

DBMS

Week 02

Database Users and Administrators

Database Users and User Interfaces

- Naive users are unsophisticated users who interact with predefined user interfaces such as web or mobile applications.
- Application programmers are professionals who write application programs, and they have various tools to develop user interfaces.
- Sophisticated users interact with the system using database query languages or data analysis software.
- Analysts fall into this category, using queries to explore data in the database.

Database Administrator (DBA)

- Schema definition: Creating the original database schema using Data Definition Language (DDL) statements.
- Storage structure and access-method definition: Specifying parameters related to physical data organization and index creation.
- Schema and physical-organization modification: Making changes to reflect evolving organizational needs or improve performance.
- Granting authorization for data access: Regulating user access to parts of the database by granting different types of authorization.
- Routine maintenance: Tasks like database backups, disk space management, and monitoring to ensure performance.

History of Database Systems

1950s and early 1960s

- Automation of data processing tasks began before the advent of computers.
- Punched cards and mechanical systems were used for data processing.
- Magnetic tapes were developed for data storage and automated tasks like payroll.

Late 1960s and early 1970s

- Hard disks enabled direct access to data, freeing it from sequential processing.
- Network and hierarchical data models emerged.
- Edgar Codd's relational model and non-procedural querying techniques were introduced, leading to relational databases.

Late 1970s and 1980s

- System R project at IBM Research improved the efficiency of relational databases.
- Commercial relational database systems like IBM DB2, Oracle, Ingres, and DEC Rdb advanced query processing techniques.
- Relational databases became competitive in terms of performance.

1990s

- Decision support and querying re-emerged as major application areas.
- Parallel database products were introduced, and object-relational support was added.
- Explosive growth of the World Wide Web increased database deployment, requiring high performance and reliability.

2000s

- Evolution of data types, including semi-structured data, XML, and JSON.
- Emergence of open-source databases like PostgreSQL and MySQL.
- Growth of social network platforms led to the development of graph databases.
- Focus on data analytics and data mining, leading to column-store databases.

2010s

- Rise of NoSQL databases for scalability and availability.
- Evolution of NoSQL systems to support stricter consistency models.
- Enterprises outsourcing data storage and applications to cloud services.
- Privacy regulations, cybersecurity challenges, and data ownership concerns gained prominence.

Class activity

- Database-management system (DBMS)
- Database-system applications
- Online transaction processing
- Data analytics
- File-processing systems
- Data inconsistency
- Consistency constraints
- Data abstraction
 - Physical level
 - Logical level
 - View level
- Instance
- Schema
 - Physical schema
 - Logical schema
 - Subschema
- Physical data independence
- Data models
 - Entity-relationship model
 - Relational data model
 - Semi-structured data model
 - Object-based data model
- Database languages
 - Data-definition language
 - Data-manipulation language
 - Procedural DML
 - Declarative DML
 - nonprocedural DML
 - Query language
- Data-definition language
 - Domain Constraints
 - Referential Integrity
 - Authorization
 - Read authorization
 - Insert authorization
 - Update authorization
 - Delete authorization
- Metadata
- Application program
- Database design
 - Conceptual design
 - Normalization
 - Specification of functional requirements
 - Physical-design phase
- Database Engine
 - Storage manager
 - Authorization and integrity manager
 - Transaction manager
 - File manager
 - Buffer manager
 - Data files
 - Data dictionary
 - Indices
 - Query processor
 - DDL interpreter
 - DML compiler
 - Query optimization
- Query evaluation engine
- Transactions
 - Atomicity
 - Consistency
 - Durability
 - Recovery manager
 - Failure recovery
 - Concurrency-control manager
- Database Architecture
 - Centralized
 - Parallel
 - Distributed
- Database Application Architecture
 - Two-tier
 - Three-tier
 - Application server
- Database administrator (DBA)

Tools

Commercial Database Systems:

- IBM DB2: A relational database management system (RDBMS) developed by IBM, known for its scalability and robustness.
- Oracle: Oracle Database is a powerful RDBMS known for its performance, security, and advanced features. It is widely used in enterprise-level applications.
- Microsoft SQL Server: Microsoft's RDBMS offers excellent integration with other Microsoft products and services and is commonly used in Windows-based environments.
- IBM Informix: Another RDBMS by IBM, Informix offers features like high availability and data replication.
- SAP Adaptive Server Enterprise (ASE): Formerly known as Sybase ASE, it's a relational database management system designed for high-performance transaction processing.
- SAP HANA: An in-memory database management system developed by SAP, HANA is known for its speed and real-time data processing capabilities.

