

GIT

Dağıtık bir versiyon (sürüm) kontrol sistemi,
kaynak kod yönetim aracıdır.

distributed

her kulananda
yerel kopyası vardır.

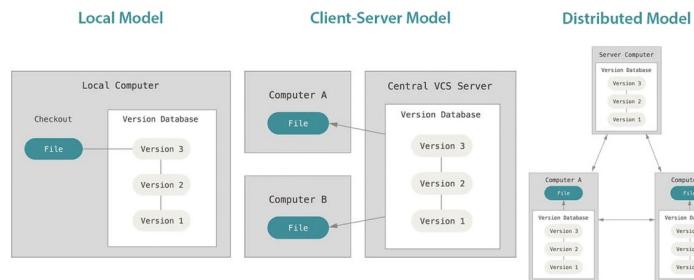
The various types of the version control systems are:

1. Local Version Control System
2. Centralized Version Control System
3. Distributed Version Control System

• **Local Data Model:** This is the simplest variations of version control, and it requires that all developers have access to the same file system.

• **Client-Server Model:** Using this model, developers use a single shared repository of files. It does require that all developers have access to the repository via the internet or a local network. This is the model used by Subversion (SVN).

• **Distributed Model:** In this model, each developer works directly with their own local repository, and changes are shared between repositories as a separate step. This is the model used by [Git](#), an open source software used by many of the largest software development projects.



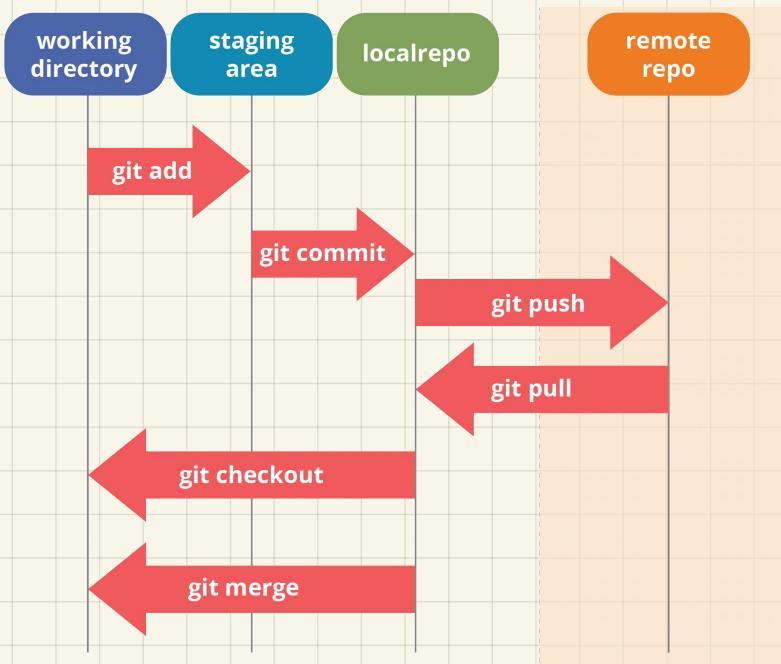
AVANTAJ

- **Hızlı ve Verimli:** sadece değişen dosya bilgilerini ek olarak saklar
- **Ekip halinde çalışmak:** işbirliği yapmak, ayrı çalışma sonra birleştirme
- **İşlem geri alma:** projelerin geçmiş versiyonlarını takip etme, değişiklikleri yönetme
- **Güvenlik:** proje bir bütün olarak birden fazla kişide bulunur

