Large-Scale and Multi-Structured Databases

Guidelines for Selecting a Database

Prof. Pietro Ducange

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

UNIVERSITÀ DI PISA

CROSSLAB Innovation for industry 4.0

From Requirements to the Application

Whenever we are asked to design and develop a specific software application, a set of steps must be performed:

1) ***Requirements elicitation*** from customer

2) ***Requirements definition***, both functional and non functional

3) ***Use case*** definition (better if with UML diagrams)

4) Identification of the main data structures (including the main procedures) and of the relations among them (***Analysis Workflow***)

5) Refinement of 4) (***Project Workflow, including DB Design***)

6) ***Implementation*** and ***test***

7) ***Deploy***

Agile Approach

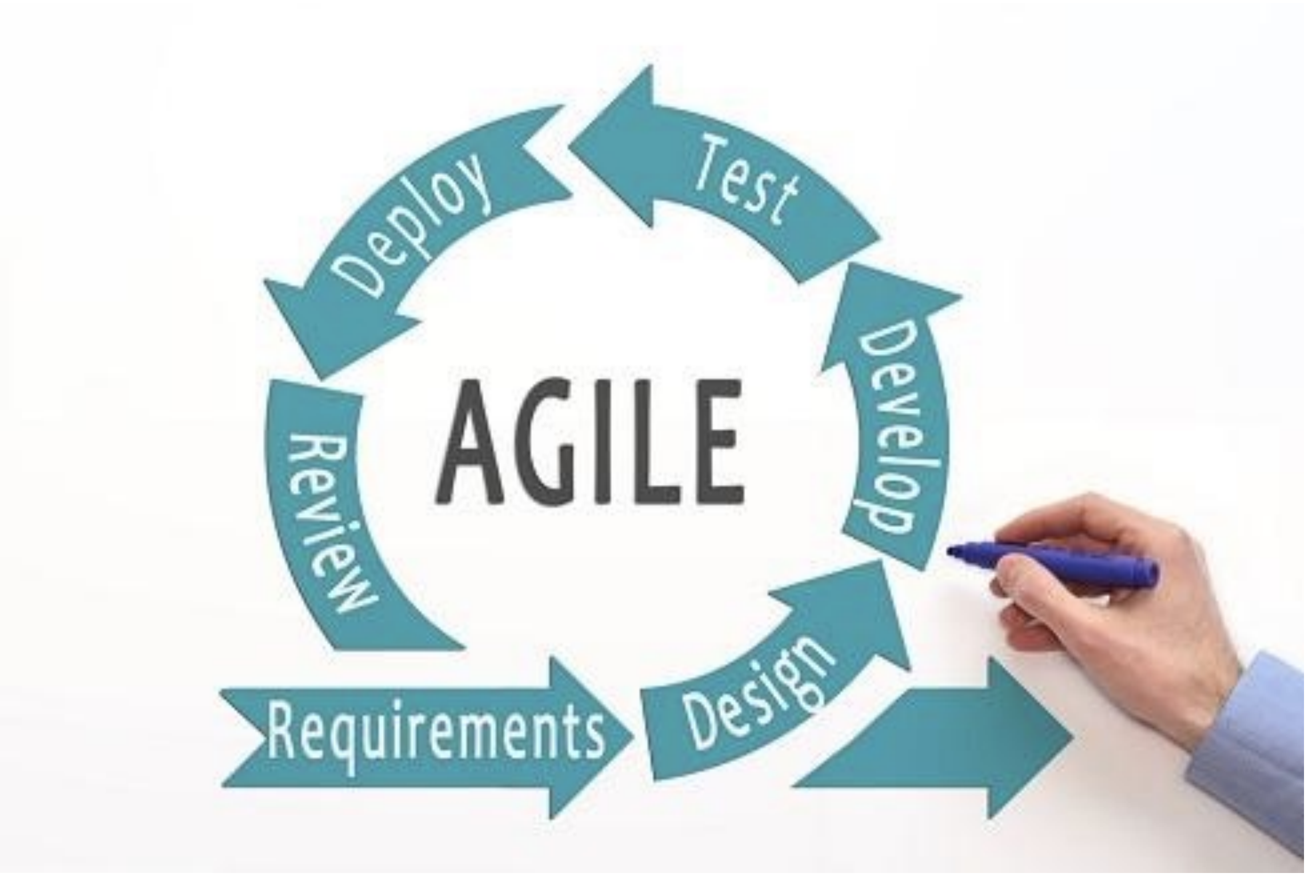

Image extracted from: “https://www.informationweek.com/youre-doing-agile-so-what/a/d-id/1329239

Role of the Architect/Engineer

To Identify the ***most suitable***:

1) Software/Hardware ***architecture***
2) ***Programming languages*** and development environments
3) ***Database management systems***

Everything must be ***driven*** by:

1) Requirements
2) ***Common sense***
3) Experience

Are Relational Databases still useful?

Answer:

Yes, of course! They do the work for which they were designed: protecting ***the data integrity*** and reducing the ***risk of anomalies*** *(immediate consistency and ACID transactions support)* .

- Relational databases ***will continue*** to support ***transaction*** processing systems and, in general, ***OTPL*** applications.

- ***Decades of experience*** with transaction processing systems and data warehouses has led to ***best practices*** and design principles that continue to meet the needs of businesses, governments, and other organizations.

The Modern Application World

Modern data management infrastructure is responsible for a ***wider range*** of applications and data types ***than ever before***.

