Relational database

From Wikipedia, the free encyclopedia

A **relational database** is a digital database whose organization is based on the relational model of data, as proposed by E.F. Codd in 1970.[1] T

The various software systems used to maintain relational databases are known as Relational Database Management Systems (RDBMS).

Virtually all relational database systems use SQL (Structured Query Language) as the language for querying and maintaining the database.

Contents

Relational model

Main article: Relational model

This model organizes data into one or more tables (or "relations") of columns and rows, with a unique key identifying each row. Rows are also called records or tuples.[2] Generally, each table/relation represents one "entity type" (such as customer or product). The rows represent instances of that type of entity (such as "Lee" or "iPhone 6") and the columns representing values attributed to that instance (such as address or price).

Keys

Each row in a table has its own unique key. Rows in a table can be linked to rows in other tables by adding a column for the unique key of the linked row (such columns are known as foreign keys). Codd showed that data relationships of arbitrary complexity can be represented using this simple set of concepts.

Part of this processing involves consistently being able to select or modify one and only one row in a table. Therefore, most physical implementations have a system-assigned, unique primary key for each table. When a new row is written to the table, the system generates and writes the new, unique value for the primary key (PK); this is the key that the system uses primarily for accessing the table. System performance is optimized for PKs. Other, more natural keys may also be identified and defined as alternate keys (AK). Often several columns may be needed to form an AK (this is one reason why a single integer column is usually made the PK). Both PKs and AKs have the ability to uniquely identify one row within a table. Additional technology may be applied that will significantly assure a unique ID across the world, a globally unique identifier; these are used when there are broader system requirements.

The primary keys within a database are used to define the relationships among the tables. When a PK migrates to another table, it becomes a foreign key in the other table. When each cell can contain only one value and the PK migrates into a regular entity table, this design pattern can represent either a one-to-one, or a one-to-many relationship. Most relational database designs resolve many-to-many relationships by creating an additional table that contains the PKs from both of the other entity tables—the relationship becomes an entity; the resolution table is then named appropriately and is often assigned its own PK while the two FKs are combined to form an AK. The migration of PKs to other tables is the second major reason why system-assigned integers are used normally as PKs; there usually is neither efficiency nor clarity in migrating a bunch of other types of columns.

Relationships

Relationships exist both among the tables. These relationships take three logical forms: one-to-one, one-to-many, or many-to-many. Most relational databases are designed so that each column in each row holds only a single value (values are "atomic").

Transactions

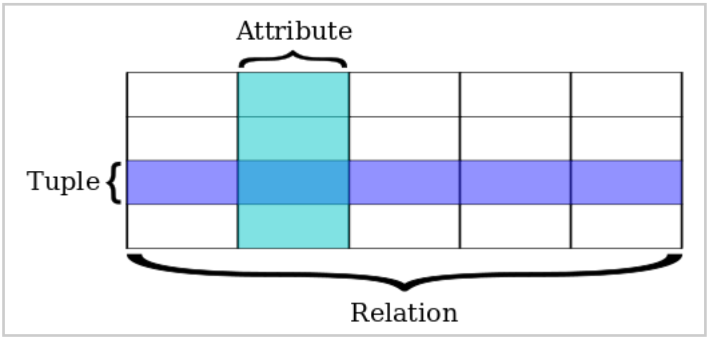
In order for a database management system (DBMS) to operate efficiently and accurately, it must have ACID transactions.[3][4][5]

Stored procedures

Most of the programming within a RDBMS is accomplished using stored procedures (SPs). Often procedures can be used to greatly reduce the amount of information transferred within and outside of a system. For increased security, the system design may also grant access to only the stored procedures and not directly to the tables. Fundamental stored procedures contain the logic needed to insert new data and update existing data. More complex procedures may be written to implement additional rules and logic related to processing or selecting the data.

Terminology

The relational database was first defined in June 1970 by Edgar Codd, of IBM's San Jose Research Laboratory.[1] Codd's view of what qualifies as an RDBMS is summarized in Codd's 12 rules. A relational database has become the predominant type of database. Other models besides the *relational model* include the hierarchical database model and the network model.



Relational database terminology.

The table below summarizes some of the most important relational database terms and the corresponding SQL term:

SQL term	Relational database term	Description
Row	***Tuple*** or ***record***	A data set representing a single item
Column	***Attribute*** or ***field***	A labeled element of a tuple, e.g. "Address" or "Date of birth"
Table	***Relation*** or ***Base relvar***	A set of tuples sharing the same attributes; a set of columns and rows
View or ***result set***	***Derived relvar***	Any set of tuples; a data report from the RDBMS in response to a query

Relations or tables

Main articles: Relation (database) and Table (database)

A *relation* is defined as a set of tuples that have the same attributes. A tuple usually represents an object and information about that object. Objects are typically physical objects or concepts. A relation is usually described as a table, which is organized into rows and columns. All the data referenced by an attribute are in the same domain and conform to the same constraints.

