HALMSTAD UNIVERSITY

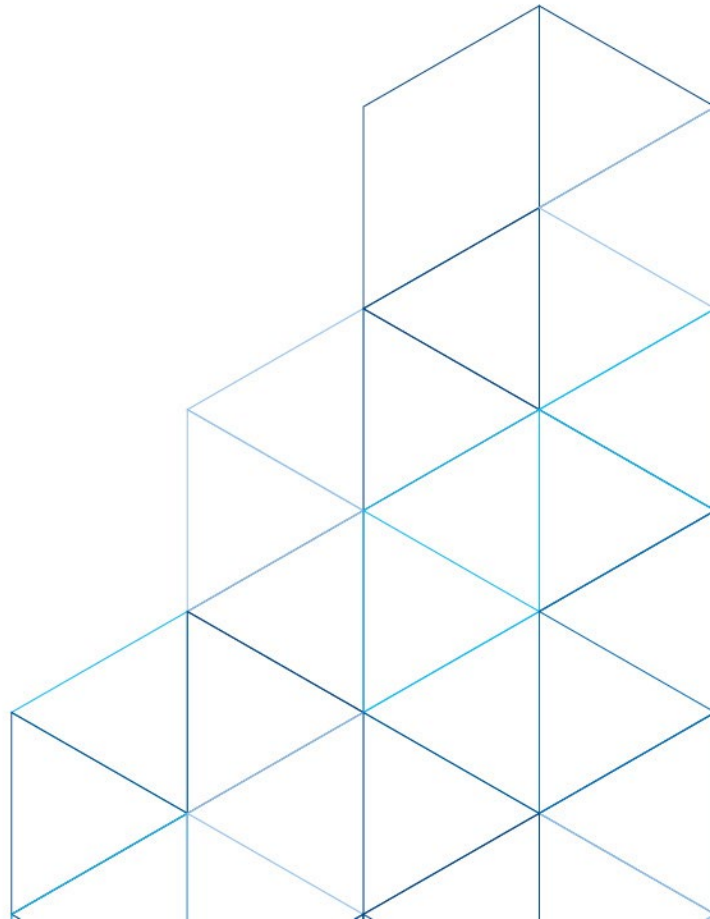
Web Systems Fundamentals and Databases (Grundläggande webbsystem och databaser) DI4020 - 11hp

Lectures 3 and 4

Wagner Ourique de Morais

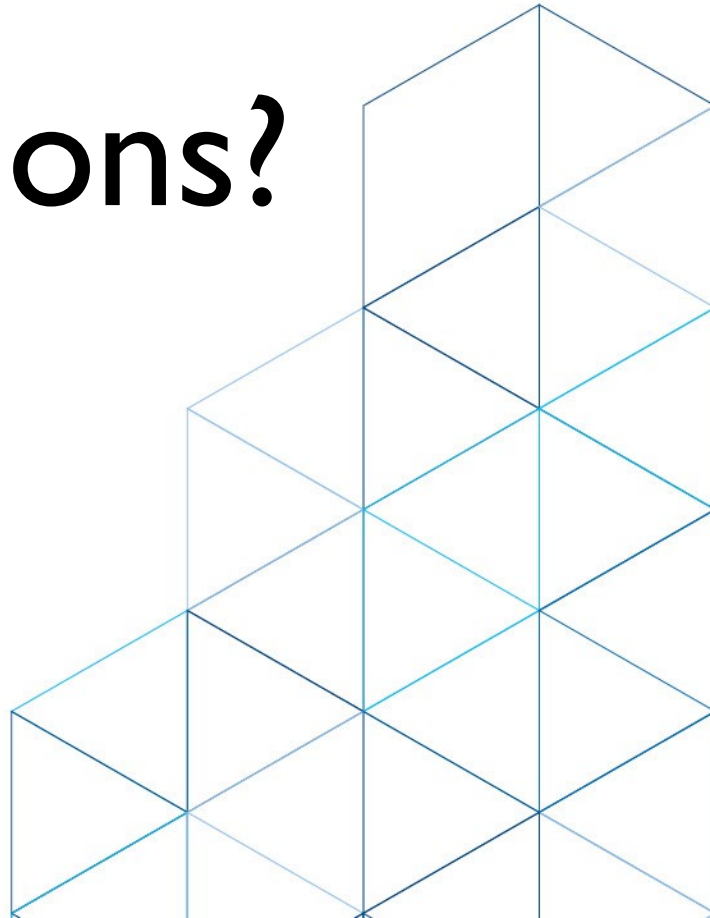
Previously in the course

- Course Introduction
- Lectures 1 and 2
  - Introduction to the field of field of database systems.



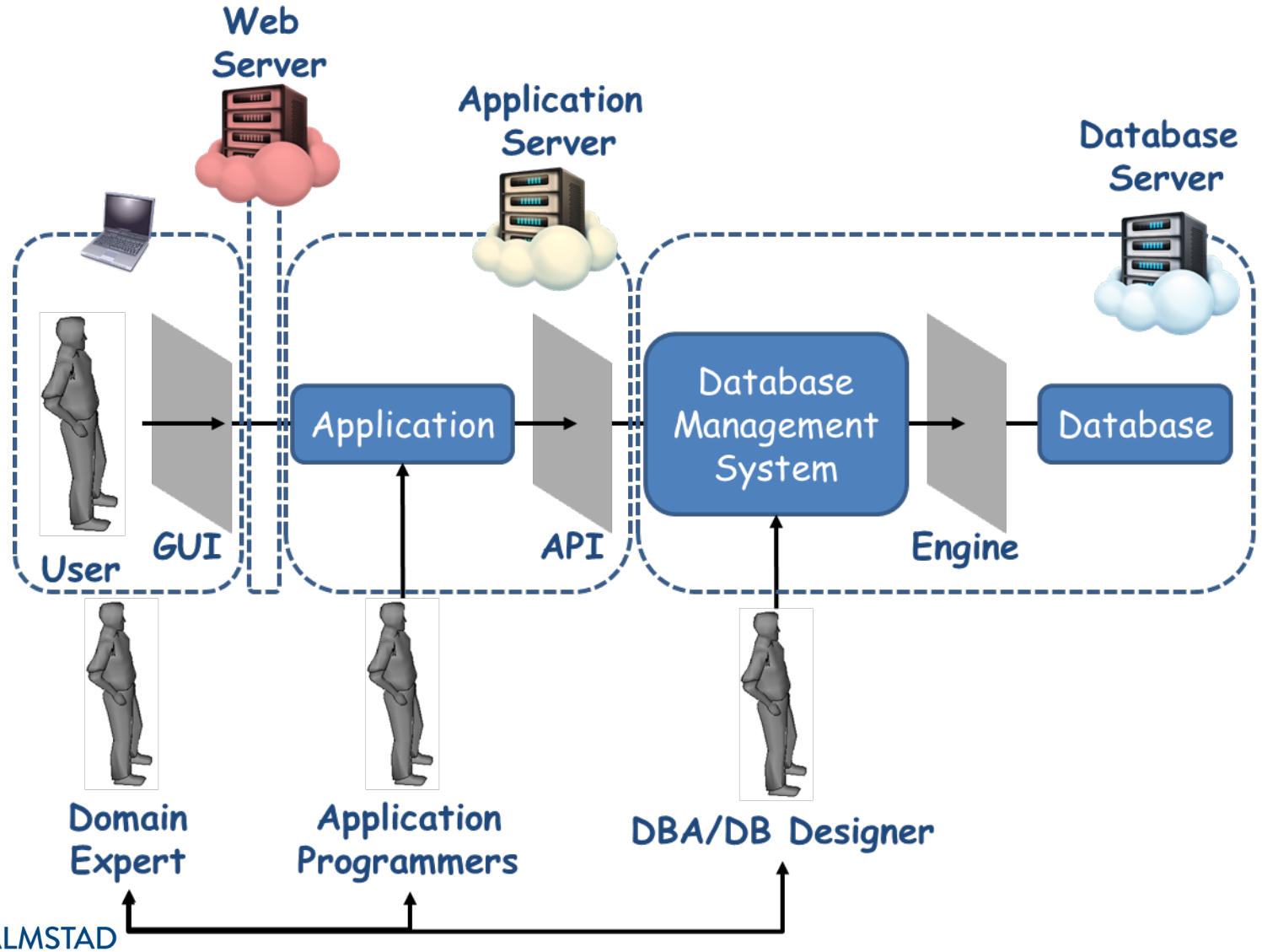
HALMSTAD UNIVERSITY

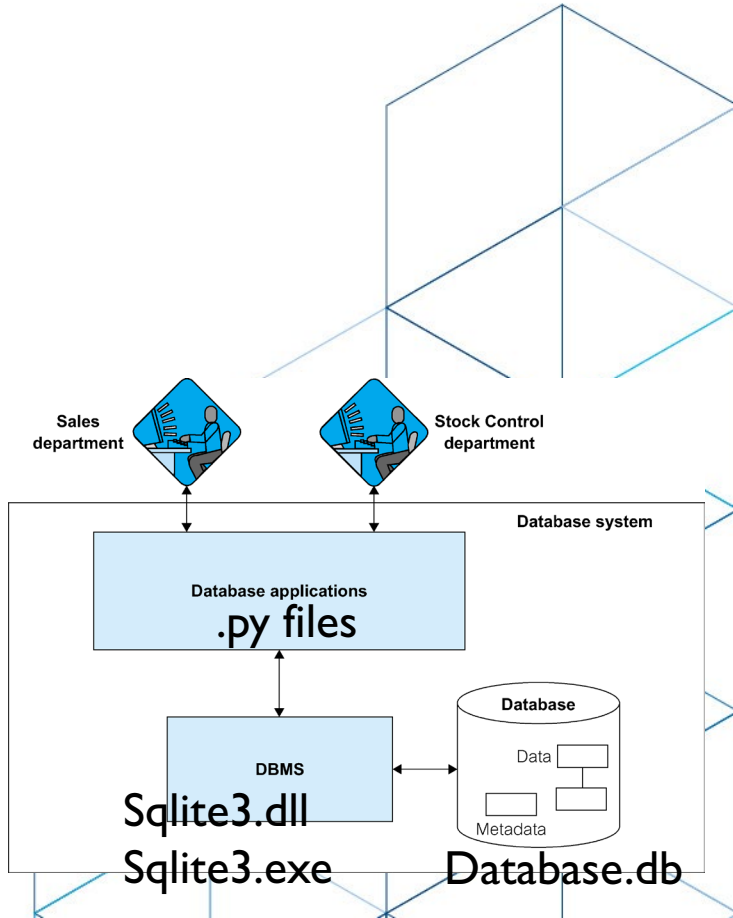
Comments or Questions?



HALMSTAD UNIVERSITY

Database systems

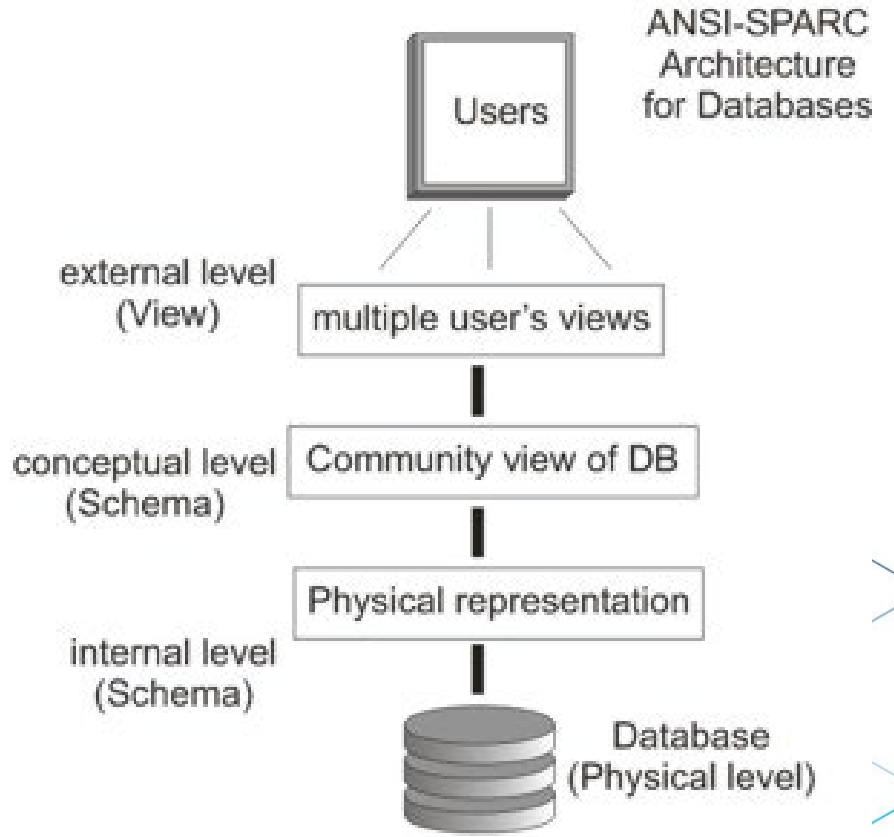


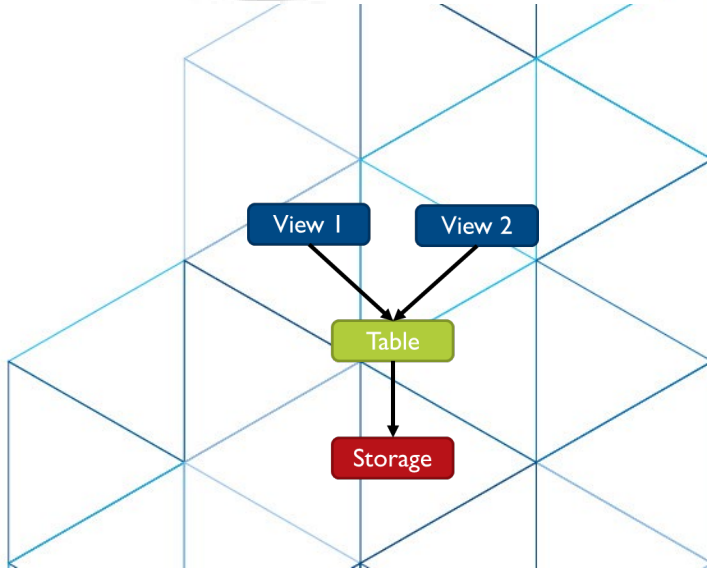


HALMSTAD UNIVERSITY

Three-Level ANSI-SPARC Architecture

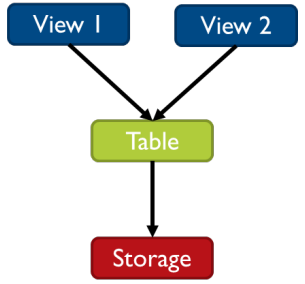
- Degrees of data abstraction
  - External, conceptual, and internal.
- Aim to separate the users' view
  - All users should be able to access same data.
  - A user's view is immune to changes made in other views.
  - Users should not need to know physical database storage details.
  - DBA should be able to change conceptual structure of database without affecting all users.
  - DBA should be able to change database storage structures without affecting the users' views.
  - Internal structure of database should be unaffected by changes to physical aspects of storage.



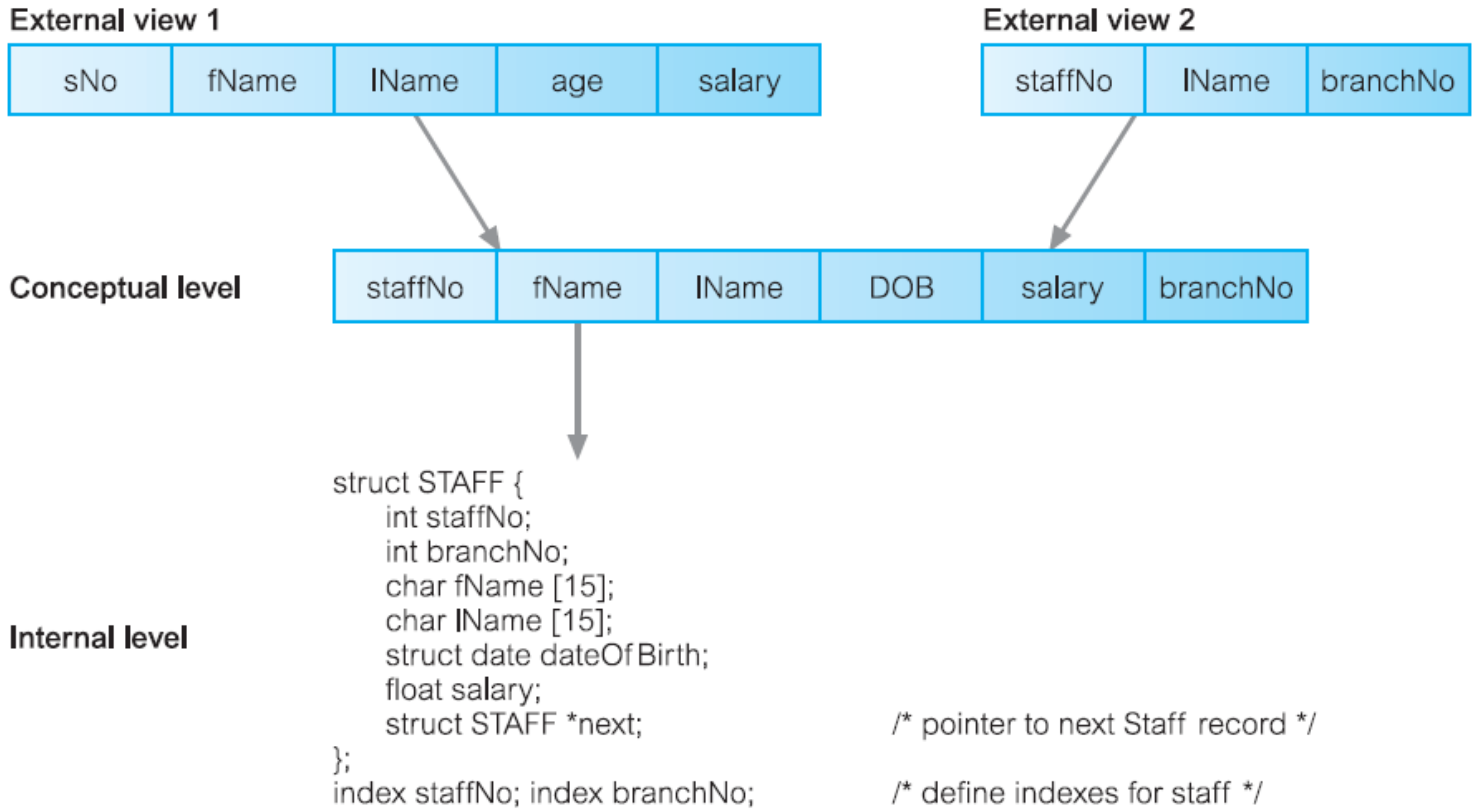


HALMSTAD UNIVERSITY

Degrees of data abstraction



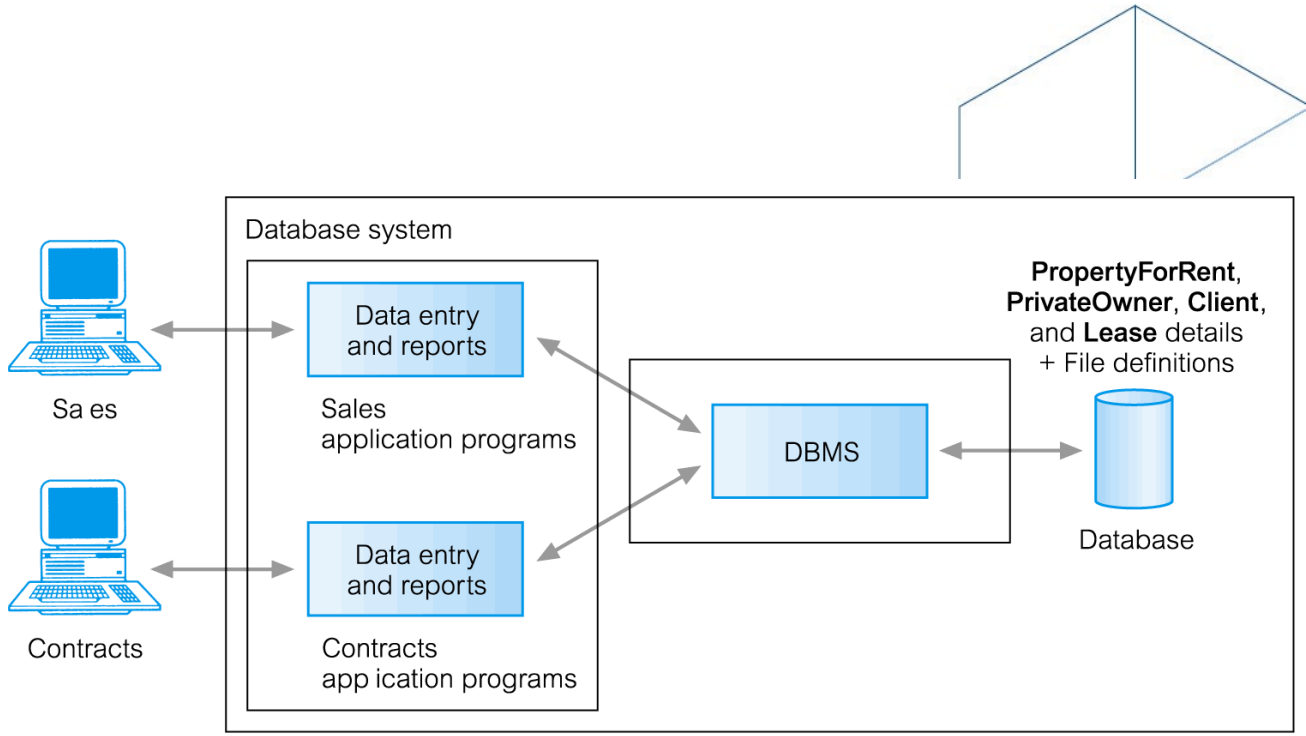
- External Level or View level
  - End users' view of the database.
  - Describes that part of database that is relevant to a particular user.
  - What information is exposed to users, what are they allowed to see…
- Conceptual Level
  - Community view of the database.
  - What tables are there, their attributes and relationships.
  - What constraints should the DBMS enforce…
- Internal Level or Physical level
  - Physical representation of the database on the computer.
  - How data is stored, where it is located, how many bytes, what type of indexes…



