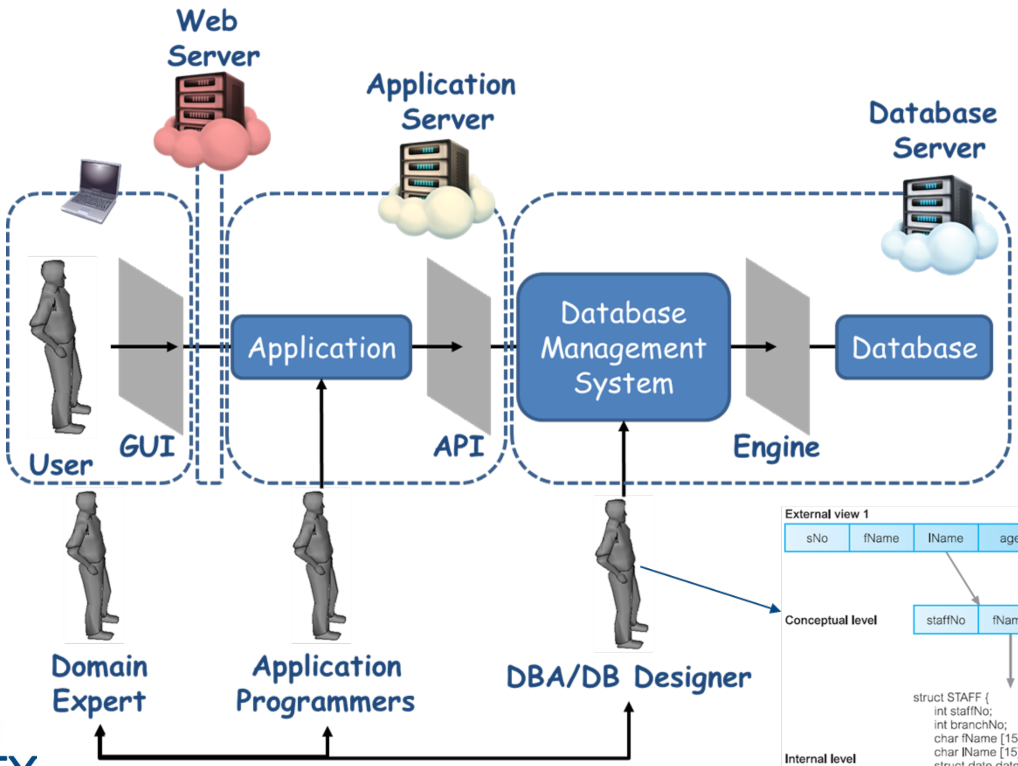
Web Systems Fundamentals and Databases (Grundläggande webbsystem och databaser) DI4020 - 11hp

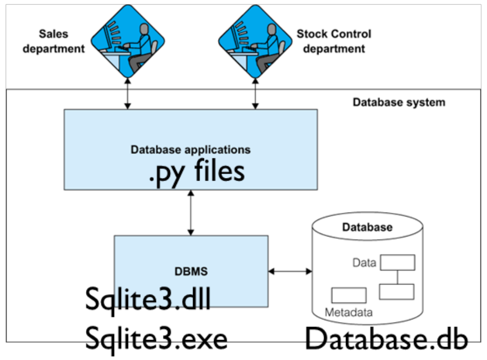
Lecture 5

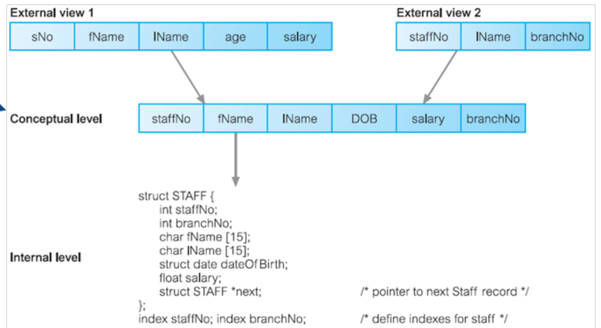
Wagner Ourique de Morais

Previously in the course

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



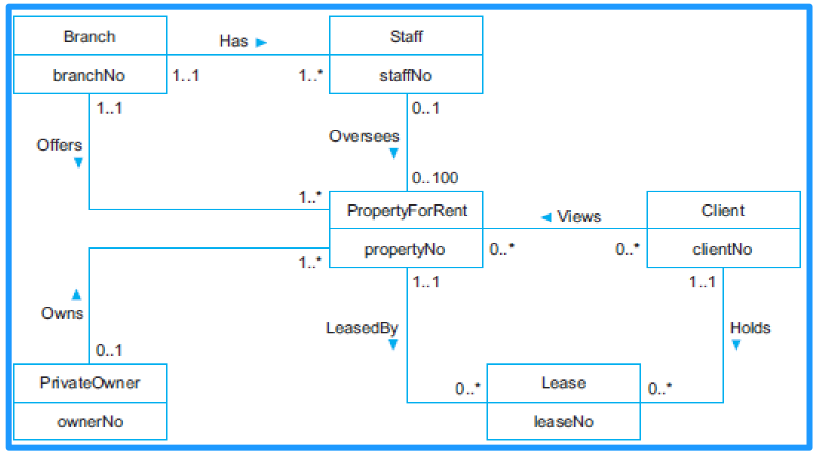


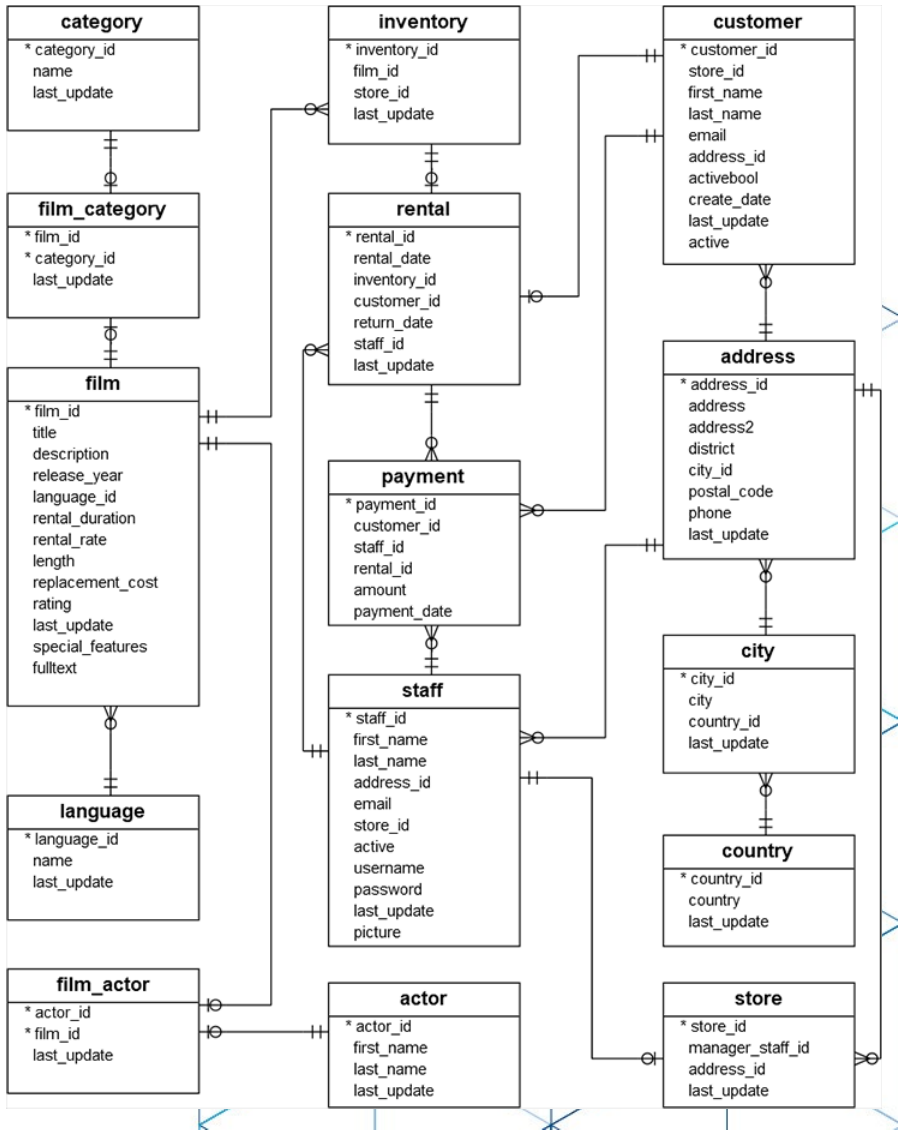


```
struct STAFF {
      int staffNo;
      int branchNo;
      char fName [15];
      char lName [15];
      struct date dateOfBirth;
      float salary;
      struct STAFF *next;              /* pointer to next Staff record */
};
index staffNo; index branchNo;         /* define indexes for staff */
```

Previously in the course

- Lectures 3 and 4
  - Types of database design
  - Database design process
    - Conceptual, logical and physical design.
  - Conceptual database design
    - Build conceptual data model independent of all physical considerations.
  - Entity-relationship (ER) data model and ER diagrams





Comments or Questions?

HALMSTAD UNIVERSITY

Objectives

- Continue on database design
  - Chapter 4 introduces the concepts behind the most popular data model, the relational model,.

  - Chapter 14 introduces the concepts behind normalization.

  - Chapter 17 presents a step-by-step methodology for logical database design for the
  - relational model.
    - *Useful for Laboratory exercise 2*

…more specifically

- Introduce the relational model and relational database design.
- Understand the terminology of the relational model.
- Learn how to identify primary, candidate, and composite keys.
- Understand the normalization and the normalization process.
- Learn how to derive a set of relations from a conceptual data model.

HALMSTAD UNIVERSITY

Related to Lab Exercise 2

- Logical database design for the relational model
  - – Develop a data model for the information managed by a state agency using a specific data model (e.g., relational), while remaining independent of any particular database management system (DBMS) and other physical implementation details.
    - Derive relations for logical data model
    - Validate relations using **normalization**
    - Validate relations against user transactions
    - Check integrity constraints
    - Review logical data model against the scenario

Review

- A **data model** is an abstraction of a real-world object or event.
  - Collection of concepts for describing data, relationships between data, and constraints on the data.
  - Has three components: a structural part, a manipulative part, and a set of integrity rules.
  - In this course, we discuss the **ER** and the **relational data model**.
- **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).
- **Database design** is the process of designing a complete database application, and it involves three phases: conceptual, logical and physical design.
  - The overall design of a database is called the **database schema**.
  - A particular representation of an external view is called an **external schema**.
  - A specific representation of a conceptual view is called a **conceptual schema**.
  - During the conceptual design, we commonly use conceptual models such as ER models.
    - Next, we use the relational model in the logical design of databases.

