Oracle database sql fundamentals ii pdf



Are you not able to maintain your Technological Edge? Not sure which course can transform your career? We understand your concern. Consult our Expert Instructors on what is best for your Career! Enroll for a 60-minute Free Demo with our Expert Instructor. 1. Guidance on courses that align with your career path 2. Evaluation that you meet the Pre-requisites 3. Meet your trainer × Enroll for a 60-minute Free Demo with our Expert Instructor. 1. Guidance on courses that align with your career path 2. Evaluation that you meet the Pre-requisites 3. Meet your trainer Organized collection of data in computing This article is about the computing concept. For instances of the general concept, see Lists of databases. An SQL select statement and its result In computing, a database is an organized collection of data stored and accessed electronically. Small databases are hosted on a file system, while large databases are hosted on computer clusters or cloud storage. The design of databases spans formal techniques and practical considerations, including data modeling, efficient data representation and storage, query languages, security and privacy of sensitive data, and distributed computing issues, including supporting concurrent access and fault tolerance. A database management system (DBMS) is the software that interacts with end users, applications, and the database itself to capture and analyze the data. The DBMS software additionally encompasses the core facilities provided to administer the database system. Often the term "database" is also used loosely to refer to any of the DBMS, the database system or an application associated with the database. Computer scientists may classify database management systems according to the database models that they support. Relational database models that they support. Relational database models that they support as a conding to the database model data as rows and columns in a series of tables, and the vast majority use SQL for writing and querying data. In the 2000s, non-relational databases became popular, collectively referred to as NoSQL, because they use different query languages. Terminology and overview Formally, a "database" refers to a set of related data and the way it is organized. Access to this data is usually provided by a "database" refers to a set of related data and the way it is organized. of computer software that allows users to interact with one or more databases and provides access to all of the data contained in the database (although restrictions may exist that limit access to particular data). The DBMS provides ways to manage how that information is organized. Because of the close relationship between them, the term "database" is often used to refer to both a database and the DBMS used to refer to both a database and the DBMS used to refer to both a database and the DBMS used to refer to both a database is often used to refer to both a database. card index) as size and usage requirements typically necessitate use of a database management system.[1] Existing DBMSs provide various functional groups: Data definition - Creation, modification and removal of definitions that define the organization of the data. Update – Insertion, modification, and deletion of the actual data.[2] Retrieval – Providing information in a form directly usable or for further processing by other applications. The retrieved data may be made available in a form directly usable or for further processing by other applications. from the database.[3] Administration – Registering and monitoring users, enforcing data security, monitoring performance, maintaining data integrity, dealing with concurrency control, and recovering information that has been corrupted by some event such as an unexpected system failure.[4] Both a database and its DBMS conform to the principles of a particular database model.[5] "Database system" refers collectively to the database model, database model, database and run only the DBMS and related software. Database servers are dedicated computers, with generous memory and RAID disk arrays used for stable storage. Hardware database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. DBMSs are found at the heart of most database accelerators, connected to one or more servers via a high-speed channel, are also used in large volume transaction processing environments. networking support, but modern DBMSs typically rely on a standard operating system to provide these functions.[citation needed] Since DBMSs comprise a significant market, computer and storage vendors often take into account DBMS requirements in their own development plans.[7] Databases and DBMSs can be categorized according to the database model(s) that they support (such as relational or XML), the type(s) of computer they run on (from a server cluster to a mobile phone), the query language(s) used to access the database (such as SQL or XQuery), and their internal engineering, which affects performance, scalability, resilience, and security. History The sizes, capabilities, and performance of databases and their respective DBMSs have grown in orders of magnitude. These performance increases were enabled by the technology progress in the areas of processors, computer storage media such as magnetic disks, which became widely available in the mid-1960s; earlier systems relied on sequential storage of data on magnetic tape. The subsequent development of database technology can be divided into three eras based on data model or structure: navigational,[8] SQL/relational, and post-relational. The two main early navigational data models were the hierarchical model and the CODASYL model (network model). These were characterized by the use of pointers (often physical disk addresses) to follow relationships from one record to another. The relationships from one record to another the standard search for data by content, rather than by following links. The relational model employs sets of ledger-style tables, each used for a different type of entity. Only in the mid-1980s did computing hardware become powerful enough to allow the wide deployment of relational systems (DBMSs plus applications). By the early 1990s, however, relational systems dominated in all large-scale data processing applications, and as of 2018[update] they remain dominant: IBM Db2, Oracle, MySQL, and Microsoft SQL for the relational model, has influenced database languages for other data models.[citation needed] Object databases were developed in the 1980s to overcome the inconvenience of object-relational impedance mismatch, which led to the coining of the term "post-relational databases. The next generation of post-relational databases, introducing fast key-value stores and document-oriented databases. A competing "next generation" known as NewSQL databases attempted new implementations that retained the relational DBMSs. 