DBMS

Week 10

Chapter 11 data analytics

Advanced SQL goes beyond intermediate-level concepts and delves into more specialized and sophisticated topics for handling complex data scenarios and optimizing database performance.

Overview of Analytics

Data Warehouse

- Large companies collect data from diverse sources, often with different schemas. To facilitate data analysis, they gather this data into a data warehouse, which stores data from multiple sources in a unified schema. This process is known as Extract, Transform, Load (ETL).

Aggregates and Reports

- Data analytics involves generating aggregates and reports to summarize data in meaningful ways. These summaries are often presented graphically in charts and dashboards, making it easier for analysts to visualize and monitor key organizational parameters.

Online Analytical Processing (OLAP)

- Efficient analysis of large data sets in real-time is essential for analysts. OLAP systems are designed for this purpose, allowing multidimensional data analysis, OLAP operations, relational representation of summaries, and graphical data visualization.

Statistical Analysis

- Statistical analysis is a crucial part of data analytics, and tools like R, SAS, and SPSS are commonly used for this purpose. R, in particular, is widely used and integrates with databases and Big Data systems, enabling parallel processing on large datasets.

Prediction

- Predictive analytics involves using historical data to make predictions about the future. For instance, banks can predict loan defaults by analyzing past customer data and building prediction models. Machine learning techniques play a significant role in finding patterns and making predictions.

Machine Learning and Data Mining

- Machine learning techniques are instrumental in data analysis and prediction. Data mining combines knowledge-discovery techniques with efficient implementations for analyzing large databases.

Business Intelligence (BI)

- The term "business intelligence" is used interchangeably with data analytics. It encompasses various activities related to data analysis and decision support within an organization.

Decision Support

- Decision support, in a narrower sense, focuses on reporting and aggregation tasks but does not typically include machine learning or data mining. SQL queries are often used for decision-support tasks, which differ from online transaction processing queries.

Data Warehousing

Data Warehouse

- A data warehouse is a centralized repository or archive of data gathered from various sources. It stores data under a unified schema and for an extended period, allowing access to historical data. Data warehouses provide a consolidated interface for data access, making it easier to perform decision-support queries.

Schema

- A schema defines the structure of data, including the organization of tables, attributes, and their relationships. In the context of data warehousing, schema integration involves creating a unified schema that combines data from different sources into a coherent format.

ETL (Extract, Transform, Load)

- ETL refers to the processes involved in collecting data from various sources (Extract), performing data transformation and cleansing (Transform), and then loading the transformed data into the data warehouse (Load). ETL is essential for preparing data for analysis in a data warehouse.

Star Schema

- A star schema is a type of data warehouse schema where a central fact table is connected to dimension tables through foreign keys. Fact tables store quantitative data (measures), while dimension tables store attributes used for grouping and filtering data.

Snowflake Schema

- A snowflake schema is a more complex data warehouse schema that extends the star schema by breaking down dimension tables into multiple levels of related tables. This creates a more normalized structure but may require more complex querying.

Column-Oriented Storage

- Column-oriented storage is a data storage method where values of the same attribute (column) are stored together in separate files or data structures. It is particularly useful for data warehouses because it allows for efficient compression and selective retrieval of columns needed for queries.

Concurrency Control

- Concurrency control is a mechanism in database systems that ensures multiple transactions can access and modify data without causing conflicts or inconsistencies. Data warehouses typically do not require concurrency control since they focus on analytical queries, not transaction processing.

Data Lake

- A data lake is a storage repository that holds vast amounts of raw data, including structured, semi-structured, and unstructured data. Unlike data warehouses, data lakes do not enforce a specific schema upfront, making them suitable for storing diverse data types. Querying data lakes often requires more flexible tools and processing.

Online Analytical Processing

Aggregation on Multidimensional Data

- Data analysis often involves finding patterns by grouping data in interesting ways.
- Multidimensional data can be analyzed using cross-tabulations (cross-tabs) or data cubes.
- Cross-tabs are two-dimensional tables that summarize data by grouping attributes.
- Data cubes are generalized versions of cross-tabs, representing data in multiple dimensions.
- Aggregations are used to summarize data within each cell of a cross-tab or data cube.

Relational Representation of Cross-Tabs

- Cross-tabs are typically used for data presentation but can't be directly stored in relational databases.
- To represent cross-tabs relationally, special values like "all" can be introduced to represent subtotals.
- Hierarchies can also be represented using relations.