HALMSTAD UNIVERSITY

Review
Database Development Process

Three phases of database design

- 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 relations (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.

HALMSTAD UNIVERSITY

Review
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

Review

- Conceptual design
  - Main entities and relationships
  - Multiplicity constraints for the relationships

Conolly, Begg & Holowczak, "Business Database Systems", 2015

Entity	Multiplicity	Relationship	Multiplicity	Entity
DistributionCenter	1..1	Stocks	1..*	DVDCopy
DistributionCenter	1..1	Supplies	1..*	Member
DistributionCenter	1..1	Has	1..*	Staff
DVD	1..*	Stars	0..*	Actor
DVD	1..1	Is	1..*	DVDCopy
DVD	1..1	AppearsIn	0..*	Wish
DVDCopy	1..5	IsSentAs	0..*	RentalDelivery
Member	1..*	Selects	1..1	MembershipType
Member	1..1	Makes	0..25	Wish
Member	1..1	Receives	1..*	RentalDelivery
Staff	1..1	Manages	0..1	DistributionCenter

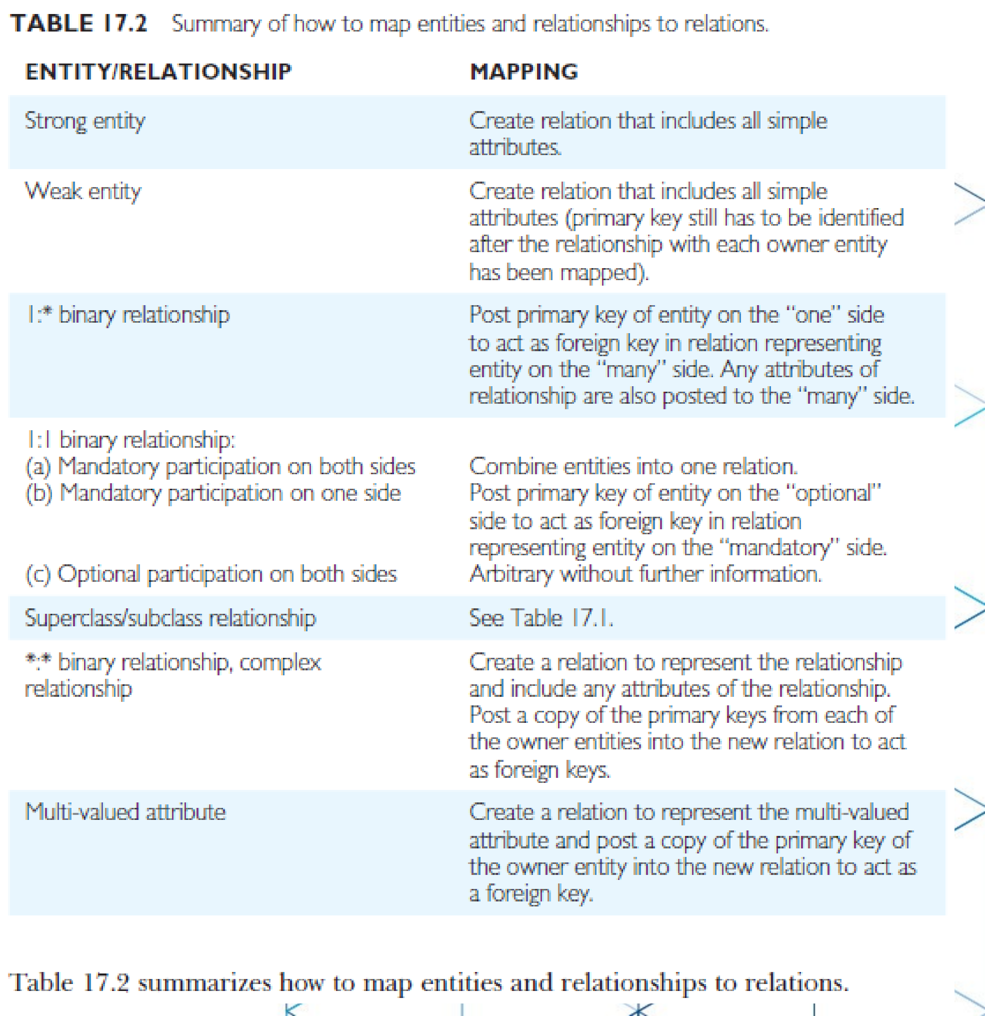


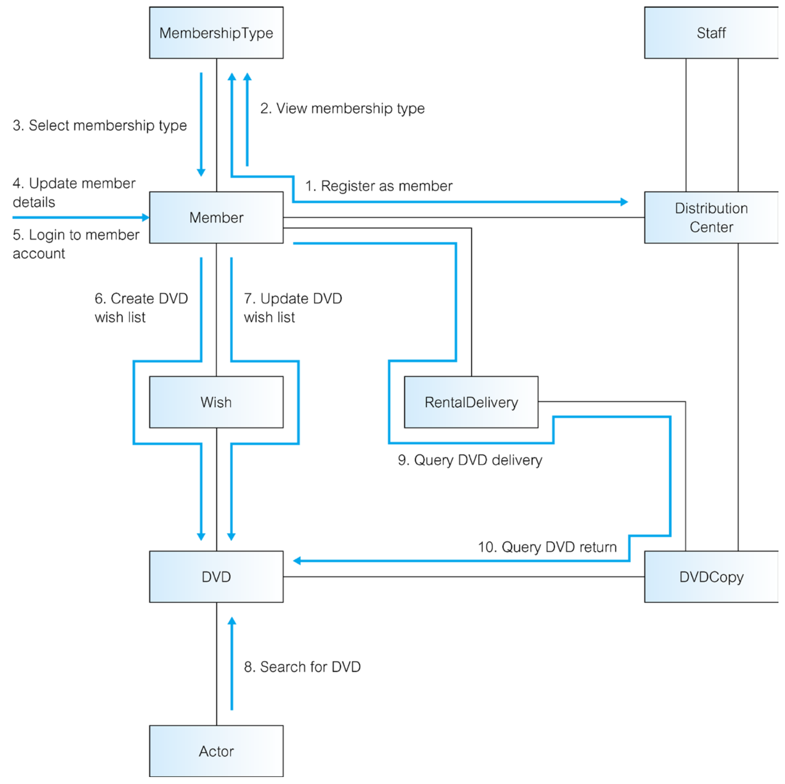
TABLE 17.2 Summary of how to map entities and relationships to relations.

ENTITY/RELATIONSHIP	MAPPING
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1:* binary relationship	Post primary key of entity on the "one" side to act as foreign key in relation representing entity on the "many" side. Any attributes of relationship are also posted to the "many" side.
1:1 binary relationship:	
(a) Mandatory participation on both sides	Combine entities into one relation.
(b) Mandatory participation on one side	Post primary key of entity on the "optional" side to act as foreign key in relation representing entity on the "mandatory" side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 17.1.
: binary relationship, complex relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.

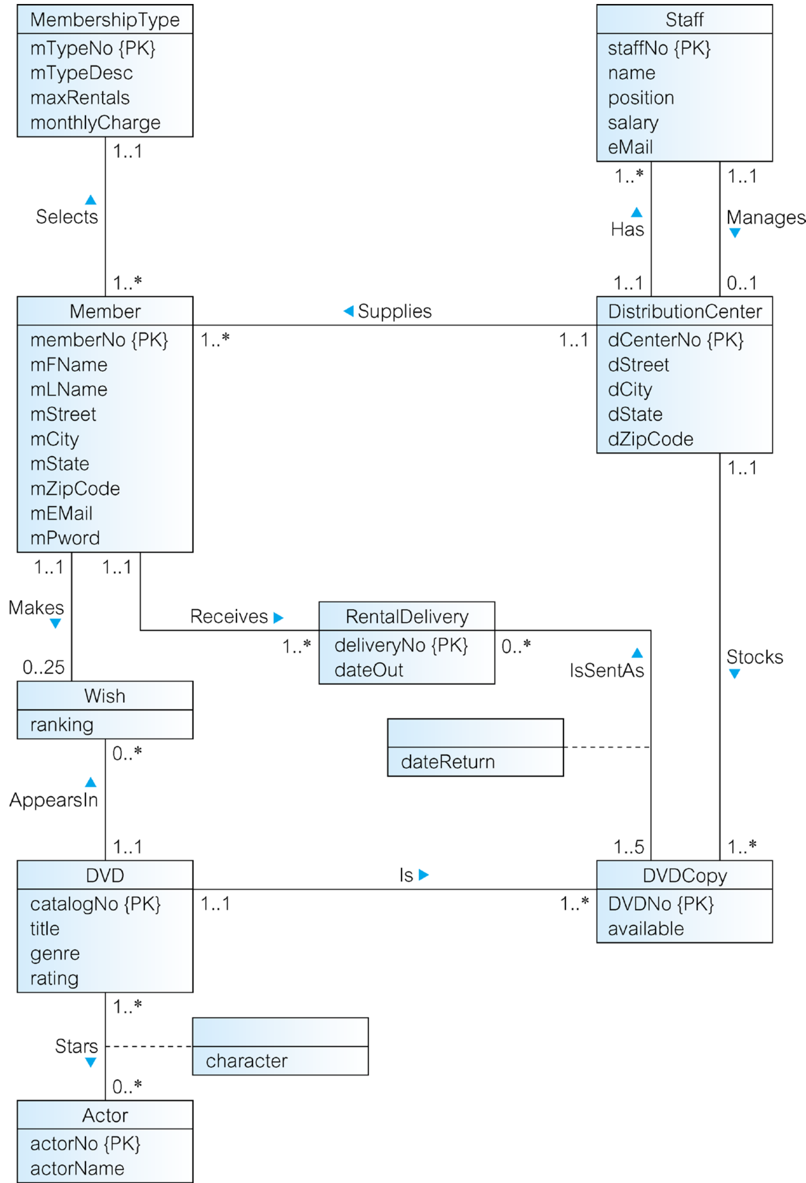
Table 17.2 summarizes how to map entities and relationships to relations.

Review

- Conceptual design
  - Final ER diagram showing all entities, relationships, and attributes.
  - Checking user transactions with pathways.



Conolly, Begg & Holowczak, "Business Database Systems", 2015



Comments or Questions?

