|  |  |
| --- | --- |
| **Date** |  |
| **Version Number** | V0.1 |
| **Owner** | Group Architecture Board |
| **Document Id** |  |
| **Primary Audience** | Group Architecture Board and IT |
| **Document Location** | Group Architecture |

[Project]

Data Architecture

|  |  |  |
| --- | --- | --- |
| Review & Approval | | |
| **Reviewed by** | **Job Title/Role** | **Date** |
|  | Data Architect |  |
|  | Quality Assurance |  |
|  | Software Development Manager |  |
| **Approved by** | **Job Title/Role** | **Approval Date** |
|  | System Design Authority |  |
|  | Project Manager |  |
|  | Software Development Manager |  |
|  | Enterprise Architect |  |

|  |  |  |  |
| --- | --- | --- | --- |
| History | | | |
| **Version #** | **Author** | **Comments** | **Date** |
| 0.1 |  |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |
| --- | --- | --- |
| Support & Reference Materials | | |
| **Ref** | **Title** | **Document Id** |
| 1 | Information Security Policy |  |
| 2 | Data Sensitivity Classification Policy |  |
| 3 | Group Architecture Policy |  |
| 4 | Group Architecture Glossary |  |
| 5 | Business Requirements – [Project] |  |
| 6 | Non-Functional Requirements – [Project] |  |
| 7 | High Level Solution Design – [Project] |  |

This Word document template is based on a hypothetical system upgrade example, which means there is an AS-IS and a TO-BE aspect so that the impact and business and technology can clearly be understood once this design is implemented. If this were a new system, then there wouldn’t need to be an AS-IS section. The hypothetical design outlined in this template consists of a client-server system comprising database servers, internet servers and development-support servers, with three environments: DEV, UAT and PROD, all on virtual networks. The example servers are modelled as on-premises VMs but could just as easily be applied to a cloud-based solution. Remove all comment sections like this here in italics before the final presentation. The Version number and the Document Identity fields should be set up in Word to automatically update when this document is checked back into SharePoint. Remove all comment sections in italics and example code before the final presentation.

Instead of using Visio, this templated used Mermaid.js or graphviz code in Visual Studio (mark-down plugins required) or mermaidchart.com for illustrations. The code for the illustrations is included in the guidance text.

# Contents

[1. Data Architecture Scope 4](#_Toc188625243)

[2. Data Architecture Design Decisions 7](#_Toc188625244)

[3. Contextual Data Architecture 9](#_Toc188625245)

[4. Conceptual Data Architecture 9](#_Toc188625246)

[5. Logical Data Architecture 10](#_Toc188625247)

[6. Physical Data Architecture 11](#_Toc188625248)

[7. Data Volumetrics 13](#_Toc188625249)

[8. Database Instance: [PROJECT] 14](#_Toc188625250)

[9. Information Architecture 19](#_Toc188625251)

[10. Data Pipelines 20](#_Toc188625252)

[11. Data Migration 21](#_Toc188625253)

[12. Data Computation scalability 24](#_Toc188625254)

[13. Data Security Architecture 25](#_Toc188625255)

[14. Data Visualisation, Reporting and Deep Insights 27](#_Toc188625256)

[15. Batch Processes 28](#_Toc188625257)

[16. Data Management 29](#_Toc188625258)

[17. Data Governance 30](#_Toc188625259)

[18. Data ownership and stewardship 31](#_Toc188625260)

[19. General Data Protection Regulation (GDPR) 33](#_Toc188625261)

[20. Database Deployment and Configuration 34](#_Toc188625262)

[Appendix A: Data Pipelines Review 35](#_Toc188625263)

[Appendix B: Database Vendors Overview 37](#_Toc188625264)

[Appendix C: Quotes 44](#_Toc188625265)

# Data Architecture Scope

## Identification

This [project] is a [functional description].

## The Function of Data Architecture

A solid data architecture is important to ensure smooth digital transformations and new application implementations since every aspect of a system is underpinned by data. The better the data quality and the better the data connections between system components, the better the system experience and the easier it is to automate processes, reports on business aspects, gain insights predict trends and make decisions. Data Architecture is both an art form and a science for connecting, managing and democratising data to enable automation, self-service reporting, data-led decision-making and the creation of new business opportunities.

**A good data architecture delivers the following:**

* Ensure one version of the truth and consistent meaning and context of data across the enterprise.
* Good connectivity to the data stores. Without connections, the data is meaningless to the business.
* A scalable strategy that handles increasing data volumes and data computations.
* Good data quality, that is sufficiently complete and meaningful for the business's purposes, yet is not duplicated nor ambiguous, and serves as the single source of truth for the business.
* Data is represented in a consistent format and can be described and categorized using business terminology.
* A governing function for people-based rules and automated processes to ensure data quality, compliance, ownership, accountability, cleansing, standardisation, categorization and security and privacy.
* Data Architecture acts as an interface between the IT project delivery function and the business stakeholders.

**Why do we need a data strategy?**

Managing Data Assets is challenging and needs a strategy to keep data under control. Consider these points:

1. We need a strategy to democratize our data by advertising the business meanings of it and making it accessible to our data-literate business colleagues without the explicit involvement of IT specialists, to foster a data-driven business culture and empower employees. Contrary to this, we need to protect our data by ensuring its integrity, privacy and security, limit data access, control its use and desensitize/depersonalize data where required. We therefore need to strike a balance between the value that is created from data democratization and the risk suffered from misusing such data.
2. We need a strategy to realize the value of our data by understanding how data contributes to achieving our business goals. We can achieve our business goals by:

* Automation of business processes (a.k.a. "Do things faster")
* Use of analytics to explore new ways of doing things (a.k.a. "Do things better")
* Creation of improved user experiences at scale (a.k.a. "Personalize experience to the individual, not some demographic user segment")
* Commercializing Data Products for internal or external use (a.k.a. "Package up data to make it useful to others")

1. We need a strategy to manage data as an asset by understanding the required data quality, structure and data processes. We do this by:

* Data discovery and documentation
* Assigning Data Ownership, Stewardship and Accountability
* Implementing Master Data Management
* Implementing data quality control

In summary, a data strategy helps us Understand, Manage, Protect and Leverage our data and sets the foundations for the business's critical capabilities.

