Università degli Studi di Modena e Reggio Emilia

Dipartimento di Ingegneria “Enzo Ferrari”
Corso di Laurea in Ingegneria Informatica

Relational Database Management System: Analisi dell’ambiente MariaDB

Relatore:
Prof.ssa Sonia Bergamaschi
Correlatore:
Phd. Luca Gagliardelli

Candidato:
Mario Cristiano

Anno Accademico 2019/2020

Indice

1. Introduzione

In questo elaborato è stata fatta un'analisi dell'ambiente MariaDB, un database management system (DBMS) relazionale, open source, nato nel 2009 dagli stessi sviluppatori di MySQL. MariaDB si sta evolvendo molto velocemente, per supportare le esigenze aziendali odierne, dall'Online Transaction Processing all'Online Analytical Processing. Il ruolo del software open source nell'infrastruttura moderna è in forte espansione. In effetti, molte aziende stanno valutando iniziative strategiche per limitare l'uso di software di proprietà. Ciò riduce i costi, favorendo, per l'organizzazione, il passaggio da spese destinate agli investimenti (Capex) a spese di tipo operativo (Opex) e consente alle imprese di beneficiare della collaborazione e dell'innovazione della comunità. Gli sviluppatori hanno lavorato anche su prodotti complementari rispetto MariaDB Server, come MariaDB MaxScale e MariaDB ColumnStore, fondamentali per grandi ambienti di tipo mission-critical. Inoltre, per il lato cloud, si sono concentrati su MariaDB SkySQL. MariaDB implementa un'architettura estendibile su più livelli, così che la comunità di utenti possa estendere nuove funzionalità al fine di soddisfare particolari esigenze. La frequenza e la qualità dei contributi della comunità sono solo alcuni tra i maggiori vantaggi di MariaDB rispetto ad altri database server. Offre, infatti, una flessibilità d'uso molto elevata, grazie ad una grande varietà di storage engine. Inoltre, MariaDB viene fornito con una serie molto più ampia e aggiornata di misure di sicurezza rispetto a MySQL, o ad altri database.
Nel presente elaborato viene brevemente descritto cos'è un DBMS e vengono paragonati i DBMS relazionali e NoSQL, con dei richiami al Modello Relazionale e al linguaggio SQL. È stata poi illustrata l'evoluzione di MariaDB, partendo da MySQL. In seguito, è stata presentata la piattaforma MariaDB, ed è stato fatto un confronto sulle novità di MariaDB rispetto MySQL. Nella parte finale dell'elaborato vengono presentati dei benchmark volti a confrontare le prestazioni delle ultime versioni di MariaDB e MySQL.

2. Cosa sono i DBMS relazionali e i DBMS NoSQL?

Il mondo in cui viviamo è sempre più digitale, e la tecnologia che ci circonda sta iniziando sempre di più a permeare la realtà in cui viviamo. Talvolta utilizziamo direttamente soluzioni digitali come app mobile di messaggistica, assistenti vocali, o servizi di banking; molto spesso, invece, usciamo da stazioni o fermate della metropolitana attraverso tornelli automatici o camminiamo per strade illuminate da lampioni, senza ricordare che questi sistemi sono gestiti in modo efficiente grazie a sistemi informatici. Infatti, soluzioni di questo tipo si basano sulla memorizzazione, gestione e aggiornamento di dati in tempo reale. Ognuno di noi possiede almeno uno smartphone, un pc, un tablet oppure un televisore. Basti pensare che 4 miliardi di persone al mondo hanno accesso alla rete e, di questi, 3,2 miliardi utilizzano i social network. E 3 miliardi usano i social da dispositivi mobili. Inoltre, ogni 60 secondi vengono mandati quasi 42 milioni di messaggi su WhatsApp, vengono caricate 500 ore di video su YouTube, su Instagram gli utenti caricano più di 347 mila storie.

Figura 1 - Quantità di dati processati ogni minuto

Negli anni il volume dei dati in circolazione è aumentato in maniera esponenziale, infatti, si è passati dai 281 Petabyte del 1986, ai circa 650 Exabyte nel 2014, fino ad arrivare agli oltre 40 Zettabyte (10^{21} byte) nel 2020.

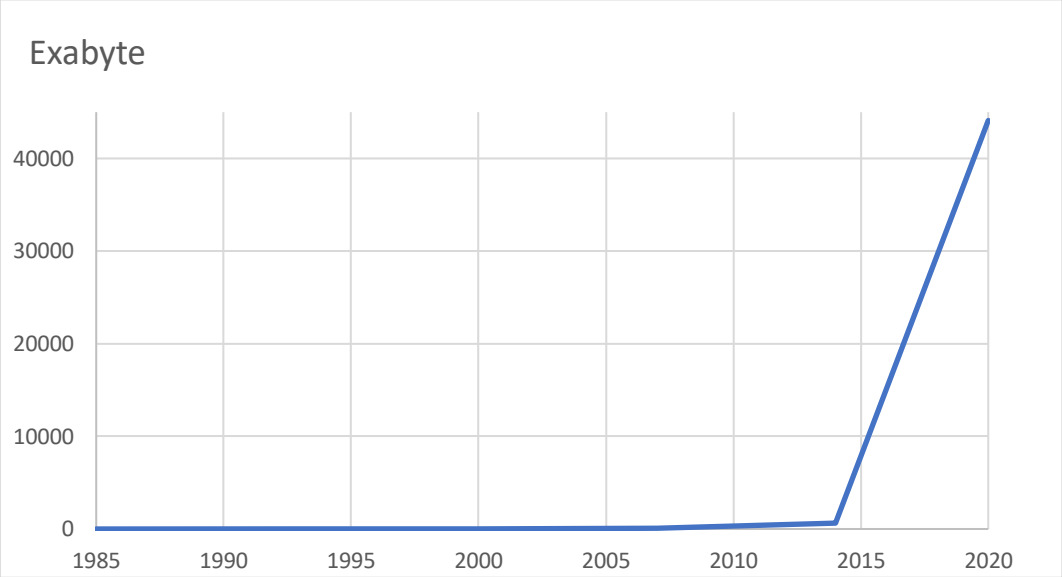

Figura 2 - Quantità di dati in circolazione

Il mondo d'oggi, quindi, si basa sullo scambio e memorizzazione di bit, e dal grande aumento di dati in circolazione, non più gestibili dai normali sistemi informatici. È stato coniato, nell'informatica moderna, il termine di Big Data (letteralmente "grandi masse di dati"). Per definizione si riferisce a quell'insieme di dati la cui dimensione va oltre la capacità dei tipici strumenti di acquisizione, archiviazione, gestione e analisi utilizzati dai database. Di fatto si tratta di una mole di dati così estesa in termini di volume, velocità e varietà da richiedere metodi e tecnologie specifiche per l'estrazione, memorizzazione e analisi. I Big Data ci pongono davanti a due sfide principali: la prima è come raccogliere questi grandi volumi di dati e la seconda è come analizzare i dati raccolti. Molte aziende hanno investito tante risorse negli ultimi anni nella Big Data Analysis, finanziando lo sviluppo di nuovi software e nuovi DBMS per la gestione di questi dati. Ma cos'è un DBMS?

Un DBMS (Database Management System) è un sistema di gestione di basi di dati, le cui principali funzioni sono quelle di garantire il mantenimento della corretta strutturazione dei dati nei diversi database gestiti e di facilitare l'accesso delle applicazioni ai dati, tramite opportune istruzioni impartite al sistema operativo. A queste funzionalità di base si aggiungono quelle di interrogazione (query) e modifica del database e la possibilità di produzione di documentazione di sintesi per il controllo delle operazioni.

Negli anni, lo sviluppo dei DBMS ha visto una crescita notevole, e si è affermata quasi ovunque la classe dei DBMS relazionali. Per interagire con tale classe di database vi è, oggi, un linguaggio considerato in tutto e per tutto uno standard, e denominato SQL.
Ma esistono anche i DBMS non relazionali, come, per esempio, quelli NoSQL. NoSQL è l'acronimo di "Not only SQL" e viene usato generalmente per indicare quei DBMS che non usano un modello di dati relazionale. I DBMS NoSQL sono appositamente ottimizzati per applicazioni che necessitano di grandi volumi di dati, latenza bassa e modelli di dati flessibili. Con grandi quantità di dati è difficile trattare tabelle relazionate tra loro da delle chiavi primarie univoche, poiché i dati sono caratterizzati da una forte eterogeneità. Inoltre, bisogna adottare modalità di memorizzazione differenti, come i sistemi distribuiti dove i dati sono partizionati per riga o per colonna. Infine, è richiesta una notevole velocità di acquisizione ed elaborazione di tali dati. Le implementazioni di NoSQL possono essere categorizzate in base alla memorizzazione:

- chiave-valore: è possibile memorizzare e recuperare in maniera rapidissima coppie di chiave e valore. Tra questi DBMS, uno dei più conosciuti è Amazon DynamoDB, utilizzato da Amazon nella gestione del proprio carrello di acquisto, oppure Redis.
- a colonne: DBMS in cui vengono memorizzati i dati ottimizzati per la ricerca su colonne, e sono progettati per gestire enormi quantità di dati distribuiti su più server. Tra questi DBMS vi è Cassandra.
- per documenti: i dati vengono incapsulati e codificati in documenti attraverso codifiche standard come XML, YAML, JSON e BSON. I documenti vengono poi indirizzati nel database tramite una chiave univoca che rappresenta quel documento. Questo tipo di DBMS è indicato quando si devono salvare dati che non hanno sempre la stessa struttura. Tra i database documentali più utilizzati troviamo MongoDB e CouchDB.
- a grafo: utilizza nodi e archi per archiviare le informazioni. Alcuni DBMS che fanno parte di questa categoria sono Neo4j, Infinite Graph e InfoGrid.

Il termine NoSQL fu usato per la prima volta nel 1998 per una base di dati relazionale, open source, che non usava un'interfaccia SQL, dal suo autore Carlo Strozzi. Il suo RDBMS NoSQL è diverso dal concetto attuale di DBMS NoSQL, per questo egli stesso suggerì che, poiché l'attuale movimento NoSQL “si discosta completamente dal modello relazionale, avrebbe dovuto quindi chiamarsi più appropriatamente “NoREL” (Not Relational)”.

In mezzo a questi due mondi, si inserisce MariaDB, un DBMS relazionale, che unisce in maniera ibrida le caratteristiche di un DBMS relazionale e le capacità di archiviazione e gestione di un DBMS NoSQL. In particolare, MariaDB ColumnStore, un motore di archiviazione a colonne che utilizza un'architettura di dati distribuita in modo parallelo, è progettato per processare grandi volumi di dati, per essere scalabile, per avere prestazioni eccezionali, e per avere una risposta in tempo reale alle query analitiche.

Modello Relazionale

Prima di parlare di DBMS relazionale è giusto introdurre il Modello Relazionale, coniato da Codd nel 1970 per semplificare la scrittura di interrogazioni sui database e per favorire l'indipendenza dei dati. Venne reso disponibile come modello logico in DBMS reali nel 1981 e ad oggi è uno dei modelli logici più utilizzati. Tale modello è basato sul concetto matematico fondamentale di relazione.
È possibile definire una *relazione* R su n domini D_1, D_2, ..., D_n, non necessariamente distinti, come un sottoinsieme del prodotto cartesiano $D_1 \times D_2 \times ... \times D_n$:

$$R \subseteq D_1 \times D_2 \times ... \times D_n$$

dove il valore n viene chiamato grado della relazione, e il numero di tuple viene chiamato cardinalità della relazione.
Una *tupla* t ordinata, invece, può essere definita come:

$$t = (v_1, v_1, ..., v_n),\ v_i \text{ appartenente a } D_i, \text{ per ogni } 1 \leq i \leq n$$

dove D_1, D_2, ..., D_n sono domini non necessariamente distinti.

