



DevOps external course

Database Administration

Lecture 4.1

Module 4 **Database Administration**

Serge Prykhodchenko



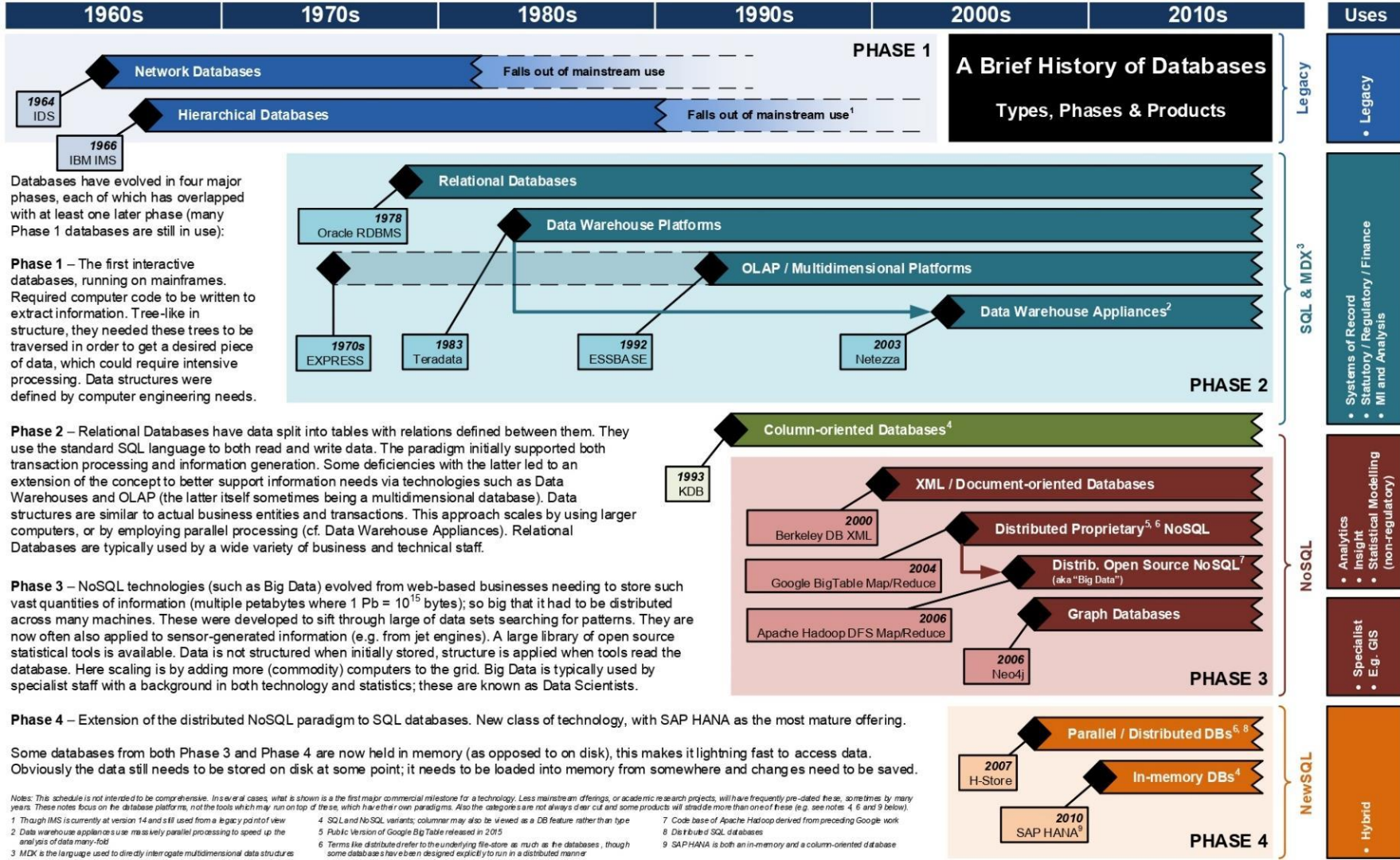
Agenda

- Databases
- DB Administration
- MySQL
- DB in clouds
- NonSQL-databases
- Q&A

DATABASES

What is a database?

- A database is one of the important components for many applications and is used for storing a series of data in a single set. In other words, it is a group/package of information that is put in order so that it can be easily accessed, manage and update.
- A database is a persistent repository of data stored in a computer. The data represent recorded information. By persistent we mean that the data remain available indefinitely, after the software applications that use or create the data are closed, and even when the computer systems on which the data are stored reboot or crash due to software or hardware failures.

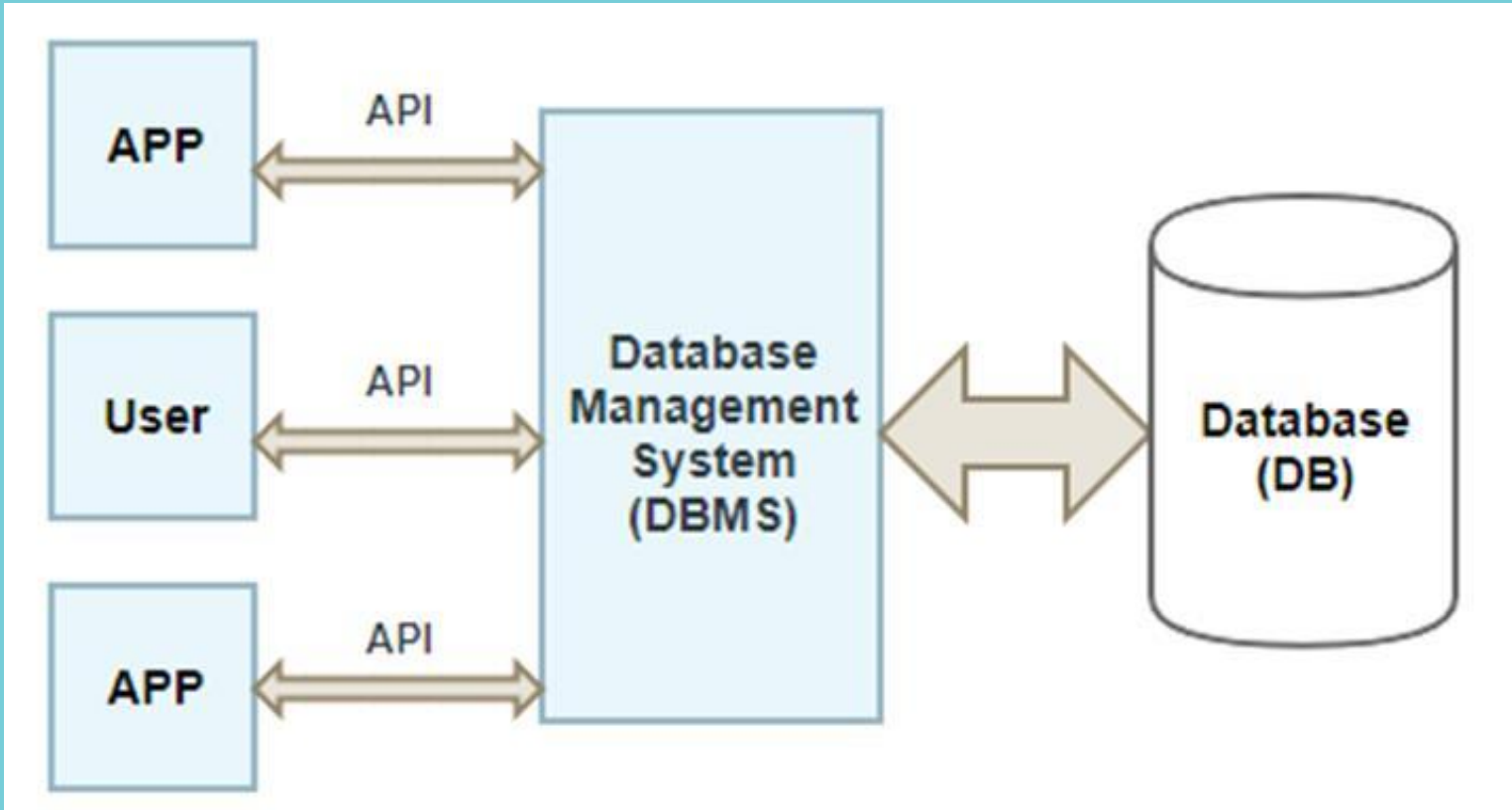


Evolution of databases

- NewSQL
- NoSQL
- Object-oriented
- Relational
- Network
- File system



Database Management System



Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages of DBMS

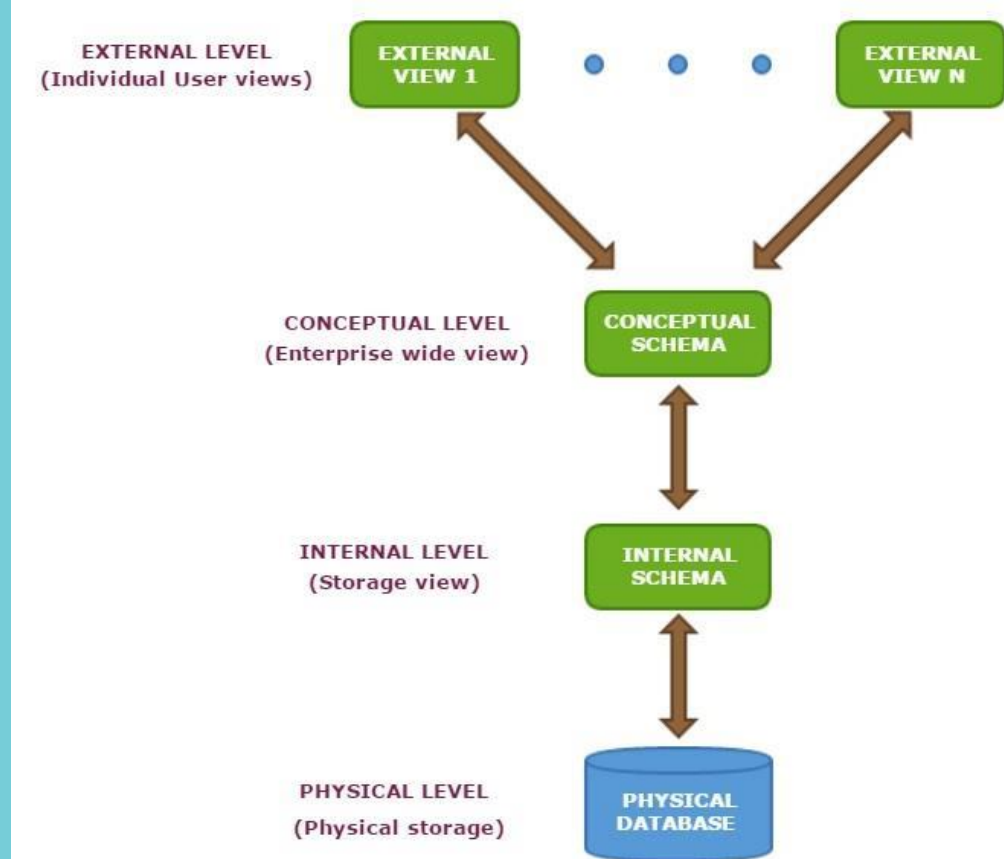
- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time:** It reduces development time and maintenance need.
- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **Multiple user interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.
- **Size:** It occupies a large space of disks and large memory to run them efficiently.
- **Complexity:** Database system creates additional complexity and requirements.
- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

Database Architecture

Three Level Database Architecture



Internal Level

- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored in a block.
- The physical level is used to describe complex low-level data structures in detail.