HALMSTAD UNIVERSITY

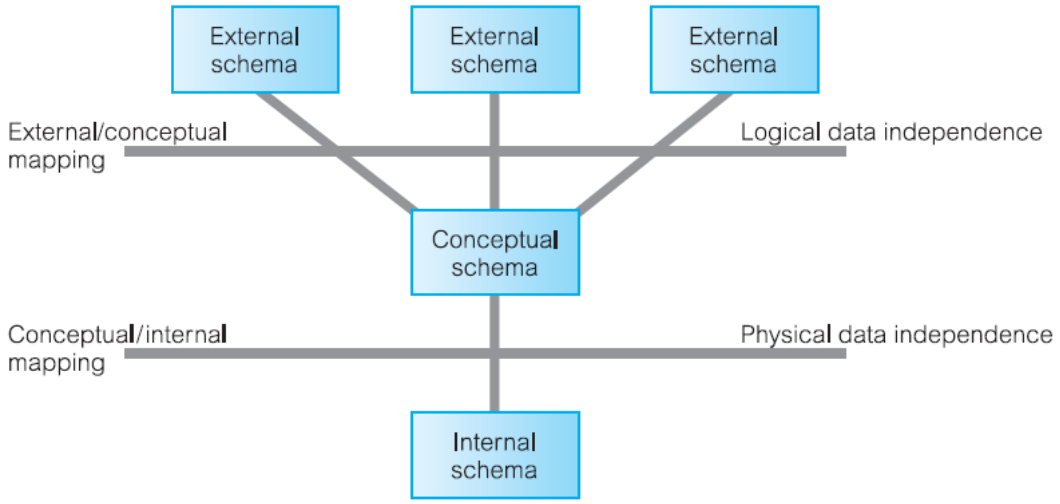
Database Views

- Allows each user to have his or her own view of the database.
- A view is essentially some subset of the database.
- **Benefits**
  - Reduce complexity
  - Provide a level of security
  - Provide a mechanism to customize the appearance of the database
  - Present a consistent, unchanging picture of the structure of the database, even if the underlying database is changed



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentF nish)

Data Independence



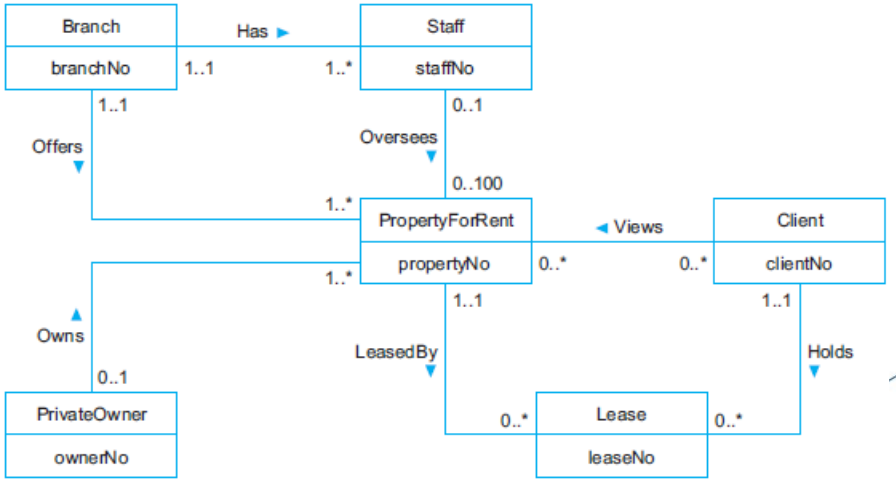
- Logical Data Independence
  - Refers to immunity of external schemas to changes in conceptual schema (e.g. addition/removal of entities).
    - Modify table definitions without having change the application's views.
      - Add/drop/rename attributes for a table or rename a table.
  - Should not require changes to external schema or rewrites of application programs.
- Physical Data Independence
  - Refers to immunity of conceptual schema to changes in the internal schema (e.g. using different file organizations, storage structures/devices).
    - Change how/where database objects are represented in the physical storage.
    - Change to 64-bits for integers and move a table to another disk/machine.
  - Should not require change to conceptual or external schemas.

Data Models

- A **data model** is an abstraction of a real-world object or event.
  - Integrated collection of concepts for describing data, relationships between data, and constraints on the data in an organization.
  - Data Model comprises:
    - a structural part;
    - a manipulative part;
    - possibly a set of integrity rules.

- **Data abstraction** is about omitting or hiding specific details of how the data are managed.
  - The ANSI/SPARC architecture defines three levels of data abstraction: external (subset), conceptual (what), and internal (how).

- Purpose
  - To represent data in an understandable way, ie. to hide storage details and present the users with a conceptual view of the database.
    - Programs refer to the data model constructs rather than data storage details.



Entity–Relationship Data Model

Branch

branchNo	street	city	postCode
B005	22 Deer Rd	London	SW1 4EH
B007	16 Argyll St	Aberdeen	AB2 3SU
B003	163 Main St	Glasgow	G11 9QX
B004	32 Manse Rd	Bristol	BS99 1NZ
B002	56 Clover Dr	London	NW10 6EU

Staff

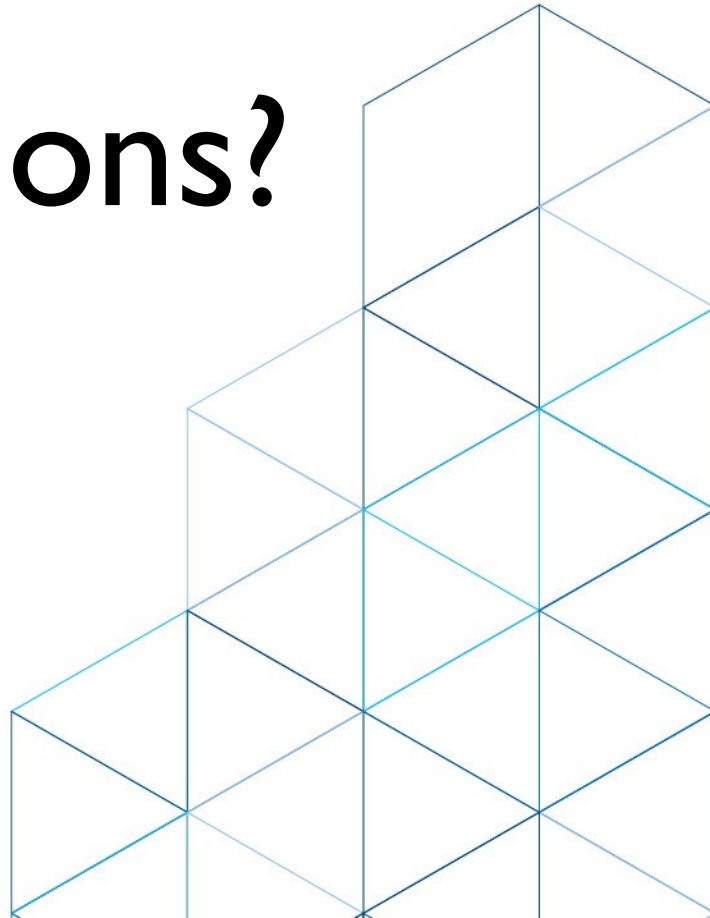
staffNo	fName	lName	position	sex	DOB	salary	branchNo
SL21	John	White	Manager	M	1-Oct-45	30000	B005
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000	B003
SA9	Mary	Howe	Assistant	F	19-Feb-70	9000	B007
SG5	Susan	Brand	Manager	F	3-Jun-40	24000	B003
SL41	Julie	Lee	Assistant	F	13-Jun-65	9000	B005

Relational Data Model

Masri, E., & Navathe, S. (2016). Fundamentals of database systems.

HALMSTAD UNIVERSITY

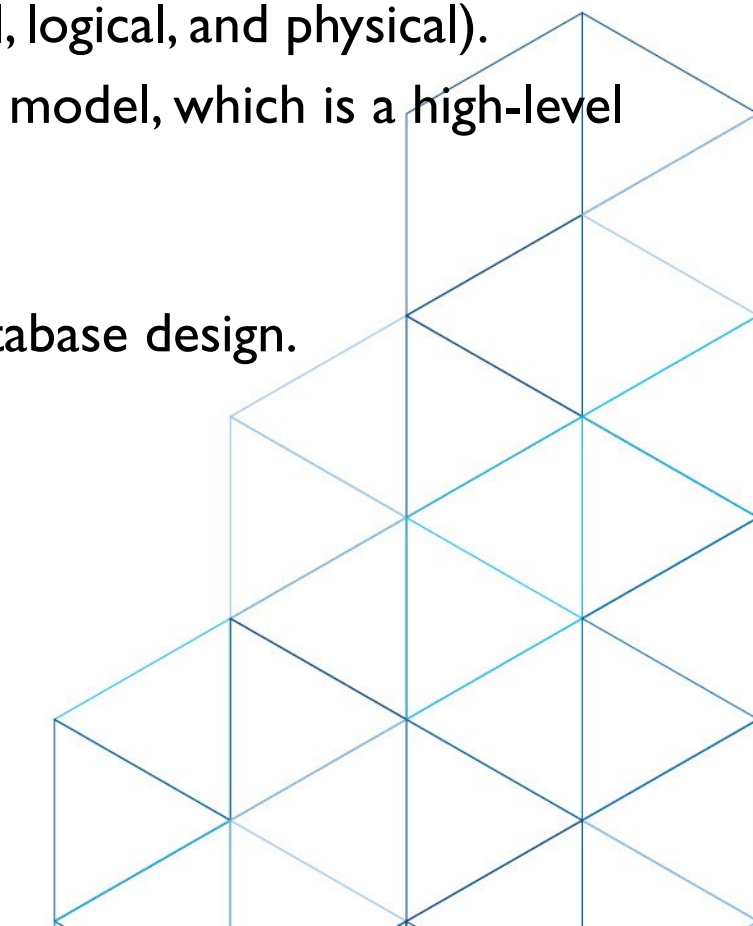
Comments or Questions?



HALMSTAD UNIVERSITY

Objectives

- Introduce and discuss database design
  - Chapter 10 presents an overview of the main stages of the database system development lifecycle, emphasizing the importance of database design (conceptual, logical, and physical).
  - Chapter 12 introduces the concepts of the Entity–Relationship (ER) model, which is a high-level conceptual data model technique in database design.
  - Chapter 16 presents a step-by-step methodology for conceptual database design.
    - *Useful for Laboratory exercise 1.*



…more specifically

- Introduce database design.
  - Conceptual, logical and physical design.
- Introduce types of database design
  - Database design from for new systems (from requirements to model)
  - Database design from existing data (contained in files, spreadsheets, databases)
  - Database design from database redesign (migration, integration, corrections and changes)
- Understand the data modeling/database design process
- Understand the entity-relationship (ER) data model and ER diagrams
  - Entities, attributes, and relationships

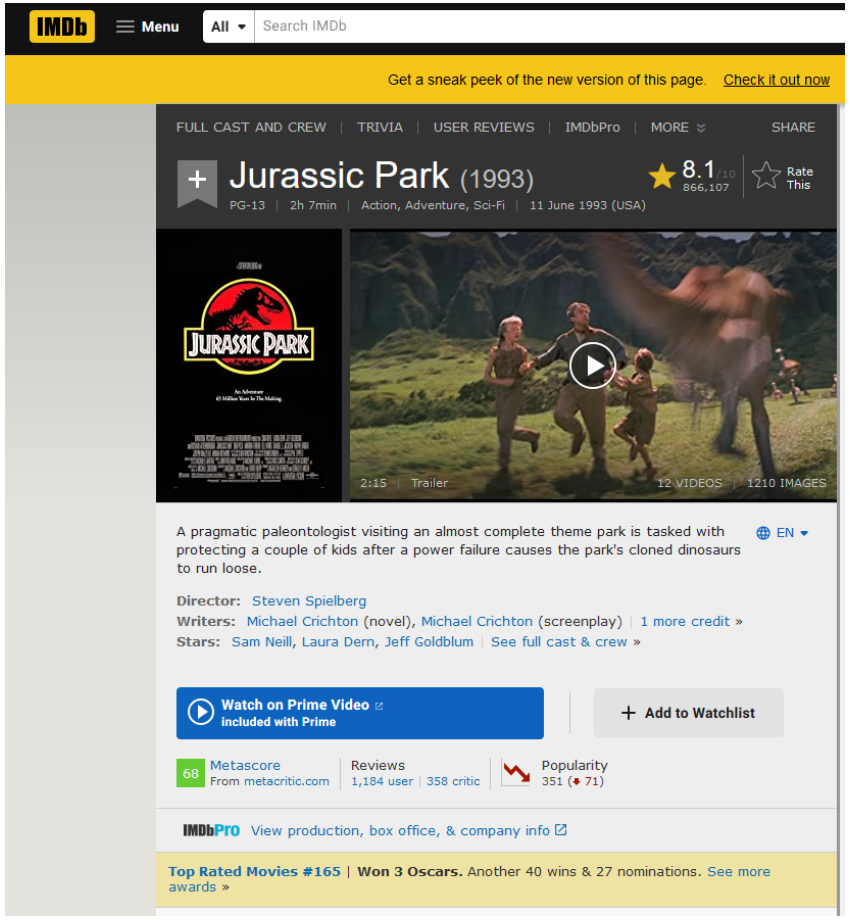
…more specifically

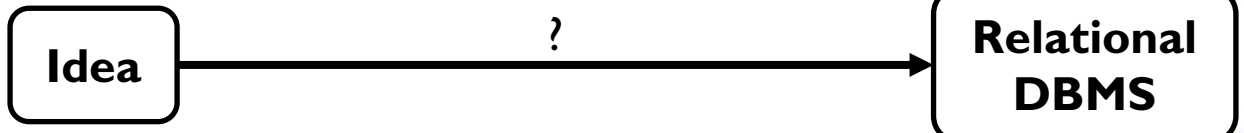
- SW Dev. Life Cycle
- DB Dev. Life Cycle
- Database Design
  - Database Design
    - For a New System
    - From Existing Data
    - From Database Redesign
  - Conceptual database design
  - Logical database design for the relational model
  - Physical database design for relational databases

- ER diagram
  - Entity
    - Strong vs weak entity
  - Attributes
    - Attribute domain
    - Simple or composite attributes
    - Single-valued or multi-valued attributes
    - Derived attribute
    - Keys
      - Unique Identifiers: Primary Key, Candidate Key, Alternate Key, Super Key, Natural Key
      - Relationships Between Tables: Foreign Key
      - Composite Keys
      - Artificial: Surrogate Key
  - Relationships
    - Types
      - Degree, Recursive, Role Names
    - Attributes
    - Structural Constraints
      - Cardinality: 1:1, 1:N, M:N
      - Participation: Total vs. Partial Participation

HALMSTAD UNIVERSITY

Internet Movie Database (IMDb)





HALMSTAD UNIVERSITY

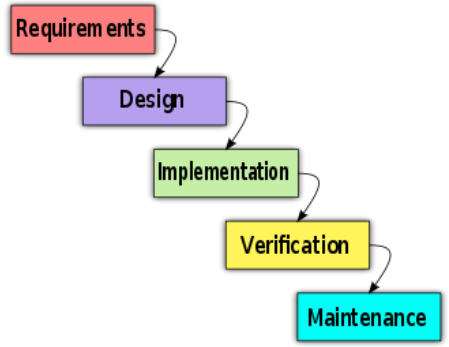
Manufacturing system

- A manufacturing company produces products. The following product information is stored: product name, product ID and quantity on hand. These products are made up of many components. Each component can be supplied by one or more suppliers. The following component information is kept: component ID, name, description, suppliers who supply them, and products in which they are used. Use Figure B.1 for this exercise.
- Assumptions
  - A supplier can exist without providing components.
  - A component does not have to be associated with a supplier.
  - A component does not have to be associated with a product. Not all components are used in products.
  - A product cannot exist without components.

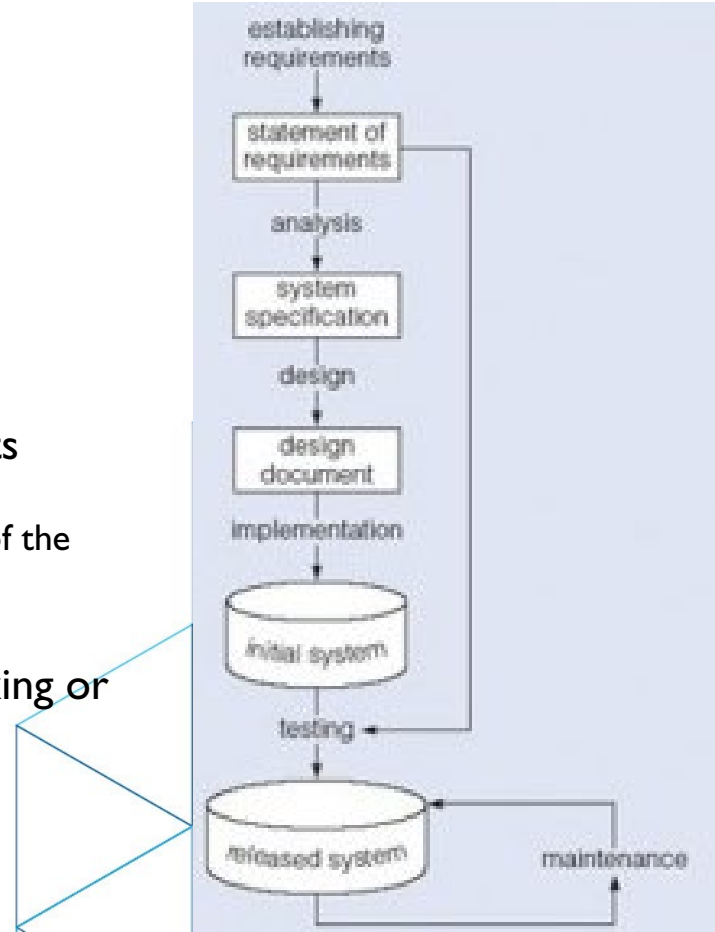
Property Rental System

- A housing agency currently rents out various types of domestic accommodations, including apartments and houses, to clients. The clients, such as students or groups of students, may require one-year leases or, in the case of families, longer-term rentals.
- Details are maintained for each property, including:
  - Address (street, city, postal code, country)
    - Owner details (document number, name, address, telephone)
    - Lead tenant details (document number, name, telephone)
  - Tenancy start date
  - Tenancy end date
  - Rent amount
  - Type of property (e.g., apartment, detached house, terrace house)
  - Furnished/Unfurnished status (for furnished properties, additional details about furnishings are kept, such as beds and sofas)
  - Number of bedrooms
  - Number of bathrooms
  - Optionally, additional information about the property is:
    - A textual description
    - Photographs
- A history of occupancy, including periods when the property is unoccupied, needs to be maintained.
- To facilitate property searches, it has been decided to store details of individual rooms within a property. This includes:
  - Room type (e.g., bedroom, bathroom, kitchen)
  - Room dimensions (width and length in meters)
  - Heating method (e.g., radiator, fireplace)
  - For kitchens: appliances (e.g., microwave, refrigerator, freezer, cutlery)
  - For bedrooms: furniture (e.g., bed, dresser)
  - For the living room: furniture (e.g., sofa, TV)
  - Any special features (e.g., garage, patio windows)
- The tourist business has started to increase in the area, and the agency plans to expand into holiday lettings. This will involve shorter rental periods and a greater need for up-to-date availability information.

