DBMS

Week 08

Modeling Temporal Data

Temporal Data and Time Intervals

- Temporal data refers to data that have an associated time interval during which they are valid.
- Time intervals can have associated start and end dates, with special values like "null" or a distant future date (e.g., 9999-12-31) indicating ongoing validity.

Challenges in Modeling Temporal Data

- Modeling temporal data is challenging due to various factors, including the need to track time-varying attribute values, valid time intervals for entities, and valid times for relationships.
- Adding temporal information directly to an Entity-Relationship (E-R) diagram can complicate the design and comprehension.

Approaches to Temporal Database Design

- Database designers typically start by designing the entire database without considering temporal aspects.
- Temporal information is then added to specific relations that require it by introducing attributes for start and end times.

Temporal Functional Dependencies

- Temporal functional dependencies (TFDs) replace traditional functional dependencies in temporal databases.
- A TFD asserts that an attribute has only one value within a specific time interval.

Primary Keys and Foreign Keys in Temporal Relations

- Primary keys in temporal relations need to ensure that tuples with the same key values do not have overlapping time intervals.
- Temporal foreign keys must consider the valid time intervals of referenced tuples.

Temporal History Relations

- Some designers create history relations to store changes over time while keeping the original relations unchanged.

SQL:2011 Support for Temporal Data

- SQL:2011 introduced support for temporal data, allowing attributes to specify valid time intervals and supporting temporal primary keys and foreign keys.
- Relational algebra operations can be extended to handle temporal relations.

Relational Algebra Operations for Temporal Data

- Selection and projection operations on temporal relations maintain the valid time intervals of tuples.
- Temporal join operations create results with valid times based on the intersection of input tuples' valid times.

Challenges in Database Systems

- While SQL:2011 introduced temporal features, support for them varies among database systems.
- Some databases use workarounds and exclude constraints to enforce temporal constraints.

Class activity

- Decomposition
  - Lossy decompositions
  - Lossless decompositions
- Normalization
- Functional dependencies
- Legal instance
- Superkey
- R satisfies F
- Functional dependency
  - Holds
  - Trivial
- Closure of a set of functional dependencies
- Dependency preserving
- Third normal form
- Transitive dependencies
- Logically implied
- Axioms
- Armstrong's axioms
- Sound
- Complete
- Functionally determined
- Extraneous attributes
- Canonical cover
- Restriction of F to Ri
- Dependency-preserving decomposition
- Boyce–Codd normal form (BCNF)
- BCNF decomposition algorithm
- Third normal form (3NF)
- 3NF decomposition algorithm
- 3NF synthesis algorithm
- Multivalued dependency
  - Equality-generating dependencies
  - Tuple-generating dependencies
  - Embedded multivalued dependencies
- Closure
- Fourth normal form (4NF)
- Restriction of D to Ri
- Fifth normal form
- Domain-key normal form (DKNF)
- Atomic domains
- First normal form (1NF)
- Unique-role assumption
- Denormalization
- Crosstabs
- Temporal data
- Snapshot
- Temporal functional dependency
- Temporal primary key
- Temporal foreign-key
- Temporal join

Part four

Application Design and Development

Chapter 8 complex data types

The relational model is very widely used for data representation for many application domains. One of the key requirements of the relational model is that data values be atomic: multivalued, composite, and other complex data types are disallowed by the core relational model. However, there are many applications where the constraints on data types imposed by the relational model cause more problems than they solve. In this chapter, we discuss several non-atomic data types that are widely used, including semi-structured data, object-based data, textual data, and spatial data.

Semi-structured Data

Flexible Schema

- Relational databases have fixed schemas with a fixed number of attributes.
- Schema changes, like adding new attributes, are rare and may require changes to application code.
- Such designs work well for many organizational applications.

Complex Data and Schema Changes

- Some applications deal with complex data that evolves rapidly, with frequent schema changes.
- Examples include user profiles with various attributes and complex data types.
- Semi-structured data models accommodate these needs.

Data Exchange

- Data exchange between applications and web services is crucial in modern architecture.
- JSON and XML data models are widely used for this purpose.
- They enable efficient data exchange and processing between backend servers and clients.

Overview of Semi-structured Data Models

- Relational data models can be extended to support modern application storage and data exchange needs.
- Flexible schema, multivalued data types, and nested data types are features of semi-structured data models.