Conceptual Level

- The conceptual schema describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- The conceptual schema describes the structure of the whole database.
- The conceptual level describes what data are to be stored in the database and also describes what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level

External Level

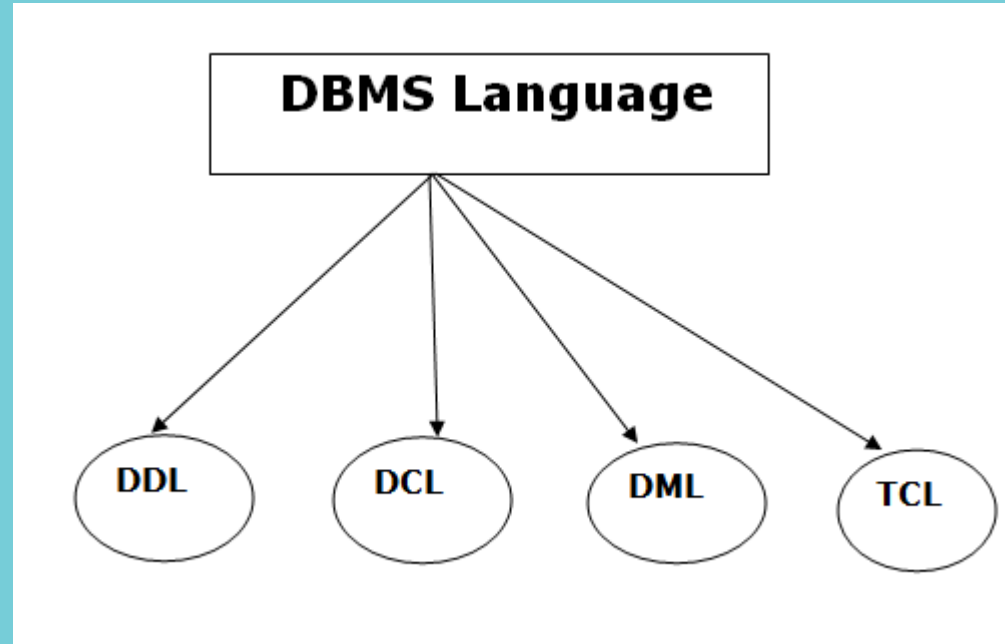
- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems

Data model Schema and Instance

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A database schema can be represented by using the visual diagram. That diagram shows the database objects and relationship with each other.
- A database schema is designed by the database designers to help programmers whose software will interact with the database. The process of database creation is called data modeling.

Types of Database Language

- DDL - Data Definition Language. It is used to define database structure or pattern
- DML - Data Manipulation Language. It is used for accessing and manipulating data in a database
- DCL - Data Control Language. It is used to retrieve the stored or saved data
- TCL - Transaction Control Language is used to run the changes made by the DML statement



Data Definition Language

- It is used to create schema, tables, indexes, constraints, etc. in the database.
- Using the DDL statements, you can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Here are some tasks that come under DDL:

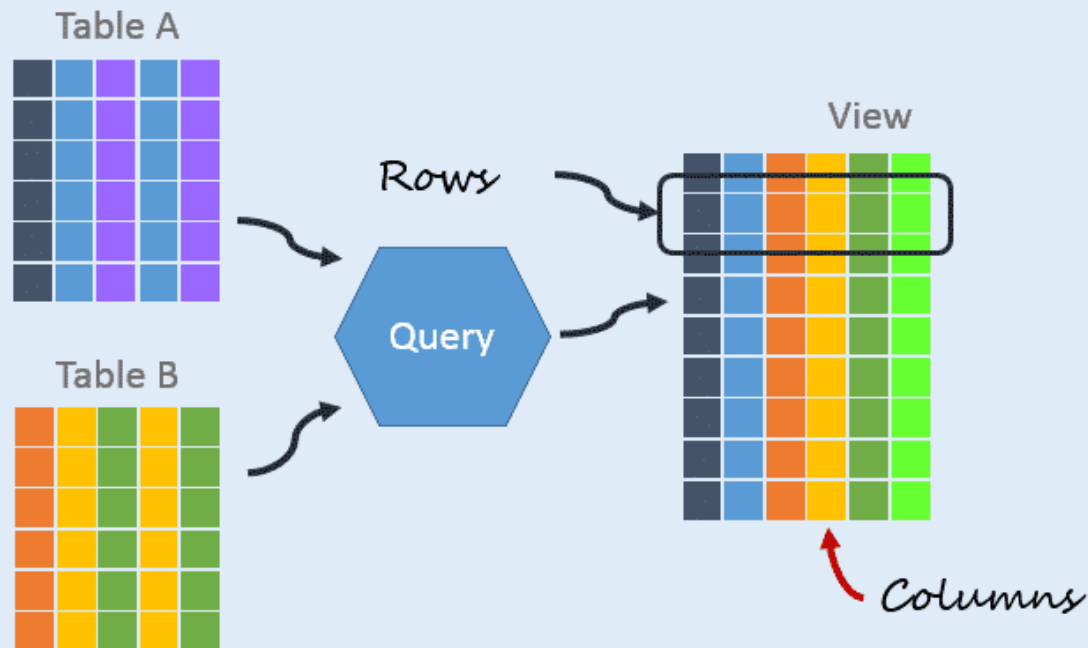
- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

Data Manipulation Language

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.
- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

Anatomy of a View



Characteristics

- One or more source tables make up a view
- Query follows "SELECT STATEMENT" format
- Views generally read-only
- Views don't require additional storage

Data Control Language and Transaction Control Language

Here are some tasks that come under DCL:

- Grant: It is used to give user access privileges to a database.
- Revoke: It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

TCL can be grouped into a logical transaction.

Here are some tasks that come under TCL:

- Commit: It is used to save the transaction on the database.
- Rollback: It is used to restore the database to original since the last Commit.

ER model

- ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity relationship diagram.
 - An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
 - A relationship is used to describe the relation between entities.

ACID

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties of database transactions intended to guarantee validity even in the event of errors, power failures, etc.

In the context of databases, a sequence of database operations that satisfies the ACID properties (and these can be perceived as a single logical operation on the data) is called a transaction. For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction

DB ADMINISTRATION

History

Classical approaches to filling the concept of "DBA" began to form after the publication of the working report of the group on databases of the American National Standards Institute ANSI/X3/SPARC in 1975. This report described a three-tier DBMS architecture. This architecture defined three DBA roles: conceptual schema administrator, external schema administrator, and storage administrator. In a very small system, these roles could be played by one person; in a large system, a group of people could be assigned to each role. Each role was assigned a set of functions, and all these functions together constituted the functions of the DBA.

In 1980 - 1981 it became accepted in the American literature to include in the functions of the ADB:

- organizational and technical planning of the database,
- database design,
- providing support for application development,
- DB operation management.

DBA Administration function

- analysis of the subject area;
- database structure design, data integrity assurance;
- initial loading and maintenance of the database;
- data protection, user policies;
- work with users;
- analysis of user requests to the database;
- analysis of the effectiveness of the functioning of the SDS and the development and optimization of the system;
- ensuring the transition to a new version of the DBMS;
- database backup and recovery;
- organizational and methodological work.

Malfunctioning DBMS

One of the main requirements for a DBMS is reliable storage of data in external memory. Storage reliability refers to the fact that the DBMS must be able to restore the last consistent state of the DB after any hardware or software failure. Two possible types of hardware failures are commonly considered: so-called soft failures, which can be interpreted as a sudden shutdown of the computer (for example, an emergency power off), and hard failures, characterized by the loss of information on external memory media.

Examples of software failures can be a DBMS crash (due to an error in the program or some hardware failure) or a user program crash, as a result of which some transaction remains incomplete. The first situation can be viewed as a special kind of soft hardware failure; when the latter occurs, it is required to eliminate the consequences of only one transaction.

Journaling

To restore the database, you need to have some additional information. In other words, maintaining reliable data storage in a database requires redundant data storage, and that part of them that is used for recovery must be stored especially reliably. The most common method for maintaining such redundant information is to maintain a database change log.

The journal is a special part of the database that is inaccessible to DBMS users and is maintained very carefully (sometimes two copies of the journal are maintained, located on different physical disks), which receives records of all changes to the main part of the database. In different DBMSs, database changes are logged at different levels: sometimes a log entry corresponds to some logical operation of a database change (for example, an operation to delete a row from a relational database table), and sometimes a record corresponds to a minimal internal operation of modifying an external memory page. Some systems use both approaches simultaneously.

Write Ahead Log

In all cases, the strategy of "ahead of time" writing to the log (the so-called Write Ahead Log - WAL protocol) is followed. Roughly speaking, this strategy consists in the fact that a record about a change of any database object must get into the external memory of the log before the changed object gets into the external memory of the main part of the database. It is known that if the WAL protocol is correctly observed in the DBMS, then using the log you can solve all the problems of restoring the database after any failure.

The simplest recovery situation is an individual rollback of a transaction. Strictly speaking, this does not require a system-wide database changelog. It is enough for each transaction to maintain a local log of the database modification operations performed in this transaction, and to roll back the transaction by performing the reverse operations, following from the end of the local log. In some DBMSs they do this, but in most systems local logs do not support, and individual transactions are rolled back according to the system-wide log, for which all records from one transaction are linked in a reverse list (from end to beginning).

Soft failures

In the event of a soft failure, the external memory of the main part of the database may contain objects modified by transactions that were not completed at the time of the failure, and there may be no objects modified by transactions that had successfully completed by the time of the failure (due to the use of RAM buffers, the contents of which disappear during a soft failure). If you follow the WAL protocol, the external log memory must be guaranteed to contain records related to the modification operations of both types of objects. The goal of the recovery process after a soft failure is the state of the external memory of the main part of the database, which would arise when all completed transactions were committed to external memory, and which would not contain any traces of unfinished transactions. To achieve this, they first roll back uncommitted transactions (undo), and then replay (redo) those operations of completed transactions whose results are not mapped to external memory. This process contains many subtleties related to the overall organization of buffer and log management.

Hard failures

To restore the database after a hard failure, a log and an archive copy of the database are used. Roughly speaking, an archive copy is a complete copy of the database by the time the journal starts filling (there are many options for a more flexible interpretation of the meaning of an archive copy). Of course, for a normal database recovery after a hard failure, it is necessary that the log does not disappear. As already noted, especially increased requirements are imposed on the safety of the journal in external memory in the DBMS. Then database recovery consists in the fact that, based on the archive copy, the work of all transactions that had ended by the time of the failure is reproduced in the log. In principle, you can even reproduce the work of incomplete transactions and continue their work after the end of recovery. However, in real systems this is usually not done because the recovery process from a hard failure is quite long.

Access control

- After obtaining the right to access the DBMS, the user automatically receives the **privileges** associated with his identifier. This can relate to procedures for accessing database objects, to operations on data. For the main objects of the database, tables can be built, which indicate the set of actions available to each user of the system.
- Each possible action on the data in the table is assigned a binary value, the overall result of possible operations is obtained by summing the values entered by the user.
- Privileges in a DBMS can be divided into two categories: **security privileges** and **access privileges**. Security privileges allow you to perform administrative actions, access privileges determine the access rights of subjects to certain objects.
- Before proceeding with the assignment of privileges, they must be created.

Privilege

Privileges can be subdivided according to the **types of objects** to which they belong: tables and views, procedures, databases, database server.

With regard to **tables**, the following access rights can be defined: the right to select, delete, update, add, the right to use foreign keys that refer to this table. By default, the user does not have any access rights to tables or views.

For **procedures**, you can grant the right to execute them, but you do not specify privileges on the right to access objects processed by the procedures. This allows you to allocate uncontrolled access to perform well-defined operations on the data.

In relation to the **database**, the allocated rights are actually prohibitive: a limit on the number of row I / O operations, the number of rows returned by one query.

Integrity support

- Ensuring data integrity is just as important as access control. The main enemies of databases are hardware, administrator, application, and user errors, not attackers. From the point of view of DBMS users, the main means of maintaining data integrity are: **constraints; regulations.**

- **Constraints** can be supported directly within the relational data model, or they can be set during table creation. Table constraints can refer to a group of columns, individual attributes. Reference constraints are responsible for maintaining the integrity of the relationships between tables. Restrictions are imposed by the owner of the table and affect the result of subsequent data operations. Constraints are a static element of maintaining integrity because they are they either allow the action to be performed or not.

Rules

Rules allow you to call the execution of specified procedures when certain changes to the database. Rules are associated with tables and are triggered when these tables change. Unlike constraints, which provide control over relatively simple conditions, rules allow you to check and maintain relationships of any complexity between data items in a database. In the context of information security, it should be noted that the creation of a rule associated with a table can be implemented only by the owner of this table. The user whose actions cause the rule to be triggered must have only the necessary access rights to the table. Thus, **the rules implicitly extend the user's privileges**. Such extensions must be controlled administratively, since even a minor change in the rule can dramatically affect the security of the data. There is a clear caveat when using rules as an information security tool: a mistake in a complex system of rules is fraught with unpredictable consequences for the entire database.

Logging and auditing

Audit - checking that all provided controls are in place and meet the level of security specified.