Software Development Life Cycle



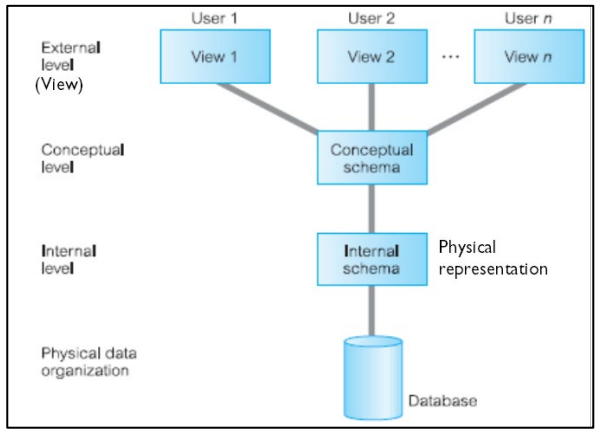
- Series of steps involved in the software development process
  - Establishing requirements
    - What stakeholders want from the system.
  - Analysis
    - Representation of what a system should do (not how it may be realized)
  - Design
    - Described how a system should be constructed.
  - Implementation
    - Construction of a computer system according to a given design
  - Testing
    - Compares the implemented system against the design documents and requirements specification
      - Produces an acceptance report or, more usually, a list of errors and bugs that require a review of the analysis, design and implementation.
  - Maintenance
    - Deal with changes in the requirements or the implementation environment, bug fixing or porting of the system to new environments.

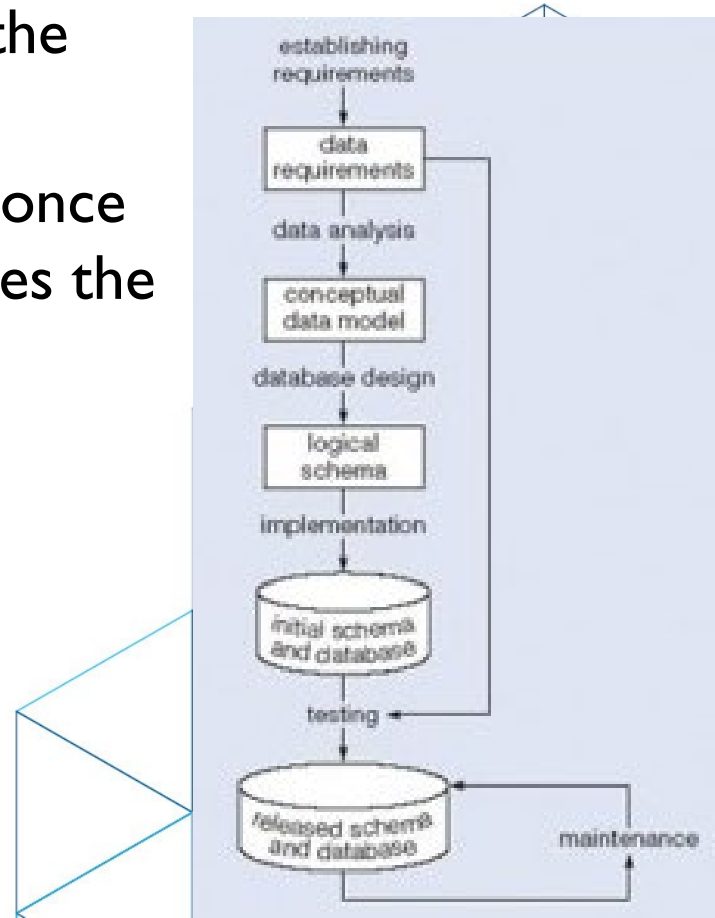


HALMSTAD UNIVERSITY

Database Development Life Cycle

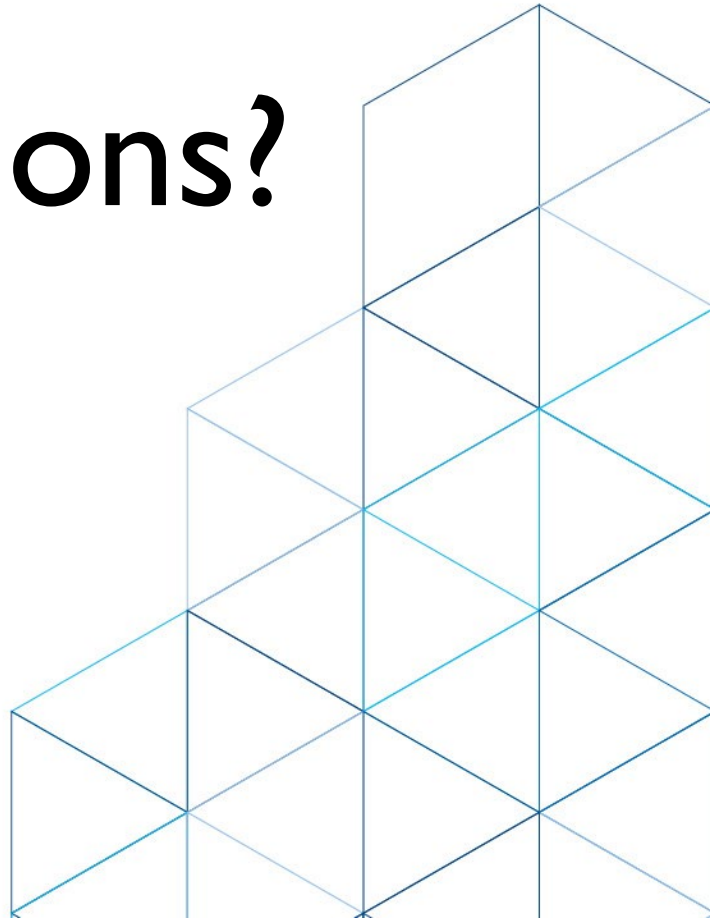
- Separate the development of a database from the user processes that make use of the database.
- Use the three-schema architecture as a basis for distinguishing the activities associated with a schema.
- Represent the constraints to enforce the semantics of the data once within a database, rather than within every user process that uses the data.





HALMSTAD UNIVERSITY

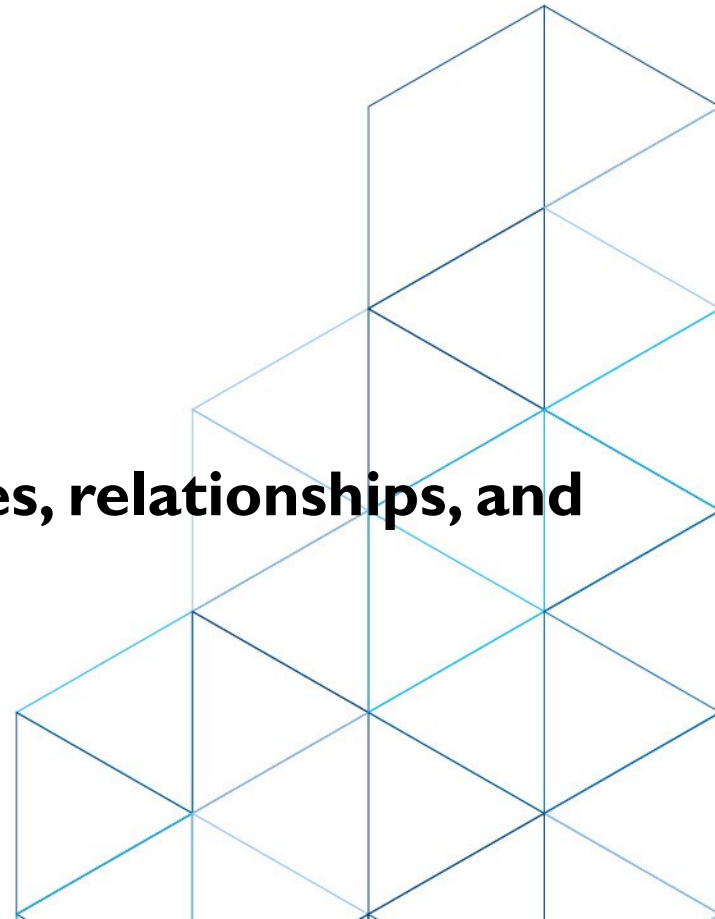
Comments or Questions?



HALMSTAD UNIVERSITY

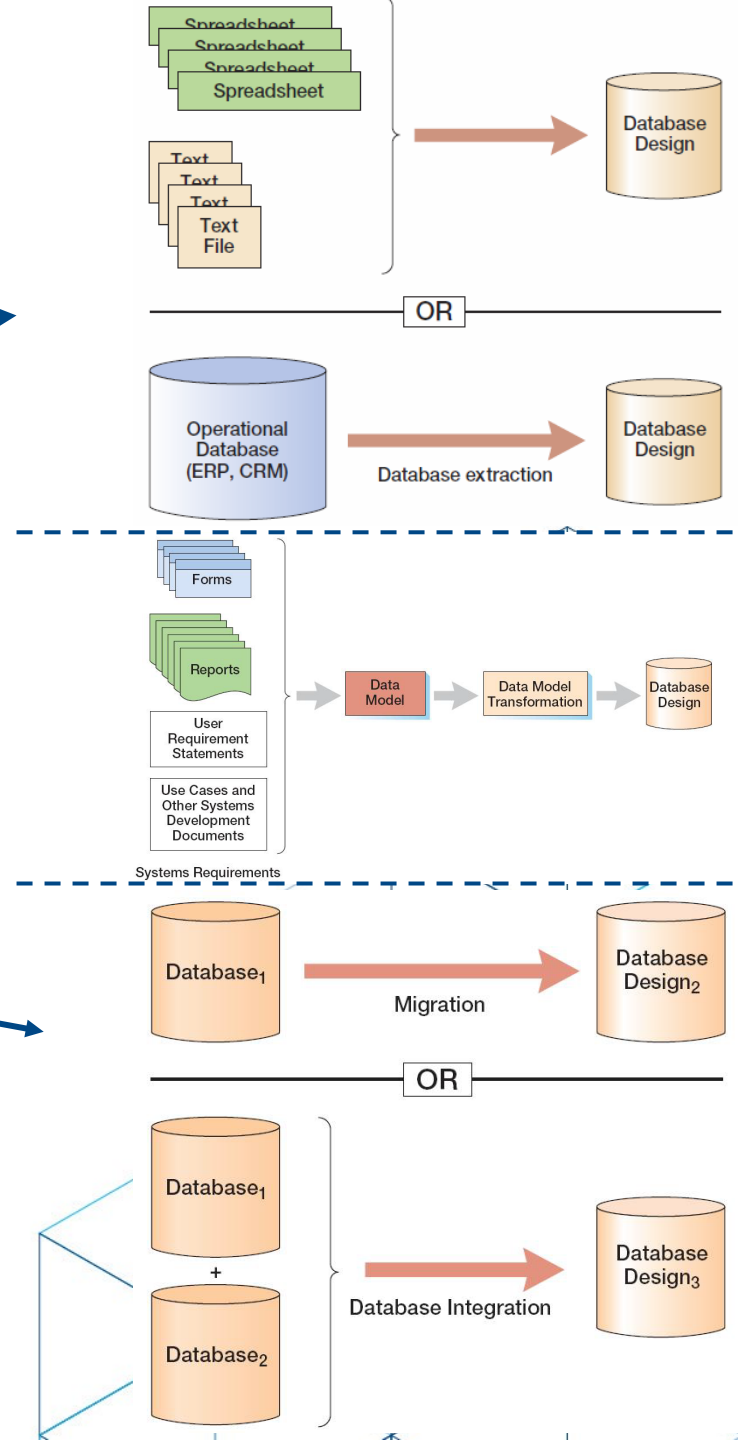
Database Design

- *"Process of creating a design for a database that will support the enterprise's mission statement and mission objectives for the required database system."*
- Main purposes of data modeling include:
  - to assist in **understanding** the meaning (semantics) of the data.
    - Each user's perspective of the data;
    - Nature of the data itself, independent of its physical representations;
    - Use of data across user views.
  - to facilitate **communication** about the information requirements.
- Building data model requires answering questions about **entities, relationships, and attributes.**



Three Types of Database Design

- Database Design from Existing Data
  - Analyze spreadsheets and other data tables
  - Extract data from other databases
  - Design using normalization principles
- Database Design from New Systems Development
  - Create a data model from application requirements
  - Transform data model into database design
- Database Design from Database Redesign
  - Migrate databases to newer databases
  - Integrate two or mode databases
  - Reverse engineer and design new databases using normalization principles and data model transformations



Database Design (Main) Approaches

1. Top-down
   - Involves the identification of entities and relationships between the entities, and finally the associated attributes.
   - Appropriate strategy for the design of complex databases.
   - Entity-Relationship (ER) model
2. Bottom-up
   - Involves the identification of the required attributes and their subsequent aggregation into relations.
   - Appropriate for the design of simple databases with a relatively small number of attributes.
3. Inside-out
   - Related to the bottom-up approach but involves the identification of a set of major entities first.
4. Mixed
   - Uses both the bottom-up and top-down approaches.

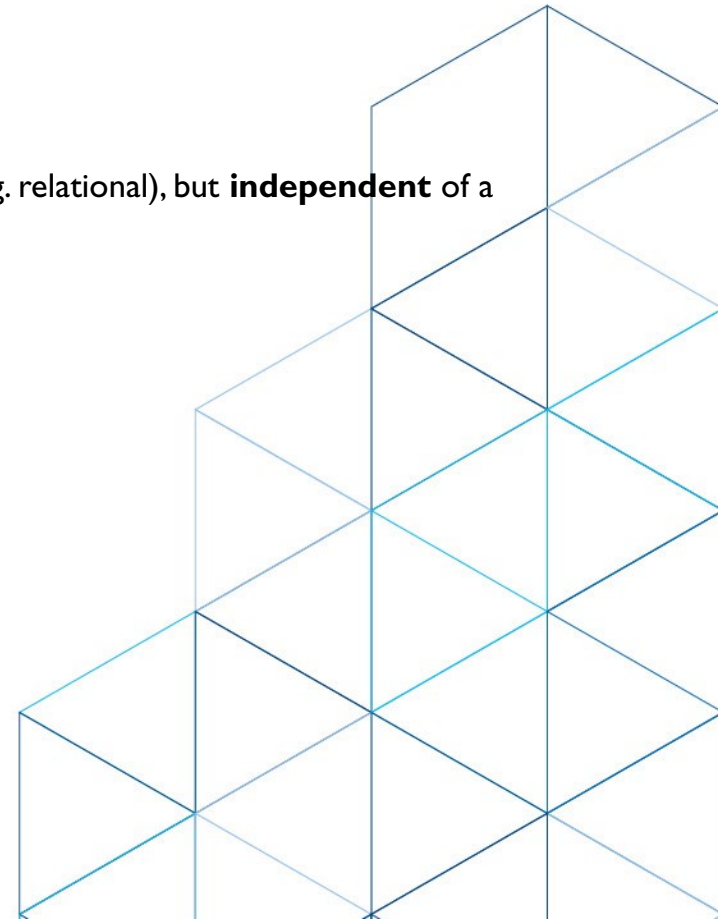
Database Development Process

- Requirements Gathering
  - Understand the proposed system.
- Conceptual Design
  - Produces a conceptual data model that describes in detail what data the database will contain and the constraints the data must satisfy.
  - Independent of all physical considerations, i.e. of the DBMS that will be used.
    - It is about "What is required?", not "How is it achieved?"
- Logical Design
  - Specification based on a specific data model (e.g. relational), but independent of a particular DBMS and other physical considerations.
    - Specification of all the tables and constraints satisfying the conceptual data model.
- Physical Design
  - Adapt the logical design to a particular DBMS.
    - Construction of a database according to the specification of a logical schema.
  - Can change slightly to fit into the limitations of a DBMS or to take advantage of DBMS-specific features, i.e. tailored to a specific DBMS system.
  - Populating the Database.

Three phases of database design

Database Design process

- Conceptual database design
  - The process of constructing a **model of the data** used in an enterprise, **independent** of all **physical** considerations.
    1. Identify entity types, relationship types and attributes.
    2. Determine attribute domains
    3. Determine candidate, primary, and alternate key attributes
    4. Check model for redundancy
    5. Validate conceptual data model against user transactions
    6. Review conceptual data model with user
- Logical database design for the relational model
  - The process of constructing a **model of the data** used in an enterprise **based on a specific data model** (e.g. relational), but **independent** of a particular **DBMS** and other **physical** considerations.
    1. Derive relations for logical data model
    2. Validate relations using normalization
    3. Validate relations against user transactions
    4. Check integrity constraints
    5. Review logical data model with user
- Physical database design for relational databases
    1. Translate logical data model for target DBMS
       - Design base relations, representation of derived data and general constraints
    2. Design file organizations and indexes
    3. Design user views
    4. Design security mechanisms
    5. Consider the introduction of controlled redundancy
    6. Monitor and tune the operational system



HALMSTAD UNIVERSITY

Critical Success Factors in Database Design

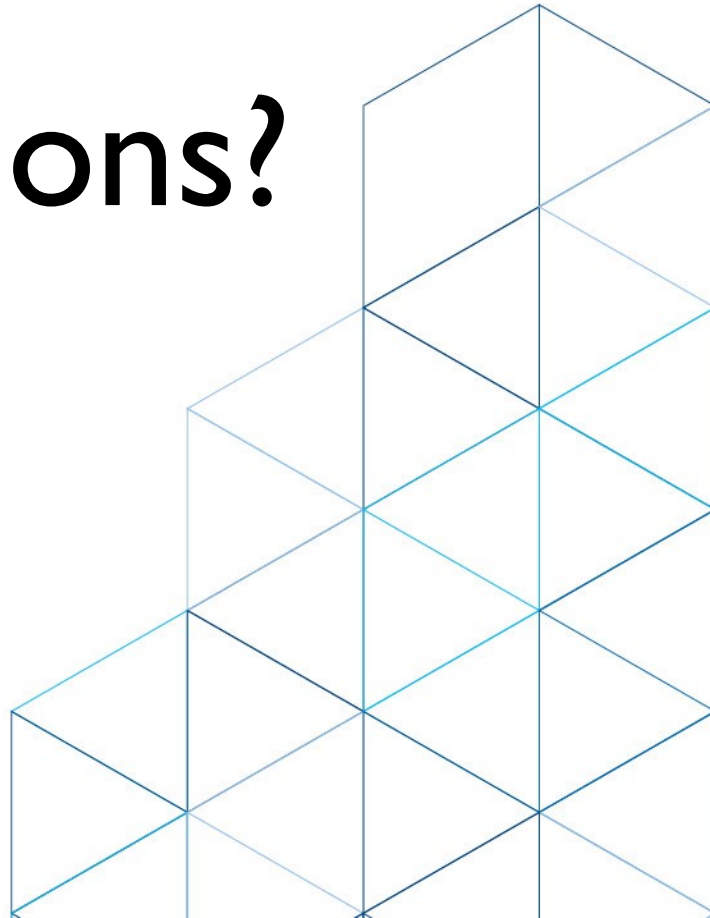
- Work interactively with the users as much as possible.
- Follow a structured methodology throughout the data modeling process.
- **Employ a data-driven approach.**
- Incorporate structural and integrity considerations into the data models.
- **Combine conceptualization, normalization, and transaction validation techniques into the data modeling methodology**.
- **Use diagrams to represent as much of the data models as possible.**
- Use a Database Design Language (DBDL) to represent additional data semantics.
- Build a data dictionary to supplement the data model diagrams.
- Be willing to repeat steps.

```
    Domain PropertyNumber:        variable length character string, length 5
    Domain Street:                variable length character string, length 25
    Domain City:                  variable length character string, length 15
    Domain Postcode:              variable length character string, length 8
    Domain PropertyType:          single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'
    Domain PropertyRooms:         integer, in the range 1–15
    Domain PropertyRent:          monetary value, in the range 0.00–9999.99
    Domain OwnerNumber:           variable length character string, length 5
    Domain StaffNumber:           variable length character string, length 5
    Domain BranchNumber:          fixed length character string, length 4
PropertyForRent(
    propertyNo    PropertyNumber  NOT NULL,
    street        Street          NOT NULL,
    city          City            NOT NULL,
    postcode      Postcode,
    type          PropertyType    NOT NULL DEFAULT 'F',
    rooms         PropertyRooms   NOT NULL DEFAULT 4,
    rent          PropertyRent    NOT NULL DEFAULT 600,
    ownerNo       OwnerNumber     NOT NULL,
    staffNo       StaffNumber,
    branchNo      BranchNumber    NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,
    FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
                                  ON UPDATE CASCADE ON DELETE NO ACTION,
    FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)
                                  ON UPDATE CASCADE ON DELETE NO ACTION);
```

Entity name	*Description*	*Aliases*	*Occurrence*
Staff	General term describing all staff employed by *DreamHome*.	Employee	Each member of staff works at one particular branch.
PropertyForRent	General term describing all property for rent.	Property	Each property has a single owner and is available at one specific branch, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.

