

Database Modeling and Database Systems — Units 3 & 4

Dr.-Ing Anna Androvitsanea

IU Internationale Hochschule GmbH

November 5, 2023

TOPIC OUTLINE

Manipulating Records in Databases

6

NoSQL Database System

7

UNIT 3

CONCEPTION AND MODELING OF RELATIONAL DATABASES

Study goals

- ▶ Design a database using an Entity Relationship Model (ERD).
- ▶ Model relationships among tables within an ERD.
- ▶ Normalize the tables in a database.

EXPLAIN SIMPLY

1. Why is it necessary to design a database before starting implementation?
2. Is data redundancy a desired or undesired feature in a relational database?
3. Must a Primary Key be defined in each table in a relational database?

Introduction

Before a relational database can be created, a model of the data to be stored is created and then optimized for efficient use as a relational data schema. This lesson introduces the **Entity Relationship Model**—a standard data modeling structure—and teaches which relationship types to consider. We will also discuss the concept of normal forms.

The Entity Relationship Model

The Entity Relationship Model (ERM) is an iterative process of creating a structure for a relational database. This process involves:

- ▶ Creating a technical data model from a business data model.
- ▶ Determining how data are structured in a Database Management System (DBMS).
- ▶ Using graphical modeling languages for design and documentation.

Elements of an ER Model

- ▶ **Entities:** Represented as rectangles in diagrams.
- ▶ **Attributes:** Shown as ovals connected to their entities.
- ▶ **Keys:** Attributes that uniquely identify an entity instance, often underlined.
- ▶ **Relationships:** Illustrated as lines connecting entities; can have attributes too.

TIP: Relationships in ER models are akin to those in UML class diagrams.

Graphical Notation of Entities and Attributes in ER Models

Different notations for ER diagrams:

- ▶ **Chen notation:** Uses rectangles for entities and ovals for attributes.
- ▶ **Martin notation** (Crow's foot): Displays entities in rectangles with attributes listed inside.
- ▶ **UML class diagram:** Models entities as classes with attributes inside.

Relationships and Cardinalities in ER Models

Three main relationship types:

1. **1:1 Relationships:** One entity is related to one other entity.
2. **1:N Relationships:** One entity is associated with many other entities.
3. **N:M Relationships:** Many entities are associated with many other entities.

1:1 Relationships

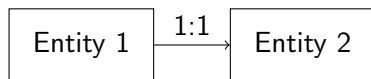


Figure: Schematic representation of a 1:1 relationship.

- ▶ Bijective: Each element in one entity set is associated with exactly one element in another entity set.
- ▶ Variants with optionality (1:C, C:C) cannot be represented with Chen notation.

1:N Relationships

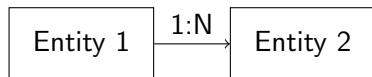


Figure: Schematic representation of a 1:N relationship.

- ▶ One entity is related to several others.
- ▶ Variants include 1:CN, C:CN, and C:N, reflecting the optionality and number of connections.

N:M Relationships

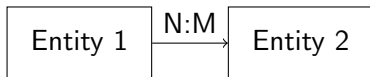


Figure: Schematic representation of an N:M relationship.

- ▶ Several entities on both sides are related to each other.
- ▶ Variants such as N:CM and CN:CM reflect optionality and numbers of connections.

ER - Model for the hospital db

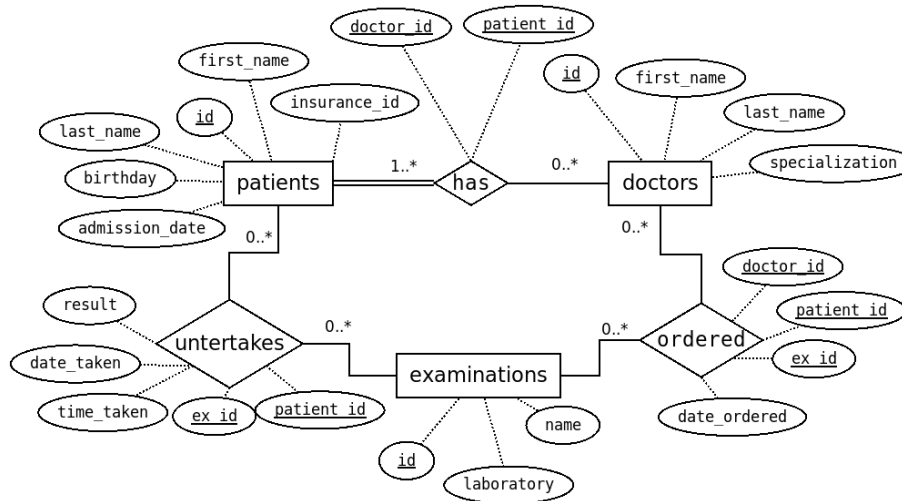


Figure: ER - Model for the hospital db on Chen notation

ER Model - Hospital Database Overview

The hospital database consists of **three entities**:

- ▶ **Patients**: Identified by **id**, including personal details and **admission_date**.
- ▶ **Doctors**: Identified by **id**, including names and **specialization**.
- ▶ **Examinations**: Identified by **id**, with **name** and the **laboratory**.

ER Model - Relationships Overview

The database also includes **three relationships**:

- ▶ **has**: Between **patients** and **doctors**.
- ▶ **undertakes**: Between **patients** and **examinations**.
- ▶ **ordered**: Between **doctors** and **examinations**.

Patients-Doctors Relationship (has)

- ▶ Connects patients to doctors with **patient_id** and **doctor_id**.
- ▶ Each patient must have at least one doctor (**full participation**).
- ▶ A doctor may have zero or more patients (**partial participation**).

Patients-Examinations Relationship (undertakes)

- ▶ Associates tests with patients through **patient_id** and **ex_id**.
- ▶ Not all patients are required to take tests (**partial participation**).
- ▶ Tests can be taken by multiple patients.
- ▶ Includes **results**, **date_taken**, and **tim_taken**.

Doctors-Examinations Relationship (ordered)

- ▶ Links doctors to tests they order using **doctor_id** and **ex_id**.
- ▶ Doctors can order any number of tests (**partial participation**).
- ▶ Each test can be ordered by one doctor (**one-to-one cardinality**).
- ▶ Includes **date_ordered** for tracking.

Recap

Up to now, we've covered:

- ▶ The importance of data modeling for relational databases.
- ▶ The Entity Relationship Model and its elements.
- ▶ Graphical notations and relationship types in ER models.

Next steps: Practice modeling with ER diagrams and understand the implications of different relationship types on database design.

Normal Forms of Databases

Database Normalization

What is Database Normalization?

- ▶ A process for organizing data in a database.
- ▶ It involves creating tables and establishing relationships between those tables according to rules designed to protect the data and to make the database more flexible by eliminating redundancy and inconsistent dependency.
- ▶ Normalization is used to minimize the duplication of information and to ensure that only related data is stored in each table.

The Goals of Normalization

- ▶ To free the database from unwanted insertions, updates, and deletion dependencies.
- ▶ To reduce the need for restructuring the collection of relations, as new types of data are introduced, and thus increase the life span of application programs.
- ▶ To make the relational model more informative to users and to ensure that the data is represented accurately and consistently.
- ▶ To make the database neutral to the query statistics, where it performs equally well regardless of the types of data retrieval queries that are issued.

Overview of Normal Forms

- ▶ Normal forms are defined as the standards for the organization of data in databases.
- ▶ There are several normal forms, each with more strict rules than the previous one: First Normal Form (1NF), Second Normal Form (2NF), Third Normal Form (3NF), and Boyce-Codd Normal Form (BCNF).
- ▶ Higher normal forms (4NF and 5NF) deal with more complex scenarios and are not as commonly applied as the first three forms.
- ▶ The aim is to achieve the desired form of normalization suitable for the use case and to balance the trade-offs between data redundancy and query performance.

Dependencies and Normal Forms

Table 26: Table for Article Delivery Example

Article Delivery		
Vendor	Customer	Articles
MediaService, Hattstadt	Sophie Weber, 182932	Edge Of Tomorrow; EAN 838883
MovieGroup, Hamburg	Sophie Weber, 182932	Cry Baby; EAN 1222233
MovieGroup, Hamburg	Ralf Fischer, 488389	In the Morning; ISBU XX-NHN-223
FilmService, Berlin	Maria Bolz, 299376	Edge Of Tomorrow; EAN 838883

Source: Jörg Dreikauss, 2022.

