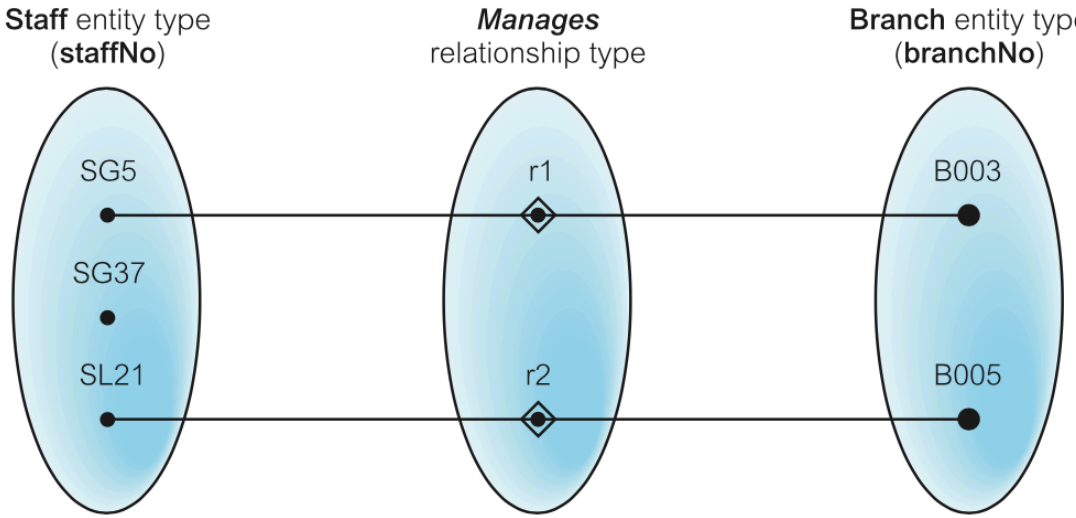
- one-to-one (1:1), e.g. a member of staff manages a branch
- one-to-many (1:*), e.g. a member of staff oversees properties for rent
- many-to-many (*:*), e.g. a newspapers advertise properties for rent

Attention

Not all integrity constraints can be represented in an ER model.

