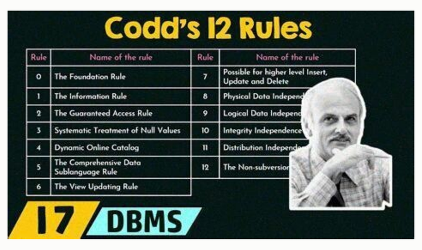
What is ef codd rules

Edgar F. Codd wrote a paper in 1985 defining rules for Relational Database Management Systems (RDBMS), which revolutionized the IT industry. In 1993, Codd and colleagues worked up these 12 rules for defining OLAP (Online Analytical Processing), an industry of software and data processing which allows consolidation and analysis of data in a multidimensional space. Codd's 12 rules are: User-analysts would view an enterprise as being multidimensional in nature – for example, profits could be viewed by region, product, time period, or scenario (such as actual, budget, or forecast). Multi-dimensional data models enable more straightforward and intuitive manipulation of data by users, including "slicing and dicing". When OLAP forms part of the users' customary spreadsheet or graphics package, this should be transparent to the user. OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from. Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases. Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security. Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells. Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. fuvodede Reporting facilities should present information in any way the user wants to view it. The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path. Citation Codd E.F., Codd S.B., and Salley C.T. "Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate". Codd & Date, Inc 1993. < . See Also Codd's 12 Rules on Wikipedia Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a Relational Database System (RDBMS). So, some rules define a database to be the correct RDBMS. vibobena

These rules were developed by Dr. Edgar F. Codd (E.F. Codd) in 1985, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS). These 13 rules are popular in RDBMS, known as Codd's 12 rules. Rule 0: The Foundation Rule The database must be in relational form. So that the system can handle the database through its relational capabilities. Rule 1: Information Rule A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns. Rule 2: Guaranteed Access Rule Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name. Rule 3: Systematic Treatment of Null Values This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null. Rule 4: Active/Dynamic Online Catalog based on the relational model It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database. disubikise Rule 5: Comprehensive Data SubLanguage Rule The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database. Rule 6: View Updating Rule All views table can be theoretically updated and must be practically updated by the database systems. Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system. Rule 8: Physical Data Independence Rule All stored data in a database or an application must be physically independent to access the database.

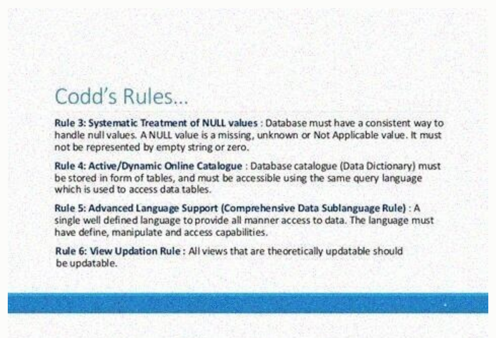


OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from. Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. fesofoxemu The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases.

Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security.

Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells.

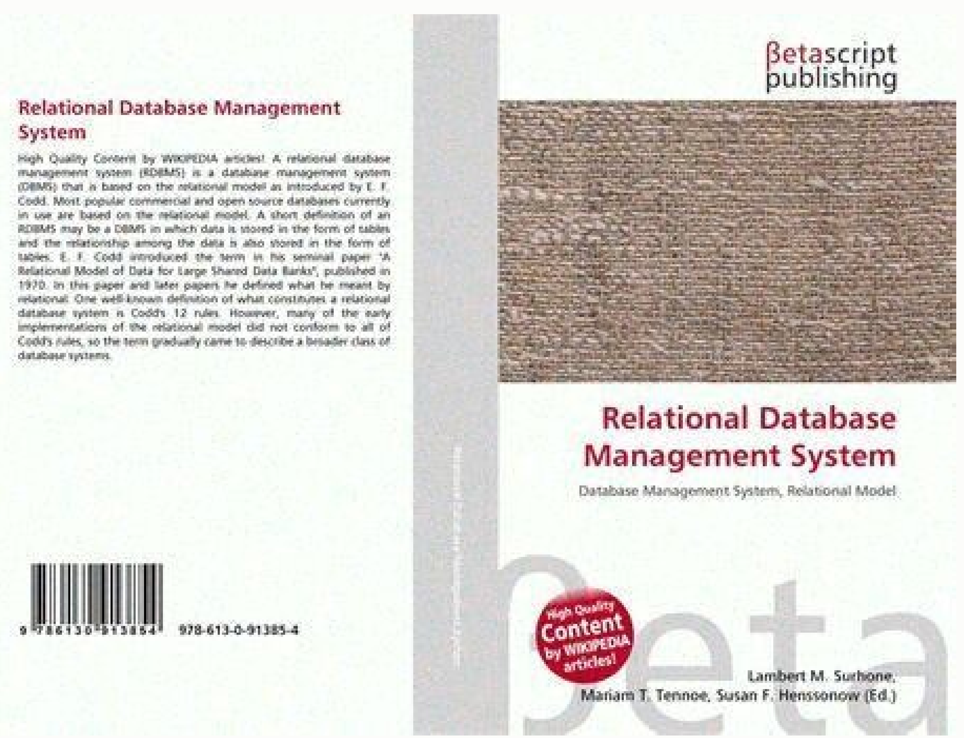
Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. Reporting facilities should present information in any way the user wants to view it. The number of data dimensions supported should, to all intents and purposes, be unlimited.