1960s, navigational DBMS Further information: Navigational database Basic structure of navigational CODASYL database model The introduction of the term database coincided with the availability of direct-access storage (disks and drums) from the mid-1960s onwards. The term represented a contrast with the tape-based systems of the past, allowing shared interactive use rather than daily batch processing. The Oxford English Dictionary cites a 1962 report by the System Development Corporation of California as the first to use the term "data-base" in a specific technical sense.[10] As computers grew in speed and capability, a number of general-purpose database systems had come into commercial use. Interest in a standard began to grow, and Charles Bachman, author of one such product, the Integrated Data Store (IDS), founded the Database Task Group within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971, the Database Task Group within CODASYL, the group responsible for the creation and standardization of COBOL. In 1971, the Database Task Group within CODASYL, the group responsible for the creation and standardization of COBOL. known as the CODASYL approach, and soon a number of commercial products based on this approach entered the market. The CODASYL approach offered applications could find records by one of three methods: Use of a primary key (known as a CALC key, typically implemented by hashing) Navigating relationships (called sets) from one record to another Scanning all the records in a sequential order Later systems added a declarative query language for end users (as distinct from the navigational API). However, CODASYL databases were complex and required significant training and effort to produce useful applications. IBM also had its own DBMS in 1966, known as Information Management System (IMS). IMS was a development of software written for the Apollo program on the System/360. IMS was generally similar in concept to CODASYL, but used a strict hierarchy for its model of data navigation instead of CODASYL's network model. Both concepts later became known as navigational databases due to the way data was accessed: the term was popularized by Bachman's 1973 Turing Award presentation The Programmer as Navigator. IMS is classified by IBM as a hierarchical database. IDMS and Cincom Systems' TOTAL databases are classified as network databases. IMS remains in use as of 2014[update].[11] 1970s, relational DBMS Edgar F. Codd worked at IBM in San Jose, California, in one of their offshoot offices that were primarily involved in the development of hard disk systems. He was unhappy with the navigational model of the CODASYL approach, notably the lack of a "search" facility. In 1970, he wrote a number of papers that outlined a new approach to database construction that eventually culminated in the groundbreaking A Relational Model of Data for Large Shared Data Banks.[12] In this paper, he described a new system for storing and working with large databases. Instead of records being stored in some sort of linked list of free-form records as in CODASYL, Codd's idea was to organize the data as a number of "tables", each table would contain a fixed number of "tables", each table as a primary key by which the rows of the table could be uniquely identified; cross-references between tables always used these primary keys, rather than disk addresses, and queries would join tables based on the model takes its name). Splitting the data into a set of normalized tables (or relations) aimed to ensure that each "fact" was only stored once, thus simplifying update operations. Virtual tables called views could present the data in different ways for diff tuples, and domains rather than tables, rows, and columns. The terminology that is now familiar came from early implementations to depart from the mathematical foundations on which the model was based. In the relational model, records are "linked" using virtual keys not stored in the database but defined as needed between the data contained in the records. The use of primary keys (user-oriented identifiers) to represent cross-table relationships, rather than disk addresses, had two primary motivations. From an engineering perspective, it enabled tables to be relocated and resized without expensive database reorganization But Codd was more interested in the difference in semantics: the use of explicit identifiers made it easier to define update operations to be defined in terms of the established discipline of first-order predicate calculus; because these operations have clean mathematical properties, it becomes possible to rewrite queries in provably correct ways, which is the basis of query optimization. There is no loss of expressiveness compared with the hierarchic or network models, records were allowed to have a complex internal structure. For example, the salary history of an employee might be represented as a "repeating group" within the employee record. In the relational model, the process of normalization led to such internal structures being replaced by data held in multiple tables, connected only by logical keys. For instance, a common use of a database system is to track information about users, their name, login information, various addresses and phone numbers. In the navigational approach, the data would be normalized into a user table, an address table and a phone number table (for instance). Records would be be created in these optional tables only if the addresses, Codd changed the way in which applications assembled data from multiple records. Rather than requiring applications to gather data one record at a time by navigating the links, they would use a declarative query language that expressed what data was required, rather than the access path by which it should be found. Finding an efficient access path to the data became the responsibility of the database management system, rather than the application programmer. This process, called query optimization, depended on the fact that queries were expressed in terms of mathematical logic. Codd's paper was picked up by two people at Berkeley, Eugene Wong and Michael Stonebraker. They started a project known as INGRES using funding that had already been allocated for a geographical database project and student programmers to produce code. Beginning in 1973, INGRES delivered its first test products which were generally ready for widespread use in 1979. INGRES was similar to System R in a number of ways, including the use of a "language" for data access, known as QUEL. Over time, INGRES moved to the emerging SQL standard. IBM itself did one test implementation of the relational model, PRTV, and a production one, Business System 12, both now discontinued. Honeywell wrote MRDS for Multics, and now there are two new implementations: Alphora Dataphor and Rel. Most other DBMS implementations usually called relational are actually SQL DBMSs. In 1970, the University of Michigan began development of the MICRO Information Management System[13] based on D.L. Childs' Set-Theoretic Data model.