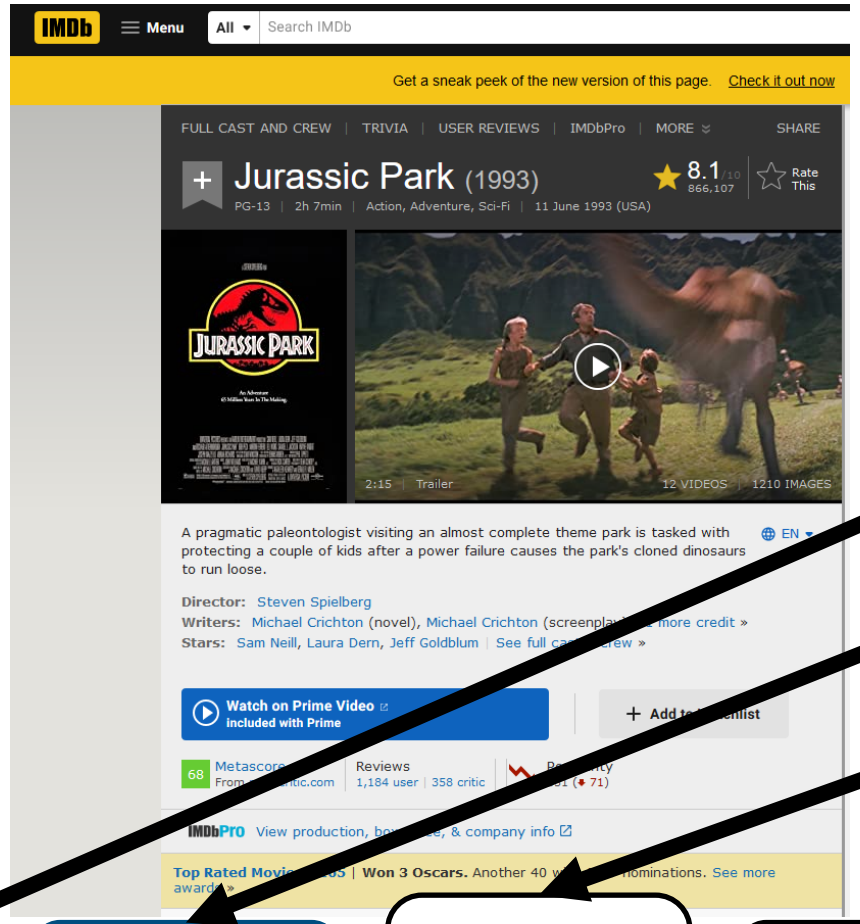
HALMSTAD UNIVERSITY

Comments or Questions?



HALMSTAD UNIVERSITY

IMDB Database Development Process



HALMSTAD UNIVERSITY

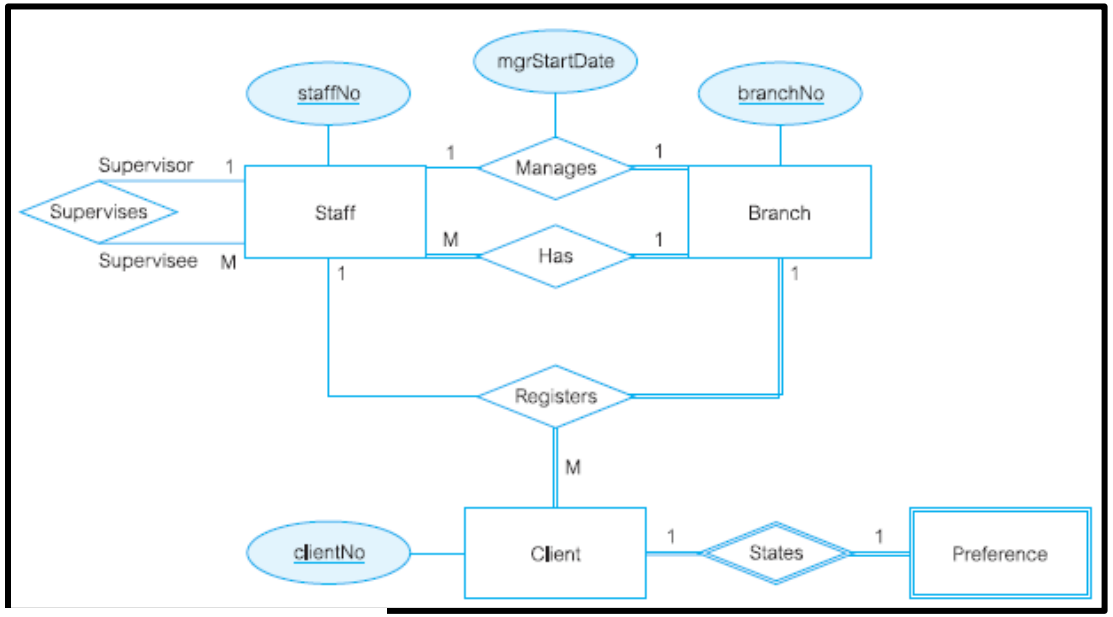
Entity-Relationship (ER) modeling in database design

- Top-down approach to database design.
- Start by identifying the important data (called entities) and relationships between the data.
- Then add more details such as the information (called attributes) we want to hold about the entities and relationships, and any constraints on the entities, relationships, and attributes.

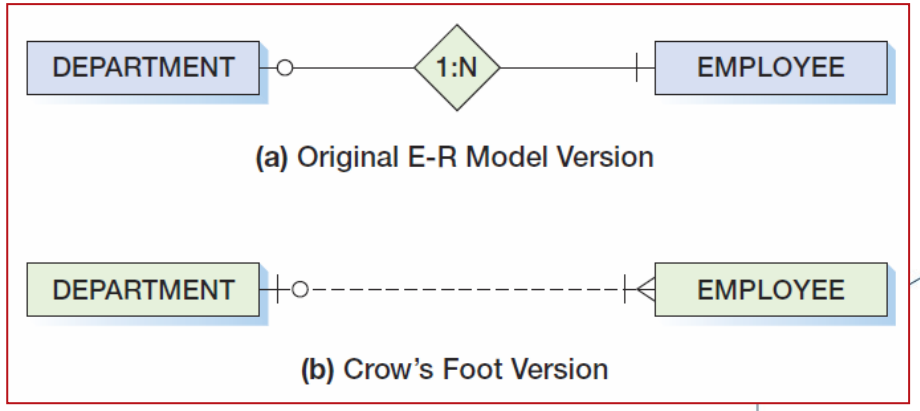
- It is about designing a database by sketching it – an Entity-Relationship diagram
  - Assists in understanding the meaning of the data.
  - Facilitates communication with users.

Entity-Relationship (ER) diagram

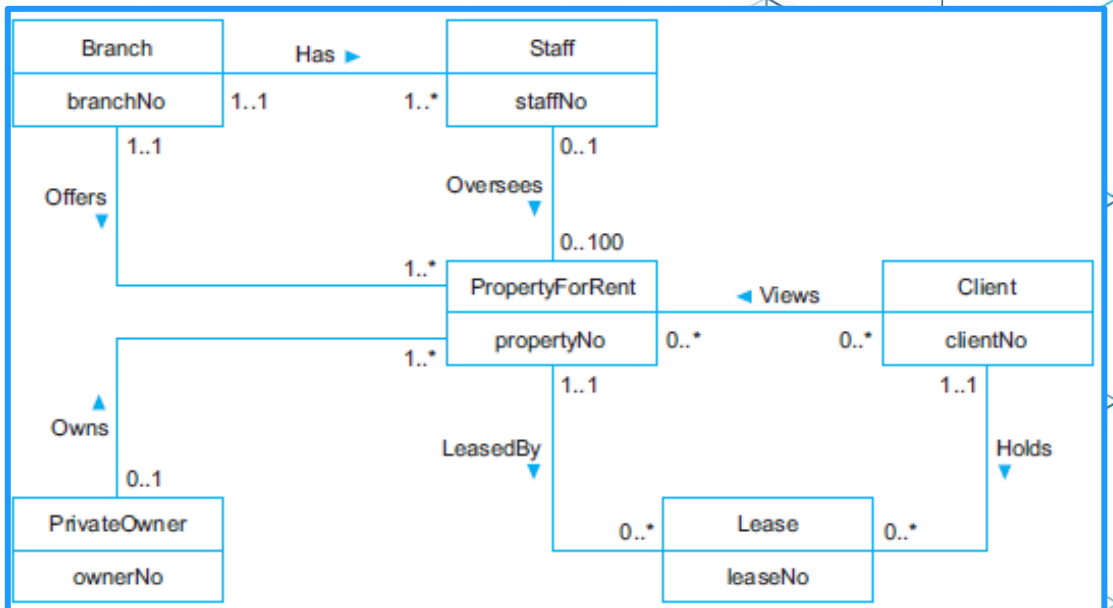
- Set of concepts and graphical symbols that can be used to create conceptual schemas.
- Versions:
  - Original E-R model by Peter Chen (1976)
  - Extended E-R model is an extension to the Chen model
  - Information Engineering (IE) by James Martin (1990); uses "crow's foot" notation, is easier to understand.
  - Unified Modeling Language (UML) by the Object Management Group



Chen notation

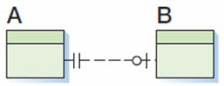
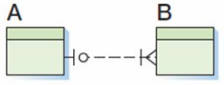
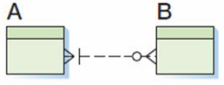
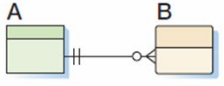


UML (Unified Modeling Language) notation

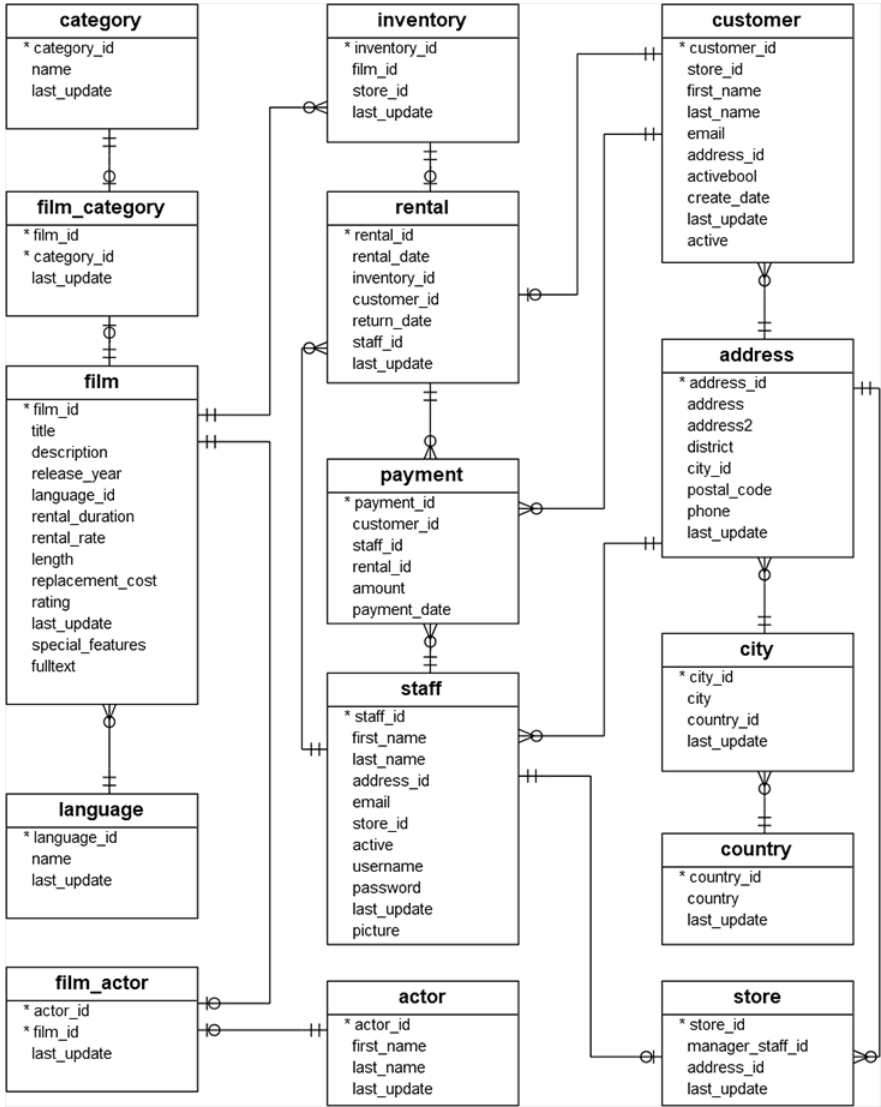


Entity-Relationship (ER) diagram

- James Martin's
  "crow's foot" notation

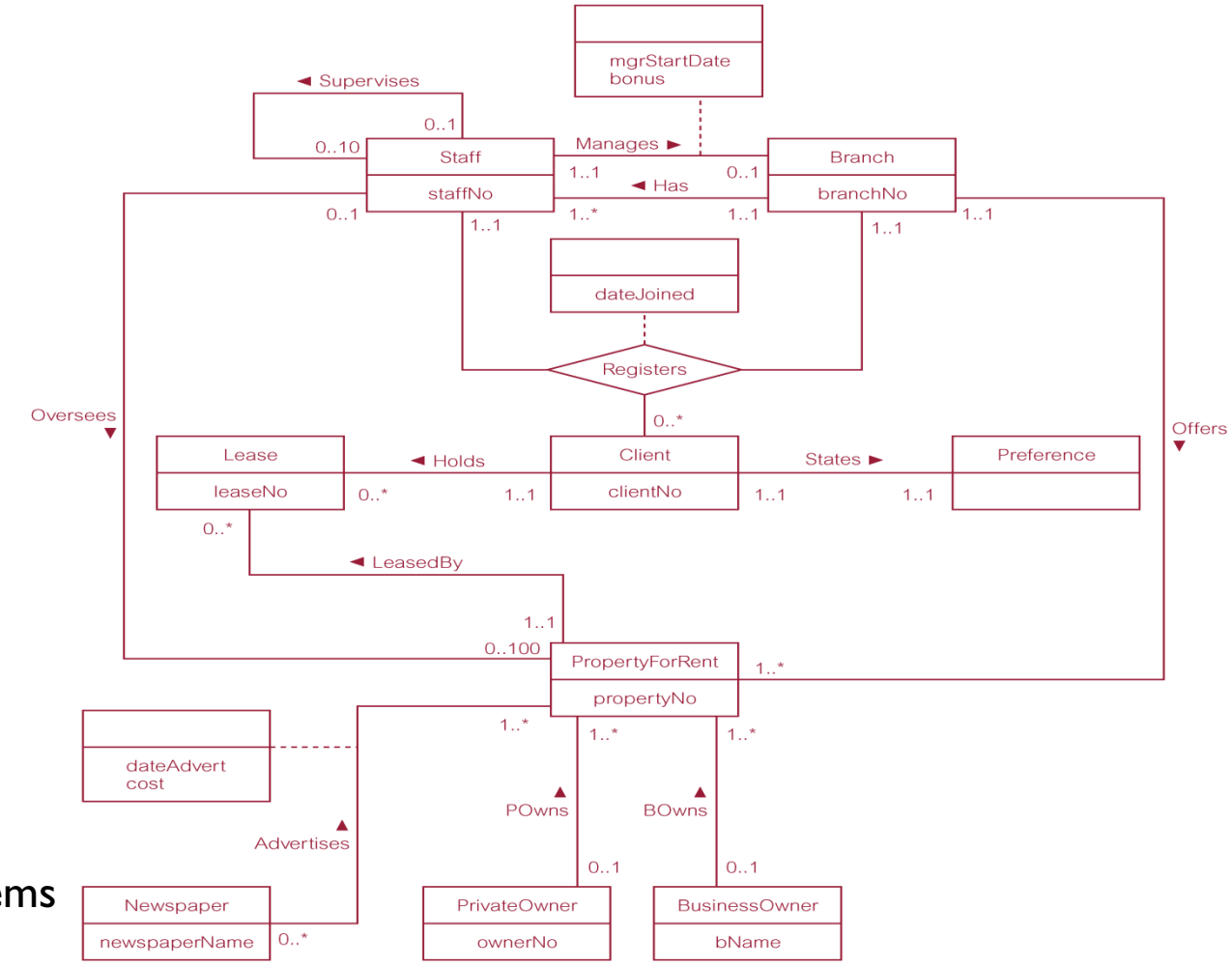
DEPARTMENT DepartmentName BudgetCode OfficeNumber	DEPARTMENT entity; DepartmentName is identifier; BudgetCode and OfficeNumber are attributes.
	1:1, nonidentifying relationship. A relates to zero or one B; B relates to exactly one A. Identifier and attributes not shown.
	1:N, nonidentifying relationship. A relates to one or many Bs; B relates to zero or one A. Identifier and attributes not shown.
	Many-to-many, nonidentifying relationship. A relates to zero or more Bs; B relates to one or more As.
	1:N identifying relationship. A relates to zero, one, or many Bs. B relates to exactly one A. Identifier and attributes not shown. For identifying relationships, the child must always relate to exactly one parent. The parent may relate to zero, one, many, or a combination of these minimum cardinalities.

https://www.postgresqltutorial.com/postgresql-sample-database/



Example ER diagram

- The coursebook* illustrates concepts using a case study based on a fictitious property management company called DreamHome.



