POLITECNICO DI TORINO

Corso di Laurea Magistrale in

INGEGNERIA GESTIONALE

Tesi di Laurea Magistrale

Modelli di Anomaly Detection in Real Time su piani di esecuzione e metriche di performance di un Database

Relatori

Prof.ssa Tania CERQUITELLI

Dott. Antonio GRECO

Candidato

Sonia REGIS

Anno accademico 2021-2022

Ringraziamenti

Come conclusione di questo percorso, che è durato ben cinque anni della mia vita (oppure solo, dipende un po' dai punti di vista), non volevo scrivere qualcosa di banale. Stranamente da quello che dico di solito, sono convinta di aver ottenuto questo risultato principalmente con le mie forze e la mia tenacia, ma soprattutto la mia organizzazione. Non credo di aver compiuto un'impresa incredibile, ma solo di aver raggiunto quello che mi ero prefissata. Nonostante ciò, bisogna però ammettere che non sono stati anni semplici per vari motivi che, bene o male, avete avuto modo di conoscere. Ora però, alla fine dei conti, penso di non aver mai visto il Poli come una montagna impossibile da scalare, anche se decisamente impegnativa.

Passiamo ora ai ringraziamenti veri e propri e ai relativi pianti.

Alla mia famiglia, che ha sborsato la grana necessaria ed è sempre stata al mio fianco, ognuno a modo suo.

A Papy, che le volte che mi ha detto un "Brava" possono contarsi sulle dita di una mano, anche se so che è sempre stato orgoglioso dei suoi bimbi.

A Mum che, con la sua gioia contagiosa mi ha sempre regalato attimi di spensieratezza.

A Simo che, con i suoi discorsi motivazionali, già alla fine dei primi appelli del primo anno valutava se fosse opportuno continuare a studiare ingegneria; si vedeva già dall'inizio la tua forte fiducia nelle mie capacità.

A Barbara, che considero ormai parte della famiglia, come fosse una sorella maggiore che, purtroppo, non ho mai avuto.

E ora passiamo agli amici, le persone che hanno superato l'accurata fase di scrematura iniziale e che mi sopportano davvero. Sopportano i miei continui sbalzi d'umore, le infinite lamentele e i momenti di sfogo. Ma, se siete ancora qui, vorrà dire che avrò anche dei pregi...

A Fede, il coinquilino perfetto che mi ha accompagnata in questo ultimo anno, con cui ho condiviso molte belle cose; ci siamo trovati su diversi aspetti, dalla convivenza alla palestra. Siamo sempre presenti l'uno per l'altra, soprattutto per le nostre abituali chiacchierate notturne.

A Dani, uno dei pochi amici rimasti a Imperia; le estati trascorse al mare senza te, Edy e Federico non sarebbero state lo stesso. Siamo sempre andati d'accordo, nonostante non siano mai mancate discussioni, perché facciamo a gara a chi sia più testardo. E, comunque, le feste che organizziamo noi sono incredibili.

Ad Ali, compagna di mille avventure ed altrettante disgrazie e, senza dubbio, la più ritardataria fra tutte. Penso tu sia la persona che mi conosce meglio e non sai quanto sia contenta che il nostro rapporto sia diventato sempre più profondo.

A Simona, una persona speciale con cui ho avuto la fortuna di aver condiviso la fine di questo percorso e l'inizio di un altro ancora più importante. Chi mi conosce sa quello che rappresenti per me e quanto sia fondamentale la tua presenza nella quotidianità. Sei un raggio di sole nelle mie giornate.

Ai compagni del Poli che ho incontrato, e con cui sono riuscita incredibilmente a rimanere in buoni rapporti, che sono stati presenti nella vita di tutti i giorni; senza di voi mi sarei addormentata decisamente molte più volte a lezione.

Grazie a tutti voi per starmi sempre vicina, vi assicuro che è la miglior cosa che possiate fare.

Indice

Introduzione

Al giorno d'oggi la maggior parte delle aziende è capace, attraverso sistemi informatici, a collezionare e manipolare una grande quantità di dati.

Questo elaborato mira, a seguito dei sempre più complessi processi di decision making, all'identificazione di modelli di Anomaly Detection in Real Time nell'ambito della gestione dei Database, pratica ancora non troppo diffusa in ambito aziendale. La corretta gestione delle anomalie e dei disservizi del cliente è di fondamentale importanza in questo tipo di settore e la possibilità di intervenire in maniera molto rapida potrebbe determinare un vantaggio competitivo rilevante rispetto ai competitors, influenzato anche dalla capacità di offrire un servizio più efficiente.

Un outlier rappresenta un punto che si discosta in maniera considerevole dal resto dei punti appartenenti ad un dataset, di conseguenza con Anomaly Detection si intende la capacità di individuare questi valori anomali.

Attualmente la Real Time Analytics viene raffigurata come una delle principali fonti di vantaggio competitivo a livello aziendale, in quanto incide in maniera diretta sul modello di business di un'azienda. A seguito di un utilizzo di tecnologie sempre più sofisticate, è aumentata la possibilità di sfruttare le capacità di manipolazione dei Big Data, al fine di migliorare le prestazioni aziendali.

In passato, l'elaborazione dei dati era molto più lenta rispetto ad oggi e indirizzava le aziende verso previsioni poco attendibili.

Con l'espressione Real Time Analytics si intendono degli insight in tempo reale [1],

ovvero delle intuizioni caratterizzate dalla velocità con cui le informazioni vengono fornite una volta raccolti i dati, che permettono di elaborare strategie di business sempre più complesse.

Attraverso un'osservazioni dei dati storici in concomitanza con quelli in tempo reale, è possibile aumentare la consapevolezza di ciò che è accaduto in passato per poter reagire in maniera più rapida e puntuale su quello che accade ora.

Un'acquisizione immediata dei dati permette alle aziende di agire su di essi e prevenire l'accadimento di determinate problematiche prima che si possano verificare o prima che il cliente possa accorgersene.

Tra i principali vantaggi [2] della Real Time Analytics troviamo:

- **Identificazione errori**, che determina una successiva risposta altrettanto repentina, in modo da aumentare l'efficienza operativa dell'azienda.
- **Miglior tempo di reazione**, in quanto si accorge più velocemente di cambiamenti che, talvolta, possono essere improvvisi.
- **Aumento dei profitti**, dovuto ad una maggiore cura nei confronti dei clienti (customer care in Real Time), che intercetta le loro esigenze, le trasforma in richieste tangibili e le implementa con più rapidità.
- **Risparmio dei costi**, condizionato da un minor impegno, in termini di tempo, delle risorse umane dedicate a queste attività.