Free/Public Domain Database Systems:

- MySQL: An open-source relational database management system widely used in web applications. It's known for its speed and ease of use.
- PostgreSQL: An open-source object-relational database system that is highly extensible and offers advanced features, making it suitable for complex applications.
- SQLite: A self-contained, serverless, and zero-configuration database engine. It's often embedded in applications due to its lightweight nature.

Part two

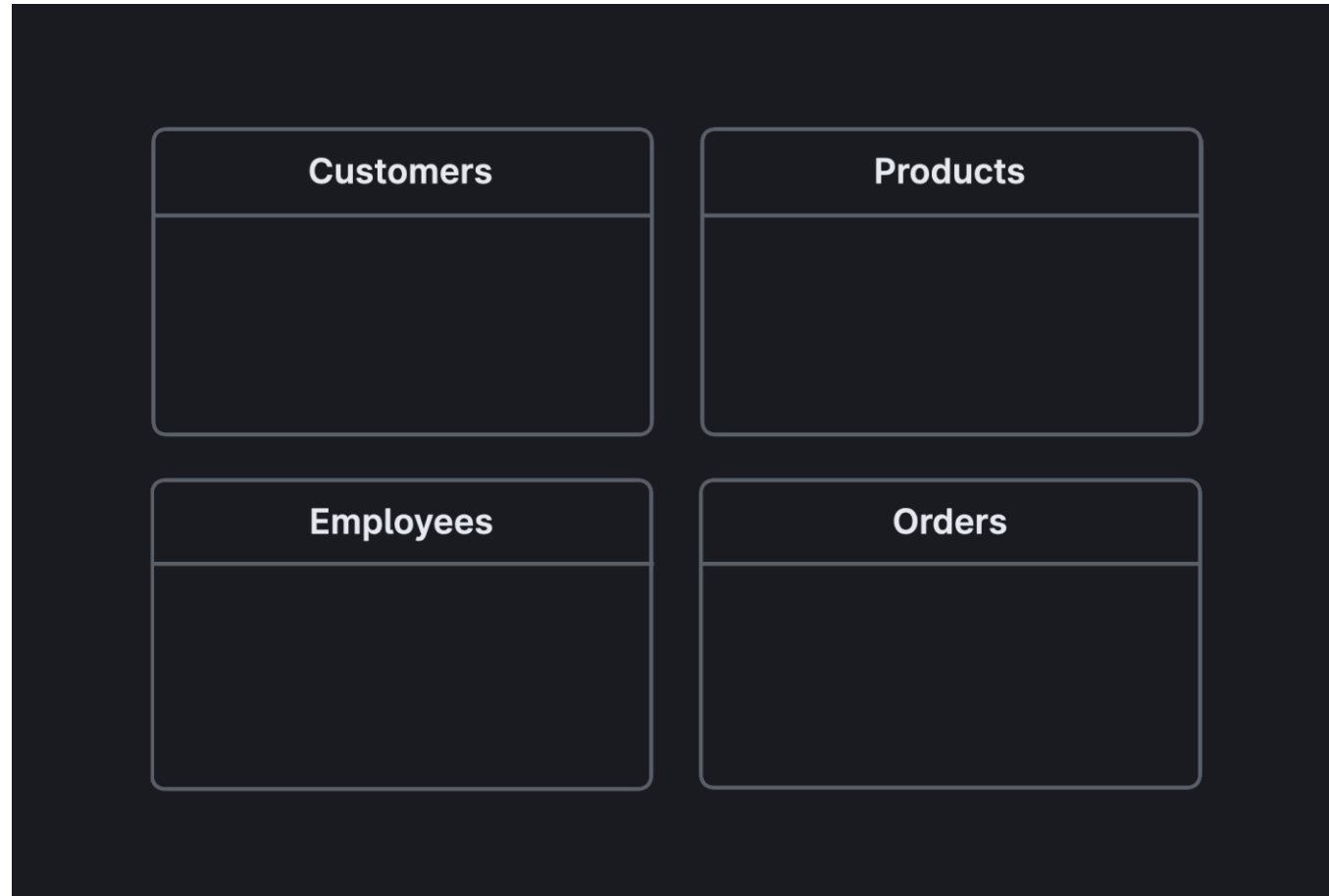
Relational Databases

Chapter 2 Introduction to the Relational Model

The relational model remains the primary data model for commercial data-processing applications. It attained its primary position because of its simplicity, which eases the job of the programmer, compared to earlier data models such as the network model or the hierarchical model. It has retained this position by incorporating various new features and capabilities over its half-century of existence. Among those additions are object-relational features such as complex data types and