## Document Purpose

The purpose of this Data Architecture Document (DAD) is to describe the designed strategy and architecture applied to the [project], in terms of data management principles, governance policies, privacy, security and access control. Specifically, this document provides:

* A description of the contextual, conceptual and logical data models that holds the core information objects that support the intended application, with guidance on how the physical models would need to be constructed, where appropriate.
* The high-level approach used to exploit the data models in the targeted solution to achieve a seamless data flow and timely delivery, as described by the non-functional requirements.
* An overview of all the transformations that the data will undergo as it traverses through its data pipeline.
* Specifications of what constitutes acceptable data quality and the approaches that will be used to achieve that data quality
* A summary of all automation processes that relate to the initial data take-on, preparation, ongoing management, rationalization, cleansing and transmission of data though the system, limited to the scope of the [project].
* An overview of reports (tabular and graphic) that will be produced as part of the overall solution, and a brief overview of any reporting tools used to produce these.
* An overview of the types of data analysis and data science-based business intelligence that will be performed on the data
* The high-level design of how security is implemented at a data level, as opposed to the security at a network of application level
* A summary of what data belongs to what data sensitivity classification
* Backing up and data vaulting strategies
* Recoverability strategies for loss of data

## Document Scope

This document presents the data-related aspects of the [project].

This Data Architecture Document will:

* Present a data-oriented design
* Describe the software interfaces to the data
* Highlight attributes of the data architecture that relate to non-functional requirements specified in the NFR [Ref. 6]. These include the Security, Safety, Performance, Maintainability and Availability.

## Intended Audience

The target audience for this document includes business, technical and non-technical, governance, and project management stakeholders. Specific users of this document include solution architects, data architects, developers and test analysts. This Data Architecture Document uses technical terms that should be understandable to this audience. A general Architecture Glossary exists to aid understanding [Ref. 4].

# Data Architecture Design Decisions

## Overriding Design Decisions

The table below identifies key design decisions that were specifically made for the [project]’s data. The decisions were made with SMEs to ensure agreement across all components that constitute the project and complement the overall design set out in the HLD Solution Architecture.

| **ID Number** | **Decision** |
| --- | --- |
| DAD-01 |  |
| DAD-02 |  |

Table 1 Data Architecture-specific Decisions

The design from the Solution Architecture HLD and the design decisions taken from Table 1 above are used to shape the data architecture below. The decision rationale is based on the following:

## Limits of legacy database vs new database

[Describe limits in the legacy database that were relevant in choosing the new database, if any.]

Consider the following trade-offs:

* Limit on horizontal scaling
* Transaction ability and ACID compliance
* Database modelling and flexibility

## Benchmarking comparisons

## Choice of Database type and vendor

Refer to Appendix C: Database Vendors on page 30 for a comparative list of database vendors.

## Alternative upgrade approaches explored

[Provide a brief overview of what alternative remedies/approaches/vendors have been explored with reasons. Consider these points:

Vertical scaling by upgrading memory or processors

Horizontal scaling using sharding, clustering, replication, multiple-read-only etc. techniques

Disk I/O too slow, or too much Disk I/O instead of in-memory processing

Possible database configuration changes

Database tuning: Table design, indexed, compaction strategies, garbage collections, memory usage, query and stored procedure fine-tuning

Storage partitioning schemes for physical storage devices

Network tuning, data pipe finessing, client application tuning, API finessing

Cache the database front-end

Experts and vendors consulted?]

# Contextual Data Architecture

The contextual data model and data pipeline describe how data is mapped to the business activities to meet the business requirements and flows through the application [Ref. 1]. The sources of external data and internal data generation are described here too.

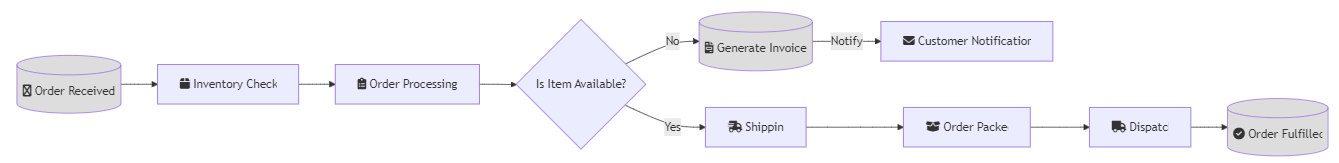


Figure 1 AS-IS Contextual data diagram: Customer order fulfilment

[Instead of using Visio, you can use this Mermaid.js code in Visual Studio or mermaidchart.com for this illustration:]

flowchart LR

%% Nodes for the Order Fulfillment Process

A[("fab:fa-cart-plus Order Received")]

B["fa:fa-box Inventory Check"]

C["fa:fa-clipboard-list Order Processing"]

D{"Is Item Available?"}

E["fa:fa-shipping-fast Shipping"]

F["fa:fa-truck Dispatch"]

G[("fa:fa-file-invoice Generate Invoice")]

H["fa:fa-envelope Customer Notification"]

I["fa:fa-box-open Order Packed"]

J[("fa:fa-check-circle Order Fulfilled")]

%% Edge connections between nodes

A --> B --> C --> D

D -- Yes --> E --> I --> F --> J

D -- No --> G -- Notify --> H

%% Styling for specific nodes

style A fill:#dddddd

style G fill:#dddddd

style J fill:#dddddd

# Conceptual Data Architecture

## Overview

Describe the high-level data entities that support the application, and how they relate to one. Row details are not included since they will be shown in the eventual data model.

A diagram of a product

Description automatically generated

Figure 2 AS-IS Conceptual data diagram: Customer order fulfilment

Mermaid.js code for this diagram:

---

title: Order example

---

erDiagram

CUSTOMER ||--o{ ORDER : places

ORDER ||--o{ ORDER-ITEM : includes

ORDER-ITEM }o--|| PRODUCT : refers

ORDER ||--o| SHIPMENT : sent\_via

INVENTORY |o--|| PRODUCT : has

EVENT

LOG

## Description

[Add further details here…]

# Logical Data Architecture

## Overview

This section describes the structures used where structured data is used in the application, e.g., normalised, dimensional, and hierarchical, key-value pairs, streamed data, time-series data, images, and documents. Any pertinent characteristics in unstructured data used by the application are also enumerated here.

[Examples: Update the database design here. It is sufficient only to show the significant tables.]

A diagram of a company

Description automatically generated

Figure 3 Detailed database design

[mermaid.js for this diagram:]

erDiagram

    CUSTOMER ||--o{ ORDER : places

    ORDER ||--o{ ORDER-ITEM : includes

    ORDER-ITEM }o--|| PRODUCT : refers

    ORDER ||--o| SHIPMENT : sent\_via

    CUSTOMER {

        string customerID PK

        string name

        string email

    }

    ORDER {

        string orderID PK

        date orderDate

        float totalAmount

    }

    PRODUCT {

        string productID PK

        string name

        float price

    }

    ORDER-ITEM {

        string orderID FK

        string productID FK

        int quantity

    }

    SHIPMENT {

        string shipmentID PK

        string carrier

        date shipDate

        date deliveryDate

    }

## Description

[Add further details here…]

# Physical Data Architecture

## Overview

This section describes the physical implementation of each database, whether the database is cloud-hosted or hosted on-premise.