TEMEL KOMUTLAR

- **git init:** Bir Git deposu oluşturur. Bu komutu, bir projeyi Git ile izlemeye başlamak için proje dizininde çalıştırabilirsiniz.
- **git clone <repo-url>:** Bir uzak Git deposunu yerel makinenize kopyalar. <repo-url>, kopyalamak istediğiniz reposunun URL'sini temsil eder.
- **git add <dosya-adı>:** Bir dosyayı Git tarafından izlenen dosyalara ekler. Değişiklikleri Git'in takip etmesini istediğiniz dosyaları bu komutla belirtmelisiniz. Örneğin, git add dosya.txt dosyasını izlemeye alır.
- **git commit -m "< açıklama >":** Yapılan değişiklikleri bir commit olarak kaydeder. < açıklama >, commit için açıklayıcı bir mesajdır ve yapılan değişikliği tanımlar.
- **git push:** Yereldeki değişiklikleri uzak bir Git sunucusuna gönderir. Bu komut, yerel deponuzdaki commit'leri uzak sunucuya yükler ve paylaşmanızı sağlar.
- **git pull:** Uzaktaki bir Git deposundan değişiklikleri alır ve yerel projenize birleştirir. Bu komut, başka bir geliştiricinin yaptığı değişiklikleri güncellemenizi sağlar.
- **git branch:** Mevcut dalları listeler. Bu komutu çalıştırarak mevcut dalları görüntüleyebilir ve hangi dalda olduğunu öğrenebilirsiniz.
- **git checkout -b <yeni-dal>:** Yeni bir dal oluşturur ve üzerine geçer. <yeni-dal>, oluşturmak istediğiniz yeni dalın adını temsil eder.
- **git checkout <dal>:** Mevcut dalı değiştirir. <dal>, üzerine geçmek istediğiniz dalı temsil eder.
- **git merge <dal>:** İlgili dalı mevcut dala birleştirir. Bu komut, farklı dallardaki değişiklikleri birleştirerek kodu entegre etmenizi sağlar.
- **git status:** Değişiklikleri ve depo durumunu kontrol eder. Bu komutu çalıştırarak hangi dosyaların değiştiğini, hangi dosyaların staged (hazır) olduğunu ve hangi dosyaların commit edilmesi gerektiğini görebilirsiniz.
- **git log:** Commit geçmişini görüntüler. Bu komut, yapılan commit'leri, commit kimliklerini, tarihleri ve commit mesajlarını görüntüler.
- **git diff:** Değişiklikleri gösterir. Bu komut, çalışma dizinindeki değişiklikleri ve henüz staged (hazır) olmayan değişiklikleri gösterir. Kullanımı git diff veya git diff dosya.txt şeklindedir. İkinci kullanım, belirli bir dosyadaki değişiklikleri görüntüler.
- **git rm:** Bir dosyayı Git deposundan ve dosya sisteminizden kaldırır. Kullanımı git rm dosya.txt şeklindedir. Bu komutta dosyayı silip, bir sonraki commit'te silinmiş olarak işaretleyebilirsiniz. Staged ortamına eklenmiş bir dosyanın takibinin bırakılması yani untracked (izlenmeyen) hale getirilmesi sağlayan komuttur.

`git rm --cached <dosya veya klasor_adi>`



visual studio

- A** (Added) Staged ortamına eklenen
- U** (Untracked) Staged ortamına eklemeyen
- M** (Modified) Local Repository'de olup değişiklik yapılmış

Her commit benzersiz kimliğe (unique ID) sahip olur.

.gitignore

İzlenmeyecek dosya ve dizinleri belirlemek için kullanılan bir dosya

Derleme çıktıları

Yerel yapılandırma

Geçici dosyalar

- **Derleme Çıktıları:** Derlenmiş dosyalar veya derleme sürecinde oluşan geçici dosyalar (*.class, *.o, *.pyc vb.) .gitignore dosyasına eklenerek Git deposunda izlenmez.
- **Yerel Yapılandırma:** Kullanıcının yerel yapılandırma ayarları veya kimlik bilgileri (config.json, credentials.ini vb.) .gitignore dosyasına eklenerek Git deposuna eklenmez.
- **Geçici Dosyalar:** Çalışma ortamına veya metin düzenleyiciye özgü geçici dosyalar (*.swp, ~*, .DS_Store vb.) .gitignore dosyasına eklenerek izlenmez.

Each of the files in any current working Git repository is either:

- **tracked** – these are all the files or directories Git knows about. These are the files and directories newly staged (added with `git add`) and committed (committed with `git commit`) to the main repo.
- **untracked** – these are any new files or directories created in the working directory but that have not yet been staged (or added using the `git add` command).
- **ignored** – these are all the files or directories that Git knows to completely exclude, ignore, and not be aware of in the Git repository. Essentially, this is a way to tell Git which untracked files should remain untracked and never get committed.

