A Chetan Varma 27-06-2025

Task 1: Database System Research

Below is a comparison of major relational database systems based on core technical specifications. These metrics help evaluate performance, scalability, and suitability for enterprise-level applications.

Key Specifications:

Specification	MySQL	SQL Server	PostgreSQL	MariaDB	Oracle Database	Amazon Aurora
Max CPU Cores	No hard limit (scales to 64+ cores)	Up to 640 cores (Enterprise)	Scales linearly up to 64 cores	Similar to MySQL	Hundreds of cores (Exadata)	Scales with AWS vCPU
Connections per Second	Up to 80,000/sec	~100,000/sec*	Variable with pooling	~150,000/sec*	~100,000/sec (with RAC)*	Variable (serverless scaling)
Data Volume Supported	Up to 256 TB	Up to 524 PB (DW edition)	Petabyte-scale	~300 TB	Petabyte-scale	Petabyte-scale
Read Replicas	Up to 5 (standard)	Up to 8 secondaries	Multiple replicas	3 to 5 replicas	Up to 30 (Data Guard)	Up to 15 Aurora Replicas

Task 2: Non-Cloud Data Warehousing Architectures

Organizations that choose not to adopt cloud infrastructure often rely on **on-premise** or **data-center-based** data warehousing solutions. These are especially preferred where data sensitivity, regulatory compliance, and low-latency access are crucial.

Two of the most widely adopted design methodologies in non-cloud data warehousing are **Kimball's dimensional modeling** and **Inmon's normalized modeling**. These methodologies guide how data is structured, stored, and accessed for analytical purposes. Each has its own strengths and is chosen based on business needs, data size, and reporting requirements.

Apart from methodologies, companies implement warehousing solutions using traditional proprietary tools or open-source platforms, depending on budget, scalability, and performance needs.

Modeling Methodologies Overview

Methodology	Approach	Modeling Style	Key Components	Best Used When
Kimball	Bottom-up	Dimensional (Star Schema)	Fact and Dimension Tables	Rapid delivery of analytics and BI
Inmon	Top-down	Normalized (3NF - 5NF)	Enterprise Data Warehouse (EDW)	Enterprise-wide consistency and control

Kimball Methodology

This method focuses on building **data marts first**, which are then integrated into a comprehensive warehouse. It uses **star schemas**, where a central fact table (e.g., sales) connects to dimension tables (e.g., customer, time, product). It is well-suited for businesses needing **quick insights** and **fast query performance** in BI tools.

Inmon Methodology

Inmon's approach starts with a **centralized data warehouse**, which is highly normalized to ensure data consistency and integrity. From this central model, subject-specific data marts are created. This is best for **large enterprises** that need **consistent data definitions** and centralized governance.

On-Premise Data Warehousing Tools

There are a range of traditional and open-source tools used to build on-premise data warehouses:

Tool	Type	Description
Teradata	Proprietary	High-performance MPP (massively parallel processing) system used in finance, telecom
IBM Netezza	Proprietary	Appliance-based data warehouse optimized for fast query performance
Apache Hive	Open-source	SQL engine built on Hadoop, best for batch processing and large-scale ETL
Greenplum	Open-source	PostgreSQL-based MPP platform for advanced analytics and parallel workloads

Teradata and **IBM Netezza** are commonly found in large enterprises due to their reliability and performance under massive data loads. **Apache Hive** is often used in Hadoop ecosystems for large-scale data lake querying. **Greenplum** provides an open-source alternative that combines the robustness of PostgreSQL with parallel execution for scalable analytics.

Task 3: AWS vs. Non-Cloud Tools Comparison

When comparing cloud-based solutions like AWS with traditional on-premise data systems, several critical differences emerge in terms of scalability, cost, and management.

Scalability:

Cloud platforms such as AWS offer automatic scaling of compute and storage resources. Services like Amazon Redshift and Aurora can dynamically adjust performance based on workload demand. In contrast, on-premise systems like Teradata or Oracle require manual hardware upgrades, which are time-consuming and limited by physical capacity.

Cost Model:

AWS operates on an operational expense (Opex) model, allowing users to pay for only the resources they consume. This makes budgeting more flexible and efficient. On-premise solutions, on the other hand, follow a capital expenditure (Capex) model, which requires significant upfront investment in servers, storage, and licensing.

Maintenance and Setup:

Cloud services can be deployed within minutes using self-service portals, and maintenance is handled entirely by the provider. In contrast, on-premise systems require weeks or even months to deploy and need dedicated IT staff for ongoing management, software updates, and hardware repairs.

Flexibility:

AWS integrates seamlessly with other cloud services like S3, Lambda, and QuickSight, enabling broader analytics and automation workflows. On-premise environments are typically more rigid and may suffer from vendor lock-in or limited compatibility with modern toolchains.

In summary, while cloud solutions offer faster deployment, scalability, and lower upfront costs, on-premise systems still play a role in environments with strict data control, compliance, or customization needs.

Feature	AWS Cloud (e.g., Redshift, Aurora)	On-Premise (e.g., Teradata, Oracle)
Scalability	Auto-scaling, elastic compute/storage	Fixed hardware, manual upgrades

Cost Model	Operational expense (Opex), pay-as-you-go	Capital expense (Capex), upfront cost
Setup Time	Minutes, self-service	Weeks to months, manual provisioning
Maintenance	Managed by AWS	Requires in-house IT teams
Flexibility	Easy integration with cloud services	Limited integrations, vendor lock-in