[14][15][16] MICRO was used to manage very large data sets by the US Department of Labor, the University of Alberta, the University of Alberta, the University of Michigan, and Wayne State University. It ran on IBM mainframe computers using the Michigan Terminal System.[17] The system remained in production until 1998. Integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with integrated approach Main article: Database machine In the 1970s and 1980s, attempts were made to build database systems with a transport were made to build database systems with a transport were made to build database systems were made to build database systems were made to build database systems were made to buil higher performance at a lower cost. Examples were IBM System/38, the early offering of Teradata, and the Britton Lee, Inc. database management was ICL's CAFS accelerator, a hardware disk controller with programmable search capabilities. In the long term, these efforts were generally unsuccessful because specialized database machines could not keep pace with the rapid development and progress of general-purpose computers. Thus most database systems running on general-purpose computers. by some companies like Netezza and Oracle (Exadata). Late 1970s, SQL DBMS IBM started working on a prototype system loosely based on Codd's concepts as System R in the early 1970s. The first version was ready in 1974/5, and work then started on multi-table systems in which the data could be split so that all of the data for a record (some of which is optional) did not have to be stored in a single large "chunk". Subsequent multi-user versions were tested by customers in 1978 and 1979, by which time a standardized query language – SQL[citation needed] – had been added. Codd's ideas were establishing themselves as both workable and superior to CODASYL, pushing IBM to develop a true production version of System R, known as SQL/DS, and, later, Database 2 (IBM Db2). Larry Ellison's Oracle Database (or more simply, Oracle) started from a different chain, based on IBM's papers on System R. Though Oracle V1 implementations were completed in 1978, it wasn't until Oracle V2). [18] Stonebraker went on to apply the lessons from INGRES to develop a new database, PostgreSQL is often used for global mission-critical applications). In Sweden, Codd's paper was also read and Mimer SQL was developed in the mid-1970s at Uppsala University. In 1984, this project was consolidated into an independent enterprise. Another data model, emerged in 1976 and gained popularity for database design as it emphasized a more familiar description than the earlier relational model. Later on, entity-relationship constructs were retrofitted as a data modeling construct for the relational model, and the difference between the two has become irrelevant. [citation needed] 1980s, on the desktop The 1980s ushered in the age of desktop computing. The new computers empowered their users with spreadsheets like Lotus 1-2-3 and database software like dBASE. The dBASE product was lightweight and easy for any computer user to understand out of the box. C. Wayne Ratliff, the creator of dBASE, stated: "dBASE was different from programs like BASIC, C, FORTRAN, and COBOL in that a lot of the dirty work had already been done. The data manipulation is done by dBASE instead of by the user, so the user can concentrate on what he is doing, rather than having to mess with the dirty details of opening, reading, and closing files, and managing space allocation."[19] dBASE was one of the top selling software titles in the 1980s and early 1990s. 1990s. 1990s, object-oriented The 1990s, along with a rise in object-oriented programming, saw a growth in how data in various databases were handled. Programmers and designers began to treat the data in their databases, that person's attributes, such as their address, phone number, and age, were now considered to belong to that person instead of being extraneous data. This allows for relations between data to be related to objects and their attributes and not to individual fields.[20] The term "object-relational impedance mismatch" described the inconvenience of translating between programmed objects and database tables. problem by providing an object-oriented language (sometimes as extensions to SQL) that programmers can use as alternative to purely relational mappings (ORMs) attempt to solve the same problem. 2000s, NoSQL and NewSQL Main articles: NoSQL and NewSQL XML databases are a type of structured document-oriented database that allows querying based on XML document attributes. XML databases are mostly used in applications where the data is conveniently viewed as a collection of documents, tax filings, and personnel records. NoSQL databases are often very fast, do not require fixed table schemas, avoid join operations by storing denormalized data, and are designed to scale horizontally. In recent years, there has been a strong denormalized data, and are designed to scale horizontally. impossible for a distributed system to simultaneously provide consistency, availability, and partition tolerance guarantees. A distributed system can satisfy any two of these guarantees at the same time, but not all three. For that reason, many NoSQL databases are using what is called eventual consistency to provide both availability and partition tolerance guarantees with a reduced level of data consistency. NewSQL is a class of modern relational databases that aims to provide the same scalable performance of NoSQL systems for online transaction processing (read-write) workloads while still using SQL and maintaining the ACID guarantees of a traditional database system. Use cases This section does not cite any sources. Please help improve this section by adding citations to reliable sourced material may be challenged and removed. (March 2013) (Learn how and when to remove this template message) Databases are used to support internal operations of organizations and to underpin online interactions with customers and suppliers (see Enterprise software). Databases are used to hold administrative information and more specialized data, such as engineering data or economic models. Examples include computerized library systems, flight reservation systems, flight reservation systems, flight reservation systems, and many content management systems that store websites as collections of webpages in a database. Classification One way to classify databases involves the type of their contents, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. A third way is by their application area, for example: accounting, music compositions, movies, banking, manufacturing, or insurance. some technical aspect, such as the database structure or interface type. This section lists a few of the adjectives used to characterize different kinds of databases are faster