RDBMS

How RDBMS works

As mentioned before, an RDBMS will store data in the form of a table. Each system will have varying numbers of tables with each table possessing its own unique primary key. The primary key is then used to identify each table.

Within the table are rows and columns. The rows are known as records or horizontal entities; they contain the information for the individual entry. The columns are known as vertical entities and possess information about the specific field.

Before creating these tables, the RDBMS must check the following constraints:

- Primary keys -- this identifies each row in the table. One table can only contain one primary key. The key must be unique and without null values.
- Foreign keys -- this is used to link two tables. The foreign key is kept in one table and refers to the primary key associated with another table.
- Not null -- this ensures that every column does not have a null value, such as an empty cell.
- Check -- this confirms that each entry in a column or row satisfies a precise condition and that every column holds unique data.
- Data integrity -- the integrity of the data must be confirmed before the data is created.

Assuring the integrity of data includes several specific tests, including entity, domain, referential and user-defined integrity. Entity integrity confirms that the rows are not duplicated in the table. Domain integrity makes sure that data is entered into the table based on specific conditions, such as file format or range of values. Referential integrity ensures that any row that is re-linked to a different table cannot be deleted. Finally, user-defined integrity confirms that the table will satisfy all user-defined conditions.

Advantages of relational database management system

The use of an RDBMS can be beneficial to most organizations; the systematic view of raw data helps companies better understand and execute the information while enhancing the decision-making process. The use of tables to store data also improves the security of information stored in the databases. Users are able to customize access and set barriers to limit the content that is made available. This feature makes the RDBMS particularly useful to companies in which the manager decides what data is provided to employees and customers.

Furthermore, RDBMSes make it easy to add new data to the system or alter existing tables while ensuring consistency with the previously available content.

Other advantages of the RDBMS include:

- Flexibility -- updating data is more efficient since the changes only need to be made in one place.

- Maintenance -- database administrators can easily maintain, control and update data in the database. Backups also become easier since automation tools included in the RDBMS automate these tasks.
- Data structure -- the table format used in RDBMSes is easy to understand and provides an organized and structural manner through which entries are matched by firing queries.

On the other hand, relational database management systems do not come without their disadvantages. For example, in order to implement an RDBMS, special software must be purchased. This introduces an additional cost for execution. Once the software is obtained, the setup process can be tedious since it requires millions of lines of content to be transferred into the RDBMS tables. This process may require the additional help of a programmer or a team of data entry specialists. Special attention must be paid to the data during entry to ensure sensitive information is not placed into the wrong hands.

Some other drawbacks of the RDBMS include the character limit placed on certain fields in the tables and the inability to fully understand new forms of data -- such as complex numbers, designs and images.

Furthermore, while isolated databases can be created using an RDBMS, the process requires large chunks of information to be separated from each other. Connecting these large amounts of data to form the isolated database can be very complicated.

Uses of RDBMS

Relational database management systems are frequently used in disciplines such as manufacturing, human resources and banking. The system is also useful for airlines that need to store ticket service and passenger documentation information as well as universities maintaining student databases.

Some examples of specific systems that use RDBMS include IBM, Oracle, MySQL, Microsoft SQLServer and PostgreSQL.

RDBMS product history

Many vying relational database management systems arose as news spread in the early 1970s of the relational data model. This and related methods were originally theorized by IBM researcher E.F. Codd, who proposed a database schema, or logical organization, that was not directly associated with physical organization, as was common at the time.

Codd's work was based around a concept of data normalization, which saved file space on storage disk drives at a time when such machinery could be prohibitively expensive for businesses.

File systems and database management systems preceded what could be called the RDBMS era. Such systems ran primarily on mainframe computers. While RDBMSes also ran on mainframes -- IBM's DB2 being a pointed example -- much of their ascendance in the enterprise was in UNIX midrange computer deployments. The RDBMS was a linchpin in the distributed architecture of client-server computing, which connected pools of stand-alone personal computers to file and database servers.

Numerous RDBMSes arose along with the use of client-server computing. Among the competitors were Oracle, Ingres, Informix, Sybase, Unify, Progress and others. Over time, three RDBMSes came to

dominate in commercial implementations. Oracle, IBM's DB2 and Microsoft's SQL Server, which was based on a design originally licensed from Sybase, found considerable favor throughout the client-server computing era, despite repeated challenges by competing technologies.