Possono, inoltre, essere individuate tre macro categorie [3], che riguardano i più importanti ambiti di progettualità che possono incentivare le aziende a investire tempo e denaro in questo tipo di attività:

- **Monitoraggio e Alerting**, ovvero la capacità di seguire l'evoluzione dei dati in tempo reale, con la possibilità di implementare eventi di alerting personalizzato

al verificarsi di particolari situazioni, caratterizzate da pericolosità più o meno elevata.

- **Automated Decision Making**, progettualità che prevede l'automatizzazione di processi decisionali a seguito del monitoraggio e la raccolta dei dati in streaming.

- **Nuovi prodotti e servizi**, sfruttando al meglio tutte le potenzialità che possono essere estratte da questa tipologia di informazioni, con l'eventualità di ampliare le funzionalità e la gamma di prodotti già esistenti.

La trattazione copre l'intero processo di vita dei dati: Data Collection, Ingestion, Extraction, Enrichment and Visualization.

Lo studio si soffermerà sia su aspetti qualitativi che su aspetti quantitativi, alternando processi ingegneristici a ragionamenti manageriali, con l'ambizione di rendere gli argomenti più semplici e comprensibili a qualsiasi tipo di pubblico.

Nello specifico, sarà necessario fare un approfondimento teorico sull'Oracle Database, il contesto principale in cui si svilupperà questa trattazione, e la relativa architettura. Successivamente, grazie all'utilizzo di alcune viste di sistema fornite da Oracle, sono state sviluppate le tematiche dell'Anomaly Detection in tempo reale sui piani di esecuzione e dell'analisi delle metriche di performance di un Database.

Si precisa, infine, che tutte le analisi che seguiranno sono state effettuate su basi di dati di produzione di società esistenti.

Ambiente di sviluppo Il progetto è stato sviluppato all'interno del gruppo Mediamente Consulting, un'azienda di consulenza specializzata sulle tematiche e le tecnologie di Business Analytics avanzate. Fornisce ai clienti un'infrastruttura tecnologica che gestisce e analizza dati complessi, al fine di procurare alle aziende informazioni utili per poter prendere decisioni strategiche di business.

Il lavoro svolto ha avuto inizio da un framework di partenza, creato nel corso degli anni da Mediamente Consulting, e si è posto come fine ultimo, grazie all'utilizzo dei risultati ottenuti, quello di consolidare la reputazione dell'azienda ed aumentare il vantaggio competitivo rispetto ai competitors.

Capitolo 1

ORACLE DATABASE

Prima di introdurre l'argomento principale della tesi in maniera appropriata, è bene soffermarsi su alcuni elementi che vanno a descrivere meglio il contesto in cui si svilupperà questa trattazione. Verranno quindi approfondite, in questo capitolo, l'architettura e le funzionalità più significative del Database Oracle [4], uno dei RDBMS (Relational Database Management System) più utilizzati.

1.1 Introduzione all'istanza del Database Oracle

Un server Oracle [5] è rappresentato da due macro aree:

- **Database**, ovvero i file fisici in cui sono memorizzati i dati.
- **Istanza**, l'insieme delle aree di memoria e i processi in background che servono per la gestione dei file del Database.

In un Database a istanza singola esiste una relazione uno ad uno tra l'istanza e il Database. Un'istanza può accedere ad uno ed un solo Database. Con Oracle Real Application Cluster (RAC), esistono più istanze per un Database su diversi server. Ogni istanza viene eseguita su un server di Database separato, denominato nodo

nel cluster. I nodi, come si può vedere dalla seguente immagine (Figura 1.1), sono collegati tra loro e accedono a dischi virtuali condivisi.

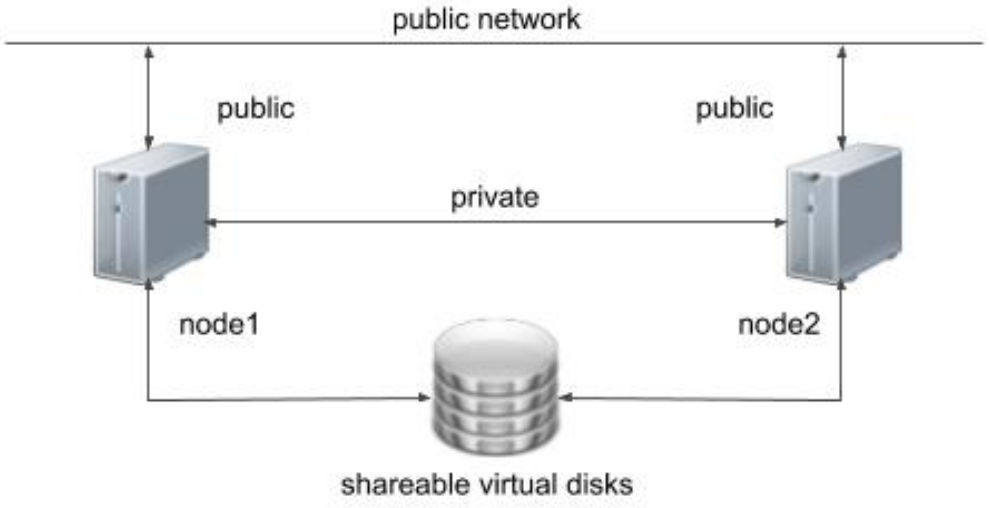

Figura 1.1: Oracle RAC

A livello fisico, l'ultima versione di Oracle ha introdotto un'architettura multi-tenant, ovvero una configurazione con più istanze; in una struttura tale, il relativo strumento di amministrazione è progettato per partizionare virtualmente e dinamicamente i suoi dati, in modo che ogni client lavori con un'istanza virtuale personalizzata (ambiente dedicato).

Nello specifico, ogni singolo Database Container (*CDB$ROOT*), ovvero un insieme di file su disco creati dall'istruzione *CREATE DATABASE*, può avere uno o più Pluggable Database (PDB) creati dall'utente [6]. Un PDB contiene il proprio set di file di dati all'interno del set complessivo di file di dati che appartiene al CDB. L'istanza del Database gestisce i dati associati al CDB e ai relativi PDB. Ogni CDB in esecuzione è associato ad almeno un'istanza del Database Oracle. Poiché un'istanza esiste in memoria e un Database è un insieme di file su disco, un'istanza può esistere senza un Database e un Database può esistere senza un'istanza.