JSON (JavaScript Object Notation)

- JSON is a textual representation of complex data types used for data exchange and storage.
- It supports primitives (integer, real, string), arrays, and objects (key-value pairs).
- JSON is widely used for communication between applications and web services.

XML (Extensible Markup Language)

- XML represents data using tags enclosed in angle brackets.
- Tags are used to create hierarchical structures, which are suitable for representing complex data and data exchange.
- SQL has been extended to support XML data types and queries.

RDF (Resource Description Framework)

- RDF is a data representation standard based on the entity-relationship model.
- Data in RDF is represented as triples, consisting of subject-predicate-object.
- RDF data forms knowledge graphs, allowing for flexible representation of facts and relationships.

SPARQL

- SPARQL is a query language for RDF data.
- It uses triple patterns with variables to query RDF knowledge graphs.
- SPARQL supports complex queries, including joins, filtering, and aggregation.

Representing N-ary Relationships

- RDF and knowledge graphs can represent complex relationships by using binary relationships and reification or additional attributes.

Knowledge Graphs

- Knowledge graphs store facts harvested from various sources and are used for answering queries based on relationships between entities.
- Examples of knowledge graphs include Wikidata, DBPedia, Freebase, and Yago.

Object Orientation

Object-Relational Database Systems

- Object-Relational Database Systems extend traditional relational database systems to include object-oriented features. They provide a richer type system that includes complex data types and object-oriented constructs. These systems allow users to create structured user-defined types, define type inheritance relationships, and even associate methods with these types. Additionally, they support table inheritance, allowing tables to be declared as sub tables of others. Object-Relational Database Systems aim to blend the strengths of both relational and object-oriented data models.

User-Defined Types

- In the context of Object-Relational Database Systems, user-defined types refer to custom data types that users can create to define the structure and behavior of data stored in the database. Users can create these types, define their attributes, and use them to represent complex data structures.

Type Inheritance

- Type inheritance in Object-Relational Database Systems allows one data type (subtype) to inherit attributes and methods from another data type (supertype). This mechanism mirrors the concept of inheritance in object-oriented programming, where a subtype inherits properties and behaviors from a parent type.

Table Inheritance

- Table inheritance is a feature that allows one database table to be declared as a sub table of another table. Sub tables inherit all the attributes present in their parent table. This concept is related to specialization/generalization relationships in entity-relationship modeling.

Reference Types in SQL

- Reference types in SQL, supported by some database systems like Oracle, allow the creation of data types that are references to other objects. These reference types can be used to establish relationships between data elements, and they often include a scope that specifies the source of the referenced data.

Object-Relational Mapping (ORM)

- Object-Relational Mapping is a programming technique that bridges the gap between the object-oriented programming language and the relational database. ORM systems facilitate the creation of a mapping between objects in the application code and tables in the database. They enable developers to work with objects in the programming language, and the ORM system translates these operations into corresponding database queries and updates.

Textual Data

Textual Data

- Textual data consists of unstructured text. It refers to any form of data represented in a textual format without a predefined structure.

Information Retrieval

- Information retrieval refers to the process of querying and retrieving unstructured textual data. It involves searching for documents or web pages that contain specific information or keywords.

Document

- In the traditional information retrieval model, textual information is organized into documents. A document can be any unit of text, such as a research paper, web page, or any text-based record.

Keyword Queries

- Keyword queries are used in information retrieval systems to retrieve documents that contain specific keywords or terms. A keyword query retrieves documents containing all the keywords in the query.

Relevance Ranking

- Relevance ranking is the process of determining how relevant a document is to a given query. It helps in ordering the search results to display the most relevant documents first.

TF-IDF (Term Frequency-Inverse Document Frequency)

- TF-IDF is a commonly used relevance ranking technique in information retrieval. It measures the relevance of a term to a document based on its term frequency (TF) and inverse document frequency (IDF). TF counts the number of times a term appears in a document, while IDF measures the rarity of the term across all documents.

Stop Words

- Stop words are common words (e.g., "and," "the," "a") that are often ignored in keyword-based information retrieval because they do not provide meaningful information for querying.

Ranking Using Hyperlinks

- Hyperlinks between documents, such as web pages, can be used to determine the importance or popularity of a document. Techniques like PageRank measure the popularity of pages based on the number and quality of links to them.