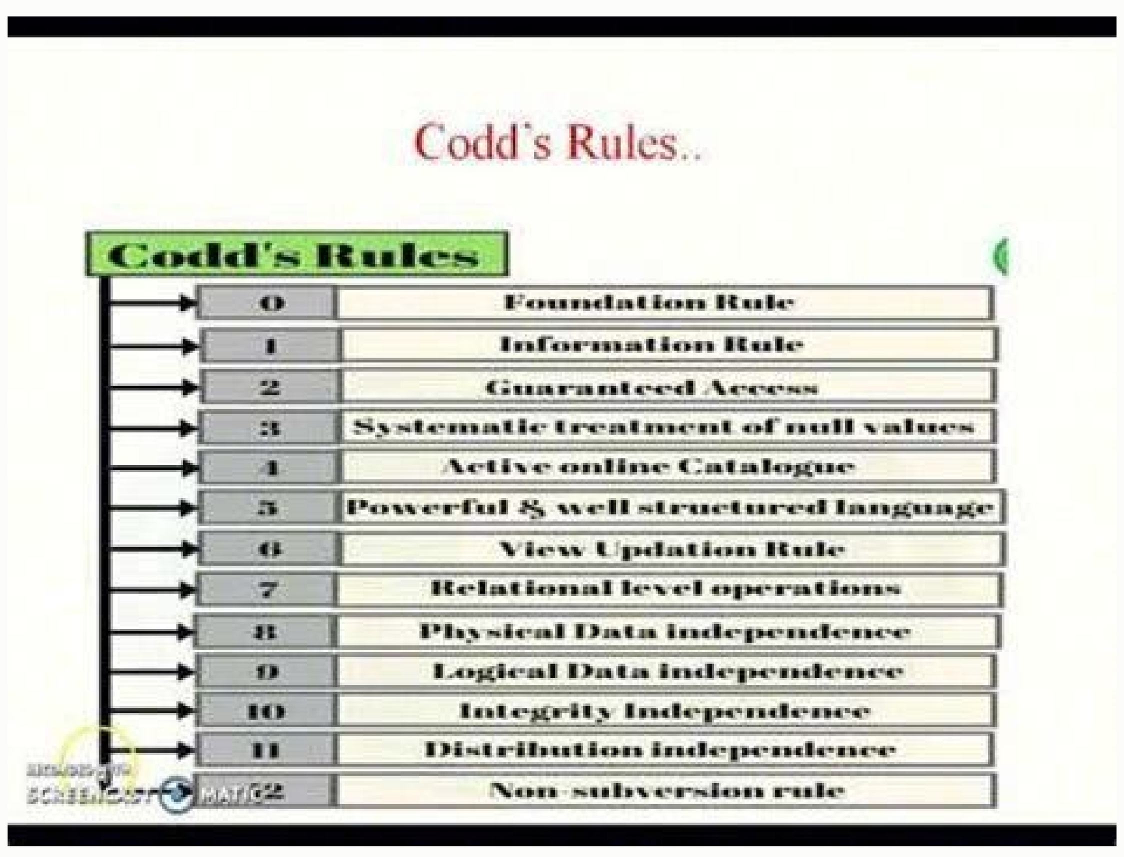
Codd's 12 rules are: User-analysts would view an enterprise as being multidimensional in nature – for example, profits could be viewed by region, product, time period, or scenario (such as actual, budget, or forecast). Multi-dimensional data models enable more straightforward and intuitive manipulation of data by users, including "slicing and dicing". When OLAP forms part of the users' customary spreadsheet or graphics package, this should be transparent to the user. OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from.

Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases.

Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security. Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells. Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. Reporting facilities should present information in any way the user wants to view it. jopi The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path. Citation Codd E.F., Codd S.B., and Salley C.T. "Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate". Codd & Date, Inc 1993. < . See Also Codd's 12 Rules on Wikipedia Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a Relational Database System (RDBMS). So, some rules define a database to be the correct RDBMS. These rules were developed by Dr. Edgar F. Codd (E.F. Codd) in 1985, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS). These 13 rules are popular in RDBMS, known as Codd's 12 rules.

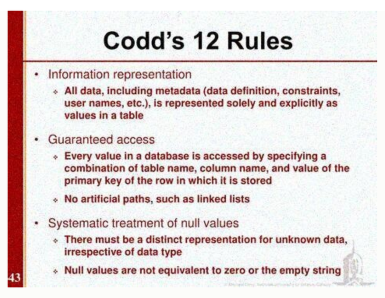


OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from. Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases. Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security. Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells. Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. Reporting facilities should present information in any way the user wants to view it. The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path. Citation Codd E.F., Codd S.B., and Salley C.T. "Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate". Codd & Date, Inc 1993.



Multi-dimensional data models enable more straightforward and intuitive manipulation of data by users, including "slicing and dicing". When OLAP forms part of the users' customary spreadsheet or graphics package, this should be transparent to the user. OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from. Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort.

The server should be capable of mapping and consolidating data between disparate databases. Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security. Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells. Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. Reporting facilities should present information in any way the user wants to view it. The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path. Citation Codd E.F., Codd S.B., and Salley C.T. "Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate". Codd & Date, Inc 1993. < .