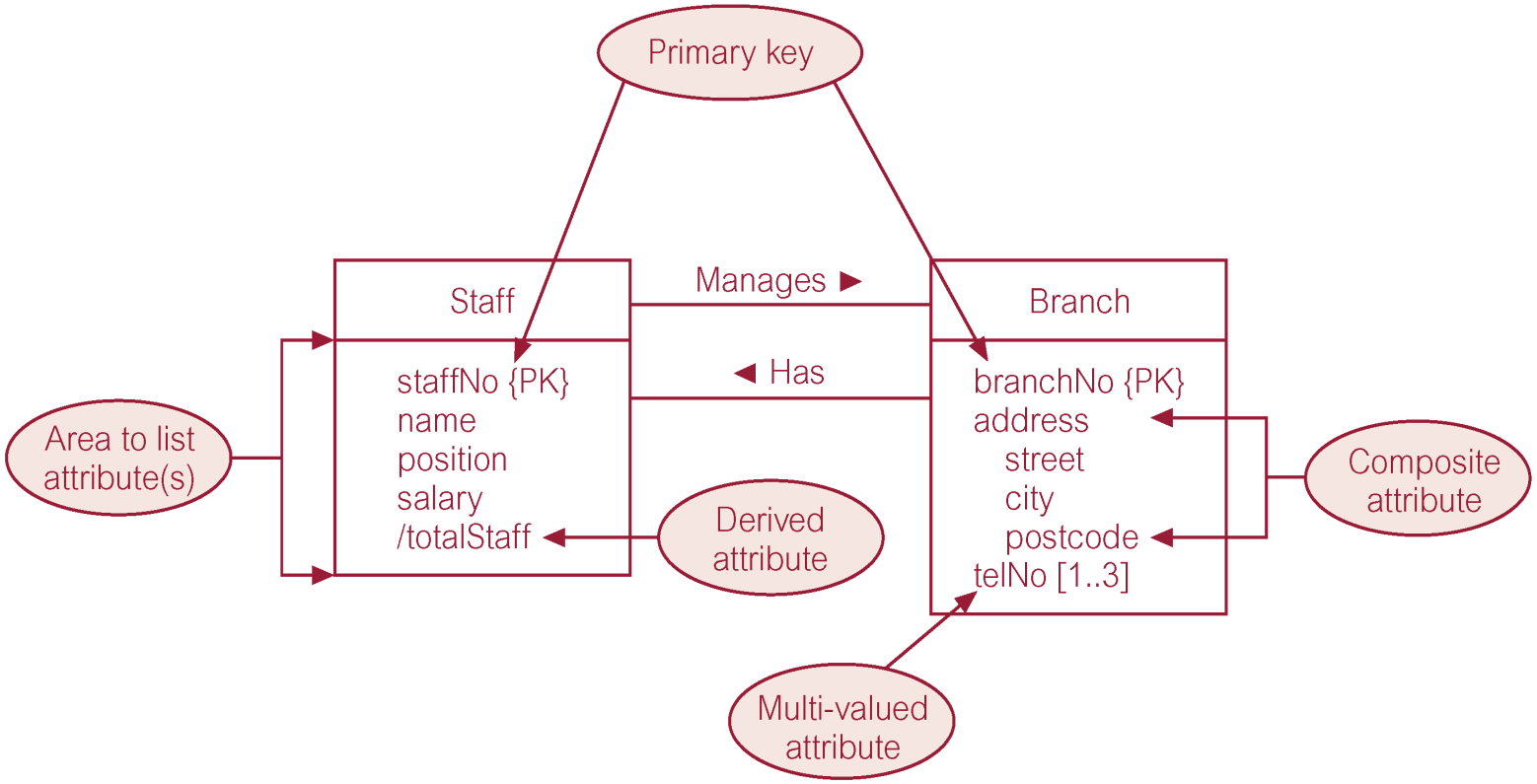
Entity–Relationship (ER) diagram of the Branch view of DreamHome

* 2015 - Connolly & Begg - Database Systems

HALMSTAD UNIVERSITY

Concepts of the ER Model

- Entity
  - Set (class or type) vs instance
  - Strong vs weak entity
- Attributes
  - Attribute domain
  - Simple or composite attributes
  - Single-valued or multi-valued attributes
  - Derived attribute
  - Keys
    - Unique Identifiers: Primary Key, Candidate Key, Alternate Key, Super Key, Natural Key
    - Relationships Between Tables: Foreign Key
    - Composite Keys
    - Artificial: Surrogate Key
- Relationships
  - Types
    - Degree, Recursive, Role Names
  - Attributes
  - Structural Constraints
    - Cardinality: 1:1, 1:N, M:N
    - Participation: Total vs. Partial Participation



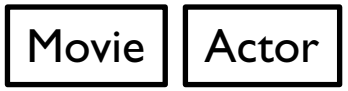
HALMSTAD UNIVERSITY

Entity

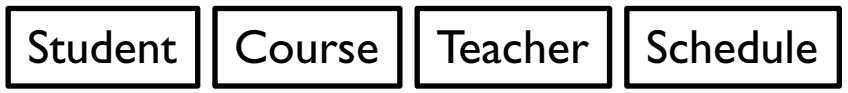
- An identifiable "real world" object or thing that users want to track.
  - A collection of entities of a given type form an **entity set**.
  - The occurrence of a particular entity is an **entity instance.**

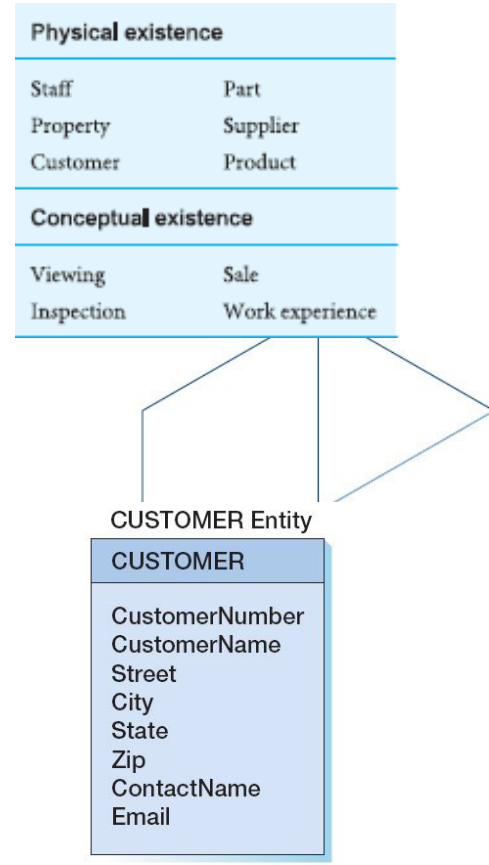
- Entity set (Entity class or Entity type)
  - A collection of similar entities, i.e. with same properties
  - Each entity set has a **key.**
  - Each attribute has a **domain.**

- Entities are drawn as **rectangles** in an ER diagram.
- Examples
  - In the IMDB database



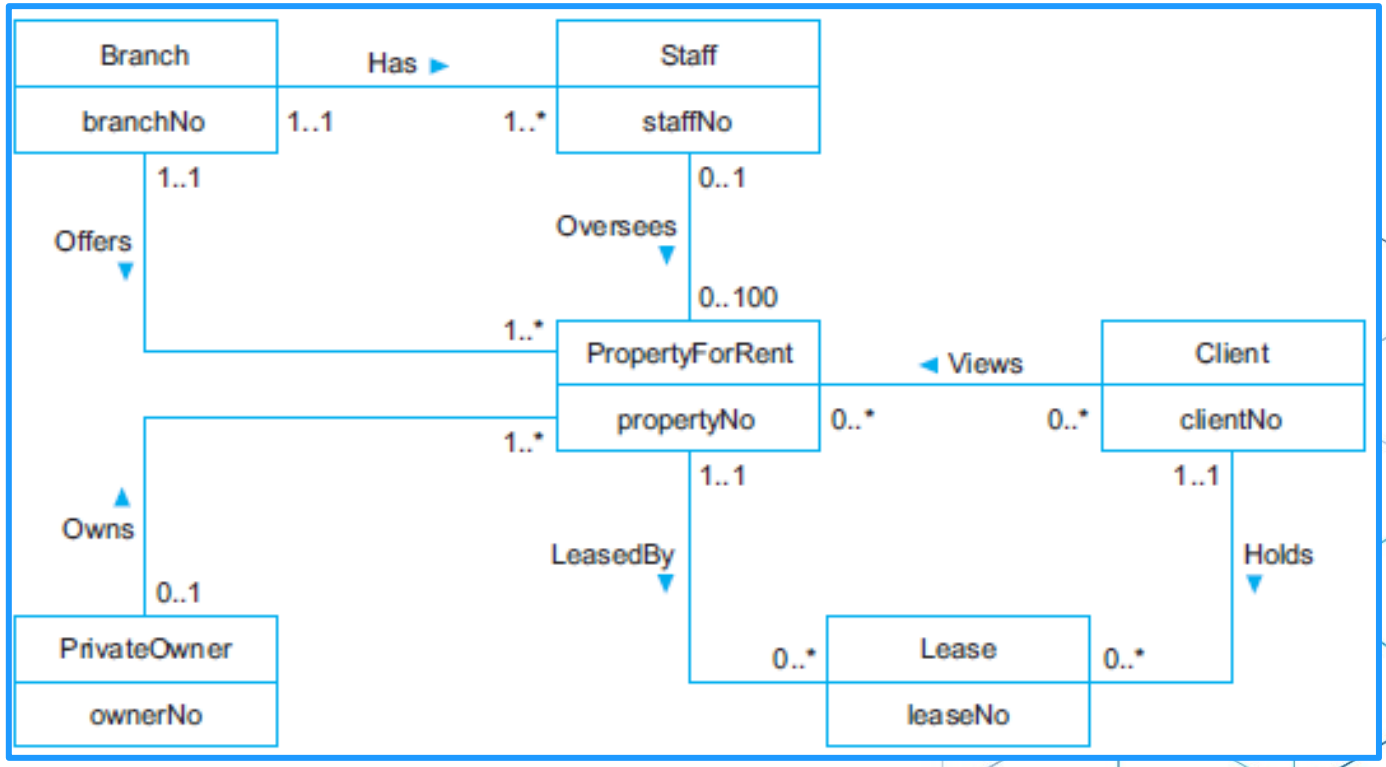
  - In an education database





Entities in an ER diagram
How many?

- Six entities (the rectangles): Branch, Staff, PropertyForRent, Client, PrivateOwner, and Lease;

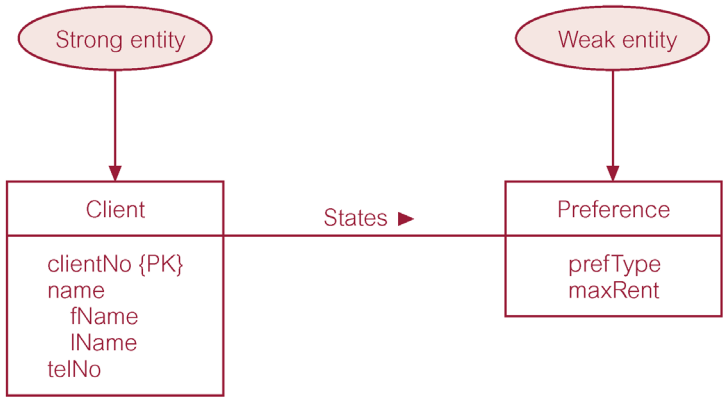


HALMSTAD UNIVERSITY

Strong and Weak Entities

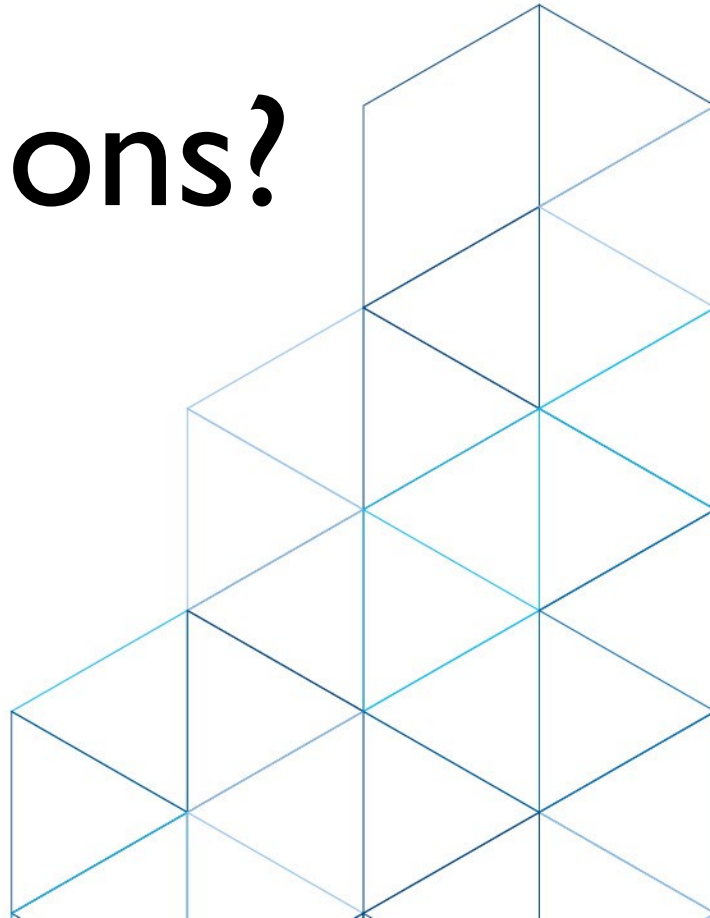
- Strong entity
  - Entity that is not dependent on the existence of another entity for its primary key.
- Weak entity
  - Entity that is partially or wholly dependent on the existence of another entity, or entities, for its primary key.
  - Does not have any key attribute of its own.
  - In the Chan notation, if an entity set is weak, it will be shown as a rectangle with a double border.

Strong entity type called Client and weak entity type called Preference



HALMSTAD UNIVERSITY

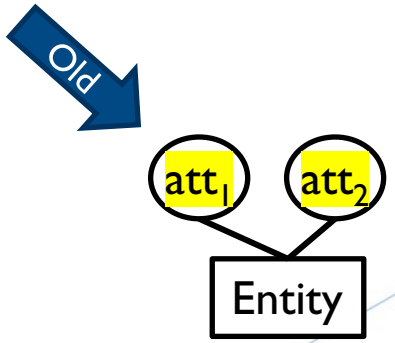
Comments or Questions?

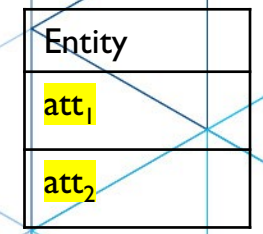


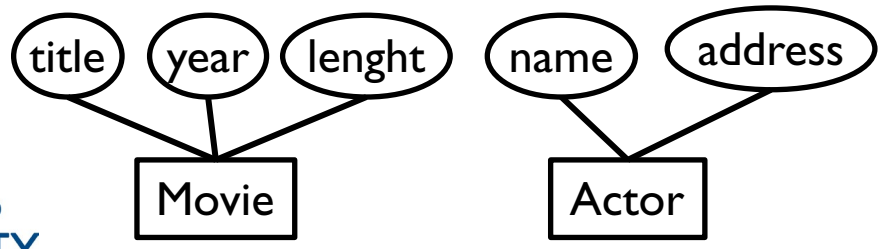
HALMSTAD UNIVERSITY

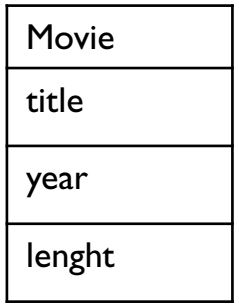
Attributes

- Entity sets have associated attributes.
- **Attributes are descriptive properties of the entities in that set.**
  - Hold values that describe each occurrence of an entity.
  - Represent the main source of data stored in the database.
  - Each attribute has a **domain** (string, integer, date, …).
- Originally drawn as ovals connected to the entity by a line,
  - Now, modeling tools show attributes in rectangular form.
- Examples
  - In the IMDB database





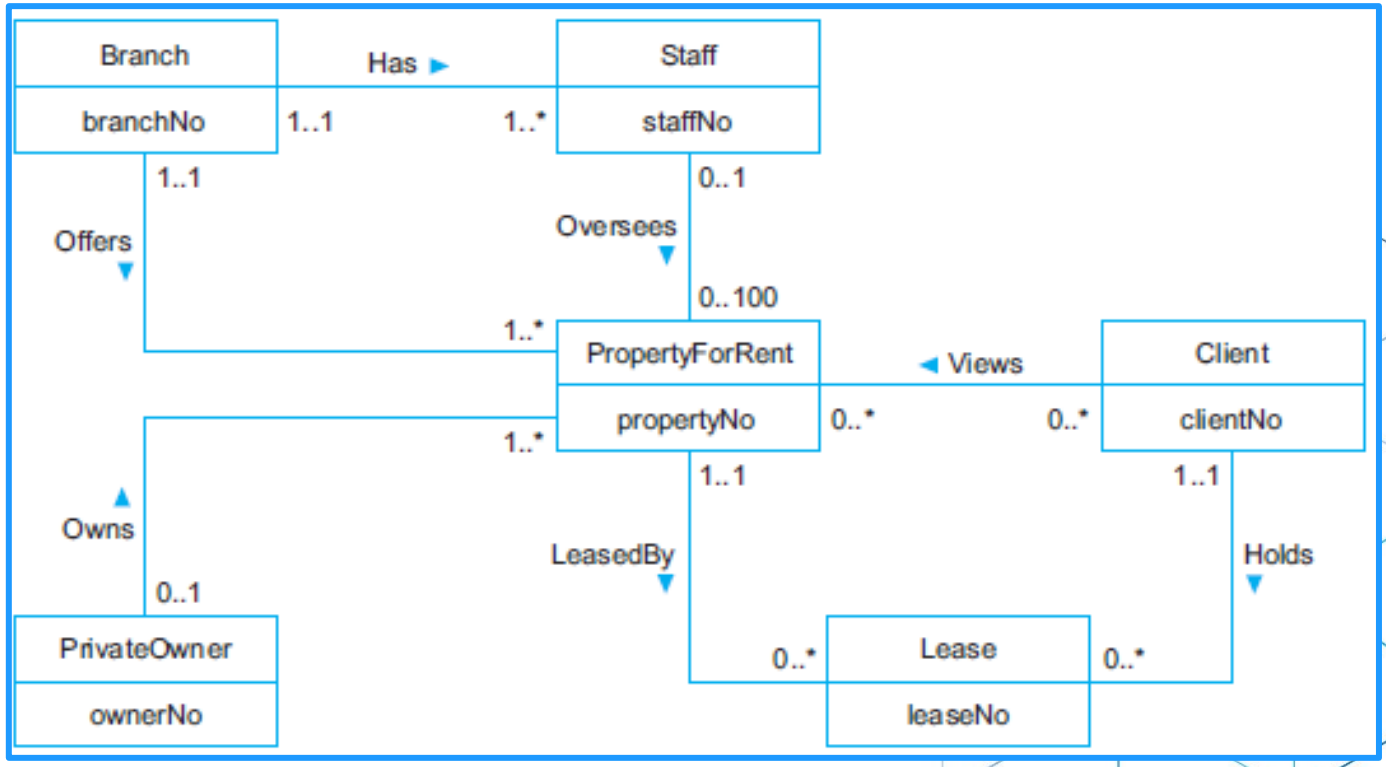




HALMSTAD UNIVERSITY

Attributes in an ER diagram
How many?

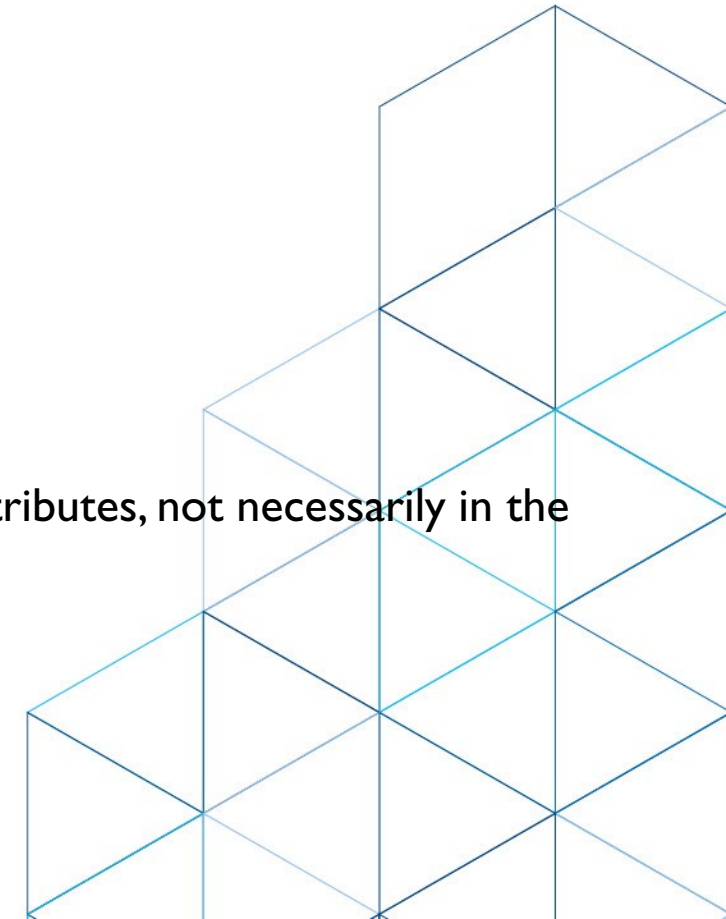
- Six entities (the rectangles): Branch, Staff, PropertyForRent, Client, PrivateOwner, and Lease;
- Six attributes, one for each entity: branchNo, staffNo, propertyNo, clientNo, ownerNo, and leaseNo.