HALMSTAD UNIVERSITY

The Relational Model

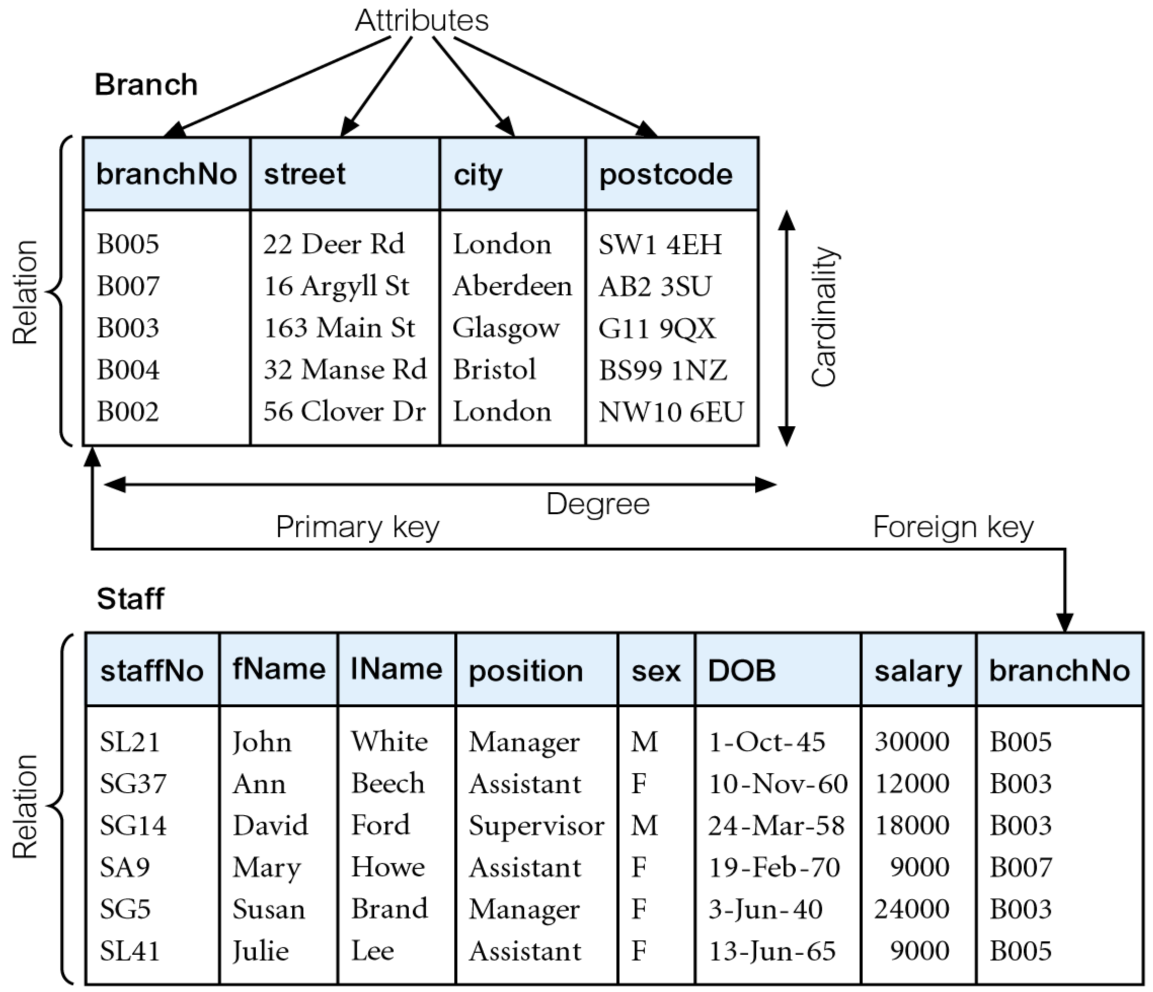
- Introduced in 1970 by Edgar F. Codd in “A relational model of data for large shared data banks”
  - To allow high degree of data independence,
  - Introduced **Normalization** as an approach to deal with data semantics, consistency, and redundancy problems.
  - Access data through high-level declarative language
- Standard model for many commercial DBMS products.

Relational Model Terminology

- In the relational model, relations are used to hold information about the objects to be represented in the database.
- A **relation** is represented as a two-dimensional **table** in which the **rows** of the table correspond to individual **records** and the table **columns** correspond to **attributes**.

- **Relation**: table with columns and rows.
- **Attribute**: named column of a relation.
- **Domain**: set of allowable values for one or more attributes.
- **Tuple**: a row (record) of a relation.
- **Degree**: number of attributes in a relation.
- **Cardinality**: number of tuples in a relation.
- **Relational Database**: collection of normalized (well structured) relations with distinct relation names.

Examples



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

Alternative Relational Model Terminology

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

Examples of Attribute Domains

Attribute	Domain Name	Meaning	Domain Definition
branchNo	BranchNumbers	The set of all possible branch numbers	character: size 4, range B001–B999
street	StreetNames	The set of all street names in Britain	character: size 25
city	CityNames	The set of all city names in Britain	character: size 15
postcode	Postcodes	The set of all postcodes in Britain	character: size 8
sex	Sex	The sex of a person	character: size 1, value M or F
DOB	DatesOfBirth	Possible values of staff birth dates	date, range from 1-Jan-20, format dd-mmm-yy
salary	Salaries	Possible values of staff salaries	monetary: 7 digits, range 6000.00–40000.00

Properties of Relations

- Relation (table) name is distinct from all other relation names in relational schema.
- Each cell of relation contains exactly one atomic (single) value.
- Each attribute (column) has a distinct name.
- Values of an attribute (column) are all from the same domain.
- Each tuple (row) is distinct, i.e., there are no duplicate tuples.
  - – No two tuples (rows) may be identical.
- Order of attributes (columns) has no significance.
- Order of tuples (rows) has no significance, theoretically.

Formal terms	Alternative 1	Alternative 2
Relation	Table	File
Tuple	Row	Record
Attribute	Column	Field

HALMSTAD UNIVERSITY

Relational Keys

- **Superkey**
  - An attribute, or set of attributes (ie columns), that uniquely identifies a tuple (ie record) within a relation (ie table).
- **Candidate Key**
  - Superkey (K) such that no proper subset is a superkey within the relation.
  - In each tuple of R, values of K uniquely identify that tuple (uniqueness).
  - No proper subset of K has the uniqueness property (irreducibility).
- **Composite Key**
  - A key consists of more than one column
- **Primary Key**
  - Candidate key selected to identify tuples uniquely within relation.
- **Alternate Keys**
  - Candidate keys that are not selected to be primary key.
- **Foreign Key**
  - Attribute, or set of attributes, within one relation that matches candidate key of some (possibly same) relation.

Integrity Constraints

- **Null**
  - Represents value for an attribute that is currently unknown or not applicable for tuple.
  - Deals with incomplete or exceptional data.
  - Represents the absence of a value and is not the same as zero or spaces, which are values.
- **Entity Integrity**
  - In a base relation, no attribute of a primary key can be null.
- **Referential Integrity**
  - If foreign key exists in a relation, either foreign key value must match a candidate key value of some tuple in its home relation or foreign key value must be wholly null.
- **General Constraints**
  - Additional rules specified by users or database administrators that define or constrain some aspect of the enterprise.

Relational Languages

- **Relational Algebra**
  - A (high-level) procedural language: it can be used to tell the DBMS how to build a new table from one or more tables in the database.
    - Find all students majoring in “CS “: $\sigma_{Major='CS'}(Students)$
- **Relational Calculus**
  - A non-procedural language; it can be used to formulate the definition of a table in terms of one or more tables.
    - Find all students majoring in “CS “: $\{t \mid t \in Students \wedge t.Major =' CS'\}$
- **Two main languages**
  - SQL (Structured Query Language)
    - `SELECT * FROM Students WHERE Major = 'CS';`
  - QBE (Query By Example), which is visual.

Representing Relational Database Schemas

- The common convention for representing a relation schema is to give the name of the relation followed by the attribute names in parentheses. Normally, the primary key is underlined.
  - RelationName (<u>primaryKey</u>, attribute1, attribute2, …, attributen)
- Ex
  - `Branch (branchNo, street, city, postcode)`
  - `Staff (staffNo, fName, IName, position, sex, DOB, salary, branchNo)`
  - `PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)`
  - `PrivateOwner (ownerNo, fName, IName, address, telNo, eMail, password)`

Example Relational Database Schema

Branch (branchNo, street, city, postcode)

Staff (staffNo, fName, lName, position, sex, DOB, salary, branchNo)

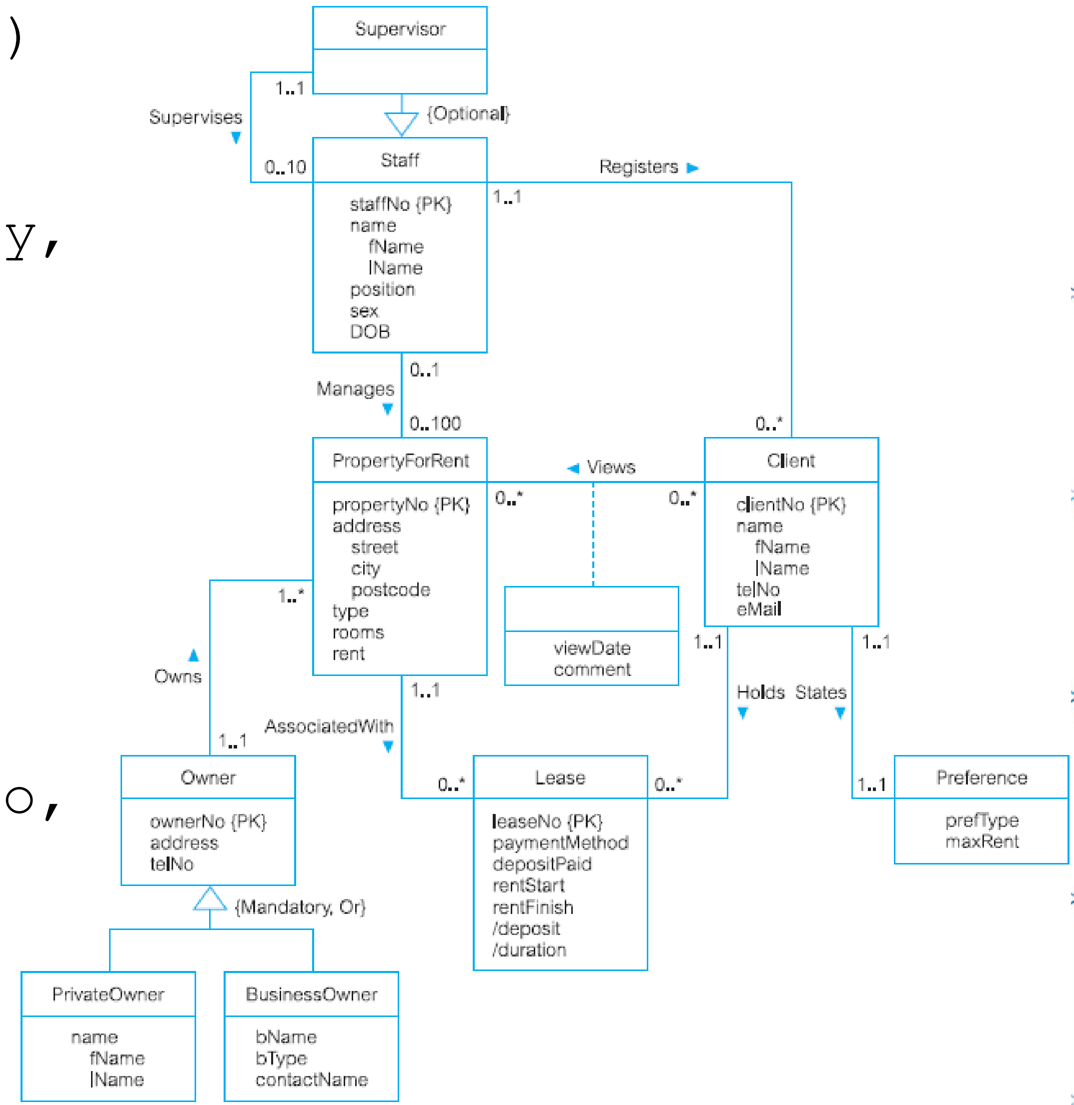
PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)