Il Modello Relazionale è quindi un modello logico di rappresentazione o strutturazione dei dati di un database implementato su sistemi di gestione di basi di dati (DBMS), detti perciò sistemi di gestione di basi di dati relazionali (RDBMS).
Una relazione viene rappresentata generalmente tramite una tabella con un numero di righe pari alla cardinalità e un numero di colonne pari al grado.

$D_1 = \{v_{11}, v_{12}\}$ $D_2 = \{v_{21}, v_{22}\}$
$D_1 \times D_2 = \{(v_{11}, v_{21}), (v_{11}, v_{22}), (v_{12}, v_{21}), (v_{12}, v_{22})\}$
$R_1 = \{\{(v_{11}, v_{21}), (v_{11}, v_{22}), (v_{12}, v_{22})\}$ cardinalità = 3, grado = 2

v_{11}	v_{21}
v_{11}	v_{22}
v_{12}	v_{22}

Si definisce *attributo* il nome dato ad un dominio di una relazione. Dato un insieme di nomi di attributi, tutti diversi, $X = (A_1, A_2, \ldots, A_n)$, si può esprimere lo *schema di relazione* R(X) come una coppia costituita dal nome della relazione R e l'insieme di nomi degli attributi X.
Una relazione si basa sul concetto di *chiave*. Per chiave di una relazione si intende un sottoinsieme dei suoi attributi che identifica univocamente ogni tupla della relazione stessa. Uno schema R(X) può avere più chiavi, dette *chiavi candidate*. Tra queste viene scelta una *chiave primaria*, mentre le rimanenti vengono chiamate *chiavi alternative*.
Per introdurre in una relazione delle tuple in cui il valore di uno o più attributi non è disponibile, il dominio di una relazione viene esteso con un *valore nullo*, denotato con null, che rappresenta assenza di informazione.

Per consentire la correttezza delle informazioni nella base di dati, vengono usati dei *vincoli di integrità*. Per esempio, la definizione di una chiave di uno schema di relazione è una dichiarazione di vincolo di integrità in quanto stabilisce l'univocità dei valori assunti dagli attributi della chiave.
Esiste poi il vincolo di *Entity Integrity*, che stabilisce che gli attributi che costituiscono la chiave primaria o alternativa di una relazione non possono assumere valore null. Questo perché altrimenti non sarebbe possibile controllare l'univocità dei valori assunti da una chiave in presenza di valori nulli.
Nel Modello Relazionale, i riferimenti tra le tuple delle relazioni vengono stabiliti tramite i valori assunti dagli attributi nelle tuple. Per assicurare che quando in una tupla si utilizza il valore di un attributo per riferirsi ad un'altra tupla, quest'ultima sia una tupla esistente, viene usato il *Vincolo di Integrità Referenziale*. Questo vincolo viene dichiarato specificando una *Foreign Key*, cioè un insieme di attributi $FK = \{FK_1, FK_2, \ldots, FK_n\}$ di uno schema di relazione R_1, e una *Chiave della Relazione riferita* $K = \{K_1, K_2, \ldots, K_m\}$ di uno schema di relazione R_2, non necessariamente distinto da R_1, con m = n. Un'istanza $r = \{r_1, r_2, \ldots\}$, dove r_i è una relazione su R_i, su un insieme di schemi di relazioni R soddisfa il vincolo di integrità referenziale se i valori nulli sulla foreign key FK di ciascuna tupla di r_1 sono valori della chiave K di r_2, o sono valori nulli.

Relational Database Management System (RDBMS)

Un modo semplice per pensare un database relazionale è paragonandolo a raccolte di dati correlate tra loro. Pensiamo per esempio ad un'azienda con una gestione delle vendite e una gestione del magazzino dove sono stoccati i prodotti. Ad ogni ordine di vendita è associato un prodotto del magazzino. Un database e un software che gestisce questo

database, Database Management System (DBMS), ci vengono in aiuto nella corretta gestione dei prodotti dell'azienda.
La maggior parte dei database oggi sono relazionali, denominati in questo modo perché si occupano di tabelle di dati correlate da un campo comune.

Tabella 1 - Prodotti magazzino

id_prodotto	**descrizione**	**prezzo**
1	Laptop	€400
2	Smartphone	€200

Tabella 2 - Ordini

cod_ordine	**id_prodotto**	**quantità**
1000	2	3
1001	1	4

Nella prima tabella sono riportati i prodotti presenti nel magazzino, mentre nella seconda tabella sono elencati gli ordini effettuati dai clienti. Le due tabelle sono correlate da un campo comune "id_prodotto", che rappresenta la chiave univoca di un determinato prodotto.
Ogni tabella è composta da righe e colonne. Ogni riga contiene dei dati su una certa entità (prodotto o ordine), e viene chiamata record (per esempio la prima riga della prima tabella contiene l'id, la descrizione e il prezzo di un laptop). Una colonna invece è un attributo e contiene dati relativi ai record (come la descrizione o il prezzo di un determinato prodotto). L'insieme di dati relativi ad attributi di una determinata riga della tabella è chiamato tupla.

Un modo per comunicare con questi dati e manipolarli è dato dall'utilizzo dell'SQL, un linguaggio di query strutturato, che permette di cercare record o apportare delle modifiche ad essi. Quasi tutti i DBMS utilizzano l’SQL, sebbene molti abbiano aggiunto delle proprie varianti.
Il modello relazionale è il migliore modo per mantenere la coerenza e la consistenza dei dati tra applicazioni e copie di database. Per altri tipi di database, ad esempio NoSQL, è difficile mantenere questo livello di coerenza immediato con una grande quantità di dati. La grande sfida di MariaDB è proprio quella di mettersi in mezzo a questi due mondi, riuscendo a coniugare le esigenze di gestione di grandi database ad un modello di manipolazione dei dati di tipo relazionale.

SQL

Il linguaggio SQL (Structured Query Language) è il linguaggio standard per la definizione, manipolazione e interrogazione delle basi di dati relazionali.
È nato nel 1974 da Donald Chamberlin, nei laboratori dell'IBM. Inizialmente si chiamava "SEQUEL" ed era lo strumento per lavorare con database che seguivano il modello relazionale. Negli anni seguenti vennero sviluppati nuovi prototipi e la nuova versione del 1977 prese il nome di SQL. Su di esso l'IBM sviluppò un database management system chiamato "System R", in grado di eseguire interrogazioni SQL. Ma poi negli anni altre società iniziarono a sviluppare i loro prodotti basati sull'SQL.
Il linguaggio SQL è stato successivamente sottoposto a varie fasi di standardizzazione, la cui versione più recente risale al 2016.
Ad oggi è il linguaggio di interrogazione più diffuso tra quelli usati per l'interazione con i principali Database Management Systems (DBMS) relazionali.

L'SQL è un linguaggio usato per interrogare e gestire basi di dati (selezione, inserimento, cancellazione dati, aggiornamento, ...) mediante l'utilizzo di costrutti di programmazione denominati query. È un linguaggio dichiarativo[1], che opera su multiset di tuple, cioè su insiemi di tuple che possono contenere duplicati, e premette di esprimere tutte le interrogazioni formulabili in algebra relazionale.

[1] A differenza dei linguaggi procedurali, un linguaggio dichiarativo non specifica la sequenza di operazioni da compiere per ottenere il risultato.

3. Uno sguardo al passato di MariaDB

Figura 3 - Michael Widenius

Inizialmente era MySQL...

Nel 1979 Michael "Monty" Widenius, informatico finlandese, scrisse UNIREG, un primo strumento molto primitivo basato su Unix per la gestione dei database, ma abbastanza in linea con gli standard dell'epoca, che sarebbe poi stata la base per il futuro MySQL.
Nel 1994 provò ad utilizzarlo come backend per i siti web ma le prestazioni di UNIREG non erano sufficienti per elaborare dinamicamente pagine web, di conseguenza iniziò a lavorare insieme a David Axmark e Allan Larsson ad un nuovo progetto evolutosi poi nella prima versione di MySQL. MySQL venne reso pubblico nel 1995, stesso anno in cui, insieme, fondarono la società MySQL AB. Nel giro di pochi anni MySQL è diventato lo standard più diffuso per i siti web, scalzando abbastanza velocemente tutta la concorrenza.

I segreti del grande successo di MySQL possono essere molteplici. La prima ragione potrebbe essere un fattore culturale, poiché il web era nato come condivisione di informazioni e MySQL è un prodotto Open Source, essendo distribuito con la licenza GNU GPL.
Inoltre, i siti web avevano bisogno di prodotti agili, e per agile si intende che MySQL era gratuito, più leggero rispetto ai concorrenti, veloce in termini prestazionali, facile da imparare e da mantenere ma, allo stesso tempo, molto robusto.
Il web degli anni '90 inoltre era molto semplice e questa semplicità si poteva riscontrare anche in MySQL. I bisogni dei siti web poi si sono evoluti nel corso degli anni abbastanza rapidamente, e con loro si è evoluto anche MySQL.

Ma, nel 2008, l'allora gigante statunitense Sun Microsystems acquisì MySQL. In quell'anno Monty Widenius lasciò il progetto originale per dedicarsi ad un progetto più specifico, il plugin Aria.
L'anno successivo, nel 2009, Oracle acquisisce la Sun Microsystems e di conseguenza anche MySQL.

Nello stesso anno Michael crea MariaDB, nato dai codici sorgente di MySQL. Lo seguirono nel suo progetto anche alcuni dei core engineers fuoriusciti da Oracle.

...l'arrivo di MariaDB

Figura 4 - Logo ufficiale di MariaDB

MariaDB nasce dal codice di MySQL. Ovviamente il team ha aggiunto molte funzionalità, ma mantenendo la compatibilità con il passato, quindi tutte quelle funzioni, plugin, query che funzionavano con MySQL continuano a funzionare su MariaDB. Aperto ai contributi della comunità, l'area di sviluppo principale era lo storage engine Aria, precedentemente chiamato Maria, da cui è derivato poi MariaDB, e si trattava di un'evoluzione di MyISAM.
Quando, all'inizio del 2008, la Sun Microsystems ha acquistato MySQL AB, Widenius si è trovato molto a disagio nella nuova situazione lavorativa, criticando la gestione dello sviluppo della versione 5.1 di MySQL. Dopo l'acquisizione della Sun da parte di Oracle, Widenius ha fondato il fork di MySQL, MariaDB, e, per la sua gestione, ha avviato una nuova società chiamata Monty Program AB.
La prima versione di MariaDB, uscita nell'aprile del 2009, è stata la 5.1, e non la 1.1 come si potrebbe immaginare, per indicare che era pienamente compatibile con MySQL 5.1. Inizialmente era disponibile solo per Linux, ma in seguito, nell'ottobre dello stesso anno, venne rilasciata la versione anche per Windows. Sono poi uscite la 5.2 e la 5.3, ancora pienamente compatibili con la 5.1 di MySQL. Al momento dell'uscita della versione di MySQL 5.5, contemporaneamente è uscita la versione 5.5 di MariaDB, sempre ad indicare la piena compatibilità tra i due.
Con la versione successiva di MySQL, la 5.6, MariaDB è passato alla 10.0. Questo per indicare che non stava più seguendo la strada di MySQL, ma stava iniziando ad imboccare una sua via. Le incompatibilità erano inizialmente molto poche, principalmente riguardo la replica (cioè non si può utilizzare MariaDB 10.0 come master e MySQL 5.6 come slave[2], mentre invece il contrario funziona) e riguardo alcune caratteristiche mancanti, come la possibilità di integrare il motore InnoDB con MySQL Memcached plugin.

