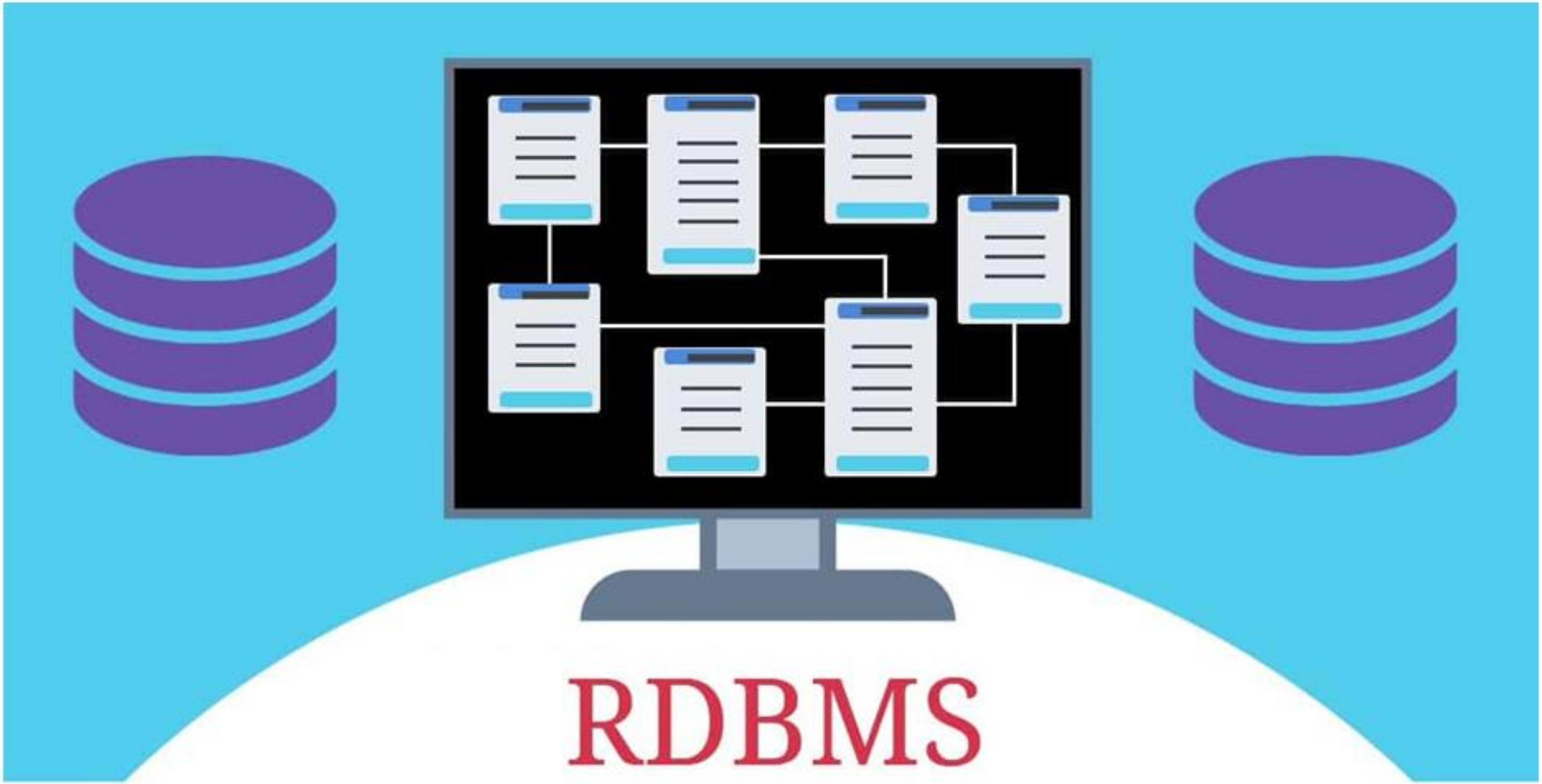
INTRODUCTION TO DATABASES CHECKPOINT

Ines BOUAICHA

Objective

Presentation that deals with 3 well known relational RDBMS which are:

1. Presenting each of the RDBMS and their functionalities
2. A comparison between the three RDBMS



RDBMS and their functionalities

Data, Database and Database Management System

Data

- Data is numerical, character or other symbols which can be recorded in a form suitable for processing by a computer. (e.g. names and addresses of students enrolling onto a university course).