The relational model specifies that the tuples of a relation have no specific order and that the tuples, in turn, impose no order on the attributes. Applications access data by specifying queries, which use operations such as *select* to identify tuples, *project* to identify attributes, and *join* to combine relations. Relations can be modified using the *insert*, *delete*, and *update* operators. New tuples can supply explicit values or be derived from a query. Similarly, queries identify tuples for updating or deleting.

Tuples by definition are unique. If the tuple contains a candidate or primary key then obviously it is unique; however, a primary key need not be defined for a row or record to be a tuple. The definition of a tuple requires that it be unique, but does not require a primary key to be defined. Because a tuple is unique, its attributes by definition constitute a superkey.

Base and derived relations

Main articles: Relvar and View (database)

In a relational database, all data are stored and accessed via relations. Relations that store data are called "base relations", and in implementations are called "tables". Other relations do not store data, but are computed by applying relational operations to other relations. These relations are sometimes called "derived relations". In implementations these are called "views" or "queries". Derived relations are convenient in that they act as a single relation, even though they may grab information from several relations. Also, derived relations can be used as an abstraction layer.

Domain

Main article: data domain

A domain describes the set of possible values for a given attribute, and can be considered a constraint on the value of the attribute. Mathematically, attaching a domain to an attribute means that any value for the attribute must be an element of the specified set. The character string "*ABC*", for instance, is not in the integer domain, but the integer value *123* is. Another example of domain describes the possible values for the field "Gender" as ("Male,"Female"). So, the field "Gender" will not accept input values like (0.1) or (M,F).

Constraints

Constraints make it possible to further restrict the domain of an attribute. For instance, a constraint can restrict a given integer attribute to values between 1 and 10. Constraints provide one method of implementing business rules in the database. SQL implements constraint functionality in the form of check constraints. Constraints restrict the data that can be stored in relations. These are usually defined using expressions that result in a boolean value, indicating whether or not the data satisfies the constraint. Constraints can apply to single attributes, to a tuple (restricting combinations of attributes) or to an entire relation. Since every attribute has an associated domain, there are constraints (**domain constraints**). The two principal rules for the relational model are known as **entity integrity** and **referential integrity**.

Primary key

Main article: Unique key

A primary key uniquely specifies a tuple within a table. In order for an attribute to be a good primary key it must not repeat. While natural attributes (attributes used to describe the data being entered) are sometimes good primary keys, surrogate keys are often used instead. A surrogate key is an artificial attribute assigned to an object which uniquely identifies it (for instance, in a table of information about students at a school they might all be assigned a student ID in order to differentiate them). The surrogate key has no intrinsic (inherent) meaning, but rather is useful through its ability to uniquely identify a tuple. Another common occurrence, especially in regard to N:M cardinality is the composite key. A composite key is a key made up of two or more attributes within a table that (together) uniquely identify a record. (For example, in a database relating students, teachers, and classes. Classes *could* be uniquely identified by a composite key of their room number and time slot, since no other class could have exactly the same combination of attributes. In fact, use of a composite key such as this can be a form of data verification, albeit a weak one.

Foreign key

Main article: Foreign key

A foreign key is a field in a relational table that matches the primary key column of another table. The foreign key can be used to cross-reference tables. Foreign keys need not have unique values in the referencing relation. Foreign keys effectively use the values of attributes in the referenced relation to restrict the domain of one or more attributes in the referencing relation. A foreign key could be described formally as: "For all tuples in the referencing relation projected over the referencing attributes, there must exist a tuple in the referenced relation projected over those same attributes such that the values in each of the referencing attributes match the corresponding values in the referenced attributes."

Stored procedures

Main article: Stored procedure

A stored procedure is executable code that is associated with, and generally stored in, the database. Stored procedures usually collect and customize common operations, like inserting a tuple into a relation, gathering statistical information about usage patterns, or encapsulating complex business logic and calculations. Frequently they are used as an application programming interface (API) for security or simplicity. Implementations of stored procedures on SQL RDBMSs often allow developers to take advantage of procedural extensions (often vendor-specific) to the standard declarative SQL syntax. Stored procedures are not part of the relational database model, but all commercial implementations include them.

Index

Main article: Index (database)

An index is one way of providing quicker access to data. Indices can be created on any combination of attributes on a relation. Queries that filter using those attributes can find matching tuples randomly using the index, without having to check each tuple in turn. This is analogous to using the index of a book to go directly to the page on which the information you are looking for is found, so that you do not have to read the entire book to find what you are looking for. Relational databases typically supply multiple indexing techniques, each of which is optimal for some combination of data distribution, relation size, and typical access pattern. Indices are usually implemented via B+ trees, R-trees, and bitmaps. Indices are usually not considered part of the database, as they are considered an implementation detail, though indices are usually maintained by the same group that maintains the other parts of the database. It should be noted that use of efficient indexes on both primary and foreign keys can dramatically improve query performance. This is because B-tree indexes result in query times proportional to log(n) where n is the number of rows in a table and hash indexes result in constant time queries (no size dependency as long as the relevant part of the index fits into memory).