Such a measure as **logging and auditing** consists in the following: detection of unusual and suspicious user actions and identification of the persons who committed these actions; assessment of the possible consequences of the violation; Giving help; organization of passive protection of information from illegal user actions: maintaining the accuracy of the entered data; active documentation support; correct user testing.

It is recommended that when organizing logging, record the facts of transfer of privileges and connections to a particular database.

MYSQL

MySQL Introduction

- Open Source (C/C++), Free
- High Performance, Low Cost, High Reliability
- LAMP (Linux+Apache+MySQL+PHP)
- Multi-OS Support (Windows, Linux, MacOS)
- API Support (C/C++, C#, Python, PHP, Java)

What is MySQL?



- MySQL Database Server
 - MySQL is open-source DB server (RDBMS)
 - World's most-popular open-source database
 - Mostly used to power web sites and small apps
 - Supports concurrency, transactions (full ACID)
 - Stored procedures, views, triggers, partitioning
 - Support clustering and replication
- Free and paid editions
 - Community Server, Enterprise, Cluster CGE



MySQL Community Server

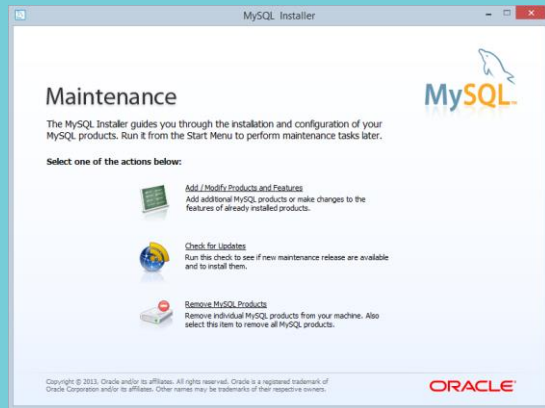
- MySQL Community Server
 - The free open-source MySQL edition
 - MySQL for Windows:
 - Pre-packaged installer available from <http://dev.mysql.com/downloads/mysql/>
 - MySQL for Linux:
 - Available through the package managers

```
sudo apt-get install mysql-server
```

(Debian / Ubuntu)

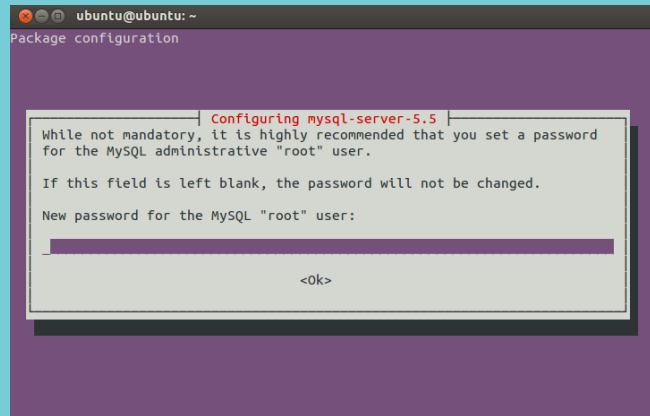
```
sudo yum install mysql-server
```

(Red Hat / CentOS)



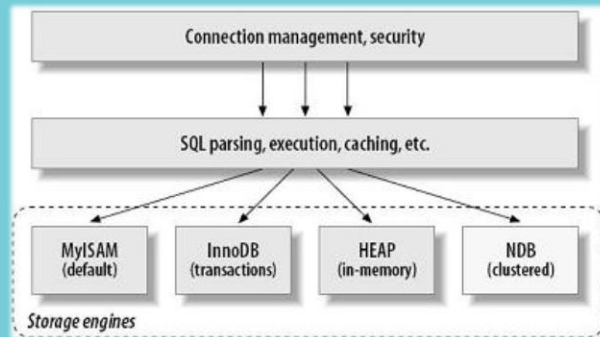
MySQL Installation

- OS: Ubuntu Linux
- Official Site:
<http://dev.mysql.com/downloads/mysql/>
 - Select Platform and download a deb package
 - Latest version: MySQL Community Server
- Using apt-get (Recommended)
 - `sudo apt-get install mysql-server mysql-client`
 - Set a password for root account
- For Windows, don't forget to set Windows system PATH environment variable



MySQL Storage Engines

- InnoDB
 - Fully ACID transactional, highly reliable
 - Recommended for most applications
- MyISAM
 - Fast, non-transactional, unreliable → forget it!
- Memory (HEAP)
 - Ultra-fast, non-persistent storage (in-memory)
- CSV
 - Stores the data in CSV (text) files (slow!)



mysql> SHOW ENGINES

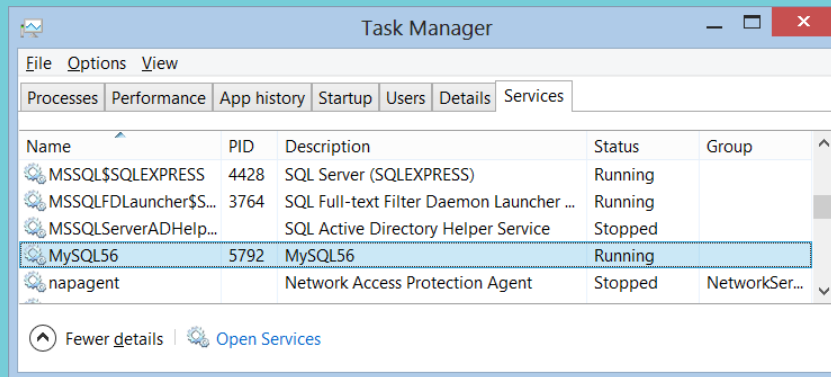
MySQL Services, Start, Stop

- MySQL services in Windows

- Just one service: **MySQLXX**

- Starting: `net start MySQL56`

- Stopping: `net stop MySQL56`

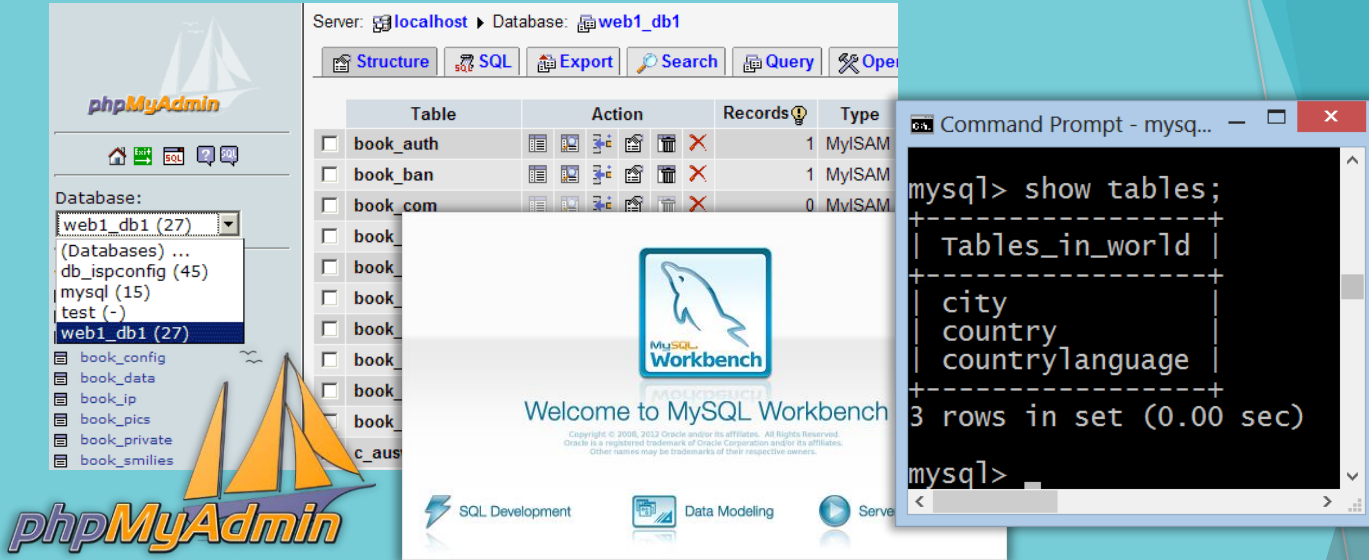


- MySQL services in Linux

- Starting: `sudo service mysql start`

- Stopping: `sudo service mysql stop`





MySQL Administration Tools

The Console MySQL Client, MySQL Workbench, phpMyAdmin

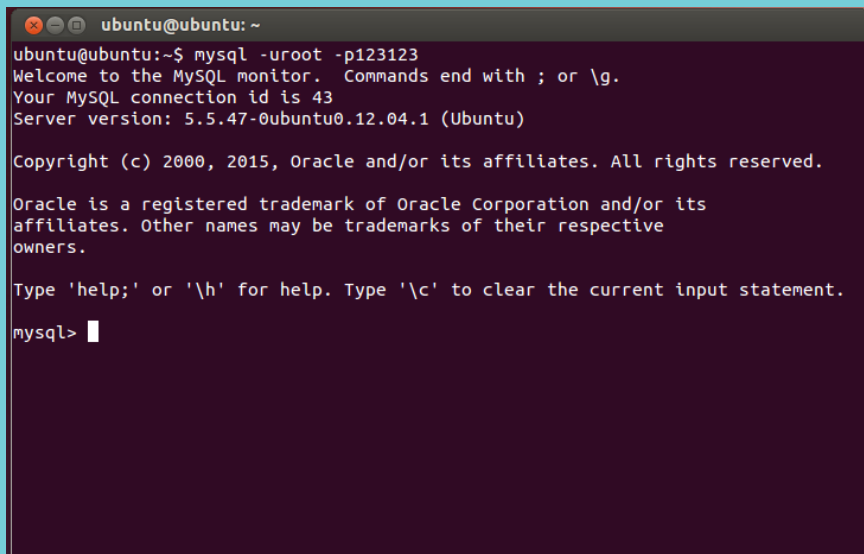
MySQL Command-Line Tool

- Login

- `mysql -u<username> -p<password>`
- `mysql -u root -p 123123`

- Logout

- `exit`

A terminal window titled 'ubuntu@ubuntu: ~' showing the execution of the 'mysql' command. The prompt is 'ubuntu@ubuntu:~\$'. The command entered is 'mysql -uroot -p123123'. The output shows a successful login to the MySQL monitor. The text displayed is: 'Welcome to the MySQL monitor. Commands end with ; or \g. Your MySQL connection id is 43 Server version: 5.5.47-0ubuntu0.12.04.1 (Ubuntu) Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved. Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners. Type \'help;\' or \'\\h\' for help. Type \'\\c\' to clear the current input statement. mysql>'. The prompt 'mysql>' is followed by a cursor.

```
ubuntu@ubuntu: ~
ubuntu@ubuntu:~$ mysql -uroot -p123123
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 43
Server version: 5.5.47-0ubuntu0.12.04.1 (Ubuntu)

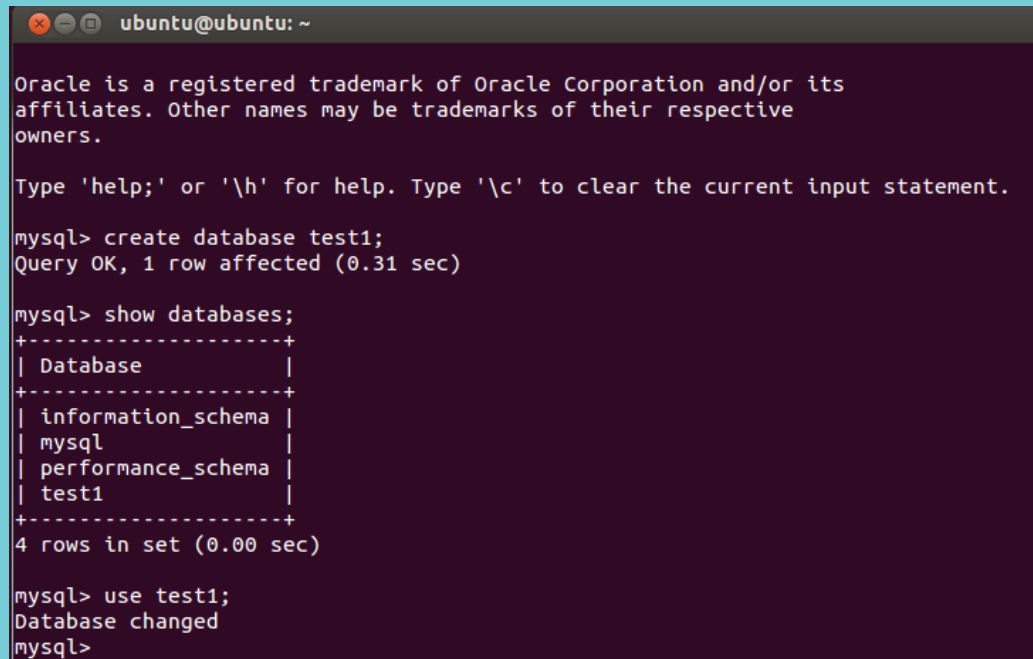
Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\\h' for help. Type '\\c' to clear the current input statement.
mysql> 
```