Database

- A Database is a collection of related data (such as an enrolling students data) arranged for speedy search and retrieval.

Database Management System

- A Database Management System is a collection of programs that allows users to specify the structure of a database, to create, query and modify the data in the database and to control access to it. (e.g. limit access to the database so that only relevant staff can access details of enrolling students).

What is DBMS ?

- A set of programs to access the interrelated data.
- DBMS contains information about a particular enterprise.
- Computerized record keeping system.
- Provides convenient environment to user to perform operations:

  -Creation, Insertion, Deletion,

  Updating & Retrieval of informati

RDBMS...

- Most popular database system.
- Simple and sound theoretical basis.
- Developed by E F Codd in the early 1970's.
- The model is based on tables, rows and columns and the manipulation of data stored within.
- Relational database is a collection of these tables.
- First commercial system: MULTICS in 1978.
- Has overtaken Hierarchical and Network models.
- Main feature: Single database can be spread across several tables.
- Examples include: Oracle, IBM's DB2, Sybase, MySQL & Microsoft Access.

RDBMS Advantages

- Increases the sharing of data and faster development of new applications
- Support a simple data structure, namely tables or relations
- Limit redundancy or replication of data
- Better integrity as data inconsistencies are avoided by storing data in one place
- Provide physical data independence so users do not have to be aware of underlying objects
- Offer logical database independence - data can be viewed in different ways by different users.
- Expandability is relatively easy to achieve by adding new views of the data as they are required.
- Support one off queries using SQL or other appropriate language.
- Better backup and recovery procedures
- Provides multiple interfaces
- Solves many problems created by other data models
- The ability to handle efficiently simple data types
- Multiple users can access which is not possible in DBMS

RDBMS is a relational database management system which is the root for SQL. It is designed totally for relational models. The relational model can be represented on a table with rows and columns. Oracle is an example of RDBMS.

RDBMS is a plan of action and employs a structure which permits to recognize, update and handle the relation database. It is arranged in tables which contains rows and columns.

Every row in a table is nothing but a record through the unique ID known as key. Employs the normalization (method of arranging data in a database for eliminating the redundancy).

Functions of RDBMS

RDBMS provides the four major functions as listed −

Security

This is one of the most important functions of RDBMS. Security management sets rules that allow accessing the database. This function also sets restraints on what specific data any user can see or write.

Accuracy

In relational database management systems multiple tables are related to one another with the use of primary key and foreign key concepts. This makes data non repetitive so there is no chance for duplication of data. Hence, accuracy of RDBMS is good.

Integrity

Data integrity enforces the three constraints. Entity integrity means a table should have the primary key.

Consistency

The data consistency in relational model is best in RDBMS for maintaining data across application and database copies.

Other differences between database management systems and relational database management systems include:

Number of allowed users. While a DBMS can only accept one user at a time, an RDBMS can operate with multiple users.

Hardware and software requirements. A DBMS needs less software and hardware than an RDBMS.

Amount of data. RDBMSes can handle any amount of data, from small to large, while a DBMS can only manage small amounts.

Database structure. In a DBMS, data is kept in a hierarchical form, whereas an RDBMS utilizes a table where the headers are used as column names and the rows contain the corresponding values.

ACID implementation. DBMSes do not use the atomicity, consistency, isolation and durability (ACID) model for storing data. On the other hand, RDBMSes base the structure of their data on the ACID model to ensure consistency.

Distributed databases. While an RDBMS offers complete support for distributed databases, a DBMS will not provide support.

Types of programs managed. While an RDBMS helps manage the relationships between its incorporated tables of data, a DBMS focuses on maintaining databases that are present within the computer network and system hard disks.