[2] Con architettura di tipo master-slave si indica un tipo di architettura per computer che permette di creare un rapporto tra hardware, in cui uno ha il pieno controllo dell'altro.
In particolare, si definisce master il dispositivo informatico che prende il controllo del bus, incomincia l'interazione, mentre con slave ci si riferisca a quello che risponde al primo.

Nel dicembre 2012 è nata la MariaDB Foundation, allo scopo di garantire che MariaDB rimanga sempre un software libero, aperto al dialogo con la comunità e con una buona interoperabilità verso altri programmi.
Nel 2013 Monty Program Ab si è fusa con SkySQL, società di consulenza formata principalmente da ex dipendenti di Oracle, Sun Microsystems e MySQL Ab, diventando MariaDB Corporation. La proprietà del marchio MariaDB è rimasta comunque alla MariaDB Foundation.

Curiosamente MySQL e MariaDB si chiamano così come rimando alle prime due figlie di Monty Widenius, My e Maria. Attualmente l'ultima versione di MySQL è la 8.0 mentre quella di MariaDB è la 10.5.

4. Panoramica su MariaDB

MariaDB Community Server

MariaDB Community Server è un database management system relazionale (RDBMS), basato sullo standard SQL, Open Source e gratuito, distribuito sotto licenza GNU GPL v2. Si basa su MySQL, ereditandone buona parte delle caratteristiche (tra cui comandi, interfacce, librerie e API), e introducendone di nuove.

MariaDB vanta una grande varietà di storage engine, ereditando da MySQL InnoDB, MyISAM, Blackhole, CSV, Memory, Archive e Merge. Come vedremo in seguito, però, ne aggiunge molti propri. Ma cos'è uno storage engine?
Uno Storage Engine è una libreria che un database management system usa per implementare la gestione fisica dei dati. Alcuni dei compiti affidati agli Storage Engine sono: scrittura, lettura e aggiornamento dei record, indicizzazione, cache, transazioni.

- InnoDB: è lo storage engine standard per MySQL. Fino alla versione 10.1 MariaDB ha utilizzato un fork di InnoDB, più ampliato e completo, XtraDB, come motore di default. Dalla versione 10.2 anche MariaDB impiega InnoDB come sistema di archiviazione standard. InnoDB consente accessi di lettura e scrittura sicuri nelle transazioni, fornisce un vincolo di integrità referenziale della chiave esterna, supporta il blocco a livello di riga, il ripristino da arresto anomalo del sistema, le transazioni XA e il controllo della concorrenza multi-versione. Nella maggior parte dei casi InnoDB offre prestazioni migliori rispetto ad altri storage engine.
- MyISAM: era il motore di archiviazione predefinito per MySQL fino alla versione 5.5, poi sostituito da InnoDB. È un motore leggero, con ridotto ingombro dei dati, non transazionale, ma con prestazioni eccezionali. Una tabella MyISAM è archiviata in tre file su disco (ci sono un file di definizione della tabella con un'estensione di *.FRM*, un file di dati con l'estensione *.MYD* e un file di indice con l'estensione *.MYI*). Supporta:
  - Indici full-text: è un indice di tipo FULLTEXT con una sintassi del tipo *MATCH() ... AGAINST()*, dove *MATCH ()* accetta un elenco separato da virgole che nomina le colonne in cui cercare, mentre *AGAINST()* richiede una stringa da cercare e un modificatore opzionale che indica il tipo di ricerca da eseguire.

- I tipi di dati GIS (Geographic Information System), i quali consentono la creazione, l'archiviazione e l'analisi di caratteristiche geografiche.
- Inserti simultanei: questa funzione consente di eseguire le *SELECT* durante le operazioni di *INSERT*, riducendo la contesa.
- Un buffer delle chiavi segmentato: risolve il problema di contesa dei thread per il buffer delle chiavi, facendo sì che ogni funzione che necessiti di tale blocco, possa acquisire solo il segmento che gli è stato assegnato.
- Il file di dati e il file di indice possono essere posizionati su dispositivi diversi per migliorare la velocità.
- Formati a lunghezza fissa, dinamici e compressi: questi possono essere impostati con l'opzione *ROW FORMAT* nell'istruzione *CREATE TABLE*, oppure verranno scelti automaticamente a seconda delle colonne contenute nella tabella.

Non supporta però le transazioni, le chiavi esterne e il blocco delle righe.

- Blackhole: come suggerisce il nome, questo storage engine accetta dati ma non li memorizza, e restituisce sempre un risultato vuoto. Tale funzionalità può essere utile nella progettazione di database distribuiti in cui i dati vengono replicati automaticamente, ma non archiviati localmente. Inoltre, questo storage engine può essere utilizzato per eseguire dei test, per esempio, sulle prestazioni, o altro. Oppure può essere utile se si desidera eseguire regole di filtro complesse su uno slave senza incorrere in alcun sovraccarico su un master.
- CSV: questo motore di archiviazione può leggere e memorizzare dati in formato CSV (Comma-Separated Values), questi poi possono essere facilmente integrati in altre applicazioni. Però non supporta le transazioni, e, inoltre, le tabelle CSV non supportano l'indicizzazione e non possono essere partizionate.
- Memory: le tabelle vengono archiviate in memoria anziché su disco. È un motore di archiviazione veloce, che però non supporta le transazioni. Lo storage engine Memory è ideale per creare tabelle temporanee o ricerche rapide, ma i dati vengono persi quando il database viene riavviato.
- Archive: è ideale per archiviare e recuperare grandi quantità di dati. Comprime i dati man mano che vengono inseriti. Una tabella che utilizza tale motore viene archiviata in due file su disco: un file di definizione della tabella con estensione *.FRM* e un file di dati con estensione *.ARZ*. Non supporta, però, le transazioni.
- Merge: è una raccolta di tabelle MyISAM identiche che possono essere utilizzate come una sola. Per "identico" si intende che tutte le tabelle hanno informazioni di colonna e indice identiche. Non è possibile unire tabelle in cui le colonne sono

elencate in un ordine diverso, o non hanno esattamente le stesse colonne o hanno gli indici in ordine diverso. Ogni indice in una tabella Merge deve corrispondere a un indice nelle tabelle MyISAM sottostanti, ma non è vero il contrario. Inoltre, una tabella Merge non può avere una PRIMARY KEY o indici UNIQUE, perché non può applicare l'unicità su tutte le tabelle sottostanti. L'unione delle tabelle aiuta a gestire più facilmente grandi volumi di dati, risultando essere una soluzione ideale per ambienti di data warehousing.

Con il comando *SHOW ENGINES* vengono mostrati i motori di archiviazione attualmente presenti su MariaDB Server:

```
MariaDB [(none)]> show engines;
+--------------------+---------+------------------------------------------------------------+--------------+----+------------+
| Engine             | Support | Comment                                                    | Transactions | XA | Savepoints |
+--------------------+---------+------------------------------------------------------------+--------------+----+------------+
| CSV                | YES     | Stores tables as CSV files                                 | NO           | NO | NO         |
| MRG_MyISAM         | YES     | Collection of identical MyISAM tables                      | NO           | NO | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful                       | NO           | NO | NO         |
|                    |         |        for temporary tables                                |              |    |            |
| Aria               | YES     | Crash-safe tables with MyISAM heritage. Used for           | NO           | NO | NO         |
|                    |         |        internal temporary tables and privilege tables      |              |    |            |
| MyISAM             | YES     | Non-transactional engine with good                         | NO           | NO | NO         |
|                    |         |        performance and small data footprint                |              |    |            |
| SEQUENCE           | YES     | Generated tables filled with sequential values             | YES          | NO | YES        |
| InnoDB             | DEFAULT | Supports transactions, row-level locking, foreign          | YES          | YES| YES        |
|                    |         |        keys and encryption for tables                      |              |    |            |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                         | NO           | NO | NO         |
+--------------------+---------+------------------------------------------------------------+--------------+----+------------+
```

MariaDB è un DBMS molto flessibile, perché offre una quantità molto variegata di storage engine. Non c'è un motore adatto a tutte le situazioni di lavoro, alcuni si comportano meglio in determinate condizioni e peggiorano in altre situazioni. Una soluzione più sicura, per esempio, richiede più risorse computazionali, rallentando il sistema. Di sicuro se vogliamo uno storage utile nella maggior parte dei casi, che supporti le transazioni e le chiavi esterne, ci si può rivolgere ad InnoDB. Mentre se desideriamo effettuare delle ricerche full-text c'è MyISAM. Si possono utilizzare poi Galera Cluster, TokuDB, Spider o ColumnStore quando si desidera suddividere il carico del database su più server o ottimizzare attraverso il ridimensionamento. Archive è uno dei migliori motori per l'archiviazione. CONNECT consente l'accesso a diversi tipi di file di testo e risorse remote come se fossero normali tabelle MariaDB. Mroonga e SphinxSE sono ottimizzati per la ricerca. Memory è ottimizzato al meglio per cache di sola lettura da altre tabelle, o per aree di lavoro temporanee.
Molti di questi motori verranno analizzati più nel dettaglio in seguito, in un confronto tra MariaDB e MySQL.

Un motore di archiviazione può essere specificato nel momento di creazione della tabella: *ENGINE = 'MyISAM'*. Nel caso in cui non venga specificato nulla, viene richiamato il motore di default, che nel caso di MariaDB è InnoDB. Per visionare lo storage engine associato ad una particolare tabella si usa la seguente istruzione:

```
SELECT ENGINE FROM information_schema.TABLES
        WHERE TABLE_SCHEMA='testDB'
        AND TABLE_NAME='table_01';
```

```
+---------+
| engine  |
+---------+
| InnoDB |
+---------+
```

testDB e *table_01* sono rispettivamente il nome del database e il nome della tabella, mentre *information_schema* è una tabella che memorizza le informazioni tecniche riguardanti il database.
Per cambiare lo storage engine di una tabella si può utilizzare, invece, la seguente istruzione: *ALTER TABLE table_01 ENGINE='MyISAM'*.

MariaDB Platform

L'obiettivo di MariaDB Platform è quello di fornire una soluzione di database il più possibile completa da diversi punti di vista, come la versatilità, la scalabilità e le possibilità architetturali.
MariaDB può essere distribuito come database transazionale, ma, già da alcuni anni ormai, con l'aggiunta di ColumnStore, gli sviluppatori hanno incominciato a competere con il mondo del data warehouse e delle soluzioni per il mondo analitico, con una soluzione di database a colonne che può combinarsi con la soluzione transazionale in una modalità di traffico ibrido, che vede insieme l'archiviazione per righe e per colonne e consente di eseguire transazioni che combinano i due modi di archiviare e gestire i dati.
Dal punto di vista della scalabilità, MariaDB può essere distribuito in diversi modi, partendo dalla singola istanza standalone, sempre più utilizzata negli ambienti di sviluppo, fino ad arrivare ad ambienti di alta affidabilità, replicati con qualsiasi scala.
L'architettura e la sua flessibilità sono ciò che distingue MariaDB Platform dagli altri sistemi. In mezzo a centinaia di DBMS disponibili molto specializzati, MariaDB Platform si può vedere come un database più general purpose, perché dietro lo standard dell'SQL si possono avere motori specializzati, in grado di trasformare le query secondo un motore di gestione dei dati specifico, in grado di ottimizzare i carichi di lavoro e di consentire di

crescere e di scalare in qualsiasi momento un'architettura da una singola istanza standalone ad una replicata, propagata o distribuita, oppure da un formato per righe ad un formato per colonne se, per esempio, in certe situazioni il carico analitico sta aumentando. La piattaforma è quindi costruita attorno all'idea di servire molti carichi di lavoro diversi, cercando di eseguirli nella maniera più efficiente possibile, su qualsiasi scala.