Normal Forms

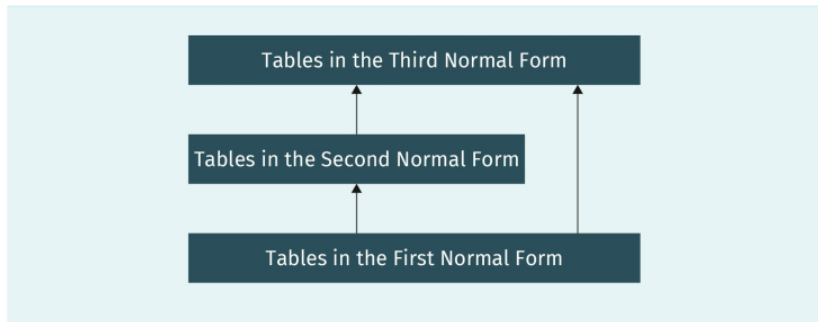


Figure: Dependencies of the normal forms. Adapted from Steiner, 2014, p. 78.

First Normal Form (1NF)

- ▶ Attributes must contain single, atomic values.
- ▶ Records are uniquely identifiable by a primary key.

Normalization to 1NF:

- ▶ Split composite attributes into individual columns.
- ▶ Define a primary key to uniquely identify records.

The diagram shows a table named 'ArticleDelivery' with the following columns: Supplier Name, Supplier City, Customer LastName, Customer FirstName, Customer Number, ArticleName, and ArticleID. The first three columns are grouped under the heading 'Elements of the Primary Key'. Arrows point from this heading to the 'Customer' sub-header and the 'ArticleID' column. Below the table, two text boxes explain dependencies: one stating that 'CustomerLastName' and 'CustomerFirstName' can be inferred from 'CustomerNumber' (making 'ArticleID' not required), and another stating that 'ArticleName' can be inferred from 'ArticleID' and 'CustomerNumber' (making it not required).

Supplier		Customer			ArticleName	ArticleID
Name	City	LastName	FirstName	Number		
Media Service	Hattstadt	Weber	Sophie	182932	Edge of Tomorrow	EAN 838883
Movie Group	Hamburg	Weber	Sophie	182932	Cry Baby	EAN 1222233
Movie Group	Hamburg	Fischer	Ralf	488389	In the Morning	ISBU XX-NHN-223
Film Service	Berlin	Bolz	Maria	299376	Edge of Tomorrow	EAN 838883

Elements of the Primary Key

The value of CustomerLastName and CustomerFirstName can be inferred from the CustomerNumber; ArticleID is not required.

The value of ArticleName can be inferred from the ArticleID; CustomerNumber is not required.

Figure: Dependencies in First Normal Form. Source: Jörg Dreikauss, 2022.

Normalization to 1NF:

Table 27: First Normal Form with Compound Primary Key

ArticleDelivery

Supplier-Name	SupplierCity	Customer Last-Name	Customer First-Name	Customer Number	ArticleName	ArticleID
MediaService	Hattstadt	Weber	Sophie	182932	Edge of Tomorrow	EAN 838883
Movie Group	Hamburg	Weber	Sophie	182932	Cry Baby	EAN 1222233
Movie Group	Hamburg	Fischer	Ralf	488389	In the Morning	ISBU XX-NHN-223
FilmService	Berlin	Bolz	Maria	299376	Edge of Tomorrow	EAN 838883

Normalization to 1NF:s

Table 28: First Normal Form with Transformed Table Including Simple Primary Key. Part 1

ArticleDelivery			
ArticleDeliveryID	SupplierName	SupplierCity	CustomerLastName
L01	MediaService	Hattstadt	Weber
L02	Movie Group	Hamburg	Weber
L03	Movie Group	Hamburg	Fischer
L04	FilmService	Berlin	Bolz

Source: Jörg Dreikauss, 2022.

Normalization to 1NF:s

Table 29: First Normal Form with Transformed Table Including Simple Primary Key. Part 2

ArticleDelivery

ArticleDeliveryID	CustomerFirst	CustomerNumber	ArticleName	ArticleID
L01	Sophie	182932	Edge of Tomorrow	EAN 838883
L02	Sophie	182932	Cry Baby	EAN 1222233
L03	Ralf	488389	In the Morning	ISBU XX-NHN-223
L04	Maria	299376	Edge of Tomorrow	EAN 838883

Source: Jörg Dreikauss, 2022.

Second Normal Form (2NF)

- ▶ Achieved by further decomposing 1NF tables with composite primary keys.
- ▶ Each attribute must depend on the *entire* primary key.

Transition to 2NF:

- ▶ Remove partial dependencies of attributes on the primary key.
- ▶ Create separate tables for entities with independent attributes.