stored procedures, support for XML data, and various tools to support semi-structured data. The relational model's independence from any specific underlying low-level data structures has allowed it to persist despite the advent of new approaches to data storage, including modern column stores that are designed for large-scale data mining.

Structure of Relational Database

Define Entities



Define Properties

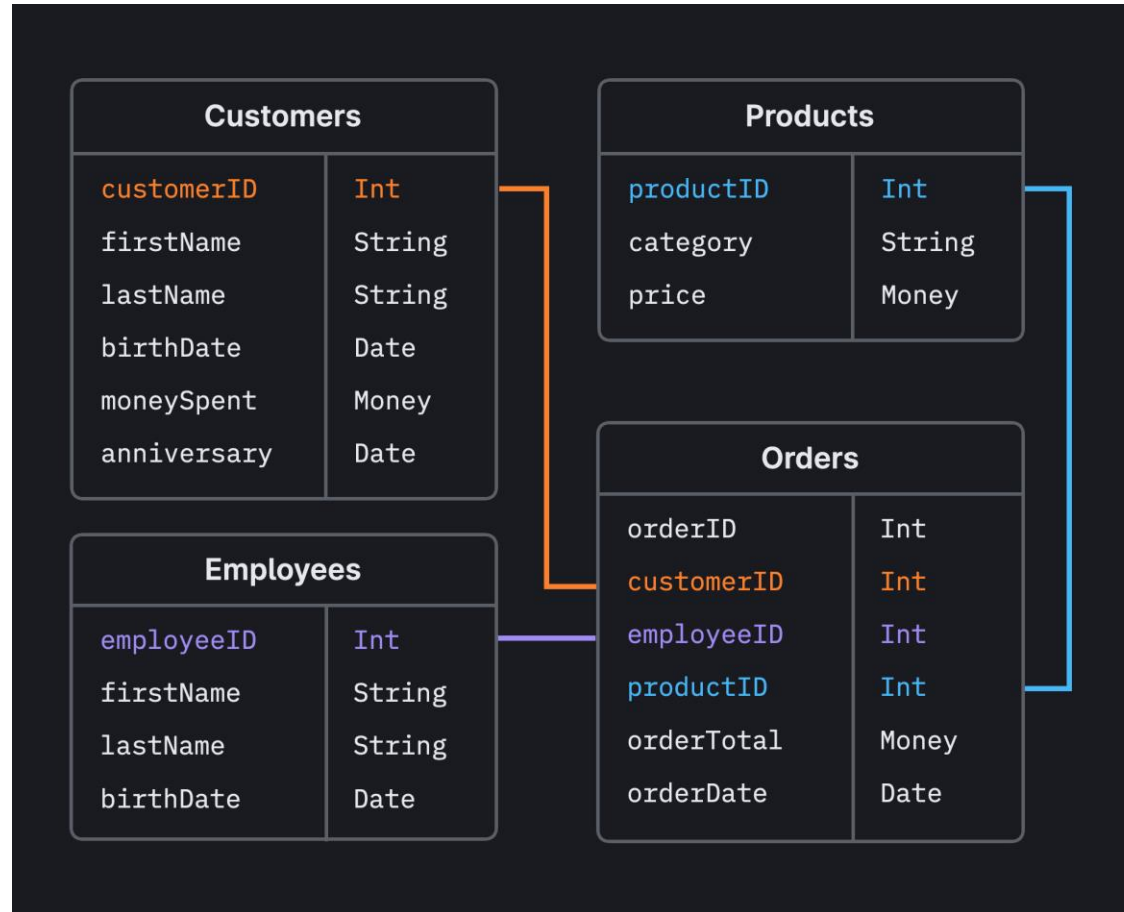
Customers	
customerID	Int
firstName	String
lastName	String
birthDate	Date
moneySpent	Money
anniversary	Date

Products	
productID	Int
category	String
price	Money

Employees	
employeeID	Int
firstName	String
lastName	String
birthDate	Date

Orders	
orderID	Int
customerID	Int
employeeID	Int
productID	Int
orderTotal	Money
orderDate	Date

Define Relationships



Tables and Relations

- A relational database is composed of tables, also known as relations.
- Each table is given a unique name.
- Each row in a table represents a relationship among a set of values and is called a tuple.
- The term "attribute" refers to a column in a table.