OLAP in SQL

- SQL supports OLAP operations using the PIVOT, CUBE, and ROLLUP clauses.
- The PIVOT clause is used to generate cross-tabulations.
- The CUBE and ROLLUP clauses are used to create data cubes and rollup results hierarchically.
- The GROUPING SETS construct allows specifying custom groupings.

Reporting and Visualization Tools

- Report generators create human-readable summary reports by integrating database queries with formatting.
- Report structures can be defined using variables, tables, graphs, and charts.
- Report generators provide features for formatting, such as headers, subtotals, and page breaks.
- Data visualization tools help users examine large datasets through graphical representations.
- Interaction is crucial for data visualization, allowing users to drill down, filter, and explore data interactively.
- Modern tools like Tableau, FusionCharts, and Google Charts offer web-based data visualization and interactive dashboards.

Data Mining

Data Mining vs. Machine Learning and Statistics

- Data mining involves analyzing large databases to find useful patterns. While it shares similarities with machine learning and statistical analysis, data mining typically deals with large volumes of data stored on disk. However, the distinction between data mining and machine learning has become less clear, as machine learning algorithms also handle large datasets.

Knowledge Discovery in Databases (KDD)

- Data mining techniques are a part of the broader process of knowledge discovery in databases (KDD). KDD involves various steps, including data preprocessing, pattern discovery, and model building.

Types of Patterns

- Data mining can uncover different types of patterns, including association rules, equations, and models. These patterns can represent relationships or knowledge within the data.

Classification

- Classification is a common data mining task where the goal is to predict the class or category to which a new instance belongs. Techniques like decision tree classifiers, Bayesian classifiers, Support Vector Machines (SVMs), and neural network classifiers can be used for classification.

Regression

- Regression involves predicting a numerical value based on input variables. Linear regression is one method used for this purpose.

Association Rules

- Data mining can discover associations between items or attributes in a dataset. For example, it can identify that customers who buy bread are likely to buy milk as well.

Clustering

- Clustering aims to group similar data points together. It can be used for various purposes, such as customer segmentation or identifying related items.

Text Mining

- Text mining applies data mining techniques to textual documents. Tasks in text mining include sentiment analysis, information extraction, and entity recognition.

Knowledge Graphs

- Information extracted from documents can be organized into knowledge graphs, which are used to provide more meaningful answers to user queries and for various other applications.

Class activity

- Decision-support systems
- Business intelligence
- Data warehousing
  - Gathering data
  - Source-driven architecture
  - Destination-driven architecture
  - Data cleansing
  - Extract, transform, load (ETL)
  - Extract, load, transform (ELT)
- Warehouse schemas
  - Fact table
  - Dimension tables
  - Star schema
  - Snowflake schema
- Column-oriented storage
- Online analytical processing (OLAP)
- Multidimensional data
  - Measure attributes
  - Dimension attributes
  - Hierarchy
  - Cross-tabulation / Pivoting
  - Data cube
  - Rollup and drill down
  - SQL group by cube, group by rollup
- Data visualization
- Data mining
- Prediction
- Classification
  - Training data
  - Test data
- Decision-tree classifiers
- Bayesian classifiers
  - Bayes' theorem
  - Naive Bayesian classifiers
- Support Vector Machine (SVM)
- Regression
- Neural-networks
- Deep learning
- Association rules
- Clustering
- Text mining
  - Sentiment analysis
  - Information extraction
  - Named entity recognition
  - Knowledge graph