Questa nuova architettura viene introdotta per facilitare la gestione dei Database

Oracle, grazie ai quali possiamo consolidare più Database in un unico CDB. Nell'architettura multi-tenant, idealmente utilizziamo il *PDB$SEED* per creare qualsiasi nuovo Database collegabile all'interno *CDB$ROOT*. Il *PDB$SEED* funge da modello per la creazione di nuovi PDB e non siamo autorizzati a modificarne la configurazione in quanto, per impostazione predefinita, si apre in modalità di sola lettura.

Si può vedere di seguito uno schema generale della struttura appena discussa (Figura 1.2).

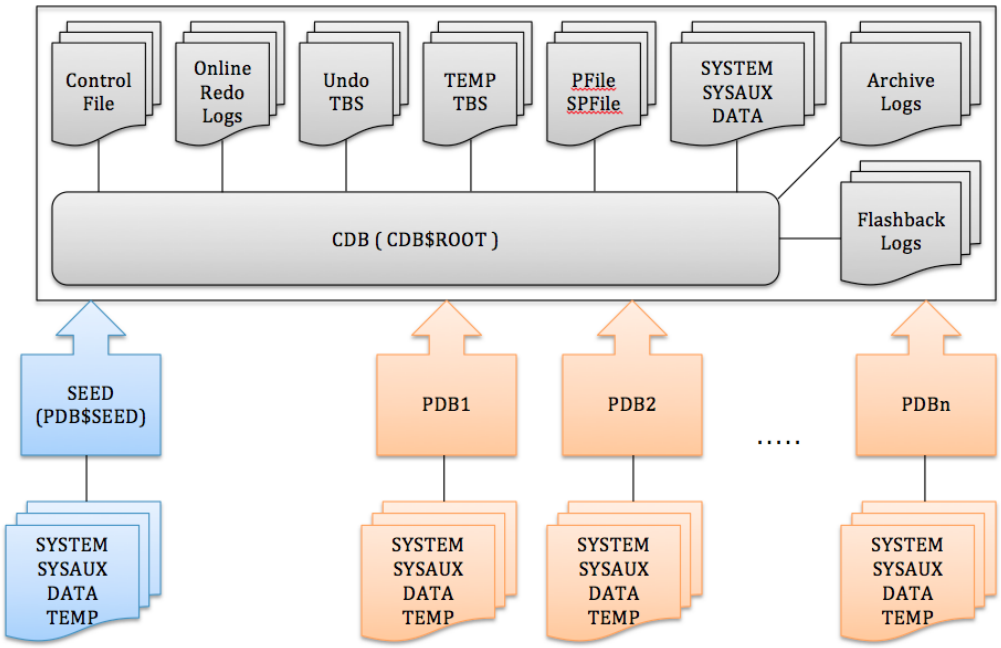

Figura 1.2: Container e Pluggable Databases

1.2 Strutture di memorizzazione logica e fisica

Oracle Database è caratterizzato da strutture logiche, che permettono di immagazzinare e gestire i dati (tabelle, indici, ecc.) e da strutture fisiche, le quali contengono le strutture logiche. Le prime vengono progettate allo stesso modo per tutti gli

hardware e i diversi sistemi operativi; ciò testimonia il fatto che le due parti siano indipendenti tra loro.

La figura seguente (Figura 1.3) è un diagramma entità-relazione per l'archiviazione fisica e logica. La notazione a zampa di gallina rappresenta una relazione uno a molti.

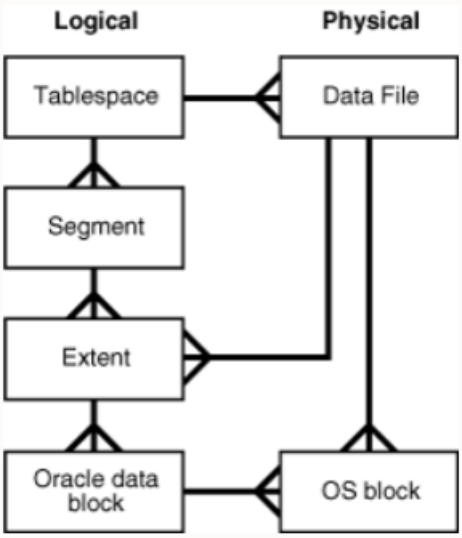

Figura 1.3: Archiviazione logica e fisica

1.2.1 Archiviazione logica

Le strutture logiche di memorizzazione, sono organizzate in maniera gerarchica. Seguendo un'analisi di tipo top down, al vertice si trovano i tablespace, metodo di suddivisione dei dati per una migliore amministrazione del Database; di default viene creato un tablespace di sistema, denominato *SYSTEM*. Ogni tablespace ha al suo interno almeno un segmento che, a sua volta, contiene almeno un blocco. Un blocco è un'unità atomica, ovvero la più piccola porzione indivisibile di memorizzazione dei dati; la sua dimensione viene scelta al momento della creazione del Database.

1.2.2 Archiviazione fisica

Le strutture fisiche di memorizzazione prevedono l'archiviazione di file di dati su disco e si dividono in tre parti principali: Data File, Redo Log File e Control File. I primi contengono tutti i dati, i secondi tengono traccia di tutte le modifiche apportate ai dati e gli ultimi contengono il nome, la data e l'ora di creazione e il percorso completo di ciascun Data File e Redo Log File.

Come si può vedere nella prossima immagine (Figura 1.4), sono presenti anche i Parameter File, che contengono una lista di parametri inizializzati e i relativi valori, e i Password File, i quali memorizzano le chiavi di accesso degli utenti.

