



دانشگاه صنعتی امیرکبیر  
(پلی تکنیک تهران)  
دانشکده ریاضیات و علوم کامپیوتر

## Database

نگارش  
آرمان صالحی، پارسا پورقاسمی

استاد درس  
دکتر قربانعلی

سر تدریسیار  
آریان فتحی

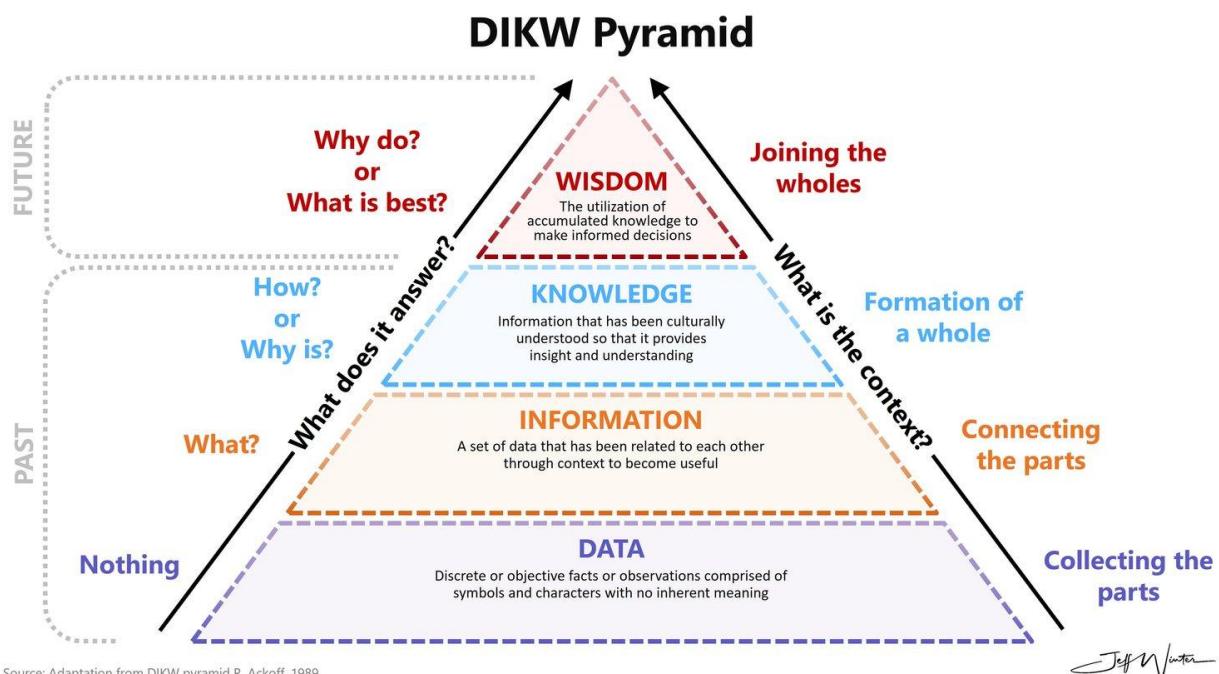
۱۴۰۴ بهار

## داده چیه؟

بین، داده (Data) یعنی یسری چیزای خام و پردازش نشده. چیزایی مثل یه عدد ساده، اسم یه شهر، یا یه تاریخ. مثلاً «۸۰/۰۹/۱۴۰۳» یا «تهران» یا «اینا به تنها ی هیچ معنی خاصی ندارن. فقط یه چیزای ثبت شده که هنوز باهاشون کاری نکردیم. حالا وقتی این دادهها رو کنار هم بذاریم و یه کم روشون فکر یا پردازش کنیم، اون موقع تبدیل می شن به اطلاعات (Information). اطلاعات یعنی دادههایی که معنی دار شدن و می توانیم ازشون یه نتیجه بگیریم یا یه چیزی رو بهتر بفهمیم. مثلاً وقتی میگیم دمای تهران در روز «۱۴۰۳/۰۹/۲۱» برابر با «۸ درجه» بود.. اینجا دیگه داریم اطلاعات می دیم، نه فقط داده خام.

حالا برسیم به مدل DIKW که یه جور هرم چهار مرحله ایه:

۱. (داده): همون چیزای خام و بی تحلیل.
۲. (اطلاعات): وقتی دادهها رو پردازش می کنی و معنیشون مشخص می شه.
۳. (دانش): وقتی اطلاعات رو توی یه زمینه هی خاص می ذاری و ارش الگو و رابطه در میاری.
۴. (خرد): این دیگه تهشه! وقتی از دانش استفاده می کنی تا توی موقعیت های سخت، درست تصمیم بگیری

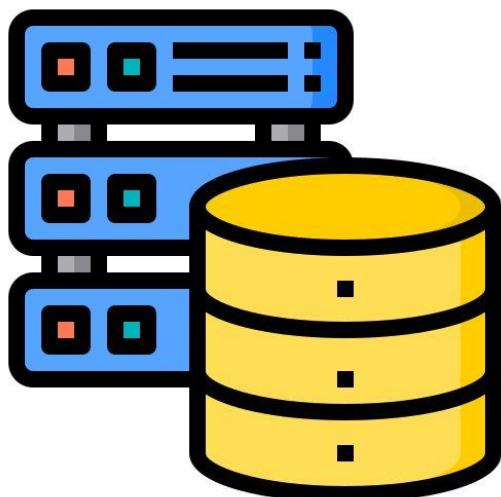


یعنی بطور خلاصه:

اول داده جمع می‌کنی، بعد باهاشون اطلاعات می‌سازی، اون اطلاعات رو تحلیل می‌کنی تا تبدیل بشه به دانش، و در نهایت با تجربه و فهم عمیق، اون دانش رو به خرد تبدیل می‌کنی.