Tools

- Data Warehouse Systems:
  - Teradata: Provides scalable data warehousing solutions.
  - Teradata Aster: Offers advanced analytics and data warehousing capabilities.
  - SAP IQ (formerly Sybase IQ): A high-performance columnar database.
  - Amazon Redshift: A cloud-based data warehousing solution by AWS.
- ETL (Extract, Transform, Load) Tools:
  - Informatica: Offers a comprehensive ETL platform for data integration.
  - Business Objects: Provides ETL capabilities for SAP's Business Intelligence suite.
  - IBM InfoSphere: IBM's data integration and ETL solution.
  - Microsoft Azure Data Factory: Azure's cloud-based ETL service.
  - Microsoft SQL Server Integration Services (SSIS): Part of SQL Server for ETL.
  - Oracle Warehouse Builder: ETL tool for Oracle databases.
  - Pentaho Data Integration: An open-source ETL tool.
- OLAP (Online Analytical Processing) Tools:
  - Microsoft Analysis Services: Part of the Microsoft BI stack.
  - Oracle OLAP: Provides OLAP capabilities for Oracle databases.
  - IBM Cognos TM1: An IBM OLAP solution.
  - SAP BW (Business Warehouse): Part of SAP's analytics platform.
  - Mondrian OLAP server: An open-source OLAP server.
- Big Data Processing:
  - Apache Kafka: Used for building ETL systems and real-time data streaming.
  - Apache NiFi: Facilitates data integration and movement.
  - Apache Kylin: An OLAP engine for big data analytics.
- Data Visualization Tools:
  - Tableau: Popular for creating interactive and shareable dashboards.
  - FusionCharts: Offers interactive and animated charts.
  - Plotly: A Python graphing library for interactive visualizations.
  - Datawrapper: Simplifies the creation of responsive web charts.
  - Google Charts: Provides a variety of chart types for web applications.
- Machine Learning and Data Science:
  - Python: Widely used for machine learning with libraries like NumPy, SciPy, Pandas, Scikit-Learn, TensorFlow, and Keras.
  - R: Popular for statistical analysis and machine learning.
  - NLTK (Natural Language Toolkit): Python library for natural language processing.
  - Scrapy: Python framework for web crawling.
  - Matplotlib, Bokeh: Python libraries for data visualization.
  - Orange: An open-source data visualization and analysis tool.
- Data Mining Tools:
  - RapidMiner: An open-source data science platform for analytics.
  - Weka: A collection of machine learning algorithms for data mining tasks.
  - Orange: Offers data visualization and machine learning capabilities.
  - SAS Enterprise Miner, IBM Intelligent Miner, Oracle Data Mining: Commercial data mining solutions.

Part six

Storage Management and Indexing

Chapter 12 physical storage systems

We have emphasized the higher-level models of a database. For example, at the conceptual or logical level, we viewed the database, in the relational model, as a collection of tables. Indeed, the logical model of the database is the correct level for database users to focus on. This is because the goal of a database system is to simplify and facilitate access to data; users of the system should not be burdened unnecessarily with the physical details of the implementation of the system

Overview of Physical Storage Media

Cache

- Cache memory is the fastest but most costly form of storage. It is managed by the computer hardware and is relatively small in size. Database systems may consider cache effects when designing data structures and algorithms for query processing.

Main Memory

- Main memory is the storage medium used for data that can be operated on by the computer. It is volatile, meaning that its contents are lost in the event of a power failure or system crash. Main memory capacities can range from tens of gigabytes to hundreds or thousands of gigabytes.

Flash Memory

- Flash memory is non-volatile, meaning that stored data are retained even if power is turned off. It has a lower cost per byte compared to main memory but a higher cost per byte compared to magnetic disks. Flash memory is commonly used in devices like cameras, cell phones, USB flash drives, and solid-state drives (SSDs).

Magnetic Disk Storage

- Magnetic disks, also known as hard disk drives (HDDs), are the primary medium for long-term online storage of data. They are non-volatile and provide capacities ranging from 500 gigabytes to multiple terabytes. While cheaper than SSDs, magnetic disks offer lower performance in terms of data access operations per second.

Optical Storage

- Optical storage media like DVDs and Blu-ray discs are used for long-term storage, including backups of database contents. They use laser light to write and read data and have capacities of up to 128 gigabytes. However, optical storage is not suitable for storing active database data due to longer access times.

Tape Storage

- Magnetic tape storage is primarily used for backup and archival data, including data that must be stored safely for legal reasons. Tapes have high capacities (up to multiple terabytes) and are cost-effective for long-term storage. However, access to data is sequential, making it slower compared to magnetic disks.

Storage Hierarchy

- Storage media can be organized into a hierarchy based on speed, cost, and volatility. The fastest and most expensive storage, such as cache and main memory, is referred to as primary storage. Secondary storage includes media like flash memory and magnetic disks. Tertiary storage, the lowest level, includes media like magnetic tapes and optical disk jukeboxes.

Volatility

- Storage systems from main memory up in the hierarchy are volatile, meaning data is lost in case of power failure or system crashes. Storage systems from flash memory down are non-volatile, retaining data even when power is turned off.