Attributes and Domains

- Each attribute has a specific domain, which is the set of permitted values for that attribute.
- The domain of an attribute is atomic, meaning it consists of indivisible values.
- Null values can be used to signify unknown or missing values.

Tables' Instances

- An instance of a relation is a specific set of rows (tuples) for that relation.
- Rows in a table represent specific instances of relationships among attribute values.

Data Representation

- The relational model uses tables to represent relationships between entities.
- Each table corresponds to a mathematical concept called a relation.

Attribute Identification

- Each attribute in a table is identified by its name.
- Attributes contain data values that belong to the specified domain.

Sorting and Ordering

- The order of tuples within a relation does not affect its meaning.
- Relations are sets of tuples, so their order is not significant.

Domain Atomicity

- Domains of attributes are atomic, meaning they consist of indivisible values.
- The atomicity property simplifies data handling and processing.

Null Values

- Null values represent unknown or missing information in attributes.
- Nulls should be used cautiously as they can introduce complexities in data operations.

Benefits of the Relational Model

- The relational model's structured nature has practical advantages for storage and processing efficiency.
- Its well-defined structure is suitable for static applications with consistent data requirements.

Database Schema

Schema and Instance

- A relational database has two important components: schema and instance.
- The schema represents the logical design of the database, while the instance is a snapshot of data at a specific time.

Relation Correspondence

- In programming, a relation corresponds to a variable, and a relation schema corresponds to a type definition.
- A relation schema includes attributes and their domains (permitted values), while the instance corresponds to values of the variables.

Attribute Domains

- Each attribute has a domain, which is the set of permitted values for that attribute.
- The specific domain definitions are discussed later when exploring the SQL language.

Relation Changes

- A relation instance represents data at a specific time and may change as the database is updated.
- A relation's schema, however, generally remains constant.

Common Attributes and Relations

- Relations can share common attributes to relate tuples across different relations.
- Using common attributes allows data to be connected and queried based on shared values.

Relation Names

- The same name may be used to refer to both the schema and the instance of a relation.
- When needed, you can explicitly differentiate between the schema and the instance.

Keys

Uniqueness Constraint

- Tuples within a relation need to be uniquely distinguishable using their attribute values.
- No two tuples in a relation should have the same values for all attributes.

Superkeys

- A superkey is a set of one or more attributes that collectively uniquely identify a tuple in a relation.
- For example, the ID attribute in the instructor relation serves as a superkey.
- A superkey may include extraneous attributes.

Candidate Keys

- A candidate key is a minimal superkey, meaning no proper subset of the key can uniquely identify tuples.
- Different sets of attributes can be candidate keys for a relation.
- Primary keys are chosen from the set of candidate keys.

Primary Keys

- Primary keys are chosen by the database designer and serve as the principal means of identifying tuples within a relation.
- Primary keys should be carefully chosen to ensure uniqueness and stability.
- They should consist of attributes with values that rarely or never change.

Key Attributes in Schema

- Primary key attributes are usually listed before other attributes in the relation schema.
- They are also underlined to distinguish them.

classroom(building, room_number, capacity)
department(dept_name, building, budget)
course(course_id, title, dept_name, credits)
instructor(ID, name, dept_name, salary)
section(course_id, sec_id, semester, year, building, room_number, time_slot_id)
teaches(ID, course_id, sec_id, semester, year)
student(ID, name, dept_name, tot_cred)
takes(ID, course_id, sec_id, semester, year, grade)
advisor(s_ID, i_ID)
time_slot(time_slot_id, day, start_time, end_time)
prereq(course_id, prereq_id)

Foreign-Key Constraints

- Foreign-key constraints ensure that values in a referencing relation attribute must also exist in the referenced relation's primary key.
- Attributes in the referencing relation are foreign keys referencing the primary key of another relation.
- These constraints enforce data integrity and relationships between tables.