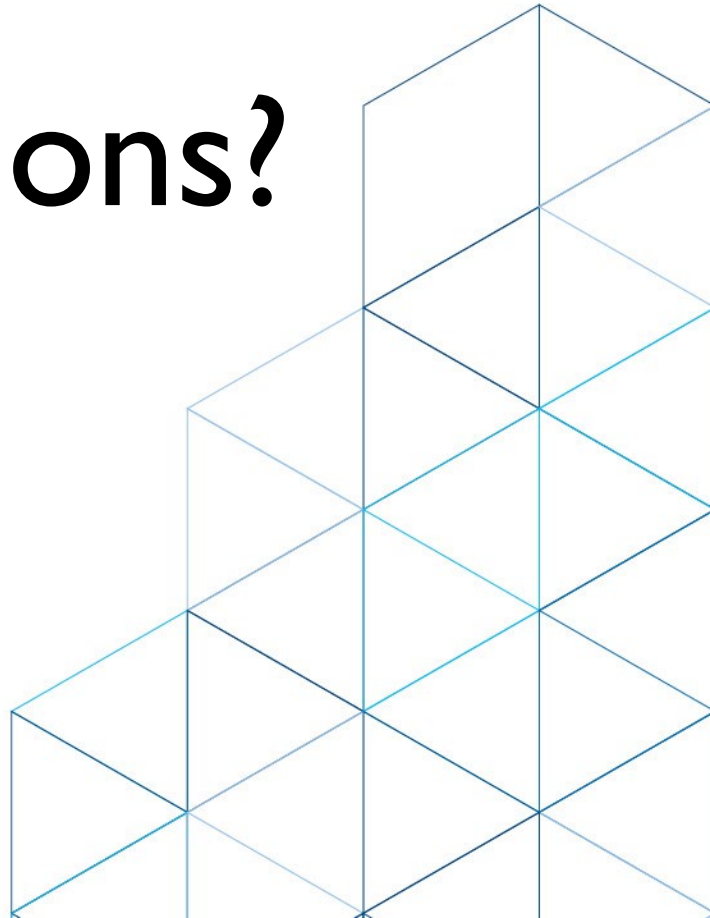
HALMSTAD UNIVERSITY

Classifying Attributes

- Simple or composite
  - A simple attribute is composed of a single component.
    - *Year, title, salary, birth date…*
  - A composite attribute is composed of multiple components.
    - *Name or full name (first name + last name), address (street, city, postcode)*
- Single-valued or multi-valued
  - A single-valued attribute holds a single value for an entity occurrence.
    - Most attributes are single-valued
    - *Course id, person nummer, birth date*
  - A multi-valued attribute holds multiple values for an entity occurrence.
    - A store (entity) can have *multiple telephone numbers* (+4671.., +4672..)
- Derived attribute
  - Represents a value that is derivable from value of a related attribute, or set of attributes, not necessarily in the same entity.
  - *Rent duration*, which can be calculated (derived) from rent start and rent finish.
  - *Age*, which can be calculated (derived) from birth date.



Comments or Questions?

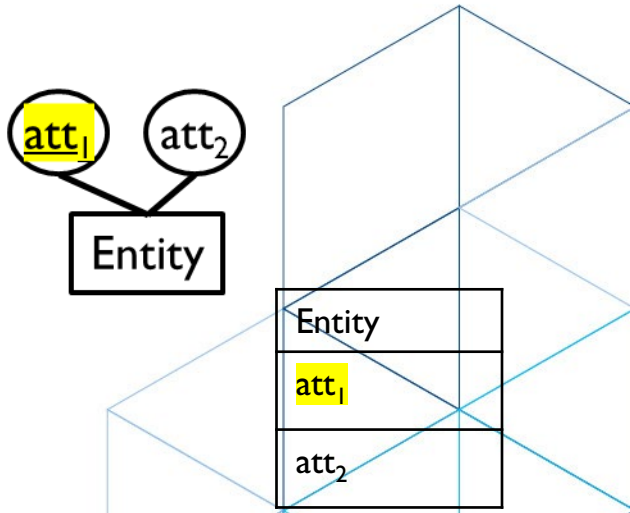


HALMSTAD UNIVERSITY

Keys

- When you have your preliminary entities and attributes defined, you can start thinking about keys.
  - Because you want to distinguish entities (instances) of an entity type.
- A key is a combination of one or more attributes that is used to identify entities in an entity set.

- There are several types of keys:
  - Super Keys
    - An attribute, or set of attributes, that uniquely identifies an entity in an entity set.
  - Candidate Keys
    - Minimal set of attributes that uniquely identifies each occurrence of an entity type.
  - Primary Keys
    - Candidate key selected to **uniquely identify** each occurrence of an entity type.
    - The ideal primary key is short, numeric, and never changes.
  - Composite Keys
    - A candidate key that consists of two or more attributes.
  - Natural Keys
    - An attribute that "naturally" belongs to the entity, such as a student ID or a phone number.
  - Surrogate Keys
    - Keys that have no business meaning and are often integers incremented row by row.
    - They can also be things such as time stamps of auto-generated GUIDs.



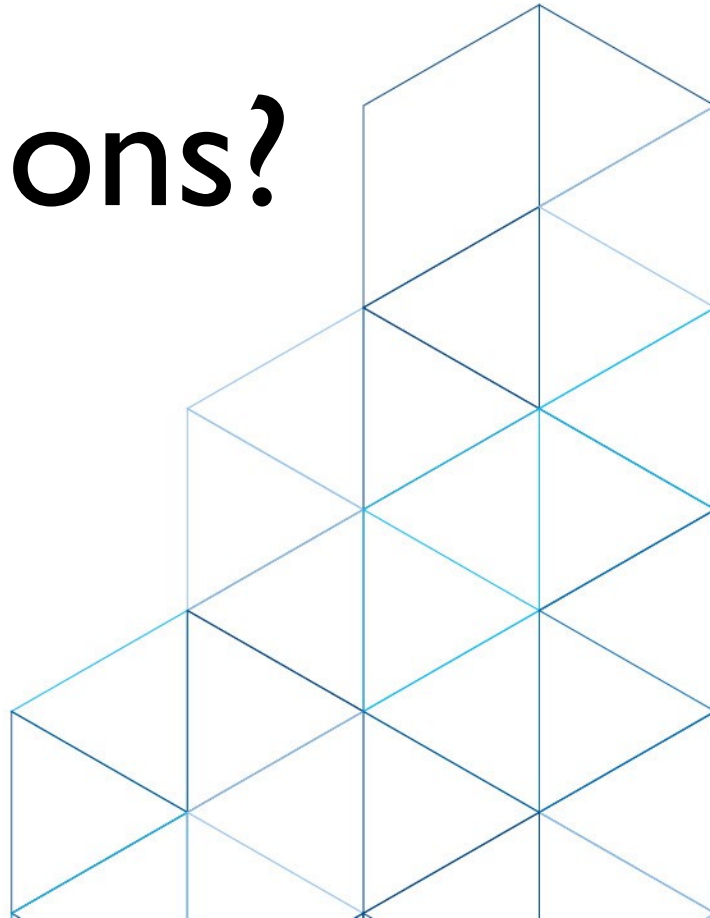
HALMSTAD UNIVERSITY

Natural vs Surrogate Keys

- **Natural Key**
  - Advantages
    - Protect better against accidentally repeating the same information.
    - Belong to the entity and tend to make foreign keys more understandable.
  - Disadvantages
    - Hard to guarantee their uniqueness.
    - May require clumsy composite keys consisting of several attributes to be unique.
    - Subject to changes in business rules (think of Social Security numbers, for instance).

- **Surrogate Key**
  - Advantages
    - Are always unique.
    - Do not contain any business logic and are therefore not subject to changes in business rules.
    - Are easier to define and use.
  - Disadvantages
    - Automatically grant uniqueness to a row, making it easier to accidentally insert the same information twice.
    - No relation to the data, making database relations less readable.

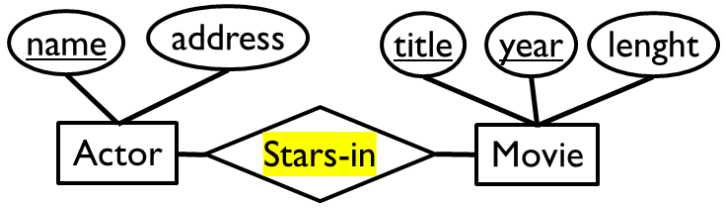
Comments or Questions?

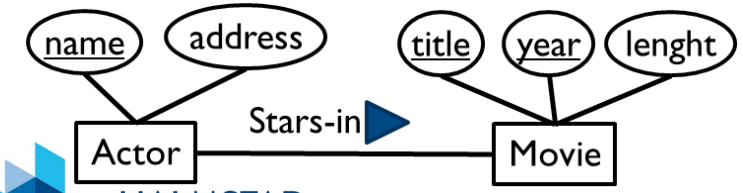


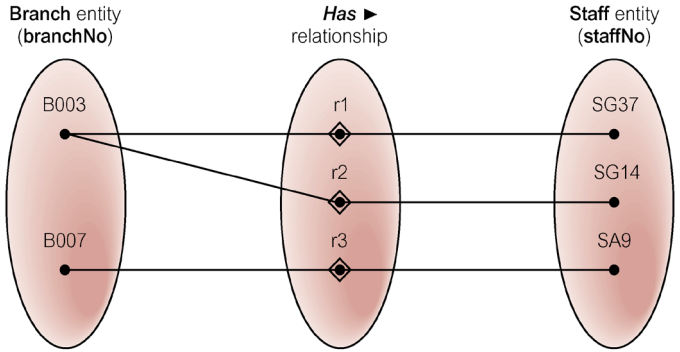
HALMSTAD UNIVERSITY

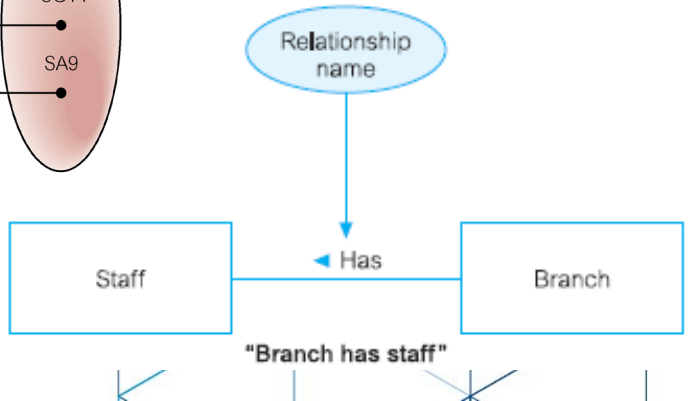
Relationships

- Relationships are connections or associations among two or more entity sets (Wagner teaches DI4020).
- Have a relationship type
  - Set of meaningful associations among entity types.
- Drawn as a diamond between the related entities, connected to the entities by lines.
- A relationship is often named with a verb form.





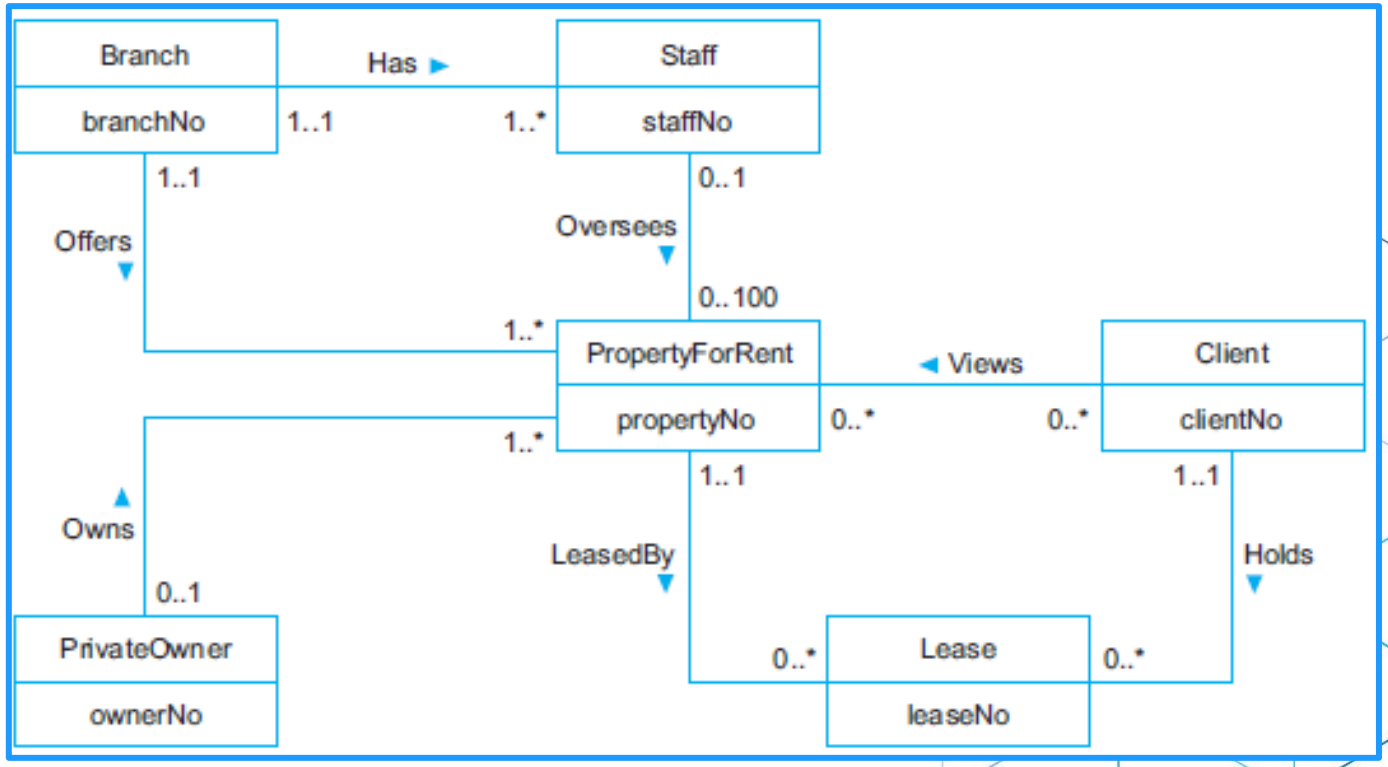




HALMSTAD UNIVERSITY

Relationships in an ER diagram
How many?

- Six entities (the rectangles): Branch, Staff, PropertyForRent, Client, PrivateOwner, and Lease;
- Six attributes, one for each entity: branchNo, staffNo, propertyNo, clientNo, ownerNo, and leaseNo.
- Seven relationships (the names adjacent to the lines): Has, Offers, Oversees, Views, Owns, LeasedBy, and Holds;

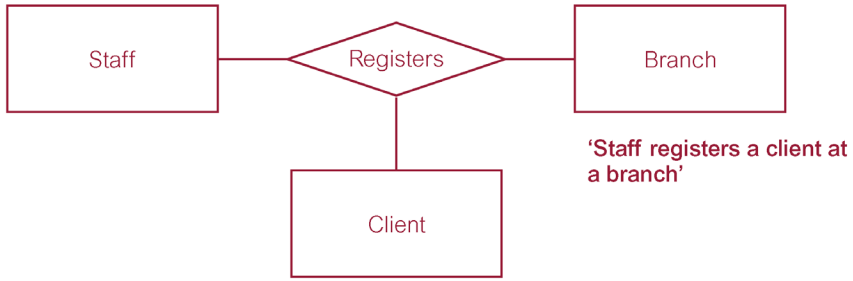


HALMSTAD UNIVERSITY

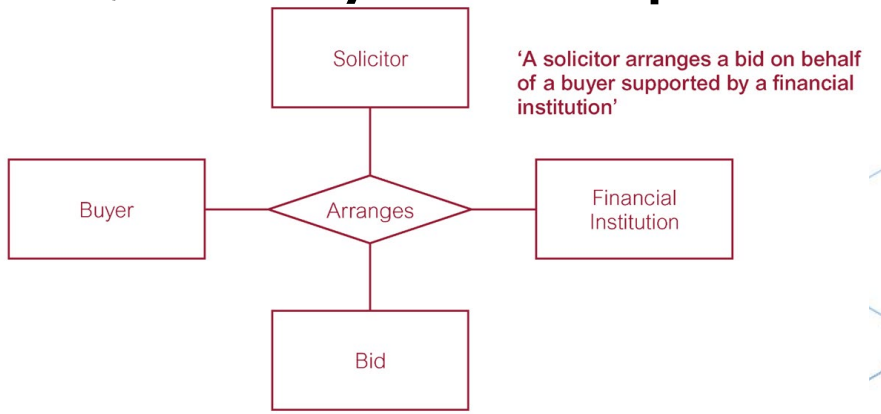
Relationship Types

- Degree of a Relationship
  - Number of participating entities in relationship.
- Relationship of degree
  - **Binary relationship**

  - **Ternary relationship**



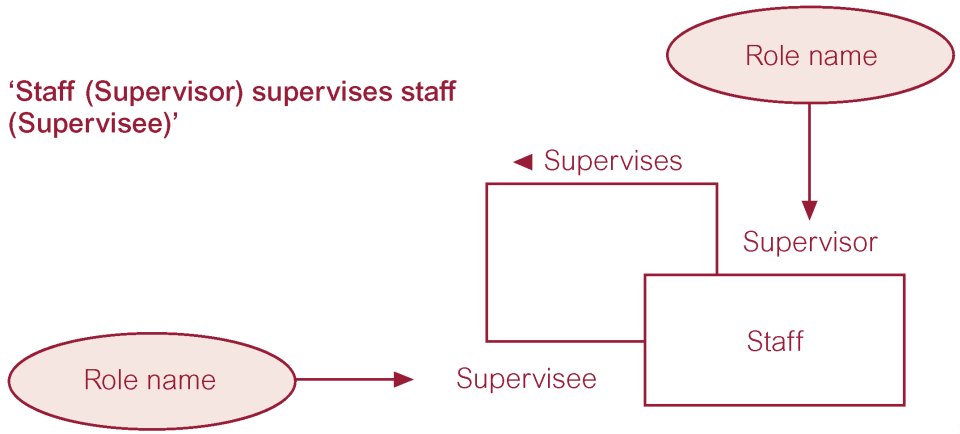
  - **Quaternary relationship**

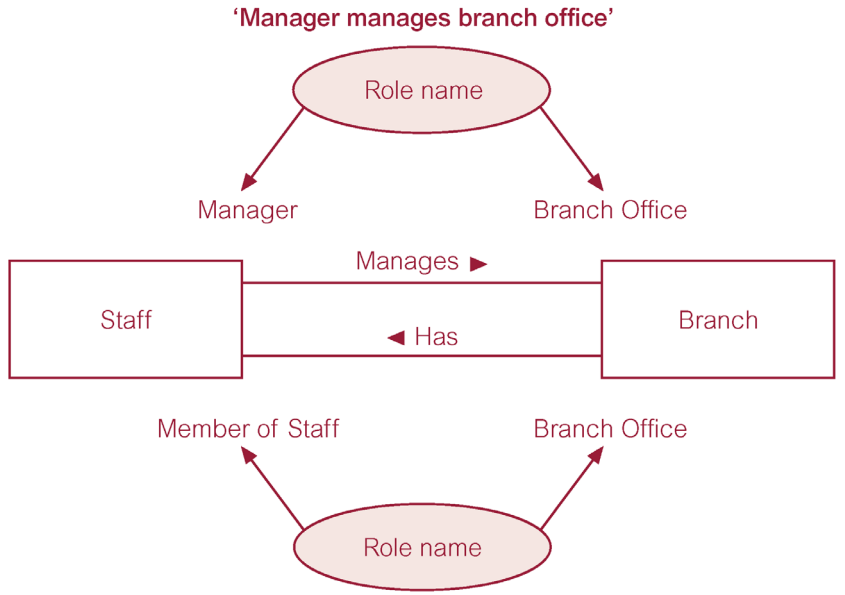


HALMSTAD UNIVERSITY

Relationship Types

- Recursive Relationship
  - Relationship type where same entity type participates more than once in different roles.
- Role names
  - Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship.