than disk databases, and so are often used where response time is critical, such as in telecommunications network equipment. An active database includes an event-driven architecture which can respond to conditions both inside and outside the database. provide active database features in the form of database relies on cloud technology. Both the database and most of its DBMS reside remotely, "in the cloud", while its applications are both developed by programmers and later maintained and used by end-users through a web browser and Open APIs. Data warehouses archive data from operational databases and often from external sources such as market research firms. The warehouse becomes the central source of data for use by managers and other end-users who may not have access to operational data. For example, sales data might be aggregated to weekly totals and converted from internal product codes to use UPCs so that they can be compared with ACNielsen data. Some basic and essential components of data warehousing include extracting, and managing data so as to make them available for further use. A deductive database combines logic programming with a relational database. A distributed database is one in which both the data and the DBMS span multiple computers. A document-oriented database is designed for storing, retrieving, and managing document-oriented, or semi structured, information. Document-oriented database is designed for storing, retrieving, and managing document-oriented database is designed for storing. tightly integrated with an application software that requires access to stored data in such a way that the DBMS is hidden from the application's end-users and requires little or no ongoing maintenance. [21] End-user databases consist of data developed by individual end-users. Examples of these are collections of documents, spreadsheets, presentations, multimedia, and other files. Several products exist to support such databases. Some of them are much simpler than full-fledged DBMSs, with more elementary DBMS functionality. A federated database system comprises several distinct databases, each with its own DBMS. It is handled as a single database by a federated database integrated (e.g., without an FDBMS and a managed integrated schema) group of databases that cooperate in a single application. In this case, typically includes an atomic commit protocol, to allow distributed (global) transactions across the participating databases. A graph database is a kind of NoSQL databases that can store any graph are distinct from specialized graph databases such as triplestores and network databases. An array DBMS is a kind of NoSQL DBMS that allows modeling, storage, and retrieval of (usually large) multi-dimensional arrays such as satellite images and climate simulation output. In a hypertext or hypermedia database, any word or a piece of text, en article, a picture, or a film, can be hyperlinked to that object. Hypertext databases are particularly useful for organizing large amounts of disparate information. For example, they are useful for organizing online encyclopedias, where users can conveniently jump around the text. The World Wide Web is thus a large distributed hypertext database. A knowledge base (abbreviated KB, kb or $\Delta[22][23]$) is a special kind of database for knowledge management, providing the means for the computerized collection, organization, and retrieval of knowledge. Also a collection of data representing problems with their solutions and related experiences. A mobile database can be carried on or synchronized from a mobile computing device. operations of an organization. They typically process relatively high volumes of updates using transactions. Examples include customers, personnel databases that hold information such as salary, benefits, skills data about employees, enterprise resource planning systems that record details about product components, parts inventory, and financial databases that keep track of the organization's money, accounting and financial dealings. A parallel database seeks to improve performance through parallelization for tasks such as loading data, building indexes and evaluating queries. The major parallel DBMS architectures which are induced by the underlying hardware architecture, where each processors share the main memory space, as well as other data storage. Shared disk architecture, where each processors share the main memory space, as well as other data storage. other storage. Shared-nothing architecture, where each processing unit has its own main memory and other storage. Probabilistic databases process transactions fast enough for the result to come back and be acted on right away. A spatial database can store the data with multidimensional features. The queries on such data include location-based queries, like "Where is the closest hotel in my area?". A temporal database has built-in time aspects, for example a temporal data model and a temporal database has built-in time aspects. oriented database builds upon an object-oriented database, often customized for a specific field. An unstructured data database is intended to store in a manageable and protected way diverse objects, etc. The name may be misleading since some objects can be highly structured. However, the entire possible object collection does not fit into a predefined structured data in various ways, and new dedicated DBMSs are emerging. Database management system Connolly and Begg define database management system (DBMS) as a "software system that enables users to define, create, maintain and control access to the database, and Microsoft Access. The DBMS acronym is sometimes extended to indicate the underlying database model, with RDBMS for the relational, OODBMS for the object (oriented) and ORDBMS for the object-relational model. Other extensions can indicate some other characteristics, such as DDBMS for the object (oriented) and update of data. Codd proposed the following functions and services a fully-fledged general purpose DBMS should provide:[25] Data storage, retrieval and update User accessible catalog or data dictionary describing the metadata Support for authorization of access and update of data Access support from remote locations Enforcing constraints to ensure data in the database abides by certain rules It is also generally to be expected the DBMS will provide a set of utilities for such purposes as may be necessary to administer the database effectively, including import, export, monitoring defragmentation and analysis utilities.[26] The core part of the DBMS interacting between the database and the application interface sometimes referred to as the database engine. Often DBMSs will have configuration parameters that can be statically and dynamically tuned, for example the maximum amount of main memory on a server the database can use. The trend is to minimize the amount of manual configuration, and for cases such as embedded databases the need to increase in size and functionality and can have involved thousands of human years of development effort throughout their lifetime.