Client (clientNo, fName, lName, telNo, prefType, maxRent, eMail)

PrivateOwner (ownerNo, fName, lName, address, telNo, eMail, password)

Viewing (clientNo, propertyNo, viewDate, comment)

Registration (clientNo, branchNo, staffNo, dateJoined)



HALMSTAD UNIVERSITY

Views

- **Base Relation**
  - Named relation corresponding to an entity in conceptual schema, whose tuples are physically stored in database.

- **View**
  - Dynamic result of one or more relational operations operating on base relations to produce another relation.
  - A virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.
  - Contents of a view are defined as a query on one or more base relations.
  - Views are dynamic, meaning that changes made to base relations that affect view attributes are immediately reflected in the view.

HALMSTAD UNIVERSITY

Purpose of Views

- Provides powerful and flexible security mechanism by hiding parts of database from certain users.
- Permits users to access data in a customized way, so that same data can be seen by different users in different ways, at same time.
- Can simplify complex operations on base relations.

Updating Views

- All updates to a base relation should be immediately reflected in all views that reference that base relation.
- If view is updated, underlying base relation should reflect change.
- There are restrictions on types of modifications that can be made through views:
  - Updates are allowed if query involves a single base relation and contains a candidate key of base relation.
  - Updates are not allowed involving multiple base relations.
  - Updates are not allowed involving aggregation or grouping operations.
- Classes of views are defined as:
  - theoretically not updateable;
  - theoretically updateable;
  - partially updateable.

HALMSTAD UNIVERSITY

Comments or Questions?

What can you tell about the following relations?

- Can you see potential problems associated with data redundancy?

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.

StaffDistributionCenter

staffNo	name	position	salary	dCenterNo	dAddress	dTelNo
S1500	Tom Daniels	Manager	48000	D001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0003	Sally Adams	Assistant	30000	D001	8 Jefferson Way, Portland, OR 97201	503-555-3618
S0010	Mary Martinez	Manager	51000	D002	City Center Plaza, Seattle, WA 98122	206-555-6756
S3250	Robert Chin	Assistant	33000	D002	City Center Plaza, Seattle, WA 98122	206-555-6756
S2250	Sally Stern	Manager	48000	D004	2 W. El Camino, San Francisco, CA 94087	822-555-3131
S0415	Art Peters	Manager	42000	D003	14 – 8th Avenue, New York, NY 10012	212-371-3000

StaffDistributionCenter table has redundant data; the details of a distribution center are repeated for every member of staff.

HALMSTAD UNIVERSITY

What can you tell about the following relations?

- Alternative

Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Now, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation, to represent where each member of staff is located.

Data Redundancy and Update Anomalies

- Major aim of relational database design is to group attributes into relations to minimize data redundancy.
- Potential benefits for implemented database include:
  - Updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
  - Reduction in the file storage space required by the base relations thus minimizing costs.
- However, relations can contain redundant data.
  - Redundant data wastes disk space and creates maintenance problems.
    - If data that exists in more than one place must be changed, the data must be changed in exactly the same way in all locations.
  - Relations that contain redundant information may potentially suffer from update anomalies.

HALMSTAD UNIVERSITY

Data Redundancy and Update Anomalies

- Types of update anomalies include
  - – Insertion anomaly
    - Can’t insert because some data is missing
  - – Deletion anomaly
    - Deleting a tuple accidently causes other data to be lost.
    - Modification anomaly
  - – A same data is stored in more than one place, more than one place to update.

Normalization

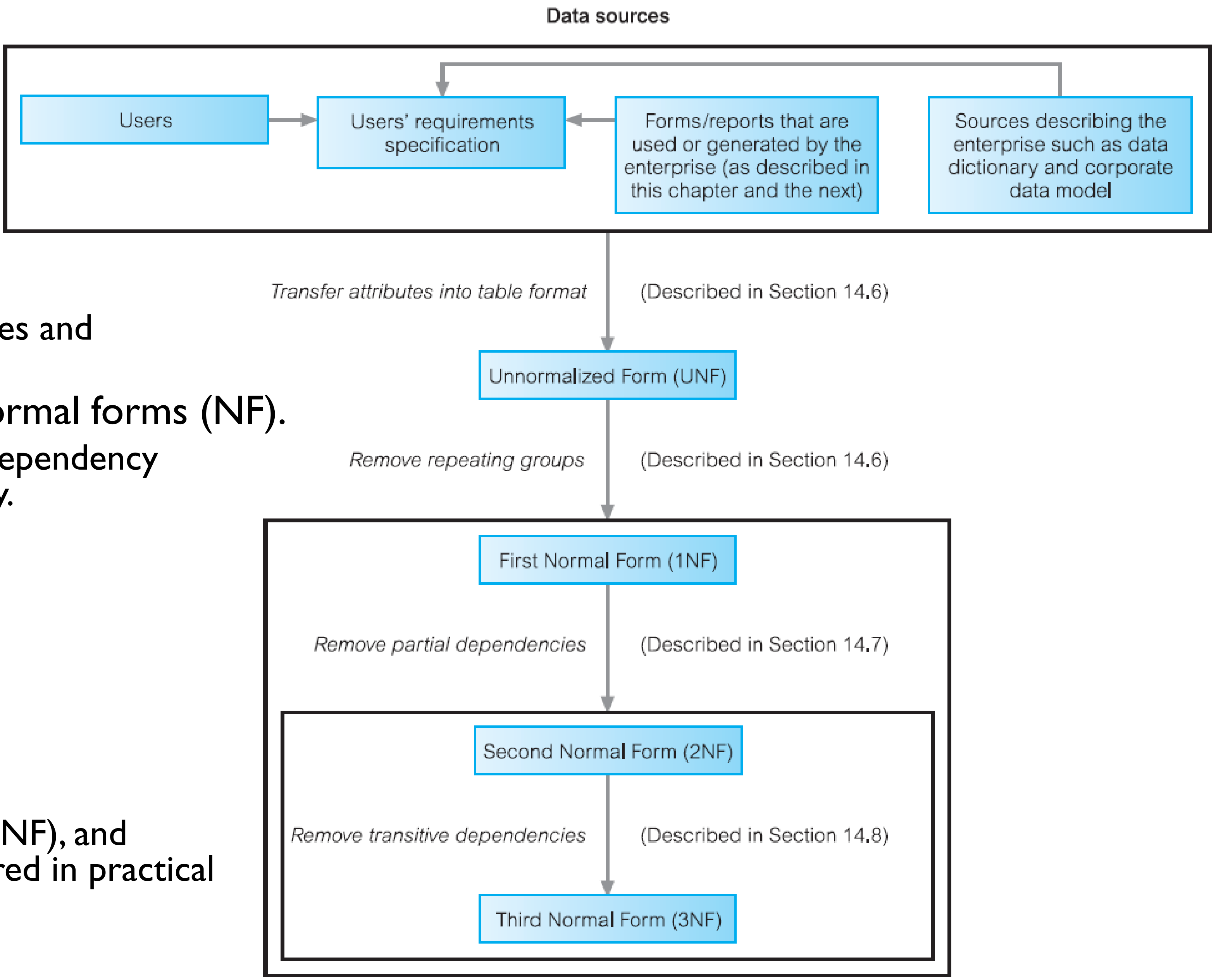
- Technique for analyzing and producing a set of suitable (appropriately structured) relations based on their different functional dependencies and primary key.
  - Helps avoid data redundancy, insertion, update & deletion anomaly.
  - Facilitates access and maintenance of the data.
  - Takes up minimal storage space.

- Normalization provided mechanisms for transforming schemas in order to remove redundancy.
  - Often executed as a series of steps.
  - Each step corresponds to a specific normal form, which has known properties.

- Characteristics of a suitable set of relations:
  - The minimal number of attributes necessary to support the data requirements of the enterprise;
  - Attributes with a close logical relationship are found in the same relation;
  - Minimal redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