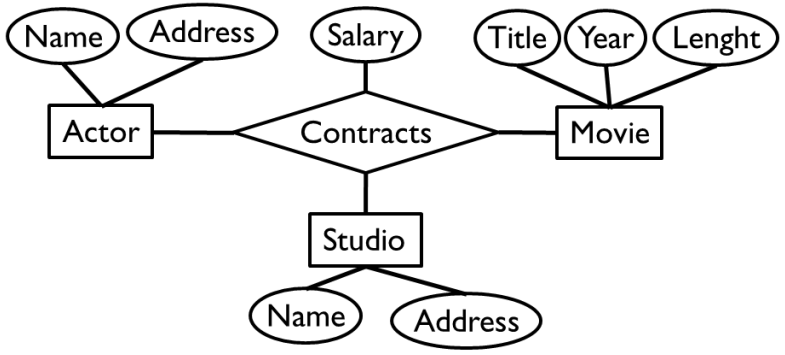


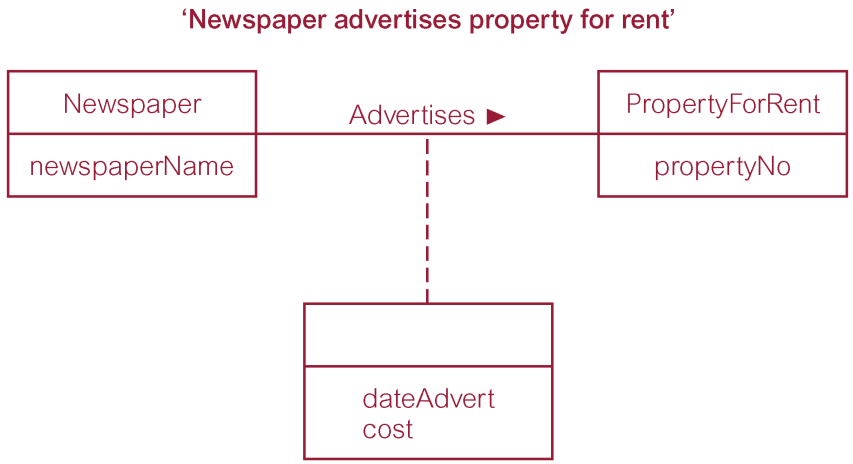


HALMSTAD UNIVERSITY

Relationship Attributes

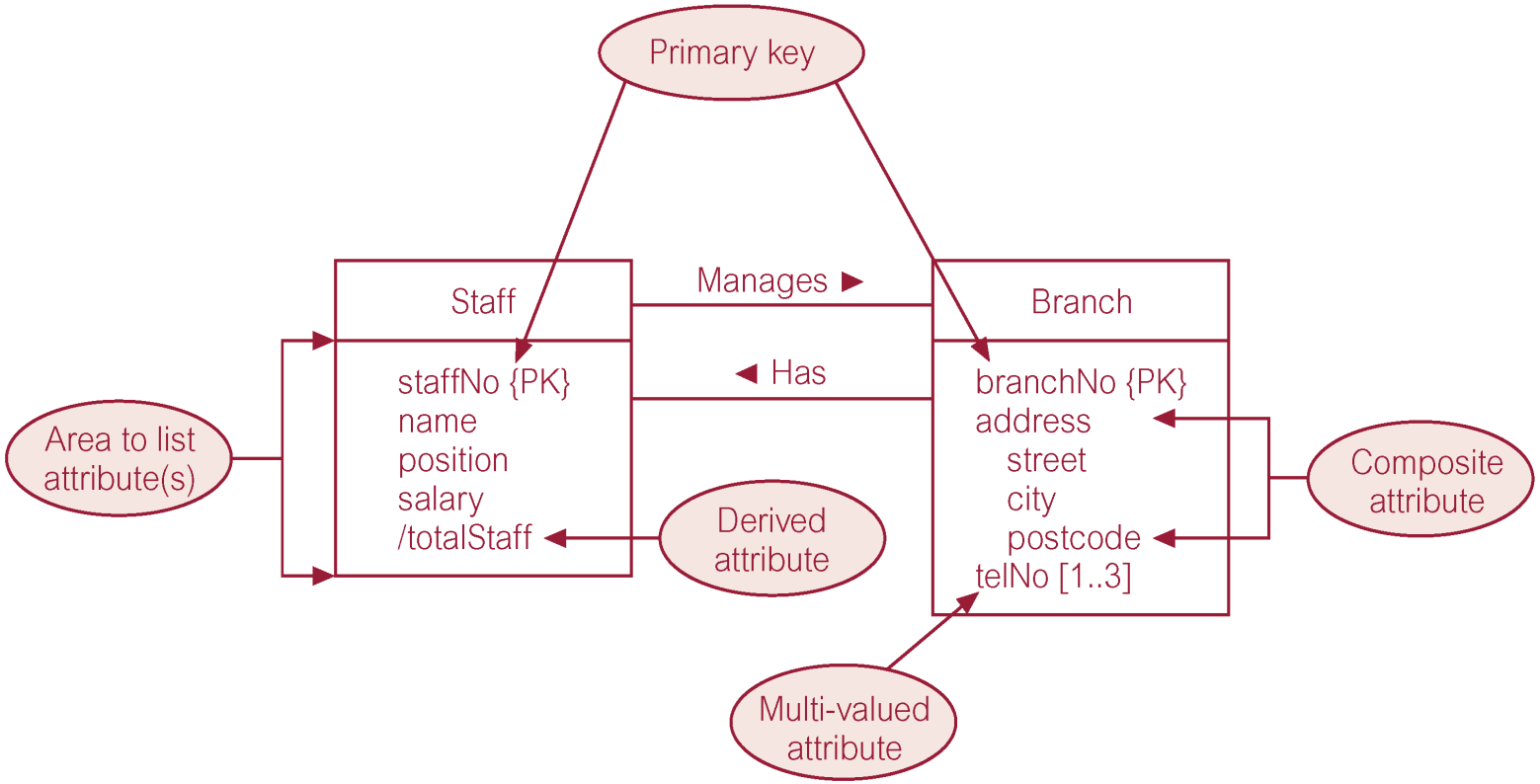
- Relationships can also have attributes.
- Represent a property of the relationship between the entities.
- Relationship attributes make only sense in the context of a relationship.





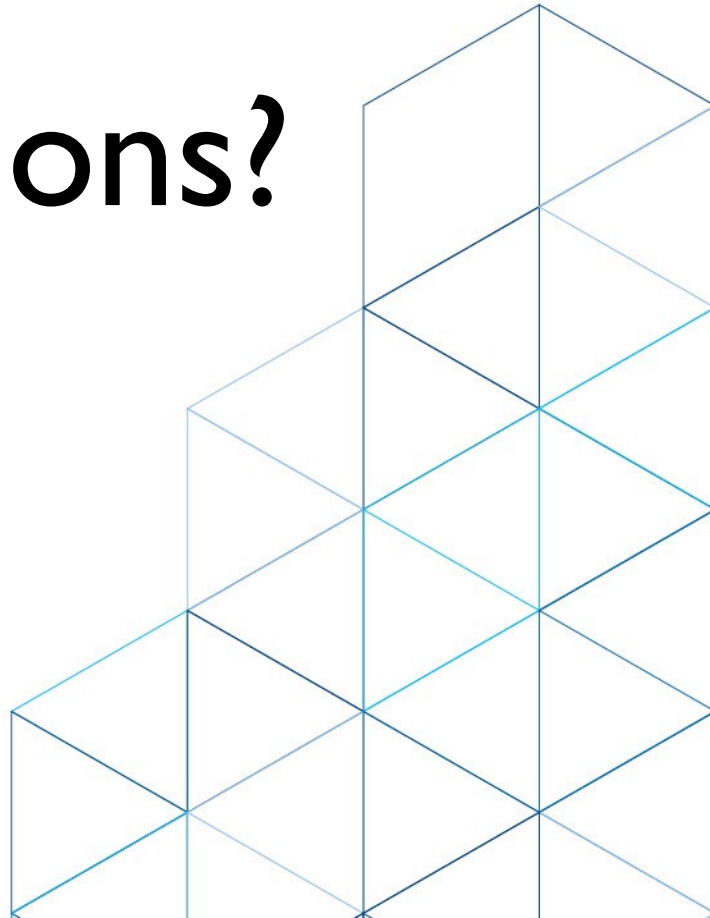
HALMSTAD UNIVERSITY

ER diagram of Staff and Branch entities and their attributes



HALMSTAD UNIVERSITY

Comments or Questions?



HALMSTAD UNIVERSITY

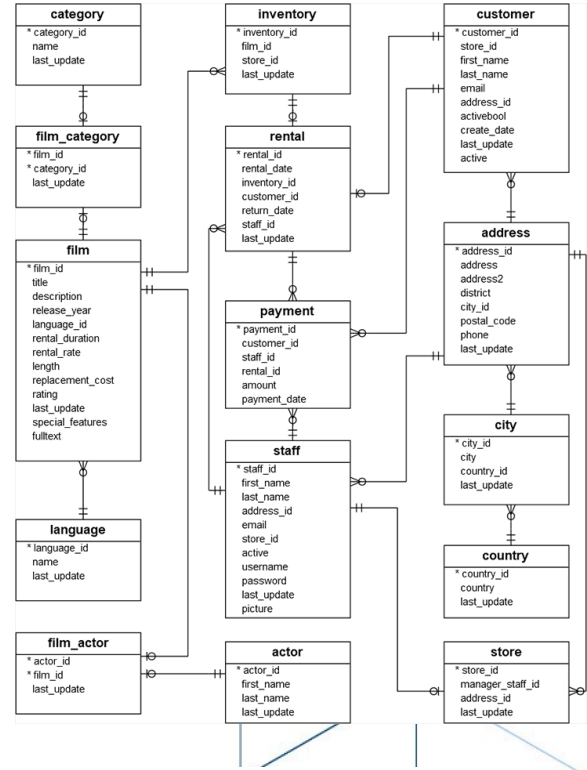
Structural Constraints

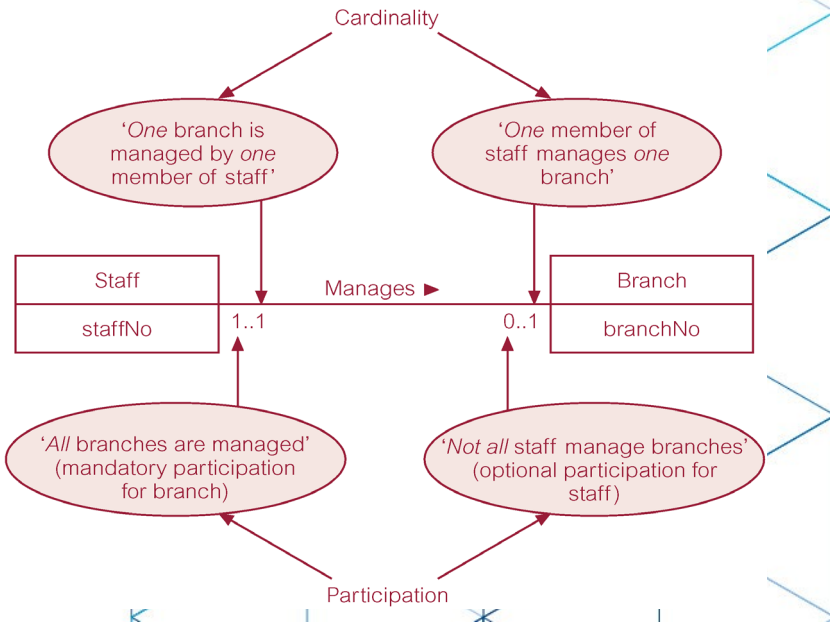
- Constraints reflect the restrictions on the relationships as perceived in the "real world."
  - A property for rent must have an owner.
  - Each branch must have staff.
- Represents policies (called business rules) established by user or company.
- Main type of constraint on relationships is called **multiplicity**.
  - Multiplicity constraints on relationships
    - Represents the number of occurrences of one entity that may relate to a single occurrence of an associated entity.
      - The lower and an upper **cardinality**
    - Multiplicity is made up of two types of restrictions on relationships: cardinality and participation.

Structural Constraints

- Multiplicity is made up of two types of restrictions on relationships:
  - Cardinality
    - Cardinality means "count," and is expressed as a number.
    - Maximum cardinality is the maximum number of entity instances that can participate in a relationship.
    - Minimum cardinality is the minimum number of entity instances that must participate in a relationship.
  - Participation
    - Determines whether all or only some entity occurrences participate in a relationship (Total vs. Partial Participation).

https://www.postgresqltutorial.com/postgresql-sample-database/



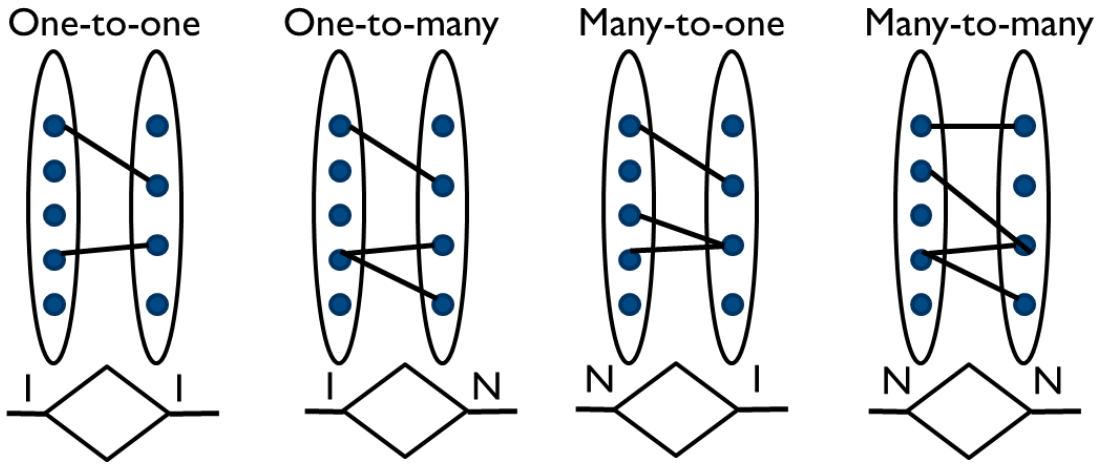


HALMSTAD UNIVERSITY

Structural Constraints

- The most common degree for relationships is binary.
- Binary relationships are generally referred to as being:
  - one-to-one (1:1)
  - one-to-many (1:*)
  - many-to-many (*:*) n-to-n, many-many

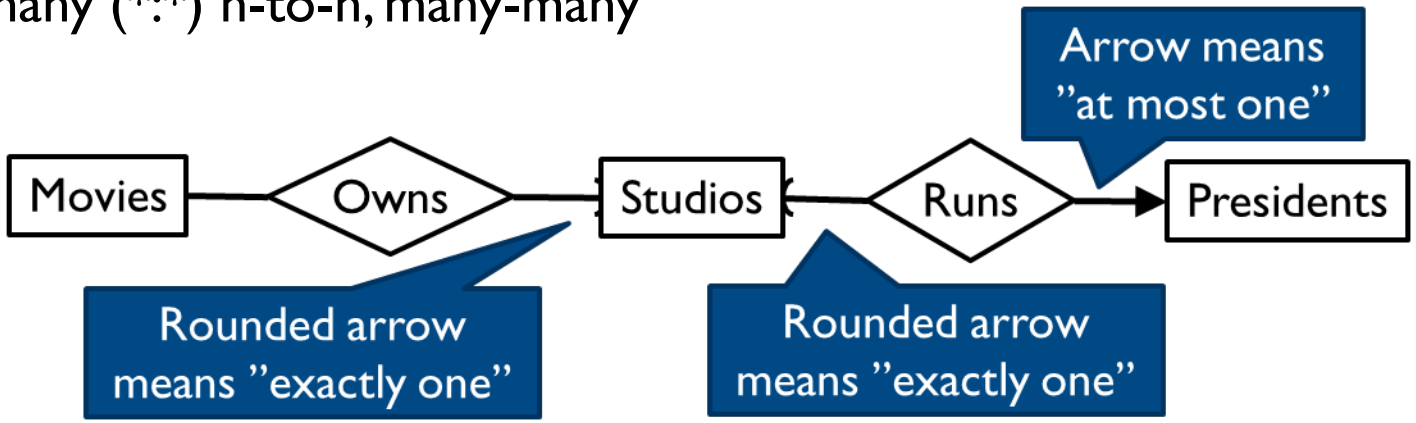
Cardinality of a given entity in relation to another.



Structural Constraints

- The most common degree for relationships is binary.
- Binary relationships are generally referred to as being:
  - one-to-one (1:1)
  - one-to-many (1:*)
  - many-to-many (*:*) n-to-n, many-many

Cardinality of a given entity in relation to another.

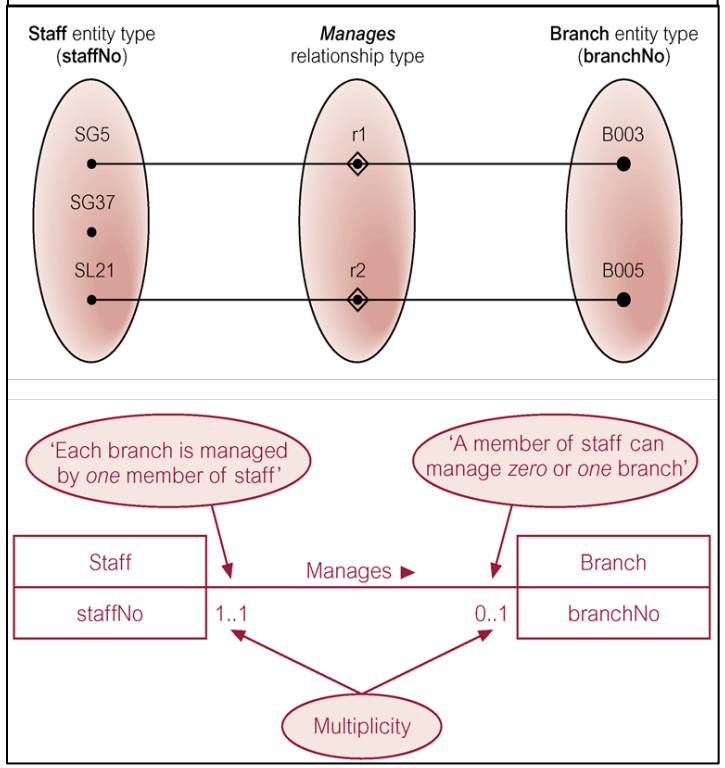


Referential integrity constraint that every movie must be owned by one studio, and this studio is present in the *Studios* entity set.

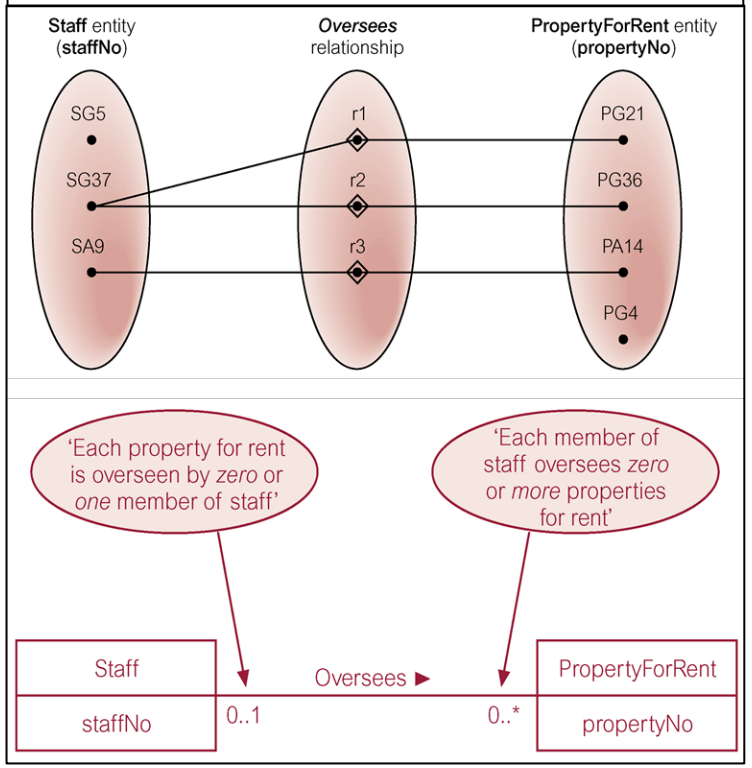
Referential integrity constraint that every president runs a studio that exists in the Studios entity set.