MySQL Command-Line Tool

- Input your SQL after “mysql>”

A screenshot of a terminal window with a dark background. The window title is 'ubuntu@ubuntu: ~'. The terminal shows the MySQL command-line tool interface. It starts with a copyright notice for Oracle, followed by instructions to type 'help;' or '\h' for help and '\c' to clear the input. The user enters 'mysql> create database test1;', and the response is 'Query OK, 1 row affected (0.31 sec)'. Then the user enters 'mysql> show databases;', and the response is a table listing four databases: information_schema, mysql, performance_schema, and test1. The user then enters 'mysql> use test1;', and the response is 'Database changed'. The prompt 'mysql>' is shown at the bottom.

```
ubuntu@ubuntu: ~  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
  
mysql> create database test1;  
Query OK, 1 row affected (0.31 sec)  
  
mysql> show databases;  
+-----+  
| Database                |  
+-----+  
| information_schema      |  
| mysql                   |  
| performance_schema      |  
| test1                   |  
+-----+  
4 rows in set (0.00 sec)  
  
mysql> use test1;  
Database changed  
mysql>
```

MySQL Command-Line Tool

```
mysql> create table mytable(data int);
Query OK, 0 rows affected (0.08 sec)

mysql> show tables;
+-----+
| Tables_in_test1 |
+-----+
| mytable          |
+-----+
1 row in set (0.00 sec)

mysql> insert into mytable (data) values (1),(2),(3),(4);
Query OK, 4 rows affected (0.00 sec)
Records: 4  Duplicates: 0  Warnings: 0

mysql> select * from mytable;
+-----+
| data |
+-----+
| 1    |
| 2    |
| 3    |
| 4    |
+-----+
4 rows in set (0.00 sec)
```

```
mysql> select * from mytable where data > 2;
+-----+
| data |
+-----+
| 3    |
| 4    |
+-----+
2 rows in set (0.00 sec)

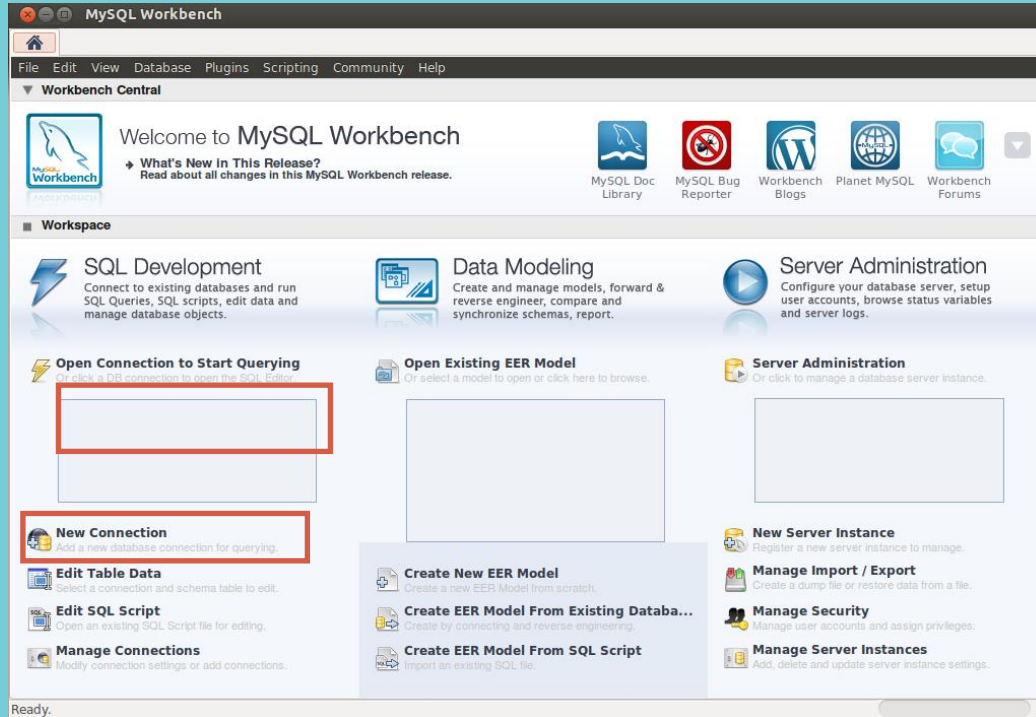
mysql> delete from mytable where data = 3;
Query OK, 1 row affected (0.32 sec)

mysql> select * from mytable where data > 2;
+-----+
| data |
+-----+
| 4    |
+-----+
1 row in set (0.00 sec)
```

MySQL Workbench Installation

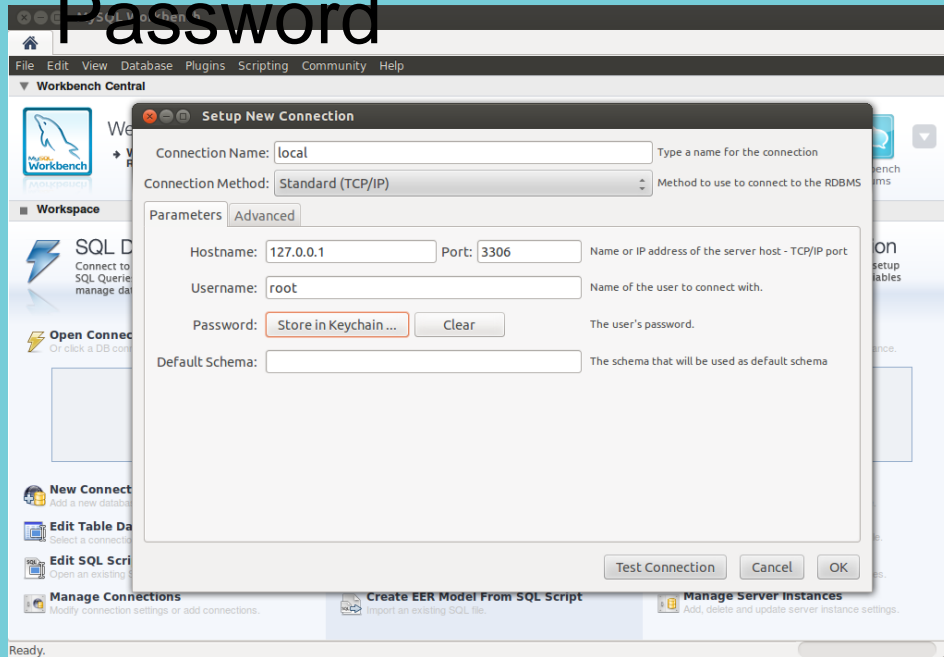
- A visual tool for MySQL
- Official Site:
<http://dev.mysql.com/downloads/workbench/>
 - Select Platform and download a deb package
 - Latest version: MySQL Workbench 6.3.6
- Using apt-get (Recommended)
 - `sudo apt-get install mysql-workbench`