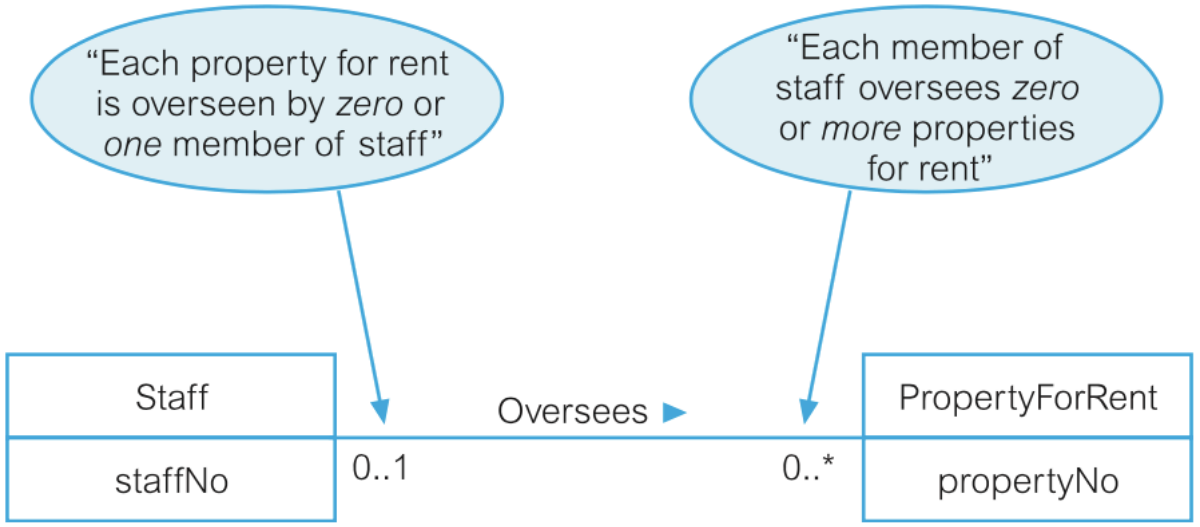
1:1 Relationship

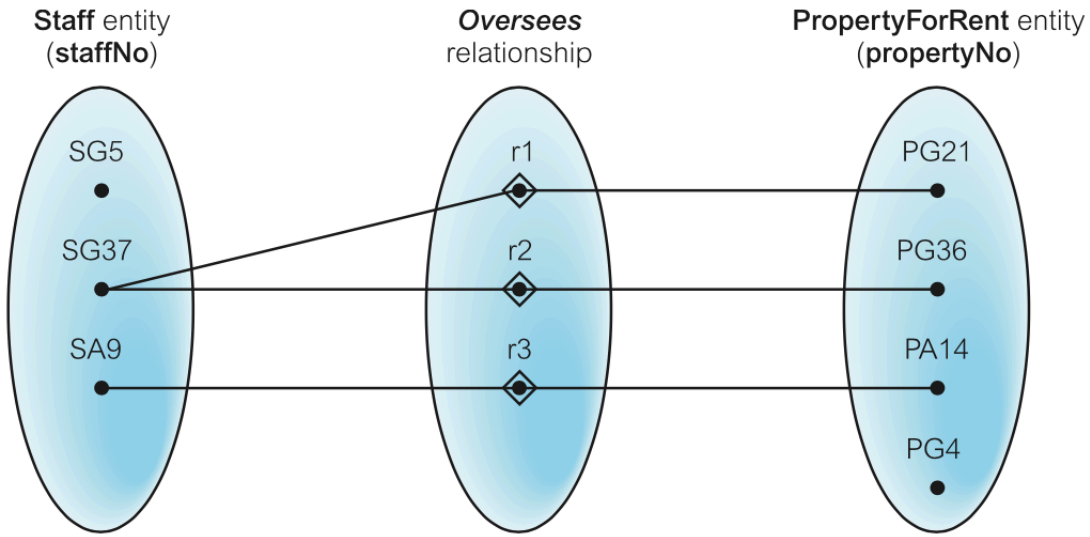




[Connolly & Begg]