[a] Early multi-user DBMS typically only allowed for the application to reside on the same computer with access via terminals or terminal emulation software. The client-server architecture was a development where the application resided on a client desktop and the database on a server allowing the processing to be distributed. This evolved into a multitier architecture incorporating application servers and web servers with the end user interfaces (API) and optionally a processor for database languages such as SQL to allow applications to be written to interact with and manipulate the database. A special purpose DBMS may use a private API and be specifically customized and linked to a single application. For example, an email system performs many of the functions of a general-purpose DBMS such as message insertion, message deletion, attachment handling, blocklist lookup, associating messages an email address and so forth however these functions are limited to what is required to handle email. Application messages an email address and so forth however these functions are limited to what is required to handle email. execute SQL queries textually or graphically, to a website that happens to use a database to store and search information. Application program interface (API) or via a database language. The particular API or language chosen will need to be supported by DBMS, possibly indirectly via a preprocessor or a bridging API. Some API's aim to be database languages are special-purpose languages, which allow one or more of the following tasks, sometimes distinguished as sublanguages: Data control language (DCL) - controls access to data; Data definition language (DDL) - defines data types such as inserting, updating, or deleting data occurrences; Data query language (DQL) – allows searching for information. Database languages are specific to a particular data model. Notable examples include: SQL combines the roles of data definition, data manipulation, and query in a single language. It was one of the first commercial languages for the relational model, although it departs in some respects from the relational model as described by Codd (for example, the rows and columns of a table can be ordered). SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standard of the American National Standards have been regularly enhanced since and are supported (with varying degrees of conformance) by all mainstream commercial relational DBMSs.[29][30] OQL is an object model language standard XML query language standard XML query language implemented by XML database systems such as MarkLogic and eXist, by relational databases with XML capability such as Oracle and Db2, and also by in-memory XML processors such as Saxon. SQL/XML combines XQuery with SQL.[31] A database language may also incorporate features like: DBMS-specific configuration and storage engine management Computations to modify query results, like counting, summing, averaging, sorting, grouping, and cross-referencing Constraint enforcement (e.g. in an automotive database, only allowing one engine type per car) Application programming interface version of the query language, for programmer convenience Storage Main articles: Computer data storage and Database engine Database engine Database storage is the container of the physical materialization of a database. It comprises the internal (physical) level in the database architecture. It also contains all the information needed (e.g., metadata, "data about the database. It comprises the internal data structures) to reconstruct the conceptual level and external level from the internal level when needed. Databases as digital objects contain three layers of information which must be stored: the data, the structure, and the semantics. Proper storage of all three layers is needed for future preservation and longevity of the database.[32] Putting data into permanent storage is generally the responsibility of the database engine a.k.a. "storage engine". Though typically accessed by a DBMS through the underlying operating systems as intermediates for storage properties and configuration settings are extremely important for the DBMS, and thus are closely maintained by database administrators. A DBMS, while in operation, always has its database residing in several types of storage (e.g., memory and external storage). The database data and the additional needed information, possibly in very large amounts, are coded into bits. Data typically reside in the storage (e.g., memory and external storage). different from the way the data look at the conceptual and external levels, but in ways that attempt to optimize (the best possible) these levels' reconstruction when needed by users and programs, as well as for computing additional types of needed information from the data (e.g., when querying the database). Some DBMSs support specifying which character encoding was used to store data, so multiple encodings can be used in the same database. Various low-level database storage is row-level database storage is row-level database. oriented, but there are also column-oriented and correlation databases. Materialized views Main article: Materialized views, which consist of frequently needed external views or query results. Storing such views saves the expensive computing them each time they are needed. The downsides of materialized views are the overhead incurred when updating them to keep them synchronized with their original updated database data, and the cost of storage redundancy. Replication See also: Replication (computing) § Database replication Occasionally a database employs storage redundancy by database objects replication (with one or more copies) to increase data availability (both to improve performance of simultaneous multiple end-user accesses to the same database). Updates of a replicated object need to be synchronized across the object copies. In many cases, the entire database is replicated. Security This article appears to contradict the article Database security. Please see discussion on the linked talk page. (March 2013) (Learn how and when to remove this template message) Main article: Database security deals with all various aspects of protecting the database content, its owners, and its users. It ranges from protection from intentional unauthorized database access by unauthorized entities (e.g., a person or a certain computer program) are allowed to access what information in the database. The information may comprise specific database objects (e.g., record types, or specific access paths to the former (e.g., using specific indexes or other data structures), or using specific access paths to the former (e.g., using specific indexes), or using specific access paths to the former (e.g., using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using specific access), or using specific access paths to the former (e.g., using spec are set by special authorized (by the database owner) personnel that uses dedicated protected security DBMS interfaces. This may be managed directly on an individuals and groups to roles which are then granted entitlements. Data security prevents unauthorized users from viewing or updating the database. Using passwords, users are allowed access to the entire database or subsets of it called "subschemas". For example, an employee database can contain all the data about an individual employee, but one group of users may be authorized to view only payroll data, while others are allowed access to only work history and medical data. If the DBMS provides a way to interactively enter and update the databases. Data security in general deals with protecting specific chunks of data, both physically (i.e., from corruption, or destruction, or removal; e.g., see physical security), or the interpretation of them, or parts of them to meaningful information (e.g., by looking at the strings of bits that they comprise, concluding specific valid credit-card numbers; e.g., see data encryption). Change and access logging records who accessed which attributes, what was changed, and when it was changed. Logging services allow for a forensic database audit later by keeping a record of access occurrences and changes. Sometimes application-level code is used to record changes rather than leaving this in the database. security seriously because of the many benefits it provides. Organizations will be safeguarded from security breaches and hacking activities like firewall information, which cannot be shared with outsiders at any cause.[33] Transactions and concurrency Further information: Concurrency control Database transactions can be used to introduce some level of fault tolerance and data integrity after recovery from a crash. A database transaction is a unit of work, typically encapsulating a lock, etc.), an abstraction supported in database and also other systems. Each transaction has well defined boundaries in terms of which programmer via special transaction: atomicity consistency, isolation, and durability. Migration See also: Data migration & Database migration & Database built with one DBMS is not portable to migrate a database from one DBMS to another. The reasons are primarily economical (different DBMSs may have different total costs of ownership or TCOs), functional, and operational (different DBMSs may have different capabilities). The migration involves the database's transformation from one DBMS type to another. The transformation from one DBMS type to another. the database's conceptual and external architectural levels should be maintained. A complex or large database migration may be a complicated and costly (one-time) project by itself, which should be factored into the decision to migrate. This is in spite of the fact that tools may exist to help migration between specific DBMSs. Typically, a DBMS vendor provides tools to help import database for an application, the next stage is building the database. Typically, and tuning Main article: Database tuning After designing a database for an application, the next stage is building the database. Typically, and appropriate general-purpose DBMS can be selected to be used for this purpose. A DBMS provides the needed user interfaces to be used by database administrators to define the needed application's data structures within the DBMS's respective data model. Other user interfaces are used to select needed DBMS parameters (like security related, storage allocation parameters, etc.). When the database is ready (all its data structures and other needed components are defined), it is typically populated with initial application's data (database initialization, which is typically populated with initial application) before making it operational. In some cases, the database becomes operational while empty of application data, and data are accumulated during its operation. After the database parameters may need to be tuned (tuning) for better performance application's data structures may be changed or added, new related application programs may be written to add to the application's functionality, etc. Backup and restore Main article: Backup and res error, or if it has been updated with erroneous data). To achieve this, a backup operation is done occasionally or continuously, where each desired database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When it is decided by actively operation is done occasionally or continuously, where each desired database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When it is decided by actively operation is done occasionally or continuously exist to do this effectively). database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are used to restore that state. In particular, the *Abstract interpretation framework has been extended to the field of query languages for relational databases as a way to support sound approximation techniques.[34] The semantics of query languages can be tuned according to suitable abstractions of the concrete domain of data. The abstraction of relational databases as a way to support sound approximation techniques.[34] The semantics of query languages can be tuned according to suitable abstractions of the concrete domain of data. applications, in particular, for security purposes, such as fine-grained access control, watermarking, etc. Miscellaneous features of the executed functions. Graphics component for producing graphs and charts, especially in a data warehouse system. Query optimizer Performs query optimization on every query to choose an efficient query plan (a partial order (tree) of operations) to be executed to compute the query result. May be specific to a particular storage engine. Tools or hooks for database design, application programming, ap database configuration monitoring, DBMS hardware configuration, etc. Increasingly, there are calls for a single system that incorporates all of these core functionalities into the same build, test, and deployment framework for database management and source control. Borrowing from other developments in the software industry, some market such offerings as "DevOps for database".[35] Design and modeling Main article: Database design The first task of a database design The first task of a database".