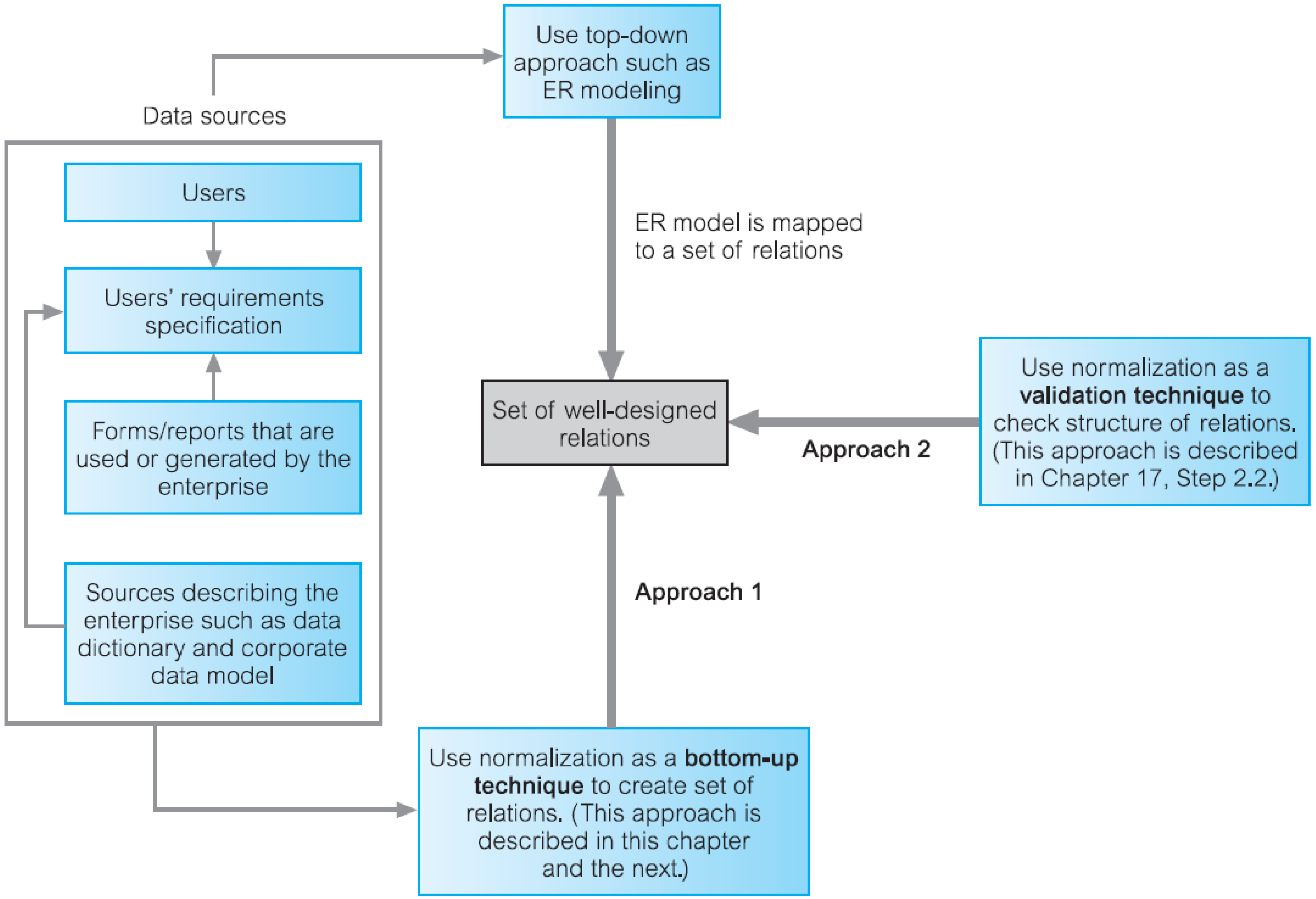
Normalization

- Aim
  - To eliminate undesirable dependencies and ensure data integrity.
- Normalization theory defines six normal forms (NF).
  - Each normal form involves a set of dependency properties that a schema must satisfy.
- Normal Forms
  - First Normal Form (1NF)
    - Ensures atomicity of data.
  - Second Normal Form (2NF)
    - Ensures no partial dependencies.
  - Third Normal Form (3NF)
    - Ensures no transitive dependencies.
  - 4NF (Boyce Codd Normal Form-BCNF), and 5NF do exist, but are rarely considered in practical design.


Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise (as described in this chapter and the next)
Sources describing the enterprise such as data dictionary and corporate data model
Transfer attributes into table format
(Described in Section 14.6)
Unnormalized Form (UNF)
Remove repeating groups
(Described in Section 14.6)
First Normal Form (1NF)
Remove partial dependencies
(Described in Section 14.7)
Second Normal Form (2NF)
Remove transitive dependencies
(Described in Section 14.8)
Third Normal Form (3NF)

HALMSTAD UNIVERSITY

How Normalization Supports Database Design


Use top-down approach such as ER modeling
Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise
Sources describing the enterprise such as data dictionary and corporate data model
ER model is mapped to a set of relations
Set of well-designed relations
Approach 2
Use normalization as a **validation technique** to check structure of relations. (This approach is described in Chapter 17, Step 2.2.)
Approach 1
Use normalization as a **bottom-up technique** to create set of relations. (This approach is described in this chapter and the next.)

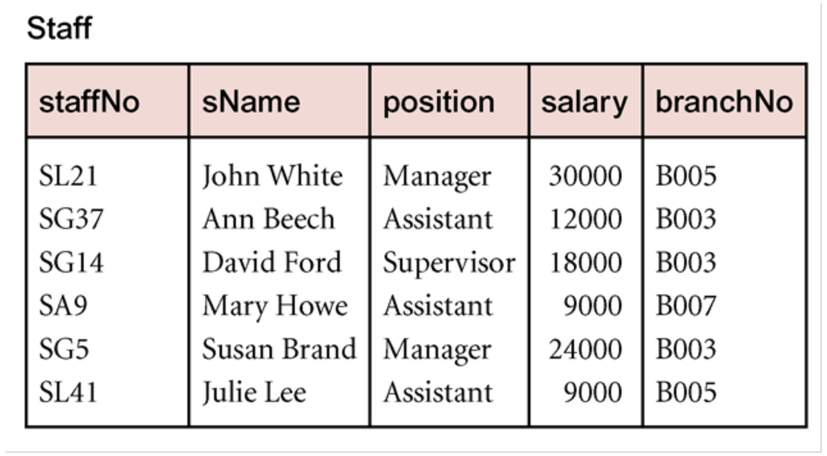
HALMSTAD UNIVERSIT

Normalization

- Advantages
  - Minimizes data redundancy and saves storage space.
  - Improves data integrity (accurate and consistent).
  - Protects from anomalies.
  - Facilitates querying and simplifies updates.
- Disadvantages
  - Many small tables may degrade performance or exceed open file and memory capacities.
  - Normalized databases often involve more tables and JOIN operations, making queries more complex.
  - Joining tables at runtime can be resource-intensive, particularly in large databases.
  - In some cases, for performance reasons, denormalization (introducing redundancy) might be considered, which goes against normalization principles.
  - Design Overhead.

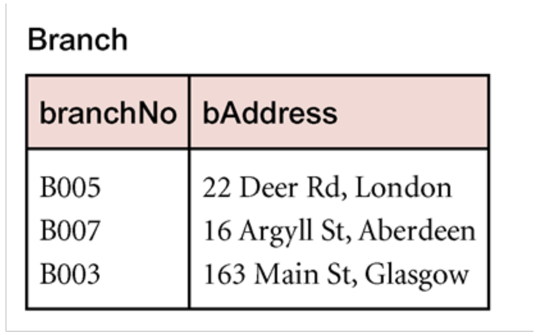
Problems associated with data redundancy

- Illustrated by comparing the Staff and Branch relations with the StaffBranch relation.



Staff

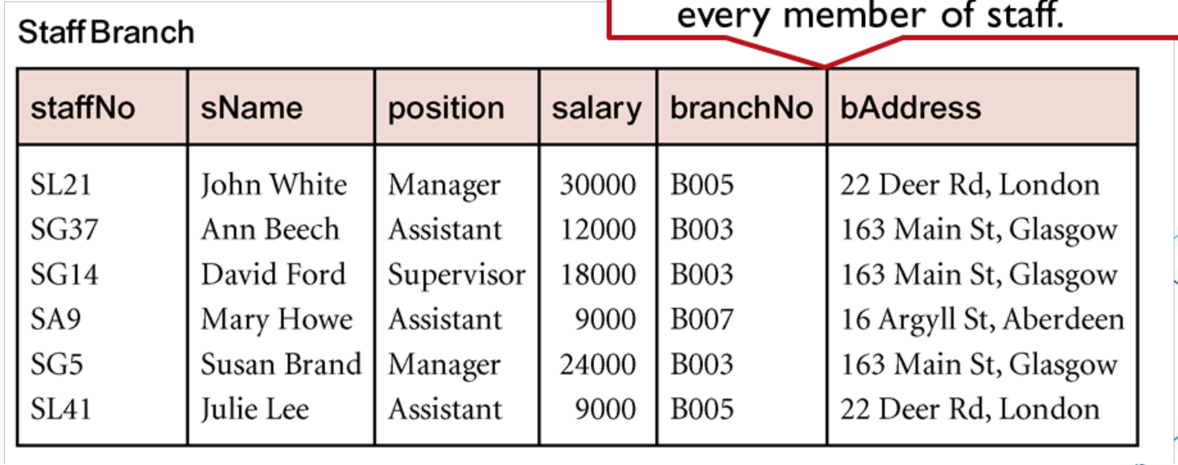
staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005



Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Redundant data
- Details of a branch are repeated for every member of staff.



StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

HALMSTAD UNIVERSITY

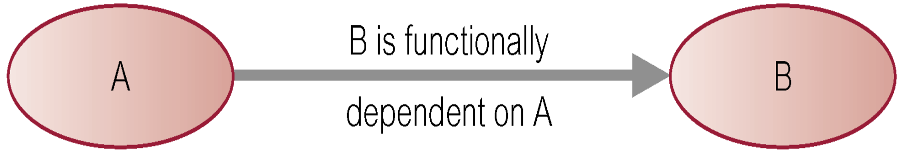
Example from 2014 - Watt - Database Design 2e

- UNF
  - Student_Grade_Report (StudentNo, StudentName, Major, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)
- 1NF
  - Student (StudentNo, StudentName, Major)
  - StudentCourse (StudentNo, CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation, Grade)
- 2NF
  - Student (StudentNo, StudentName, Major)
  - CourseGrade (StudentNo, CourseNo, Grade)
  - CourseInstructor (CourseNo, CourseName, InstructorNo, InstructorName, InstructorLocation)
- 3NF
  - Student (StudentNo, StudentName, Major)
  - CourseGrade (StudentNo, CourseNo, Grade)
  - Course (CourseNo, CourseName, InstructorNo)
  - Instructor (InstructorNo, InstructorName, InstructorLocation)

HALMSTAD UNIVERSITY

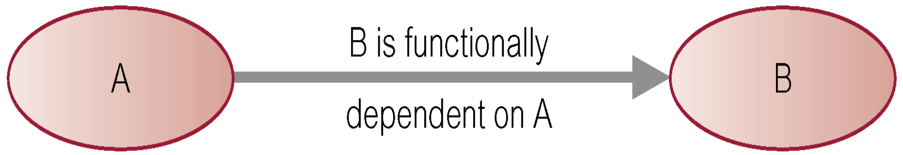
Functional Dependencies

- Play a crucial role in database normalization, helping to minimize redundancy and ensure data integrity.
- **Describes the relation of one attribute to another attribute**.
- For example, consider a table with columns a and b, where b is functionally dependent on a (denoted a → b).
  - If we know the value of a, we find only one value of b in all the records that has this value of a, at any moment in time. However, for a given value of b there may be several different values of a.
- The determinant of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.
  - A is the determinant of B.