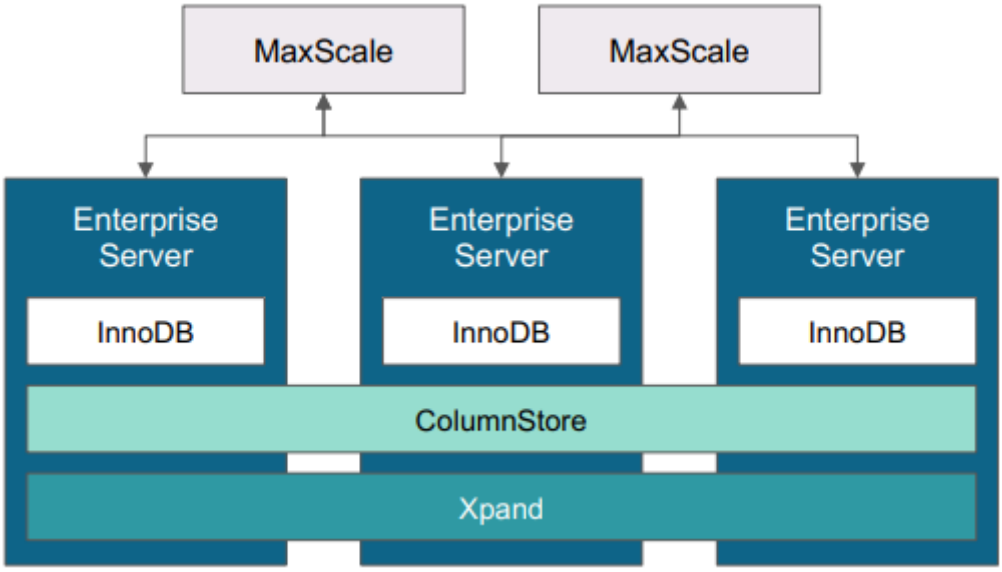

Figura 5 - MariaDB Platform

MariaDB Platform è composta da:

- MariaDB Enterprise Server
- MariaDB MaxScale
- MariaDB ColumnStore
- MariaDB Xpand
- MariaDB Connectors
- MariaDB SkySQL

MariaDB Enterprise Server

MariaDB Enterprise Server ha le sue radici nella versione community di MariaDB, alle quali sono state aggiunte e migliorate alcune funzionalità, tra cui plugin, engine e strumenti interni, rendendolo di fatto più adatto a soluzioni ad alte prestazioni.
Di fatto MariaDB Enterprise Server è una versione migliorata, rafforzata e più sicura rispetto MariaDB Community Server, creata per fornire ai clienti l'affidabilità, la stabilità e il supporto a lungo termine necessario. Inoltre, esso fornisce ai clienti una maggiore efficienza operativa quando si tratta di supportare l'azienda e applicazioni mission-critical

su database di grandi dimensioni. Per esempio, include funzionalità per eseguire backup frequenti senza influire sulle applicazioni, applicando la crittografia end-to-end completa.

Plugin e strumenti aggiuntivi:

- Enterprise Audit: lo scopo di questo plugin è quello di registrare l'attività del server. Per ogni sessione viene registrato chi si è connesso al server (nome utente e host), quali query sono state eseguite, quali tabelle sono state visitate e le variabili del server che sono state modificate. Nella versione enterprise, il plugin è stato ridisegnato per sistemi che prevedono un livello di sicurezza molto più raffinato rispetto a quello fornito nella versione community.
- Enterprise Backup: strumento di backup nato per avere un bassissimo impatto sulle prestazioni di sistema, svolgendo un'attività meno invasiva e meno bloccante dal punto di vista delle prestazioni.
- Enterprise Cluster: è una soluzione di replica multi-primaria (cioè le modifiche apportate a qualsiasi nodo del cluster vengono replicate su ogni altro nodo del cluster) che funge da alternativa alla replica singola. Rispetto alla versione community, sono stati arricchiti dal punto di vista della sicurezza elementi di crittografia end-to-end nelle soluzioni clusterizzate.
- Enterprise Federation: consente l'accesso ad alte prestazioni alle tabelle nei database remoti utilizzando connessioni ODBC (Open DataBase Connectivity).
- Xpand: aggiunge il supporto per l'SQL distribuito senza dover cambiare i database o modificare le applicazioni.
- HashiCorp Vault Encryption Plugin: consente di utilizzare HashiCorp Vault (progettato per controllare l'accesso a dati sensibili) per gestire le chiavi di crittografia dei dati inattivi di MariaDB.

Ci sono stati poi miglioramenti interni riguardanti il motore InnoDB e la replica: viene impedito l'arresto dei nodi primari prima del completamento di tutta la replica, per evitare che un arresto provochi la perdita dei dati.

MariaDB MaxScale

MariaDB MaxScale è un proxy[3] di database avanzato per la piattaforma MariaDB, che inoltra le istruzioni del database a uno o più server. Essendo un proxy consente di astrarre il livello applicativo dal livello infrastrutturale sottostante, e quindi funge da strato di

[3] Tipo di server che funge da intermediario per le richieste da parte dei client verso altri server.

disaccoppiamento tra lo strato applicativo e quello di database. Le applicazioni del client devono conoscere l'indirizzo ip e la porta di MaxScale a cui connettersi, senza conoscere se si tratta di un nodo primario o se si hanno due o più repliche.
MaxScale rende molto più semplice dal punto di vista dello sviluppo delle applicazioni l'essere indipendenti dal database, senza essere per forza consapevoli di cambi all'infrastruttura. Anche dal punto di vista del database administrator si è più liberi di apportare modifiche all'infrastruttura senza dover influire o impattare sull'applicazione.
Una caratteristica importante è il Query routing, ovvero l'indirizzamento delle query, e quindi il fatto di poter dividere il carico di lettura e scrittura sui server a disposizione.
Un'altra caratteristica importante di MaxScale è quella di garantire failover automatico, quindi di essere in grado di rendersi conto se un nodo primario è caduto e automaticamente promuovere una migliore replica possibile a ruolo di nuovo nodo primario e rimandare le query che non sono state eseguite, a causa di questo fail, al nuovo nodo primario. Inoltre, vengono informati tutti i nodi che il nuovo nodo primario è quello appena promosso.
Stando tra lo strato applicativo e quello infrastrutturale, MaxScale può anche giocare il ruolo di Database firewall[4], andando a gestire il traffico delle query dal nodo primario, e bloccando attacchi malevoli al database.
Altre caratteristiche di MaxScale sono la gestione delle cache delle query e la possibilità di acquisire dei dati modificati, gestendo quindi il Change-data-capture. Infine, quando più istanze MaxScale vengono utilizzate in una distribuzione, il monitoraggio cooperativo deve garantire che solo un'istanza MaxScale esegua operazioni di failover automatico in un dato momento. Sfrutta la funzione GET_LOCK() di MariaDB, quindi, solo un’istanza sarà veramente in grado di prendere la gestione di quell'architettura.

MariaDB ColumnStore

Con la versione 1.5, ColumnStore non è più un'installazione separata orientata all'analisi per colonne dei dati in modalità analitica, ma è diventato un vero e proprio storage engine, integrato come tutti gli altri engine di sistema in MariaDB. ColumnStore fornisce l'archiviazione per colonne e distribuita, quindi si ha la possibilità di poter parallelizzare su più nodi e quindi scalare sia verticalmente e sia orizzontalmente l'esecuzione delle query complesse. Sono stati eliminati gli indici, perché le colonne sono indici loro stesse. Inoltre, è stato aggiunto il supporto per l'archiviazione ad oggetti. Per esempio, se si

[4] È un componente hardware o software di una rete, che fornisce una protezione in termini di sicurezza informatica della rete stessa.

dispone di un cloud computing[5] elevato c'è la possibilità di separare la parte di calcolo dallo spazio di archiviazione, considerando anche che l'Object storage è molto meno costoso dello storage a blocchi.
Un'altra caratteristica dell'ultima versione di ColumnStore è una sempre più marcata integrazione con Kafka[6]. Quindi se si dispone di un'infrastruttura che conta molto sulla distribuzione di messaggi tramite Kafka, MariaDB ColumnStore può attingere a quell'ambiente, estrarre i dati e utilizzarli per sfruttare le capacità di analisi e di data warehousing che ne sono propri.
È sempre possibile però combinare i dati di ColumnStore con quelli di InnoDB per mescolare e abbinare le query con delle join di dati supportati da engine diversi.

MariaDB Xpand

MariaDB Xpand è uno Storage Engine transazionale, con supporto ad ACID[7] e all'SQL distribuito. Ogni nodo Xpand è in grado di eseguire sia letture che scritture. Quando una query viene ricevuta da un'istanza di MariaDB Enterprise Server, viene valutata da un ottimizzatore di query, e porzioni della query vengono inviate ai singoli nodi Xpand pertinenti. I risultati vengono poi raccolti e un singolo set di risultati viene restituito al client. La memoria e l'archiviazione dei nodi Xpand non sono condivise. Non si parla, quindi, di distribuzione dei dati, ma di transazioni distribuite, con i dati che si spostano in automatico nel sistema, per mantenere un'elevata affidabilità, garantendo sempre una loro replica, e, inoltre, per evitare di avere dei punti di aggregazione dove tutte le transazioni si accumulano e finiscono per essere concentrate in un nodo in particolare.
Xpand mantiene due repliche di tutti i dati, utilizzando un processo di ribilanciamento che viene eseguito in background. Può subire, quindi, un guasto di un singolo nodo o di una zona senza perdita di dati.

---

[5] Paradigma di erogazione di servizi offerti su richiesta da un fornitore a un cliente finale attraverso la rete internet.

[6] Apache Kafka è una piattaforma di event streaming distribuita e open-source, che consente l'archiviazione, la lettura e l'analisi dei dati, in grado di gestire miliardi di eventi al giorno. L'event streaming è la pratica di acquisire dati in tempo reale da fonti di eventi come database, sensori, dispositivi mobili, servizi cloud e applicazioni software sotto forma di flussi di eventi, memorizzare questi flussi di eventi in modo durevole per un successivo recupero, manipolare, elaborare e reagire ai flussi di eventi in tempo reale, e infine instradare i flussi di eventi a diverse tecnologie di destinazione secondo diversa necessità. Kafka, quindi, scrive e legge flussi di eventi, inclusa l'importazione e l'esportazione continua dei dati da altri sistemi, archivia flussi di eventi in modo durevole e affidabile per tutto il tempo desiderato, ed elabora flussi di eventi nel momento in cui si verificano o in modo retrospettivo.

[7] Acronimo di Atomicità, Coerenza, Isolamento e Persistenza ed indica le proprietà logiche che devono avere le transazioni nella base di dati, tipiche di un database relazionale.

Infine, se il carico su MariaDB Xpand aumenta, si può ridimensionare l'architettura, aggiungendo nuovi nodi. Il processo di ribilanciamento, poi, ridistribuisce i dati dai nodi esistenti.

MariaDB Connectors

Un Connector MariaDB è un'API che permette di collegare applicazioni scritte in un determinato linguaggio a database MariaDB. MariaDB supporta i linguaggi C, C++, Java, JavaScript, ODBC, Python.