Normalization to 2NF:s

Table 30: ArticleDelivery, Second Normal Form

ArticleDelivery			
SupplierName	SupplierCity	CustomerNumber	ArticleID
MediaService	Hattstadt	182932	EAN 838883
MovieGroup	Hamburg	182932	EAN 1222233
MovieGroup	Hamburg	488389	ISBU XX-NHN-223
FilmService	Berlin	299376	EAN 838883

Source: Jörg Dreikauss, 2022.

Normalization to 2NF:s

Table 31: Customer, Second Normal Form

Client		
CustomerLastName	CustomerFirstName	CustomerNumber
Weber	Sophie	182932
Fischer	Ralf	488389
Bolz	Maria	299376

Source: Jörg Dreikauss, 2022.

Normalization to 2NF:s

Table 32: ArticleDelivery, Second Normal Form

Articles	
ArticleName	ArticleID
Edge of Tomorrow	EAN 838883
Cry Baby	EAN 1222233
In the Morning	ISBU XX-NHN-233

Source: Jörg Dreikauss, 2022.

Third Normal Form (3NF)

- ▶ Based on 1NF and 2NF criteria.
- ▶ Non-key attributes must depend only on the primary key.
- ▶ No transitive dependencies between non-key attributes.

Achieving 3NF:

- ▶ Eliminate transitive dependencies by further decomposition.
- ▶ Ensure all non-key attributes depend directly on the key.

Normalization to 2NF:s

Figure 20: Third Normal Form with a Simple Primary Key

ArticleDelivery			
ArticleDeliveryID	SupplierName	CustomerNumber	ArticleID
L01	MediaService	182932	EAN 838883
L02	MovieGroup	182932	EAN 1222233
L03	MovieGroup	488389	ISBU XX-NHN-223
L04	FilmService	299376	EAN 838883

Supplier	
SupplierName	SupplierCity
MediaService	Hattstadt
MovieGroup	Hamburg
FilmService	Berlin

Articles	
ArticleName	ArticleID
Edge of Tomorrow	EAN 838883
Cry Baby	EAN 1222233
In the Morning	ISBU XX-NHN-223

Client		
Customer LastName	Customer FirstName	CustomerNumber

First Normal Form (1NF) Example

Table **Orders** is in 1NF, which requires that there are no repeating groups or arrays, and all values are atomic.

OrderID	CName	OrderDate	PID	PName	PQuantity
1	John Doe	2023-01-01	101	Apples	3
2	Jane Smith	2023-01-02	102	Oranges	5

Table: Example of an Orders table in First Normal Form (C: Customer, P: Product)

First Normal Form (1NF) Explanation

- ▶ The table 'Orders' is in 1NF because:
 - ▶ Each cell contains a single value.
 - ▶ Each record is unique and can be identified by 'OrderID'.
 - ▶ There are no repeating groups or arrays.
- ▶ This is the base requirement for a table to be considered normalized.
- ▶ The atomicity of values ensures that each piece of data is in its own domain.

Second Normal Form (2NF) Example

Tables **Orders** and **Order Details** are in 2NF, which requires no partial dependency of any column on the primary key.

OrderID	CName	OrderDate
1	John Doe	2023-01-01
2	Jane Smith	2023-01-02

Table: Orders Table

OrderID	PID	PName	PQuantity
1	101	Apples	3
2	102	Oranges	5

Table: Order Details Table

Second Normal Form (2NF) Explanation

- ▶ The 'Orders' and 'Order Details' tables are in 2NF because:
 - ▶ They were already in 1NF.
 - ▶ All non-key attributes are fully functional-dependent on the primary key.
 - ▶ We've removed partial dependencies by separating the 'Order Details' from the 'Orders' table.
- ▶ 2NF is about eliminating redundancy that occurs when a table has composite primary keys.

Third Normal Form (3NF) Example

Tables **Orders** and **Customers** are in 3NF, which requires that there are no transitive dependencies of non-key attributes.

OrderID	CustID	OrderDate
1	1	2023-01-01
2	2	2023-01-02

Table: Orders Table

CustID	CName	CAddress
1	John Doe	123 Elm St.
2	Jane Smith	456 Oak St.

Table: Customers Table

Note: CustID is the abbreviation for CustomerID, CName for CustomerName, and CAddress for CustomerAddress.

Third Normal Form (3NF) Explanation

- ▶ The 'Orders' and 'Customers' tables are in 3NF because:
 - ▶ They meet all the requirements of 2NF.
 - ▶ No non-key attribute depends on another non-key attribute.
 - ▶ This eliminates transitive dependencies, ensuring that non-key attributes are only dependent on primary keys.
- ▶ 3NF tables are more efficient for updates and maintain data integrity.