Referential Integrity Constraints

- Referential integrity constraints generalize foreign-key constraints.
- They require that values in specified attributes of the referencing relation exist in the referenced relation, whether the referenced attribute is a primary key.

Schema Diagrams

Database Schema Diagrams

- Database schema diagrams are graphical representations of the database structure, including relations, attributes, primary keys, and foreign-key relationships.
- Each relation is depicted as a box, with the relation's name displayed at the top in blue, and attributes listed inside the box.

Primary-Key Attributes

- Primary-key attributes are indicated in the schema diagram by being underlined within the respective relation box.
- Underlining helps visually distinguish primary keys from other attributes.

Foreign-Key Constraints

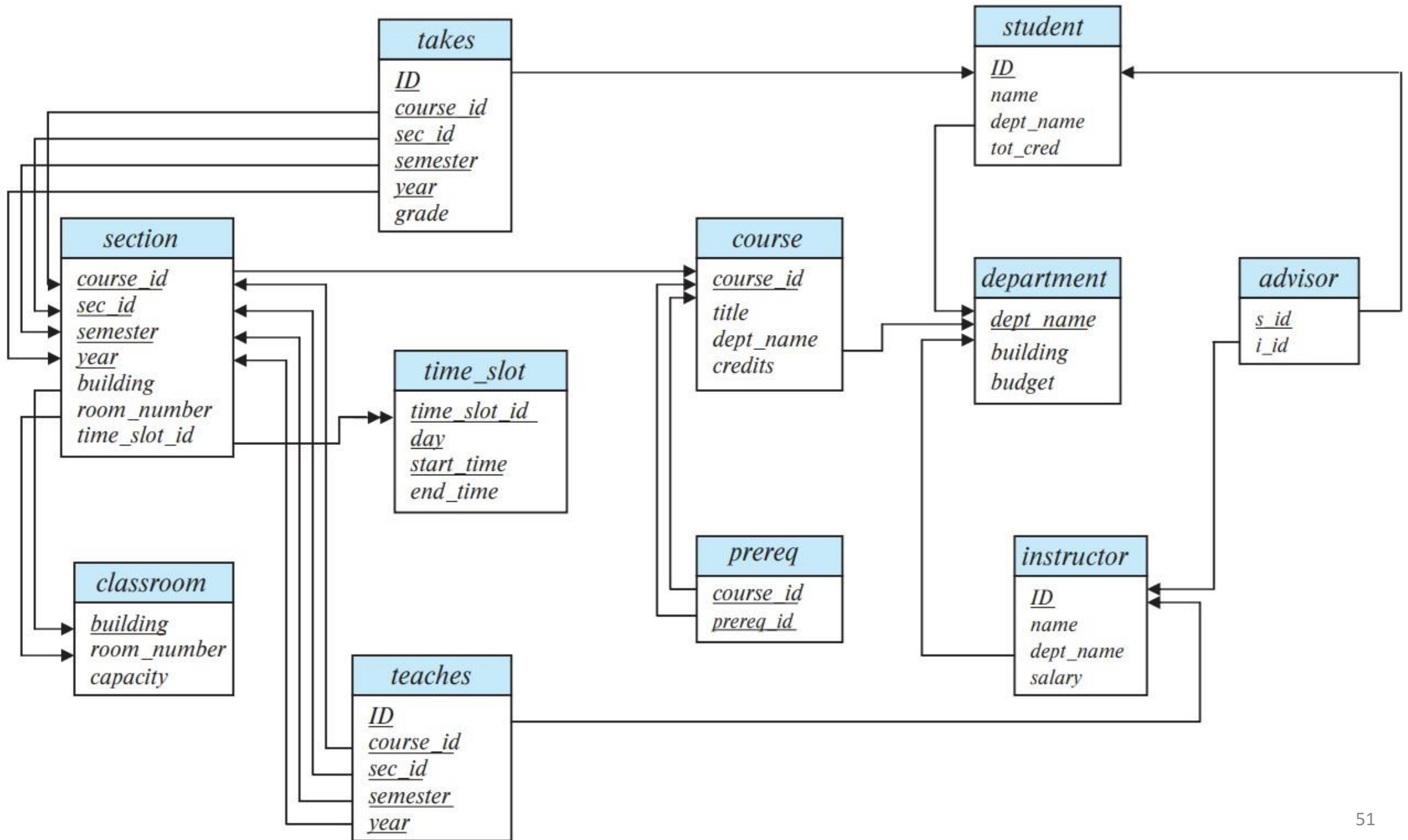
- Foreign-key constraints are depicted using arrows that connect the foreign-key attributes of a referencing relation to the primary key of the referenced relation.
- The arrow direction indicates the flow of the reference from one relation to another.
- In the context of the diagram, foreign keys visually represent relationships between tables.