MariaDB SkySQL

MariaDB SkySQL è il primo e unico database-as-a-service (DBaaS)[8] a portare tutta potenza di MariaDB Platform nell'ambiente cloud, combinando potenti strumenti aziendali con il supporto a livello mondiale. È ideato per applicazioni aziendali e di tipo mission-critical, sia che si tratti di un singolo database di sviluppo oppure di migliaia di database.
MariaDB SkySQL fornisce:

- MariaDB Platform for Transactions: costruita su MariaDB Enterprise Server e il motore di archiviazione InnoDB con archiviazione persistente su SSD, per fornire l'elaborazione transazionale.
- MariaDB Platform for Analytics: costruita su MariaDB Enterprise Server e il motore di archiviazione MariaDB ColumnStore, per fornire un'elaborazione analitica sostenuta da uno storage ad oggetti scalabile.
- MariaDB Platform per transazioni intelligenti: basata su MariaDB Enterprise Server e sui motori di archiviazione InnoDB e MariaDB ColumnStore, per fornire HTAP (Hybrid Transactional / Analytical Processing).
- MariaDB Platform per Scale-Out Transactions: costruita su MariaDB Enterprise Server e il motore di storage Xpand per l'SQL distribuito con High Availability (HA), alta disponibilità, e fault tolerance, tolleranza ai guasti.

MariaDB gestisce tutte le esigenze hardware e infrastrutturali, nonché backup e monitoraggi, tramite un team globale di amministratori di database (SkyDBA). I servizi

[8] DBaaS (noto anche come servizio di database gestito) è un servizio di cloud computing che consente agli utenti di accedere e utilizzare un sistema di database cloud senza acquistare e configurare il proprio hardware, installare il proprio software di database o gestire il database da soli. Il fornitore di servizi cloud si occupa di tutto, dagli aggiornamenti periodici ai backup per garantire che il sistema di database rimanga disponibile e sicuro.

di SkySQL vengono attualmente forniti utilizzando GCP (Google Cloud Platform) e GKE (Google Kubernetes Engine[9]). Inoltre, SkySQL è progettato con una solida architettura di sicurezza, crittografando i dati in transito attraverso i protocolli SSL (Secure Sockets Layer) e TLS (Transport Layer Security).

Figura 6 - MariaDB SkySQL

Architettura combinata

MariaDB Platform include più motori di archiviazione per consolidare una varietà ampia di carichi di lavoro del database, da carichi a scrittura intensiva (come, ad esempio, applicazioni IoT) fino ad archiviazioni scalabili chiave-valore (tipiche di database NoSQL), il tutto senza sacrificare prestazioni, scalabilità e flessibilità. Quindi, la piattaforma, implementa un'architettura di archiviazione collegabile, perché carichi di lavoro diversi hanno caratteristiche di archiviazione diverse. Per esempio, la struttura dati ottima per lavori di scrittura e lettura non è la migliore per carichi di lavoro ad alta intensità di scrittura e viceversa. La stessa cosa vale per i carichi di lavoro transazionali e analitici. Consentendo a diverse istanze di database o tabelle di utilizzare diversi motori di archiviazione, MariaDB Platform può supportare comunque una grande varietà di carichi di lavoro.

[9] Kubernetes è una piattaforma open source, sviluppata da Google, utilizzata per l'orchestrazione e la gestione automatica dei container.

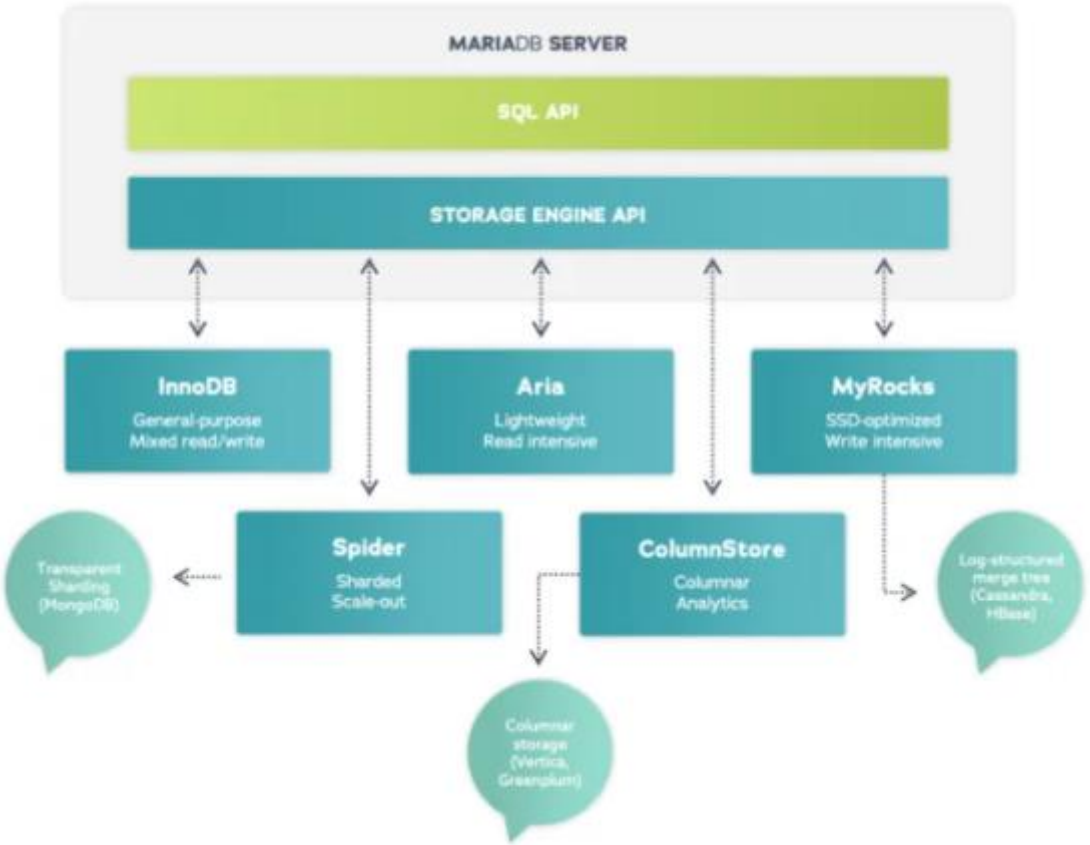

Figura 7 - Architettura combinata degli Storage MariaDB

Nella figura sotto, a sinistra, è riportato un esempio che mostra il motore Spider combinato con InnoDB per scalare parallelamente letture, scritture e archiviazione. Questa combinazione può essere utilizzata con il tipo di dati JSON per creare una distribuzione NoSQL, scalabile e flessibile. Nell'esempio di destra, invece, per supportare vari microservizi, gli stessi dati vengono archiviati in diversi motori di archiviazione: MyRocks per le scritture, InnoDB per le letture e ColumnStore per le analisi.

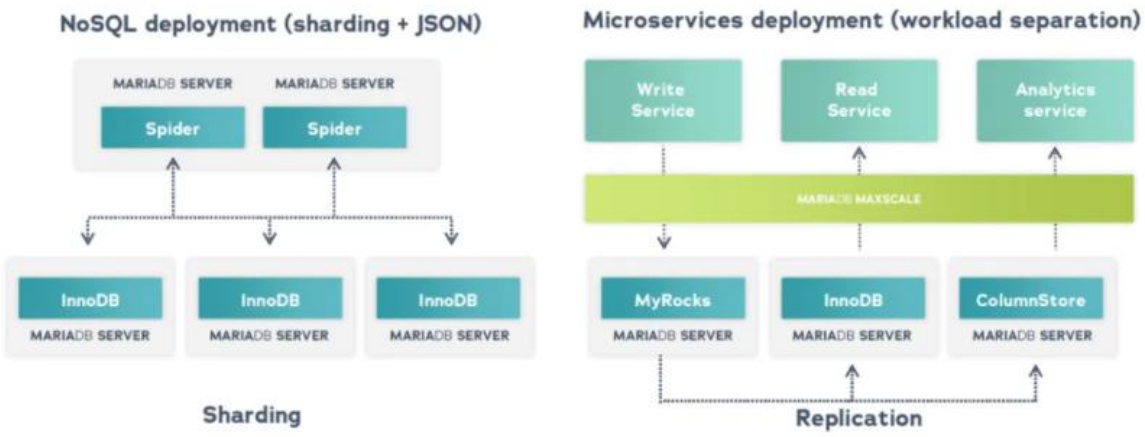

Figura 8 - Esempi di combinazione degli Storage

MariaDB: unione di sistemi analitici e transazionali

Abbiamo già detto che l'insieme di dati su cui è possibile effettuare ricerche, inserimenti e modifiche viene detto database. Lo strumento per gestire questi dati, invece, è il Data Base Management System (DBMS). In base alla loro finalità, i DBMS possono essere classificati in due categorie:

- sistemi transazionali: forniscono il supporto alla attività di tipo operativo e gestionale.
- sistemi di analisi: forniscono il supporto alla attività di tipo strategico e decisionale.

I sistemi On Line Transaction Processing (OLTP), o DBMS transazionali, registrano, modificano e mostrano dei record in tempo reale, minimizzando la possibilità di generare errori. Per questo motivo viene fatto larghissimo uso delle transazioni, cioè di operazioni in sequenza viste come un insieme atomico, che vengono eseguite con successo in blocco oppure falliscono in blocco. Un sistema del genere garantisce la coerenza dei dati del sistema, conferendogli le proprietà ACID, e rendendo tutte le operazioni fluide e di rapida esecuzione. Un database OLTP deve essere ottimizzato per sostenere l'attività di un numero molto elevato di utenti che sollecitano la base di dati con operazioni continue di inserimento, ricerca e aggiornamento delle informazioni.
I sistemi On Line Analytical Processing (OLAP), o DBMS per l'analisi, invece, non lavorano sui dati più recenti, ma coprono un arco temporale molto vasto. Vengono aggiornati periodicamente e sono in continua crescita, poiché accumulano dati nel tempo. Forniscono delle sintesi generate come aggregazioni di dati di dettaglio, e consentono operazioni di analisi a supporto delle decisioni strategiche aziendali.

In un mondo in cui l'era digitale sta creando sempre di più nuove sfide, la sicurezza e l'integrità dei dati non è mai stata così importante. La quantità di dati e la loro varietà stanno portando le aziende ad innovarsi velocemente, e a decidere tra l'affidabilità dei database relazionali e la flessibilità dei database non conformi a determinati schemi. Di fronte a questa scelta, molte aziende hanno scelto di implementare più tipi di database, aumentando sia il costo che la complessità delle loro infrastrutture.
Con rigide definizioni di schemi e transazioni ACID, i database relazionali garantiscono la sicurezza e l'integrità dei dati. Ma per schemi più flessibili?
La potenza di MariaDB Server sta proprio nel fatto di garantire la sicurezza dei dati per l'azienda e allo stesso tempo di supportare la flessibilità dello schema e dei dati semi-strutturati. Di fatto MariaDB Platform è un database open source, aziendale, per

l'elaborazione transazionale, analitica o un ibrido dei due su larga scala. Attraverso le colonne dinamiche e le funzioni JSON, MariaDB Server consente alle aziende di utilizzare un unico database per dati strutturati e semi-strutturati, relazionali e JSON, insieme. È possibile creare uno schema flessibile con un modello di dati relazionale, senza sacrificare l'integrità dei dati, le transazioni e l'SQL. Infatti, è possibile combinare, interrogare con l'SQL e modificare con transazioni ACID dati relazionali, semi-strutturati e JSON.