Boyce-Codd Normal Form (BCNF) Explanation

Table: Orders Table in BCNF

OrderID (PK)	OrderDetails
1	Details for Order 1
2	Details for Order 2

Table: Customers Table in BCNF

CustomerID (PK)	Name	Address
C001	John Doe	123 Apple St.
C002	Jane Smith	456 Berry Ave.

- ▶ Both tables satisfy 3NF conditions.
- ▶ In BCNF, every determinant (a column that can determine other columns) is a candidate key.
- ▶ BCNF handles anomalies that 3NF does not, making it useful for complex schemas with overlapping candidate keys.

Why Design a Database Before Implementation?

Foundation for Efficiency

Designing a database before implementation is akin to architectural planning before building a house. It ensures that:

- ▶ All necessary data relationships and structures are thoughtfully laid out.
- ▶ The database can scale efficiently without significant rework.
- ▶ Data integrity and security are maintained from the start.

Is Data Redundancy Desired in a Relational Database?

Minimizing Data Redundancy

In relational databases, data redundancy is generally undesired because:

- ▶ It can lead to inconsistencies and inaccuracies.
- ▶ It consumes unnecessary storage space.
- ▶ It complicates data management and increases maintenance workload.

The goal is to design the database to ensure data is stored only once and referenced elsewhere as needed.

Is a Primary Key Required in Every Table?

The Role of Primary Keys

Yes, a primary key is required in each table because:

- ▶ It uniquely identifies each record in the table, which is essential for data integrity.
- ▶ It enables efficient data retrieval and relationships between tables.
- ▶ Without a primary key, it becomes difficult to ensure that each row represents a unique piece of information.

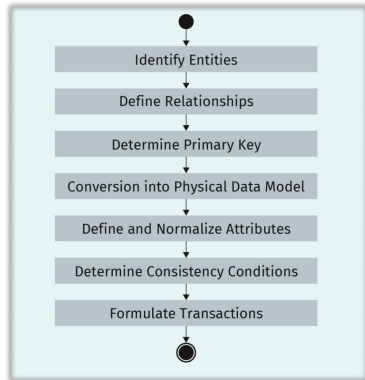
UNIT 4

CREATING RELATIONAL DATABASES

Study goals

- ▶ Design a database through standard processes.
- ▶ Convert a conceptual design model into a physical model.
- ▶ Use SQL to define database tables.

GRAPHICAL NOTATION OF ENTITIES AND ATTRIBUTES IN ER MODELS



Source of the graphic: Steiner, 2014, p. 83.

Introduction to Unit 4

- ▶ From Conceptual Models to Physical Databases
- ▶ Key Design Activities in Database Creation
- ▶ Relationship Mapping: Theory to Practice
- ▶ Introduction to SQL and Table Generation

Overview of Database Design Process

- ▶ **Requirements Analysis:** Gathering data requirements from stakeholders.
- ▶ **Conceptual Design:** Creating an ER diagram or other high-level data models.
- ▶ **Logical Design:** Translating the conceptual model into a logical model for a specific DBMS.
- ▶ **Schema Refinement:** Normalizing and optimizing to remove redundancy and improve performance.
- ▶ **Physical Design:** Deciding on storage details, indexing strategies.
- ▶ **Implementation:** Using DDL to create the database schema.
- ▶ **Maintenance:** Making ongoing adjustments and improvements.

Iterations and Feedback Loops in Database Design

- ▶ **Iterative Nature:** Database design often requires revisiting and refining earlier stages based on new feedback.
- ▶ **Stakeholder Communication:** Regularly checking in with stakeholders to ensure the database aligns with their needs.
- ▶ **Testing and Validation:** Confirming that the database design supports all required data operations and maintains data integrity.

From Requirements to Data Models

- ▶ **Translating Business Rules:** Converting business processes into data structures within the model.
- ▶ **Identifying Key Components:** Pinpointing crucial entities, relationships, and constraints in the data model.
- ▶ **Supporting Data Operations:** Ensuring the model can handle all necessary operations, including CRUD and complex transactions.
- ▶ **Prototyping and Validation:** Using prototypes and conceptual models to confirm requirements and design decisions with end-users.

What Constitutes an Entity?