- Operating System files. Each Operating System (such as macOS, Windows, and Linux) generates system-specific hidden files that other developers don't need to use since their system also generates them. For example, on macOS, Finder generates a `.DS_Store` file that includes user preferences for the appearance and display of folders, such as the size and position of icons.
- Configuration files generated by applications such as code editors and IDEs (IDE stands for Integrated Development Environment). These files are custom to you, your configurations, and your preferences settings.
- Files that get automatically generated from the programming language or framework you are using in your project and compiled code-specific files, such as `.o` files.
- Folders generated by package managers, such as npm's `node_modules` folder. This is a folder used for saving and tracking the dependencies for each package you install locally.
- Files that contain sensitive data and personal information. Some examples of such files are files with your credentials (username and password) and files with environment variables like `.env` files (`.env` files contain API keys that need to remain secure and private).
- Runtime files, such as `.log` files. They provide information on the Operating System's usage activities and errors, as well as a history of events that have taken place within the OS.

Kullandığınız yazılım diline, projeye özel .gitignore dosyaları araştırılabilir

*.class → tüm .class uzantılı dosyalar
build/ → “build” adlı dizini belirtir.

! → harici tutma

GitHub Benzeri Repo Hosting Platformları

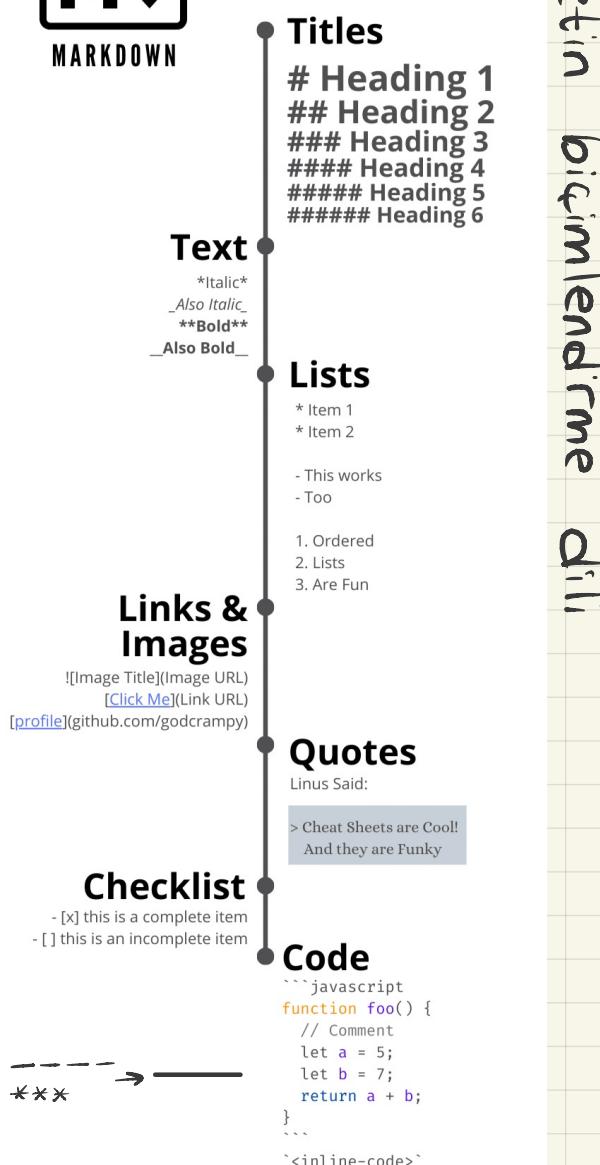
- Gitlab:** "GitLab nedir?" sorusunun yanıtı GitLab ile GitHub arasındaki farkın daha iyi anlaşılmasına açısından oldukça önemli. GitLab web tabanlı bir Git depo uygulaması olarak tanımlanabilir. Bu depo servisi sürekli entegrasyon (CI), sürekli teslimat (CD), hata kayıt, kod gözden geçirme ve wiki desteğiyle çalışıyor. **çalışıyor**
- Bitbucket:** Bitbucket, 2010 yılında Atlassian firması tarafından satın alınması ile beraber Mercurial ile birlikte Git desteği de vermeye başlayan ve günümüzde de hala sadece Git ve Mercurial versiyon kontrol sistemlerini (VCS) destekleyen, yazılım projeleri kodları için web tabanlı bir depolama servisidir.

MARKDOWN

A TINY CHEATSHEET ON



Markdown is a lightweight markup language. Its design allows it to be converted to many output formats like HTML



Metin biçimlendirme dili

Geliştiricilerin belirli aralıklarla ana sunucuya gönderdikleri değişiklikler conflict oluşturabilir / sistemde sorun ortaya çıkarabilir —

Integration (Merge) Hell

Bunu çözmek için CI sunucularında otomatik testler, analizler, kalite kontrol işlemleri yapılır.