MySQL Workbench Login

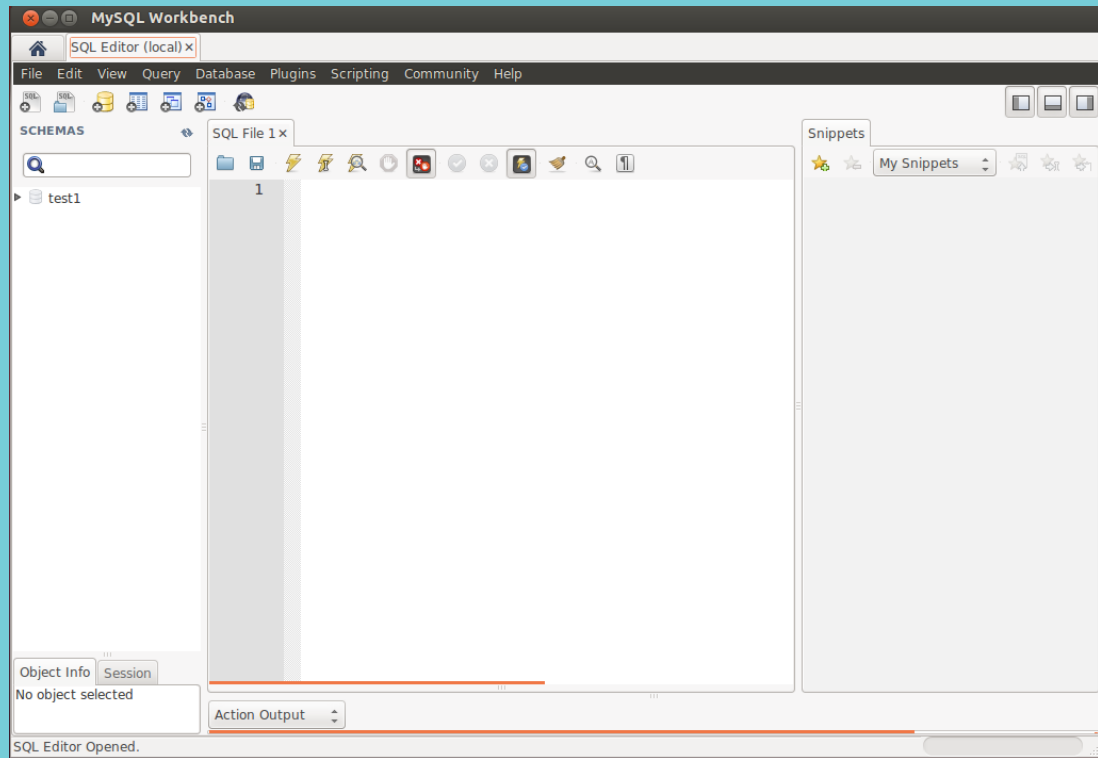


MySQL Workbench Login

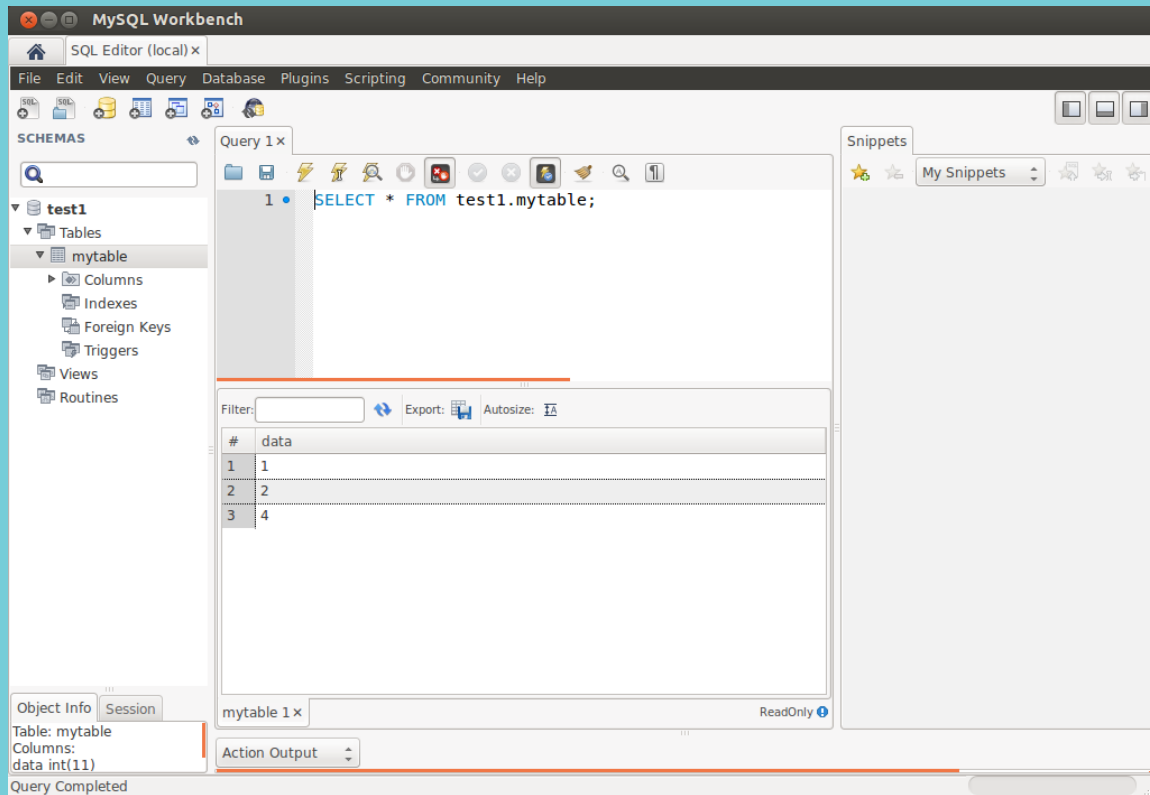
- Hostname, Port, Username, Password



MySQL Workbench UI

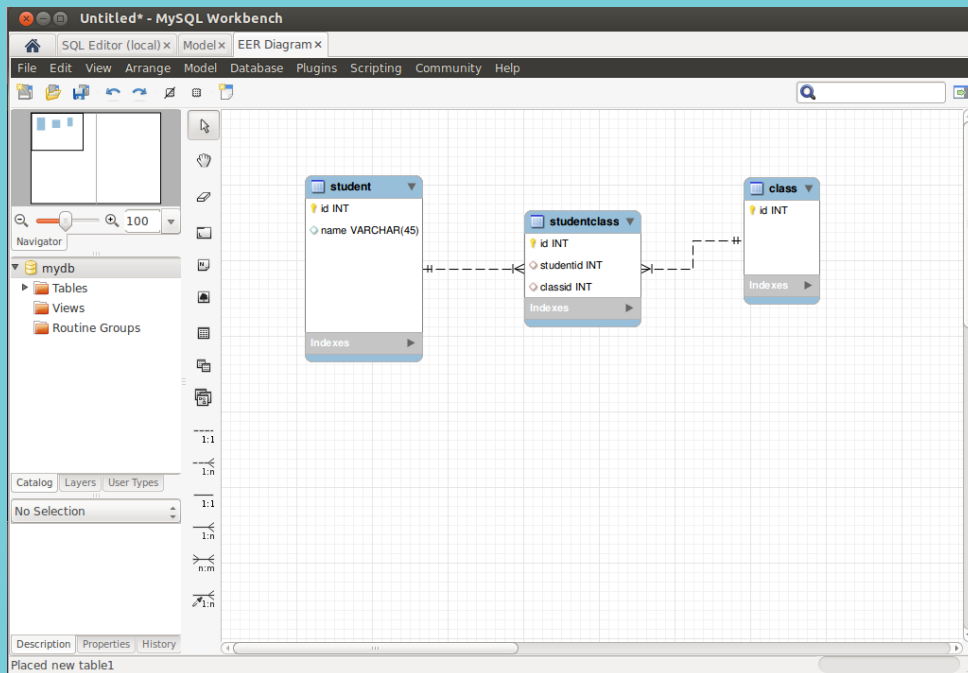


MySQL Workbench Usage



MySQL Workbench ER Model & Forward Engineering

- File->New Model->Add Diagram
- File->Export
- Get SQL Script



DB example

```
create table students(
```

```
    sid int,
```

```
    name varchar(40),
```

```
    dept varchar(40),
```

```
    age int,
```

```
    primary key(sid)
```

```
);
```

```
create table sc (
```

```
    sid int references students(sid) ON DELETE CASCADE ON UPDATE CASCADE,
```

```
    cid int ,
```

```
    semester int,
```

```
    cname varchar(40),
```

```
    grade int
```

```
);
```

```
create table courses(
```

```
    cid int,
```

```
    cname varchar(40),
```

```
    spring int,
```

```
    teacher varchar(40),
```

```
    primary key(cid)
```

```
);
```

phpMyAdmin Tool

- phpMyAdmin – Web-based open-source MySQL admin tool

The screenshot displays the phpMyAdmin web interface. On the left, a sidebar shows a tree view of databases and tables. The 'world' database is selected, showing tables: 'city', 'country', and 'countrylanguage'. The main panel shows the 'Structure' tab for the 'world' database. It lists the tables with their row counts and storage engines. Below the table list, there are options to 'Check All' and a dropdown for 'With selected:'. At the bottom, there are links for 'Print view' and 'Data Dictionary', and a 'Create table' button.

| Table | Action | Rows | Type |
|--|--|--------------|---------------|
| <input type="checkbox"/> city | Browse Structure Search Insert Empty Drop | ~4,188 | InnoDB |
| <input type="checkbox"/> country | Browse Structure Search Insert Empty Drop | ~239 | InnoDB |
| <input type="checkbox"/> countrylanguage | Browse Structure Search Insert Empty Drop | ~984 | InnoDB |
| 3 tables | Sum | 5,411 | InnoDB |

☐ Check All With selected: ▼

Print view Data Dictionary

Create table

Creating Databases and Tables

```
CREATE DATABASE books
```

- Creating a database
- Creating tables

```
USE books;
```

```
CREATE TABLE authors (  
    id INT NOT NULL AUTO_INCREMENT,  
    name VARCHAR(50) NOT NULL,  
    PRIMARY KEY (id)  
);
```

```
CREATE TABLE books (  
    id INT NOT NULL AUTO_INCREMENT,  
    name VARCHAR(150) NOT NULL,  
    isbn VARCHAR(13) NULL,  
    PRIMARY KEY (id)  
);
```

Edit Tables and Table Data

- Altering tables

```
ALTER TABLE books ADD COLUMN author_id INT NULL AFTER isbn;  
ALTER TABLE books ADD INDEX FK_books_authors_idx (author_id ASC);  
ALTER TABLE books ADD CONSTRAINT FK_books_authors  
FOREIGN KEY (author_id) REFERENCES authors (id);
```

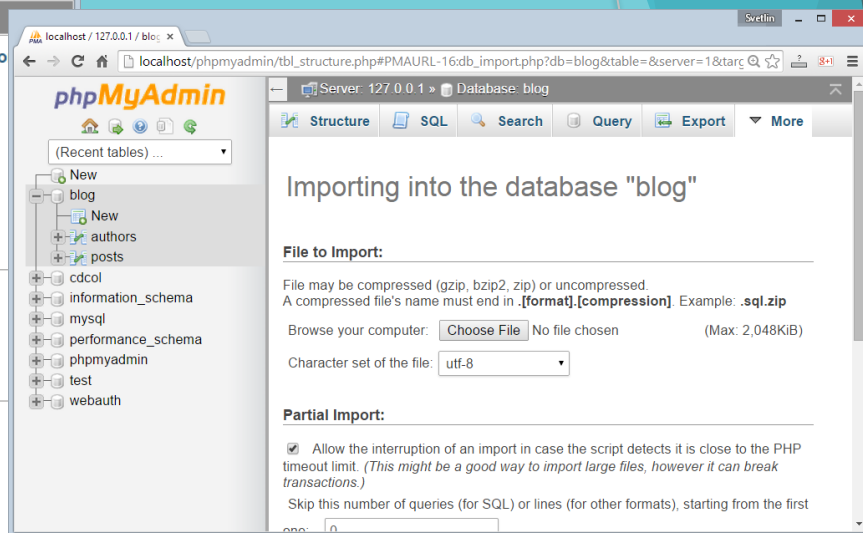
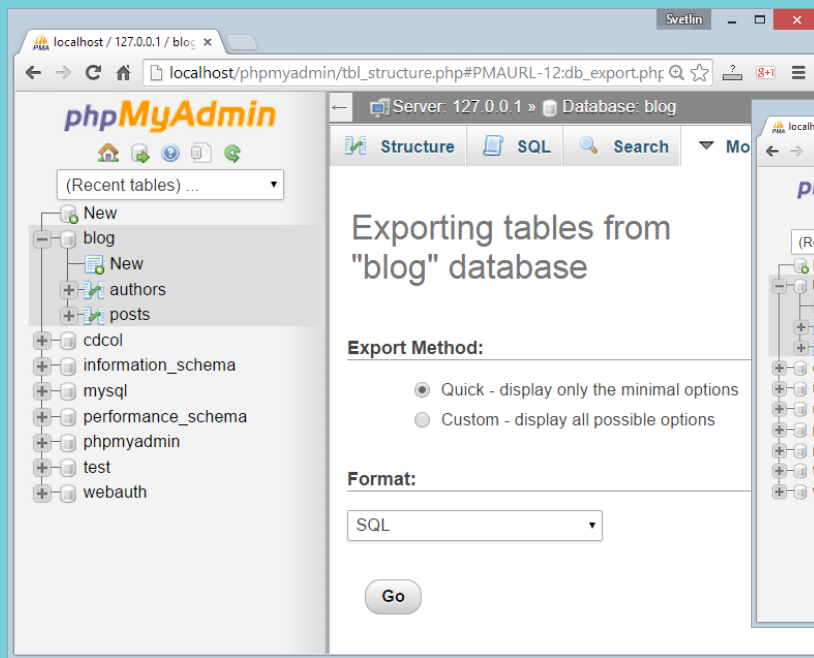
- Inserting data rows

```
INSERT INTO authors (name) VALUES ('Nakov');  
INSERT INTO books (name, author_id, isbn)  
VALUES ('Intro C#', 1, '9789544005276');
```


Moving a MySQL Database

- To move MySQL database to another server
 - Use the SQL export / SQL import features
- Export a database to SQL script
 - MySQL Workbench → Server Administration → Data Export → Export to Self-Contained File
 - phpMyAdmin → Export → SQL
- Import a database from SQL script
 - Just execute the script in Workbench
 - phpMyAdmin → Import → SQL





Import / Export MySQL Database

Other References

- <https://en.wikipedia.org/wiki/MySQL>
- <http://www.runoob.com/mysql/mysql-install.html>
- <http://blog.csdn.net/ithomer/article/details/5131863>
- <http://www.runoob.com/python/python-mysql.html>

MySQL Server Administration

- Server configuration
- The data directory, particularly the mysql system schema
- The server log files
- Management of multiple servers on a single machine

<https://dev.mysql.com/doc/refman/8.0/en/server-administration.html>

MySQL Database Server Administration (Job Example)

- Optimizing MySQL parameters for maximum performance.
- Updating the operating system and installed software.
- Automated monitoring of the availability of the server and the services it provides, and reporting of failures. Working mode 24 x 7.
- Automated monitoring of the server for software and hardware failures, and reporting failures.

Working mode 24 x 7.

- Organization of backup of main files for subsequent quick recovery of the server in case of failure.
- Securing the server from unauthorized access (setting iptables, fail2ban).
- Diagnostics of the reasons for slow work, slowlog analysis.
- Prompt response in case of server inoperability within 30 minutes.

General Security Issues

- Security Guidelines
- Keeping Passwords Secure
- Making MySQL Secure Against Attackers
- Security-Related mysqld Options and Variables
- How to Run MySQL as a Normal User
- Security Issues with LOAD DATA LOCAL
- Client Programming Security Guidelines

Backup and Recovery Types

- Physical (Raw) Versus Logical Backups
- Online Versus Offline Backups
- “hot” versus “cold” versus “warm”
- Local Versus Remote Backups
- Snapshot Backups (for MySQL is available through third-party solutions such as Veritas, LVM, or ZFS)
- Full Versus Incremental Backups
- Full Versus Point-in-Time (Incremental) Recovery
- Backup Scheduling, Compression, and Encryption

<https://dev.mysql.com/doc/refman/8.0/en/backup-types.html>

mysqldump

Logs

| Log Type | Information Written to Log |
|------------------------|---|
| Error log | Problems encountered starting, running, or stopping mysqld |
| General query log | Established client connections and statements received from clients |
| Binary log | Statements that change data (also used for replication) |
| Relay log | Data changes received from a replication master server |
| Slow query log | Queries that took more than <u>long query time</u> seconds to execute |
| DDL log (metadata log) | Metadata operations performed by DDL statements |

Replication

Replication (from Latin replico - I repeat) is the replication of data changes from the main

database server on one or more dependent servers.