When OLAP forms part of the users' customary spreadsheet or graphics package, this should be transparent to the user. OLAP should be part of an open systems architecture which can be embedded in any place desired by the user without adversely affecting the functionality of the host tool. The user should not be exposed to the source of the data supplied to the OLAP tool, which may be homogeneous or heterogeneous. The OLAP tool should be capable of applying its own logical structure to access heterogeneous sources of data and perform any conversions necessary to present a coherent view to the user. The tool (and not the user) should be concerned with where the physical data comes from. Performance of the OLAP tool should not suffer significantly as the number of dimensions is increased. The server component of OLAP tools should be sufficiently intelligent that the various clients can be attached with minimum effort. The server should be capable of mapping and consolidating data between disparate databases. Every data dimension should be equivalent in its structure and operational capabilities. The OLAP server's physical structure should have optimal sparse matrix handling. OLAP tools must provide concurrent retrieval and update access, integrity and security. Computational facilities must allow calculation and data manipulation across any number of data dimensions, and must not restrict any relationship between data cells. Data manipulation inherent in the consolidation path, such as drilling down or zooming out, should be accomplished via direct action on the analytical model's cells, and not require use of a menu or multiple trips across the user interface. Reporting facilities should present information in any way the user wants to view it. The number of data dimensions supported should, to all intents and purposes, be unlimited. Each generic dimensions should enable an essentially unlimited number of user-defined aggregation levels within any given consolidation path. Citation Codd E.F., Codd S.B., and Salley C.T. "Providing OLAP (On-line Analytical Processing) to User-Analysts: An IT Mandate". Codd & Date, Inc 1993. < . See Also Codd's 12 Rules on Wikipedia Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a Relational Database System (RDBMS). So, some rules define a database to be the correct RDBMS. These rules were developed by Dr. Edgar F.
Codd (E.F. Codd) in 1985, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS against his relational model, and if a database follows the rule, it is called a true relational database (RDBMS). These 13 rules are popular in RDBMS, known as Codd's 12 rules. Rule 0: The Foundation Rule The database must be in relational form. So that the system can handle the database through its relational capabilities. Rule 1: Information Rule A database contains various information, and this information must be stored in each cell of a table in the form of rows and columns. Rule 2: Guaranteed Access Rule Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name. Rule 3: Systematic Treatment of Null Values This rule defines the systematic treatment of Null values in database records. The null value has various meanings in the database, like missing the data, no value in a cell, inappropriate information, unknown data and the primary key should not be null. Rule 4: Active/Dynamic Online Catalog based on the relational model It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database. Rule 5: Comprehensive Data SubLanguage Rule The relational database supports various languages, and if we want to access the database, the language must be the explicit, linear or well-defined syntax, character strings and supports the comprehensive: data definition, view definition, data manipulation, integrity constraints, and limit transaction management operations. If the database allows access to the data without any language, it is considered a violation of the database. Rule 6: View Updating Rule All views table can be theoretically updated and must be practically updated by the database systems. Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.
Rule 8: Physical Data Independence Rule All stored data in a database or an application must be physically independent to access the database. Each data should not depend on other data or an application.
If data is updated or the physical structure of the database is changed, it will not show any effect on external applications that are accessing the data from the database. Rule 9: Logical Data Independence Rule It is similar to physical data independence. It means, if any changes occurred to the logical level (table structures), it should not affect the user's view (application). For example, suppose a table either split into two tables, or two table joins to create a single table, these changes should not be impacted on the user view application. Rule 10: Integrity Independence Rule A database must maintain integrity independence when inserting data into table's cells using the SQL query language. All entered values should not be changed or rely on any external factor or application to maintain integrity. It is also helpful in making the database-independent for each front-end application. Rule 11: Distribution Independence Rule The distribution independence rule represents a database that must work properly, even if it is stored in different locations and used by different end-users. Suppose a user accesses the database through an application; in that case, they should not be aware that another user uses particular data, and the data they always get is only located on one site. The end users can access the database, and these access data should be independent for every user to perform the SQL queries. Rule 12: Non Subversion Rule The non-submersion rule defines RDBMS as a SQL language to store and manipulate the data in the database. If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data. Next TopicSQL EXCEPT In this chapter, you will learn about Dr. Codd's OLAP rules created by his own, which, according to him, a database must obey to be regarded as a real relational database.Codd's Rules for OLAP ToolsIn 1993, Dr. E.F. Codd originated twelve rules as the basis for selecting OLAP tools. The publication of these rules was the result of research carried out on behalf of Arbor Software and has resulted in a formalized redefinition of the requirements for OLAP tools.
These rules are:Multi-dimensional conceptual view of the databaseConcept of transparencyConcept of accessibilityConsistent reporting performanceClient-server architectureGeneric dimensionalityDynamic sparse matrix handlingMulti-user supportUnrestricted cross-dimensional operationsIntuitive data manipulationFlexible reportingUnlimited dimensions and aggregation levelsLet us discuss all of these rules in brief:Multi-dimensional conceptual view: OLAP tools should allow users with a multi-dimensional model that keeps up a correspondence to users' views of the enterprise and is intuitively analytical and straightforward to use. Interestingly, this rule is given various levels of support by sellers who disagree that a multi-dimensional conceptual view of data can be delivered without multi-dimensional storage.Transparency: The OLAP technology has the underlying database and architecture, and the likely heterogeneity of input data sources that should be apparent to users. This necessity is to preserve the user's productivity and proficiency with familiar front-end environments and tools.Accessibility: The OLAP tool also lets to access data needed for the analysis from all heterogeneous enterprise data sources such as relational, non-relational, and legacy methods.Consistent reporting performance: With the number of dimensions, levels of aggregations, and the size of the database raises, users ought not to perceive any significant fall in performance. There should be no change in the way the key figures are calculated, and the system models must have to be strong enough to cope with changes to the enterprise model.Client-server architecture: The OLAP system should be proficient enough to operate efficiently in a client-server environment.