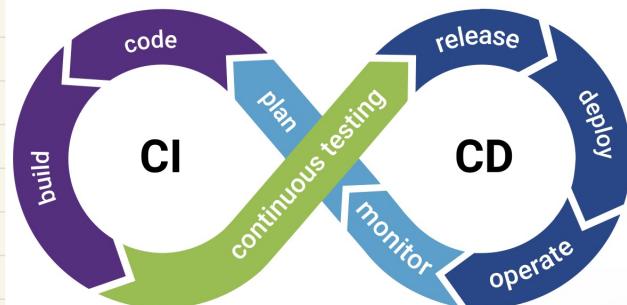
Yazılım kalitesi iyileşmekte, entegrasyon için ayrı zaman harcanmamakta

Continuous Integration — kodların geliştirildikten sonra sürekli bir kontrol mekanizmasının olmasıdır.

CD (Continuous Delivery/ Deployment) — sürekli doğrulama ve dağıtım mekanizmasıdır.

CI/CD pipeline : a practice focused on improving software delivery throughout the software development lifecycle via automation.

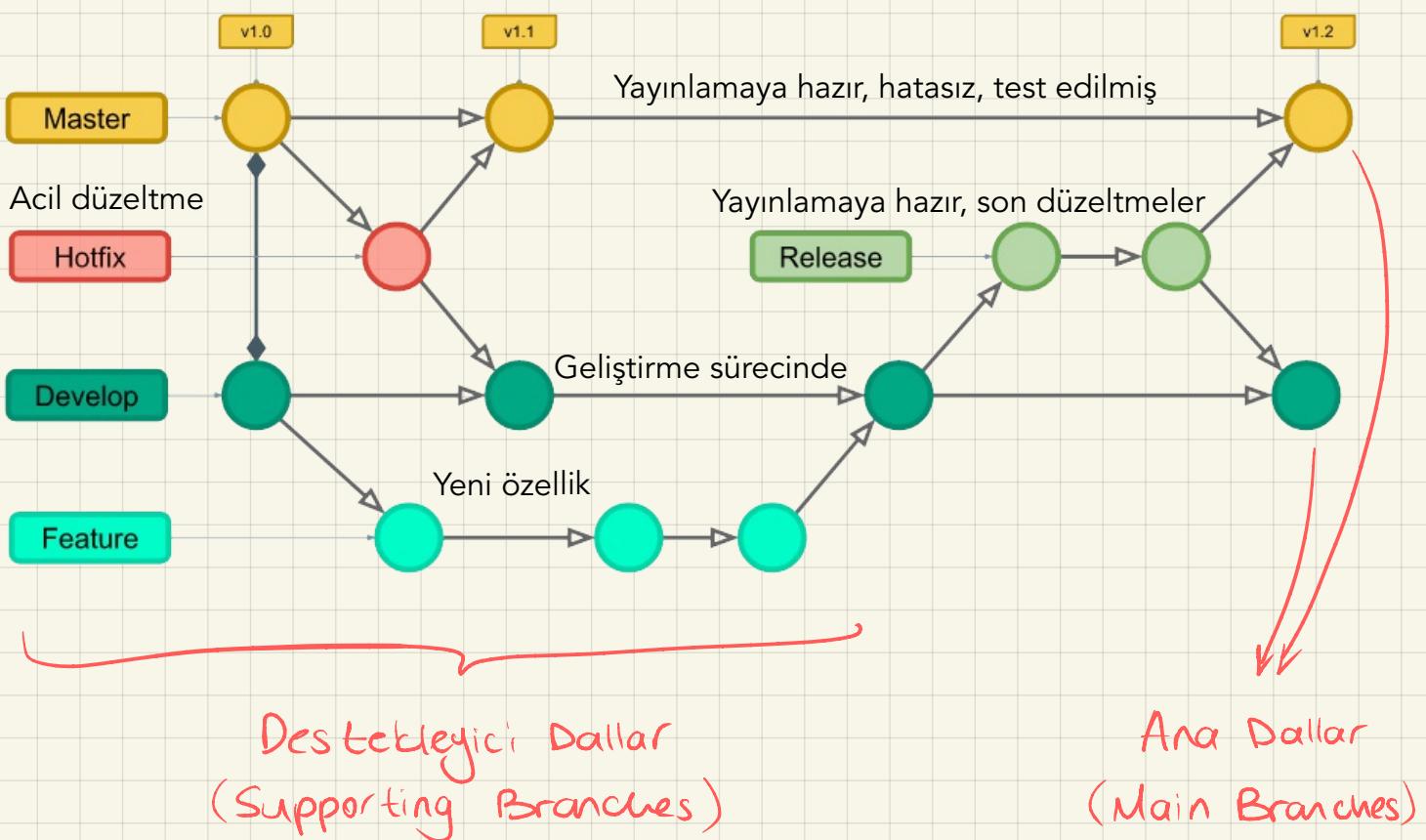
CI ile başlayan sürekli entegrasyon
CD ile sürekli olarak dağıtilır.
Testler CI aşamasında çalıştırılıp
uygulanın yeni sürümü CD
aşamasında dağıtilır.