## پایگاه داده چیه؟

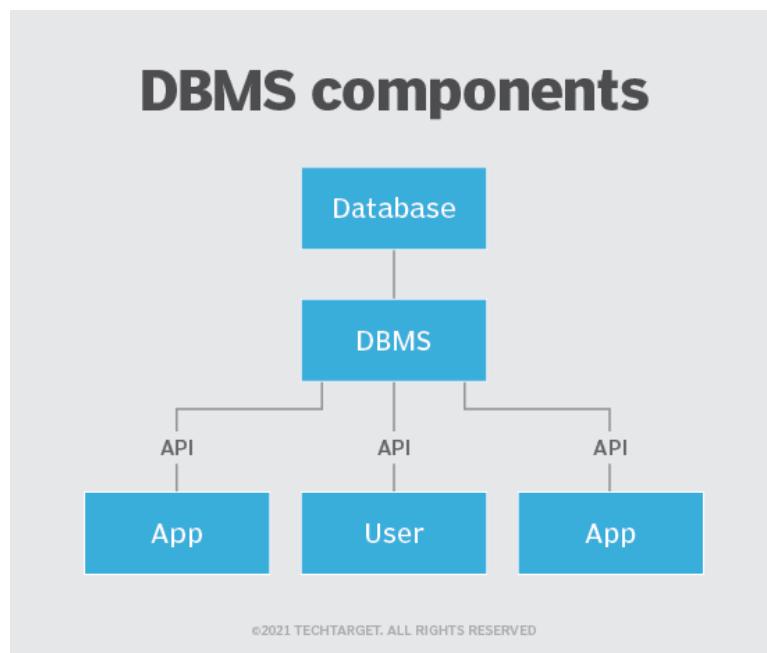
یه پایگاه داده (Database) در واقع یه جاییه که کلی داده رو به صورت منظم و مرتب نگه می‌داریم تا هر وقت خواستیم، راحت بتونیم بهشون دسترسی داشته باشیم، اونا رو ویرایش کنیم یا چیزی ازشون دربیاریم. فکر کن یه عالمه داده مثل «تهران»، «۱۴۰۳/۰۹/۲۱» و «۸ درجه» رو همینطوری پراکنده و بنظم ذخیره کرده باشیم. خب اگه بخوایم بعداً پیدا شون کنیم یا استفاده شون کنیم، کلی دردسر داریم. ولی وقتی اینا رو توی یه پایگاه داده ذخیره می‌کنیم، مثلاً توی یه جدول که ستون هاش «نام شهر»، «تاریخ» و «دما» هستن، اون وقت همه چی سر جاشه و با یه جستجوی ساده می‌تونیم پیدا شون کنیم یا تغییر شون بدیم .



## چیه ؟ DBMS

DBMS یا همون سیستم مدیریت پایگاه داده یه نرم افزاره که کمک می کنه بتونی اطلاعات رو خیلی مرتب و اصولی ذخیره کنی، بخونی، ویرایش کنی یا پاکشون کنی. یه جورایی مثل یه مغز منظم که اطلاعات رونگه می داره و هر وقت خواستی، خیلی راحت بهت تحويل می ده.

مثلاً فرض کن یه برنامه نوشته برای ثبت اطلاعات دانشجوها، نمره ها، استادها و ... DBMS میاد مثل یه مدیر منظم، همه این اطلاعات رو توی یه جای امن نگه می داره و بهت امکان می ده خیلی راحت باهاشون کار کنی.



### چه کارایی با DBMS می تونیم بکنیم؟

۱. **تعريف داده ها :** یعنی ساختن جدولها و مشخص کردن اینکه چه نوع اطلاعاتی می خوای ذخیره کنی.
۲. **ثبت و تغییر داده ها :** اطلاعات جدید وارد کن، یا قدیمیا رو تغییر بدیه یا پاک کن.
۳. **گرفتن اطلاعات :** می تونی خیلی سریع از بین میلیون ها داده، دقیقاً اون چیزی که می خوای رو پیدا کنی.
۴. **مدیریت کاربران :** کنترل اینکه کی به چی دسترسی داشته باشه، امنیت و اینا.

## ویژگی‌های باحال DBMS

- اطلاعات رویه‌جا و منظم ذخیره می‌کنه.
- امنیت داره، کسی بی‌اجازه نمی‌تونه دست بزنه.
- امکان بکاپ گرفتن داره؛ اگه یه وقت سیستم پرید، اطلاعات نپره.
- از قانون‌های خاصی پیروی می‌کنه که اطلاعات همیشه درست و سالم بموون (ACID معروف).
- می‌تونی با رابط گرافیکی یا حتی از طریق برنامه‌نویسی باهاش ارتباط بگیری.

## اجزای اصلی DBMS

۱. موتور ذخیره‌سازی : داده‌ها رو روی دیسک ذخیره می‌کنه.
۲. کاتالوگ متادیتا : مثل یه دفترچه یادداشت که می‌نویسه چه جدول‌هایی داری، چه نوع داده‌هایی تو شونه و غیره.
۳. زبان دسترسی به داده : مثل SQL که می‌تونی باهاش اطلاعات رو بخونی، بنویسی یا حذف کنی.
۴. موتور بهینه‌سازی : کوئری‌ها (دستورهای شما) رو به بهترین شکل ممکن اجرا می‌کنه.
۵. قفل‌گذاری : نذاره چند نفر همزمان یه داده رو دستکاری کنن و قاطی پاتی شه.
۶. مدیر گزارشات : هر تغییری که اتفاق می‌افته رو ثبت می‌کنه که بعداً اگه مشکلی پیش او مده، بشه بازیابی کرد.
۷. ابزارهای مدیریتی : برای بکاپ، بازیابی، بررسی صحت اطلاعات و این جور چیزهایی.