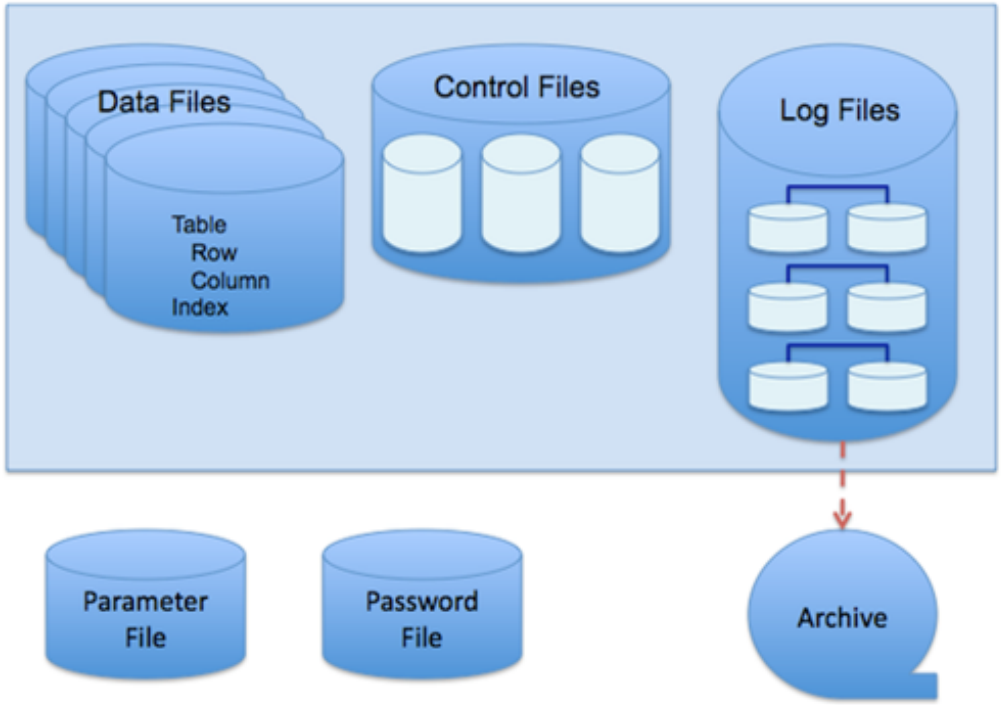

Figura 1.4: Memorizzazione fisica

È bene entrare più nel dettaglio su come Oracle scriva nei Redo Log File in maniera circolare: quando un file è pieno viene scritto quello successivo, assicurandosi però di memorizzare le informazioni negli Archive Log prima che vengano sovrascritte.

1.3 Struttura dell'istanza del Database

L'istanza è una raccolta di processi in esecuzione sul sistema operativo e la relativa memoria che interagisce con l'archiviazione dei dati. Costituisce l'interfaccia tra l'utente e il Database. I processi in grado di comunicare con il client e accedere al Database sono forniti dall'istanza. Questi processi vengono eseguiti in background e non sono sufficienti per mantenere il principio ACID [7], termine che contiene le iniziali delle proprietà fondamentali delle transazioni all'interno di un Database.

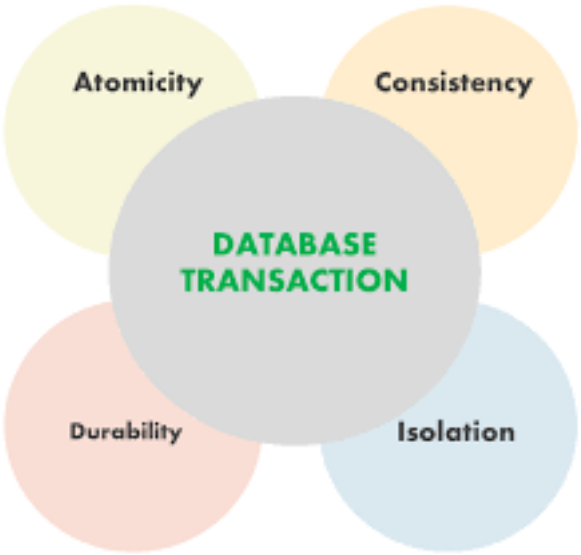

Figura 1.5: Proprietà delle transazioni di un Database

- **Atomicity**, gli effetti devono essere, al termine della stessa, totalmente resi visibili oppure non ne deve essere mostrato nessuno, cioè deve essere eseguita in modo atomico. Questa proprietà viene spesso garantita dal DBMS attraverso le operazioni di *UNDO* per annullare una transazione e di *REDO* per ripetere una transazione. Se questa proprietà viene rispettata il Database non rimane mai in uno stato intermedio inconsistente.

- **Consistency**, garantisce che al termine dell'esecuzione di una transazione i vincoli di integrità siano soddisfatti. Infatti, se questa proprietà viene rispettata, il Database si trova in uno stato coerente sia prima che dopo la transazione.

- **Isolation**, offre certezza sull'indipendenza tra le transazioni, cioè l'eventuale fallimento di una o più transazioni non deve minimamente interferire con altre in esecuzione. Quindi, affinché l'isolamento sia possibile, ogni transazione deve sempre avere accesso a una base di dati consistente.

- **Durability**, fornisce sicurezza riguardo alla permanenza nel sistema dei risultati di una transazione completata con successo, ovvero che non possano più essere persi. Ovviamente c'è un piccolo intervallo temporale tra il momento in cui la base di dati si impegna a scrivere le modifiche e la scrittura vera e propria; questo lasso di tempo è un vero e proprio punto debole e, di conseguenza, risulta sempre necessario garantire, per esempio con un log, che non si verifichino perdite di dati dovuti a malfunzionamenti.

Per assicurare queste proprietà quasi tutti i DBMS implementano un sottosistema di ripristino (Recovery), che permette il mantenimento delle informazioni anche a fronte di guasti ai dispositivi di memorizzazione, per esempio attraverso l'uso di back-up su supporti diversi.

Più specificamente, un'istanza è composta da tre parti:

- **System Global Area** (SGA)

- **Program Global Area** (PGA)

- **Processi in background**

1.3.1 SGA e PGA

Le aree di memoria vengono utilizzate per contenere i dati, le informazioni sul dizionario dei dati, i comandi *SQL*, il codice *PL/SQL*, ecc. Le due principali sono la SGA e la PGA, create dal Database Oracle quando un processo server viene avviato.

Una PGA è un'area di memoria che contiene dati e informazioni di controllo per un processo server. L'accesso alla PGA è esclusivo per quel processo; esiste una PGA per ogni processo server. Inoltre, anche i processi di background allocano la loro PGA.

La SGA contiene dati ed informazioni per il controllo di un'istanza Oracle, si occupa della cache, dei dati bufferizzati, dei comandi *SQL* e delle informazioni sull'utente. La SGA, a differenza della PGA, è una regione di memoria condivisa, formata da Database Buffer Cache, Redo Log Buffer, Shared Pool, Large Pool e Java Pool.