GIT FLOW

Önerilen bir yazılım geliştirme süreci ve Git tabanlı bir dal (branch) modelidir.
Sürüm yönetimi stratejisi

- düzenli - paralel sürdürülüyor } Projenin



Personal Access Token (PAT) and Secure Shell (SSH) in Github

Personal Access Tokens are a way to authenticate and authorize access to your GitHub account.

Alternative to using your account password directly for authentication.

- Primarily used for non-interactive scenarios, such as accessing the GitHub API, using Git on the command line, or interacting with GitHub repositories through various integrations and services.

Secure Shell (SSH) is a cryptographic network protocol used for secure remote communication and login across networks.

- With SSH, you generate a public-private key pair on your local machine. The public key is added to your GitHub account, and the private key is stored securely on your local machine.
- Commonly used by developers and system administrators for securely pushing, pulling, and managing Git repositories hosted on GitHub.

SQL

Veri: ölçüm, sayım, gözlem veya araştırma sonucu elde edilen ham bilgi

Veritabanı: verilerin organize bir şekilde depolanmasını sağlayan, modifiye edebilebilen sistemler

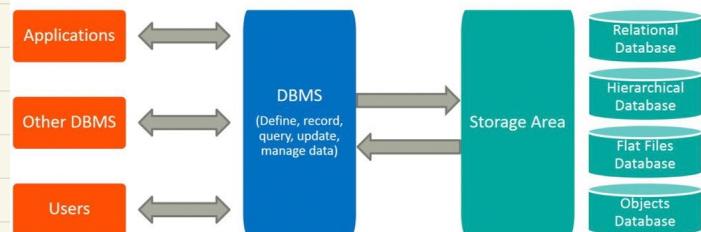
hepsine veritabanı deniyor ↗

- Verilerin Düzenli Saklanması:** Veritabanları, verileri yapılandırılmış bir şekilde saklama imkanı sağlar. Veriler, tablolar, sütunlar ve satırlar gibi yapılar içinde düzenlenir. Bu sayede verilerin mantıklı bir şekilde organize edilmesi ve daha kolay erişilebilmesi sağlanır.
- Veri Bütünlüğü:** Veri tabanları, veri bütünlüğünü korumak için mekanizmalar sunar. Veri bütünlüğü, verilerin doğruluğunu ve tutarlığını ifade eder.
- Veri Erişimi:** Veritabanları, verilere hızlı ve etkili bir şekilde erişim sağlar. Veritabanlarına indeksleme gibi yöntemler uygulanarak verilere hızlı bir şekilde erişmek mümkün olur.
- Veri Paylaşımı:** Veritabanları, birden çok kullanıcı arasında veri paylaşımını kolaylaştırır. Birden çok kullanıcı, aynı veritabanına erişebilir ve verilere güvenli bir şekilde erişim izinleri tanımlanabilir.
- Veri Güvenliği:** Veri tabanları, verilerin güvenliğini sağlamak için çeşitli güvenlik mekanizmaları sunar. Verilere yetkilendirme ve erişim kontrolleri uygulanabilir.
- Veri Yönetimi ve Analizi:** Veritabanları, verilerin etkin bir şekilde yönetilmesini ve analiz edilmesini sağlar. Veriler, sorgular ve raporlar kullanılarak işlenebilir, filtrelenir ve analiz edilir.

Veritabanı Yönetim Sistemleri

Veritabanı yönetim sistemleri, veritabanı ile etkileşimde bulunarak kullanıcıların verilere erişmelerini, sorgulamalarını ve veri manipülasyonu yapmalarını sağlar.