## مزایای استفاده از DBMS

- اطلاعات تکراری کمتر می‌شن.
- چند نفر می‌توانن همزمان ازش استفاده کنن.
- نگهداری و آپدیت راحت‌تره.
- امنیتش خوبه.
- از اطلاعات بکاپ می‌گیره.
- رابط کاربری داره که کار رو آسون می‌کنه.

## معایبش چیه؟

- هزینه سخت‌افزار و نرم‌افزارش ممکنه بالا باشه.
- حجم زیادی از حافظه رو می‌گیره.
- یه کم پیچیده‌ست و کار کردن باهاش نیاز به یادگیری داره.
- اگه کل سیستم بپره و بکاپ نداشته باشی، همه چیز می‌رده رو هوا

## RDBMS چیه؟

RDBMS که مخفف Relational Database Management System هستش یا همون پایگاهداده رابطه‌ای، یه نوع سیستم ذخیره‌سازی اطلاعاته که داده‌ها رو توی جدول‌هایی با سطر (row) و ستون (column) نگه می‌داره. اینجوری خیلی راحت می‌تونی بفهمی چه داده‌هایی با هم مرتبطان و چه جوری به هم وصل می‌شن.

تو این مدل، رابطه‌ها (relationships) همون ارتباط‌های منطقی‌ان که بین جدول‌های مختلف ایجاد می‌کنی تا بفهمی مثلًا "کدوم سفارش برای کدوم مشتریه؟" یا "این کتاب رو کدام نویسنده نوشته؟"

### تعريف دقیق‌تر پایگاهداده رابطه‌ای:

پایگاهداده رابطه‌ای (RDB) داده‌ها رو توی جدول‌هایی ذخیره می‌کنه که هر جدول شامل ردیف‌ها و ستون‌هاست. ویژگی خاص این مدل اینه که می‌تونی بین جدول‌های مختلف، ارتباط برقرار کنی. مثلًا با استفاده از یه ستون مشترک، دو تا جدول رو به هم وصل کنی و بفهمی که داده‌ها چه ربطی به هم دارن.

### مدل پایگاهداده رابطه‌ای از کجا اومند؟

توی دهه ۷۰ میلادی، یه نفر به اسم E.F. Codd از شرکت IBM این مدل رو پیشنهاد داد. اون زمان، بیشتر سیستم‌ها از ساختارهای درختی و سلسله‌مراتبی استفاده می‌کردند، ولی کاد گفت که بهتره از جدول استفاده کنیم تا بتونیم داده‌ها رو راحت‌تر و منعطف‌تر مدیریت کنیم.

فکرش رو بکن مثل اینه که کلی فایل اکسل داری، هر فایل یه جدول داده‌ست و این جدول‌ها می‌تونن از طریق یه ستون مشترک به هم وصل بشن.

### ساختار جدول‌ها در مدل رابطه‌ای چطوریه؟

هر جدول شامل:

• ستون (Column) : مشخص می‌کنه که چه نوع داده‌ای قراره اینجا ذخیره بشه، مثلًا اسم، سن، شماره...

• سطر (Row / Record / Tuple) : مقدار واقعی اون داده‌ها رو نگه می‌داره، مثلًا "محمد - ۲۵ - ۹۱۲..."

## کلیدها در پایگاهداده رابطه‌ای:

• **Primary Key**: یه شناسه‌ی خاص که هر ردیف با اون از بقیه جدا می‌شه (مثلاً کدملی یا شماره دانشجویی).

• **Foreign Key**: یه ستون که به کلید اصلی یه جدول دیگه اشاره می‌کنه و با اون ارتباط برقرار می‌کنه.

## نمونه‌هایی از سیستم‌های مدیریت پایگاهداده رابطه‌ای (RDBMS) :

برای ساختن، به روزرسانی، و مدیریت دیتابیس‌های رابطه‌ای از برنامه‌هایی به اسم RDBMS استفاده می‌کیم.

معروف‌ترین‌هاش اینا هستن:

• MySQL

• PostgreSQL

• MariaDB

• SQL Server

• Oracle Database

•

## پایگاهداده‌های رابطه‌ای ابری:

توی فضای ابری هم مدل‌های رابطه‌ای وجود دارن که به‌شکل مدیریت‌شده (Managed Service) ارائه می‌شن، مثل:

• Cloud SQL

• Cloud Spanner

• AlloyDB

این سرویس‌ها کلی کار مثل نگهداری، به‌روزرسانی، بکاپ گرفتن و... رو خودشون انجام می‌دن، و تو فقط با داده‌هات کار می‌کنی!

## پایگاهداده NoSQL چیه؟

پایگاهدادههای NoSQL که بهشون غیررابطه‌ای هم می‌گن برای یه هدف خاص طراحی شدن و برخلاف پایگاهدادههای سنتی، ساختار خشک و ثابت ندارن. یعنی باهاشون می‌تونی خیلی راحت‌تر و آزادتر داده‌هات رو ذخیره کنی و اپلیکیشن‌های مدرن‌تری بسازی.

NoSQL‌ها به خاطر راحتی استفاده، سرعت بالا، و عملکرد خوبشون توي حجم بالا، کلی طرفدار دارن.