Referential Integrity Constraints

- If there's a referential integrity constraint that's not necessarily a foreign-key constraint (i.e., the referenced attribute doesn't have to be a primary key), it is represented by a two-headed arrow.

Representation in Design Tools

- Many database management systems provide tools with graphical interfaces for creating schema diagrams.
- These tools allow users to visually design and model the database schema, helping with database design and understanding relationships.



Difference from Entity-Relationship Diagrams (ERDs)

- The passage notes that while there might be visual similarities, schema diagrams and entity-relationship diagrams (ERDs) are distinct notations.
- ERDs, covered in Chapter 6, provide a different way to represent relationships, attributes, and entities in a more abstract manner.

Relational Query Languages

Query Languages Overview

- A query language is used by users to request information from a database.
- Query languages are typically higher-level than standard programming languages, focusing on data retrieval rather than general computation.

Categories of Query Languages


- Imperative Query Language: In an imperative query language, the user specifies a sequence of operations to be performed on the database to compute the desired result. State variables are used to update the computation.
- Functional Query Language: In a functional query language, the computation is expressed as the evaluation of functions that operate on database data or results from other functions. Functions are side-effect free and don't modify the program state.
- Declarative Query Language: In a declarative query language, the user describes the desired information without specifying the steps or functions required to obtain it. The database system determines how to retrieve the information.

Imperative Query Language



```
total_gpa = 0
count = 0
for student in students:
    if student.department == "Computer Science":
        total_gpa += student.gpa
        count += 1
average_gpa = total_gpa / count
```


Functional Query Language



```
students  
|> filter(department = "Computer Science")  
|> map(gpa)  
|> average
```

Declarative Query Language



```
SELECT AVG(gpa)  
FROM students  
WHERE department = 'Computer Science';
```

Examples of Query Languages

- Relational Algebra: The relational algebra, a functional query language, is a theoretical foundation for the SQL query language. It involves operations like selection, projection, union, and join.
- Tuple and Domain Relational Calculus: These declarative query languages use logical expressions to describe desired information.
- SQL (Structured Query Language): SQL, widely used in practice, includes elements from imperative, functional, and declarative approaches. It allows users to specify queries, updates, and other operations on relational databases.

Characteristics of Query Languages

- "Pure" query languages like the relational algebra and relational calculus are formal and concise, but they lack some of the user-friendly features found in commercial query languages like SQL.
- Commercial query languages often include syntactic features that make queries more intuitive and user-friendly.

The Relational Algebra

Unary and Binary Operations

- The relational algebra consists of operations that take one or two relations as input and produce a new relation as the output.
- Unary operations like select, project, and rename operate on a single relation.
- Binary operations like union, Cartesian product, and set difference operate on pairs of relations.

Select Operation

- The select operation is used to retrieve tuples that satisfy a given predicate.
- The predicate is expressed as a subscript to σ (sigma), and the argument relation is in parentheses.
- Operations that are used:
 - $=$ (equal)
 - \neq (not equal)
 - \leq (less than equal)
 - \geq (greater than equal)
 - $<$ (less than)
 - $>$ (greater than)
 - \wedge (and)
 - \vee (or)
 - \neg (not)
- $\sigma_{\text{dept name} = \text{"Physics"}} (\text{instructor})$
- $\sigma_{\text{salary} < 9000} (\text{instructor})$
- $\sigma_{\text{dept name} = \text{"Physics"} \wedge \text{salary} > 9000} (\text{instructor})$
- $\sigma_{\text{dept name} = \text{building}} (\text{instructor})$

Project Operation

- The project operation returns a relation with specified attributes from the input relation.
- Attributes to be included in the result are listed as subscripts to Π , and the argument relation is in parentheses.
- $\Pi_{ID, name, salary}(\text{instructor})$
- $\Pi_{ID, name, salary/12}(\text{instructor})$