Database Management System



farklı kaynaklardan gelen sorgular DBMS yazılımı sayesinde farklı veri tabanlarında kullanılır.

DBMS types

The types of DBMS based on data model are as follows –

- Relational database.
- Object oriented database.
- Hierarchical database.
- Network database.



- Hiyerarşik Veritabanı
- Ağ Veritabanı
- İlişkisel Veritabanı (RDBMS)
- Nesneye Yönelik Veritabanı

two dimensional tables
attributes
values

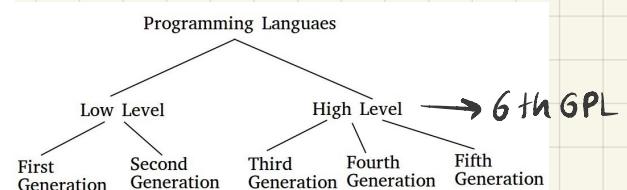
SQL (Structured Query Language)

İlişkisel veritabanı yönetim sistemlerinde (RDBMS) kullanılan bir sorgulama dili
Bildirimsel (declarative) ve Dördüncü Nesil programlama dili

categorization of programming languages
based on their historical development and
features

Fourth Generation Languages :

These are languages that consist of statements that are similar to statements in the human language. These are used mainly in database programming and scripting. Examples of these languages include Perl, Python, Ruby, SQL, and MatLab(MatrixLaboratory).



→ Imperative (Zorunlu) vs Declarative (Bildirimsel)

Imperative Programming	Declarative Programming
We specify how to get the desired result by providing detailed instructions.	We specify what result we expect from the program.
Imperative programming specifies and directs the control flow of the program.	Declarative programming specifies the expected result and core logic without directing the program's control flow.
The programmer makes the major decisions about how the program works.	The compiler makes the major decisions about how the program works.
It is easy to learn and understand.	The code is clean, readable, and effective.
It uses mutable variables, i.e., the values of variables can change during program execution.	It uses immutable variables, i.e., the values of variables cannot change.
Due to the use of mutable variables, imperative programming often changes the program's state by changing the internal data.	Declarative programming doesn't change the program's state.
Procedural Programming and Object-Oriented Programming are examples of imperative programming.	Functional Programming and Logic Programming are examples of declarative programming.
C, C++, Java, PHP, JavaScript, Python, etc., are programming languages that focus on the imperative paradigm.	SQL, LISP, Scala, Haskell, Prolog, Absys, Alice, etc., are programming languages that focus on the declarative paradigm.

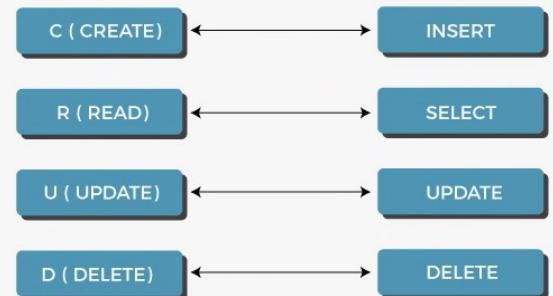
PostgreSQL

- **Açık kaynak kodlu**
- **Çeşitli veri tipi desteği**
- **Güçlü dökümantasyon ve topluluk desteği**
- **Tüm işletim sistemlerine uygunluk**
- **ACID uyumluluk**
 - **Atomicity (Atomiklik):** Atomiklik, bir veritabanı işleminin tüm veya hiç olarak kabul edilmesini ifade eder.
 - **Consistency (Tutarlılık):** Tutarlılık, veritabanının belli bir kural setine uygun olmasını ifade eder.
 - **Isolation (İzolasyon):** İzolasyon, eş zamanlı olarak çalışan işlemlerin birbirlerini etkilememesini sağlar.
 - **Durability (Dayanıklılık):** Dayanıklılık, bir işlemin tamamlandıktan sonra yapılan değişikliklerin kalıcı olarak saklanacağını garanti eder.
- **Güvenlidir**
- **Büyük verilerle kolay çalışma**

→ CRUD Operations (Create, Read, Update, Delete)

When creating a database or building APIs, you want users to be able to manipulate any data either by fetching, updating, deleting, or adding more data. These operations are made possible through CRUD operations.