Storage Interfaces

Disk Interfaces

- Magnetic disks and solid-state disks (SSDs) are connected to computer systems through high-speed interfaces. Two common interfaces for disks are Serial ATA (SATA) and Serial Attached SCSI (SAS). SATA-3 offers speeds of up to 6 gigabytes per second, while SAS version 3 supports rates of 12 gigabits per second.

NVMe Interface

- Non-Volatile Memory Express (NVMe) is a logical interface standard designed to better support SSDs. It is typically used with the PCIe (Peripheral Component Interconnect Express) interface, providing high-speed data transfer within computer systems.

Remote Disk Connections

- Disks can be situated remotely and connected to a computer system via a high-speed network. In storage area network (SAN) architectures, numerous disks are connected to multiple server computers via a high-speed network. This network setup often uses technologies like iSCSI, Fiber Channel (FC), or InfiniBand.

Storage Organization with RAID

- Redundant Arrays of Independent Disks (RAID) is a storage organization technique commonly used in SANs. It provides a logical view of a large and reliable disk by combining multiple physical disks.

Network Attached Storage (NAS)

- NAS is an alternative to SAN. In NAS, networked storage is presented as a file system, and it uses networked file system protocols such as NFS (Network File System) or CIFS (Common Internet File System).

Cloud Storage

- Cloud storage is becoming increasingly popular, where data is stored in the cloud and accessed via APIs. However, it may have higher latency, making it less suitable as the underlying storage for databases. It is often used for storing objects and is discussed in more detail in Section 21.7.

Magnetic Disks

Magnetic Disk

- A magnetic disk is a type of secondary storage device used in computer systems to store data. It consists of one or more circular platters coated with a magnetic material where data is recorded.

SSD (Solid-State Drive)

- SSD is a storage device that uses NAND-based flash memory to store data. It has no moving parts and offers faster data access and lower power consumption compared to traditional magnetic disks.

Per-Byte Cost

- Per-byte cost refers to the cost incurred for storing one unit of data (typically one byte) on a storage device. It's a measure of the cost-effectiveness of storage media.

Cylinder

- In the context of magnetic disks, a cylinder is a collection of tracks located at the same position on all platters within the disk assembly. It's used for efficient access to data across multiple platters.

Access Time

- Access time is the time it takes to locate and retrieve data from a storage device, including seek time (time to position the read-write head) and rotational latency (time to wait for the desired sector to rotate under the head).

Seek Time

- Seek time is the time required for the read-write head to move to the correct track on the disk's platter to access the desired data.

Rotational Latency

- Rotational latency is the time spent waiting for the desired data sector to rotate under the read-write head after the head has reached the correct track.

Data-Transfer Rate

- Data-transfer rate is the rate at which data can be read from or written to the disk, typically measured in megabytes per second (MB/s).

Block

- A block is a fixed-sized logical unit of data storage on a disk. It's the minimum amount of data that can be read from or written to the disk.

Page

- In some contexts, a page may refer to a block, but in others, especially in the context of flash memory, it might refer to a different unit of storage.

Sequential Access Pattern

- A sequential access pattern involves accessing data blocks in a consecutive or ordered manner. This pattern is characterized by minimal seek times and high data transfer rates.

Random Access Pattern

- A random-access pattern involves accessing data blocks in a non-sequential or arbitrary order. This pattern results in frequent seek operations and lower data transfer rates compared to sequential access.

Mean Time to Failure (MTTF)

- MTTF is a measure of reliability that indicates the average time a device or system can be expected to operate without failing. It's often used to estimate the lifespan of storage devices.

Flash Memory

NAND Flash

- NAND flash is a type of non-volatile semiconductor memory widely used for data storage. It stores data in memory cells made of floating-gate transistors. NAND flash is known for its high-density storage and relatively fast data access times. It is commonly used in USB drives, memory cards, and solid-state drives (SSDs).

NOR Flash

- NOR flash is another type of non-volatile semiconductor memory. It is like NAND flash but is primarily used for different purposes. NOR flash allows random access to individual memory cells, making it suitable for code storage, such as firmware in devices like microcontrollers and embedded systems.

Solid-State Drive (SSD)

- A solid-state drive is a data storage device that uses NAND flash or other non-volatile memory technology to store data. SSDs are designed to replace traditional hard disk drives (HDDs) and offer several advantages, including faster data access times, lower power consumption, and greater durability due to the absence of moving parts. They are commonly used in laptops, desktop computers, and data centers to improve storage performance.