In tutto ciò, però, le applicazioni devono essere responsabili della gestione sia dello schema che dell'integrità dei dati, ma a vantaggio di un supporto più flessibile per entrambi i modelli, e di un utilizzo di query e modelli di dati più semplici.

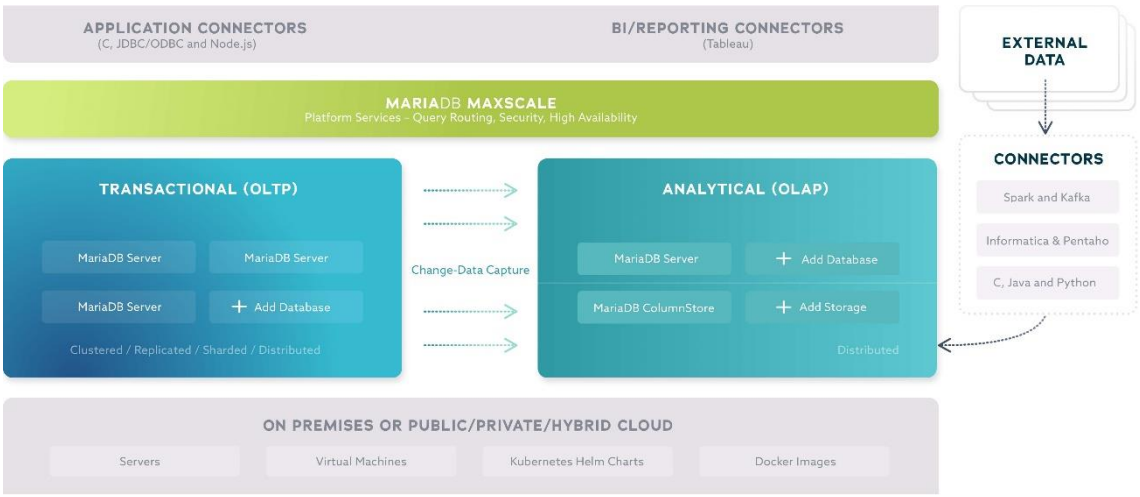

Figura 9 - MariaDB and Transactional e Analytical systems

5. Confronto tra MariaDB e MySQL

MariaDB nasce come un fork di MySQL, e nonostante erediti molte caratteristiche da quest'ultimo, negli anni ci sono stati diversi cambiamenti, tra aggiunte, rimozioni e miglioramento delle prestazioni. In particolar modo in questa sezione vengono presentate le principali differenze tra MariaDB 10.5 e MySQL 8.0, ultime versioni dei rispettivi DBMS.

Nuovi Storage Engine

MariaDB è distribuito con diversi Storage Engine, alcuni dei quali non possono funzionare con MySQL. In aggiunta agli Storage Engine standard InnoDB, MyISAM, Blackhole, CSV, Memory, Archive e Merge, in MariaDB 10.5 sono presenti anche i seguenti Storage Engine:

MyRocks

MyRocks è uno Storage Engine, open source, originariamente sviluppato dal Database Engineering Team di Facebook. Si distingue rispetto agli altri Storage Engine per performance e scalabilità. Può essere usato soprattutto per carichi di lavoro che richiedono una grande compressione (rispetto ad InnoDB, ha una compressione dei dati due volte più efficiente) ed efficienza di I/O[10] (ha un'amplificazione di scrittura 10 volte inferiore rispetto a InnoDB). Utilizza un'architettura Log Structured Merge (LSM), che presenta vantaggi rispetto agli algoritmi B-Tree, per fornire un'acquisizione efficiente dei dati, come il caricamento rapido dei dati in blocco.

Aria

Aria è un nuovo Storage Engine per MySQL e MariaDB, sviluppato con l'obiettivo di creare uno Storage Engine di default che sia transazionale e non-transazionale.
È Stato sviluppato nel 2007 dagli ingegneri di MySQL. Esso inizialmente si chiamava Maria, come la figlia più giovane di Monty Widenius, ma poiché il server di database MariaDB e lo Storage Engine Maria venivano confusi, è stato deciso di cambiargli il nome. Così in un contest chiamato "Rename Maria", tenutosi nel 2010, Monty ha deciso di chiamarlo Aria.

[10] Input/Output.

L’obiettivo di Aria è costituire un'alternativa crash-safe a MyISAM (MyISAM è stato lo Storage Engine predefinito in MySQL fino alla versione 5.5, poi sostituito da InnoDB). Ossia, quando mysqld (core dump, programma che avvia il server MySQL sul sistema) si riavvia dopo un crash, Aria riporta tutte le tabelle allo stato in cui si trovavano all'inizio di un'istruzione o all'inizio del precedente LOCK TABLES.

Al momento la versione di Aria è la 1.5, ma gli ingegneri stanno lavorando ad una versione successiva, la 2.0, che si pone come obiettivo uno Storage Engine pienamente transazionale, che abbia almeno tutte le funzionalità principali di InnoDB.

FederatedX

Questo Storage Engine è un fork di Federated (presente in MySQL), il quale non è più mantenuto da Oracle. Con FederatedX vengono aggiunte nuove funzionalità come le transazioni, il supporto alle partizioni, e una nuova struttura delle classi per permettere agli sviluppatori di scrivere classi per le connessioni ad altri RDBMS senza dover modificare quelle di base di FederatedX. Inoltre, vengono corretti vecchi bug. Esso funziona sia con MariaDB che con MySQL.
È uno Storage Engine che funge da collegamento verso un'altra tabella, che generalmente si trova su un server SQL differente, attraverso la libreria libmysql. Quindi l’utilizzatore può dialogare con RDBMS remoti, interrogando le tabelle e modificandone i dati. Attualmente, siccome FederatedX può utilizzare solo libmysql, riesce a parlare soltanto con MySQL. Il progetto futuro naturalmente è di permettergli di interfacciarsi con altri RDBMS usandoli come fonti di dati.

Normalmente i file di database sono locali, quindi se viene creata una tabella, verrà creato un file locale con il nome di quella tabella. Un handler poi legge, inserisce, elimina e modifica i dati su questo file.
Con lo Storage Engine FederatedX, invece, non ci sono file locali che mantengono i dati di ogni tabella, ma un database esterno che li contiene.

OQGRAPH

OQGRAPH (Open Query Graph), è sviluppato da Open Query, e, nonostante si presenti come uno Storage Engine, ha un’architettura completamente diversa, trattandosi di fatto di una libreria, con interfaccia SQL, che simula i grafi.

Per creare una tabella OQGRAPH, occorre prima creare una tabella di supporto, la quale conterrà i dati effettivi:

```
CREATE TABLE oq_backing (
  origid INT UNSIGNED NOT NULL,
  destid INT UNSIGNED NOT NULL,
  weight DOUBLE NOT NULL,
  PRIMARY KEY (origid, destid),
  KEY (destid)
);
```

Per la tabella OQGRAPH, l'istruzione CREATE deve seguire esattamente il formato qui sotto (qualsiasi differenza porterà un errore):

```
CREATE TABLE oq_graph (
  latch VARCHAR(32) NULL,
  origid BIGINT UNSIGNED NULL,
  destid BIGINT UNSIGNED NULL,
  weight DOUBLE NULL,
  seq BIGINT UNSIGNED NULL,
  linkid BIGINT UNSIGNED NULL,
  KEY (latch, origid, destid) USING HASH,
  KEY (latch, destid, origid) USING HASH
)
ENGINE=OQGRAPH
data_table='oq_backing' origid='origid' destid='destid';
```

I dati devono essere inseriti solo nella tabella di supporto, non nella tabella OQGRAPH. Avendo creato la tabella oq_graph collegata a una tabella di supporto, sarà possibile interrogare direttamente oq_graph.

SphinxSE

SphinxSE di per sé non gestisce i dati ma è un client built-in che permette a MariaDB di comunicare con searchd (il daemon[11] di Sphinx), eseguire ricerche e ottenere i risultati.

[11] Demone, è un programma eseguito in background, cioè senza che sia sotto il controllo diretto dell'utente, che tipicamente fornisce un servizio all'utente stesso.

SphinixSE viene utilizzato come alternativa più veloce e personalizzabile alla ricerca full-text incorporata in MariaDB.

CONNECT

CONNECT è stato introdotto in MariaDB 10.0, e permette di connettersi a dati esterni o remoti di diverso tipo. Per ogni varietà di dato supportato, lo Storage Engine fornisce un Table Type corrispondente. Quindi permette di utilizzare direttamente i dati da diverse fonti, come file nativi, tabelle di altri DBMS e tabelle "virtuali" speciali, senza essere obbligati a caricarli fisicamente nel database.
Alcuni dei Table Type servono per accedere a dei dati su file, altri per connettersi a database remoti tramite il protocollo di MySQL o lo standard ODBC.

Sequence

Il motore Sequence permette la creazione di sequenze di numeri (interi positivi), crescenti o decrescenti, dati un valore di partenza, un valore di arrivo e un incremento. Crea poi automaticamente tabelle virtuali al momento del bisogno. Non è possibile creare esplicitamente una tabella di sequenza. Inoltre, sono tabelle di sola lettura, transizionali e con il supporto XA[12], e non vengono mai scritte su disco.

Per usare una Sequence table, si selezionano i dati da una tabella virtuale che segue questo nome **seq_**FROM**_to_**TO**_step_**STEP.
Ad esempio:
*SELECT * FROM seq_1_to_13_step_3;*
restituisce una sequenza di numeri da 1 a 13 con un incremento di 3.

Spider

Spider è uno Storage Engine con funzionalità di partizionamento dei dati orizzontale[13] e supporto alle transazioni XA. Inoltre, consente di gestire tabelle di diverse istanze di MariaDB come se fossero sulla stessa istanza.
Quando viene creata una tabella con Spider, essa si collega ad un'altra tabella su un server remoto. I dati possono essere inseriti sul server locale, ma vengono immagazzinati sul server remoto. La tabella remota può essere di qualsiasi Storage Engine. Il collegamento,

[12] Le transazioni XA sono progettate per essere transazioni distribuite, dove un transaction manager (l'applicazione) controlla una transazione che coinvolge più risorse, solitamente DBMS.
[13] Il partizionamento orizzontale è un principio di progettazione del database in base al quale le righe di una tabella vengono conservate separatamente su database server o posizioni fisiche differenti.

poi, alla tabella si ottiene concretamente stabilendo la connessione da un server MariaDB locale a un server MariaDB remoto. Il collegamento è condiviso per tutte le tabelle che fanno parte di una stessa transazione.

Mroonga

Mroonga, precedentemente denominato Groonga, è un motore di database, incluso per la prima volta in MariaDB 5.3, basato su colonne, che offre una ricerca full-text per varie lingue, incluse cinese, giapponese e coreano.

Questi non sono i soli Storage Engine presenti in MariaDB. In passato ne erano presenti anche altri, che poi sono stati rimossi oppure disabilitati.

XtraDB

Fino alla versione 10.1, MariaDB utilizzava XtraDB come Storage Engine di default. Dalle successive versioni è stato poi sostituito da InnoDB (Storage Engine di default per MySQL). Questo perché nonostante i miglioramenti apportati a XtraDB rispetto InnoDB, lo sviluppo richiedeva degli sforzi molto elevati. Inoltre, MySQL aveva recuperato il ritardo, implementando quasi tutte le nuove funzionalità di XtraDB.
XtraDB è un fork dello Storage Engine InnoDB sviluppato e mantenuto da Percona. Il suo scopo è quello di correggere i bug esistenti e implementare nuove funzionalità. Eredita da InnoDB la robustezza, l'affidabilità, le transazioni ACID e un'architettura MVCC (MultiVersion Concurrency Control) avanzata. Partendo da queste fondamenta è stato poi progettato per essere più scalabile sull'hardware moderno, includendo una serie di novità utili in ambienti ad alte performance e un sistema di configurazione più dettagliato. In particolare, è progettato per scalare meglio su core multipli, ed usare la memoria in maniera più efficiente. XtraDB è completamente compatibile con InnoDB e, man mano che Oracle rilascia nuove versioni, esse vengono integrate in XtraDB.