- ▶ **Definition:** An entity represents a real-world object or concept with an independent existence in the database context.
- ▶ **Characteristics:** Entities have attributes that give them specific properties and identity.
- ▶ **Identification:** Entities are uniquely identifiable by a primary key within the database.
- ▶ **Examples:** In a university database, entities could be Students, Professors, Courses, etc.

Technical vs. Business Entities

- ▶ **Business Entities:** Reflect the organization's operational elements, like Customers, Orders, and Products.
- ▶ **Technical Entities:** Used for database functionalities and may not have a direct business context, like UserAccounts or LogRecords.
- ▶ **Importance of Distinction:** Recognizing the difference is crucial for effective database design and aligning with business objectives.

Refinement of Entity Characteristics

- ▶ **Process:** Refining entity characteristics involves detailing out the attributes and relationships of the entity.
- ▶ **Normalization:** This step often includes normalization to ensure efficient data structure and avoid redundancy.
- ▶ **Evolution:** Entities may evolve as the understanding of the system deepens or as business requirements change.
- ▶ **Collaboration:** Refinement is usually a collaborative process involving feedback from stakeholders.

Refinement of Entity Characteristics

- ▶ **Process:** Refining entity characteristics involves detailing out the attributes and relationships of the entity.
- ▶ **Normalization:** This step often includes normalization to ensure efficient data structure and avoid redundancy.
- ▶ **Evolution:** Entities may evolve as the understanding of the system deepens or as business requirements change.
- ▶ **Collaboration:** Refinement is usually a collaborative process involving feedback from stakeholders.

Understanding Cardinalities

- ▶ **Defining Quantities:** Cardinalities define the numerical aspects of relationships between entities (one-to-one, one-to-many, many-to-many).
- ▶ **Constraints and Rules:** They help establish constraints and rules that govern the associations between data sets.
- ▶ **Design Decisions:** Cardinalities affect database normalization and table structures, influencing design decisions.
- ▶ **Optimization:** Correct cardinalities are essential for database performance and optimization.

Real-world vs. Data Model Relationships

- ▶ **Mimicking Reality:** Data model relationships strive to accurately represent the complex relationships found in the real world.
- ▶ **Simplification:** However, they may simplify or abstract certain aspects to fit the constraints and purposes of the database.
- ▶ **Business Logic:** Relationships in data models must enforce the business logic of the application, sometimes going beyond the direct real-world analogs.
- ▶ **Flexibility and Scalability:** They must be flexible enough to accommodate changes in business processes and scalable to handle growth.

What are Primary Keys?

- ▶ **Unique Identifiers:** Primary keys are unique identifiers for records within a database table.
- ▶ **Uniqueness Guarantee:** They guarantee that each record can be uniquely distinguished from all others.
- ▶ **Indexing:** Primary keys are typically indexed, improving search performance within the database.
- ▶ **Referential Integrity:** They are crucial for establishing relationships between tables, maintaining referential integrity.

Simple vs. Compound Primary Keys

- ▶ **Simple Primary Keys:** A single attribute that uniquely identifies a record. Ideal for straightforward scenarios.
- ▶ **Compound Primary Keys:** A combination of two or more attributes that together create a unique identifier. Used when no single attribute is unique on its own.
- ▶ **Design Considerations:** The choice between simple and compound keys affects database normalization, schema complexity, and query simplicity.
- ▶ **Performance Impacts:** Compound keys can impact database performance and should be used judiciously.

Use of Artificial/Surrogate Keys

- ▶ **Artificial Keys:** These are system-generated keys, often a sequence or auto-increment number, that have no inherent meaning outside of their role as an identifier.
- ▶ **Surrogate Keys:** A type of artificial key used when natural keys (real-world meaning) are not suitable, often due to complexity or lack of uniqueness.
- ▶ **Advantages:** They can simplify database design, especially when natural keys are changeable or cumbersome.
- ▶ **Considerations:** While they add simplicity, they may also introduce an additional layer of abstraction, requiring careful consideration in system design.

Translating Conceptual to Physical Models

- ▶ **Detailing the Design:** Transitioning from high-level conceptual models to detailed physical models suitable for implementation.
- ▶ **Mapping Entities to Tables:** Converting entities into tables, attributes into columns, and relationships into foreign keys.
- ▶ **Defining Data Types:** Assigning appropriate SQL data types to each attribute based on the nature of the data it will hold.
- ▶ **Performance Considerations:** Indexing strategies, partitioning, and physical storage decisions are made to optimize performance and storage efficiency.