RAID

Benefits of RAID

- Improved Performance: RAID allows multiple disks to operate in parallel, improving the rate at which data can be read or written. This parallelism enhances performance.
- Reliability: RAID introduces redundancy, meaning extra information is stored that can be used to rebuild lost data in case of disk failure. This improves the reliability of data storage.

RAID Levels

- RAID encompasses different disk organization techniques or levels. The passage mentions four commonly used RAID levels:
  - RAID 0: Non-redundant striping.
  - RAID 1: Disk mirroring with block striping.
  - RAID 5: Block-interleaved distributed parity.
  - RAID 6: P + Q redundancy (uses error-correcting codes).

Hardware vs. Software RAID

- RAID can be implemented using software modifications, known as software RAID, or with specialized hardware, known as hardware RAID. Hardware RAID often offers advantages such as non-volatile RAM for write caching, improved power failure recovery, and hot-swapping of disks.

Rebuild and Scrubbing

- Importance of rebuilding data on a failed disk and scrubbing to detect and correct latent failures. Hardware RAID systems often perform these tasks more efficiently.

Choice of RAID Level

- Selecting the appropriate RAID level depends on factors like cost, performance requirements, and data safety. The passage provides guidance on choosing RAID levels based on these considerations.

Other RAID Applications

- RAID concepts have been generalized beyond traditional disk arrays. For example, they are applied to flash memory devices (SSDs), tape arrays, and data broadcast systems to enhance reliability and recoverability.

Disk-Block Access

Buffering

- Blocks that are read from disk are temporarily stored in an in-memory buffer. Both the operating system and the database system use buffering to satisfy future requests efficiently. This helps reduce the number of disk reads and improves response times.

Read-Ahead

- In cases of sequential access, consecutive blocks from the same track are read into an in-memory buffer, even if there are no pending requests for those blocks. Read-ahead anticipates future requests and minimizes the time wasted in disk seeks and rotational latency. However, it's not very effective for random block accesses.

Scheduling

- Disk-arm-scheduling algorithms aim to optimize the order of accessing tracks to minimize disk-arm movement and reduce access time. The elevator algorithm is a commonly used approach, where the arm moves in one direction until it reaches the end, then reverses direction and repeats the process.

File Organization

- Organizing blocks on disk to match expected access patterns can significantly reduce block access time. For example, if a file is expected to be accessed sequentially, storing its blocks sequentially on adjacent cylinders can be beneficial. Modern disks use logical block numbering to maintain a sense of block adjacency.

Extent Allocation

- Instead of allocating individual blocks, the operating system often allocates larger groups of consecutive blocks called extents to files. This reduces the overhead of seeking for each block and can improve sequential access performance.

Defragmentation

- Over time, files can become fragmented with blocks scattered across the disk. Defragmentation utilities can scan the disk and rearrange blocks to decrease fragmentation, improving disk performance.

Non-Volatile Write Buffers

- To ensure data durability, updates need to be written to disk, which can be a slow process. Non-volatile write buffers, often implemented using technologies like NVRAM (battery-backed-up RAM or flash memory), speed up write operations by temporarily storing data in a non-volatile buffer. This allows the operating system or database system to acknowledge writes quickly and then reorder the writes for efficient disk access.

Class activity

- Physical storage media
  - Cache
  - Main memory
  - Flash memory
  - Magnetic disk
  - Optical storage
  - Tape storage
- Volatile storage
- Non-volatile storage
- Sequential-access
- Direct-access
- Storage interfaces
  - Serial ATA (SATA)
  - Serial Attached SCSI (SAS)
  - Non-Volatile Memory Express (NVMe)
  - Storage area network (SAN)
  - Network attached storage (NAS)
- Magnetic disk
  - Platter
  - Hard disks
  - Tracks
  - Sectors
  - Read–write head
  - Disk arm
  - Cylinder
  - Disk controller
  - Checksums
  - Remapping of bad sectors
- Disk block
- Performance measures of disks
  - Access time
  - Seek time
  - Latency time
  - I/O operations per second (IOPS)
  - Rotational latency
  - Data-transfer rate
  - Mean time to failure (MTTF)
- Flash Storage
  - Erase Block
  - Wear leveling
  - Flash translation table
  - Flash Translation Layer
- Storage class memory
  - 3D-XPoint
- Redundant arrays of independent disks (RAID)
  - Mirroring
  - Data striping
  - Bit-level striping
  - Block-level striping
- RAID levels
  - Level 0 (block striping, no redundancy)
  - Level 1 (block striping, mirroring)
  - Level 5 (block striping, distributed parity)
  - Level 6 (block striping, P + Q redundancy)
- Rebuild performance
- Software RAID
- Hardware RAID
- Hot swapping
- Optimization of disk-block access
  - Disk-arm scheduling
  - Elevator algorithm
  - File organization
  - Defragmenting
  - Non-volatile write buffers
  - Log disk