Composition of Operations

- The result of a relational operation is itself a relation, allowing operations to be composed into expressions.
- Expressions can be built using multiple relational-algebra operations, like composing arithmetic operations.
- $\Pi_{ID, name, salary} (\sigma_{salary < 9000}(\text{instructor}))$

Cartesian-Product Operation (\times)

- The Cartesian product combines information from two relations, forming pairs of tuples.
- Resulting tuples combine attributes from both relations, with unique attribute naming conventions to avoid ambiguity.
- $r = \text{instructor} \times \text{teaches}$ is:
 - (instructor.ID, instructor.name, instructor.dept name, instructor.salary, teaches.ID, teaches.course id, teaches.sec id, teaches.semester, teaches.year)
- $r = \text{instructor} \times \text{teaches}$ is:
 - (instructor.ID, name, dept name, salary, teaches.ID, course id, sec id, semester, year)

Join Operation (\bowtie)

- The join operation combines tuples from two relations based on a predicate, typically involving common attributes.
- Natural join simplifies the predicate by using attributes present in both relations.
- Joining can involve attribute renaming using the rename operation (ρ).
- $r \bowtie_{\theta} s = \sigma_{\theta} (r \times s)$
 - $\sigma_{\text{instructor.ID} = \text{teaches.ID}} (\text{instructor} \times \text{teacher})$
 - $\text{instructor} \bowtie_{\text{instructor.ID} = \text{teaches.ID}} \text{teacher}$

Set Operations (Union, Intersection, Set Difference)

- Union (\cup) combines tuples from two relations, eliminating duplicates.
- Intersection (\cap) finds tuples common to both relations.
- Set difference ($-$) returns tuples present in one relation but not the other.
- $\Pi_{ID, name} (\sigma_{\text{semester} = \text{"Fall"} \wedge \text{year} = 2017} (\text{section})) \cup \Pi_{ID, name} (\sigma_{\text{semester} = \text{"Spring"} \wedge \text{year} = 2018} (\text{section}))$

Assignment Operation (\leftarrow)

- The assignment operation allows storing the result of a relational-algebra expression in a temporary relation variable.
- Resulting variables can be used in subsequent expressions.
- $F17C \leftarrow \Pi_{ID, name} (\sigma_{\text{semester} = \text{"Fall"} \wedge \text{year} = 2017} (\text{section}))$

Rename Operation (ρ)

- The rename operation assigns a new name to a relation or attributes in the result of an expression.
- Can be used to distinguish between multiple instances of the same relation within an expression.

Class activity

- Table
- Relation
- Tuple
- Attribute
- Relation instance
- Domain
- Atomic domain
- Null value
- Database schema
- Database instance
- Relation schema
- Keys
 - Superkey
 - Candidate key
 - Primary key
 - Primary key constraints
- Foreign-key constraint
 - Referencing relation
 - Referenced relation
- Referential integrity constraint
- Schema diagram
- Query language types
 - Imperative
 - Functional
 - Declarative
- Relational algebra
- Relational-algebra expression
- Relational-algebra operations
 - Select σ
 - Project Π
 - Cartesian product \times
 - Join \bowtie
 - Union \cup
 - Set difference $-$
 - Set intersection \cap
 - Assignment \leftarrow
 - Rename ρ

Tools

Commercial Database Systems:

- IBM DB2: A relational database management system (RDBMS) developed by IBM, known for its scalability and robustness.
- Oracle: Oracle Database is a powerful RDBMS known for its performance, security, and advanced features. It is widely used in enterprise-level applications.
- Microsoft SQL Server: Microsoft's RDBMS offers excellent integration with other Microsoft products and services and is commonly used in Windows-based environments.
- IBM Informix: Another RDBMS by IBM, Informix offers features like high availability and data replication.
- SAP Adaptive Server Enterprise (ASE): Formerly known as Sybase ASE, it's a relational database management system designed for high-performance transaction processing.
- SAP HANA: An in-memory database management system developed by SAP, HANA is known for its speed and real-time data processing capabilities.

Free/Public Domain Database Systems:

- MySQL: An open-source relational database management system widely used in web applications. It's known for its speed and ease of use.
- PostgreSQL: An open-source object-relational database system that is highly extensible and offers advanced features, making it suitable for complex applications.
- SQLite: A self-contained, serverless, and zero-configuration database engine. It's often embedded in applications due to its lightweight nature.