Nel grafico seguente (Figura 1.6) sono riportati i componenti principali di un'istanza del Database Oracle.

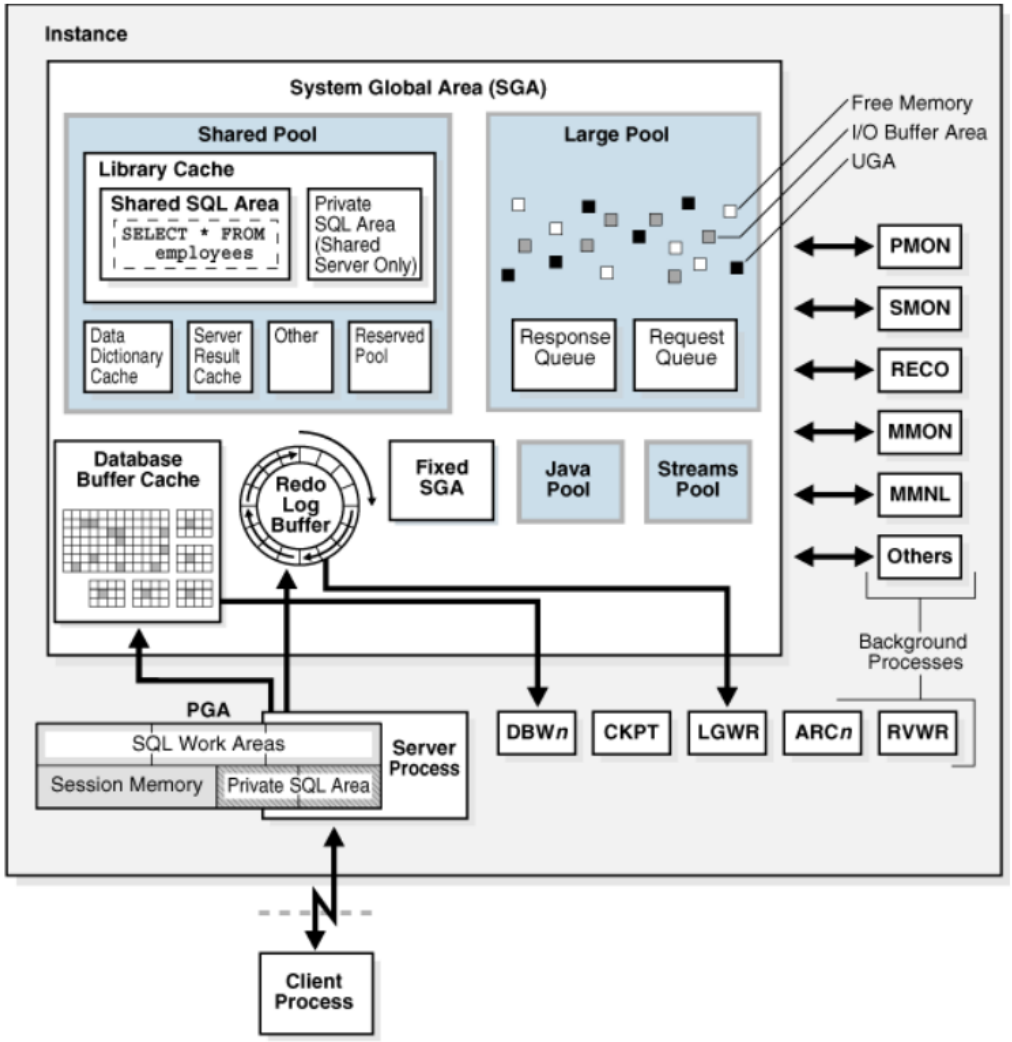

Figura 1.6: I processi Oracle e la SGA

Il Database Buffer Cache contiene i blocchi di dati. La Shared Pool è formata da Library Cache, Data Dictionary Cache, Server Result Cache, Reserved Pool e Enqueues. La prima include Shared e Private SQL Area, al cui interno troviamo le query eseguite in precedenza e i cursori, mentre il dimensionamento della seconda è di fondamentale importanza in quanto, nel caso sia troppo piccola, verranno generate molte query ricorsive (Input/Output eccessivo).

1.3.2 Processi in background

I processi in background vengono eseguiti per impostazione predefinita, in un'istanza di Database di lettura/scrittura, avviata con un file di parametri di inizializzazione configurati di default.

Questa sezione descrive alcuni dei processi obbligatori più rilevanti:

- **PMON**, responsabile del monitoraggio di altri processi, ne rileva la loro interruzione in maniera anomala e ha la funzione di eseguire il loro ripristino.
- **SMON**, responsabile di una serie di compiti di pulizia a livello di sistema, quali il ripristino dell'istanza all'avvio della stessa, il recupero di transazioni terminate che sono state ignorate e la pulizia di segmenti temporanei inutilizzati.
- **DBW**, scrive il contenuto dei buffer del Database nei file di dati se un processo server non riesce a trovare un buffer riutilizzabile o, in maniera periodica, per far avanzare il checkpoint (posizione da cui inizia il ripristino dell'istanza).
- **CKPT**, meccanismo che aggiorna il Control File e le intestazioni del file di dati con le informazioni sul checkpoint e segnala al DBW di scrivere i blocchi su disco (Figura 1.7).