The architecture should permit optimal performance, flexibility, adaptability, scalability, and interoperability.Generic dimensionality: Every data dimension must be the same in both structure and operational capabilities, i.e., the basic structure, formulae, and reporting should not be biased towards any one dimension.Dynamic sparse matrix handling: The OLAP system should be able to cope up with the physical schema to the specific analytical model that optimizes sparse matrix handling to achieve and maintain the required level of performance.Multi-user support: The OLAP system should be able to hold up a group of users working at the same time on the same or different models of the enterprise's data.Unrestricted cross-dimensional operations: The OLAP system must be able to identify the dimensional hierarchies and automatically perform associated roll-up calculations across dimensions.Intuitive data manipulation: Slicing and cubing, consolidation (roll-up), and other manipulations can be accomplished via direct 'point-and-click' or 'drag-and-drop' actions on the cells of the cube.Flexible reporting: The capability of arranging rows, columns, and cells in a way that facilitates analysis by an intuitive visual presentation of analytical reports must exist.Unlimited dimensions and aggregation levels: Depending on business needs, an analytical model may have some dimensions, each having multiple hierarchies.Found This Useful? Share This Page!Page 2In this chapter, you will learn about the various relational algebras that are used in maintaining a database.
In particular, we concentrate on the relational algebra as defined by Codd in the year 1971 as the basis for relational languages. Informally, here you will understand the relational algebra as a (high-level) procedural language: which can be used to tell the DBMS how to build a new relation from one or more relations in the database.What is Relational Algebra?The relational algebra is a theoretical procedural query language which takes an instance of relations and does operations that work on one or more relations to describe another relation without altering the original relation(s). Thus, both the operands and the outputs are relations. So the output from one operation can turn into the input to another operation, which allows expressions to be nested in the relational algebra, just as you nest arithmetic operations. This property is called closure: relations are closed under the algebra, just as numbers are closed under arithmetic operations.The relational algebra is a relation-at-a-time (or set) language where all tuples are controlled in one statement without the use of a loop. There are several variations of syntax for relational algebra commands, and you use a common symbolic notation for the commands and present it informally.The primary operations of relational algebra are as follows:SelectProjectUnionSet differentCartesian productRenameSelect Operation (σ)It selects tuples that satisfy the given predicate from a relation.Notation − σp(r)Here σ stands for selection predicate, and r stands for relation, and p is a propositional logic formula which may use connectors like and, or, and not.σ predicate(R): This selection operation functions on a single relation R and describes a relation that contains only those tuples of R that satisfy the specified condition (predicate).Example:σteacher = "database"(Names)Output - It selects tuples from names where the teacher is 'database.'Project Operation (∏)The Projection operation works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and eliminating duplicates.Produce a list of salaries for all staff, showing only the staffNo, fName, lName, andsalary details.ΠstaffNo, fName, lName, salary(Staff)In the below-mentioned example, the Projection operation defines a relation that contains only the designated Staff attributes staffNo, fName, lName, and salary, in the specified order. The result of this operation is shown in the figure belowUnion OperationFor R ∪ S, The union of two relations, R and S, defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated. R and S must be union-compatible.For a union operation to be applied, the following rules must hold −r and s must have the same quantity of attributes.Attribute domains must be compatible.Duplicate tuples get automatically eliminated.Set differenceFor R − S The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.Example:∏ writer (Nobels) − ∏ writer (papers)Cartesian productFor R × S, the Cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.Example:σwriter = 'gauravray'(Articles X Notes)Join OperationsTypically, you want only combinations of the Cartesian product which satisfy certain situations, and so you can normally use a Join operation instead of the Cartesian product operation. The Join operation, which combines two relations to form a new relation, is one of the essential operations in the relational algebra. There are various types of Join operation, each with subtle differences, some more useful than others:Theta joinEquijoin (a particular type of Theta join)Natural joinOuter joinSemijoinRename Operation (ρ)The results of relational algebra are also relations but without any name. The rename operation provides database designers to rename the output relation. The rename-operation is denoted using a small Greek letter rho (ρ).It is written as:ρ x (E)Found This Useful? Share This Page!Page 3In this chapter, you will learn about the relational calculus and its concept about the database management system. A certain arrangement is explicitly stated in relational algebra expression, and a plan for assessing the query is implied. In the relational calculus, there is no description and depiction of how to assess a query; Instead, a relational calculus query focuses on what is to retrieve rather than how to retrieve it.What is Relational Calculus?Relational calculus is a non-procedural query language, and instead of algebra, it uses mathematical predicate calculus. The relational calculus is not the same as that of differential and integral calculus in mathematics but takes its name from a branch of symbolic logic termed as predicate calculus. When applied to databases, it is found in two forms.
These areTuple relational calculus which was originally proposed by Codd in the year 1972 andDomain relational calculus which was proposed by Lacroix and Pirotte in the year 1977In first-order logic or predicate calculus, a predicate is a truth-valued function with arguments. When we replace with values for the arguments, the function yields an expression, called a proposition, which will be either true or false.Example:For example, steps involved in listing all the employees who attend the 'Networking' Course would be:SELECT the tuples from COURSE relation with COURSENAME = 'NETWORKING'PROJECT the COURSE_ID from above resultSELECT the tuples from EMP relation with COURSE_ID resulted above.Tuple Relational CalculusIn the tuple relational calculus, you will have to find tuples for which a predicate is true. The calculus is dependent on the use of tuple variables. A tuple variable is a variable that 'ranges over' a named relation: i.e., a variable whose only permitted values are tuples of the relation.Example:For example, to specify the range of a tuple variable S as the Staff relation, we write:Staff(S)To express the query 'Find the set of all tuples S such that F(S) is true,' we can write:{S | F(S)}Here, F is called a formula (well-formed formula, or wff in mathematical logic). For example, to express the query 'Find the staffNo, fName, lName, position, sex, DOB, salary, and branchNo of all staff earning more than £10,000', we can write:{S | Staff(S) ∧ S.salary > 10000}Example:{t | TEACHER (t) and t.SALARY>20000}- It implies that it selects the tuples from the TEACHER in such a way that the resulting teacher tuples will have a salary higher than 20000. This is an example of selecting a range of values.{t | TEACHER (t) AND t.DEPT_ID = 6}- T select all the tuples of teachers' names who work under Department 8. Any tuple variable with 'For All' (?) or 'there exists' (?) condition is termed as a bound variable. In the last example, for any range of values of SALARY greater than 20000, the meaning of the condition does not alter.