Chapter 13 data storage structures

The characteristics of physical storage media, focusing on magnetic disks and SSDs, and saw how to build fast and reliable storage systems using multiple disks in a RAID structure.

Database Storage Architecture

Persistent Data Storage

- Persistent data, such as those used in databases, are stored on non-volatile storage devices like magnetic disks or SSDs. These devices are typically block-structured, meaning data is read from or written to them in units of blocks.

Use of Operating System Files

- Most databases use operating system files as an intermediate layer for storing records. These files abstract away some of the details of the underlying block structure. However, databases still need to be aware of blocks for efficient access and recovery from failures.

Organizing Records in Files

- The organization of records in database files is an important decision. Records can be stored in sorted order, in the order they are created, or in arbitrary order. Section 13.3 discusses various file organization techniques.

Data Dictionary

- Databases maintain a data dictionary that contains information about relational schemas and storage organization. The data dictionary is crucial for tasks like locating and retrieving records of a relation based on its name.

Database Buffer

- Since main memory is needed for CPU access, databases that are larger than available main memory need to fetch data from non-volatile storage. Section 13.5 describes how databases use a region of memory called the database buffer to store blocks that are fetched from non-volatile storage.

Column-Oriented Storage

- Column-oriented storage, which stores all values of a particular column together, is discussed in Section 13.6. This storage approach is particularly useful for analytical query processing.

Main-Memory Databases

- Some applications require very fast data access and have small enough data sizes to fit the entire database into the main memory of a server. These databases are known as main-memory databases. Section 13.7 discusses storage organization in main-memory databases and how they optimize data structures and query processing for in-memory data.

Storage Class Memory

- The passage mentions the development of non-volatile memory known as storage class memory, which allows direct access to individual bytes or cache lines. This type of memory can be advantageous for main-memory database architectures.

File Organization

Database Files

- A database is mapped into multiple files that are maintained by the operating system. These files are permanently stored on disks.

Logical Organization

- Each file is logically organized as a sequence of records. These records are mapped onto disk blocks, which serve as both storage allocation units and units for data transfer.

Block Sizes

- Most databases use block sizes ranging from 4 to 8 kilobytes (KB) by default, although some databases allow block sizes to be specified when creating a database instance. Larger block sizes can be advantageous in specific applications.

Record Structure

- Records are organized within blocks, with each block potentially containing multiple records. The exact set of records in a block depends on the chosen physical data organization.

Record Size

- The passage assumes that no record is larger than a block. While this is realistic for many data-processing applications, strategies for handling larger data items are discussed later.

Record Deletion and Insertion

- Managing deleted and newly inserted records is discussed. Records may be marked as deleted, and space may be reused for new records or kept open for future insertions. Techniques are provided to avoid moving records for efficient insertions and deletions.

Variable-Length Records

- The passage introduces the concept of variable-length records, which arise from attributes like strings or repeating fields. Variable-length records are represented using offset and length values in the record structure.

Slotted-Page Structure

- Variable-length records are stored within blocks using a slotted-page structure. The block header contains information about the number of entries, the end of free space, and an array that records the location and size of each record. Records are stored contiguously in the block, and space management techniques are used for insertions, deletions, and updates.

Large Objects

- Databases often store data larger than a block as large objects. These large objects may be stored separately from the rest of the record or within the database. They can be managed as files in a file system or using file structures within the database.

File System Integration

- Some databases support integration with the file system for managing large objects. This integration helps enforce constraints, access authorizations, and ensures referential integrity between files and database records.

Organization of Records in Files

Heap File Organization

- In a heap file organization, records can be placed anywhere in the file where there is available space. There is no specific ordering of records.
- Records may be inserted at the end of the file or in locations with available space after deletions.
- A free-space map is commonly used to track which blocks have free space for storing records.
- Efficiently locating blocks with free space is essential to optimize insertion.