PBXT

Questo Storage Engine è incluso in MariaDB, tuttavia dalla versione 5.5, pur essendo ancora incluso, è disabilitato di default, infatti bisogna compilarlo esplicitamente per utilizzarlo. Il motivo di ciò è dovuto al fatto che PBXT non è più mantenuto attivamente. È stato sviluppato da PrimeBase, è transazionale e utilizza il multi-versioning dei record. PBXT è pienamente ACID compliant: ciò significa che può essere utilizzato come alternativa agli altri Storage Engine transazionali di MariaDB (come XtraDB o InnoDB).

Alcune caratteristiche di PBXT sono:

- Supporto MVCC: MVCC sta per Multi-version Concurrency Control, controllo della concorrenza multi-versione. MVCC permette di leggere un database senza creare dei lock.
- Lock a livello di riga: Durante gli aggiornamenti, PBXT usa i lock a livello di riga. Questo lock è usato anche durante le istruzioni SELECT, FOR, UPDATE.
- Rollback[14] e recupero veloci: PBXT usa un metodo specializzato per individuare i dati superflui, eliminando, di fatto, la necessità di cancellarli. Per questo motivo sia il rollback delle transazioni, sia il recupero dopo un arresto anomalo sono estremamente veloci.
- Rilevamento dei deadlock[15]: PBXT individua immediatamente qualsiasi tipo di deadlock.
- Scrittura non replicata: PBXT utilizza uno storage basato sui log e questo rende possibile scrivere i dati transazionali direttamente nel database, senza che debbano passare per il log delle transazioni.
- Integrità referenziale: PBXT supporta la definizione delle chiavi esterne, comprese le DELETE e le UPDATE a cascata.
- BLOB streaming: In combinazione con lo Streaming Engine BLOB, PBXT può inviare un flusso di dati binario direttamente al o dal database.

TokuDB

È uno Storage Engine destinato all'uso in ambienti ad alte prestazioni e ad alta intensità di scrittura, offrendo una maggiore compressione dei dati e performance migliori.

Cassandra

Lo Storage Engine Cassandra è stato sviluppato per connettere un server MariaDB a tabelle create con Apache Cassandra. Essendo Cassandra un DBMS non relazionale, l'obiettivo principale di Cassandra SE (Storage Engine) è l'integrazione dei dati tra i mondi SQL e NoSQL. E da un punto di vista logico permette di far apparire famiglie di colonne di Cassandra come una tabella relazionale in MariaDB, da cui si possono inserire, aggiornare e selezionare dati. È possibile scrivere join su questa tabella, ed è possibile unire i dati archiviati in MariaDB con i dati archiviati in Cassandra.

[14] È un'operazione che permette di riportare la base di dati a una versione o stato precedente.

[15] Situazione in cui due o più processi o azioni si bloccano a vicenda, aspettando che uno esegua una certa azione che serve all'altro e viceversa.

IBMDB2I

Oracle lo ha rimosso in MySQL 5.1.55 ma il codice è rimasto in MariaDB fino alla versione 5.5. IBMDB2I è stato progettato per consentire a MySQL di archiviare i dati su tabelle IBM DB2 (Relational Database Management System di IBM).

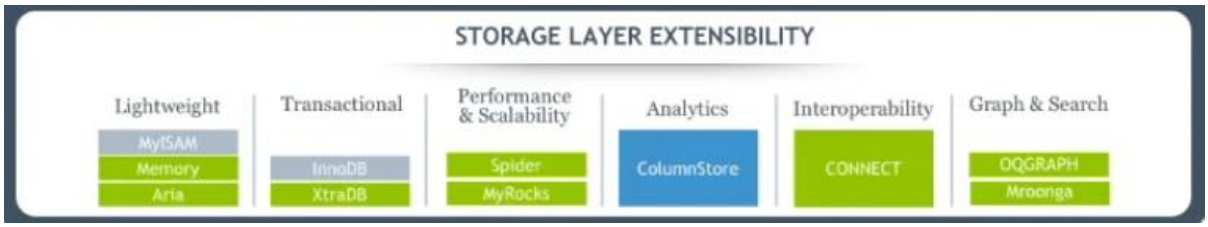

Figura 10 – Storage Layer Extensibility

MariaDB Galera Cluster e thread pool

Inoltre, le differenze più notevoli rispetto MySQL sono che MariaDB include, di default, il motore di archiviazione Aria (di cui è già stato fatto cenno), Galera Cluster e ha un'implementazione del thread pool diversa.

Galera Cluster è un cluster sincrono multi-master[16] per MariaDB. Supporta solo i motori di archiviazione XtraDB e InnoDB. Tra le sue funzionalità emergono:

- Replica sincrona: a differenza della replica asincrona, quella sincrona garantisce che se avvengono modifiche in uno dei nodi del cluster, esse avvengano anche negli altri nodi in modo "sincrono", cioè nello stesso momento.
- Topologia multi-master attivo-attivo: si ha una replica bidirezionale dei dati tra due database che vengono entrambi attivamente aggiornati (mentre la replica attivo-passivo indica la replica unidirezionale da un database master che viene attivamente aggiornato a un database slave che non viene aggiornato tranne che dal processo di replica).
- Lettura e scrittura in tutti i nodi del cluster.
- Controllo automatico dei membri: i nodi che falliscono vengono esclusi dal cluster.
- Unione dei nodi automatica.

[16] La replica multi-master è un metodo di replica dei database, che permette ai dati di essere immagazzinati in un gruppo di computer e aggiornati da qualsiasi membro del gruppo. Il sistema di replica multi-master è responsabile della propagazione delle modifiche ai dati apportate da ciascun membro al resto del gruppo e della risoluzione di eventuali conflitti che potrebbero sorgere tra modifiche simultanee apportate da membri diversi.

- Replica parallela, a livello di riga.
- Connessioni dirette dei client, con look & feel nativo di MariaDB.

L'obiettivo di avere un server software scalabile (come, per esempio, un DBMS) comporta mantenere le migliori performance quando il numero di client aumenta. MySQL tradizionalmente assegnava un thread ad ogni connessione da parte dei client, così con l'aumentare degli utenti concorrenti le performance calavano. Avere molti thread attivi può rappresentare un grosso problema, perché l'aumento del numero di connessioni comporta un cambiamento di contesto sempre più intensivo e un peggior incapsulamento delle cache della CPU. Quindi una soluzione ideale sarebbe quella di mantenere il numero dei thread inferiore alla quantità di client.
Dalla versione 5.5 di MariaDB viene implementato un pool dinamico/adattivo che si occupa da solo della creazione di nuovi thread quando la domanda aumenta, e di ritirarli quando sono inattivi.
I thread pool sono più efficienti in quelle situazioni in cui le query sono relativamente brevi e il carico è legato soprattutto alla CPU. Se il carico di lavoro non fosse sulla CPU, si potrebbe preferire limitare il numero di thread per risparmiare memoria per i buffer del database.

Miglioramento delle prestazioni

MariaDB non primeggia solo grazie all'ampia scelta di motori di database ma anche per l'efficiente ottimizzatore per le query SQL. Infatti, prima di MariaDB 10.0, l'ottimizzatore di MySQL e MariaDB si basava su motori di archiviazione (come, ad esempio, InnoDB) per fornire statistiche circa l'ottimizzazione delle query. Tuttavia, tale approccio aveva delle carenze:

- I motori di archiviazione fornivano statistiche scadenti. Infatti, le statistiche di InnoDB non erano memorizzate su disco, il che significava che al riavvio del server le statistiche dovevano essere ricalcolate, e ciò risultava essere un calcolo inutile oltre al fatto che portava a piani di query incoerenti. Il problema fu poi risolto con l'introduzione delle statistiche persistenti.
- Le statistiche erano fornite tramite lo storage engine di MySQL, il quale poneva molte restrizioni sul tipo di dati forniti. Per esempio, non era possibile ottenere dati sulla distribuzione del valore in una colonna non indicizzata.
- C'era poco controllo sulle statistiche. Per esempio, non c'era modo di fissare i valori statistici correnti o fornire alcuni valori singolarmente.

Dalla versione 10.0, per l'ottimizzatore delle richieste, MariaDB punta a tabelle di statistica indipendenti dal motore di archiviazione, che vengono salvate come tabelle tradizionali nel database (*mysql.table_stats* , *mysql.column_stats* e *mysql.index_stats*) e consentono la registrazione di più valori per individuare un piano ideale per l'esecuzione di dichiarazioni SQL. L'utilizzo o l'aggiornamento dei dati di queste tabelle è controllato dalla variabile *use_stat_tables*.
Da MariaDB 5.3/5.5 ci sono stati, progressivamente, molti miglioramenti di ottimizzazione anche riguardo le subquery[17].

[17] Una subquery (o interrogazione nidificata) è una query presente all'interno di un'altra interrogazione: la query interna, cioè la subquery, passa i risultati alla query esterna.

6. Benchmark

In questa sezione vengono trattati i benchmark eseguiti con SysBench sui seguenti DBMS:

- MariaDB 10.5.5
- MySQL 8.0.21

In particolare, sono state valutate le prestazioni di MariaDB rispetto MySQL per quanto riguarda lo storage engine di default InnoDB, e le prestazioni del motore Aria rispetto MyISAM in MariaDB, analizzando, in termini di transazioni per secondo eseguite, i benefici introdotti dall'ultima versione di MariaDB e le prestazioni di Aria e MyISAM.
I benchmark in questione sono stati eseguiti su una macchina a 4 core, con 6 GB di RAM, e con i dati memorizzati su HDD.

SysBench è uno strumento di benchmark multi-thread con script basati su LuaJIT (Lua Just-In-Time Compiler). È stato scritto originariamente da Peter Zaitsev, nel 2004. Il suo scopo era fornire uno strumento per eseguire benchmark sintetici di MySQL e dell'hardware su cui girava. È stato progettato per eseguire test di CPU, memoria e I/O, e aveva anche un'opzione per eseguire il carico di lavoro OLTP su un database MySQL. A differenza delle versioni inziali, con script codificati, ora si ha la possibilità di personalizzare i benchmark utilizzando il linguaggio LUA. Tali script gestiscono l'input dai parametri della riga di comando, definiscono le modalità che il benchmark dovrebbe utilizzare, preparano i dati, e definiscono come verrà eseguito il benchmark. Di fatto, con SysBench, sono disponibili statistiche dettagliate su velocità e latenza, inclusi gli istogrammi, ed è in grado di generare e tenere traccia di centinaia di milioni di eventi al secondo, con un basso overhead[18] anche con migliaia di thread simultanei.
SysBench, quindi, è uno strumento molto utile per eseguire una grande varietà di benchmark, che coprono la maggior parte dei casi. Inoltre, con l'introduzione degli script LUA, è diventato estremamente configurabile, consentendo di preparare benchmark personalizzati. È molto utilizzato anche per confrontare le prestazioni di hardware diversi, e, per esempio, per la simulazione e la valutazione della fattibilità di determinati progetti.

[18] Definisce le risorse accessorie, ma necessarie, in termini di memoria, tempo di esecuzione o di banda occupata, per ottenere un determinato risultato.