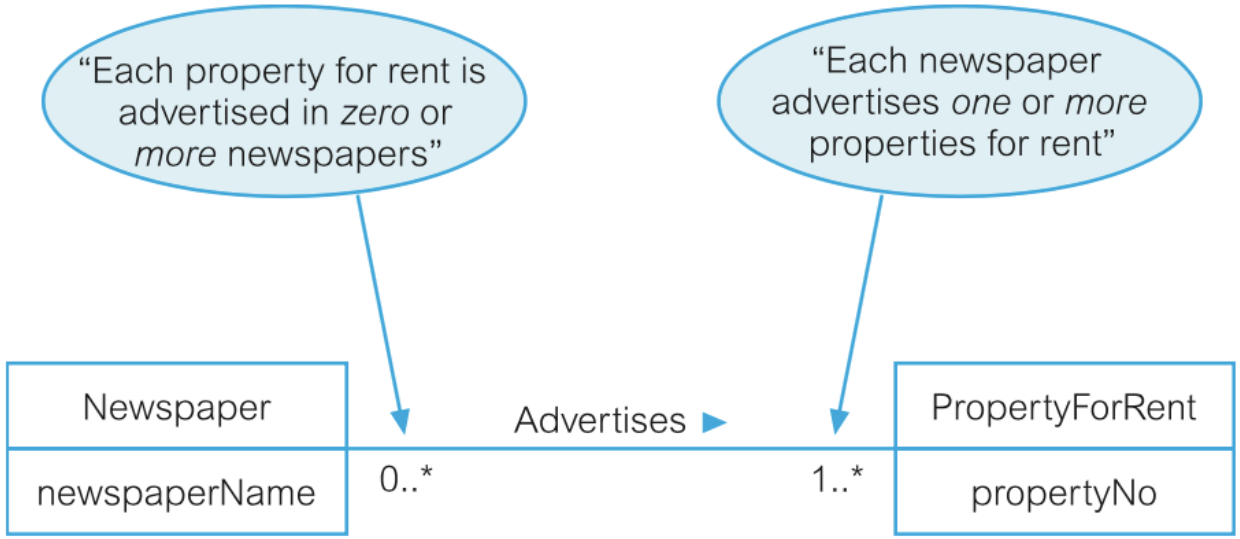
The 1:* Relationship

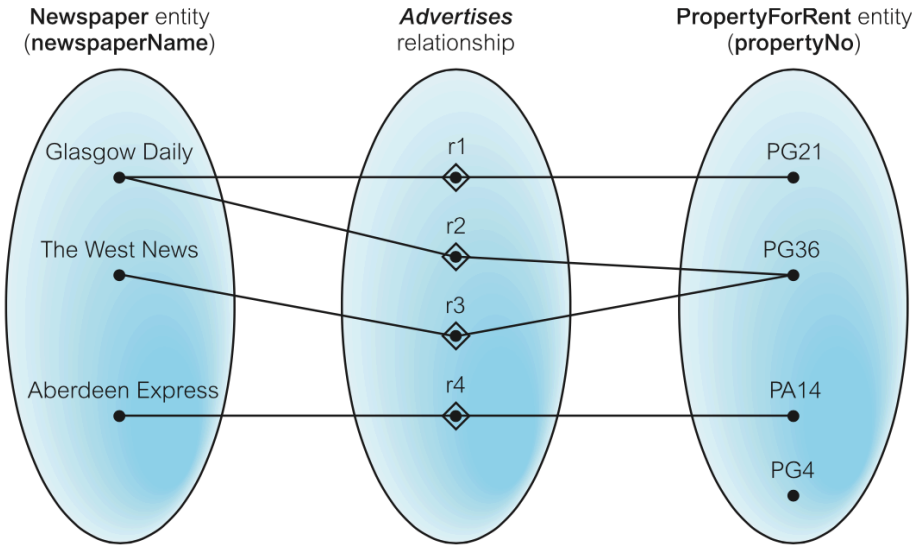




[Connolly & Begg]

: Relationship



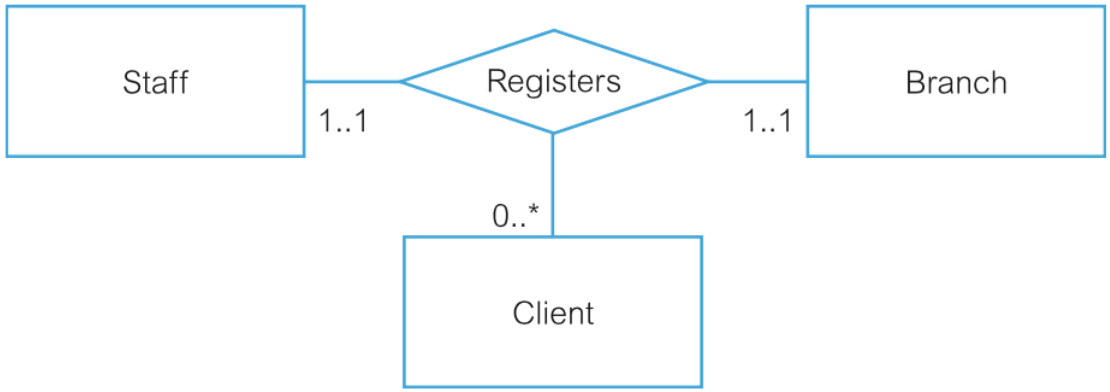


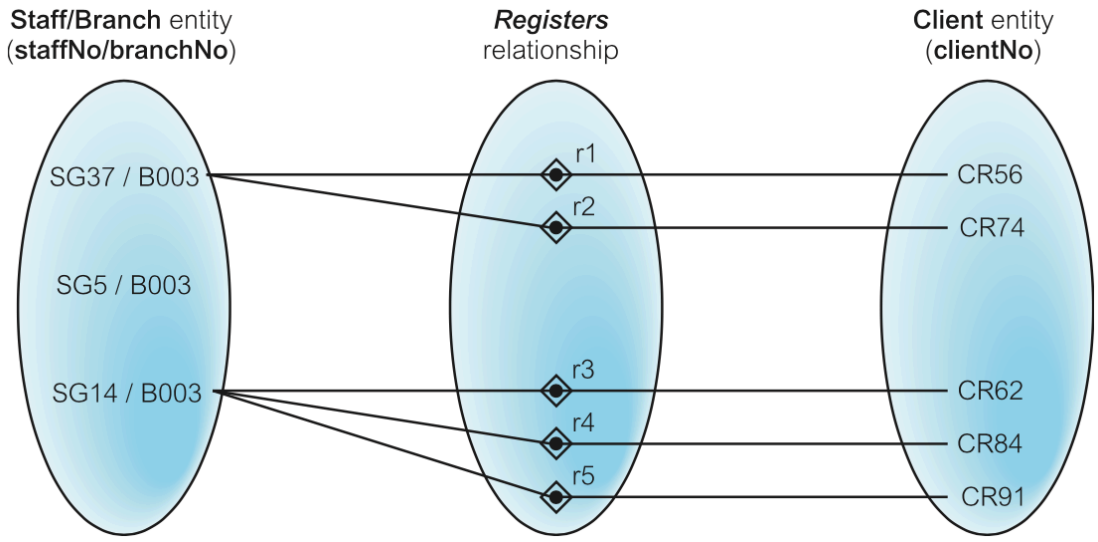
[Connolly & Begg]

The ER Model (11)

Multiplicity (of complex relationships)

The number (or range) of possible occurrences of an entity type in an *n*-ary relationship when the other (*n*–1) values are fixed.