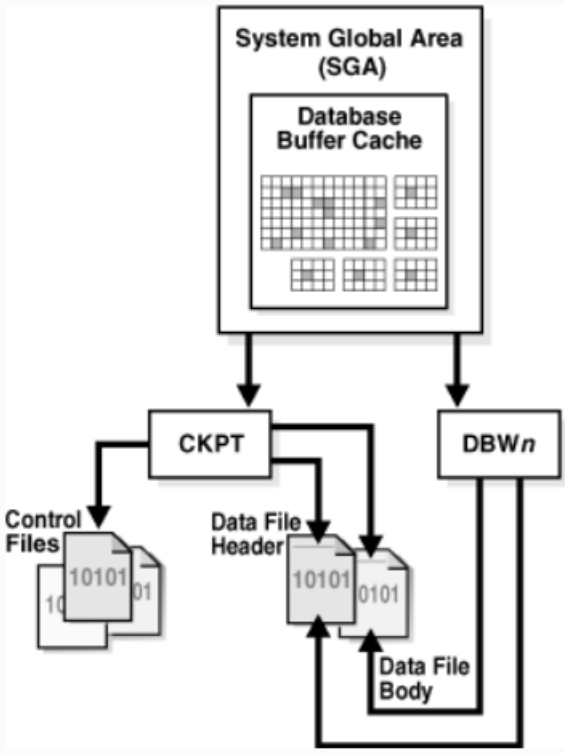

Figura 1.7: Processo del checkpoint

1.4 Piani di esecuzione

1.4.1 Hard e Soft Parsing

La compilazione di un'istruzione SQL è composta da due fasi: la fase di analisi e la fase di esecuzione. Oracle si accerta che l'istruzione sia corretta (sia dal punto di vista sintattico, sia che gli oggetti coinvolti esistano e siano accessibili) e verifica se il piano di esecuzione ad essa associata esista già nella cache della libreria. Se questo è presente, ci troviamo nel caso di Soft Parsing, in quanto non vi è la necessità di caricare il codice *SQL* nella Shared Pool; in caso contrario (Hard Parsing), al fine di facilitare una seconda esecuzione le volte successive, l'Execution Plan viene archiviato

in quell'area di memoria. In entrambe le casistiche, l'istruzione verrà eseguita, come mostrato di seguito (Figura 1.8).

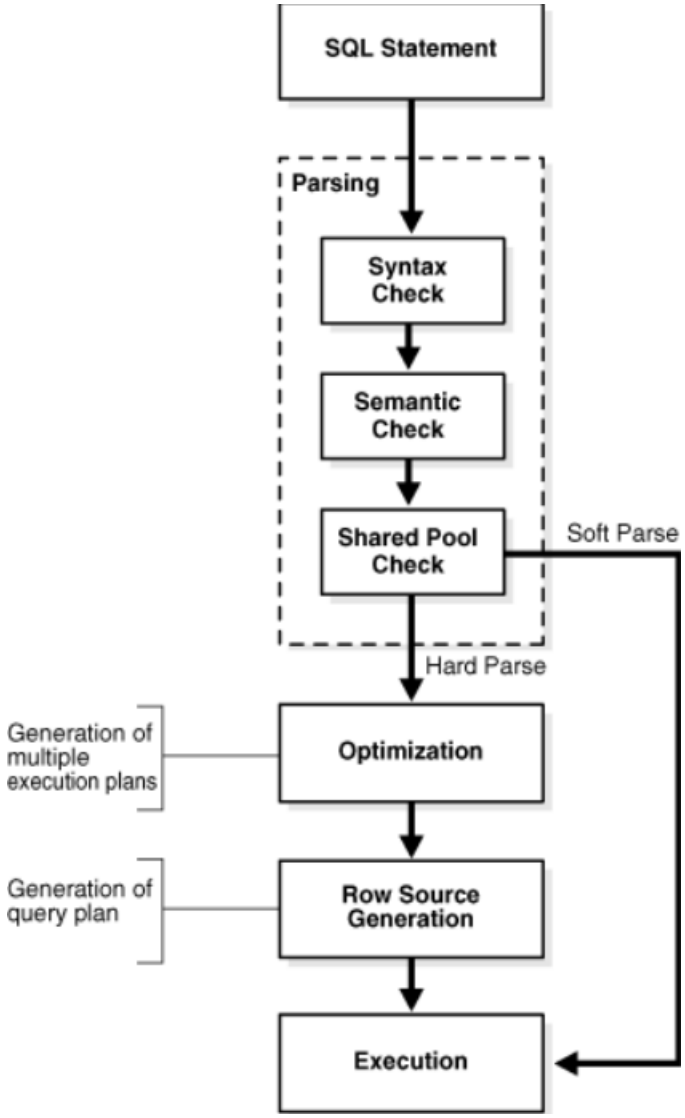

Figura 1.8: Soft e Hard Parsing

Esistono però casi in cui query, con identificatore identico (*SQL_ID*), generano più di un figlio: ciò implica un peggioramento del piano di esecuzione e, in concreto, un prolungamento dei tempi di risposta. Questo non per forza va ad impattare sulle performance di un Database, ma potrebbe essere una delle principali cause di

peggioramento delle stesse.

1.4.2 Ottimizzatore

L'ottimizzatore è software di database integrato che determina il metodo più efficiente per un'istruzione *SQL* per accedere ai dati richiesti, valuta e, se necessario, riscrive la query per massimizzare la sua efficienza.

L'ottimizzatore sceglie il piano con il costo più basso tra tutti i piani candidati considerati e utilizza le statistiche disponibili per calcolare i costi. Per una query specifica in un determinato ambiente, il calcolo dei costi tiene conto dei fattori dell'esecuzione della query come I/O, CPU e comunicazione.

Poiché il Database dispone di numerose statistiche e strumenti interni, l'ottimizzatore è generalmente in una posizione migliore rispetto all'utente per determinare il metodo ottimale di esecuzione delle istruzioni. Per questo motivo, tutte le istruzioni *SQL* utilizzano l'ottimizzatore.

L'ottimizzazione delle query è il processo generale di scelta del mezzo più efficiente per eseguire un'istruzione *SQL*. Trattandosi di un linguaggio non procedurale, quindi, l'ottimizzatore è libero di unire, riorganizzare ed elaborare in qualsiasi ordine.

Il Database ottimizza ogni istruzione *SQL* in base alle statistiche raccolte sui dati a cui si accede. L'ottimizzatore determina il piano ottimale per un'istruzione *SQL* esaminando più metodi di accesso, come la scansione completa della tabella o l'analisi dell'indice, diversi metodi di join come loop nidificati e hash join, diversi ordini di join e possibili trasformazioni.

Per una determinata query e ambiente, l'ottimizzatore assegna un costo numerico relativo a ciascuna fase di un possibile piano, quindi calcola insieme questi valori per generare una stima dei costi complessivi per il piano; la stessa procedura viene eseguita per tutti i piani alternativi. Successivamente, l'ottimizzatore sceglie il piano

con il preventivo più basso e, per questo motivo, viene talvolta chiamato Cost Based Optimizer (CBO).

L'ottimizzatore contiene tre componenti: il trasformatore, lo stimatore e il generatore di piani.

Il grafico seguente illustra i componenti appena citati (Figura 1.9).