Sequential File Organization

- In a sequential file organization, records are stored in sequential order based on a search key attribute.
- Records are linked using pointers to allow fast retrieval in search-key order.
- Insertions and deletions may disrupt physical sequential order, but pointer chains are used to maintain order.
- Over time, reorganization may be required to restore the physical sequential order, especially when inserts and deletes are frequent.

Multitable Clustering File Organization

- In this organization, records of multiple relations are stored together in the same block within a file to optimize certain types of join operations.
- Clustering is done based on a specified attribute (cluster key) that defines which records are stored together.
- This organization can improve query performance for join operations but may increase the cost of certain other queries.
- Oracle database systems support Multitable clustering using clusters and cluster keys.

Partitioning

- Partitioning involves splitting a large relation into smaller, separate relations based on attribute values (e.g., by year).
- It helps optimize query performance by reducing the size of each relation, particularly for queries that involve partitioning attributes.
- Partitioning can also be used to store different parts of a relation on different storage devices, optimizing access speed and storage space usage.

Data-Dictionary Storage

Metadata in a Relational Database

- Metadata refers to "data about data" in a database system. In the context of a relational database, it includes information about the structure of the database, such as the schema of relations, attributes, data domains, views, and integrity constraints.

Data Dictionary or System Catalog

- Metadata, including relational schemas and other information about relations, is stored in a structure known as the data dictionary or system catalog. This catalog serves as a repository for all the metadata needed to manage the database.

Types of Metadata

- The system catalog stores various types of metadata, including:
  - Names of relations
  - Names of attributes in each relation
  - Data domains and attribute lengths
  - Definitions of views and their names
  - Integrity constraints (e.g., key constraints)
  - Information about database users, their default schemas, and authentication data
  - Authorization information for each user
  - Statistical and descriptive data about relations and attributes

Storage Organization and Location

- The metadata may also include information about the storage organization of relations and where each relation is physically stored. This can be important for efficient data retrieval.

Metadata about Indexes

- In Chapter 14, the passage mentions that metadata about indexes on relations is also stored, including index names, the relation being indexed, indexed attributes, and the type of index.

Representing Metadata as Relations

- Metadata information is often represented as relations within the database itself. This approach simplifies the system's structure and allows for efficient access to system data. The choice of how to represent metadata as relations is made by system designers.

Illustrative Schema

- The passage includes an illustrative schema diagram of a toy data dictionary, which represents part of the system metadata. In practice, real implementations store more extensive metadata.

Nonnormalized Form

- System metadata is often stored in a nonnormalized form to achieve faster access. While normalization can be applied, denormalized representations are chosen for performance reasons.

Accessing Metadata

- When retrieving records from a relation, the database system consults the Relation metadata to determine the location and storage organization of the relation. This information is used to fetch records from the relation.

In-Memory Data Structures

- Database systems frequently load system metadata into in-memory data structures during startup to enable efficient access. This ensures that metadata is readily available for query processing.

Database Buffer

Buffer Manager

- The buffer manager is responsible for managing the allocation of main memory space (referred to as the buffer) for storing copies of disk blocks. When programs in a database system need to access a block from disk, they make requests to the buffer manager. The buffer manager ensures that frequently used blocks are kept in memory to minimize the need for disk access.

Buffer Replacement Strategy

- When there is no available space in the buffer to accommodate a new block, a block needs to be evicted (removed) from the buffer to make room. The choice of which block to evict depends on the buffer replacement strategy. Traditional operating systems often use the Least Recently Used (LRU) scheme, but database systems may employ more sophisticated strategies based on their specific knowledge of block access patterns.

Toss-Immediate Strategy

- This strategy involves immediately freeing the space occupied by a block once it is no longer needed, even if it has been recently accessed. This strategy can be effective when processing operations that only require blocks to be accessed once.

Most Recently Used (MRU) Strategy

- MRU is an alternative to LRU. It assumes that the most recently used block is the one most likely to be accessed again next. This strategy is suitable when processing patterns indicate that recently accessed blocks will not be revisited soon.

Concurrency Control and Buffer Management

- Concurrency control mechanisms may impact buffer management decisions. Delayed requests due to concurrency control requirements can influence the choice of which blocks to keep in the buffer.