Bound variables are those ranges of tuple variables whose meaning will not alter if another tuple variable replaces the tuple variable.In the second example, you have used DEPT_ID= 8, which means only for DEPT_ID = 8 display the teacher details. Such a variable is called a free variable. Any tuple variable without any 'For All' or 'there exists' condition is called Free Variable.Domain Relational CalculusIn the tuple relational calculus, you have use variables that have a series of tuples in a relation. In the domain relational calculus, you will also use variables, but in this case, the variables take their values from domains of attributes rather than tuples of relations. A domain relational calculus expression has the following general format:{d1, d2, . . . , dn | F(d1, d2, . .
. , dm)} m ≥ nwhere d1, d2, . . . , dn, . . . , dm stand for domain variables and F(d1, d2, . . . , dm) stands for a formula composed of atoms.Example:select TCHR_ID and TCHR_NAME of teachers who work for department 8, (where suppose - dept. 8 is Computer Application Department){ | ?
TEACHER Λ DEPT_ID = 10}Get the name of the department name where Karlos works:{DEPT_NAME |< DEPT_NAME > ? DEPT Λ ? DEPT_ID (? TEACHER Λ TCHR_NAME = Karlos)}It is to be noted that these queries are safe. The use domain relational calculus is restricted to safe expressions; moreover, it is equivalent to the tuple relational calculus, which in turn is similar to the relational algebra.Found This Useful? Share This Page!Page 4In the previous chapters, you have learned about the various forms of relational algebra and relational calculus and their uses with the database management system. In this chapter, you will get to know about the various forms of languages that are used to deal with the database.What are database Sub languages?A data sublanguage mainly has two parts:Data Definition Language (DDL) andData Manipulation Language (DML).The Data Definition Language is used for specifying the database schema, and the Data Manipulation Language is used for both reading and updating the database. These languages are called data sub-languages as they do not include constructs for all computational requirements.Computation purposes include conditional or iterative statements that are supported by the high-level programming languages. Many DBMSs can embed the sublanguage is a high-level programming language such as 'Fortran,' 'C,' C++, Java, or Visual Basic. Here, the high-level language is sometimes referred to as the host language as it is acting as a host for this language. To compile the embedded file, the commands in the data sub-language are first detached from the host-language program and are substituted by function calls.
The pre-processed file is then compiled and placed in an object module, which gets linked with a DBMS-specific library that is having the replaced functions and executed based on the requirement. Most data sub-languages also supply non-embedded or interactive commands which can be input directly using the terminal.Data Definition LanguageData Definition Language (DDL) statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:CREATE - used to create objects in the databaseALTER - used to alters the structure of the databaseDROP - used to delete objects from the databaseTRUNCATE - used to remove all records from a table, including all spaces allocated for the records are removedCOMMENT - used to add comments to the data dictionaryRENAME - used to rename an objectData Manipulation LanguageA language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the lists of tasks that come under DML:SELECT - It retrieves data from a databaseINSERT - It inserts data into a tableUPDATE - It updates existing data within a tableDELETE - It deletes all records from a table, the space for the records remainMERGE - UPSERT operation (insert or update)CALL - It calls a PL/SQL or Java subprogramEXPLAIN PLAN - It explains the access path to dataLOCK TABLE - It controls concurrencyData Control LanguageThere are two other forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Databases. To perform any operation in the database, such as for creating tables, sequences, or views, we need privileges. Privileges are of two types,System - creating a session, table, etc. are all types of system privilege.Object - any command or query to work on tables comes under object privilege. DCL is used to define two commands. These are:Grant - It gives user access privileges to a database.Revoke - It takes back permissions from the user.Transaction Control Language (TCL)Transaction Control statements are used to run the changes made by DML statements.
It allows statements to be grouped into logical transactions.COMMIT - It saves the work doneSAVEPOINT - It identifies a point in a transaction to which you can later roll backROLLBACK - It restores the database to original since the last COMMITSET TRANSACTION - It changes the transaction options like isolation level and what rollback segment to useFound This Useful? Share This Page!Page 5Database normalization is a database schema design technique, by which an existing schema is modified to minimize redundancy and dependency of data.Normalization split a large table into smaller tables and define relationships between them to increases the clarity in organizing data.Some Facts About Database NormalizationThe words normalization and normal form refer to the structure of a database.Normalization was developed by IBM researcher E.F. Codd In the 1970s.Normalization increases clarity in organizing data in Databases.Normalization of a Database is achieved by following a set of rules called 'forms' in creating the database.Database Normalization RulesFirst Normal Form (1NF)Each column is unique in 1NF.Example:Sample Employee table, it displays employees are working with multiple departments.EmployeeAgeDepartmentMelvin32Marketing, SalesEdward45Quality AssuranceAlex36Human ResourceEmployee table following 1NF:EmployeeAgeDepartmentMelvin32MarketingMelvin32SalesEdward45Quality AssuranceAlex36Human ResourceSecond Normal Form (2NF)The entity should be considered already in 1NF, and all attributes within the entity should depend solely on the unique identifier of the entity.Example:Sample Products table:productIDproductBrand1MonitorApple2MonitorSamsung3ScannerHP4Head phoneJBLProduct table following 2NF:Products Category table:productIDproduct1Monitor2Scanner3Head phoneBrand table:brandIDbrand1Apple2Samsung3HP4JBLProducts Brand table:pbIDproductIDbrandID111212323434Third Normal Form (3NF)The entity should be considered already in 2NF, and no column entry should be dependent on any other entry (value) other than the key for the table.If such an entity exists, move it outside into a new table.3NF is achieved, considered as the database is normalized.Boyce-Codd Normal Form (BCNF)3NF and all tables in the database should be only one primary key.Fourth Normal Form (4NF)Tables cannot have multi-valued dependencies on a Primary Key.Fifth Normal Form (5NF)A composite key shouldn't have any cyclic dependencies.Well, this is a highly simplified explanation for Database Normalization. One can study this process extensively, though. After working with databases for some time, you'll automatically create Normalized databases, as it's logical and practical.Found This Useful? Share This Page!Page 6In this chapter, you will learn about the methodology for the database design stage of the database system development lifecycle for relational databases.
