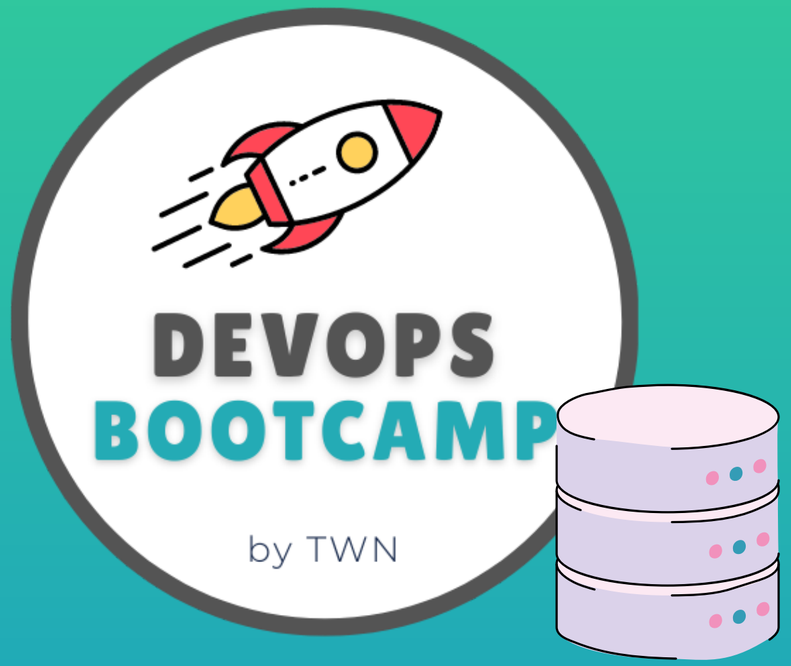


Databases

Key Takeaways

TECHWORLD WITH NANA

Databases are used to **persist data**

- Users can create, read, update or delete entries in a database
- Also called CRUD operations

Database in software development process - 1

- Developers need **database for local development**
  - Connect to a development database to develop the new features
  - Connect to a test database to test the new feature with realistic data
  - ...

Database in software development process - 2

2 ways for developers to work with DB

Option 1

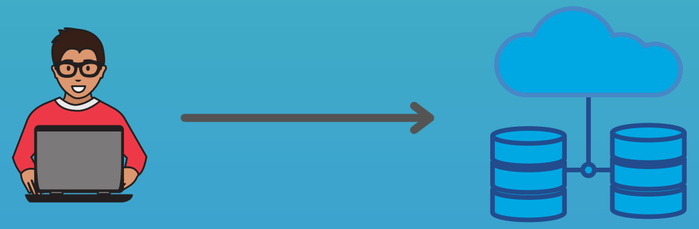
- **Each developer installs** DB **locally**
- Each developer has **own DB with own test data**

- ✓ **can't mess up** someone else's test DB data
- ✗ start from empty DB | need **realistic test data**
- ✗ install and setup DB for EVERY developer

Ideal solution: have both!

Option 2

- Shared DB **hosted remotely**



- ✓ no need for local installation | start coding right away
- ✓ test data available right from the start
- ✗ **can't play around without affecting others**!

TECHWORLD WITH NANA

Configure Database Connection - 1

How does the application talk to the database?

- The DB connection is configured in the application's code
- Each programming language has a library/module for DB connection

- You have to tell the library:

WHICH database to talk to

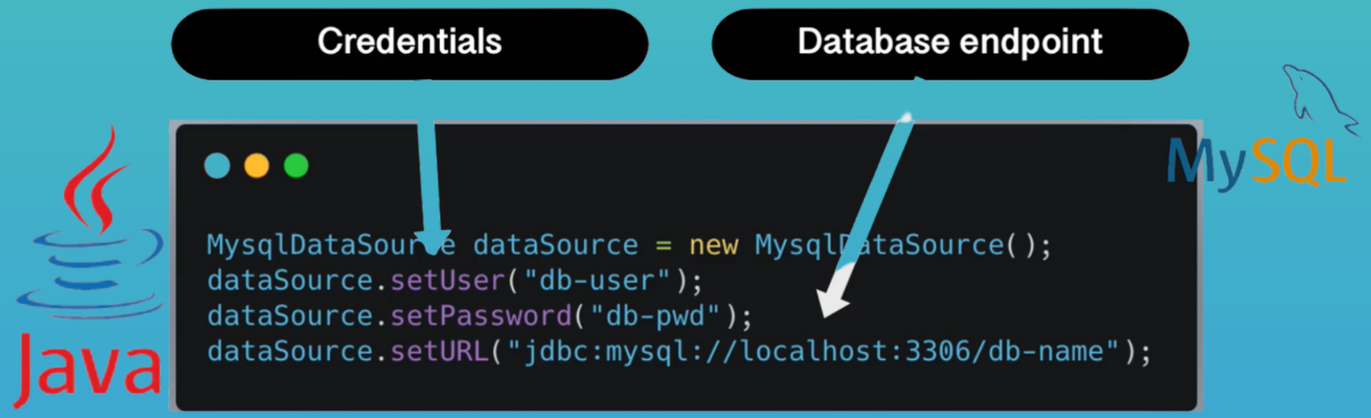
=

Database Endpoint

HOW to **AUTHENTICATE** with that DB

=

Credentials



Example: **Java** Code connecting to a **Mysql** database

TECHWORLD WITH NANA

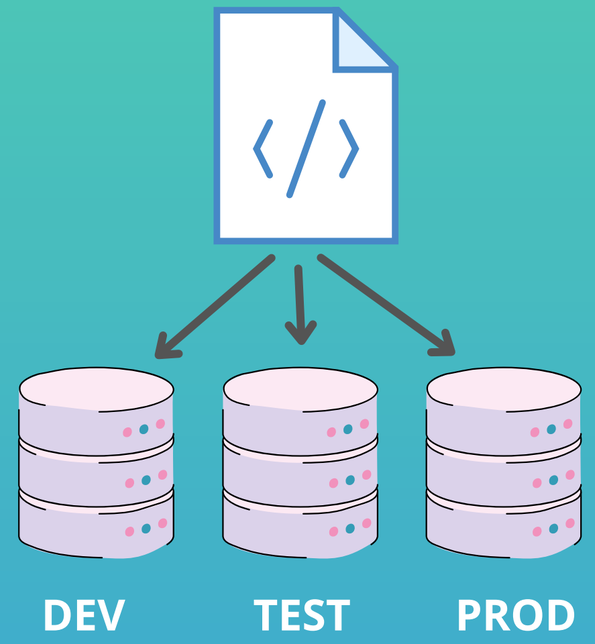
Configure Database Connection - 2

Better: **Do not hardcode** database connection data in source code

Best Practice: **DON'T check in credentials** in code

- **Define in 1 place** as environment variables
- Configure DB connection from OUTSIDE

1. Define only variables in code
2. Set endpoint and credentials from outside for each environment (for example dev, test, prod)
3. Depending on environment (dev, test, prod) connects to different DB!



Configure Database Connection - 3

- The way to do this, is to **pass environment variables on application start-up**

How to pass envrionment variables:

1) From Command Line

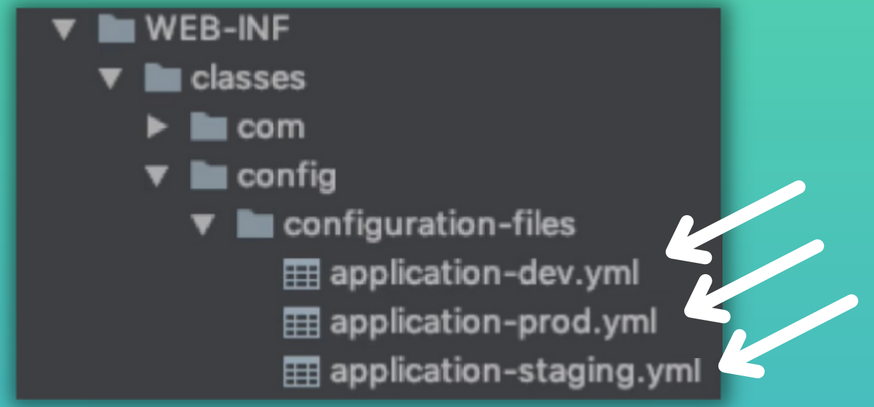
2) Configure in code editor

3) Use properties/configuration files

- The best option is to pass them via a properties or configuration file

Configure Database Connection - 4.1

- In a Java/Spring application you can define the values in a Spring **properties file:**