- performance and scalability
- fault tolerance
- data backup
- lazy computation
- Backup from replica

Checking replication status `mysql> SHOW SLAVE STATUS`

SQL-injection

SQL injection is a dangerous vulnerability that arises from insufficient filtering of user input, which allows you to modify database queries. The result of the exploitation of SQL injection is to gain access to data that the user would not normally have access to.

- In addition to bypassing authentication, SQL injection is used to fetch information from databases, invoke denial of service (DoS), exploit other vulnerabilities (like XSS), etc

<https://habr.com/ru/post/148151/>

Mysql system tables

The mysql database, which is used for server administration, contains 24 system tables (tables of privileges, performance, etc.)

Show databases;

Use mysql;

show tables;

User table

Determines whether the user trying to connect to the server is allowed to do this. Contains username, password, and privileges.

show columns from user;

User table

User table

Determines whether the user trying to connect to the server is allowed to do this. Contains username, password, and privileges.

show columns from user;

Initially this table contains the root user with the password you set and the hostname '%'. By default, root can log in from any host and has full privileges and access to all databases. The table also contains an entry for the user '%', which must be deleted immediately, since it provides access to any user

delete from user where user = '%';

| | | | |
|---------------|----------|-----|---|
| Host | char(60) | PRI | |
| User | char(16) | PRI | |
| Password | char(8) | | |
| Select_priv | char(1) | | N |
| Insert_priv | char(1) | | N |
| Update_priv | char(1) | | N |
| Delete_priv | char(1) | | N |
| Create_priv | char(1) | | N |
| Drop_priv | char(1) | | N |
| Reload_priv | char(1) | | N |
| Shutdown_priv | char(1) | | N |
| Process_priv | char(1) | | N |
| File_priv | char(1) | | N |

db table

- Determines which databases which users and from which hosts are allowed to access. In this table, you can grant each user access to databases and assign privileges.

show columns from db;

- By default, all privileges are set to 'N'. For example, let's give the user john access to the library database and give him select, insert and update privileges.

insert into db (host, user, db, select_priv, insert_priv, update_priv) values

('%.domain.com', 'john', 'library', 'Y', 'Y', 'Y');

- The privileges set on the db table only apply to the library database. If you set these privileges in the user table, then they will be distributed to other databases, even if access to them is not explicitly set.

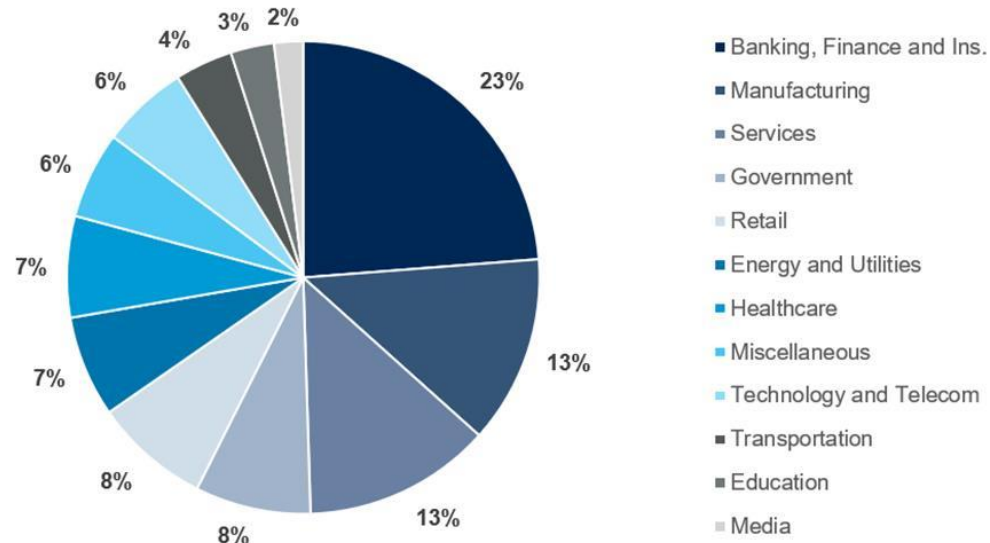
DATABASES IN CLOUDS

The Future of the DBMS Market Is Cloud

- DBMS revenue growth is in the cloud (68%).
- Increasing Relevance of the Cloud and CloudOnly DBMS Vendors Gartner estimates for 2018 DBMS revenue grew 18.4% to \$46 billion
- AWS and Microsoft represent 72% and 75% of total market growth
- By 2022, 75% of all databases will be deployed or migrated to a cloud platform
- four of the top five vendors in revenue order (Oracle, Microsoft, AWS and IBM) are also cloud service providers

Industry Sectors With Inquiries About Cloud and dbPaaS

Sample size 2,281

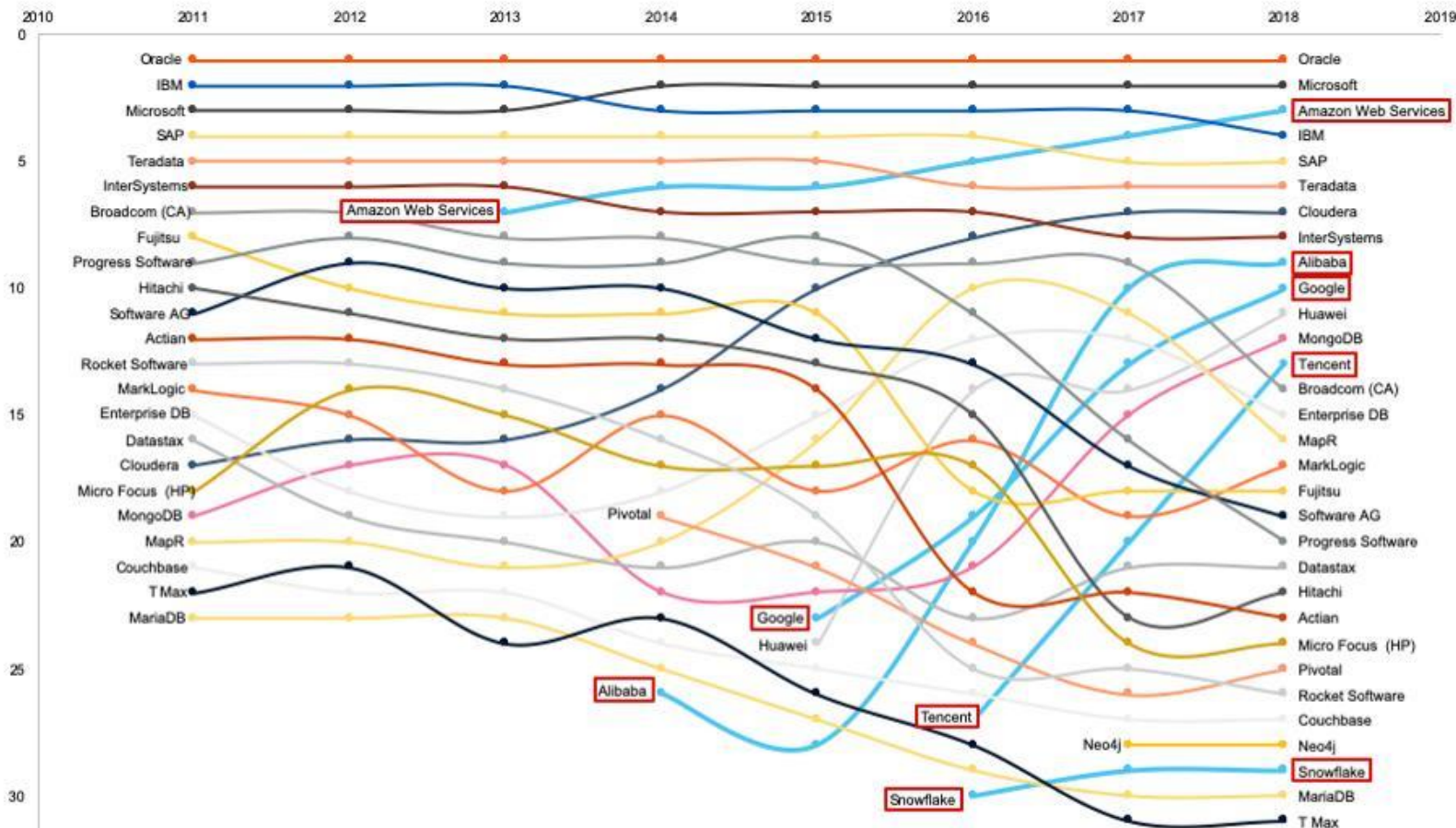


Source: Gartner (June 2019)

ID: 347472

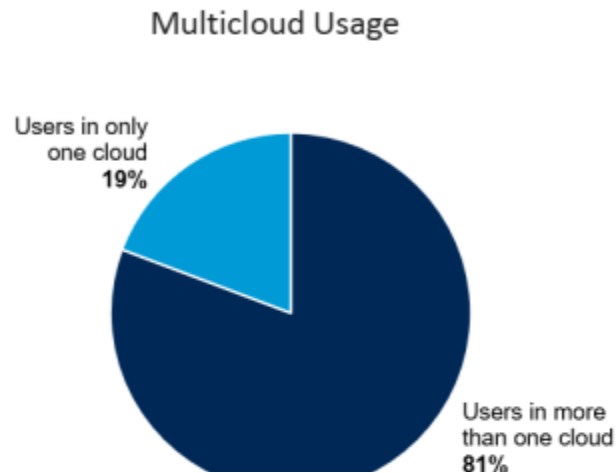
Gartner Market Share Ranking, 2011-2018

Rank



DBMS Cloud Services Revenue

| | Revenue | | | Revenue Growth | |
|-------------|----------|----------|-----------|----------------|---------|
| | 2016 | 2017 | 2018 | 2017 | 2018 |
| Amazon | 1,700.64 | 3,615.90 | 6,319.11 | 112.62% | 74.76% |
| Microsoft | 53.38 | 918.27 | 2,149.40 | 1620.11% | 134.07% |
| Alibaba | 96.93 | 213.44 | 460.55 | 120.21% | 115.77% |
| Oracle | 100.16 | 224.76 | 373.12 | 124.40% | 66.01% |
| Google | 101.47 | 164.36 | 285.49 | 61.98% | 73.70% |
| Tencent | 21.87 | 110.85 | 247.30 | 406.96% | 123.09% |
| Huawei | 77.42 | 70.99 | 137.87 | -8.32% | 94.22% |
| IBM | 57.28 | 73.35 | 120.22 | 28.06% | 63.90% |
| Cloudera | 23.85 | 45.35 | 79.21 | 90.15% | 74.66% |
| MongoDB | 9.12 | 8.77 | 65.70 | -3.90% | 649.33% |
| Other | 127.07 | 171.69 | 250.74 | | |
| Grand Total | 2,369.19 | 5,617.73 | 10,488.70 | 137.12% | 86.71% |



CLOUD WARS

Top 5 Cloud Vendors by 2019 Revenue

| Vendor | 2019 Cloud Revenue | 2019 Cloud Growth Rate |
|----------------|--------------------|------------------------|
| 1. Microsoft | \$44.7 B | 39% |
| 2. Amazon | \$35.03 B | 37% |
| 3. IBM | \$21.2 B | 14% |
| 4. Salesforce* | \$17.1 B (est.) | 25% (est.) |
| 5. Google | \$8.92 B | 53% |

*Salesforce reports on fiscal year ending Jan. 31, so had to estimate

AWS DATABASE TYPES

Management

- **Unmanaged** – managed by you

example: set up EC2, install DB into EC2

+ you have more fine-tuned control over how your solution handles changes in load, errors, and situations where resources become unavailable

- **Managed** - Scaling, fault tolerance, and availability are typically built into the service.

example: set up RDS

+ You manage: Application optimization

AWS manages: OS installation and patches; Database software installation and patches; Database backups; High availability; Scaling; Power and racking and stacking servers; Server maintenance

Database types

| Database type | Use cases | AWS service |
|---------------|---|--|
| Relational | Traditional applications, ERP, CRM, e-commerce | Amazon Aurora Amazon RDS Amazon Redshift |
| Key-value | High-traffic web apps, e-commerce systems, gaming applications | Amazon DynamoDB |
| In-memory | Caching, session management, gaming leaderboards, geospatial applications | Amazon ElastiCache for Memcached /for Redis |
| Document | Content management, catalogs, user profiles | Amazon DocumentDB |
| Graph | Fraud detection, social networking, recommendation engines | Amazon Neptune |
| Time Series | IoT applications, DevOps, industrial telemetry | Amazon Timestream |
| Ledger | Systems of record, supply chain, registrations, banking transactions | Amazon QLDB |