Support of database normalization. An RDBMS can be normalized, but a DBMS cannot.


cassandra
MariaDB
CouchDB
SQLite
OrientDB
CUBRID
neo4j
MySQL
Firebird
PostgreSQL

A comparison between the three RDBMS

General information for MySQL, PostgreSQL and SQL Server

	MySQL	PostgreSQL	SQL Server
Maturity	Initial release was in 1995	Initial release was in 1989	MSMS SQL Server for OS/2 was released in 1989 (together with Sybase) SQL Server 6.0 was released in 1995 marking the end of collaboration with Sybase.
Language	Written in C, has a few C++ modules	Written in C	Mostly C++ with a few exceptions
Cost	Open source / Owned by Oracle and has several paid editions	Completely free / Open source	SQL Server Express is a free edition, but it is limited to using 1 processor, 1 GB memory and 10 GB database files.

Data changes for MySQL, PostgreSQL and SQL Server

	MySQL	PostgreSQL	SQL Server
Row Updates	Updates happen in place, changed data is copied to the rollback segment. This makes vacuuming and index compaction very efficient. MySQL is slower for reads, but writes are atomic and if columns in a secondary index change, this does not require changes to all indexes.	Updates are being implemented as inserts + mark as delete for vacuum. All indexes have a link to the physical id of the row. This has an update amplifying effect because when the column gets updated, new row with new physical id gets created and all indexes require updates, even those which are not referring to the changed column to get a pointer to the new row physical id.	Row-Store database engine: In-Memory database engine: updates implemented as insert + mark for delete. Garbage collector is not non-blocking and parallel Columnstore database engine: in-place updates
Vacuum / Defragmentation	Vacuuming and index compaction are very efficient.	Vacuum performs full tables scans to find the deleted rows and quite heavy process/might impact users' workload.	In-memory garbage collector might add max ~15% overhead, usually much less.

Querying the Data for MySQL, PostgreSQL and SQL Server

	MySQL	PostgreSQL	SQL Server
The buffer pool / cache that serves queries	MySQL cache that serves user queries is called a buffer pool. This cache can be set to the size as large as needs, leaving only enough memory for other processes on the server. You can split the buffer pool into multiple parts to minimize contention for memory structures and you can pin tables to the buffer pool. Table scan or mysqldump evicts older data.	PostgreSQL maintains shared memory for data pages and, due to the fact that it is a process-based system, each connection has a native OS process of its own and has its own memory. Process is releasing the memory after the execution has finished. Therefore, has problems scaling past hundreds of active connections.	SQL Server memory is called buffer pool and its size can be set as large as needed, no option to set multiple buffer pools.
Constraints support	Supports primary keys, foreign keys, not-null constraints, unique constraints, default constraints, does not support CHECK constraints	Supports primary keys, foreign keys, not-null constraints, check constraints, unique constraints, default constraints, exclusion constraints	Supports primary keys, foreign keys, not-null constraints, check constraints, unique constraints, default constraints
Temporary tables	Supports CTE, No support for global temp tables (available outside the session scope) and no table variables. Interesting fact: You cannot refer to a TEMPORARY table more than once in the same query. For example, the following does not work: SELECT * FROM temp_table JOIN temp_table AS t2;	Supports CTE, Global and local temporary tables and table variables (using table name as a type name). Interesting fact: if you create two tables with the same name, one is temporary and another one is regular table CREATE TEMP TABLE X (...) and CREATE TABLE X (...), "select * from x" will always bring data from temporary table.	Supports CTE, Global and local temporary tables and table variables.