This section describes the physical implementation of each database, the database management systems used in the application, data storages and their file systems, directory structures, and any special features offered by the database system in use. Aspects such as table partitioning, data sharding across servers, table partitioning across storage nodes, clustering and horizontal scaling techniques, and geographical data distribution techniques are also described here. Furthermore, aspects of the physical data architecture that relate to disaster recovery (DR), backing up and restoring, and data vaulting and disposal are also described here.

## Reasons for using a different database system

If there is a change of database from the AS-IS to the TO-BE, then here are some good reasons why to support such a move.

Only select a new database if the current problems are very difficult to fix. First, consider:

* Can the existing database be configured to fix these problems/give more breathing room?
* Can the life be extended by adding a cache, a replication scheme, data sharding across servers or table partitioning-schema across multiple storages?
* Do the costs justify the benefit?
* Can the risks be mitigated?

Account for database migration time, skills acquisition time, interface changes and different infrastructure requirements.

Example reasons why a different database system may be required:

* Too much data and projected growth will exceed current capability
* Growing Latency
* Excessive Disk I/O
* Increasing demand for more memory
* The license cost model is costly

## Vendor choices

If this is a new project or a database technology upgrade, describe the vendor choice based on the technical capabilities to meet the data requirements for this application. The non-function requirements that should be fully stated in [Ref. 6].

* Are ACID transactional guarantees required in the design?
* Is eventual data consistency acceptable?
* Is most database access to read or to write?
* What database paradigm(s) does the solution require?
  + Append-only
  + Big Data
  + Columnar
  + Distributed
  + Document
  + Data Warehouse
  + Graph
  + NoSQL
  + Key-Value
  + ORM
  + Realtime
  + Relational
  + Streaming
  + Temporal
  + Time-series
  + Wide Column

Refer to Appendix C: Database Vendors on page 30 for a comparative overview of database vendors.

# Data Volumetrics

## Database Volumetrics

[Repeat for each component that undergoes significant data ingress and egress. This is useful when determining storage design and network design, and ultimately in the database design where ingress data could get lost if it can’t handle the traffic.]

Table 2 lists the estimated data traffic for the database in a typical production setting for reading and writing data.

| **Operation** | **Transactions per Second (TPS)** | **Data volume per transaction in KB** | **Data volume per second (KB/s)** |
| --- | --- | --- | --- |
| Insert | ... KB/s | … KB | … KB |
| Select | ... KB/s | … KB | … KB |
| Update | ... KB/s | … KB | … KB |
| Delete | ... KB/s | … KB | … KB |

Table 2 Expected transaction rates

| **Operation** | **R/W Ratio of Transaction** | **R/W Ratio of Data volume per transaction** |
| --- | --- | --- |
| R/W Ratio |  |  |

Table x: Read / Write Ratio for the estimated Data traffic

*[Explain how these ratios were used to guide aspects of the data architecture.]*

* **Physical tablespaces on the server**

[Example: The back-end RDBMS is SQL Server 20XX SP2 and is hosted by [company] in its data centre. The database holds configurations, user metrics and client content. The full-text search function is reserved for in the design but is not currently used.]

The tables below show the physical database architecture for all environments, where only one tablespace is used. There is no partitioning of any data tables.

## Component B Volumetrics

[Etc…]

# Database Instance: [PROJECT]

## Storage Mounts, Sizing & Growth Projection

[Estimate the anticipated growth per year to calculate the initial storage required before storage must be extended]

### Database: APPDB

| **Storage** | **Mount** | **Size GB** | **%Growth per year** |
| --- | --- | --- | --- |
| **Data Tablespace 1** | G: |  |  |
| **Indexes Tablespace 1** | H: |  |  |
| **Data Log Tablespace 1** | I: |  |  |
| **Table Partitions** | None | N/A | N/A |
| **Full-Text Search data** | K: | 1G |  |
| **Full-Text Search indexes** | K: | 1G |  |

Table 3 Application database server mount points

### Database: REPORTDB

| **Storage** | **Mount** | **Size GB** | **%Growth per year** |
| --- | --- | --- | --- |
| **Data Tablespace 1** | G: | TBA |  |
| **Indexes Tablespace 1** | H: | TBA |  |
| **Data Log Tablespace 1** | I: | TBA |  |
| **Table Partitions** | None | N/A | N/A |
| **Full-Text Search data** | K: | TBA |  |
| **Full-Text Search indexes** | K: | TBA |  |

Table 4 Report Database server mounts

### Database: TEMPDB

| **Storage** | **Mount** | **Size GB** | **%Growth per year** |
| --- | --- | --- | --- |
| **Temp Data** | J: | TBA |  |
| **Temp Indexes** | J: | TBA |  |

Table 5 TempDB server mount points

## Data Storage Design

[Provide physical details of the actual storage used to support based on the initial storage requirements and expected annual growth before the storage must be extended, or the anticipated lifetime of this system.]

This section describes the storage for each of the system environments, which are DEV, QA and PROD:

* Structured data in the relational database:
* Table spaces on the RDBMS, if more than the default
* SAN Data tiering used for each table space on the RDBMS
* Unstructured data (documents, audio, images, video)
* Non-relational databases a.k.a. “no-SQL” databases, e.g. key-value stores, graph databases, document stores, column stores
* File systems for heterogeneous files.
* Amend this table accordingly or cloud storage.

### DEV Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL01 (DEV Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 6 DEV Database server storage mounts

### QA Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL02 (QA Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 7 QA Database server storage mounts

### PROD Environment Database Storage Design

| **Server Node:** | | **PVUKxxxyyySQL03 (PROD Database server)** | | | |
| --- | --- | --- | --- | --- | --- |
| **Local/SAN** | **OS/Bin/Data/Page** | **FS** | **Mapping** | **Tier** | **Size (GB)** |
| SAN | OS | NTFS | C-drive | 2 | 32 |
| SAN | Page | NTFS | D-drive | 2 | 8 |
| SAN | Binaries | NTFS | E-drive | 2 | 32 |
| SAN | Application Data | NTFS | F-drive | 2 | 32 |
| SAN | DB Data | NTFS | G-drive | 2 | 128 |
| SAN | DB Indexes | NTFS | H-drive | 2 | 32 |
| SAN | DB Log | NTFS | I-drive | 1 | 32 |
| SAN | DB TempDB | NTFS | J-drive | 1 | 32 |
| SAN | DB Backup | NTFS | J-drive | 3 | 128 |
| SAN | Full text Search | NTFS | K-drive | 1 | 256 |

Table 8 PROD Database server storage mounts

## Initial Data Take-on

This solution does not require and initial data take-on.

## Data Storage scalability

[Describe how storage will scaled and when it will be considered necessary, based on based on the future-proofing requirements laid out in the NFR over the lifetime of the system]

## Data Replication

### Replication Technique

Describe the primary database to secondary database replication, role reversal scenarios, physical locations, connectivity, and expected network latency The NFR should state the allowed latency for data consistency between replicated database instances, and what the geographical replication requirements are,

What replication approach is used?