As the 20th century drew to an end, lower-cost, open source versions of RDBMSes began to find use, particularly in web applications.

Eventually, as distributed computing took greater hold, and as cloud architecture became more prominently employed, RDBMSes met competition in the form of NoSQL systems. Such systems were often specifically designed for massive distribution and high scalability in the cloud, sometimes forgoing SQL-style full consistency for so-called *eventual consistency* of data. But, even in the most diverse and complex cloud systems, the need for some guaranteed data consistency requires RDBMSes to appear in some way, shape or form. Moreover, versions of RDBMSes have been significantly restructured for cloud parallelization and replication.

A comparison between the three RDBMS

Name	Microsoft SQL Server X	MySQL X	PostgreSQL X
Description	Microsofts flagship relational DBMS	Widely used open source RDBMS	Widely used open source RDBMS
Primary database model	Relational DBMS	Relational DBMS	Relational DBMS
Secondary database models	Document store Graph DBMS Spatial DBMS	Document store Spatial DBMS	Document store Spatial DBMS DB-Engines Ranking Trend Chart
Website	www.microsoft.com/en-us/sql-server	www.mysql.com	www.postgresql.org
Technical documentation	docs.microsoft.com/en-US/sql/sql-server	dev.mysql.com/doc	www.postgresql.org/docs
Developer	Microsoft	Oracle	PostgreSQL Global Development Group
Initial release	1989	1995	1989
Current release	SQL Server 2019, November 2019	8.0.25, May 2021	13.3, May 2021
License	commercial	Open Source	Open Source
Cloud-based only	no	no	no

DBaaS offerings (sponsored links)		ScaleGrid for MySQL: Fully managed MySQL hosting on AWS, Azure and DigitalOcean with high availability and SSH access on the #1 multi-cloud DBaaS.	ScaleGrid for PostgreSQL: Fully managed PostgreSQL hosting on AWS, Azure and DigitalOcean with high availability and SSH access on the #1 multi-cloud DBaaS.
Implementation language	C++	C and C++	C
Server operating systems	Linux Windows	FreeBSD Linux OS X Solaris Windows	FreeBSD HP-UX Linux NetBSD OpenBSD OS X Solaris Unix Windows
Data scheme	yes	yes	yes
Typing	yes	yes	yes
XML support	yes	yes	yes
Secondary indexes	yes	yes	yes
SQL	yes	yes	yes
APIs and other access methods	ADO.NET JDBC ODBC OLE DB Tabular Data Stream (TDS)	ADO.NET JDBC ODBC Proprietary native API	ADO.NET JDBC native C library ODBC streaming API for large objects
Supported programming languages	C# C++ Delphi Go Java JavaScript (Node.js) PHP Python R Ruby Visual Basic	Ada C C# C++ D Delphi Eiffel Erlang Haskell Java JavaScript (Node.js) Objective-C OCaml Perl PHP Python Ruby	.Net C C++ Delphi Java JavaScript (Node.js) Perl PHP Python Tcl

		Scheme Tcl	
Server-side scripts	Transact SQL, .NET languages, R, Python and (with SQL Server 2019) Java	yes	user defined functions
Triggers	yes	yes	yes
Partitioning methods	tables can be distributed across several files (horizontal partitioning); sharding through federation	horizontal partitioning, sharding with MySQL Cluster or MySQL Fabric	partitioning by range, list and (since PostgreSQL 11) by hash
Replication methods	yes, but depending on the SQL-Server Edition	Multi-source replication Source-replica replication	Source-replica replication
MapReduce	no	no	no
Consistency concepts	Immediate Consistency	Immediate Consistency	Immediate Consistency
Foreign keys	yes	yes	yes
Transaction concepts	ACID	ACID	ACID
Concurrency	yes	yes	yes
Durability	yes	yes	yes
In-memory capabilities	yes	yes	no
User concepts	fine grained access rights according to SQL-standard	Users with fine-grained authorization concept	fine grained access rights according to SQL-standard