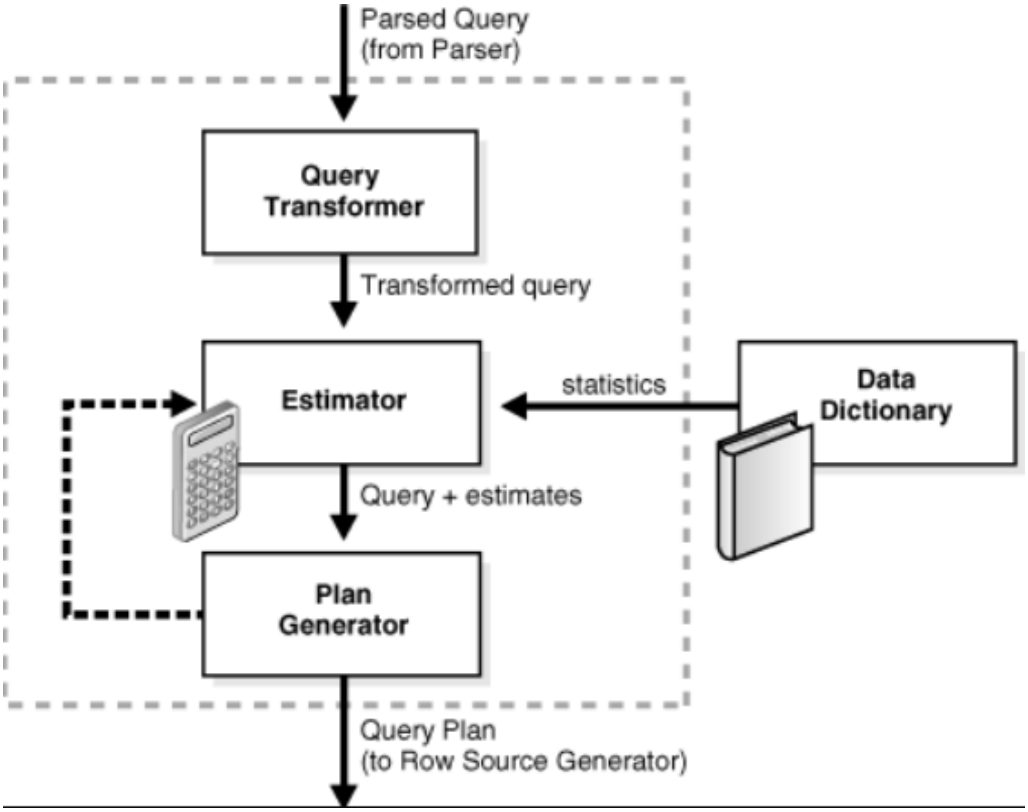

Figura 1.9: Componenti dell'ottimizzatore

Il trasformatore di query decide se è vantaggioso riscrivere l'istruzione *SQL* originale in un'istruzione *SQL* semanticamente equivalente con un costo inferiore, lo stimatore determina il costo complessivo di un determinato piano di esecuzione e il generatore di piani esplora vari piani per un blocco di query provando diversi percorsi di accesso.

1.4.3 Piani di esecuzione

A volte il risultato del processo appena discusso è la creazione di query multiple correlate. Vengono quindi generati uno o più piani di query per l'esecuzione con le prestazioni migliori; questi specificano le opzioni di esecuzione, come il tipo e l'ordine di combinazione, le opzioni di aggregazione e i requisiti di distribuzione dei dati.

Un piano di esecuzione descrive un metodo di esecuzione consigliato per un'istruzione *SQL*. Il piano mostra la combinazione dei passaggi che Oracle Database utilizza per eseguire un'istruzione *SQL*. Ogni passaggio recupera le righe di dati fisicamente dal Database o le prepara per l'utente che ha inserito l'istruzione.

Lo studio del piano di esecuzione di una query è uno step imprescindibile per migliorare le prestazioni di un Database relazionale e velocizzarne i tempi di esecuzione. La lettura del QEP (Query Execution Plan), noto anche come piano di accesso, è il primo passo da fare quando occorre migliorare le tempistiche di una query troppo lenta; la sua comprensione fornisce molte informazioni sulle operazioni svolte dal Database e sulle strategie da attuare per ottimizzarne le performance. Quando una query devia dal suo normale tempo di risposta accettato, molto probabilmente è dovuto a un cambiamento nel suo piano di esecuzione.

Nel grafico seguente, l'ottimizzatore genera due possibili piani di esecuzione per un'istruzione *SQL* di input, utilizza le statistiche per stimare i costi, li confronta e quindi sceglie il piano con il costo più basso (Figura 1.10).