* Synchronous - wait for the replication to complete before committing the transaction
* Asynchronous - data is written to the secondary database after it is already committed on the primary server
* Peer-to-peer
* Key-based
* Batch-based
* Etc…

## High Availability

If the NFR requires this, explain how this is achieved. Examples are

* Oracle RAC
* High Availability via storage design, e.g. a SAN
* Servers on a clustered file system
* Cloud-based solutions

## Distributed System Design

Aspects of the distributed system design are considered here, if any. Possible server distribution approaches are:

* Data sharding a.k.a. fragmentation (each server holds a range of data e.g. geographic unit)
* Multiple servers on a clustered file system
* Data replication at the storage level, e.g. on storage area networks (SANs) and on many cloud offerings

## Data Interfaces

[This section describes the high-level technical interfaces, data pipelines and user interfaces between the data stores, the applications, the middleware and the users. It should demonstrate how data can safely, speedily and readily be accessed by these components, and transferred in bulk between them. Any bespoke data services that are used for connectivity are described separately.]

## Data Services

This section describes the data services that exist to access the data from the application. Such services can either be via a function library such as ODBC, JDBC or DBLIB, web services such as ReST-full API or GraphQL, or vendor-bespoke data services as offered in the latest generation of data architectures like Snowflake and Kafka. Also for consideration here are any inter-database service procedure interfaces (SPIs).

# Information Architecture

This section describes and classifies the types of information that this application treats, the provenance of the information and the peripheral systems that intend to consume the information.

## AS-IS Information Classes

[Include this section if there is a change in the information classes]

## TO-BE Information Classes

[Example of data sets and the respective security classifications. Also indicate what type of operations will be performed to these data sets.]

| **Data Type** | **Source / Mastered by** | **Security Class** | **CRUD** |
| --- | --- | --- | --- |
| **User Data** | LDAP | Confidential | R |
| **Process Events** | Application | Internal | CR |
| **Parts Data** |  | Secret | R |
| **Search results** |  | Secret | C |
| **Etc..** |  |  |  |

Table 9 TO-BE Information architecture summary

## AS-IS Information Flows

[Include this section if there is a change in the information flow in the solution]

## TO-BE Information Flows

[Example Figure of information flow between components]

Information flows include all information transfers between architectural components, and can be in the style of:

* Batched, bulk data transfers
* Streamed continuous data transfers
* Discrete data transfers using any number of inter-process communication protocols.

The information flows are summarized and described in below in Table x below.

| **Ref** | **Source** | **Destination** | **Description** | **Frequency** | **Size** | **Batched/Streamed/API** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Client A | Server B | Client content data deployment | Daily | 100MB | Batched |
| etc... |  |  |  |  |  |  |

Table 10 Information flows

## Data Partner Services

No data is directly provided by other third-party data partners.

# Data Pipelines

Data Pipeline Processing and Computation

## Introduction

Computation on the database itself are, for example, in-flight string manipulations, calculations, aggregations, execution of stored procedures. Other data based computations are performed on other external devices, such as data science-oriented operations, and machine learning (ML) and artificial intelligence (AI) operations such as pattern and image recognitions. As the size of the data grows, the capability of the database management system should also grow, and so should the computational data capability grow.

The following processes are performed on the data pipeline:

* Data profiling
* Data cleansing
* Data enrichment
* General data processing

# Data Migration

This section describes the data migration aspects of the project. This template below describes underlying principles for implementing a successful data migration. The technical details will however vary depending on the project nature.

## Summary:

* Data migration often entails the movement and transformation of data between two differing data models
* Data migration is difficult and is a poorly understood discipline
* The greater the business change, the more complex and riskier the data migration.
* Poor-quality source data risks data migration failure

### 1. Understand Source Data Quality in terms of existing structure, content and quality:

* Data Profiling to identify missing values, inconsistent formats or outliers
* Data Validation by running validation rules, that check for accuracy, value ranges, correct data types
* Data Cleansing by removing wrong or irrelevant content, correcting wrong values, deduplication, normalization, correcting errors identified during profiling and validation

### 2. Understand the current AS-IS data model in terms of source entities, semantic meanings, relationships & attributes

* Involve the BAs or Data Stewards / Data Owners of the source data
* Document your findings

### 3. Understand the target TO-IS data model in terms of source entities, semantic meanings, relationships & attributes

* Involve the BAs or Data Stewards / Data Owners of the source data
* Show how source data needs to be structurally changed to fit into the target data model
* Consider storage requirements for the new data
* Document your findings

### 4. Determine how data is manipulated on the target system compared to the source system

Consider changes in business processes, data classifications, life cycles, new attributes and semantic meaning of the business entities. The recommended steps are:

Step 1: Establish a Business Glossary

A Business Glossary lists:

* Business actors
* Business entities
* Business states for these actors and entities

And then describe the differences:

* Current definition and rules
* New definition and rules

| **Term** | **AS-IS description** | **AS-IS business rules** | **TO-BE definition** | **TO-BE business rules** |
| --- | --- | --- | --- | --- |
| Lead | *[old definition]* | *[old business rules]* | *[new definition]* | *[new business rules]* |
| Opportunity |  |  |  |  |
| Customer |  |  |  |  |
| etc... |  |  |  |  |

Table 11 AS-IS vs TO-BE Business Glossary

Step 2: First round data mapping

* Field-based mapping
* Merge and split fields where required

Step 3: Second round data mapping

* Apply definition and business changes as logic statements to the data map obtained thus far, to be applied in the transformation logic after data ingestion.

## The ETL/ELT

The Extraction, Loading and Transformation process typically consists of these steps:

[graphviz code for this diagam:]

```graphviz

digraph ETLProcess {

rankdir=LR;

size="10,5";

ETL0 [shape = cylinder, label="Source"];

ETL1 [shape = rarrow, label="Source\nExtraction"];

ETL2 [shape = cylinder, label="Landing\nArea"];

ETL3 [shape = rarrow, label="Data\nProfiling"];

ETL4 [shape = folder, label="Profiling\nReport"];

ETL5 [shape = rarrow, label="Data\nCleansing"];

ETL6 [shape = cylinder, label="Staging\nArea"];

ETL7 [shape = rarrow, label="Load to\nTaget"];

ETL8 [shape = cylinder, label="Target\nArea"];

ETL0->ETL1

ETL1->ETL2

ETL2->ETL3

ETL3->ETL4

ETL4->ETL5

ETL5->ETL6

ETL6->ETL7

ETL7->ETL8

}```



Figure 4 Stages of a typical ETL Process

### High level technical view of the Data Migration

[graphviz code for this diagam:]

```graphviz

digraph TechDataMigration {

rankdir=LR;

size="10,5";

START [shape = oval, label="Start"]

ETL [shape = box, label="ETL"];

ETL2 [shape = box, label="ETL"];

TEST [shape = cylinder, label="Test\nSystem"];