The methodology is depicted as a bit by bit guide to the three main phases of database design, namely: conceptual, logical, and physical design.The primary aim of each phase is as follows:Conceptual database design - to build the conceptual representation of the database, which has the identification of the important entities, relationships, and attributes.Logical database design - to convert the conceptual representation to the logical structure of the database, which includes designing the relations.Physical database design - to decide how the logical structure is to be physically implemented (as base relations) in the target Database Management System (DBMS).Introduction to the Database Design MethodologyA structured approach that uses procedures, techniques, tools, and documentation help to support and make possible the process of design is called Design Methodology.A design methodology encapsulates various phases, each containing some stages, which guide the designer in the techniques suitable at each stage of the project. A design methodology also helps the designer to plan, manage, control, and evaluate database development and managing projects. Furthermore, it is a planned approach for analyzing and modeling a group of requirements for a database in a standardized and ordered manner.Conceptual Database DesignIn this design methodology, the process of constructing a model of the data is used in an enterprise, independent of all physical considerations. The conceptual database design phase starts with the formation of a conceptual data model of the enterprise that is entirely independent of implementation details such as the target DBMS, use of application programs, programming languages used, hardware platform, performance issues, or any other physical deliberations.Critical Success Factors in Database DesignThe following planning strategies are often critical to the success of database design:Deal with task interactively with the users as much as possible.Follow a prearranged methodology throughout the data modeling process.Make use of a data-driven approach.Incorporate structural and integrity considerations into the data models.Combine conceptualization, normalization, and transaction validation methods into the data modeling methodology.Use figures for representing as much of the data models as possible.Use a Database Design Language (DBDL) to represent additional data semantics that cannot usually be represented in a diagram.Build a data dictionary to add-on the data model diagrams and the DBDL.Be willing to repeat steps.These factors are constructed into the methodology that is presented for database design.What are the steps for Conceptual Database Design?Conceptual database design steps are:Build a conceptual data modelRecognize entity typesRecognize the relationship typesIdentify and connect attributes with entity or relationship typesDetermine attribute domainsDetermine candidate, primary, and alternate key attributesConsider the use of improved modeling concepts (optional step)Check model for redundancyValidate the conceptual model against user transactionsReview the conceptual data model with userBuilding a Conceptual Data ModelThe first step in conceptual database design is to build one (or more) conceptual data replica of the data requirements of the enterprise. A conceptual data model comprises these following elements:entity typestypes of relationshipattributes and the various attribute domainsprimary keys and alternate keysintegrity constraintsThe conceptual data model is maintained by documentation, including ER diagrams and a data dictionary, which is produced throughout the development of the model.Found This Useful? Share This Page!Page 7You have already come across the basics of what methodologies are and their stages. You have gathered the basic concept of what conceptual methodology is and how it works within the main stages of the database system development life cycle.This stage is made up of three phases:ConceptualLogical andPhysical database designIn this chapter, you will learn and understand the basic concepts of Logical Methodology, i.e., the second stage of the database development life cycle.Details on Logical MethodologyA local logical data model is used to characterize the data requirements of one or more but not all user views of a database, and a universal logical data model represents the data requirements for all user views. The final step of the logical database design phase is to reflect on how well the model can support possible future developments for the database system.Logical Database Design Methodology for the Relational ModelThe objective of logical database design methodology is to interpret the conceptual data model into a logical data model and then authorize this model to check whether it is structurally correct and able to support the required transactions or not.In this step of the database development life cycle, the main purpose is to translate the conceptual data model created in conceptual methodology (of the previous chapter) into a logical data model of the data requirements of the enterprise. This objective can be achieved by following the activities given below:Obtain the relations for the logical data modelAuthorize those relations using normalizationValidate those relations against user transactionsCheck integrity control and its limitationEvaluate the logical data model with userCombine logical data models into the global model (This step is an optional one)Check for future growth and developmentThe structure of the relational schema is authorized using normalization. It then makes sure to ensure that the relations are capable of supporting the transactions given in the users' requirements specification. You can then check those all-important integrity constraints that are characterized by the logical data model. At this stage, the logical data model is authorized by the users to ensure that they consider the model to be a true demonstration of the data requirements for the enterprise.Derive Relations for Logical Data ModelThe relationship that an entity has with other entities is characterized using the primary key or foreign key's concept. In deciding where to post the foreign key attribute(s), firstly, you must have to identify the 'parent' and 'child' entities that are involved in that relationship.
The parent entity refers to the entity that posts a copy of its primary key into the relation that represents the child entity to act as the foreign key. You can describe how relations are obtained for the following structures that may occur in a conceptual data model:strong entity typesweak entity typesone-to-many (1:*) binary relationship typesone-to-one (1:1) binary relationship typesone-to-one (1:1) recursive relationship typessuperclass/subclass relationship typesmany-to-many (*:*) binary relationship typescomplex relationship typesmulti-valued attributesValidate Relations Using NormalizationIn the previous step, you have derived a set of relations from signifying the conceptual data model created in the earlier step. Now, in the next step, you have to validate the groupings of attributes in each relation using the rules of normalization. The purpose of normalization is to ensure that the position of relations has a minimal and yet sufficient number of attributes necessary to support the data requirements of the enterprise.Validate Relations Against User TransactionsThe primary purpose of this step is to validate the logical data model to make certain that the model supports the required transactions, as the users' requirements specification. By using the relations, the primary key / foreign key links within the relations, the ER diagram, and the data dictionary, you can attempt to perform the operations manually. If you can resolve all transactions in this way, you can validate the logical data model against the transactions.Found This Useful? Share This Page!Page 8This physical methodology is the third and final phase of the database design methodology.
Here, the designer must decide how to translate the logical database design (i.e., the entities, attributes, relationships, and constraints) into a physical database design, which can ultimately be implemented using the target DBMS. As the various parts of physical database design are highly reliant on the target DBMS, there may be more than one method of implementing any given portion of the database. Consequently, to do this work appropriately, the designers must be fully aware of the functionality of the target DBMS.