### چرا باید از NoSQL استفاده کنیم؟

برای خیلی از اپ‌های امروزی مثل برنامه‌های موبایل، وب‌سایت‌ها، و حتی بازی‌ها، استفاده از NoSQL گزینه‌ی خیلی خوبیه، چون:

#### انعطاف‌پذیرن:

ساختار این دیتابیس‌ها انعطاف‌پذیره، یعنی لازم نیست از اول بگی چه ستونی داری و همه‌چی رو دقیق مشخص کنی. هر موقع خواستی می‌تونی تغییرش بدی و این باعث می‌شه توسعه‌ی اپلیکیشن سریع‌تر و راحت‌تر پیش بره. واسه همین برای داده‌های نصفه‌نیمه یا بدون ساختار (مثل یه پست شبکه اجتماعی، پیام کاربر، عکس و...) عالی‌ان.

#### مقیاس‌پذیرن:

NoSQL‌ها جوری طراحی شدن که به جای خریدن یه سرور قوی و گرون، می‌تونی چند تا سرور معمولی رو به هم وصل کنی و با هم ازشون استفاده کنی اینطوری هم ارزون‌تر درمیاد، هم عملکردش تو حجم بالا بهتره. توی فضای ابری هم معمولاً این کارا اتوماتیک انجام می‌شن.

#### سریع‌ان:

چون برای مدل‌های خاصی از داده طراحی شدن، توی اون کار مخصوصشون خیلی سریع‌تر از پایگاهدادههای رابطه‌ای عمل می‌کنن.

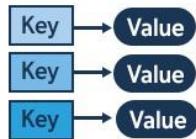
#### قابلیت‌های زیاد دارن:

هر نوع NoSQL با توجه به نوع داده‌ای که نگه می‌داره، یه سری قابلیت‌ها و API‌های مخصوص خودش رو داره که خیلی هم کار راه‌اندازان.

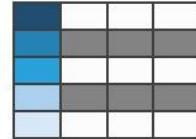
# انواع پایگاهداده‌های NoSQL

## NoSQL

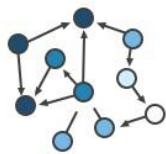
### Key-Value



### Column-Family



### Graph



### Document



چهار نوع اصلی NoSQL که هرکدام برای یه جور کار طراحی شدن:

### ۱- پایگاهداده‌های سندی (Document-based)

داده هارو به شکل سند (Document) ذخیره میکنن، درست مثل فایل های JSON. مثلاً اطلاعات یه کاربر رو با اسم، سن، شماره تماس، آدرس و... توی یه سند می‌ذارن.

مناسب برای: سیستم‌هایی مثل فروشگاه اینترنتی، وبسایت‌ها، بلاگ‌ها، و...

Document 1

```
{  
  "_id": "buttercreambirthdaycake",  
  "name": "Buttercream Birthday Cake",  
  "ingredients": ["cake", "birthday spirit", "cream"],  
  "levelOfMixins": "high",  
  "creaminess": "medium"  
}
```

Document 2

```
{  
  "_id": "greenmintchip",  
  "name": "Green Mint Chip",  
  "ingredients": ["mint", "chocolate chips"],  
  "levelOfMixins": "medium",  
  "creaminess": "high",  
  "description": "Peppermint cream with crunchy,  
  bittersweet chocolate.",  
  "brand": "Jeni's Ice Cream",  
  "popularity": 10,  
  "availableCities": ["Atlanta", "Austin", "Chicago"]  
}
```

## ۲- پایگاهداده کلید-مقدار (Key-Value)

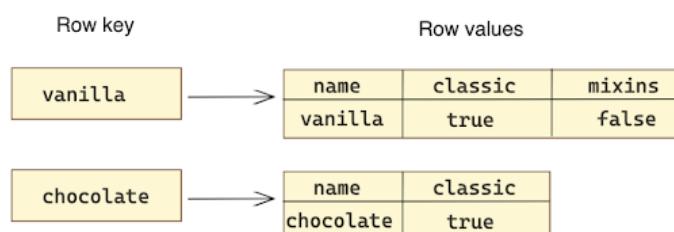
ساده‌ترین نوعه. هر چیزی را با یه کلید (Key) ذخیره می‌کنی، و برash یه مقدار (Value) تعیین می‌کنی. مثلًا:

Key	Value
Name	Raman Sharma
Account Number	21657243
Amount	567543

## ۳- ذخیره‌سازی ستونی (Wide-Column Stores)

داده‌ها رو توی جدول ذخیره می‌کنن، ولی ستون‌هاش انعطاف‌پذیر و دینامیکه. مثلًا هر ردیف می‌تونه ستون‌های مخصوص خودش رو داشته باشه.

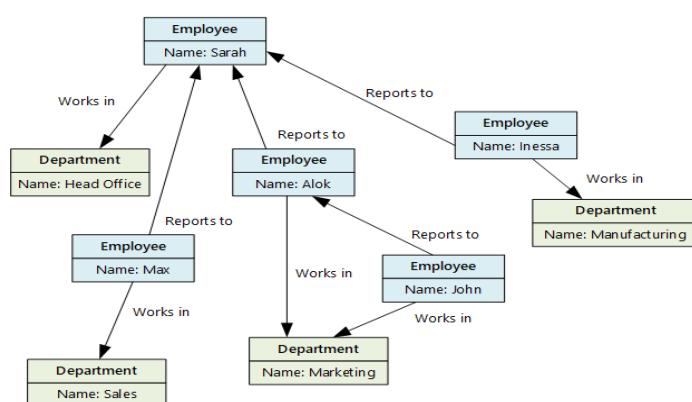
مناسب برای: آنالیز داده، سیستم‌های گزارش‌گیری با داده‌های حجمی



## ۴- پایگاهداده گرافی (Graph Databases)

داده‌ها رو به شکل نود (گره) و ارتباط بین گره‌ها (Edge) ذخیره می‌کنه. برای مثال، نودها می‌تونن افراد باشن و ارتباط بینشون "دوست بودن" یا "همکار بودن" و... باشه.