SQL
↗



SQL Cheat Sheet

BASIC COMMANDS

SELECT - retrieves data from a database
FROM - specifies which tables to retrieve data from
WHERE - specifies which rows to retrieve based on certain conditions
GROUP BY - groups rows that have the same values in the specified columns
HAVING - filters groups based on a specified condition ↗
ORDER BY - sorts the retrieved rows in a specified order

AGGREGATE FUNCTIONS

AVG() - returns the average value of a set of values
COUNT() - returns the number of rows in a table or the number of non-null values in a column
FIRST() - returns the first value in a set of values
LAST() - returns the last value in a set of values
MAX() - returns the maximum value in a set of values
MIN() - returns the minimum value in a set of values
SUM() - returns the sum of a set of values

STRING FUNCTIONS

CONCAT() - concatenates two or more strings together
INSTR() - returns the position of a substring within a string
LENGTH() - returns the length of a string
LOWER() - converts a string to lowercase
LTRIM() - removes leading spaces from a string
REPLACE() - replaces all occurrences of a specified string with another string
RTRIM() - removes trailing spaces from a string
SUBSTR() - returns a portion of a string
TRIM() - removes leading and trailing spaces from a string
UPPER() - converts a string to uppercase

DATE FUNCTIONS

CURDATE() - returns the current date
CURTIME() - returns the current time
DATE() - extracts the date portion from a date/time value
DATEADD() - adds a specified time interval to a date
DATEDIFF() - returns the difference between two dates
DATEPART() - extracts a specified part of a date/time value
DAY() - returns the day of the month for a date value
MONTH() - returns the month for a date value
NOW() - returns the current date and time
YEAR() - returns the year for a date value

OTHER FUNCTIONS

COALESCE() - returns the first non-null value in a list of values
IFNULL() - returns a specified value if a value is null
NULLIF() - returns null if two values are equal

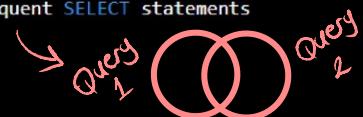
LIMIT
OFFSET
ALIAS ~ AS

JOINS

INNER JOIN - returns rows that have matching values in both tables
OUTER JOIN - returns rows that have matching values in either of the two tables
LEFT JOIN - returns all rows from the left table and any matching rows from the right table
RIGHT JOIN - returns all rows from the right table and any matching rows from the left table
FULL JOIN - returns all rows from both tables, whether or not there are matching values
CROSS JOIN - returns all rows from both tables, with each row from the first table being paired with each row from the second table

SET OPERATIONS

UNION - combines the results of two or more SELECT statements, eliminating duplicates
UNION ALL - combines the results of two or more SELECT statements, including duplicates
INTERSECT - returns rows that are present in the results of two or more SELECT statements
EXCEPT - returns rows that are present in the first SELECT statement but not in the results of any subsequent SELECT statements



SUBQUERIES

SELECT - retrieves data from a database within another SELECT statement
FROM - specifies which tables to retrieve data from within the subquery
WHERE - specifies which rows to retrieve based on certain conditions within the subquery
GROUP BY - groups rows that have the same values in the specified columns within the subquery
HAVING - filters groups based on a specified condition within the subquery

DATA MANIPULATION COMMANDS

INSERT INTO - adds a new row to a table
UPDATE - modifies existing data in a table
DELETE - removes existing rows from a table
TRUNCATE TABLE - removes all rows from a table

DATA DEFINITION COMMANDS

CREATE TABLE - creates a new table
ALTER TABLE - modifies the structure of an existing table
DROP TABLE - deletes a table
TRUNCATE TABLE - removes all rows from a table
CREATE INDEX - creates an index on a table to improve search performance
DROP INDEX - deletes an index from a table

By Steve Nouri

SQL OPERATORS

1. Logical (Mantıksal) Operators

AND, OR, ALL, ANY, BETWEEN, IN, LIKE, ILIKE, EXISTS, NOT, SOME

2. Comparison (Karşılaştırma) Operators

=, >, <, >=, <>

is not case sensitive

3. Arithmetic (Aritmetik) Operators

+, -, *, /, %

4. Bitwise Operators

&, |, ^

→ AND, OR, exclusive OR

5. Compound (Birleşik) Operators

+=, -=, *=, &= ... → Add equals, Subtract equals ...