Relational Database

Advantages

Works well with structured data

Supports ACID transactional consistency and supports “joins”

Comes with built-in data integrity

Ensures data accuracy and consistency

Constrains relationships in this system

Equipped with limitless indexing

Not designed for

Semi-structured or sparse data

Use cases

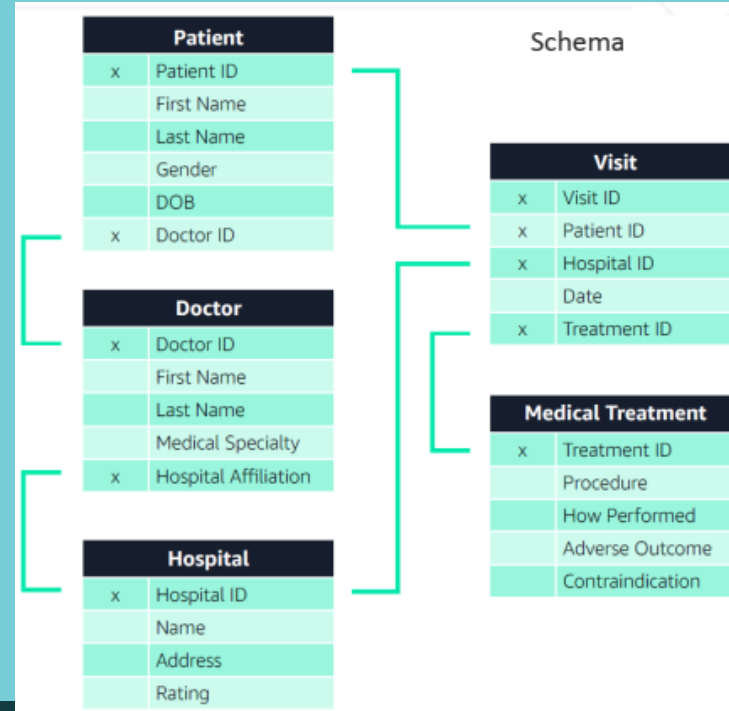
ERP apps

CRM

Finance

Transactions

Data warehousing



Application requires

- Complex transactions or complex queries
- A medium to high query or write rate – Up to 30,000 IOPS (15,000 reads + 15,000 writes)
- No more than a single worker node or shard
- High durability
- Massive read/write rates (for example, 150,000 write/second)
- Sharding due to high data size or throughput demands
- Simple GET or PUT requests and queries that a NoSQL database can handle
- Relational database management system (RDBMS) customization

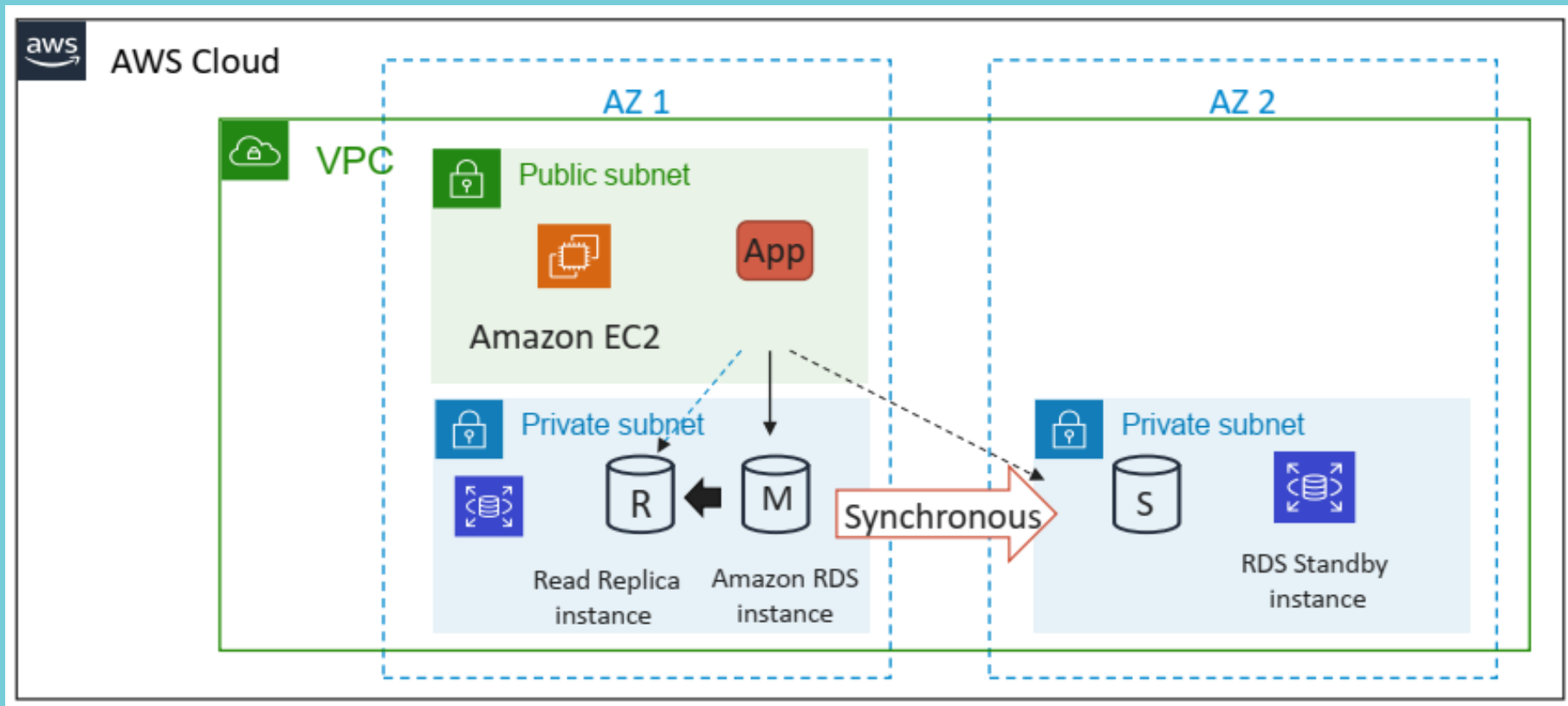


Use Amazon RDS



Do not Use Amazon RDS

Amazon RDS in VPC



Amazon RDS



Easy to administer



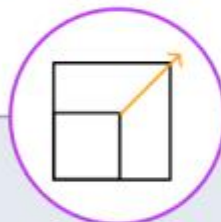
No need for infrastructure provisioning, installing and maintaining DB software

Available & durable



Automatic Multi-AZ data replication; automated backup, snapshots, failover

Highly scalable



Scale database compute and storage with a few clicks with no application downtime

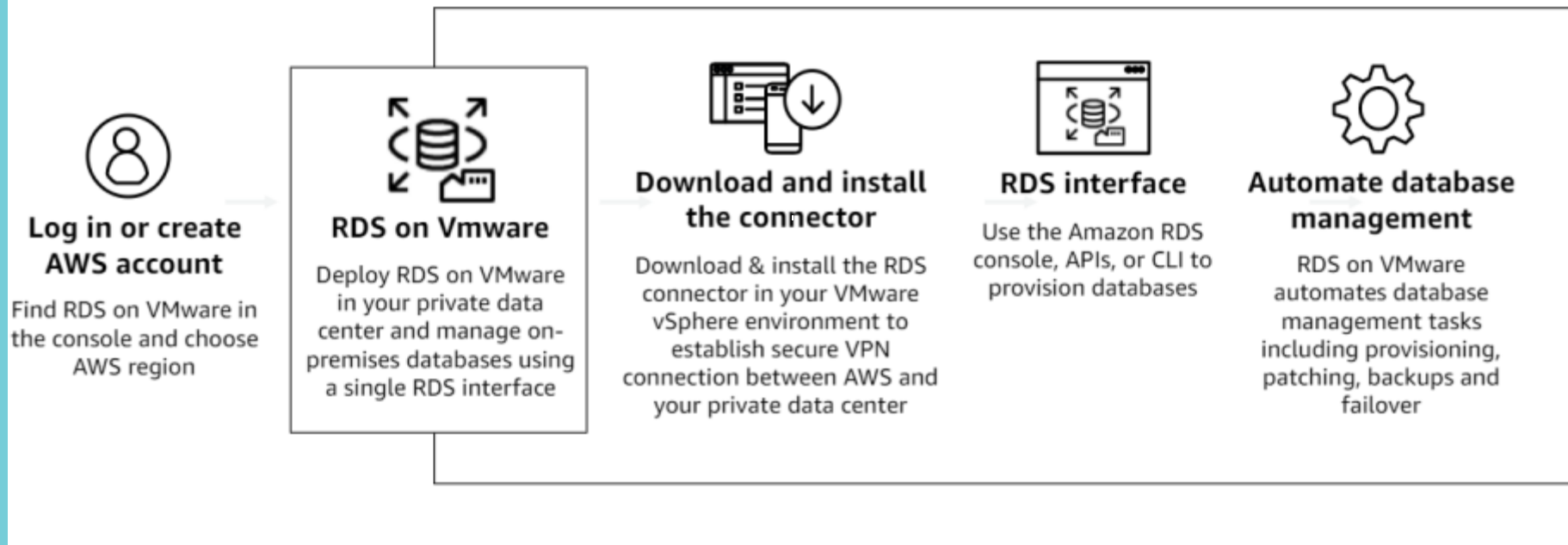
Fast & secure



SSD storage and guaranteed provisioned I/O; data encryption at rest and in transit

Managed relational database service with a choice of six popular database engines

How it work



Amazon Aurora



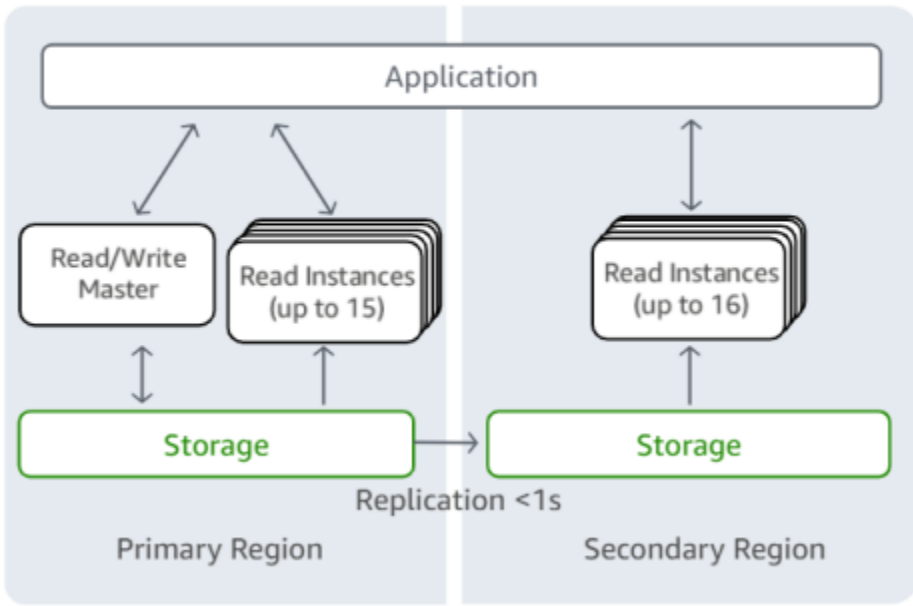
- Enterprise-class relational database
- Compatible with MySQL or PostgreSQL
- Automate time-consuming tasks (such as provisioning, patching, backup, recovery, failure detection, and repair).