مناسب برای: شبکه‌های اجتماعی، موتورهای پیشنهاددهنده (مثل سایت‌های فیلم یا خرید)



## کی باید از NoSQL استفاده کیم؟

وقتی بخوای تصمیم بگیری که از چه دیتابیسی استفاده کنی، اگه یکی از شرایط زیر رو داری، NoSQL گزینه‌ی خوبیه:

- توسعه سریع با تیم چابک(Agile)
- کار با داده‌های ساختاریافته یا نیمه‌ساختاریافته
- حجم خیلی زیاد داده
- نیاز به مقیاس‌پذیری افقی (یعنی اضافه کردن سرورهای بیشتر به جای ارتقای سخت‌افزار)
- استفاده از معماری‌های مدرن مثل **real-time data streaming** یا **microservices**

## چیه؟ SQL

یا همون Structured Query Language SQL یه زبان برنامه‌نویسیه که برای ذخیره و پردازش اطلاعات توی پایگاه‌داده‌های رابطه‌ای استفاده می‌شه. با SQL می‌تونی اطلاعات رو توی دیتابیس ذخیره کنی، به روزرسانی یا حذفشون کنی، دنبال یه چیزی بگردی یا داده‌ها رو بازیابی کنی. حتی می‌تونی باهاش عملکرد دیتابیس رو هم بهینه‌سازی و مدیریت کنی.

## چرا از SQL استفاده می‌کنیم؟

یه زبان خیلی پرکاربرده که تقریباً توی همه‌ی نوع‌های برنامه‌نویسی و پروژه‌ها استفاده می‌شه. برنامه‌نویس‌ها و تحلیل‌گرهای داده یاد می‌گیرند چون خیلی راحت با زبان‌های دیگه ترکیب می‌شه. مثلاً می‌تونی توی برنامه‌های جاوا از کوئری‌های SQL استفاده کنی تا با دیتابیس‌هایی مثل Oracle یا Microsoft SQL Server کار کنی.

از طرف دیگه، یاد گرفتنش هم آسونه چون بیشتر دستورهاش از کلمات انگلیسی ساده مثل SELECT, DELETE, UPDATE وغیره ساخته شده.

## چطوری کار می‌کنی؟ SQL

وقتی یه کوئری SQL اجرا می‌کنی (مثلاً می‌خوای یه سری اطلاعات از دیتابیس بگیری)، این درخواست از چند مرحله‌ی نرم‌افزاری مختلف عبور می‌کنه تا جواب نهایی برگردد:

### 1. Parser

اول از همه، یه چیزی به اسم parser وارد عمل می‌شه. اون میاد متن دستور SQL رو بررسی می‌کنه و کلمات خاص رو تبدیل به نشونه‌هایی می‌کنه که راحت‌تر فهمیده بشن. بعدش چک می‌کنه که دستور از نظر قواعد درست باشه. مثلاً بررسی می‌کنه که ته دستور ؛ گذاشتی یا نه. اگه مشکلی باشه، خطای می‌دهد.

علاوه بر اون، parser بررسی می‌کنه که آیا کاربری که این کوئری رو فرستاده مجاز هست یا نه. مثلاً فقط مدیر سیستم اجازه داره اطلاعات رو حذف کنه.

### 2. Relational Engine

این قسمت، مثل مغز اصلیه. میاد یه برنامه‌ی کارآمد می‌نویسه که چطور اطلاعات مورد نظر رو پیدا کنه، بنویسه یا تغییر بده. اگه کوئری قبل‌اگرا شده باشه، ممکنه از همون روش قبلی استفاده کنه. این برنامه‌ی میانی به شکل یه زبان سطح پایین‌تر به نام byte code نوشته می‌شه تا اجراش سریع‌تر باشه.

### 3. Storage Engine

این بخش، byte code رو اجرا می‌کنه و مستقیماً با هارد دیسک یا فایل‌های دیتابیس کار می‌کنه. یعنی داده‌ها رو از روی دیسک می‌خونه، یا اطلاعات جدید ذخیره می‌کنه. وقتی کارش تمام شد، نتیجه رو به برنامه‌ای که درخواست داده بود، برمی‌گردونه.

## SQL چجوری کار می‌کنه؟

SQL مخصوص کار با داده‌های ساخت‌یافته‌ست؛ یعنی داده‌هایی که بینشون رابطه وجود داره. مثلاً فرض کن اطلاعات مشتری‌ها، سفارش‌ها و محصولات رو داریم که همه‌شون یه‌جوری به هم وصلن SQL. اینا رو توی جدول‌هایی ذخیره می‌کنه که هر جدول یه سری ستون (ویژگی) و ردیف (رکورد) داره. چون ساختار این دیتابیس از قبل مشخصه، باید قبل از وارد کردن دیتا، بدونی قراره چه چیزی ذخیره کنی و چطوری به هم ربط دارن.

مقیاس‌پذیری (Scalability) :

SQL معمولاً به صورت عمودی مقیاس‌پذیره؛ یعنی برای اینکه بتونی فشار بیشتری رو تحمل کنی، باید سرور قوی‌تری بگیری مثلاً CPU یا رم یا حافظه‌ی SSD رو بیشتر کنی. البته بعضی وقتاً می‌تونن به صورت افقی هم مقیاس‌پذیر باشن (با شارد کردن یا پارتیشن‌بندی)، ولی این روش توی SQL خیلی قوی یا راحت نیست.