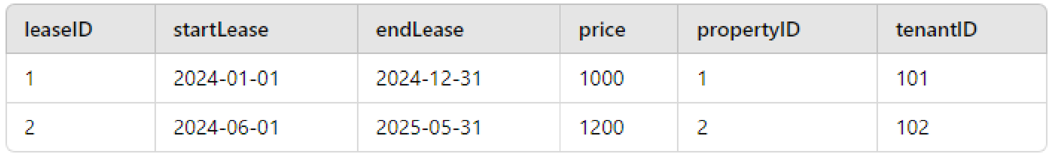
Functional Dependencies

- Play a crucial role in database normalization, helping to minimize redundancy and ensure data integrity.
- **Describes the relation of one attribute to another attribute**.
- A functional dependency is a constraint between two sets of attributes in a relation. It indicates that the value of one attribute (or a set of attributes) uniquely determines the value of another attribute.
- If A → B, it means that for each unique value of A, there is only one corresponding value of B in the relation R.
  - If we know the value of A, we find only one value of B in all the records that has this value of A, at any moment in time. However, for a given value of B there may be several different values of A.
- The determinant of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.
  - A is the determinant of B.



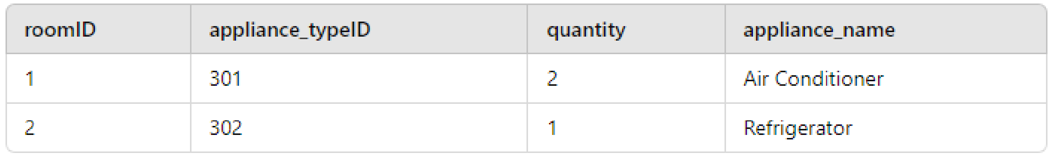
Types of Functional Dependencies

- **Functional Dependency**
  - When a non-key attribute is functionally dependent on the entire primary key, and not just part of it.
    - This means that every non-key attribute is uniquely determined by the full primary key in a relation/talbe.



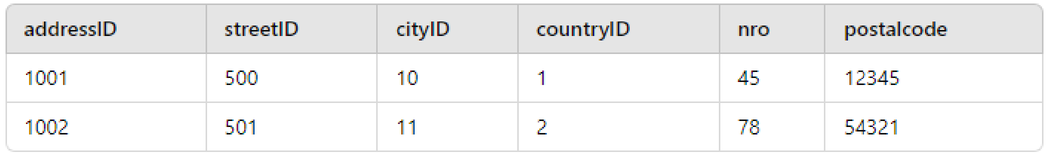
leaseID	startLease	endLease	price	propertyID	tenantID
1	2024-01-01	2024-12-31	1000	1	101
2	2024-06-01	2025-05-31	1200	2	102

- **Partial Dependency**
  - When a non-key attribute is functionally dependent on part of a composite primary key, rather than the whole composite primary key, ie. when one of the primary keys determines another attribute or group of attributes.



roomID	appliance_typeID	quantity	appliance_name
1	301	2	Air Conditioner
2	302	1	Refrigerator

- **Transitive Dependency**
  - When a non-key attribute is functionally dependent on another non-key attribute, instead of depending directly on the primary key.



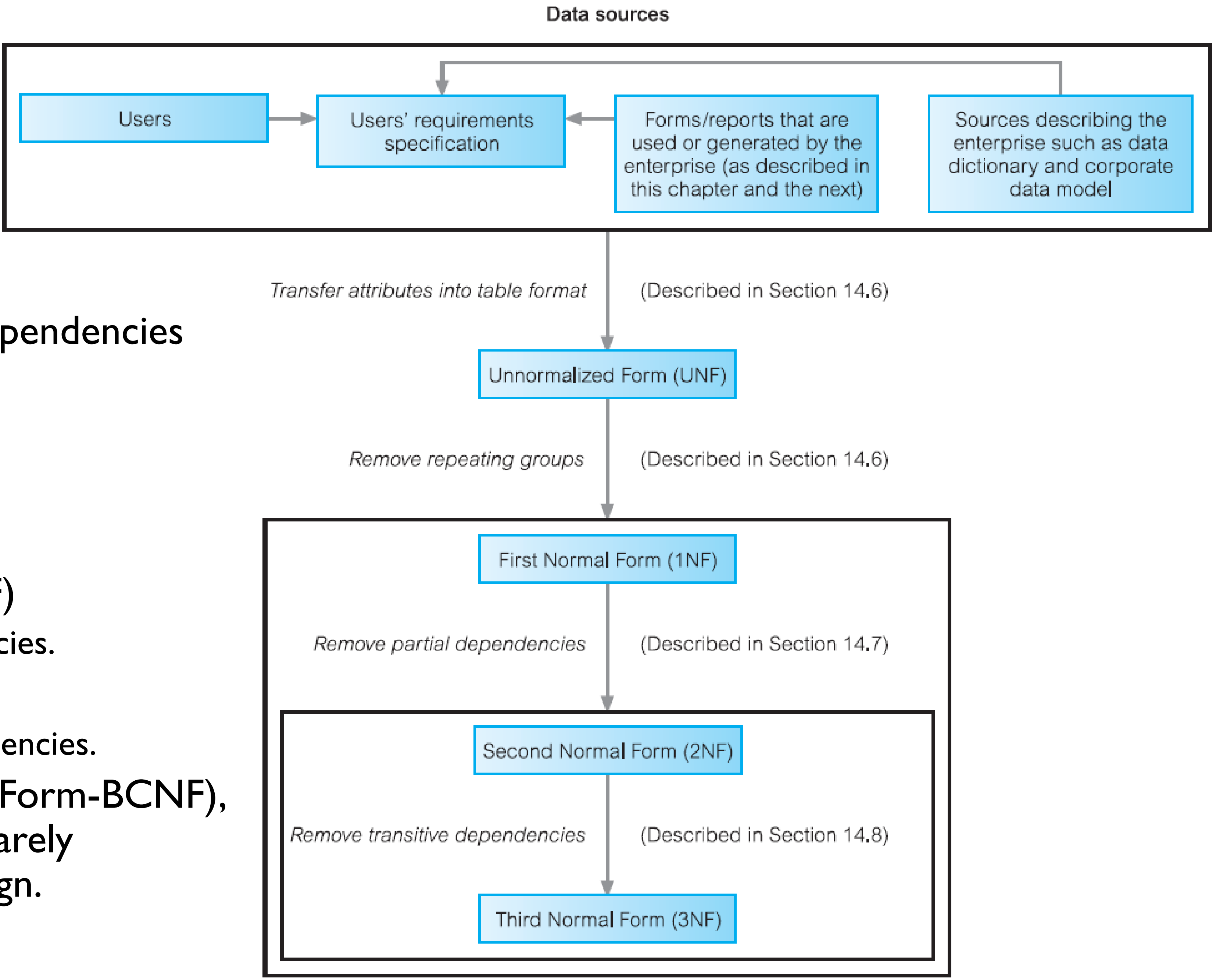
addressID	streetID	cityID	countryID	nro	postalcode
1001	500	10	1	45	12345
1002	501	11	2	78	54321

Main characteristics of functional dependencies used in normalization

- Functional dependencies are used to identify anomalies (insert, update, delete anomalies) and guide the normalization process to reduce redundancy.
- Normalization Forms Using Functional dependencies:
  - – First Normal Form (1NF): Removes repeating groups.
  - – Second Normal Form (2NF): Eliminates partial dependencies.
  - – Third Normal Form (3NF): Eliminates transitive dependencies.
- Understanding functional dependencies is crucial to designing efficient databases that minimize redundancy while ensuring data consistency

Normalization

- Aim
  - To eliminate undesirable dependencies and ensure data integrity.
- Normal Forms
  - First Normal Form (1NF)
    - Ensures atomicity of data.
  - Second Normal Form (2NF)
    - Ensures no partial dependencies.
  - Third Normal Form (3NF)
    - Ensures no transitive dependencies.
  - 4NF (Boyce Codd Normal Form-BCNF), and 5NF do exist, but are rarely considered in practical design.


Data sources
Users
Users' requirements specification
Forms/reports that are used or generated by the enterprise (as described in this chapter and the next)
Sources describing the enterprise such as data dictionary and corporate data model
Transfer attributes into table format
(Described in Section 14.6)
Unnormalized Form (UNF)
Remove repeating groups
(Described in Section 14.6)
First Normal Form (1NF)
Remove partial dependencies
(Described in Section 14.7)
Second Normal Form (2NF)
Remove transitive dependencies
(Described in Section 14.8)
Third Normal Form (3NF)

HALMSTAD UNIVERSITY

Solution

- **Partial Dependency**
  - Entities/Relations with composite keys (like room_furniture, room_appliance, room_heating, and property_special_feature) may contain partial dependencies.
  - Solution
    - **Move dependent attributes to a separate table with a new primary key.**
    - **Link the new table back to the original table using a foreign key.**
- **Transitive Dependency**
  - property (owner details)
  - lease (tenant details)
  - address (street → city → country hierarchy)
  - room_appliance (appliance type → name)
  - Solution
    - **Ensure non-key attributes depend only on the primary key.**

Dependency Type	Definition	Example in ER Model	Fix (Normalization Step)
Functional Dependency	A non-key attribute is uniquely determined by the primary key.	leaseID → startLease, endLease, price, propertyID, tenantID	No fix needed (this is normal)
Partial Dependency	A non-key attribute is dependent on only part of a composite key.	(roomID, furniture_typeID) → furniture_name	Move dependent attributes to separate tables (2NF)
Transitive Dependency	A non-key attribute depends on another non-key attribute instead of the primary key.	propertyID → ownerID → owner_firstname, owner_lastname	Move transitive dependencies to separate tables (3NF)

Unnormalized Form (UNF)

- A relation that contains one or more repeating groups.

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

- To create an unnormalized table
- Transform the data from the information source (e.g. form) into table format with columns and rows.