We are experiencing ***the era of***:

1) Big Data

2) Internet of Things

3) Large scale web-application

4) Mobile application

In this new era, often ***performance*** and ***availability*** are more important than consistency and ACID transactions!

The Available Choices

On the basis of our requirements, analysis and project workflows, we can choice among:

- ***Relational databases***, such as PostgreSQL and MySQL,
- ***Key-value databases***, such as Redis, Riak, and DinamoDB
- ***Document databases***, such as MongoDB and CouchDB
- ***Column family databases***, such as Cassandra and HBase
- ***Graph databases***, such as Neo4j and Titan

Requirements Guide Us

The ***functional requirements*** are related to the type of queries that describe how the data will be used. Moreover, they also influence the ***nature of relations*** among entities and the ***needs of flexibility*** in data models.

When choosing a database we have to analyze other factors (***non functional requirements***) such as:

- ***The volume*** of reads and writes
- ***Tolerance*** for inconsistent data in replicas
- ***Availability*** and ***disaster recovery*** requirements
- ***Latency*** requirements

Key-Values Database

These databases are preferable when:

- The application has to handle ***frequent but small*** read and write operations.
- ***Simple data*** models have been designed
- ***Simple queries*** are needed (no complex queries and filtering on different fields)

These databases do not allow too much flexibility and the possibility of making queries on different attributes.

Key-Values Database

In the following, we show some examples of applications that may exploit these databases:

- ***Caching data*** from relational databases to improve performance
- ***Tracking*** transient attributes in a web application, such as a ***shopping cart***
- ***Storing configuration*** and user data ***information*** for mobile applications
- ***Storing large objects***, such as images and audio files

Document Databases

These type of databases are suitable for applications that need:

- To store ***varying attributes***, in terms of number and type
- To store and ***elaborate large amount of data***
- To make ***complex queries*** (on different types of attribute), to use ***indexing*** and to make ***advanced filtering*** on attributes.

Flexibility is the main feature of document databases, along with high ***performance capabilities*** and ***easy of use***.

Embedded documents allow to use few collections of documents that store together attributes frequently accessed together (***denormalization***).

Document Databases

In the following, we show some examples of applications that may exploit these databases:

- Back-end support for websites with high volumes of reads and writes
- Managing data types with ***variable attributes***, such as products
- Applications that use ***JSON*** data structures
- Applications benefiting from ***denormalization*** by embedding structures within structures

Column Databases

These type of databases are suitable for applications that:

- Require the ability to ***frequently read/write*** to the database
- Are ***geographically distributed*** over multiple data centers
- Can tolerate some ***short-term inconsistency*** in replicas
- Manage ***dynamic fields***
- Actually have to deal with ***huge amount*** of data, such as hundreds of terabytes

Column Databases are normally deployed on clusters of multiple servers on different data centers. If data handled by the application is small enough to run with a single server, then avoid these databases.

Column Databases: Cassandra performance on Google

Setup:

- 330 Google Compute Engine virtual machines,
- 300 1TB Persistent Disk volumes,
- Debian Linux,
- Datastax Cassandra 2.2, triple replication, quorum equal to 2.

Results:

One million writes per second to Cassandra with a median latency of 10.3 ms and 95% completing under 23 ms.

With ⅓ of node failures, the systems maintained the 1 million writes per second (though with higher latency).

Check: https://cloudplatform.googleblog.com/2014/03/cassandra-hits-one-million-writes-per-second-on-google-compute-engine.html for more details.

Column Databases

In the following, we show some examples of applications that may exploit these databases:

- ***Security analytics*** using network traffic and log data mode
- ***Big Science***, such as bioinformatics using genetic and proteomic data
- ***Stock market analysis*** using trade data
- ***Web-scale applications*** such as search
- ***Social network*** services

Graph Databases

These type of databases are suitable for applications that:

- Need to represent their domain as ***networks*** of connected entities

- Handle instances of entities that have ***relations*** to other instances of entities

- Require to ***rapidly traverse*** paths between entities.

Graph Databases: some considerations

- Two ***orders in an e-commerce*** application probably have no connection to each other. They might be ordered by the same customer, but that is a shared attribute, not a connection.

- A ***game*** player's configuration and game state have little to do with other game players' configurations. Entities like these are readily modeled with key-value, document, or relational databases.

- Let consider ***proteins*** interacting with other proteins or ***employees*** working with other employees. In all of these cases, there is some type of connection, link, or direct relationship between two instances of entities.

Graph Databases

In the following, we show some examples of applications that may exploit these databases:

- Network and IT ***infrastructure management***
- ***Business process*** management
- ***Recommending*** products and services
- ***Social networking***

Let’s Cooperate!

When choosing a solutions for data storage and management, we are ***not obliged*** to use ***just one type*** of database management systems.

Since different types of SQL and NoSQL DBMSs have different features, that can be usefully together in a specific application, they may be used ***concurrently***.

Some use cases:

- ***Large-scale graph processing***, such as with large social networks, may actually use ***column family databases*** for storage and retrieval. Graph operations are built on top with a graph in-memory database.
- ***OLTP*** handled with a Relational Database (for ACID transactions), ***OLAP*** in charge to MongoDB for fast analytics.

Suggested Readings

Chapter 15 of the book "*Dan Sullivan, NoSQL For Mere Mortals, Addison-Wesley, 2015*".

Images

If not specified, the images shown in this lecture have been extracted from:

"Dan Sullivan, NoSQL For Mere Mortals, Addison-Wesley, 2015"