Window / Analytical functions	Supports: CUME_DIST, FIRST_VALUE, LAG, LAST_VALUE, LEAD, PERCENT_RANK, ROW_NUMBER, RANK, DENSE_RANK, NTILE, NTH_VALUE No PERCENTILE_CONT, PERCENTILE_DISC functions.	Supports functions: CUME_DIST, FIRST_VALUE, LAG, LAST_VALUE, LEAD, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK, ROW_NUMBER, RANK, DENSE_RANK, NTILE, NTH_VALUE	Supports functions: CUME_DIST, FIRST_VALUE, LAG, LAST_VALUE, LEAD, PERCENTILE_CONT, PERCENTILE_DISC, PERCENT_RANK, ROW_NUMBER, RANK, DENSE_RANK, NTILE. Yet no NTH_VALUE function
Parallel query execution	MySQL will usually use 1 CPU per query.	Query plans can leverage multiple CPUs	Query plans can leverage multiple CPUs
Indexes	Supports index-organized tables - clustered indexes. Does not support persisted indexes / materialized views	Supports index-organized table, but updates are manual until ProstgreSQL 11 when it is automatic. Supports persisted indexes/materialized views.	Supports index-organized tables - clustered indexes that automatically maintains rows order.
Multiple indexes usage in single query	Multiple indexes might be used for the single query.	Multiple indexes might be used for the single query. If we have separate indexes on x and y, one possible implementation of a query like WHERE x = 5 AND y = 6 is to use each index with the appropriate query clause and then AND together the index results to identify the result rows.	Multiple indexes might be used for a single query (index intersection feature).
Multicolumn indexes	Multi-column indexes can have up to 16 columns	Multi-column indexes can have up to 32 columns	Multi-column indexes can have up to 16 columns
Partial indexes (an index built over a subset of a table using filter)	Does not support partial indexes	Supports partial indexes	Supports partial indexes

Join algorithms	MySQL executes joins between tables using only a nested-loop algorithm or variations of it.	Supports nested-loop joins, Hash joins and merge joins algorithms.	Supports nested-loop joins, hash joins and merge joins algorithms.
Query execution plan reuse	Maintains caches for prepared statements and stored programs on a per-session basis. Statements cached for one session are not accessible to other sessions.	Caches query plans only as long as the prepared statement is open. The query plan is disposed when the prepared statement is closed.	Has shared execution plan cache to enable queries to reuse execution plans
Statistics	Maintains persistent and non-persistent statistics (cleared on server restart)	Maintains statistics used by the planner, they are being updated by ANALYZE or VACUUM or CREATE INDEX	Maintains persistent statistics
Memory-optimized tables	MySQL has got an ability to store tables in memory. The tables that are created in memory do not support transactions, their data is vulnerable to crashes. Those tables should be used as a temporary area or as a read-only caches.	Does not offer any in-memory engine.	In-memory OLTP is integrated into SQL Server's database engine
Columnstore or row- store	MariaDB have recently launched the column store engine for MySQL which was designed as a massively parallel database in an environment with multiple servers. It can be used instead of InnoDB storage engine.	Row-store. Does not offer any columnar storage engine.	SQL Server offers column store indexes to query large tables

JSON and DATA Type Support for MySQL, PostgreSQL and SQL Server

	MySQL	Postgresql	SQL Server
JSON data type	MySQL has JSON data type support and also supports in place partial updates over the JSON instead of replacing the whole document however there are many limitations. It does not support indexing for JSON but there are workarounds.	PostgreSQL supports JSON data type and supports partial updates	SQL Server supports JSON data type and supports partial updates
Additional Advanced data types	Supports Geospatial data type. No user-defined types.	Supports Geospatial and lots of advanced data types, such as multi-dimensional arrays, user-defined types, etc.	Supports Geospatial data type, Hierarchical data

Sharding/Partitioning/Replication for MySQL, PostgreSQL and SQL Server

	MySQL	PostgreSQL	SQL Server
Partitioning support	Supports HASH partitioning (use HASH function on any column to split table into N partitions), RANGE or LIST partitioning that can be based on several columns and KEY partitioning which is similar to HASH but based on some auto generated number.	Supports RANGE and LIST partitioning but partitions and indexes on them must be manually created and old-style partitioning via table inheritance (when querying the parent table, all children tables are being queries as well, children tables have constraints on partitioning column. Interesting fact: Children tables can have more columns that parent table and indexes must be applied separately on children tables.)	Supports RANGE partitioning.
Sharding support	No good sharding implementation (MySQL Cluster is rarely deployed due to many limitations)	There are dozens of forks of Postgres which implement sharding but none of them yet haven't been added to the community release.	No standard sharding implementation.
Replication	Master-slave replication based on statements or based on changed rows Group replication with master server automatic election	Master - slave replication based on changed rows and log shipping.	Database level: Availability Groups master-multiple slaves Log shipping On data level: Master-slave / Bi-directional master-slave/ and master-master (merge) replication