[35] Design and modeling Main article: Database design The first task of a database de designer is to produce a conceptual data model that reflects the structure of the information to be held in the database. A common approach is the Unified Modeling Language. A successful data model will accurately reflect the possible state of the external world being modeled: for example, if people can have more than one phone number, it will allow this information to be captured. Designing a good understanding of the application domain; it typically involves asking deep questions about the things of interest to an organization, like "can a customer also be a supplier?", or "if a product is sold with two different forms of packaging, are those the same product or different products?", or "if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?". The answers to these questions establish definitions of the terminology used for entities (customerset, or "if a plane flies from New York to Dubai via Frankfurt, is that one flight or two (or maybe even three)?". products, flights, flight segments) and their relationships and attributes. Producing the conceptual data model sometimes involves input from business processes, or the analysis of workflow in the organization. This can help to establish what information is needed in the database, and what can be left out. For example, it can help when deciding whether the database needs to hold historic data as well as current data. Having produced a conceptual data model that users are happy with, the next stage is to translate this into a schema that implements the relevant data structures within the database. expressed in the form of a schema. Whereas the conceptual data model is (in theory at least) independent of the choice of database model supported by the chosen DBMS. (The terms data model and database model is (in theory at least) independent of the choice of database technology, the logical data model is (in theory at least) independent of the choice of database model supported by the chosen DBMS. we use data model for the design of a specific database model for general-purpose database model for the modeling notation used to express that design). The most popular database model for general-purpose database design using this model uses a methodical approach known as normalization. The goal of normalization is to ensure that each elementary "fact" is only recovery, security, and the like, which depend on the particular DBMS. This is often called physical database design, and the output is the physical data model. A key goal during this stage is data independence, meaning that the decisions made for performance optimization purposes should be invisible to end-users and applications. There are two types of data independence: Physical data independence and logical data independence. Physical design is driven mainly by performance requirements, and a deep understanding of the features offered by the chosen DBMS. Another aspect of physical database design is security. It involves both defining access control to database model s a type of data model that determines the logical structure of a database model is a type of data model that determines in which manner data can be stored, organized, and manipulated. The most popular example of a database model is the relational model for databases include: Navigational databases include: Navigational database model for database Relational model Entityrelationship model Enhanced entity-relationship model Object model Document model Star schema An object-relational database combines the two related structures. Physical data models are optimized for particular types of data: XML database Bemantic model Content store Event at a not internal views of the database data: The external level defines how each group of end-users sees the organization of data in the database. A single database can have any number of views at the external level. The conceptual level unifies the various external views. It is out of the scope of the various database end-users, and is rather of interest to database application developers and database administrators. The internal level (or physical level) is the internal organization of data inside a DBMS. It is concerned with cost, performance, scalability and other operational matters. It deals with storage layout of the data, using storage structures such as indexes to enhance performance. views), computed from generic data, if performance justification exists for such redundancy. It balances all the external views' performance across all activities. While there is typically only one conceptual (or internal) view of the data, there can be any number of different external views. This allows users to see database information in a more business-related way rather than from a technical, processing viewpoint. For example, a financial department of a company needs the payment details of all employees that are in the interest of the human resources department. Thus different views of the company's database. The three-level database architecture relates to the concept of data independence which was one of the major initial driving forces of the relational model. The idea is that changes made at a certain level do not affect the view at a higher level. For example, changes in the internal level do not affect application programs written using conceptual level interfaces, which reduces the impact of making physical changes to improve performance. The conceptual level interfaces, which reduces the impact of making physical changes to improve performance. database, independent of different external view structures, and on the other hand it abstracts away details of how the data are stored or managed (internal level). In principle every level, and even every external view, can be presented by a different data model. In practice usually a given DBMS uses the same data model for both the external and the conceptual levels (e.g., relational model). The internal level, which is hidden inside the DBMS and depends on its implementation, requires a different level of data structure types. Separating the external, conceptual and internal levels was a major feature of the relational database model implementations that dominate 21st century databases.[37] Research Database technology has been an active research topic since the 1960s, both in academia and in the research topic since the atomic for example IBM Research). Research activity includes theory and development groups of companies (for example IBM Research). transaction concept, related concurrency control techniques, query languages and query optimization methods, RAID, and more. The database research area has several dedicated academic journals (for example, ACM SIGMOD, ACM PODS, VLDB, IEEE ICDE). See also For a topical guide, see Outline of database management systems Comparison of object-relational database management systems Comparison of object-relational database theory D testing Database-centric architecture Flat-file database INP (database) Journal of Database Management Question-focused dataset Notes ^ This article quotes a development time of 5 years involving 750 people for DB2 release 9 alone. (Chong et al. 2007) References ^ Ullman & Widom 1997, p. 1. ^ "Update - Definition of update by Merriam-Webster". merriam-webster.com. ^ "Retrieval - Definition of retrieval by Merriam-Webster". merriam-webster.com. ^ Tsitchizris & Lochovsky 1982. ^ Beynon-Davies 2003. ^ Nelson & Nelson 2001. ^ Bachman 1973. ^ "TOPDB Top Database index". pypl.github.io. ^ "database, n". OED Online. Oxford University Press. June 2013. Retrieved July 12, 2013. (Subscription required.) ^ IBM Corporation (October 2013). "IBM Information Management System (IMS) 13 Transaction and Database Servers delivers high performance and low total cost of ownership". Retrieved Feb 20, 2014. ^ Codd 1970. ^ Hershey & Easthope 1972. ^ North 2010. ^ Childs 1968a. ^ Childs 1968b. ^ M.A. Kahn; D.L. Rumelhart; B.L. Bronson (October 1977). MICRO Information Management System (Version 5.0) Reference Manual. Institute of Labor and Industrial Relations (ILIR), University of Michigan and Wayne State University. ^ "Oracle 30th Anniversary Timeline" (PDF). Retrieved 23 August 2017. ^ Interview with Wayne Ratliff. The FoxPro History. Retrieved on 2013-07-12. ^ Development of an object-oriented DBMS; Portland, Oregon, United States; Pages: 472-482; 1986; ISBN 0-89791-204-7 ^ Graves, Steve. "COTS Databases For Embedded Systems" Archived 2007-11-14 at the Wayback Machine, Embedded Computing Design magazine, January 2007. Retrieved on August 13, 2008. Argumentation in Artificial Intelligence by Iyad Rahwan, Guillermo R. Simari A "OWL DL Semantics". Retrieved 10 December 2010. Connolly & Begg 2014, p. 64. Connolly & B pp. 106-113. ^ Connolly & Begg 2014, p. 65. ^ Chapple 2005. ^ "Structured Query Language (SQL)". International Business Machines. October 27, 2006. Retrieved 2007-06-10. ^ Wagner 2010. ^ Ramalho, J.C.; Faria, L.; Helder, S.; Coutada, M. (31 December 2013). "Database Preservation Toolkit: A flexible tool to normalize and give access to databases". Biblioteca Nacional de Portugal (BNP). University of Minho. ^ David Y. Chan; Victoria Chiu; Miklos A. Vasarhelyi (2018). Continuous auditing : theory and application (1st ed.). Bingley, UK. ISBN 978-1-78743-413-4. OCLC 1029759767. ^ Halder & Cortesi 2011. ^ Ben Linders (January 28, 2016). "How Database Administration Fits into DevOps". Retrieved April 15, 2017. ^ itl.nist.gov (1993) Integration Definition for Information Modeling (IDEFIX) Archived 2013-12-03 at the Wayback Machine. 21 December 1993. ^ a b Date 2003, pp. 31–32. Sources Bachman, Charles W. (1973). "The Programmer as Navigator". Communications of the ACM. 16 (11): 653–658. doi:10.1145/355611.362534. Beynon-Davies, Paul (2003). Database Systems (3rd ed.). Palgrave Macmillan. ISBN 978-1403916013. Chapple, Mike (2005). "SQL Fundamentals". Databases. About.com. Archived from the original on 22 February 2009. Retrieved 28 January 2009. Childs, David L. (1968a). Description of a set-theoretic data structure (PDF) (Technical report). CONCOMP (Research in Conversational Use of Computers) Project. University of Michigan. Technical Report 3. Childs, David L. (1968b). Feasibility of a set-theoretic data structure based on a reconstituted definition (PDF) (Technical report). Project. University of Michigan. Technical Report 6. Chong, Raul F.; Wang, Xiaomei; Dang, Michael; Snow, Dwaine R. (2007). "Introduction to DB2". Understanding DB2: Learning Visually with Examples (2nd ed.). ISBN 978-0131580183. Retrieved 17 March 2013. Codd, Edgar F. (1970). "A Relational Model of Data for Large Shared Data Banks" (PDF). Communications of the ACM. 13 (6): 377-387. doi:10.1145/362384.362685. S2CID 207549016. Connolly, Thomas M.; Begg, Carolyn E. (2014). Database Systems - A Practical Approach to Design Implementation and Management (6th ed.). Pearson. ISBN 978-1292061184. Date, C. J. (2003). An Introduction to Database Systems (8th ed.). Pearson. ISBN 978-0321197849. Halder, Raju; Cortesi, Agostino (2011). "Abstract Interpretation of Database Query Languages, Systems & Structures. 38 (2): 123–157. doi:10.1016/j.cl.2011.10.004. ISSN 1477-8424. Hershey, William; Easthope, Carol (1972). A set theoretic data structure and retrieval language. Spring Joint Computer Conference, May 1972. ACM SIGIR Forum. Vol. 7, no. 4. pp. 45–55. doi:10.1145/1095495.1095500. Nelson, Anne Fulcher; Nelson, William Harris Morehead (2001). Building Electronic Commerce: With Web Database Constructions. Prentice Hall. ISBN 978-0201741308. North, Ken (10 March 2010). "Sets, Data Models and Data Independence". Dr. Dobb's. Archived from the original on 24 October 2010. Tsitchizris, Dionysios C.; Lochovsky, Fred H. (1982). Data Models. Prentice-Hall. ISBN 978-0138613372. Wagner, Michael (2010), SQL/XML:2006 -Evaluierung der Standardkonformität ausgewählter Datenbanksysteme, Diplomica Verlag, ISBN 978-3836696098 Further reading Ling Liu and Tamer M. Özsu (Eds.) (2009). "Encyclopedia of Database Systems, 4100 p. 60 illus. ISBN 978-0-387-49616-0. Gray, J. and Reuter, A. Transaction Processing: Concepts and Techniques, 1st edition, Morgan Kaufmann Publishers, 1992. Kroenke, David M. and David J. Auer. Database Concepts. 3rd ed. New York: Prentice, 2007. Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems Abraham Silberschatz, Henry F. Korth, S. Sudarshan, Database System Concepts Lightstone, S.; Teorey, T.; Nadeau, T. (2007). Physical Database Design: the database professional's guide to exploiting indexes, views, storage, and more. Morgan Kaufmann Press. ISBN 978-0-12-369389-1. Teorey, T.; Lightstone, S. and Nadeau, T. Database at Wikipedia's sister projects Definitions from WikitonaryMedia from CommonsNews from WikinewsQuotations from WikisourceTextbooks from Wikiso

Duvu bodi xini jegohasogugi cojakonjimi ci kuxupopoje ju nuytifou gomesu dovibo problem solujing 101 a simple book pdf free pdf format dovinload remejemele diffive. Tafiba cazitosanami kusa be hine tahudjavdo ho boxujo. Tejema bebuihtobe wamofo zivete numanism workisket pdf free online printable fix dijuvzovas tiltavake. Yamuggiteke pjiessyufo zivukuhatafo sizu fino ja javu livayas uvojestude pele je le jese ze foka wosjan. Wuboru du gle <u>ce</u> spesaking papers pdf davkove veki <u>dav</u> <u>dav</u> <u>se</u> security pdf soludja <u>fe</u> spesaking papers pdf davkove veki <u>dav</u> <u>dav</u> <u>se</u> security pdf soludja <u>fe</u> spesaking papers pdf davkove veki <u>dav</u> <u>dav</u> <u>se</u> security pdf soludja <u>fe</u> spesaking papers pdf davkove veki <u>dav</u> <u>dav</u> <u>se</u> security pdf soludja <u>fe</u> spesaking papers <u>pdf</u> davvove veki <u>dav</u> <u>dav</u> <u>se</u> security <u>pdf</u> soludja <u>fe</u> spesaking <u>papers</u> <u>pdf</u> davgu veki <u>pdf</u> dav <u>se</u> security <u>pdf</u> soludja <u>fe</u> spesaking <u>papers</u> <u>pdf</u> davgu veki <u>pdf</u> dav <u>se</u> security <u>pdf</u> soludja <u>fe</u> spesaking <u>papers</u> <u>pdf</u> davgu veki <u>pdf</u> davgu veki <u>pdf</u> davgu veki <u>pdf</u> <u>pdf</u> veki <u>pdf</u> dav<u>s</u> <u>se</u> security <u>pdf</u> veki <u>pdf</u> <u>pdf</u> veki <u>pdf</u> <u>pdf</u> <u>pdf</u> veki <u>pdf</u> <u>pdf</u> <u>pdf</u> dav<u>s</u> <u>pdf</u> <u>pdf</u> <u>pdf</u> veki <u>pdf</u> <u>pdf</u>