3 property files for each environment

Reference the value in the Java App

```
datasource:
    type: com.zaxxer.hikari.Hikar
    url: jdbc:postgresql://localh
    username: db-name
    password: db-name
```

Values defined in a properties file

```
import javax.sql.DataSource;

@Configuration
@EnableTransactionManagement
public class DatabaseConfiguration {

    private final Logger log = LoggerFactory.getLogger(DatabaseConfiguration.class);
    private final Environment env;

    public DatabaseConfiguration(Environment env) {
        this.env = env;
    }

    @Bean
    public DSLContext db(DataSource dataSource) {
        return DSL.using(dataSource, SQLDialect.POSTGRES);
    }
```

Usage in Java code

TECHWORLD WITH NANA

Configure Database Connection - 4.2

- Nodejs application with **configuration file:**

config.json

```
{
  "development": {
    "mongodb": {
      "host": "localhost",
      "port": 27017,
      "database": "db-name"
    }
  },
  "staging": {
    "mongodb": {
      "host": "mongo-host-staging",
      "port": 27017,
      "database": "staging-db-name"
    }
  },
  "production": {
    "mongodb": {
      "host": "mongo-host-prod",
      "port": 27017,
      "database": "prod-db-name"
    }
  }
}
```

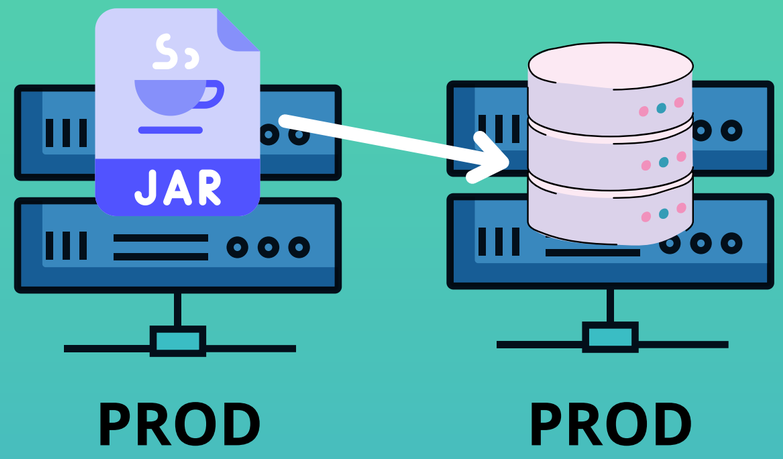
Using a configuration file is:

- ✓ cleaner
- ✓ more transparent

TECHWORLD WITH NANA

Databases in Production

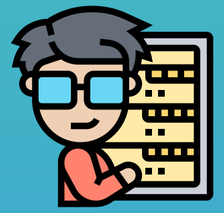
- Application connects to database
- This means database needs to be available , when application starts
- So **before** application is deployed, we need to **install and configure database**



Data is important so we need to secure them:

1. **Replicate** database
2. Do regular **backups**
3. Make sure it **performs under high load**

This may be done by:



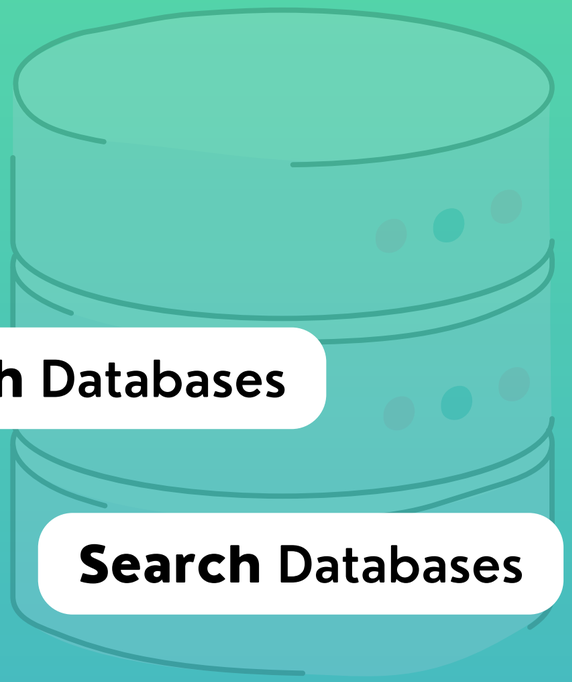
System Administrator

Database Engineer

DevOps Engineer

Database Types

Database Types - 1



Key Value Databases

Relational Databases

Graph Databases

Wide Column Databases

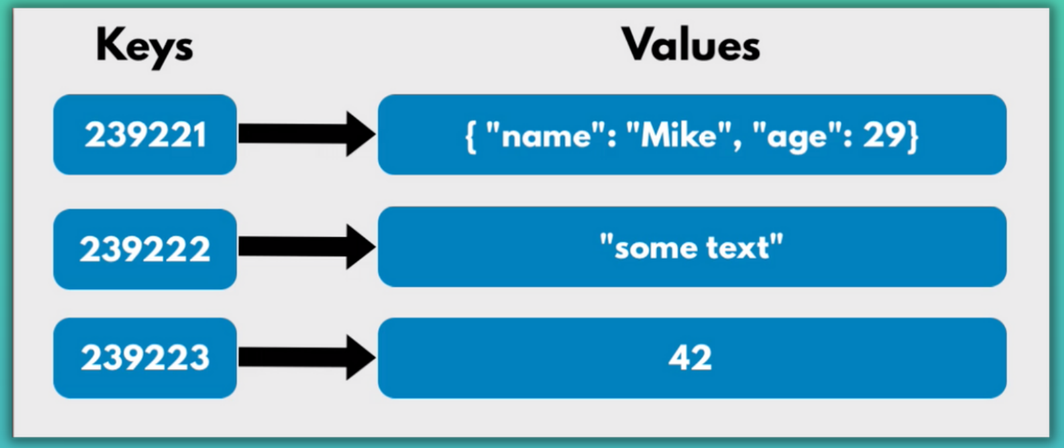
Document Databases

Search Databases

- Many different database types available
- They have **different characteristics for different use cases**
- Each of them has benefits and disadvantages

Database Types - 2

Key Value Databases



Examples:

Redis

Memcached

etcd from Kubernetes

Characteristics

- Data is saved in key-value pairs
- unique key
- no joins
- in-memory

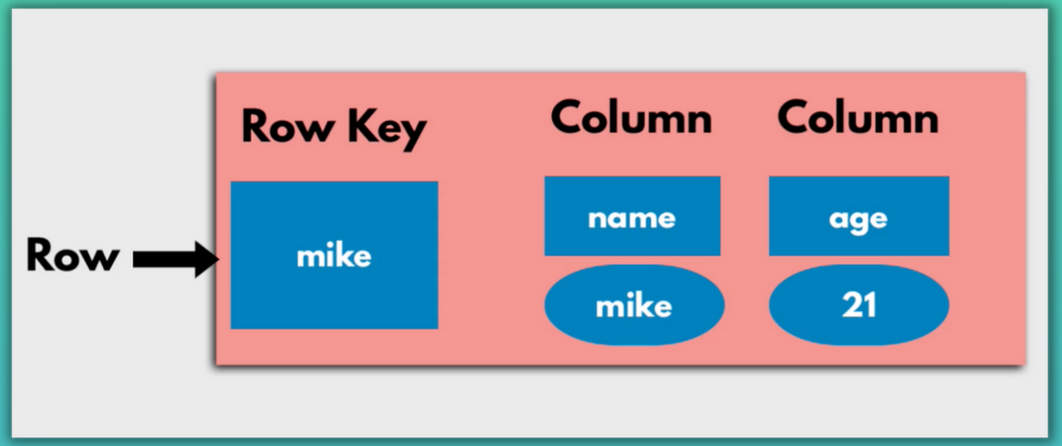
Best for:

- Caching
- Message Queue

- very fast
- limited storage
- no primary database

Database Types - 3

Wide Column Databases



Examples:

cassandra

APACHE HBASE

Characteristics

- A two-dimensional key-value store
- Data is saved in tables, rows and columns
- Names and format can vary from row to row (schema-less)
- Queries similar to SQL

Best for:

- Time-Series
- IoT-Records

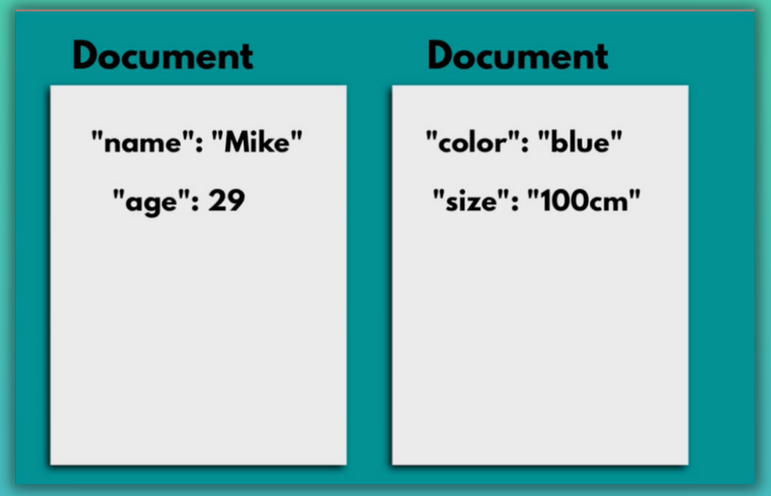
- ✓ scalable
- ✗ no joins
- ✗ limited compared to relational DB
- ✗ no primary DB

TECHWORLD WITH NANA

Database Types - 4

Document Databases

Collection:



Examples:

MongoDB

DynamoDB

CouchDB

Characteristics

- Data is stored as JSON-like documents
- Schema-less
- No joins
- Denormalized

Best for:

- Mobile Apps
- Game Apps
- CMS
- Most Apps

faster to read

easy to get started

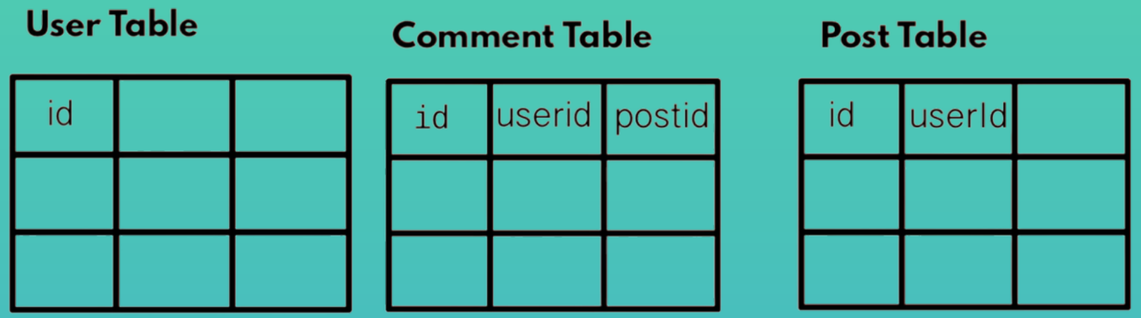
primary database

slower writes

graphs

Database Types - 5.1

Relational Databases



Examples:

MySQL ORACLE PostgreSQL

Characteristics

- Data is stored in Tables, Rows and Columns
- Schema (schema and data types need to be created first)
- Normalized to avoid duplicated data
- Most used in industry

Best for:

- structured data

- ✓ ACID
- ✓ easy to get started
- ✓ primary database
- ✗ difficult to scale
- ✗ unstructured data

Database Types - 5.2

Relational Databases

ACID = Atomicity, Consistency, Isolation, Durability

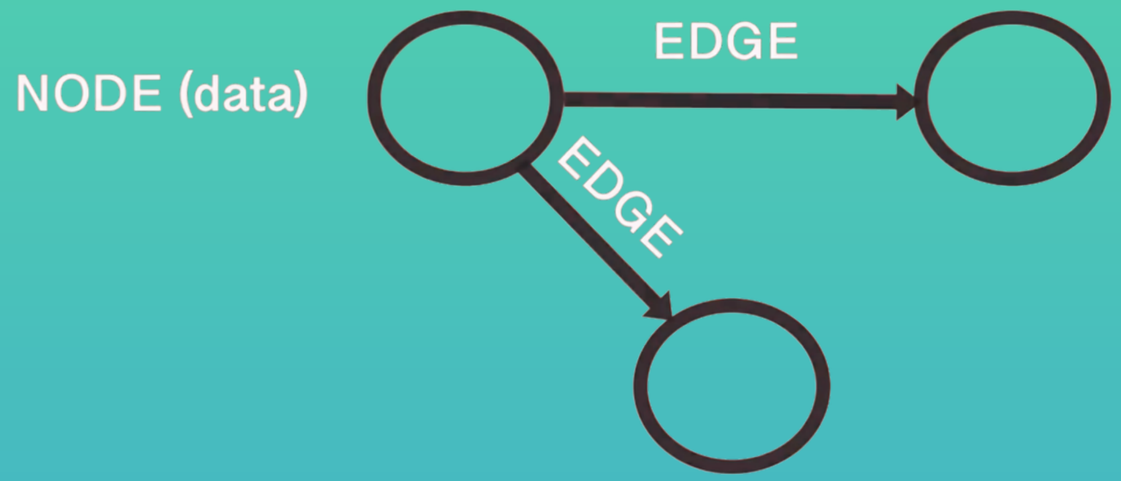
- No half-changes are updated in database
- Either ALL changes get applied or NON