Measuring Retrieval Effectiveness

- Two important metrics for measuring the effectiveness of information retrieval systems are precision and recall. Precision measures the percentage of retrieved documents that are relevant, while recall measures the percentage of relevant documents that were retrieved.

Keyword Querying on Structured Data

- Keyword querying techniques have been extended to structured and semi-structured data, where data is represented as graphs, and keyword queries are used to find paths and relationships between data elements.

Knowledge Graphs

- Knowledge graphs are used to represent structured knowledge about entities and their relationships. They can be integrated with textual information to provide more accurate answers to queries by linking entities mentioned in text to their knowledge graph entries.

Web Search Engines

- Web search engines are a type of information retrieval system that crawls the web to index web pages and provide relevant results to user queries. They use various ranking algorithms, including PageRank and knowledge graphs, to improve search results.

Spatial Data

Geographic Data

- Geographic data typically refers to information related to the Earth's surface, such as maps, land usage, elevation, political boundaries, land ownership, etc. Geographic Information Systems (GIS) are specialized database systems designed for storing and managing geographic data. This type of data is based on a round-earth coordinate system, which includes latitude, longitude, and elevation.

Geometric Data

- Geometric data encompasses spatial information about objects and how they are constructed in a two-dimensional or three-dimensional Euclidean space. This type of data uses X, Y, and Z coordinates. Geometric data can represent various shapes, such as points, lines, polygons, and complex 3D objects like buildings, cars, or aircraft.

Spatial Data

- Several database systems, including Oracle Spatial and Graph, PostGIS, SQL Server, and IBM DB2 Spatial Extender, offer support for these types of spatial data.
- Additionally, spatial databases often support various spatial operations, such as union, intersection, distance calculations, and more, depending on the specific use case.
- Spatial data plays a crucial role in various applications, including mapping and navigation services, utility network management (e.g., telecommunications, power, water supply), environmental analysis (e.g., ecology, land usage), and geographic information systems (GIS) used in land records and urban planning.
- Queries involving spatial data can include region queries (e.g., finding objects within a specified region), nearness queries (e.g., finding objects within a given distance of a point), and spatial graph queries (e.g., finding shortest paths on a road network). These queries often involve spatial predicates and may combine spatial and non-spatial criteria.

Spatial Data Representation

- Spatial data can be represented in different ways, depending on the application and requirements. Some common representations include:
  - Points: Represented by their coordinates (latitude, longitude, elevation).
  - Lines and Polylines: Composed of connected line segments, represented by a sequence of endpoint coordinates.
  - Polygons: Represented by a list of vertices that define their boundaries.
  - Circles and Ellipses: Can be represented directly or approximated using polygons.
  - Raster Data: Represented as pixel maps or bitmaps in multiple dimensions (2D or 3D).
  - Vector Data: Comprised of basic geometric objects, such as points, lines, polygons, and polyhedral, in two or three dimensions.

Class activity

- Wide column
- Sparse column
- Key-value map
- Map
- Array database
- Tags
- Triples
- Resources
- Subject
- Predicate
- Object
- Knowledge graph
- Reification
- Quads
- Linked open data
- Object-relational data model
- Object-relational database system
- Object-relational mapping
- Object-oriented database system
- Path expression
- Keywords
- Keyword query
- Term
- Relevance
- TF–IDF
- Stop words
- Proximity
- PageRank
- Precision
- Recall
- Geographic data
- Geometric data
- Geographic information system
- Computer-aided-design (CAD)
- Polyline
- Line string
- Triangulation
- Spatial network
- Spatial graph
- Raster data
- Tiles
- Vector data
- Topographical information
- Triangulated irregular network (TIN)
- Overlays
- Nearness queries
- Nearest-neighbor query
- Region queries
- Spatial graph queries
- Spatial join

Chapter 9 application development

Practically all use of databases occurs from within application programs. Correspondingly, almost all user interaction with databases is indirect, via application programs. We will study tools and technologies that are used to build applications, focusing on interactive applications that use databases to store and retrieve data. A key requirement for any user-centric application is a good user interface. The two most common types of user interfaces today for database-backed applications are the web and mobile app interfaces.