I benchmark qui trattati utilizzano "sysbench-1.0.20" con un database di 10 tabelle da 1 milione di record ciascuna e 2.5 GB di dati totali. La scelta della quantità di dati da testare deriva dal fatto che, considerando tutti i processi attivi e la quantità di RAM disponibile sul PC, non si vuole creare un "collo di bottiglia" quando vengono caricati i dati in memoria. Inoltre, l'analisi è stata concentrata su benchmark OLTP di sola lettura (read-only, RO). Infine, i test sono stati eseguiti con 1, 2, 4, 8, 16, 32, 64 e 128 thread, per simulare un certo livello di concorrenza, e con un run time (cioè in tempo di esecuzione di ogni simulazione) di 300 secondi. Nei grafici è stata riportata la media di transazioni per secondo eseguite in 300 secondi con un certo numero di thread.
Ogni transazione si compone di diversi tipi di query. Le query di sola lettura utilizzate nei benchmark sono:
SELECT c FROM sbtest%u WHERE id=?
SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ?
SELECT SUM(k) FROM sbtest%u WHERE id BETWEEN ? AND ?
SELECT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c
SELECT DISTINCT c FROM sbtest%u WHERE id BETWEEN ? AND ? ORDER BY c
"sbtest" è il database standard su cui SysBench effettua i benchmark. Si possono anche disabilitare tutte le query rivolte alla ricerca di un range di elementi, per concentrarsi sulla ricerca di un singolo record (la prima query).

Come storage engine, di default, senza quindi specificare ulteriori parametri, viene utilizzato InnoDB.

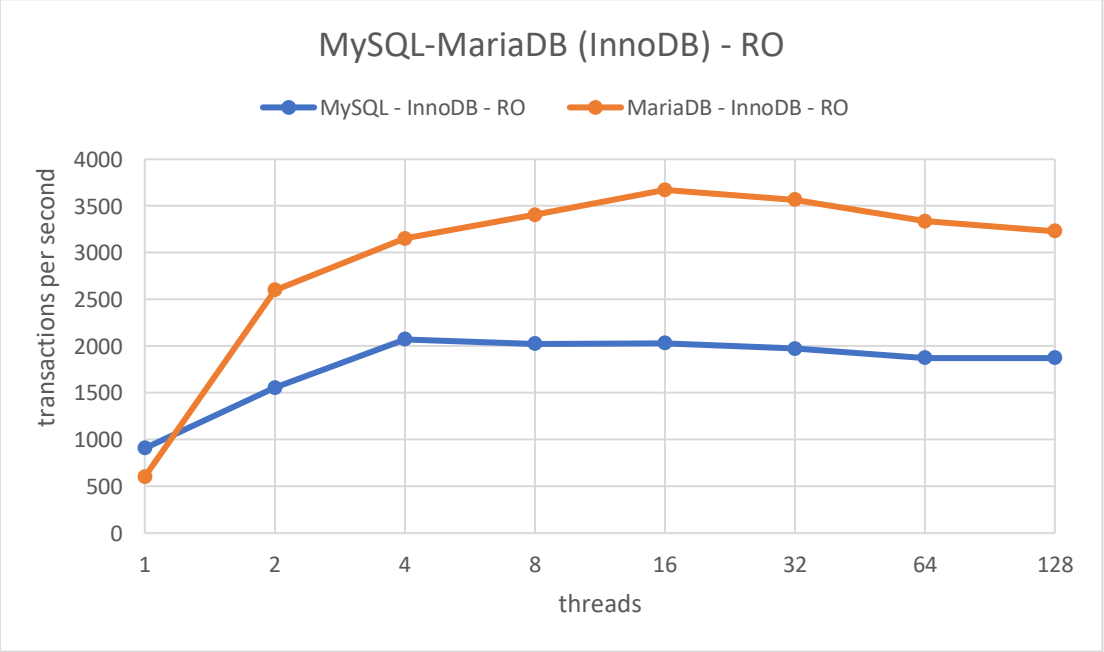

Figura 11 - MySQL-MariaDB (InnoDB) - RO

Dal grafico si nota come le prestazioni di MariaDB, seppur inferiori all'inizio, crescono molto più rapidamente, con l'aumentare dei thread, rispetto a MySQL. Con l'aumentare della concorrenza però le transazioni per secondo tendono leggermente a diminuire per entrambi i DBMS.
Come già descritto in precedenza, le prestazioni di MariaDB migliorano significativamente rispetto MySQL con l'aumentare dei thread, per una gestione migliore del thread pool. Infatti, mentre MySQL implementa un thread pool statico, MariaDB implementa un pool dinamico che si adatta alle diverse situazioni, creando nuovi thread quando c'è più bisogno e ritirando quelli non più utilizzati.

In questo secondo caso, a differenza del precedente, tra i parametri di SysBench è stato anche specificato lo storage engine da utilizzare (MyISAM e Aria).

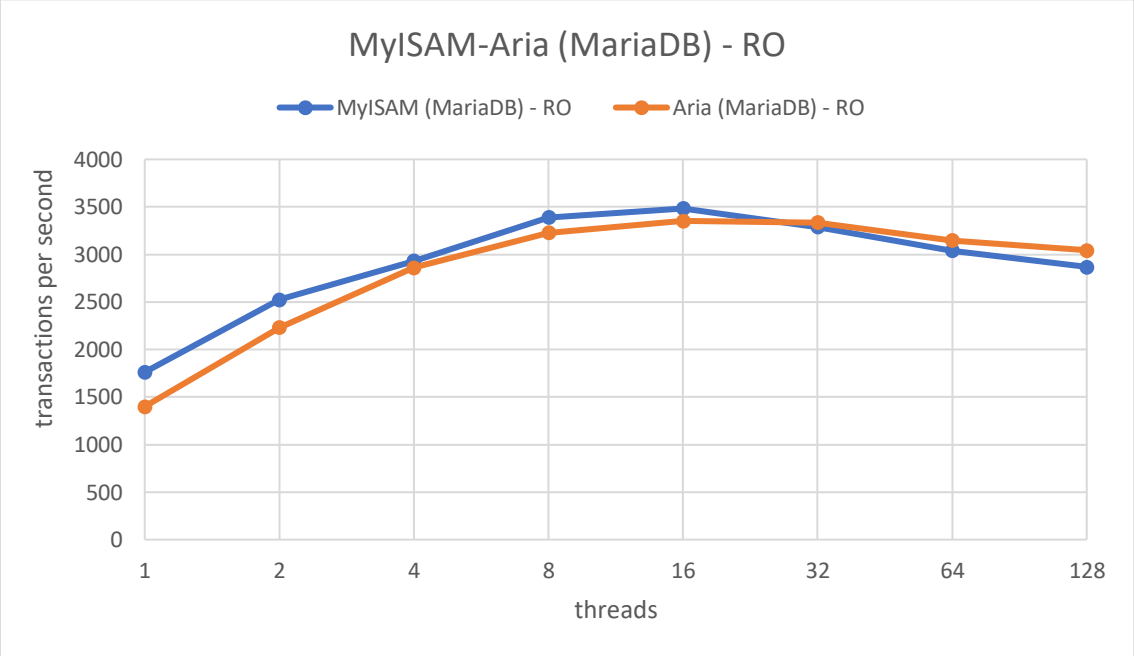

Figura 12 - MyISAM-Aria (MariaDB) -RO

Le due curve salgono molto rapidamente, ma poi si appiattiscono, fino a calare leggermente sul finale. Con l'aumentare della concorrenza, quindi, le prestazioni di entrambi tendono a calare. Si nota come le prestazioni degli storage engine Aria e MyISAM sono pressoché identiche, con una quasi sovrapposizione delle curve. Infatti, Aria, eredita buona parte delle caratteristiche di MyISAM, ma vuole rappresentare un'evoluzione fornendo un'alternativa crash-safe e, in futuro, uno storage engine con supporto transazionale.

LUA

Lua è un linguaggio di programmazione multi-paradigma e multipiattaforma, con l'interprete scritto in linguaggio ANSI C. Ha una API C per far sì che possa essere incorporato nelle applicazioni.

È stato creato nel 1993 da Roberto Ierusalimschy, Luiz Henrique de Figueiredo e Waldemar Celes, informatici brasiliani, membri del Computer Graphics Technology Group (Tecgraf) presso la Pontificia Università Cattolica di Rio de Janeiro, in Brasile. Dal 1977 al 1992, il Brasile ha adottato una politica di forti barriere commerciali e non era semplice per le aziende comprare pacchetti software all'estero. Ciò ha portato la Tecgraf a implementare da zero gli strumenti di base di cui aveva bisogno. Hanno, quindi, sviluppato due linguaggi per la gestione dei dati, SOL (Simple Object Language) e DEL (Data-Entry Language), per delle applicazioni presso la multinazionale Petrobras, attiva nel settore petrolifero. Ma per mancanza di strutture di controllo e scarse capacità di personalizzazione, e per unificare i due linguaggi, rimanendo abbastanza semplice da essere alla portata anche di tecnici senza grandi conoscenze di programmazione, nacque Lua.

È un linguaggio che permette diversi paradigmi di programmazione, come quello funzionale, ad oggetti ed orientato alla manipolazione dei dati. È, inoltre, un linguaggio di programmazione potente, dinamico e leggero, e batte qualsiasi linguaggio interpretato nei benchmark. Ha, infine, una tipizzazione dinamica, è facile da integrare con librerie esterne, ed è open source.

7. Conclusioni

Dietro la storia di MariaDB c'è quella di MySQL. Entrambi i sistemi si fondano sulla medesima struttura di database, ed entrambi si rifanno al modello relazionale. Hanno una definizione di dati e tabelle compatibile tra loro, e vengono impiegati protocolli, strutture e interfacce di programmazione identici. Inoltre, tutti i connettori MySQL sono impiegabili senza modifiche anche con MariaDB per collegare applicazioni e strumenti mediante interfacce di database standard. Per molto tempo, di fatto, i due sistemi hanno proceduto in maniera sincrona nello sviluppo. Ma col tempo sono iniziate a venire fuori le differenze, e la profonda variabilità di utilizzo di un sistema molto flessibile come MariaDB. Il fatto che MariaDB si sia affermata come un'alternativa molto valida a MySQL è anche dimostrato dal fatto che, dalla fine del 2012, MariaDB ha sostituito MySQL come installazione standard su diverse distribuzioni Linux. Inoltre, l'enorme supporto della community offre, agli sviluppatori, la possibilità di portare numerose novità sull'intera piattaforma. In futuro lo sviluppo dei due progetti software divergerà sempre più, con funzionalità esclusive sempre più numerose. Ma, per ora, MariaDB si conferma già di essere una vera e solida alternativa a MySQL.

Siti internet

- https://mariadb.org, MariaDB Foundation
- https://mariadb.com, MariaDB
- https://www.mysql.com, MySQL
- https://www.lua.org, Lua
- https://launchpad.net/sysbench, SysBench
- Articolo: "*Data Never Sleeps 8.0*", https://www.domo.com
- Articolo: "*We Are Social: Global Digital 2018*", https://wearesocial.com
- "*What's new in MariaDB Platform X5*", 2020, webinar, https://mariadb.com
- "*MariaDB vs. MySQL – what's the difference, and why does it matter*", 2020, webinar, https://mariadb.com

Bibliografia

- DOMENICO BENEVENTANO, SONIA BERGAMASCHI, FRANCESCO GUERRA, MAURIZIO VINCINI, "*Progetto di Basi di Dati Relazionali. Lezioni ed esercizi*", Pitagora Editrice Bologna, 2007
- "*MariaDB and JSON: Flexible Data Modeling*", agosto 2019, whitepaper, https://mariadb.com