→ AND kendinden hemen önce ve hemen sonraya bakar

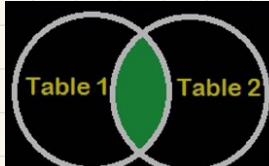
LIKE allows partial matching (using wildcards)

= checks for exact matches

Wildcard Characters

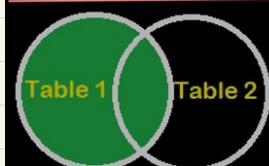
% _ [] ^ -

THE JOINS



-- INNER JOIN:

```
SELECT * FROM table1 INNER JOIN table2 ON table1.col1 = table2.col2
```

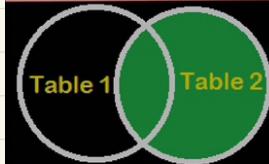


Outer Join

-- LEFT JOIN:

```
SELECT * FROM table1 LEFT JOIN table2 ON table1.col1 = table2.col2
```

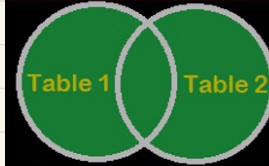
NULL değerler over



Outer Join

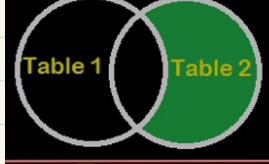
-- RIGHT JOIN:

```
SELECT * FROM table1 RIGHT JOIN table2 ON table1.col1 = table2.col2
```



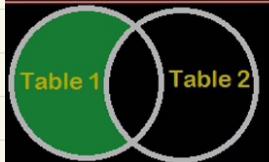
-- FULL OUTER JOIN:

```
SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.col1 = table2.col2
```



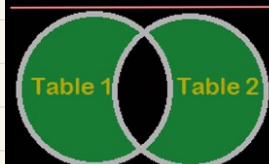
Right Outer Join with Exclusion

```
SELECT *
FROM table1
RIGHT JOIN table2 ON table1.col1 = table2.col2
WHERE table1.col1 IS NULL;
```



Left Outer Join with Exclusion

```
SELECT *
FROM table1
LEFT JOIN table2 ON table1.col1 = table2.col2
WHERE table2.col1 IS NULL;
```



Full Outer Join with Exclusion

```
SELECT *
FROM table1
FULL OUTER JOIN table2 ON table1.col1 = table2.col2
WHERE table1.col1 IS NULL OR table2.col1 IS NULL;
```

Order of Execution

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY

This order can be modified by the use of subqueries or common table expressions (CTEs). In these cases, the subquery or CTE is executed first, and the results are used in the outer query.

CTE is one-time result set that only exists for the duration of the query. Used when you want to perform multi-level aggregations.

VERİ TIPLERİ

İsim	Range
smallint	-32768 to +32767
<u>integer</u>	-2147483648 to +2147483647
bigint	-9223372036854775808 to +9223372036854775807
decimal	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
numeric	up to 131072 digits before the decimal point; up to 16383 digits after the decimal point
real	6 decimal digits precision
double precision	15 decimal digits precision
smallserial	1 to 32767
<u>serial</u>	1 to 2147483647
bigserial	1 to 9223372036854775807

Numeric Data Types

+ Boolean

↳ zero considered false
nonzero characters true

İsim	Tanım
character varying(n), <u>varchar(n)</u>	variable-length with limit
character(n), char(n)	fixed-length, blank padded
text	variable unlimited length

String Data Types

↳ boşluk bırakır

İsim	Tanım
timestamp [(p)] [without time zone]	both date and time (no time zone)
timestamp [(p)] with time zone	both date and time, with time zone
<u>date</u>	date (no time of day)
time [(p)] [without time zone]	time of day (no date)
time [(p)] with time zone	time of day (no date), with time zone
interval [fields] [(p)]	time interval

Date/Time Data Types

Terminale Erişim

Psql terminal tabanlı bir kullanıcı arayüzü,
pgAdmin grafiksel arayüzü

- **PSQL ile PostgreSQL'e bağlanmak:**
`psql -U <kullanıcı_adi>`
- **Kullanıcıya ait şifreyi girdikten sonra varsayılan veritabanı postgres'e bağlanıyor.**
`postgres=#`
- **Bulunan veritabanlarını listelemek için:**
`\l veya \list`
- **Bizim örneğimizde dvdrental veritabanına bağlanacağız.**
`\c dvdrental veya \connect dvdrental`
- **Bağlanılan dvdrental veritabanında bulunan tabloları listelemek için:**
`\dt`
- **Herhangi bir tablonun sütunlarını ve tablo detaylarını görmek için:**
`\d <tablo_adi>`
- **PSQL terminal ekranından çıkmak için:**
`\q`