The Role of Relational DBMS in Physical Modeling

- ▶ **DBMS-Specific Features:** Understanding the features and limitations of the target DBMS to effectively design the physical model.
- ▶ **SQL Script Generation:** Automated tools can generate SQL scripts for table creation, which are then fine-tuned by database developers.
- ▶ **Physical Data Integrity:** Defining constraints and triggers to maintain data integrity in line with business rules.
- ▶ **Database Administration:** Planning for backup, recovery, and security policies within the physical model.

Case Study: UML to Relational Model Conversion

- ▶ **Case Study Overview:** Presenting a real-world example of converting a UML class diagram into a relational database schema.
- ▶ **Mapping Strategies:** Discussing various approaches to map inheritance, associations, and aggregation in UML to relational tables.
- ▶ **Lessons Learned:** Sharing insights on common challenges and best practices observed during the conversion process.
- ▶ **Q&A:** Allowing the audience to ask questions specific to the case study, fostering a deeper understanding of the practical aspects of the conversion.

Defining Attributes for Entities

- ▶ **Identifying Attributes:** Determining the necessary information to be stored about each entity within the database.
- ▶ **Attribute Characteristics:** Establishing attribute properties including uniqueness, nullability, and default values.
- ▶ **Keys and Indexes:** Selecting primary keys to uniquely identify entity instances and indexes to enhance query performance.
- ▶ **Data Integrity Rules:** Applying constraints to ensure the accuracy and consistency of data across related entities.

Data Types and Their Significance

- ▶ **Data Type Selection:** Matching the nature of data to the appropriate SQL data types, such as INTEGER, VARCHAR, DATE, etc.
- ▶ **Data Type Implications:** Understanding how data type choices affect storage, performance, and data integrity.
- ▶ **Type Safety:** Ensuring that operations on data are type-safe to prevent errors and data corruption.
- ▶ **Advanced Data Types:** Exploring the use of specialized data types like ENUM, SET, or custom types for specific use cases.

Normalization: Theory and Practice

- ▶ **Fundamentals of Normalization:** Introducing the concept of normalization and its role in reducing redundancy and dependency.
- ▶ **Normal Forms:** Describing the various normal forms from 1NF to BCNF, including their rules and objectives.
- ▶ **Normalization in Design:** Applying normalization during database design to create a scalable and maintainable schema.
- ▶ **Denormalization Considerations:** Discussing scenarios where denormalization may be beneficial for performance optimization.

SQL Data Types Overview

- ▶ Understanding SQL Data Types
- ▶ Text, Numeric, and Date Data Types
- ▶ Binary and Boolean Data Types

SQL Data Types and Examples

DATA TYPES

Form	SQL Data Type	Examples
Character strings with always exactly n characters	CHAR(n)alternative: CHARACTER(n)	Country codes: DE, CN, FR, NL etc.: CHAR(2) ISBN 13 numbers: 9783836216999 CHAR(13) UUIDs: 21EC2020-3AEA-1069-A2DD-08002B30309DCHAR(36)(German) Postal codes: 45147CHAR(5)
Character strings with variable but limited length (max. n characters)	VARCHAR (n)alternative: CHARACTER VARYING(20)	proper names, designations, short descriptions: VARCHAR(55), VARCHAR(160),VARCHAR(500)
Very long character strings; the actual allowed size is limited by technical parameters within the DBMS.	TEXT	Texts, blog entries, detailed descriptions: TEXT

Source of the table: Course Book DLBCSDM001, p. 95

SQL Data Types and Examples

DATA TYPES

Form	SQL Data Type	Examples
Integers; 32 bits, value range from -2,147,483,648 to 2,147,483,647	INTEGER alternative: INT	Number, index, numbering: INTEGER
Decimal numbers with a maximum of n digits in total and exactly m digits after the decimal point.	DECIMAL(n,m)	Amounts of money: DECIMAL(9,2), Key figures: DECIMAL(4,2),
Floating point numbers, 32bit	REAL	Measured values: REAL

Source of the graphic: Course Book DLBCSDMD01, p. 96

SQL Data Types and Examples

DATA TYPES

Form	SQL Data Type	Examples
Calendar data accurate to the day, from year 1000 to year 9999 in YYYY-MM-DD format	DATE	Date, calendar days: 2014-06-14
Time data in the format HH:MM:SS	TIME	Time, to the second: 13:43:56
Exact determination of a time, exact to the second, often in the format YYYY-MM-DD HH:MM:SS	TIMESTAMP	Selected times: 2014-06-14 14:00:02; Automatic saving of creation or modification times