Binary relationships

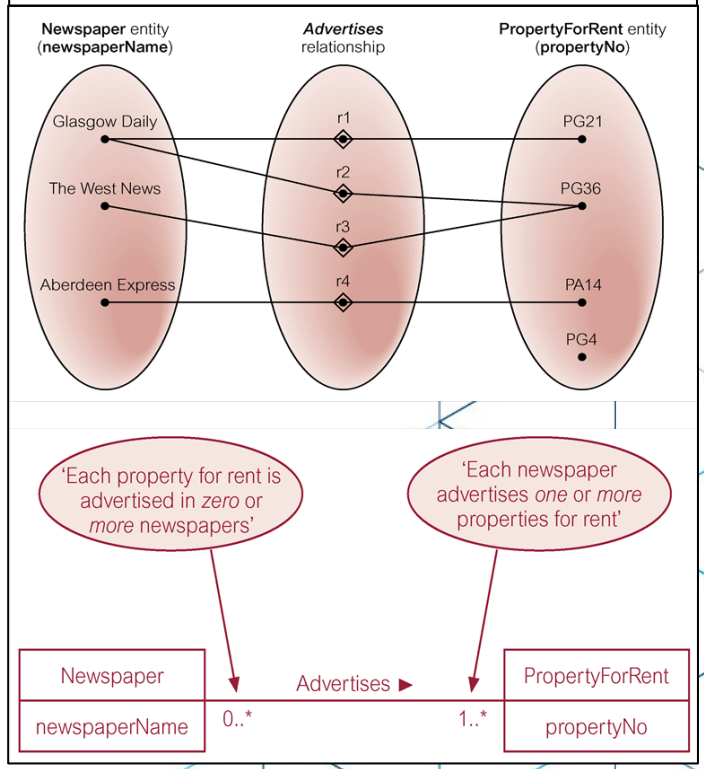
One-to-one (1:1) relationship



One-to-many (1:*) relationship



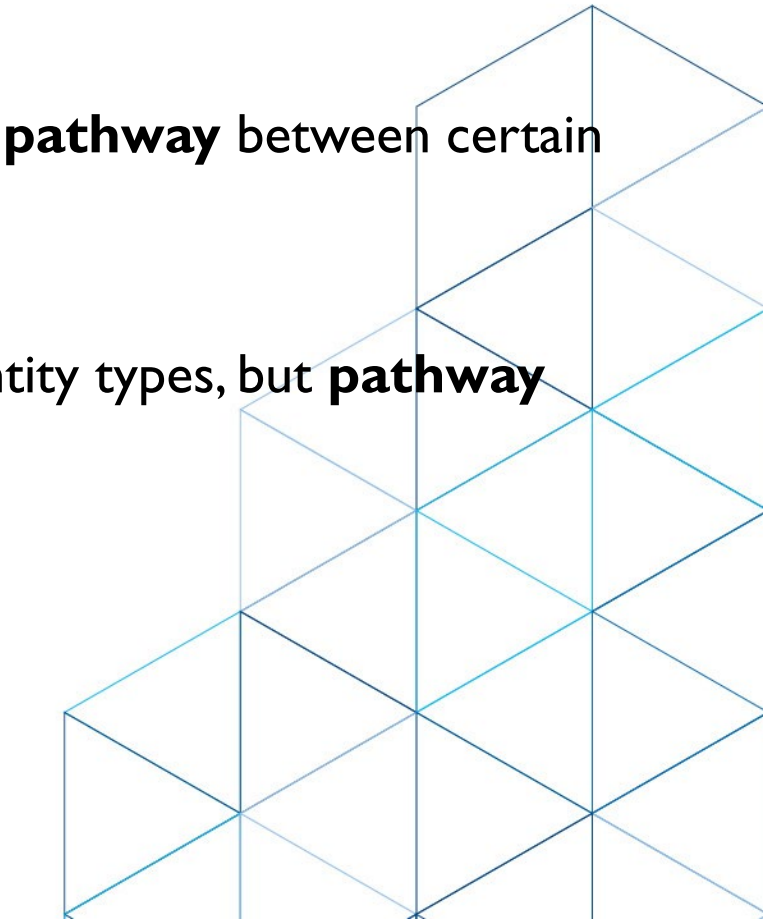
Many-to-many (*:*) relationship



HALMSTAD UNIVERSITY

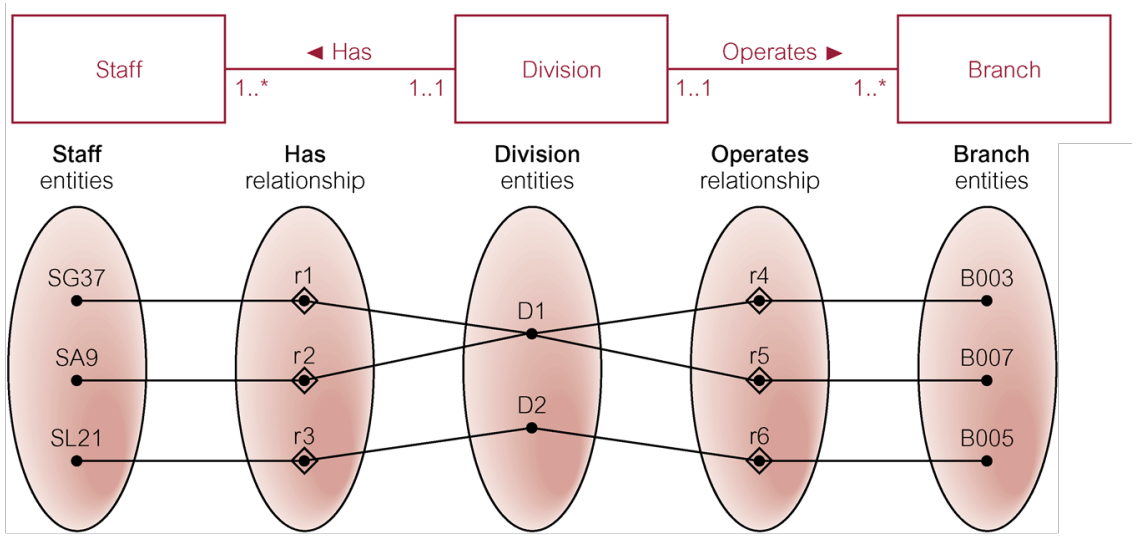
Problems with ER Models

- Problems may arise when designing a conceptual data model called connection **traps**.
- Often due to a misinterpretation of the meaning of certain relationships.
- **Fan Trap**
  - Where a model represents a relationship between entity types, but **pathway** between certain entity occurrences is **ambiguous**.
- **Chasm Trap**
  - Where a model suggests the existence of a relationship between entity types, but **pathway does not exist** between certain entity occurrences.

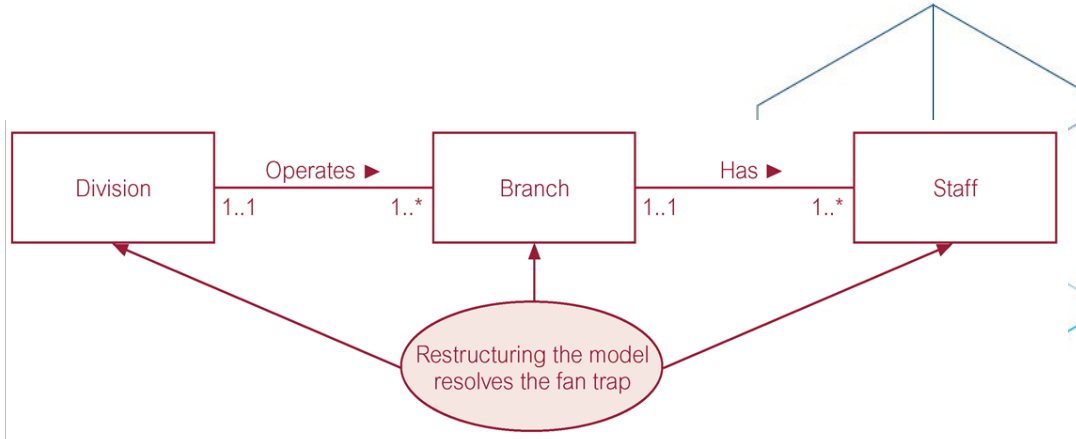


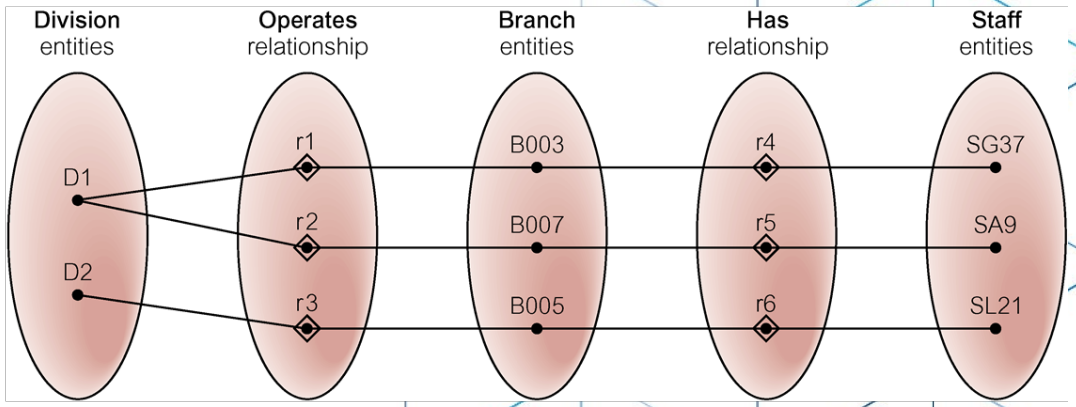
An Example of a Fan Trap and its solution

- A model represents a relationship between entity types, but **pathway** between certain entity occurrences is **ambiguous**.



At which branch office does staff number SG37 work?



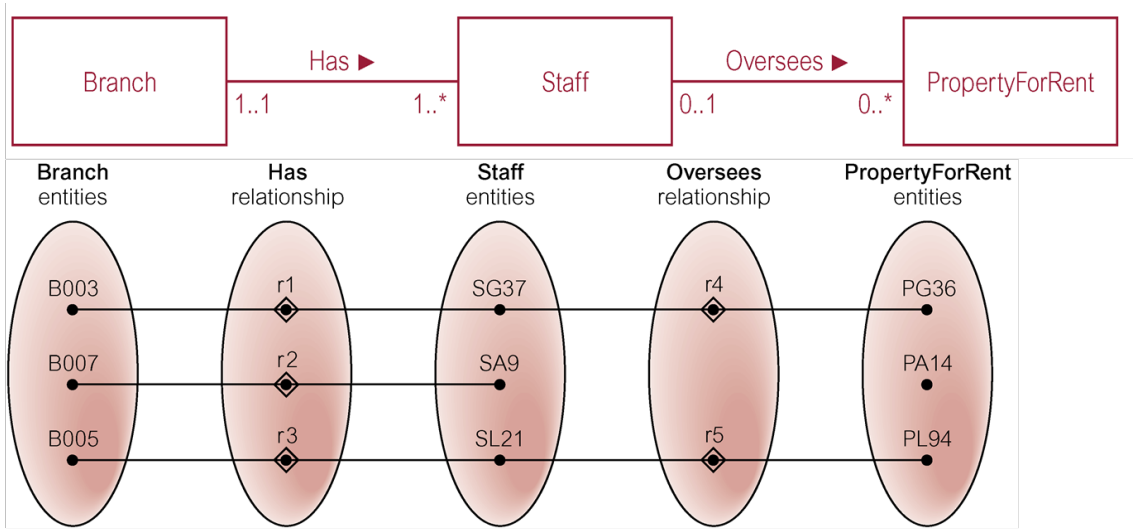


SG37 works at branch B003.

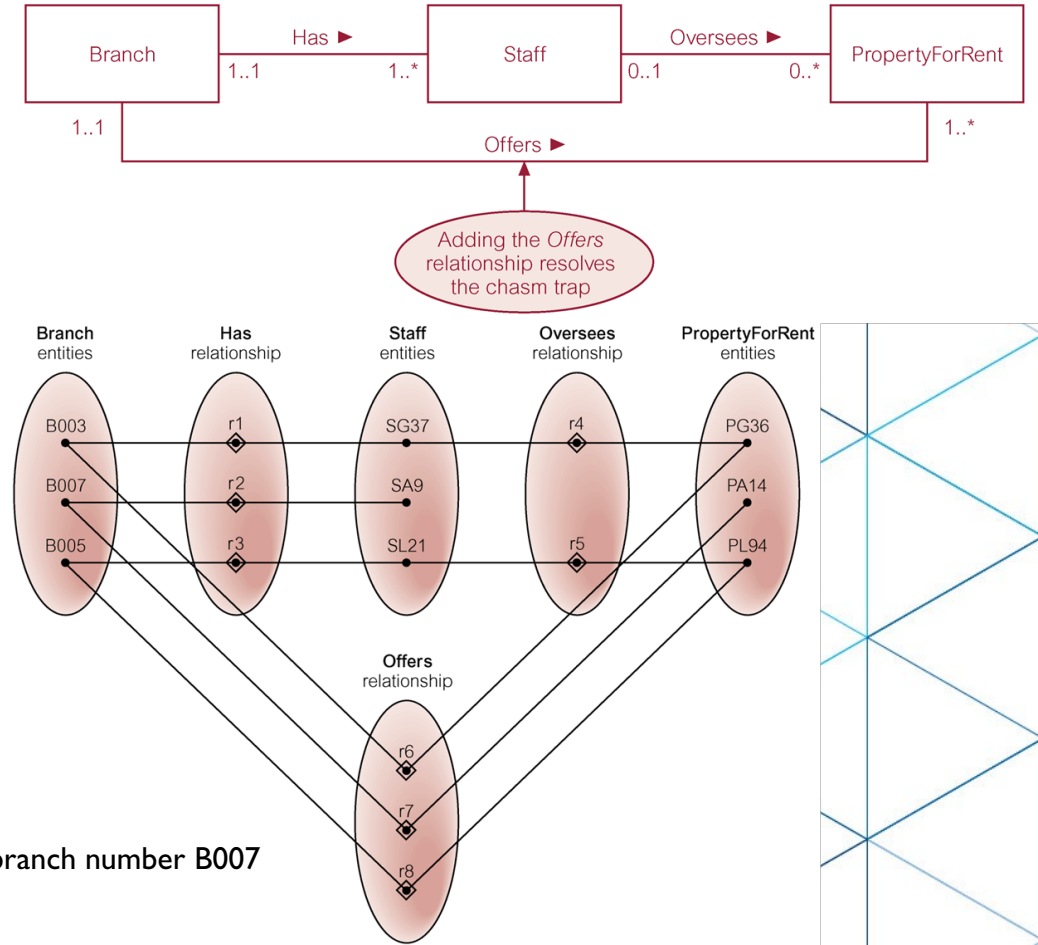
HALMSTAD UNIVERSITY

An Example of a Chasm Trap and its solution

- A model suggests the existence of a relationship between entity types, but **pathway does not exist** between certain entity occurrences.



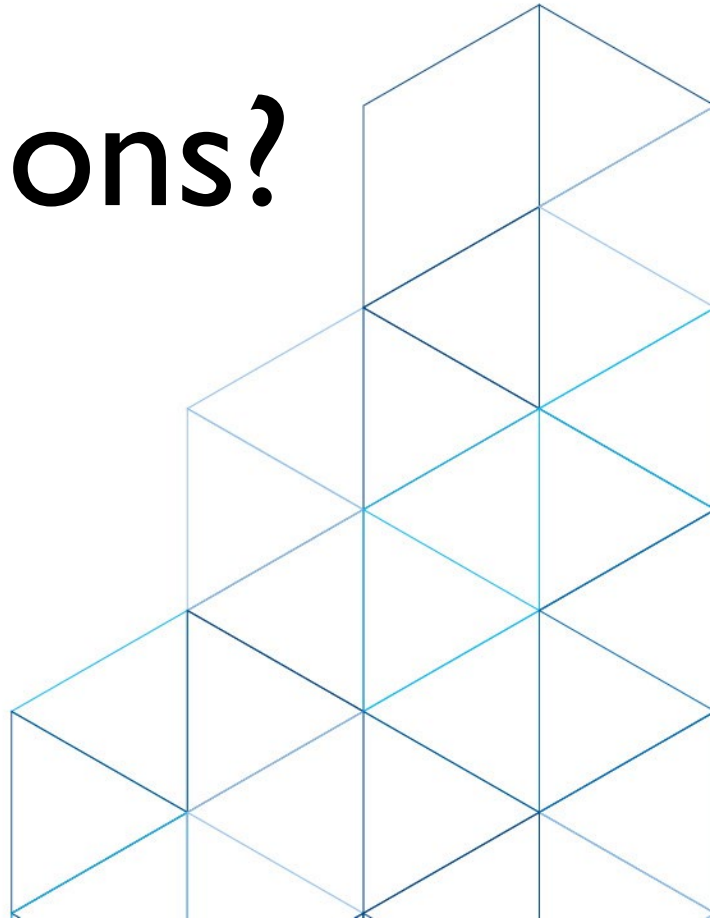
At which branch office is property PA14 available?



property number PA14 is available at branch number B007

HALMSTAD UNIVERSITY

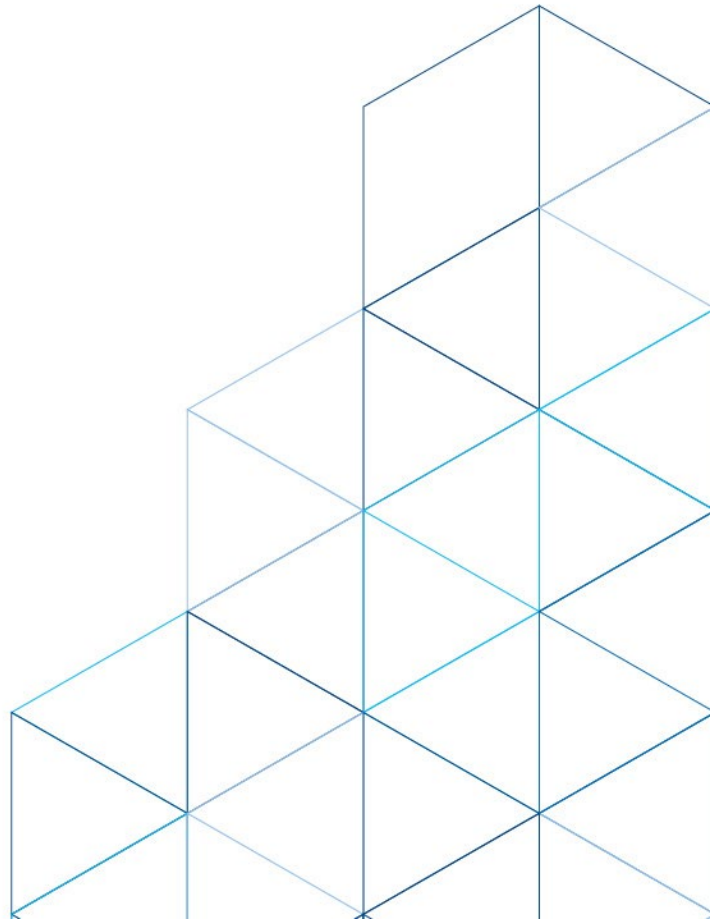
Comments or Questions?



HALMSTAD UNIVERSITY

Next

- Design of databases, e.g.
  - Lecture 5
    - Relational data model
      - Dependencies and normalisation
- Database programming
  - Lectures 6 to 8
    - Data definition, manipulation and querying in SQL



HALMSTAD UNIVERSITY