Amazon Aurora

Amazon Aurora

High-performance database for globally-distributed applications



Single Global Database with cross region replication

Replication typically completes in less than a second

No impact on database performance

Write master in one region and read replicas in other regions

Cross-region disaster recovery

Local read latency for applications with global users

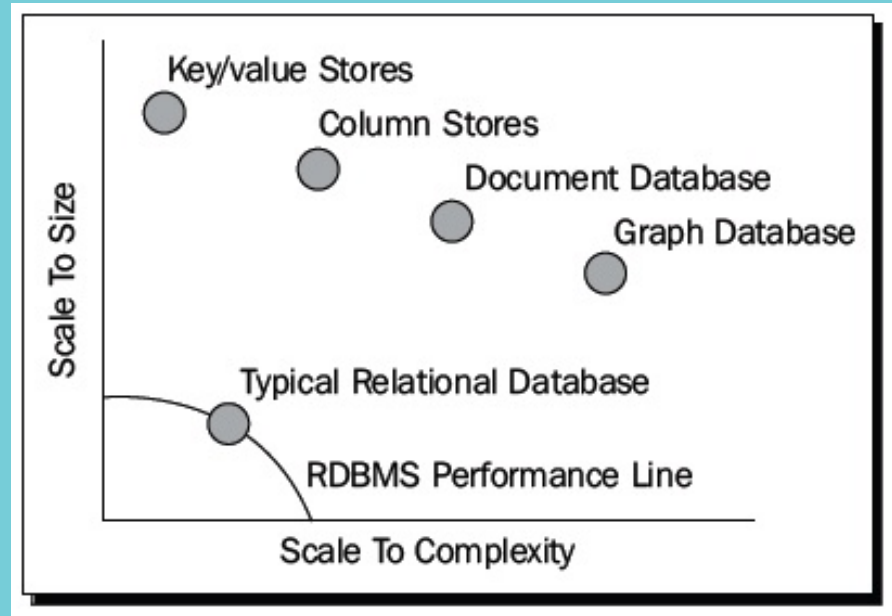
NONSQL DATABASES

Use cases

- High performance and scalable applications
- Most web applications where you would previously use SQL

Do not use for:

- Transaction-critical applications



Document databases

Advantages

- Flexible, semi-structured, and hierarchical
- Adjustable to application needs as databases evolve
- Flexible schema
- Simple hierarchical and semi-structured data
- Powerfully index for fast querying
- Naturally map documents to object-oriented programming
- Easily flows data to persistent layer
- Expressive query languages built for documents
- Capable of ad-hoc queries and aggregations across documents

Use Cases

- Catalogs
- Content management systems
- User profiles/personalization
- Mobile

Not designed for

- Explicitly defined relations between different pieces of data

```
1  [
2  {
3    "year": 2013,
4    "title": "Turn It Down, Or Else!",
5    "info": {
6      "directors": [ "Alice Smith", "Bob Jones"],
7      "release_date": "2013-01-18T00:00:00Z",
8      "rating": 6.2,
9      "genres": ["Comedy", "Drama"],
10     "image_url": "http://ia.media-imdb.com/images/N/O9ERWAU7FS797AJ7LU8HN09AMUP908RLIo5JF90EWR7LJKQ7@@_V1_Sx400_.jpg",
11     "plot": "A rock band plays their music at high volumes, annoying the neighbors.",
12     "actors": ["David Matthewman", "Jonathan G. Neff"]
13   }
14 },v
15 {
16   "year": 2015,
17   "title": "The Big New Movie",
18   "info": {
19     "plot": "Nothing happens at all.",
20     "rating": 0
21   }
22 }
23 ]
```

MongoDB

- Database for JSON objects
 - Perfect as a simple persistence layer for JavaScript objects
 - “NoSQL database”
- Data is stored as a collection of documents
 - Document: (almost) JSON object
 - Collection: group of “similar” documents
- Analogy
 - Document in MongoDB ~ row in RDB
 - Collection in MongoDB ~ table in RDB

MongoDB “Document”

```
{
  "_id": ObjectId(8df38ad8902c),
  "title": "MongoDB",
  "description": "MongoDB is NoSQL database",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": 100,
  "comments": [
    { "user": "lover", "comment": "Perfect!" },
    { "user": "hater", "comment": "Worst!" }
  ]
}
```

- `_id` field: primary key
 - May be of any type other than array
 - If not provided, automatically added with a unique `ObjectId` value
- Stored as BSON (Binary representation of JSON)
 - Supports more data types than JSON
 - Does not require double quotes for field names

MongoDB “Philosophy”

- Adopts JavaScript “laissez faire” philosophy
 - Don’t be too strict! Be accommodating! Handle user request in a “reasonable” way
- Schema-less: no predefined schema
 - Give me anything. I will store it anywhere you want
 - One collection will store documents of *any* kind with no complaint
- No need to “plan ahead”
 - A “database” is created when a first collection is created
 - A “collection” is created when a first document is inserted
- Both blessing and curse

MongoDB Demo

```
show dbs;

use demo;

show collections;

db.books.insertOne({title: "MongoDB", likes: 100});

db.books.find();

show collections;

show dbs;

db.books.insertMany([{title: "a"}, {name: "b"}]);

db.books.find();

db.books.find({likes: 100});

db.books.find({likes: {$gt: 10}});

db.books.updateOne({title: "MongoDB"}, {$set: { likes: 200 }});

db.books.find();

db.books.deleteOne({title: "a"});

db.books.drop();

show collections;

show dbs;
```

In-memory databases

Advantages

Sub-millisecond latency

Can perform millions of operations per second

Significant performance gains when compared to disk-based alternatives

Simpler instruction set

Support for rich command set (Redis)

Works with any type of database, relational or non-relational, or even storage services

Use Cases

Caching

Session store

Gaming

Leaderboards

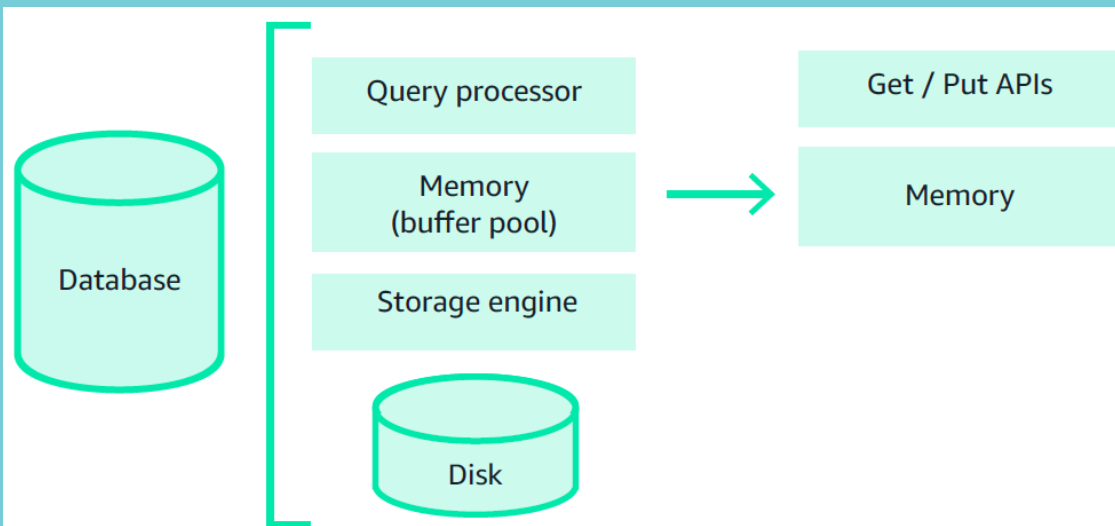
Geospatial services

Pub/sub

Real-time streaming

Not designed for

Persisting data to disk all the time



Graph databases

Advantages

Ability to make frequent schema changes

Quickly make relationships between many different types of data

Real-time query response time

Superior performance for querying related data—big or small

Meets more intelligent data activation requirements

Explicit semantics for each query—no hidden assumptions

Flexible online schema environment

Use Cases

Fraud detection

Social networking

Recommendation engines

Knowledge graphs

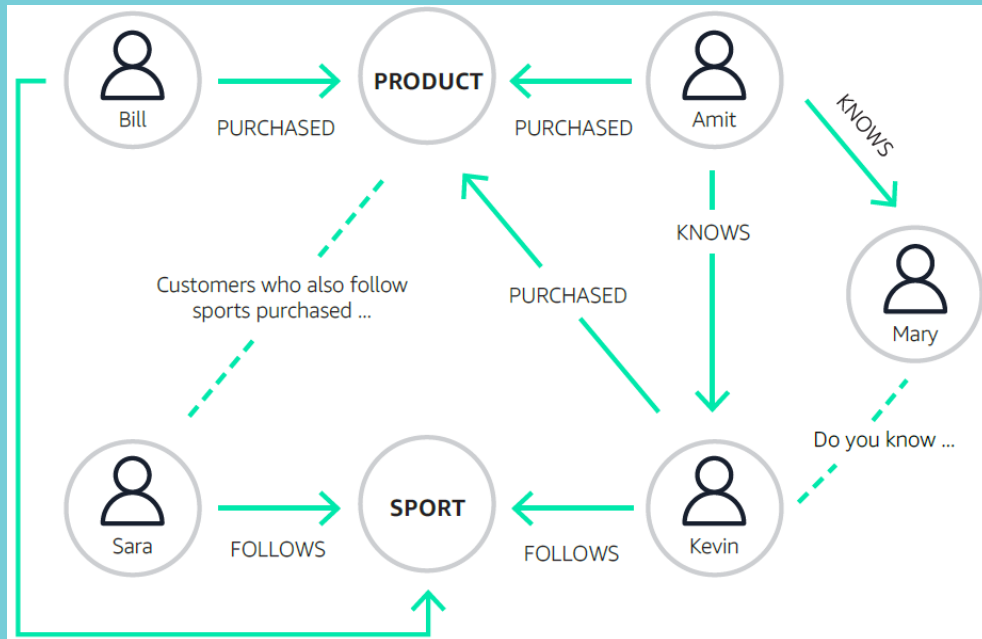
Data lineage

Not designed for

Applications that do not traverse or query relationships

Processing high volumes of transactions

Handling queries that span the entire database



Time-series databases

Advantages

Ideal for measurements or events that are tracked, monitored, and aggregated over time

High scalability for quickly accumulating time-series data

Robust usability for many functions, including: data-retention policies, continuous queries, flexible-time aggregations

Use Cases

DevOps

Application monitoring

Industrial telemetry

IoT applications

Not designed for

Data not in time-order form, such as: documents, catalogs, customer profiles

Ledger databases

Advantages

- Maintain accurate history of application data
- Immutable and transparent
- Cryptographically verifiable
- Highly scalable

Use Cases

- Finance - Keep track of ledger data such as credits and debits
- Manufacturing - Reconcile data between supply chain systems to track full manufacturing history
- Insurance - Accurately track claims history
- HR and payroll - Track and maintain a record of employee details
- Retail - Maintain an accurate log of inventory

Not designed for

- Decentralized use case (i.e., multiple entities need to read/write on data independently)

QUESTIONS & ANSWERS

COMMON LINKS

DB intro (eng) - <https://open.umn.edu/opentextbooks/textbooks/354>

DB intro (ua) -

https://web.posibnyky.vntu.edu.ua/fitki/11petuh_bazdanyh_movy_zalitiv/zmist.htm

DB intro (ru) - http://citforum.ru/database/advanced_intro/

NoSQL - <https://www.analyticsvidhya.com/blog/2020/09/different-nosql-databases-every-data-scientist-must-know/>

LINKS FOR TASK

3. <https://www.educative.io/blog/what-are-database-schemas-examples>
4. <https://www.mysqltutorial.org/mysql-create-database/>
5. <https://www.sqlshack.com/learn-sql-insert-into-table/>
6. <https://dev.mysql.com/doc/refman/8.0/en/select.html>
7. <https://www.tutorialgateway.org/sql-dml-ddl-dcl-and-tcl-commands/>
8. <https://chartio.com/resources/tutorials/how-to-grant-all-privileges-on-a-database-in-mysql/>
<https://dev.mysql.com/doc/refman/8.0/en/grant.html>
9. Look at p.6
10. <https://support.hostway.com/hc/en-us/articles/360000220190-How-to-backup-and-restore-MySQL-databases-on-Linux>
11. <https://phoenixnap.com/kb/mysql-drop-table>
12. Look at p.10

LINKS FOR TASK

13.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html

<https://docs.bitnami.com/aws/how-to/migrate-database-rds/>

14.

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_GettingStarted.CreatingConnecting.MySQL.html

15. Look at p.6

16. Look at p.10

17. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.CreateTables.html>

18. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.LoadData.html>

19. <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.Query.html>

A world map is displayed in the background, showing the continents of North America, South America, Europe, Africa, Asia, and Australia. The map is rendered in a light gray color against a white background. The text "THANK YOU!" is centered over the Atlantic Ocean, written in a black, serif, all-caps font.

THANK YOU!