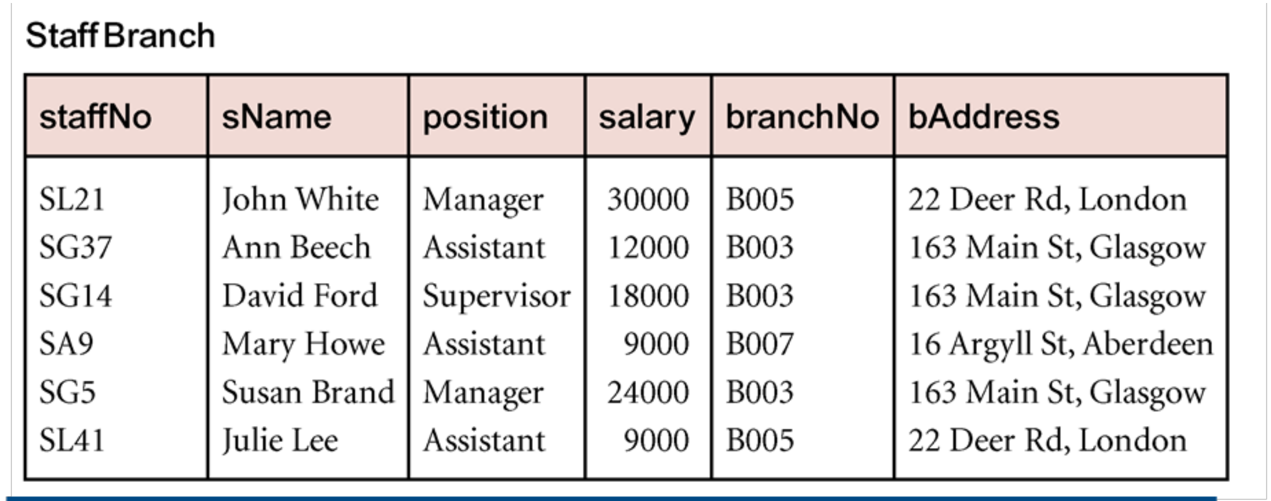
HALMSTAD UNIVERSITY

First normal form (1NF)

- A relation in which the intersection of each row and column contains one and only one value.
- Only 1NF is critical in creating appropriate tables for relational databases. All subsequent normal forms are optional.
- UNF to 1NF
  - Nominate an attribute or group of attributes to act as the primary key for the unnormalized table.
  - Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).
  - Remove the repeating group by
    - Option 1: Entering appropriate data into the empty columns of rows containing the repeating data ('flattening' the table).
    - Option 2: Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

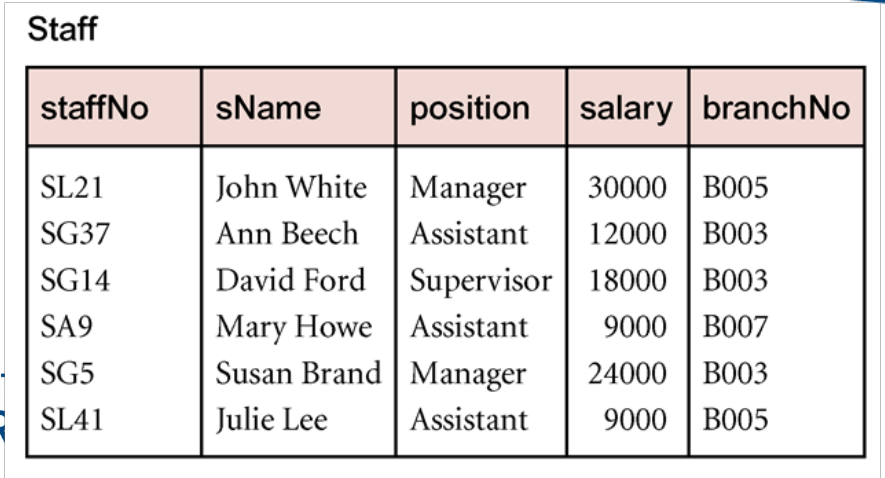
HALMSTAD UNIVERSITY

From UNF to 1NF



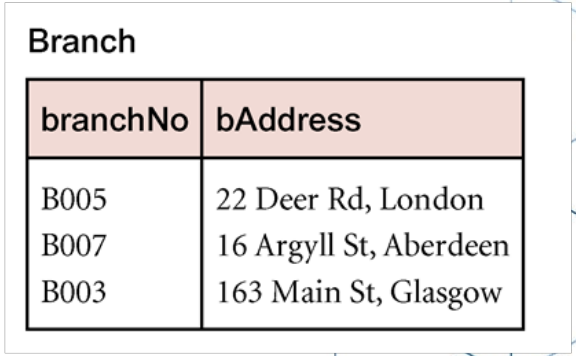
Staff Branch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London



Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

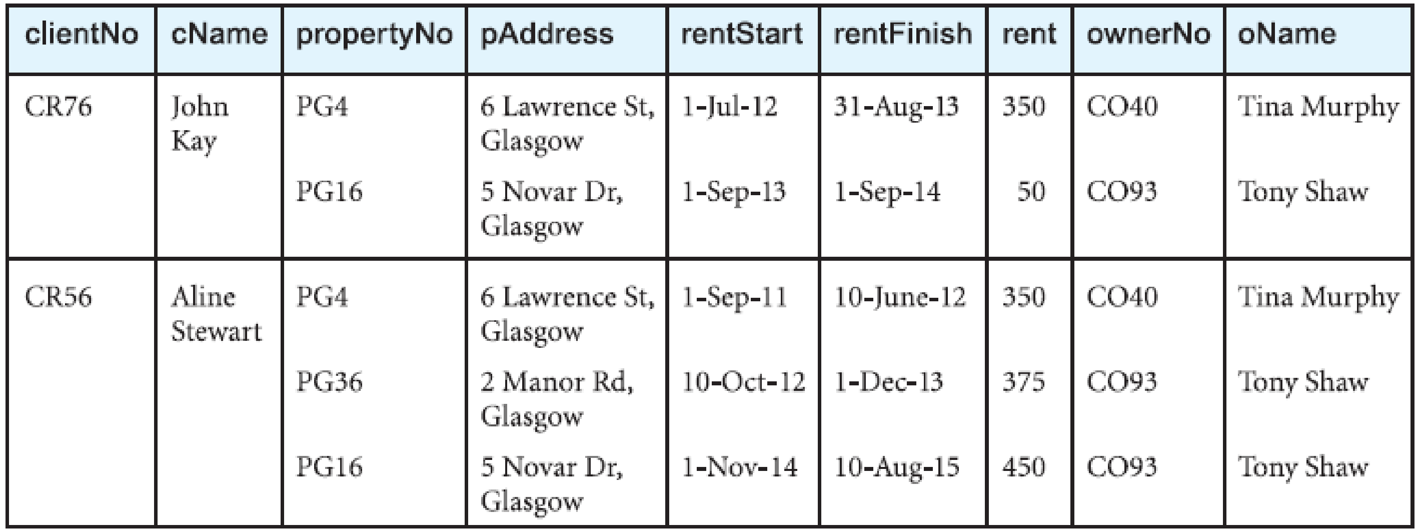


Branch

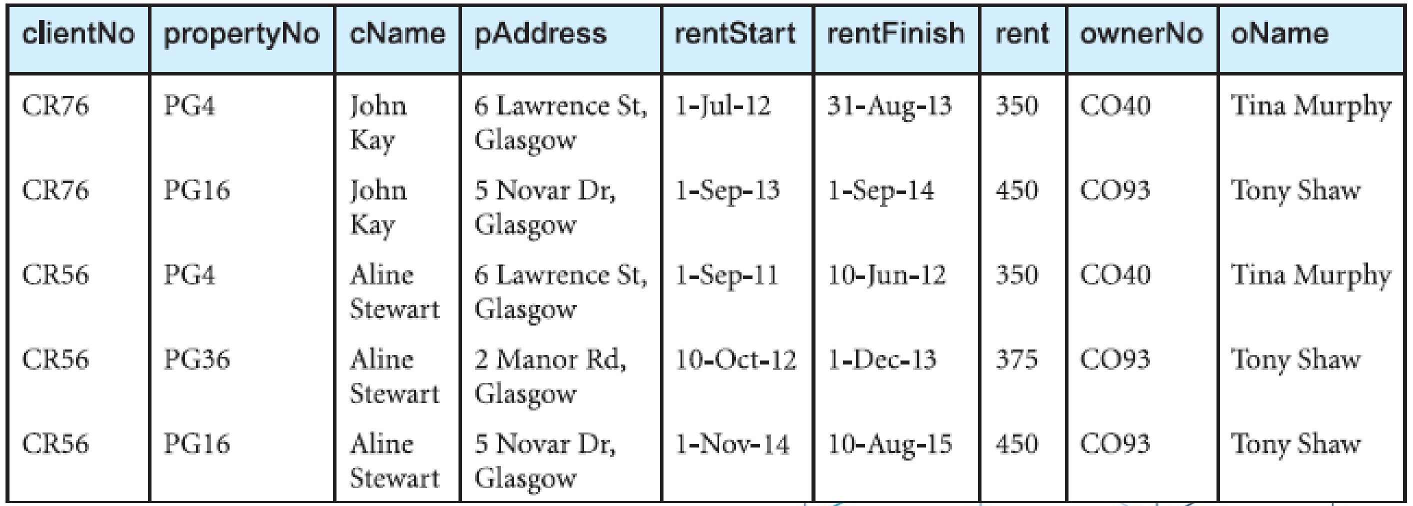
branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

From UNF to 1NF

- A table is in 1NF if:
  - The table has a primary key.
  - All attributes contain atomic values (no repeating groups or multivalued attributes).



clientNo	cName	propertyNo	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	John Kay	PG4	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
		PG16	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	50	CO93	Tony Shaw
CR56	Aline Stewart	PG4	6 Lawrence St, Glasgow	1-Sep-11	10-June-12	350	CO40	Tina Murphy
		PG36	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
		PG16	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw



clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw

HALMSTAD UNIVERSITY

Comments or Questions?

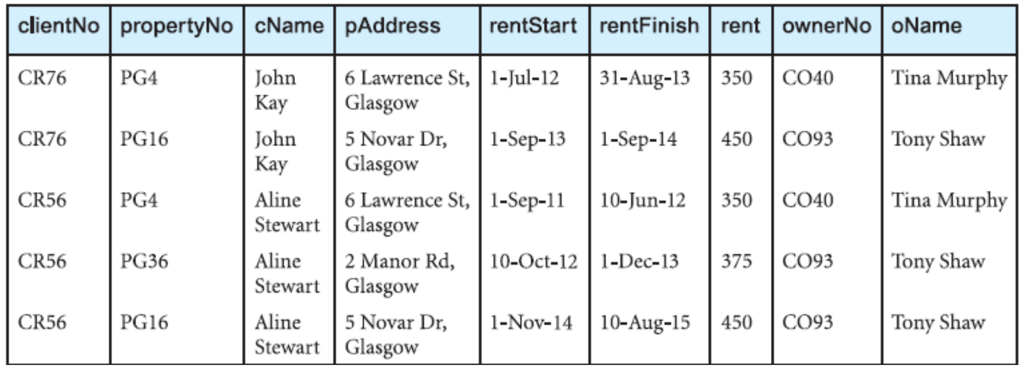
HALMSTAD UNIVERSITY

Second Normal Form (2NF)