Reordering Writes and Recovery

- To ensure data consistency and recoverability in the event of system crashes, databases often implement a logging mechanism. Writes are first logged in a specific order to a log disk or log file. This allows for delayed writing of modified blocks to the actual disk, which can be reordered for efficiency. Journaling file systems employ similar techniques to maintain file system metadata consistency.

Journaling File Systems

- Journaling file systems use log disks or log files to record write operations to ensure file system metadata consistency. They expedite system restarts by eliminating the need for extensive consistency checks after a crash.

Database Logging

- Database systems implement their own logging mechanisms to ensure the recoverability of database contents. These logs record database changes in a structured manner, allowing the system to recover to a consistent state after a crash.

Column-Oriented Storage

Row-Oriented vs. Column-Oriented Storage

- Traditionally, databases use row-oriented storage, where all attributes of a tuple (row) are stored together in a record. In contrast, column-oriented storage stores each attribute of a relation (table) separately, with values of the same attribute from successive tuples stored together.

Column-Oriented Storage Layout

- In a column-oriented storage system, each attribute is stored in a separate file, and each file is typically compressed to reduce its size. This storage layout is well-suited for analytical query processing.

Advantages of Column-Oriented Storage

- Reduced I/O: When a query accesses only a few attributes of a relation with many attributes, irrelevant attributes are not fetched from disk into memory, reducing I/O and query execution cost.
- Improved CPU Cache Performance: Column-oriented storage benefits from modern CPU architectures, as it allows for efficient caching of consecutive attribute values.
- Improved Compression: Storing values of the same type together enhances compression efficiency, reducing retrieval time from disk.
- Vector Processing: Column-oriented storage supports vector processing, which enables parallel processing of operations on attribute values.

Use Cases

- Column-oriented storage is well-suited for data warehousing applications, especially for data analysis queries that involve processing many rows but only accessing some attributes.

Drawbacks for Transaction Processing

- Tuple Reconstruction: Reconstructing a full tuple from its individual attributes can be costly, especially if many columns need to be reconstructed.
- Tuple Deletion and Update: Deleting or updating a single tuple in a compressed representation may require rewriting a large unit of compressed data.
- Decompression Overhead: Fetching data from compressed storage requires decompression, which can be less efficient for transaction processing queries that access only a few records.

Hybrid Approaches

- Some database systems offer hybrid storage, allowing both row-oriented and column-oriented representations. Tuples can be migrated from row-oriented to column-oriented storage when they are less likely to be accessed in a row-oriented manner.

Real-World Implementations

- Several databases and research projects have adopted column-oriented storage, including SAP HANA, Sybase IQ, C-Store, Vertica, Monet DB, and Vector wise.

Flexible Choices

- Some systems allow flexibility in choosing which attributes to store together, providing a spectrum of choices between pure row-oriented and pure column-oriented storage.

Storage Organization in Main-Memory Databases

Main Memory Advantages

- With modern large and affordable main memory sizes, many organizational databases can entirely fit in memory. This eliminates the need for frequent disk I/O operations to read data, significantly improving database performance.

Buffer Management

- In a traditional disk-based database, a buffer manager is used to manage the movement of data between disk and memory. However, this overhead is unnecessary in a main-memory database.

Record Access Optimization

- In disk-based databases, accessing a record involves multiple steps, including checking if the block is in the buffer and locating the record within the buffer. In contrast, a main-memory database can use direct pointers to records in memory, making record access highly efficient.

Slotted-Page Structure

- When using a slotted-page structure in a block, records may move within the block due to updates or deletions. This requires extra checks and potentially locks. To avoid these overheads, some main-memory databases avoid using a slotted-page structure and ensure that records never move due to updates.

Memory Fragmentation

- Main memory can become fragmented over time, especially if variable-sized records are inserted and deleted. Databases must implement memory management schemes or periodic compaction to prevent fragmentation.

Column-Oriented Storage

- In main-memory databases, column-oriented storage schemes are efficient because all values of a column can be stored consecutively in memory. However, when new data is appended to a relation, ensuring contiguous allocation may require reallocating existing data. To avoid this overhead, logical arrays for a column may be divided into multiple physical arrays, with an indirection table to locate specific elements efficiently.

Processing Optimizations

- Main-memory databases offer various opportunities for processing optimizations, which are explored in later chapters.