Source of the table: Course Book DLBCSDM001, pp. 96-97

SQL Data Types and Examples

DATA TYPES

Form	SQL Data Type	Examples
Binary types with variable length, but not more than n characters (n bits)	VARBINARY(n)	Store binary data such as images, audio, video, and other binary files; BINARY VARYING(50000); often also called BLOB in DBMS
Boolean Values	BOOLEAN	Boolean Values

Source of the table: Course Book DLBICSDMD01, p. 97

Ensuring Data Consistency

- ▶ **Consistency Conditions:** Rules that ensure the reliability of database transactions and maintain the validity of data.
- ▶ **Unique ID Constraints:** Mechanisms that guarantee each record can be uniquely identified within a table.
- ▶ **Data Integrity:** The accuracy and consistency of data over its lifecycle, including constraints like foreign keys to maintain referential integrity.

Defining Database Transactions

- ▶ **ACID Properties:** The set of properties—Atomicity, Consistency, Isolation, Durability—that ensure reliable processing of database transactions.
- ▶ **CRUD Operations:** The four basic functions of persistent storage: Create, Read, Update, Delete.
- ▶ **Transaction Processing:** Ensuring that all database transactions are processed reliably and that the database remains in a consistent state.

Conceptual vs. Physical Data Models

- ▶ **Conceptual Data Models:** High-level models that define entities, relationships, and constraints without considering DBMS specifics.
- ▶ **Physical Data Models:** Detailed models that include specific tables, columns, data types, and database constraints tailored to a particular DBMS.
- ▶ **Model Bridging:** The process of transforming a conceptual model into a physical model, considering performance and storage requirements.

Creating Tables: The SQL Way

- ▶ **CREATE TABLE Command:** SQL syntax used to define a new table structure in the database.
- ▶ **Column Definitions and Data Types:** Specific columns and SQL data types that describe the shape and kind of data each table will store.
- ▶ **Primary and Foreign Keys:** The use of keys to enforce data integrity and define relationships between tables.

Maintaining Referential Integrity

- ▶ **Referential Integrity:** A database concept that ensures relationships between tables remain consistent.
- ▶ **CASCADE, SET NULL, RESTRICT Options:** Different strategies for managing updates and deletions in related tables.
- ▶ **Data Integrity Design:** Creating a database schema that robustly enforces data integrity through appropriate constraints.

Database Integrity with CHECK

- ▶ **CHECK Constraint:** A condition that the data must meet before being accepted into the database table.
- ▶ **Defining Data Entry Conditions:** Using CHECK constraints to validate data upon insertion or update.
- ▶ **Practical Examples:** Scenarios where CHECK constraints prevent invalid data from corrupting the database.

Recap of the Database Creation Process

- ▶ The database creation process is a multi-stage endeavor that begins with understanding user requirements and culminates in the deployment of a fully functional database system.
- ▶ Key stages include requirements gathering, conceptual modeling, logical and physical design, implementation, and maintenance.
- ▶ Effective database creation requires meticulous planning, iterative design, and ongoing collaboration with stakeholders to ensure the database meets all functional requirements and performance criteria.

The Critical Role of Accurate Modeling

- ▶ Accurate modeling is essential for translating real-world scenarios into a structured database schema that accurately represents data relationships and constraints.
- ▶ It involves using standard modeling techniques like Entity-Relationship diagrams or UML to visualize and test the database structure before implementation.
- ▶ Proper modeling can prevent future issues such as data redundancy, integrity problems, and scalability concerns, making it a critical step in the database design process.

Best Practices for SQL Table Definition

- ▶ Defining SQL tables is more than just a technical exercise; it requires a deep understanding of the data, its interrelationships, and how it will be accessed and manipulated.
- ▶ Best practices include normalizing data to reduce redundancy, carefully selecting data types to balance performance with precision, and enforcing data integrity with appropriate key constraints.
- ▶ Regularly reviewing and updating table definitions as application requirements evolve is crucial to maintain optimal performance and data integrity.

SESSION 2

TRANSFER TASK

TRANSFER TASKS

Given the following branch table:

Branch ID	Branch address	Branch phone number	Department ID	Department name	Employee ID	Employee Name
-----------	----------------	---------------------	---------------	-----------------	-------------	---------------

1. Normalize the above table by putting it in the third normal form.
2. Draw the ERD of the normalized tables.
3. Write the SQL code that creates the normalized tables.