ساختار:

SQL از یه ساختار جدولی استفاده می‌کنه. قبل از اینکه شروع کنی به کار با دیتا، باید دقیقاً مشخص باشه که هر جدول چی داره، چه ستون‌هایی داره و چه نوع داده‌هایی قراره توش ذخیره بشه. این باعث می‌شه نظم خوبی داشته باشه ولی انعطاف‌پذیری از NoSQL هستش.

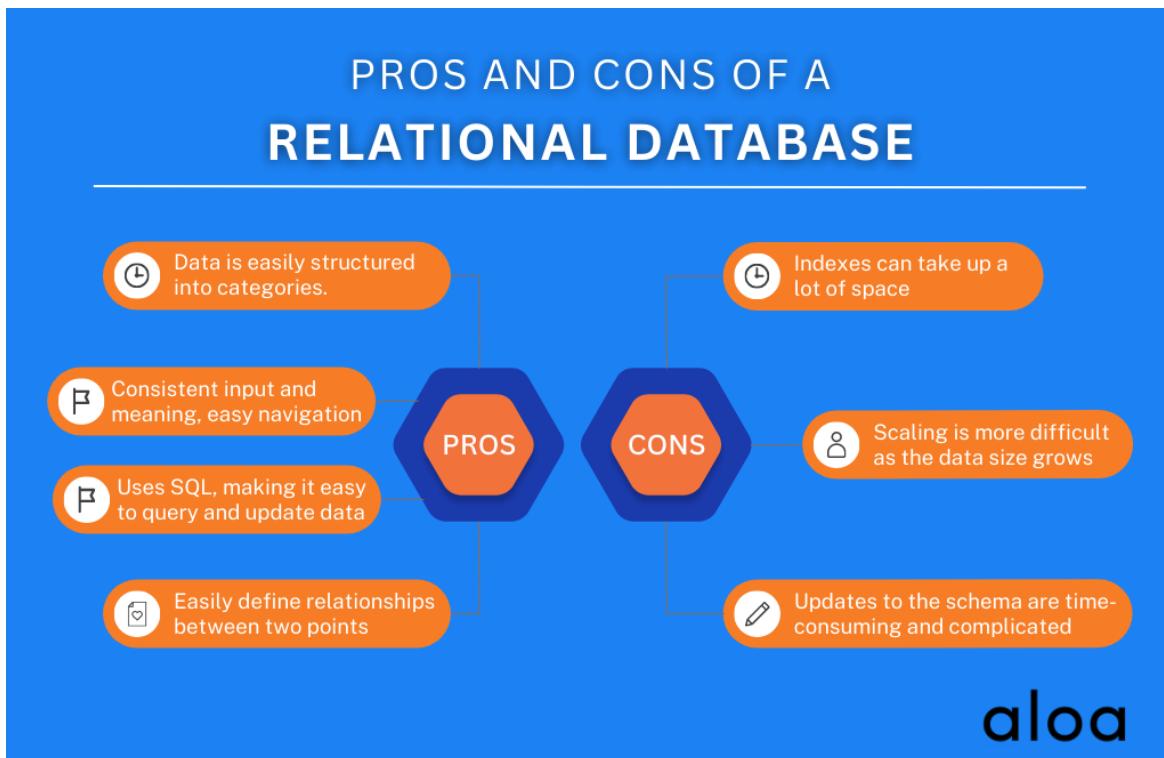
ویژگی‌های ACID:

پایگاه‌داده‌های SQL باید چهار ویژگی معروف به ACID رو داشته باشن:

- **Atomicity (اتمی بودن)** : یا تراکنش کامل انجام می‌شه یا اصلاً انجام نمی‌شه. نصفه‌نیمه نداریم!
- **Consistency (سازگاری)** : دیتابیس همیشه باید در یه حالت درست و معتبر باقی بمانه.
- **Isolation (ایزوله بودن)** : وقتی چند تا تراکنش همزمان انجام می‌شن، نباید روی هم تأثیر بذارن.
- **Durability (دوم)** : وقتی یه تراکنش کامل شد، حتی اگه سیستم خاموش بشه یا قطع بشه، اون اطلاعات از بین نمی‌ریزد.

پشتیبانی و جامعه کاربری:

چون SQL خیلی وقت و وجود دارد، جامعه‌ی بزرگی از برنامه‌نویس‌ها و منابع آنلاین دارد. کلی مثال و مستندات وجود دارد و راحت می‌توانی جواب سؤالات توپیدا کنی.



## چجوری کار می‌کنه؟ NoSQL

برخلاف SQL، NoSQL نیازی به ساختار از پیش تعریف شده ندارد. یعنی می‌توانی بدون اینکه از اول بدونی قراره چه اطلاعاتی ذخیره کنی، دیتا رو واردش کنی. به همین خاطر برای داده‌های بی‌ساختار یا نیمه‌ساختاری خیلی مناسب تره. توی NoSQL می‌توانی به راحتی ویژگی‌های جدید به داده‌هات اضافه کنی و حتی نوع داده‌ها توی رکوردهای مختلف هم می‌تونه فرق کنه.

مقیاس‌پذیری:

NoSQL خیلی راحت به صورت افقی مقیاس‌پذیره؛ یعنی به جای اینکه یه سرور قوی‌تر بخری، می‌توانی سرورهای بیشتری به سیستم اضافه کنی. واسه همین برای پروژه‌هایی با دیتای خیلی زیاد یا رشد سریع، گزینه‌ی خیلی خوبی.<sup>۵</sup>.