RESET [shape = box, label="Reset\nTest\nSystem"];

UAT [shape = box, label="UAT\nTesting"];

OK [shape = diamond, label="UAT Test\nOK?"];

PROD [shape = cylinder, label="Production\nSystem"];

MOPUP [shape = box, label="Post Migration\nMop-up"];

END [shape = oval, label="End"]

START -> ETL

ETL -> TEST

TEST -> UAT

UAT -> OK

OK -> ETL2

ETL2 -> PROD [label="Yes"]

OK -> RESET [label="No,\nrepeat"]

RESET-> ETL

PROD -> MOPUP

MOPUP -> END

}

```

A diagram of a triangle

Description automatically generated with medium confidence

Figure 5 A typical Technical Data Migration

## Data profiling

### Use cases for data profiling:

* Data migration, to move data between systems
* Data governance, which defines policies of managing data
* Master Data Management, to maintain a single source of the truth
* Data Warehousing, for analytics and insights
* Compliance, to comply with laws and regulations

The objective of data profiling is to analyse and summarize the characteristic of data in terms of structure, quality and content. The following types of profiling need to be done:

* structure profiling: tables, keys, table relationships, column data types, volumetrics, stored procedures and rules
* quality profiling: reliability, completeness, accuracy, frequency, distribution, consistency (duplicates, mismatches), timeliness (freshness, currency)
* content profiling: meaning pattens, sequences, correlations, anomalies, frequencies, value distributions in columns, relationships between elements, summations, statistics and other aggregates for trends and seasonality.

### Data Profiling Process:

An optimal profiling process would be as follows:

1. Connecting to the data and extract it
   * Identify data sources
   * Ensuring access and permission to it
   * Set up a data landing area
   * Extracting the data
2. Understand the data structure, volumes, growth projections
3. Explore source data to identify most important data. Statistical analysis to get distributions, counts, frequencies, spreads & tendencies.
4. Analyse source data quality for Accuracy, Completeness, Consistency, Reliability, missing values, outliers, inconsistent formats. Define data quality metrics.
5. Analyse data content for patterns, trends, correlations, data ranges, constraints
6. Document findings into Data Profiling report and a Data Structure Report

## Data cleansing

No data on this application needs to have any incoming data to be cleansed.

## Data enrichment

No data on this application needs to have any incoming data to be enriched.

## Reconciliation

No data on this application needs to be reconciled.

## General data processing

No further data processes to be performed on the data.

# Data Computation scalability

[Statement on computation scalability on the database itself (in-flight string manipulations, calculations, aggregations, stored procedures), based on future-proofing requirements in the NFR.]

This section describes how the computational ability of the installed system is allowed to grow as the needs for this increase.

Possible approaches are:

* Vertical scaling: add more RAM, processor cores, clock speed,
* Horizontal scaling: add more servers that run massively parallel processes (MPP), or run a map-reduce strategy. Describe how load balancing works and how the resources are negotiated.
* Elastic scaling in the cloud: The cloud vendor will combine vertical and horizontal scaling approaches and will employ its own approach to resource allocation and load balancing.
* **Distributed System Design for Computation**

[Aspects of the distributed computation design are considered here, if any. Possible distributed computation approaches are the distributed Spark database or the Apache Impala distributed computation model. The now dated MapReduce approach, as offered by Hadoop et al is also a candidate for making the computational aspects of a database distributed. AI and ML operations can be farmed out to Cloud providers]

# Data Security Architecture

## Introduction

Authentication, authorization and auditing (AAA) are the three pillars of data architecture security. This chapter describes each of these aspects from a high level point of view.

## Authentication

[Discuss the type of authentication in use by users and processes: Either Password, LDAP, SSL, PPK, API bearer token, SSO, etc…]

This method by which either a user confirms its identity is performed using multi-factor authentication.

This method by which either a client process confirms its identity is performed using an SSL certificate.

## Authorization

Authorization is the process that grants users or processes various levels of access to data, to carry out specific actions on data, i.e. create, retrieve, update and delete (CRUD), and also under which operational situations these authorizations apply, e.g. which use-cases, rules for specific environments. This may also be coupled with execution rights to specific programs that operate on the data.

## Access control for data take-on

[Describe how to connect to for the various use-cases (e.g. engineer, admin, and operator) that will perform production operations] [Roles and privileges to perform data take on]

| **Use case** | **Role Name** | **Privileges** | **Data Entities** |
| --- | --- | --- | --- |
|  |  |  |  |

Table 12 Access privileges for data take-on

Table 9 Access privileges for data take-on

## Access control in production

[Roles and privileges to perform production operations for each use case to perform data take on]

| **Use case** | **Role Name** | **Privileges** | **Data Entities** |
| --- | --- | --- | --- |
|  |  |  |  |

Table 12 Access privileges for data take-on

Table 9 Access privileges in production time

## Access control for application and database support

[Often in simple applications, the following is the case: There are no specific Roles and privileges for database support at the application level. The standard SA account for SQL Server RDBMS and the “sysdba” account for the Oracle RDBMS used.]

| **Use case** | **Role Name** | **Privileges** | **Data Entities** |
| --- | --- | --- | --- |
| SQL Server | SA | System | All application and system tables and views |
| Oracle | Sysdba | system | All application and system tables and views |
| Etc… |  |  |  |

Table 13 Access privileges for support

## Access control for development and testing

[Roles and privileges for database development]

| **Use case** | **Role Name** | **Privileges** | **Data Entities** |
| --- | --- | --- | --- |
| Schema designer |  |  |  |
| SQL Coder |  |  |  |
| Unit Tester |  |  |  |

Table 14 Access privileges for database development and testing

## Auditing

The following database-related events are logged to an audit with a timestamped user identifier or process identifier to provide a complete trace:

* All DDL changes
* All changes to static configuration and reference data
* All warnings and errors
* Backups and restores
* Shut-downs and restores of the database management system
* User logins and logouts
* Process connects and disconnects

The audit log in this case is a table called “audit”.

The audit log can be viewed through an SQL development tool and manually written SQL queries.

The audit log can only be truncated by the database administrator.

# Data Visualisation, Reporting and Deep Insights

## Introduction

This section described all data visualisation and reporting techniques used in the application. Some operational data may be also copied to a data warehouse (a dimensional model) for deeper analysis by a self-service business intelligence (BI) tool and machine learning (ML) by data scientists, in which case the mapping between business area and BI domains will also appear here.