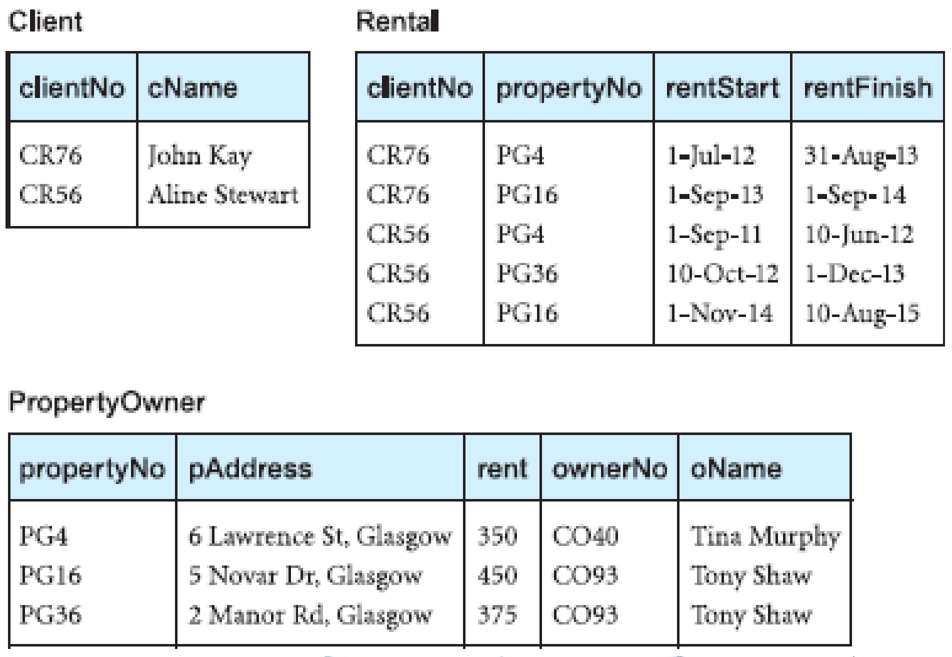
- A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.
- To assess whether a table breaks 2NF form requires identification of the primary key and functional dependencies associated with that table.
  - – Based on the concept of full functional dependency.
  - – Full functional dependency indicates that if
    - A and B are attributes of a relation,
    - B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.
- 2NF only applies to tables with composite primary keys.

From 1NF to 2NF

- Identify the primary key for the 1NF relation and
- Identify the functional dependencies in the relation.
- **No partial dependencies** (every non-key attribute depends on the entire primary key, not just part of it).
  - If partial dependencies exist on the primary key remove them by placing then in a new relation along with a copy of their determinant.



clientNo	propertyNo	cName	pAddress	rentStart	rentFinish	rent	ownerNo	oName
CR76	PG4	John Kay	6 Lawrence St, Glasgow	1-Jul-12	31-Aug-13	350	CO40	Tina Murphy
CR76	PG16	John Kay	5 Novar Dr, Glasgow	1-Sep-13	1-Sep-14	450	CO93	Tony Shaw
CR56	PG4	Aline Stewart	6 Lawrence St, Glasgow	1-Sep-11	10-Jun-12	350	CO40	Tina Murphy
CR56	PG36	Aline Stewart	2 Manor Rd, Glasgow	10-Oct-12	1-Dec-13	375	CO93	Tony Shaw
CR56	PG16	Aline Stewart	5 Novar Dr, Glasgow	1-Nov-14	10-Aug-15	450	CO93	Tony Shaw



Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

Comments or Questions?

HALMSTAD UNIVERSITY

Third Normal Form (3NF)

- A relation that is in 2NF and in which no non-primary-key attribute is transitively dependent on the primary key.
- Based on the concept of transitive dependency.
  - Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if A → B and B → C, then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).

From 2NF to 3NF

- Identify the primary key in the 2NF relation and functional dependencies.
- **No transitive dependencies** (non-key attributes must depend only on the primary key, not another non-key attribute).
  - If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant.

Client

clientNo	cName
CR76	John Kay
CR56	Aline Stewart

Rental

clientNo	propertyNo	rentStart	rentFinish
CR76	PG4	1-Jul-12	31-Aug-13
CR76	PG16	1-Sep-13	1-Sep-14
CR56	PG4	1-Sep-11	10-Jun-12
CR56	PG36	10-Oct-12	1-Dec-13
CR56	PG16	1-Nov-14	10-Aug-15

PropertyOwner

propertyNo	pAddress	rent	ownerNo	oName
PG4	6 Lawrence St, Glasgow	350	CO40	Tina Murphy
PG16	5 Novar Dr, Glasgow	450	CO93	Tony Shaw
PG36	2 Manor Rd, Glasgow	375	CO93	Tony Shaw

$\rightarrow$

PropertyForRent

propertyNo	pAddress	rent	ownerNo
PG4	6 Lawrence St, Glasgow	350	CO40
PG16	5 Novar Dr, Glasgow	450	CO93
PG36	2 Manor Rd, Glasgow	375	CO93

Owner

ownerNo	oName
CO40	Tina Murphy
CO93	Tony Shaw

propertyNo → ownserNo and ownerNo → oName

Comments or Questions?

HALMSTAD
UNIVERSITY

General Definitions of 2NF and 3NF

- Second normal form (2NF)
  - A relation that is in first normal form and every non-primary-key attribute is fully functionally dependent on any candidate key.
- Third normal form (3NF)
  - A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on any candidate key.
- Other normalization forms
  - Although other levels of normalization are possible, third normal form is considered the highest level necessary for most applications.
  - Fourth normal form, also called Boyce Codd Normal Form (BCNF), and fifth normal form do exist, but are rarely considered in practical design.

Comments or Questions?

Logical Database Design for the Relational Model

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

HALMSTAD UNIVERSITY

Logical data model and relation diagram for DreamHome

Branch (branchNo, street, city, postcode, mgrStaffNo)
Primary Key branchNo
Alternate Key postcode
Foreign Key mgrStaffNo references Manager(staffNo)

Telephone (telNo, branchNo)
Primary Key telNo
Foreign Key branchNo references Branch(branchNo)

Staff (staffNo, fName, lName, position, sex, DOB, salary, supervisorStaffNo, branchNo)
Primary Key staffNo
Foreign Key supervisorStaffNo references Staff(staffNo)
Foreign Key branchNo references Branch(branchNo)

Manager (staffNo, mgrStartDate, bonus)
Primary Key staffNo
Foreign Key staffNo references Staff(staffNo)

PrivateOwner (ownerNo, fName, lName, address, telNo)
Primary Key ownerNo

BusinessOwner (ownerNo, bName, bType, contactName, address, telNo)
Primary Key ownerNo
Alternate Key bName
Alternate Key telNo

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)
Primary Key propertyNo
Foreign Key ownerNo references PrivateOwner(ownerNo) and BusinessOwner(ownerNo)
Foreign Key staffNo references Staff(staffNo)
Foreign Key branchNo references Branch(branchNo)

Viewing (clientNo, propertyNo, dateView, comment)
Primary Key clientNo, propertyNo
Foreign Key clientNo references Client(clientNo)
Foreign Key propertyNo references PropertyForRent(propertyNo)

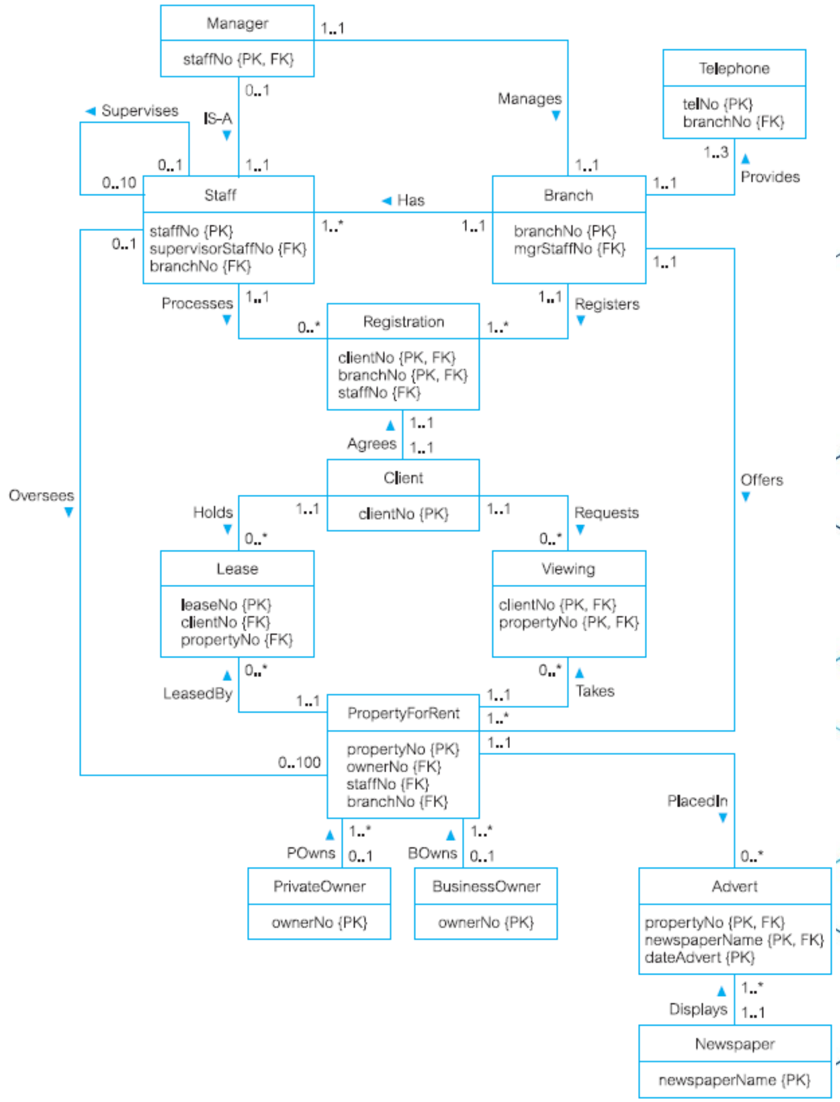
Client (clientNo, fName, lName, telNo, eMail, prefType, maxRent)
Primary Key clientNo
Alternate Key eMail

Registration (clientNo, branchNo, staffNo, dateJoined)
Primary Key clientNo, branchNo
Foreign Key clientNo references Client(clientNo)
Foreign Key branchNo references Branch(branchNo)
Foreign Key staffNo references Staff(staffNo)

Lease (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)
Primary Key leaseNo
Alternate Key propertyNo, rentStart
Alternate Key clientNo, rentStart
Foreign Key clientNo references Client(clientNo)
Foreign Key propertyNo references PropertyForRent(propertyNo)
Derived deposit (PropertyForRent.rent*2)
Derived duration (rentFinish – rentStart)

Newspaper (newspaperName, address, telNo, contactName)
Primary Key newspaperName
Alternate Key telNo

Advert (propertyNo, newspaperName, dateAdvert, cost)
Primary Key propertyNo, newspaperName, dateAdvert
Foreign Key propertyNo references PropertyForRent(propertyNo)
Foreign Key newspaperName references Newspaper(newspaperName)



Comments or Questions?

HALMSTAD UNIVERSITY

Next

- Database programming
  - Lectures 6 to 8
    - If needed, finish DB design.
    - SQL
      - Data definition, manipulation and querying in SQL