ساختار:

- پایگاهداده‌های NoSQL انواع مختلفی دارند:
- ستونی (Column-based) : داده‌ها به جای ردیف، بر اساس ستون ذخیره می‌شون. برای تحلیل‌های سریع عالیه.
  - کلید-مقدار (Key-Value) : هر داده با یه کلید ذخیره می‌شده، مثل یه دیکشنری. خیلی ساده و سریعه.
  - سندمحور (Document-based) : داده‌ها توی فرمات‌هایی مثل XML یا JSON می‌شون. هر سند می‌توانه ساختار خودش رو داشته باشه.
  - گراف (Graph) : اطلاعات به صورت گره و رابطه ذخیره می‌شون. برای شبکه‌های اجتماعی و چیزهایی که روابط پیچیده دارن خیلی خوبه.

ویژگی‌های CAP:

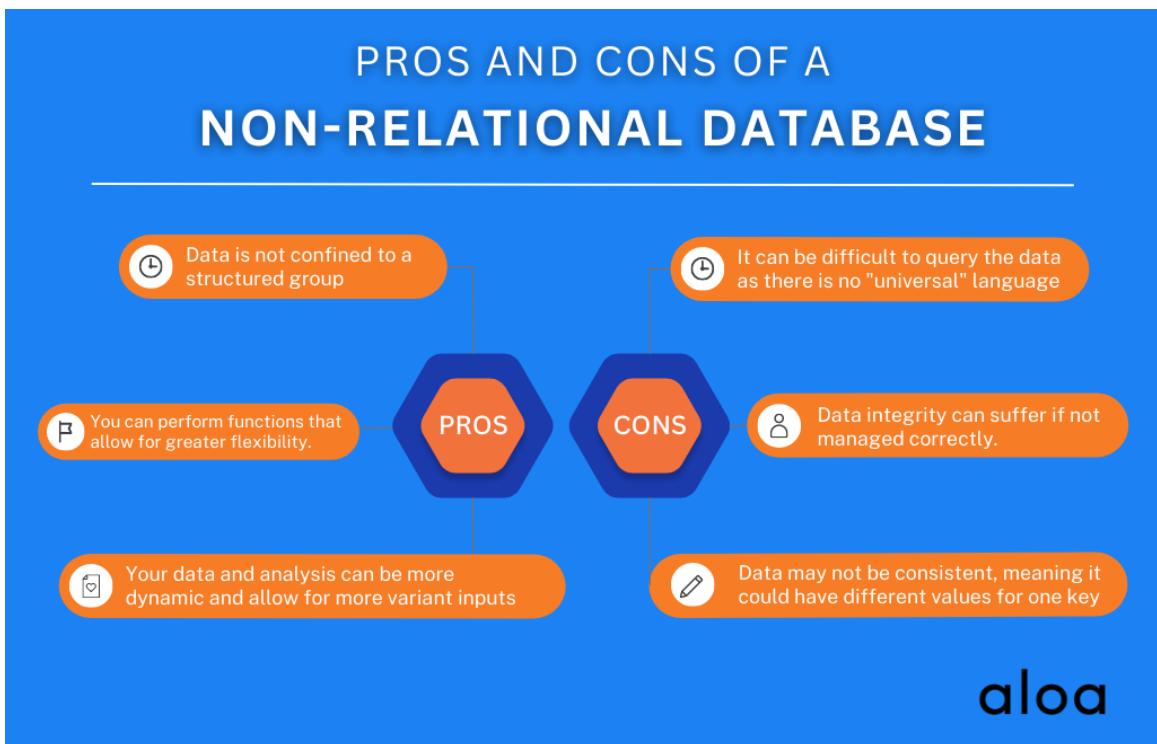
در مقابل ACID توی SQL ، اینجا یه چیزی به نام نظریه CAP داریم که می‌گه دیتابیس‌های توزیع شده فقط می‌تونن از بین این سه ویژگی، دو تاش رو تضمین کنن:

- سازگاری (Consistency) : همه‌ی گره‌ها یا یه دیتا رو می‌دان یا خطأ.
  - دسترسی‌پذیری (Availability) : همیشه یه پاسخ دارن، حتی اگه دقیق‌ترین نباشه.
  - تحمل پارتیشن (Partition Tolerance) : اگه یه بخشی از شبکه قطع بشه، دیتابیس همچنان کار می‌کنه.
- هر دیتابیس NoSQL معمولاً ترکیبی از این سه‌تا رو انتخاب می‌کنه، بستگی به اولویت‌ش.

پشتیبانی و جامعه کاربری:

جامعه‌ی NoSQL هنوز کوچیک‌تره نسبت به SQL ، ولی بیشتر پروژه‌ها اپن‌سورس هستن و می‌شه راحت‌تر تغییرشون داد. البته منابع آموزشی و پشتیبانی کمتری نسبت به SQL دارن.

## PROS AND CONS OF A NON-RELATIONAL DATABASE



### چه زمانی از SQL استفاده کنیم؟

موقعی خوبه که با داده‌های مرتبط کار می‌کنی. مثلاً اگه لازم داری بدونی هر سفارش مربوط به کدوم مشتریه و چه محصولی خریده، SQL خیلی جواب می‌ده. چون توی SQL اطلاعات فقط یه‌بار ذخیره می‌شن و بقیه‌جاهای فقط بهش اشاره می‌شه، این باعث می‌شه همیشه دیتاهای بروز باشن و دچارتکرار یا تناقض نشن.

شرکت‌های بزرگی مثل Uber، Airbnb، Netflix و آمازون (در کنار سیستم‌های خودشون) از SQL برای تحلیل و پرس‌وجو استفاده می‌کنن.