## Data Mining

There is no data mining requirement for this project. It is expected that requirements will eventually emerge. All the necessary components are in place to implement this. The current infrastructure is anticipated to be able to adequately cope with the implementation of these requirements.

## Reporting (Predefined)

There is no pre-built business reporting requirement for this project. However, there is a reporting function that currently delivers performance reports for support.

## Reporting (Ad-hoc)

There is no ad-hoc business reporting requirement for this project. However, there is a reporting function that is able to deliver this, should it be required in the future.

# Batch Processes

## Introduction

This chapter lists the batch processes that are run on the database, which perform the following types for functions:

* Function 1
* Function 2
* Etc…

The batches are controlled by the [SQL Server AT Server, CRON job, Windows AT job, Oracle DBMS\_JOB, Airflow, etc…] scheduler.

## Batch Processes

[State purpose and schedule of each batch job. Use the UNIX CRON specification to indicate the batch schedule ]

|  |  |  |  |
| --- | --- | --- | --- |
| **ProcessId** | **Name** | **Description** | **CRON Schedule** |
|  |  |  |  |
|  |  |  |  |

# Data Management

This section describes the processes that will be in place to manage the data in the database(s).

## Data Backup and Restoring

Describe the backup Schedule, what data gets backed up, where it is backed up, where and when it is vaulted to, when backed-up data is hived off. Also specify the format in which that data is backed and the tool used for creating the backup.

Describe how data would be restored from a backup, or from a vaulted backup, and where it would be backed up to. Has storage provision been made for restoring a backup into?

## Archiving and Vaulting

[What much data needs to be archived during the housekeeping cycle, and how to retrieve it whenever needed].

## Data Disposal Life-cyclef

Describe how data that is deemed useless due to compliance or irrelevancy is safely removed from media and from estate.

## Data Housekeeping Cycle

Describe how often data is manually looked at, repaired or pared off, and by whom. What are the change-control processes to sanction this in a production environment?

# Data Governance

## Summary

 Data governance outlines the overall desired structure, and how that should be achieved in terms of rules, accountabilities and processes. In contrast, data management is about the hands-on implementation of the regulations set by the data governance.

## Introduction

Data governance consists of an agreed decision-making model that ensures that the right people access the right data at the right time with the right amount of accountability. It is a continuing process that needs to be aligned with the business changes, data changes, and business colleague changes.

The points raised here set the agreed decision-making framework...

This chapter details the rules used for effective data governance on the application. Here, the personnel or staff roles of the responsible data owner, data quality stewardship, compliance, retention, disposal and security are enumerated and described. The finer aspects of data treatment to optimise data quality are also described here, e.g. data cleansing, standardisation, categorisation, enrichment, rationalization and any required encryptions.

## Data Policy

The following guidelines are applied to manage data effectiveness

## Data Standards

Ensure consisten ways of capturing, recording and presenting data in the enterprize

## Data Quality

This is the measure of useability and integrity to ensure that the data can serve its purpose. Describe the checks to ensure ongoing data integrity and

How stable is the data?

How do we know that the data can be trusted?

How do we keep on maintaining the integrity of the data?

### Data quality overview

This section describes the processes used to ensure that the requisite data quality, as specified in the non-functional requirements (NFR), is achieved. In particular, it identifies the data stores that holds the “single source of truth”, highlights special cases where the data needs to be duplicated, how incomplete data is identified and remedied, and the identification and remediation process for any data redundancies.

### Data Cleansing and Standardization

[Describe which data entities are cleansed, what aspects are cleansed, and to what format the relevant data is standardised.]

### Data Documentation

Describe the information content in the data, its provenance in terms of sources and lineage. Describe who accesses it, who monitors it, how often is it updated, and what the data is used for. Determine multiple sources of the same information.

### Categorisation and Cataloguing of data

[Describe how the data that is stored and generated by the application would be categorized into catalogs for possible use in enterprise-wide data lakes and analytics.]

## Static and Reference data

The maintenance of static and reference data is the responsibility of the data owner. This data rarely changes but it is important to keep it up to date, and to keep a history of the changes to this data.

Reference data sets held by the application are:

* [Glossary, Abbreviations, ISO country codes, ISO currency codes, exchange rates, etc…]
* [Describe the processes to update reference data]
* [Describe how the change history of static and reference data is stored and audited]

## Data lineage

[Data lineage lays out the history of data and all its transformation from creation to consumption and can tell us who owned the data at a given point in time. Describe for which data entities data lineage is maintained and how the lineage information is held.]

## Data Security

Access Management, Personally Identifiable Information

## Data Enrichment processes

[Describe what data entities would be subject to any data enrichment processes. This may entail the completion of empty fields in rows, for example in data entities such as addresses, engineering parts, supplier details. I can also entail the categorization and linking of rows in each data entity.]

## Data Rationalization and Master Data Management

[Describe the processes used to identify missing data and duplicate in entities’ data hierarchy. Example: There is no MDM requirement for this project. There are no MDM issues that arise from the implementation of this stage of the project.]

## Data Value

*[Describe the value of data that is created within the solution.]*

*[Example:]*

*[At present, the information created by this application is limited to the following:]*

| **Ref** | **Information** | **Description** | **Level of detail** | **Usage** |
| --- | --- | --- | --- | --- |
| 1 | User Interactions | Every user action is logged | Location and user-action level | Usage analysis |
| 2 | Client system | All system events | Timestamp | Perf. analysis |
| 3 | etc... |  |  |  |

Table 15 Created information

# Data ownership and stewardship

*[Enumerate which corporate entities owns which business domain's data, who is the subject matter expert, and who is the authority on resolving data discrepancies.]*

| **Ref** | **Role** | **Person(s)** | **Contact details** | **Role description** |
| --- | --- | --- | --- | --- |