They must recognize the advantages and disadvantages of each alternative approach for a particular accomplishment. For some systems, the designer may also need to select a suitable storage space/strategy that can take account of intended database usage.What is Physical Database Design?It is the process of making a description of the execution of the database on secondary storage, which describes the base relations, file organizations as well as indexes used to gain efficient access to the data and any associated integrity constraints and security measures.Comparison of Logical and Physical Database DesignIn designing and presenting a database design methodology, you have to divide the design process into three main stages or steps, also known as the Database development life cycle. These steps or stages are:ConceptualLogical andPhysical database design (as studied in the earlier chapter)The phase before the physical design is the logical database design, which is largely independent of implementation details, such as the specific functionality of the target DBMS and application programs, but is reliant on the target data model. The outcome of this process is a logical data model that consists of an ER/relation diagram, relational schema, and supporting documents that depict this model, such as a data dictionary.Logical database designs are concerned with the "what," and in contrast, physical database design is concerned with the "how." It requires diverse skills that are often found in different people. In particular, the physical database designer must know how the computer system hosts the DBMS and how it operates and must be fully conscious of the working of the target DBMS.Steps Required for Implementing Physical MethodologyThe steps of the physical database design methodology are as follows:Transform the logical data model for target DBMSDesign base relationsDesign representation of derived dataDesign general constraintsDesign file organizations and indexesAnalyze transactionsChoose file organizationsChoose indexesEstimate disk space requirementsDesign user viewsDesign security mechanismsConsider the introduction of controlled redundancyMonitor and tune the operational systemCommon Characteristics of a Physical Data ModelIt typically illustrates data requirements for a single project or application. Sometimes even a part of an applicationMay be incorporated into other physical data models by means of a repository of shared entitiesIt typically includes 10-1000 tables; although these numbers are highly variable, depending on the scope of the data modelIt has the relationships between tables that address cardinality and nullability (optionality) of the relationshipsDesigned and developed to be reliant on a specific version of a DBMS, storage location of data or technologyDatabase columns will have data types with accurate precisions and lengths assigned to them.
Columns will have nullability (optional) assignedTables and columns will have specific definitionsFound This Useful? Share This Page!Page 9Earlier, you have learned about the functions that a Database Management System (DBMS) should offer database users. Among these three closely related functions are intended to ensure that the database is reliable and remains in a steady-state, namely transaction support, concurrency control, and recovery services. This reliability and consistency must be maintained in the presence of failures of both hardware and software components and when several users are accessing the database.
In this chapter, you will be concentrating on a transaction and ACID property of DBMS.What is Transaction?A transaction is an action or series of actions that are being performed by a single user or application program, which reads or updates the contents of the database.A transaction can be defined as a logical unit of work on the database. This may be an entire program, a piece of a program, or a single command (like the SQL commands such as INSERT or UPDATE), and it may engage in any number of operations on the database. In the database context, the execution of an application program can be thought of as one or more transactions with non-database processing taking place in between.Example of a Transaction in DBMSA simple example of a transaction will be dealing with the bank accounts of two users, let say Karlos and Ray. A simple transaction of moving an amount of 5000 from Karlos to Ray engages many low-level jobs.
As the amount of Rs. 5000 gets transferred from the Karlos's account to Ray's account, a series of tasks gets performed in the background of the screen.This straightforward and small transaction includes several steps: decrease Karlos's bank account from 5000:Open_Acc (Karlos) OldBal = Karlos.bal NewBal = OldBal - 5000 Ram.bal = NewBal CloseAccount(Karlos)You can say, the transaction involves many tasks, such as opening the account of Karlos, reading the old balance, decreasing the specific amount of 5000 from that account, saving new balance to an account of Karlos, and finally closing the transaction session.For adding amount 5000 in Ray's account, the same sort of tasks needs to be done:OpenAccount(Ray) Old_Bal = Ray.bal NewBal = OldBal + 1000 Ahmed.bal = NewBal CloseAccount(B)Properties of TransactionThere are properties that all transactions should follow and possess. The four basic are in combination termed as ACID properties.
ACID properties and its concepts of a transaction are put forwarded by Haerder and Reuter in the year 1983. The ACID has a full form and is as follows:Atomicity: The 'all or nothing' property.
A transaction is an indivisible entity that is either performed in its entirety or will not get performed at all. This is the responsibility or duty of the recovery subsystem of the DBMS to ensure atomicity.Consistency: A transaction must alter the database from one steady-state to another steady state. This is the responsibility of both the DBMS and the application developers to make certain consistency.
The DBMS can ensure consistency by putting into effect all the constraints that have been mainly on the database schema such as integrity and enterprise constraints.Isolation: Transactions that are executing independently of one another is the primary concept followed by isolation. In other words, the frictional effects of incomplete transactions should not be visible or come into notice to other transactions going on simultaneously. It is the responsibility of the concurrency control sub-system to ensure adapting the isolation.Durability: The effects of an accomplished transaction are permanently recorded in the database and must not get lost or vanished due to subsequent failure. So this becomes the responsibility of the recovery sub-system to ensure durability.Found This Useful? Share This Page!Page 10When the relational model was launched for the first time, one of the chief criticisms often cited was the inadequate presentation of queries. Since then, a significant amount of research has been committed to developing highly proficient algorithms for processing and dealing with queries. There are a lot of ways to do a complex query that can be performed, and one of the targets of query processing is to decide which one is the most cost-effective. In the first-generation network and hierarchical database systems, the low-level procedural query language is generally implanted in a high-level programming language such as COBOL, and it is the job of the programmer's to select the most appropriate execution strategy. In contrast, with declarative languages such as SQL, the user identifies what data is required rather than how it is to be retrieved.In this chapter, you will be given a general idea of query processing and examine the main segments of this activity. Here you will examine the first phase of query processing, namely query decomposition, which transforms a high-level query into a relational algebra query and ensures that it is syntactically and semantically correct.Overview of Query ProcessingThis query processing activity involved in parsing, validating, optimizing, and executing a query.
The target of query processing is to change a query written in a high-level language, (usually SQL) into a correct and efficient execution strategy expressed in a low-level language (using the relational algebra) and to perform the strategy to retrieve the required data. An important aspect of query processing is query optimization. The activity of choosing an efficient execution strategy for processing a query is known as Query optimization. As there are many correspondent transformations of the same high-level query, the main aim of optimizing a query is to choose the one that minimizes resource usage. Generally, you will try reducing the total execution time of the query, which is the total of the execution times of all individual operations that make up the query.Both methods of query optimization rely on database statistics to assess the different available options properly. The accuracy and currency of these statistics have a significant bearing on the efficiency of the execution strategy chosen.Comparison of Different Processing StrategiesFind all DBAs who work at a NewDelhi branch.You can write this query in SQL as:SELECT *FROM Staff s, Branch bWHERE s.branchNo = b.branchNo AND(s.position = 'DBA' AND b.city = 'NewDelhi');The 3 equivalent relational algebra queries corresponding