Application Programs and User Interfaces

Front-End and Back-End Components

- Application programs responsible for interacting with databases consist of several components:
  - Front-End Component: This component deals with the user interface and how users interact with the application. It often includes forms, graphical elements, and user input mechanisms that allow users to input data and interact with the application. Front-end components are typically designed to provide a user-friendly experience.
  - Back-End Component: The back-end component of the application communicates with the database system to retrieve, update, or manipulate data based on user input. It contains the business logic that governs how data is accessed and processed. This component ensures that the database remains secure and enforces data integrity and business rules.

Authentication and Authorization

- Users are usually required to authenticate themselves to the application using credentials such as a username and password. Authentication ensures that users are who they claim to be. After authentication, the application checks the user's permissions to determine what actions they are authorized to perform. This helps maintain data security and access control.

Example: University Registration System

- An example provided is a university registration system. In such a system, users (students) would log in with their credentials. The application would then use this identity to retrieve information such as the user's name and enrolled courses from the database. The application provides various interfaces for actions like course registration and querying course and instructor information.

Customized User Experiences

- Application programs are used to create customized user experiences. Even when users don't explicitly fill out forms, applications can dynamically customize content based on user preferences and interactions. For example, a news website might customize content based on a user's browsing history.

Client-Server Architectures

- Traditionally, application programs ran on mainframe computers, and users interacted with them through terminals. With the growth of personal computers, client-server architectures emerged. In these architectures, client applications on personal computers communicate directly with shared databases. However, this approach had security and update challenges.

Web-Based Applications

- Web browsers and the Hypertext Markup Language (HTML) standard provide a universal front end for interacting with databases. Web-based applications can be accessed from any computer with a browser and internet connection. Web applications often use JavaScript for dynamic user interfaces. Application logic is executed on web servers, and web browsers communicate with these servers to request and display data.

Mobile Applications

- Mobile applications often communicate with back-end services through APIs (Application Programming Interfaces). These applications are installed on mobile devices and provide customized interfaces for small screens. The back end ensures security, authentication, and authorization.

API-Driven Development

- Modern application development often involves building APIs that can be used by various front-end technologies, including web and mobile. This approach allows for flexibility and reuse of back-end logic across multiple interfaces.

Web Fundamentals

Uniform Resource Locators (URLs)

- A Uniform Resource Locator (URL) is a globally unique name for each document accessible on the web. It consists of several parts, including the protocol (e.g., HTTP or HTTPS), the domain name (the name of the web server), and the path to the specific resource or document on the server. URLs can also contain parameters for program execution on the server.

Hypertext Markup Language (HTML)

- HTML is a markup language used to structure content on web pages. It uses tags to define elements like headings, paragraphs, tables, forms, and links. HTML documents are interpreted by web browsers to render web pages. The example in the passage demonstrates a simple HTML table.

HTML Forms

- HTML forms allow users to input data and interact with web applications. Forms are defined using HTML tags like <form>, <input>, <select>, and <button>. They can include various input types, such as text fields, radio buttons, checkboxes, and drop-down menus. Form data can be submitted to web servers for processing.

HTTP Methods

- The HTTP (Hypertext Transfer Protocol) defines two main methods for sending data from a browser to a web server:
  - GET Method: Data is appended to the URL as query parameters, making it visible in the URL. Typically used for retrieving data.
  - POST Method: Data is sent in the HTTP request body, making it not visible in the URL. Used for submitting data, such as form submissions.

Web Servers

- Web servers are software programs that run on server machines. They receive HTTP requests from web browsers, process those requests, and send back HTML documents as responses. Web servers can also execute programs and scripts to generate dynamic content. The Common Gateway Interface (CGI) is a standard for communication between web servers and application programs.

Cookies

- Cookies are small pieces of data stored on a user's device by web browsers. They are often used to track user sessions and store session-related information, such as session identifiers. Cookies can help maintain user state and authentication information across multiple requests.

Session Management

- Web applications typically need to manage user sessions to maintain state and authentication. Session identifiers (often stored in cookies) are used to associate subsequent requests with a specific user session. Sessions can be terminated after a period of inactivity or when a user logs out for security reasons.

Security Considerations

- Web applications may set cookies for session management only after authenticating users to enhance security. Additionally, cookies can be used to store user preferences and other session-related data.

HTML5

- HTML5 introduced new input types and attributes that enhance form capabilities, such as date and time pickers, file uploads, and input constraints.