[Connolly & Begg]

The ER Model (12)

ALTERNATIVE WAYS TO REPRESENT MULTIPLICITY CONSTRAINTS	MEANING
0..1	Zero or one entity occurrence
1..1 (or just 1)	Exactly one entity occurrence
0..* (or just *)	Zero or many entity occurrences
1..*	One or many entity occurrences
5..10	Minimum of 5 up to a maximum of 10 entity occurrences
0, 3, 6–8	Zero or three or six, seven, or eight entity occurrences

[Connolly & Begg]

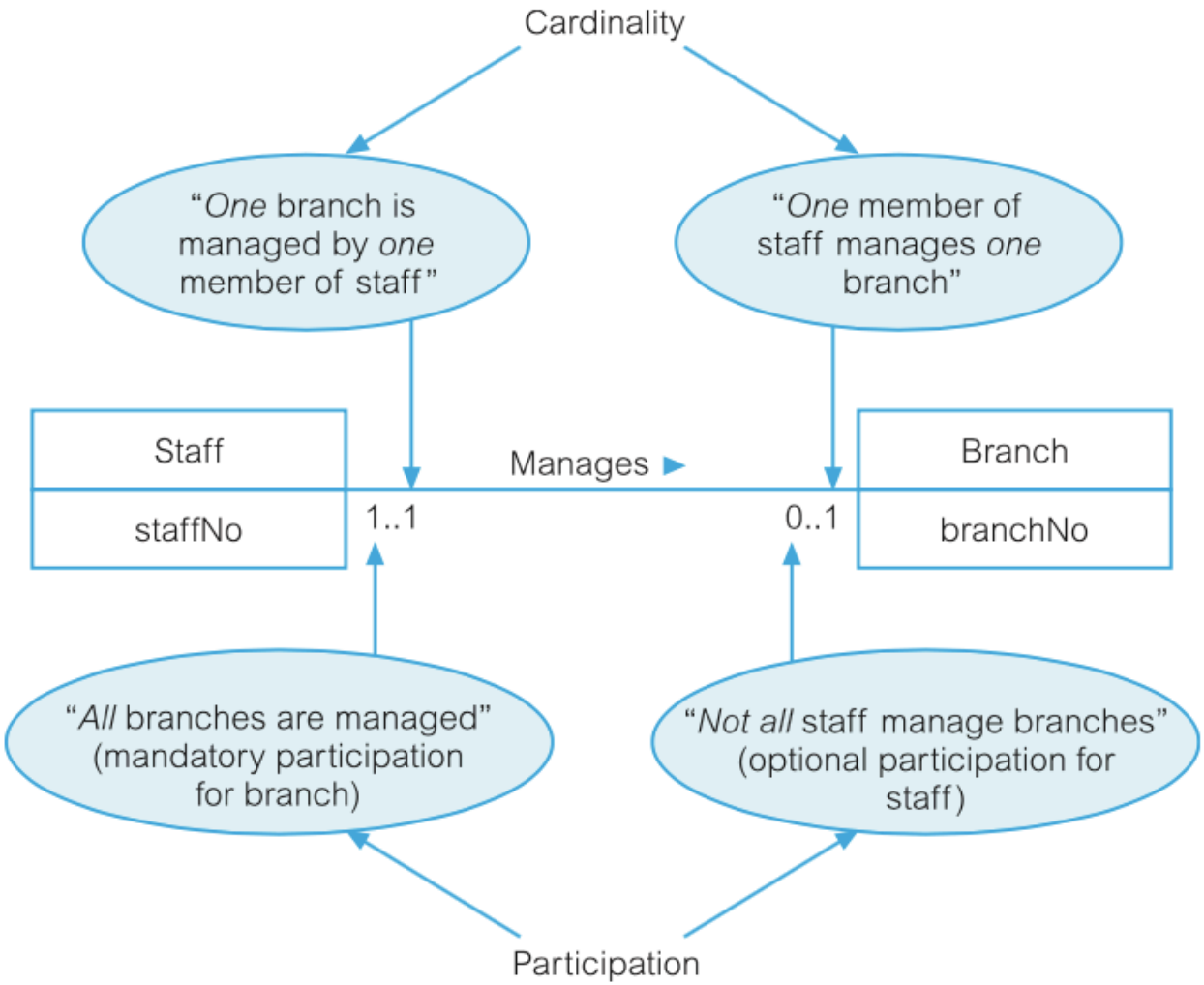
The ER Model (13)

Cardinality

Describes the 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.



[Connolly & Begg]