to the above SQL statement are:σ(position='DBA') ∧ (city='NewDelhi') ∧ (Staff.branchNo=Branch.branchNo)(Staff × Branch)σ(position='DBA') ∧ (city=' NewDelhi')(Staff branchNo=Branch.branchNo Branch)(σposition='DBA'(Staff)) Staff.branchNo=Branch.branchNo (σcity=' NewDelhi'(Branch))What is Query Decomposition in DBMS?Query decomposition is the first phase of query processing. The primary targets of query decomposition are to transform a high-level query into a relational algebra query and to check that the query is syntactically and semantically correct. The typical stages of query decomposition are analysis, normalization, semantic analysis, simplification, and query restructuring.Found This Useful? Share This Page!Page 11Database technology has transformed the database users from a paradigm of data processing where each application described and upheld its data, to one in which data is defined and managed centrally. During recent times, you have seen the fast and ever-growing developments in network and data communication technology embodied by the Internet, mobile and wireless computing, and grid computing. With the combination of these two technologies, distributed database technology may revolutionize the mode of working from centralized to decentralized. This collective technology is one of the significant reasons for developments in the database systems world. Here we will be dealing with the different issues of the distributed management system of the database. This allows users to access not only the data but also data stored at remote sites. We will be discussing the design and concepts of the distributed system.More on the Concept of DistributionA primary motivation behind the development of database systems is the need to integrate the equipped data of an organization and to provide restricted access to the data. Although integration and controlled access may involve centralization, this is not the intention. The development of computer networks promotes a decentralized mode of work.Basic Terminology used in Distributed SystemDistributed DatabaseA logically interconnected set of shared data (and a description of this data) physically scattered over a computer network.This software system allows the management of the distributed database and makes the distribution transparent to users.A Distributed Database Management System (DDBMS) contains a single logical database that is divided into a number of fragments. Every fragment gets stored on one or more computers under the control of a separate DBMS, with the computers connected by a communications network.
Each position is capable of independently process every user's requests that require access to local data (i.e., each position of the distributed system has some basic degree of local autonomy) and is also able to process data stored on other computers within the network.
Users access the distributed database via applications that are classified as those which do not need data from other sites (local applications); and also those that do need data from other sites (global applications). You will be requiring a DDBMS to have at least one global application. A DDBMS, therefore, has the following characteristics:a collection of logically related shared datan the data is split into a number of fragmentsfragments may be replicatedfragments/replicas are allocated to sitesthe sites are linked by a communications networkthe data at each site is under the control of a DBMSthe DBMS at each site can handle local applications, autonomouslyeach DBMS participates in at least one global applicationParallel DBMSIt is a DBMS that runs across multiple processors and disks that is designed to execute operations in parallel whenever achievable, in order to improve the performance of a database. Parallel DBMSs are again dependent on the principle that single-processor systems can no longer meet the growing necessities for cost-effective scalability, reliability, and performance. A powerful and financially attractive choice for a single-processor-driven DBMS is a parallel DBMS driven by multiple processors (i.e., the concept of multi-programming). Parallel DBMSs link multiple, smaller machines together into a single set to achieve the same throughput as an individual, larger machine, and often provides greater scalability and reliability than single-processor DBMSs. The three main parts for parallel DBMSs are:shared memoryshared diskshared nothingAdvantages and Disadvantages of DDBMSOrganization's choice: Many organizations are distributed over several locations. It is natural for databases used in such an application to be spread over these locations.Improved availability: A failure at one site of a DDBMS, or a failure of a communication link making some sites unreachable, does not make the entire system inoperable. Distributed DBMSs are designed to carry on the function despite such failures.Improved performance: With the concept of parallelism of distributed DBMSs, the speed of database access may be better than that achievable from a remote centralized database.Found This Useful? Share This Page!Page 12 Page 13You might have noticed the ever-increasing demands on mobile computing for providing the types of support required by a growing number of mobile workers and its technology. Such individuals require working as if they are in the office, but in reality, they are working from remote corners of different locations of any particular area, including homes, clients' premises, or simply while routing to remote locations. The 'office' may come with a remote worker in the form of a laptop or desktop, PDA (Personal Digital Assistant), or other device accessing the Internet. With the rapid growth of cellular technology, wireless medium, and satellite communications, it will soon be achievable for mobile users to access any data from anywhere at any time. However, business etiquette, practicalities, security, and costs may still bound communication such that it is not achievable for establishing online connections for as long as users want, whenever they want. Mobile databases offer a solution to some of these restrictions or problems.What is Mobile Database?Mobile Database is a database that is transportable, portable, and physically separate or detached from the corporate database server but has the capability to communicate with those servers from remote sites allowing the sharing of various kinds of data.With mobile databases, users have access to corporate data on their laptop, PDA, or other Internet access device that is required for applications at remote sites.The components of a mobile database environment include:Corporate database server and DBMS that deals with and stores the corporate data and provides corporate applicationsRemote database and DBMS usually manages and stores the mobile data and provides mobile applicationsmobile database platform that includes a laptop, PDA, or other Internet access devicesTwo-way communication links between corporate and mobile DBMS.Based on the particular necessities of mobile applications, in many of the cases, the user might use a mobile device may and log on to any corporate database server and work with data there. In contrast, in others, the user may download data and work with it on a mobile device or upload data captured at the remote site to the corporate database. The communication between the corporate and mobile databases is usually discontinuous and is typically established or gets its connection for a short duration of time at irregular intervals. Although unusual, some applications require direct communication between mobile databases. The two main issues associated with mobile databases are the management of the mobile database and the communication between the mobile and corporate databases. In the following section, we identify the requirements of mobile DBMSs.The additional functionality required for mobile DBMSs includes the capability to:communicate with the centralized or primary database server through modesrepeat those data on the centralized database server and mobile devicecoordinate data on the centralized database server and mobile devicecapture data from a range of sources such as the Internetdeal with those data on the mobile deviceanalyze those data on a mobile devicecreate customized and personalized mobile applicationsFound This Useful? Share This Page!