| 1 | Data owner, business domains A and B |  |  | Ultimate responsible across business domains |
| 2 | Data quality steward, business domain A |  |  | Subject matter expert |
| 3 | Data quality steward, business domain B |  |  | Subject matter expert |
| 4 | Data protection officer |  |  |  |

Table 16 Data governance roles

# General Data Protection Regulation (GDPR)

[Even if no personal data is held on the databases, then the data breach reporting process needs to be described.]

## Geographical scope of GDPR

[Describe the geographical scope(s) of the applicable GDPR in use here. The scopes and details of compliancy regulations of the many GDPRs varies by geographical region, but in general they all aim to achieve the same goals.]

## Data breach and reporting process

[Describe how a data breach in this application could conceivably be detected] [In addition to the standard data breach reporting process within COMPANY Maritime Systems, state any additional reporting steps required for this application.]

## Privacy incident, data complaint and subject data request handling

[Processes should exist so that it is possible to respond in a timely manner to requests about the lineage and lifecycle of private data, and to react to data complaints.]

## Implementation of GDPR

Here we describe how the guiding principles of the GDPR are adhered to in the [appname] application’s data architecture, which are:

| **GDPR Principle** | **Description and Cross-references** |
| --- | --- |
| Lawfulness, transparency and fairness |  |
| Purpose of limitation |  |
| Data minimisation |  |
| Accuracy |  |
| Storage limitation |  |
| Confidentiality and integrity |  |

Table 17 GDPR principles implementation

# Database Deployment and Configuration

Describe any special steps required beyond those recommended by the vendor to install and configure the database system(s).

This chapter describes any special steps to provision or install the RDBMS, and specific configuration that is required on the RDBMS and what data is required to prime the database before the application can be opened. The vendor’s installation and configuration instructions can be found [here].

## Special Deployment Steps

## Project-specific Database Configuration

1. Data Pipelines Review

Is the choice of data pipeline technology already a given on the IT Estate, or is there a choice?

You should us a data pipeline technology / tool, because it:

* Eliminates manual steps in the data flow
* Quick to implement a data pipeline with configuration steps
* No coding is required, avoiding expensive development cycles
* Minimal training required
* Comes with assured standard transformation, aggregation, filter and sorting components
* Specialized bespoke data transformations, aggregations, filters and sorting libraries can be integrated
* Enables a smooth automated data flow
* Combats latency and bottlenecks
* Reduces and manages errors
* It is one tool to create any type of data pipeline
* Connects to any type of data source and service
* Can simultaneously connect to multiple data sources and sinks
* Deals with both batched (a.k.a. "ETL") and streamed data (a.k.a. "ELT")

Architecturally, a data pipeline tool has these further benefits:

* *Automation:* Provide automation of the entire pipeline process
* *Scalability:* Can handle growing data volumes using horizontal and vertical scaling techniques
* *Real-time processing:* Can do both batched and real-time data streams for immediate business decision making
* *Data quality and consistency:* Assured transformations and data cleansing
* *Central Control:* All data pipelines can be monitored and controlled from one auto-generated dashboard
* *Error handling and recovery:* Provides fault tolerance, detect issues, and elegant error recovery methods without disrupting other data flows
* *Risk:* Reduces overall risk of data movements within the enterprise
* *Build vs. Buy:* Building costs far outweigh the buying and configuration costs

Selection of preferred Data Pipeline technologies:

* SQL Server Integration Server (SSIS)
* No code, configuration only
* Custom libraries can be included in the pipeline
* Connectors to major databases and services
* Azure Data Factory
* No code, configuration only
* Connectors to major databases and services
* Custom libraries can be included in the pipeline
* AWS Glue
* No code, configuration only
* Interactive notebook style iteration
* Connectors to major databases and services

1. Database Vendors Overview

This appendix does not need to be included in the final document but initially serves as a convenient source for vendor feature comparisons.

| **Rank** | **Database** |
| --- | --- |
| 1 | Oracle |
| 2 | MySQL |
| 3 | SQL Server |
| 4 | PostgreSQL |
| 5 | MongoDB |
| 6 | Redis |
| 7 | Snowflake |
| 8 | Elasticsearch |
| 9 | DB2 |
| 10 | SQLite |
| 11 | Cassandra |
| 12 | Access |
| 13 | Splunk |
| 14 | Databricks |
| 15 | MariaDB |
| 16 | Azure SQL |
| 17 | Amazon DynamoDB |
| 18 | Apache Hive |
| 19 | Google BigQuery |

Table 18 Most popular database systems in 2024

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Database** | **Paradigm** | **Features** | **Power by** | **Access** | **Use case** | **Similar** |
| **Astra DB**  DataStax Astra DB (@AstraDB) / X | Big Data NoSQL  Column-wide store | Serverless,  High availability  Zero downtime  Vector processing  Extreme performance  Global scalability  Clustering |  | C++, Node, Python, Java | AI, RAG, Real-time | Cassandra |
| Google Cloud Platform Partner in India | GCP Partner**Bigtable** | Wide-column  Key-value  NoSQL | Unlimited scalability  Auto-scalability Petabyte sizing  Data versioning  Low latency  High throughput | Google Cloud | Python, C++, Java, Go | Data consolidation  Fraud detection  Media delivery  Time series  ML & AI | HBase  Cassandra |
| **Cassandra** | Big Data NoSQL | No joins  Serverless  High availability  Zero downtime Vector processing Extreme performance,  Global scalability Clustering | Java | Cassandra Query Language (CQL) | Media delivery NetFlix  IOT sensors, streaming and log data,  Time series, Recommend engines | Scylla DB Astra DB |
| **CloudFlare D1** | Relational | Serverless,  Geographic distribution  DR |  | API  Worker threads |  | SQLite |
| **Cosmos DB** | NoSQL | Geographic scaling  Replication | Azure | API | Multiple data model types | MongoDB, DynamoDB |
| **Couchbase** | Document  Key-value | In memory  Global replication |  |  | CMS  e-commerce |  |
| **CouchDB** | Document Key-value | Memory-based  Data consistency without locking using eventual consistency | Apache | ReST API | CMS  Mobile applications |  |
| **DB2** | Relational | Enterprise | C | SDK | Finance, legacy |  |
| Dolt Logo**Dolt** | Relational | Serverless, GIT-like operations: fork, merge, push, pull |  |  |  | MariaDB. MySQL |
| **Dragonfly** | Key-Value and Object store | Persistent storage  In-memory  Low latency  Complex data  Vector searches  Serverless  Scalable  Snapshots  Replication  Geospatial  Partitioning  Transactions  Pub/Sub messaging |  | Multiple languages SDK | Caching  session management, gaming leaderboards,  25x faster than Redis  Support up to 1TB nodes | Wire-compatible with Redis  Dragonfly  Valkey  KeyDB |
| DuckDB (@duckdb) / X**DuckDB** | Columnar RDBMS | In-process, portable, small, high performance. Supports: Parquet, JSON, flat files, HTTPS- & S3 protocols | C++ | CLI client, API, Rich SQL, C++ SDK, Python |  | SQLite |