SQL vs No-SQL

Because relational or SQL databases are so popular, often we talk about SQL vs No-SQL, meaning all the other database types, which are schema-less

Database Types - 6

Graph Databases



Examples:

Neo4j

Dgraph

Characteristics

- Data is stored as Nodes and Relationships instead of tables or documents
- Directly connect Entities
- Edges are the Relationships

Best for:

- Graphs
- Patterns

easier to query

TECHWORLD WITH NANA

Database Types - 7

Search-Engine Databases

Characteristics

- Dedicated to the search of data
- Optimized for dealing with data
- Full-text search in efficient and fast way
- Similar to document-orientated database

Examples:

ElasticSearch

Solr

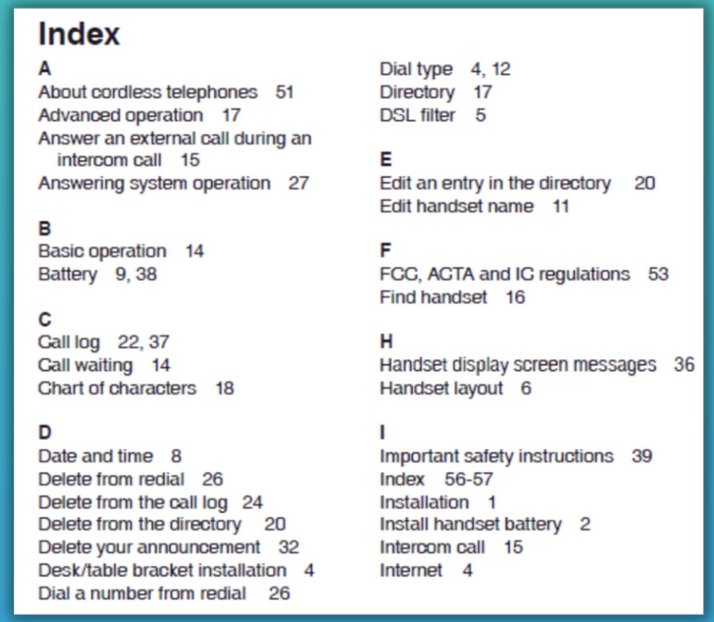
1: *You know nothing*

2: *That's what I do:*
I drink and I know things.

term	freq	documents
and	1	2
drink	1	2
do	1	2
I	3	2
know	2	1, 2

Dictionary | Postings

Usage of **indexes**



Index

A
About cordless telephones 51
Advanced operation 17
Answer an external call during an intercom call 15
Answering system operation 27

B
Basic operation 14
Battery 9, 38

C
Call log 22, 37
Call waiting 14
Chart of characters 18

D
Date and time 8
Delete from redial 26
Delete from the call log 24
Delete from the directory 20
Delete your announcement 32
Desk/table bracket installation 4
Dial a number from redial 26
Dial type 4, 12
Directory 17
DSL filter 5

E
Edit an entry in the directory 20
Edit handset name 11

F
FCC, ACTA and IC regulations 53
Find handset 16

H
Handset display screen messages 36
Handset layout 6

I
Important safety instructions 39
Index 56-57
Installation 1
Install handset battery 2
Intercom call 15
Internet 4

TECHWORLD WITH NANA

Wrap Up

Choose database type **based on** your application **needs**

Or Combinations

For example:

For most of data	→	**Relational**
To handle fast search	→	**Search Engine**
Cache	→	**Key Value**