### چه زمانی از NoSQL استفاده کنیم؟

اگه سرعت و مقیاس‌پذیری برات مهم‌تر از یکپارچگی داده‌ست، NoSQL انتخاب خوبیه. مخصوصاً وقتی با داده‌های حجمی یا بدون ساختار کار می‌کنی، یا وقتی ساختار اطلاعات ممکنه مدام تغییر کنه. مثلاً برای ذخیره لگ‌ها، کامنت‌ها، شبکه‌های اجتماعی، یا دیتای لحظه‌ای (real-time) خیلی بهتر جواب می‌ده.

دیتابیس‌هایی مثل MongoDB، Redis، CouchDB و Cassandra از انواع معروف NoSQL هستن. شرکت‌هایی مثل گوگل، آمازون و فیسبوک برای هندل کردن حجم عظیم دیتاهاشون از NoSQL هم استفاده می‌کنن.

# آشنایی با دستورات پایه‌ای SQL

همان‌طور که در درس نامه‌ی قبل خواندیم، SQL یک زبان برنامه‌نویسی (نسل چهارم) برای تعامل با پایگاه داده‌های رابطه‌ای است. در این درس نامه، با دستورات اولیه‌ای که برای کار با داده‌ها استفاده می‌شوند، آشنا می‌شویم.

## ایجاد جداول با دستور CREATE

دستور CREATE برای ساختن جداول یا سایر موجودیت‌ها در پایگاه داده استفاده می‌شود. در این دستور، مشخصات ستون‌ها مانند نوع داده و محدودیت‌ها تعریف می‌شوند. برای مثال:

```
1 | CREATE TABLE books (
2 |   id INT PRIMARY KEY AUTO_INCREMENT,
3 |   title VARCHAR(255) NOT NULL,
4 |   author VARCHAR(255),
5 |   published_year INT,
6 |   price DECIMAL(10, 2)
7 | );
```

- مشخص می‌کند که ستون id باید یکتا باشد و نمی‌تواند مقدار خالی داشته باشد.
- باعث می‌شود مقادیر ستون به صورت خودکار افزایش یابد.

## محدودیت یکتایی (Unique Constraint)

محدودیت UNIQUE تضمین می‌کند که مقدارهای یک ستون نمی‌توانند تکراری باشند. یک جدول می‌تواند چندین ستون با محدودیت UNIQUE داشته باشد. برای مثال:

```
1 | CREATE TABLE users (
2 |   id INT PRIMARY KEY,
3 |   username VARCHAR(50) UNIQUE,
4 |   email VARCHAR(255) UNIQUE
5 | );
```

## درج داده با INSERT

برای افزودن یک سطر به یک جدول از دستور INSERT استفاده می‌کنیم. برای مثال:

```
1 | INSERT INTO books (title, author, published_year, price)
2 | VALUES ('The Great Gatsby', 'F. Scott Fitzgerald', 1925, 10.99);
```

## مشاهده داده‌ها با SELECT

برای خواندن داده از جدول، دستور SELECT استفاده می‌شود. برای مثال:

```
1 | SELECT * FROM books;
```

\* تمامی ستون‌ها را بازمی‌گرداند.

همچنین می‌توان ستون‌های خاصی را نیز مشخص کرد:

```
1 | SELECT title, author FROM books;
```

## فیلتر کردن داده با WHERE

شرط‌های خاص را می‌توان با استفاده از WHERE اعمال کرد. برای مثال:

```
1 | SELECT * FROM books WHERE published_year > 2000;
```

همچنین می‌توان از عملگرهایی مانند = ، < ، > و IN نیز استفاده کرد.

## عملگر LIKE

برای جست‌وجوی الگوها در مقادیر متنی استفاده می‌شود. در این عملگر می‌توان از کاراکترهای *wildcard* مانند % و \_ استفاده کرد:

- % : هر تعداد کاراکتر (صفرا یا بیشتر).

- \_ : دقیقاً یک کاراکتر.

مثال ۱: نمایش کتاب‌هایی که عنوان آن‌ها با The شروع می‌شود:

```
1 | SELECT * FROM books WHERE title LIKE 'The%';
```

مثال ۲: نمایش کتاب‌هایی که عنوان آن‌ها دقیقاً شامل ۴ کاراکتر است:

```
1 | SELECT * FROM books WHERE title LIKE '____';
```

## مرتب‌سازی داده با ORDER BY

برای مرتب‌سازی نتایج، از ORDER BY استفاده می‌کنیم. برای مثال:

```
1 | SELECT * FROM books ORDER BY price ASC;
```

• مرتب‌سازی صعودی (پیش‌فرض). ASC

• مرتب‌سازی نزولی. DESC

## به‌روزرسانی داده‌ها با UPDATE

برای تغییر مقادیر موجود در جدول از UPDATE استفاده می‌شود. برای مثال:

```
1 | UPDATE books  
2 | SET price = 12.99  
3 | WHERE title = 'The Great Gatsby';
```

## حذف داده‌ها با DELETE

برای حذف داده‌های موجود در جدول از DELETE استفاده می‌شود. برای مثال:

```
1 | DELETE FROM books WHERE published_year < 1900;
```

## مراقب باشید !

دستور DELETE برای حذف داده‌ها از یک جدول در پایگاهداده استفاده می‌شود. اگر این دستور بدون استفاده از شرط WHERE اجرا شود، تمام رکوردهای جدول موردنظر حذف خواهند شد. این می‌تواند منجر به از دست رفتن جبران‌ناپذیر داده‌ها شود، به خصوص اگر نسخه‌ی پشتیبانی از پایگاهداده وجود نداشته باشد. فرض کنید قصد دارید تنها کتاب‌های منتشرشده قبل از سال ۱۹۵۰ را حذف کنید، اما به اشتباه دستور زیر را اجرا می‌کنید:

```
1 | DELETE FROM books;
```

این اشتباه باعث حذف کامل رکوردهای جدول خواهد شد!