| **DynamoDB** | NoSQL  Document  Key-value | Key-Value data store, "limitless" geographic scalability, low latency high availability, high durability, serverless | Amazon |  | IoT | MongoDB, Cosmos DB |
| **EdgeDB** | Relational  Graph  ORM |  |  |  | Graph data models  IOT |  |
| **ElasticSearch** | Document | Sharding, Replication,  Searching | Apache Lucene | ReST Interface | Full text searching |  |
| Fauna Logo PNG Vector (SVG) Free Download**Fauna** | Relational  Document  Temporal  ORM  Graph | Serverless  Fast  ACID  Unlimited scaling  Auto-scaling  Partitioning  Multi-tenancy | Twitter,  Cloud | GraphQL  ReST API  Web interface, Shell  FQL |  |  |
| **Firebase + Firestore** | NoSQL  Realtime  Mobile App IDK | JSON real-time synchronization,  Offline  Scalable by adding databases  Security concerns  Limited query capabilities | Google,  Cloud | JSON,  SDK,  ReST API,  RPC API |  | Supabase  Firestore  AWS Amplify |
| Google BigQuery Reviews & Ratings 2025**Google Big Query** | Append-only | Large scale data warehouse service with append-only tables. No transaction support. No foreign keys | Google | ReST HTTP/JSON API | Large scale data warehouses |  |
| **HBase** | Wide-column | Auto-sharding  Regional distribution  Good consistency | Apache | Hadoop integration | Data warehouse | BigTable |
| **InfluxDB** | Time-series  Key-value | High data ingest | Rust | SQL-like language | Real-time monitoring of IT networks and infrastructure, Predictive analytics  Powering autonomous systems. | TimescaleDB |
| **Kafka** | Streaming data | Fault tolerant, scalable, high throughput, pub/sub model for fanning selected data out, persistence | Java & Scala | Subscription to topic, streams API | Streaming apps, log analysis, machine-learning pipeline, system monitoring and alerting, replaying incidents for root cause analysis, data change capture, gradual legacy system migrations |  |
| KeyDB color icon in PNG, SVG**KeyDB** | Key-value | Faster alternative to Redis |  |  | In-memory distributed cache | Redis |
| **MariaDB, MySQL** | Relational | Serverless platforms: PlanetScale,  Sharding solutions: Percona, SingleStore | B-Tree | ODBC, C++ SDK |  |  |
| **MemCache** | Key-value  NoSQL | In-memory Serverless  Scalable  Simple data structures  Performant  Low latency  Partitioning |  | Multiple languages SDK | Caching  Session management | Redis  Dragonfly |
| **MemGraph** | Graph | Memory-first architecture Performant  Multi-threaded | C++ | Multiple languages SDK | Real-time hierarchical analytics, Holding duplicated data for quick search | Wire-compatible with Neo4l |
| **MeiliSearch** |  |  | Rust |  | Holding duplicated data for quick search |  |
| **MindsDB** | AI-tables used to imbed predictive models directly into a database |  |  |  | Integrate learning frameworks |  |
| **MongoDB** | Document oriented | Schema-less design, HA, sharding, replication | BSON format |  | Document-oriented data models, content management systems, catalogue management | Cosmos DB, DynamoDB |
| **Neo4j** | Graph database, with edges and entities | ACID, HA, clustering | Java | Cypher Query Language (CQL) | Gold standard for graph databases, recommendation engines, social networking, social media analytics, AI knowledge graphs, fraud detection |  |
| neondatabase · GitHub**Neon** | Relational | Data branching a la Git, e.g. branching, merging  Serverless  Scalable  Low latency | Rust |  |  | Postgres |
| **Oracle** | Relational | Enterprise  Partitioning  Transactions  JDK support |  | PL/SQL |  | SQL Server |
| **Postgres** | Enterprise RDBMS | ACID, OLAP, relations, transactions. Extensions: Timescale time series, sharding, serverless |  |  |  | Neon  RedShift |
| Prestodb logo - Social media & Logos Icons**Presto** | DWH  Relational  MPP |  | Facebook  Java | Connectors to most DBs | Netlix | MySQL  Amazon Athena  Trino |
| **Pulsar**  Pulsar - Audience Intelligence and Social Listening Platform | Streaming DB |  |  |  | Message Queueing | More performant than Kafka  Warpstream |
| **Redis** | Key-Value and Object store | Persistent storage  In-memory  Low latency  Complex data  Vector searches  Serverless  Scalable  Snapshots  Replication  Geospatial  Partitioning  Transactions  Pub/Sub messaging | In-memory storage | Multiple languages SDK  ODBC  JDBC | Caching  session management, gaming leaderboards | MemCache  Dragonfly  Valkey |
| **Redshift** | DWH  Relational |  | Amazon |  |  | Postgres |
| **Scylla DB** | Wide-column  Big Data  NoSQL | Highly scalable  Very low latency  Handles high volume of write operations  Sharding | C++ | C++, Rust, CQL |  | Wire-compatible with Cassandra, but faster |
| **SingleStore** | Distributed columnar Relational  DWH  Time-series | High concurrency High performance batched and streamed data ingest, query and full-text search  Vector support  Petabyte support  JSON support  Iceberg support | C++, Lucene Search | MySQL client, Python SDK | Analytics | MySQL/MariaDB (95% wire-compatible, 100 times faster)  MongoDB (100% wire-compatible)  MemSQL |
| **Snowflake** | DWH | SaaS |  | API |  |  |
| **Solr** | NoSQL | Full-text search  Faceted search DB Integration  Clustering  Highly performant  Geospatial | Java | ReST API | Document retrieval  Analytics |  |
| **Spark** | DWH | In memory caching  Big data workloads  Map-reduce  Performant  Scalable  Distributable | Apache |  | Analytics  ML, DWH, ETL,  Data Science  e.g. Tripadvisor, Yelp, Crowstrike |  |
| **SQLite (deprecated)** | Key-Value, ORM | Transactions  Weak data typing  Poor session concurrency | Google |  | Transactional applications, apps that require in-process databases |  |
| **SQL Server** | Relational | Enterprise  Partitioning  Transactions |  | T-SQL |  | Oracle |
| **Storm** | Streaming | Distributed  Realtime  Low latency  Scalablity | Apache  Clojure | JDBC  SDK to all languages | Realtime analytics,  Data stream processing  Online ML, Continuous computation  Distributed RPC  ETL | Spark |
| **Supabase** | Relational | Vector searches  Serverless  Open Source  Scalable |  | ReST API | R, Pyhon, Scala, Java SDK | Firebase  Postgres |
| **SurrealDB** | Multi-model, uses the best from all DBs, no joins but "record links" | ACID, full text search, graph, schema-less, relational, key-value, geospatial, ORM, document |  |  |  |  |
| **TimeScaleDB** | Time-series | ACID, relational, geospatial, partial aggregation, self-managed table partitioning | Cloud and On-Premise | Wire-compatible with Postgres |  | InfluxDB |
| Tidb, Powered by PingCAP**TiDB** | RDBMS | ACID, full text, graph, schema-less, relational, geospatial  Hybrid Transactional and Analytical Processing | Rust | SQL-like with JS, Graph relate statements, ReST-API |  | Wire-compatible with MySQL & MariaDB |
| **Trino** | DWH  Relational  MPP |  |  |  |  | Open-source version of Presto |
| **Valkey** | Key-Value and Object store | Persistent storage  In-memory  Low latency  Complex data  Vector searches  Serverless  Scalable  Snapshots  Replication  Geospatial  Partitioning  Transactions  Pub/Sub messaging | Google  Oracle  Snap Inc.  AWS | Multiple languages SDK | Caching  session management, gaming leaderboards | Open-source fork of Redis  MemCache  Dragonfly |
| **Vespa** |  | vector searches |  |  |  |  |
| **Xata** | Spreadsheet | Full-text search, schema editor, data branching, serverless |  | SDK for Python and TypeScript |  | Wire-compatible: Postgres & Elastic Search |

Table 19 Database system features

1. Quotes

"Seemingly small data quality issues are in reality important indications of broken business processes."

*Michael Hammer - Reengineering the Corporation [1994]*