Relational operations

Main article: Relational algebra

Queries made against the relational database, and the derived relvars in the database are expressed in a relational calculus or a relational algebra. In his original relational algebra, Codd introduced eight relational operators in two groups of four operators each. The first four operators were based on the traditional mathematical set operations:

- The union operator combines the tuples of two relations and removes all duplicate tuples from the result. The relational union operator is equivalent to the SQL UNION operator.
- The intersection operator produces the set of tuples that two relations share in common. Intersection is implemented in SQL in the form of the INTERSECT operator.
- The difference operator acts on two relations and produces the set of tuples from the first relation that do not exist in the second relation. Difference is implemented in SQL in the form of the EXCEPT or MINUS operator.
- The cartesian product of two relations is a join that is not restricted by any criteria, resulting in every tuple of the first relation being matched with every tuple of the second relation. The cartesian product is implemented in SQL as the CROSS JOIN operator.

The remaining operators proposed by Codd involve special operations specific to relational databases:

- The selection, or restriction, operation retrieves tuples from a relation, limiting the results to only those that meet a specific criterion, i.e. a subset in terms of set theory. The SQL equivalent of selection is the SELECT query statement with a WHERE clause.
- The projection operation extracts only the specified attributes from a tuple or set of tuples.
- The join operation defined for relational databases is often referred to as a natural join. In this type of join, two relations are connected by their common attributes. MySQL's approximation of a natural join is the INNER JOIN operator. In SQL, an INNER JOIN prevents a cartesian product from occurring when there are two tables in a query. For each table added to a SQL Query, one additional INNER JOIN is added to prevent a cartesian product. Thus, for N tables in a SQL query, there must be N-1 INNER JOINS to prevent a cartesian product.
- The relational division operation is a slightly more complex operation, which involves essentially using the tuples of one relation (the dividend) to partition a second relation (the divisor). The relational division operator is effectively the opposite of the cartesian product operator (hence the name).

Other operators have been introduced or proposed since Codd's introduction of the original eight including relational comparison operators and extensions that offer support for nesting and hierarchical data, among others.

Normalization

Main article: Database normalization

Normalization was first proposed by Codd as an integral part of the relational model. It encompasses a set of procedures designed to eliminate nonsimple domains (non-atomic values) and the redundancy (duplication) of data, which in turn prevents data manipulation anomalies and loss of data integrity. The most common forms of normalization applied to databases are called the normal forms.

Distributed relational databases

Distributed Relational Database Architecture (DRDA) was designed by a work group within IBM in the period 1988 to 1994. DRDA enables network connected relational databases to cooperate to fulfill SQL requests. [6] [7] The messages, protocols, and structural components of DRDA are defined by the Distributed Data Management Architecture.

Watermarking of relational databases

Digital watermarking for relational databases emerged as a candidate solution to provide copyright protection, tamper detection, traitor tracing, and maintaining integrity of relational data. Many watermarking techniques have been proposed in the literature to address these purposes. For a comprehensive survey on the current state-of-the-art and a classification according to their intent, the way they express the watermark, the cover type, the granularity level, and their verifiability, see .[8]

References

1. Codd, E.F. (1970). "A Relational Model of Data for Large Shared Data Banks". *Communications of the ACM* **13** (6): 377–387. doi:10.1145/362384.362685.
2. "A Relational Database Overview". *oracle.com*.
3. "Gray to be Honored With A. M. Turing Award This Spring". Microsoft PressPass. 1998-11-23. Archived from the original on 6 February 2009. Retrieved 2009-01-16.
4. Gray, Jim (September 1981). "The Transaction Concept: Virtues and Limitations" (PDF). *Proceedings of the 7th International Conference on Very Large Databases*. 19333 Vallco Parkway, Cupertino CA 95014: Tandem Computers. pp. 144–154. Retrieved 2006-11-09.
5. Gray, Jim, and Reuter, Andreas, *Distributed Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1993. ISBN 1-55860-190-2.
6. Reinsch, R. (1988). "Distributed database for SAA". *IBM Systems Journal* **27** (3): 362–389. doi:10.1147/sj.273.0362.
7. *Distributed Relational Database Architecture Reference*. IBM Corp. SC26-4651-0. 1990.
8. Halder R., Pal S., Cortesi A. (2010). "Watermarking Techniques for Relational Databases: Survey, Classification and Comparison". *Journal of Universal Computer Science* **16** (21): 3164–3190.