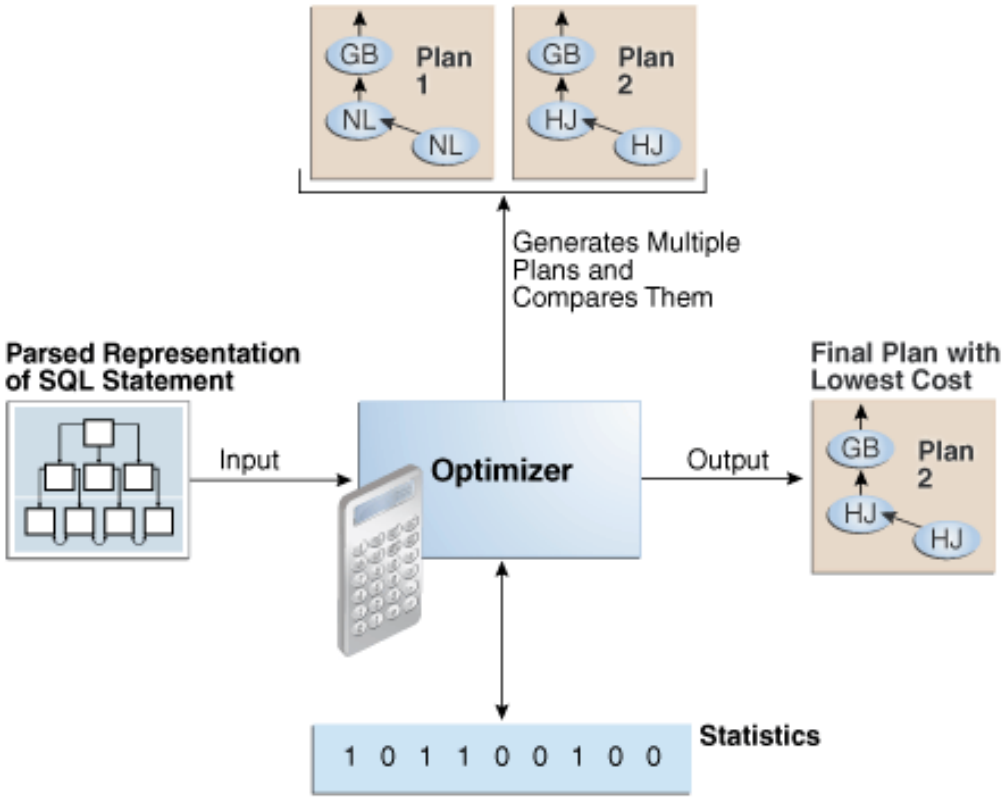

Figura 1.10: Piani di esecuzione

I piani di esecuzione sono caratterizzati da una struttura ad albero: la parte superiore dell'albero è l'istruzione principale che viene eseguita, scendendo di livello viene mostrato da quali elementi essa sia composta.

Si riporta di seguito un esempio (Figure 1.11 e 1.12).

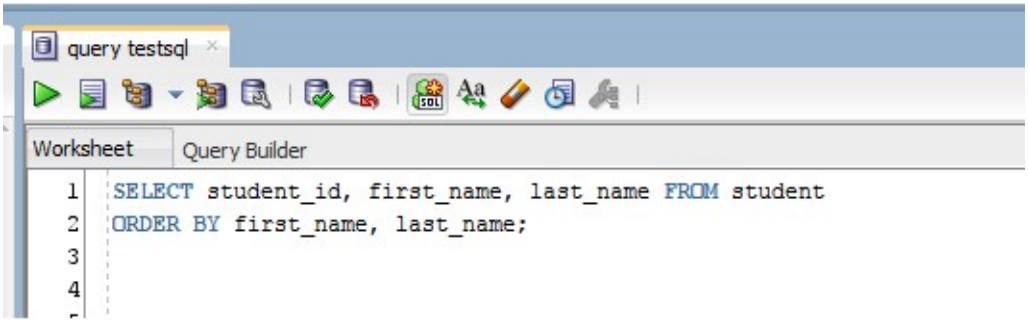

Figura 1.11: Esempio Query

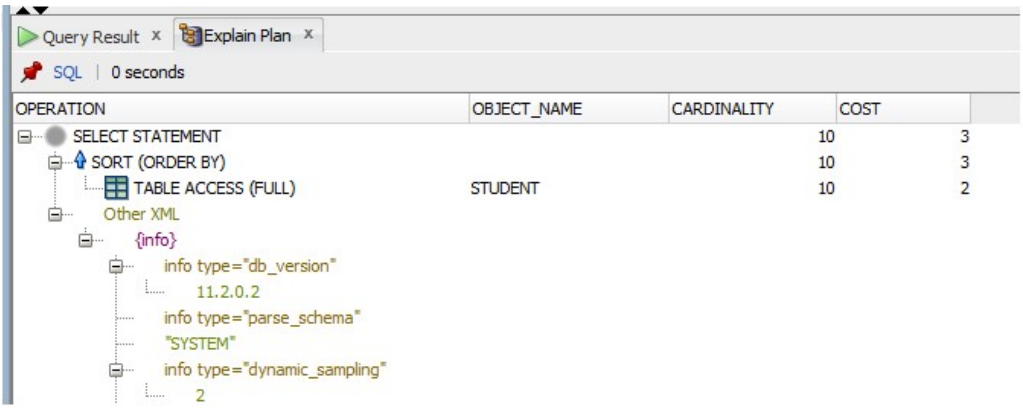

Figura 1.12: Esempio piano di esecuzione

I tre passi di cui si compone il piano di esecuzione della query riportata sopra sono i seguenti:

1. Viene eseguito un accesso alla tabella (FULL) che prende i dati dalla stessa
2. Viene eseguito un ordinamento (ORDER BY) dei dati
3. Viene scritta l'istruzione *SELECT*

Un piano di esecuzione mostra la cardinalità, che indica la frequenza delle righe nell'insieme dei risultati e il costo di ogni singola operazione e dell'intero piano, indicato alla riga 0.

Il costo è un'unità interna che il piano di esecuzione visualizza solo per consentire il confronto del piano. È un numero che rappresenta il tempo impiegato per eseguire una determinata istruzione e non possiede un'unità di misura, ma più basso è il costo, più veloce sarà l'esecuzione della query. Pertanto, non è possibile regolare o modificare il valore del costo.

Per il proseguo della trattazione di questa tematica si faccia riferimento al Capitolo 3, relativo alla vista *V$SQL_SHARED_CURSOR*, in cui verranno sviluppate le motivazioni relative alla non condivisione di piani di esecuzione.

Capitolo 2

SYSMETRIC

In questo capitolo saranno illustrate le prime operazioni effettuate sulla misurazione di metriche di performance del Database Oracle.

Nelle versioni precedenti all'11g venivano eseguite query relativamente costose in termini di risorse, che comportavano un sovraccarico eccessivo sul Database e indagavano diverse viste *V$*, come *V$SYSSTAT* E *V$STATNAME*. [8]

Ora, invece, tutte le statistiche significative sono precalcolate in automatico e rese disponibili per una consultazione immediata. Vengono raggruppate in metriche di breve e lunga durata, campionate rispettivamente ogni 15 e 60 secondi. I nomi dei gruppi di metriche sono disponibili nella vista *V$METRICGROUP*, mentre nella *V$METRICNAME* può esserne visualizzato l'elenco completo.

Oracle storicizza le informazioni sul carico di lavoro del Database nelle viste di sistema V*$SYSMETRIC*, *V$SYSMETRIC_HISTORY* e *V$SYSMETRIC_SUMMARY*. La *V$SYSMETRIC_HISTORY* [9] mostra la cronologia di tutte le metriche che, nell'ultima ora, sono state monitorate in un determinato intervallo di tempo, calcolato come differenza tra le colonne *BEGIN_TIME* e *END_TIME*; in essa esiste una singola riga per ogni metrica per ogni intervallo, ovvero quella più recente, a differenza della *V$SYSMETRIC* che contiene solo l'ultimo campione di tempo. La

vista *V$SYSMETRIC_SUMMARY* [10] possiede ulteriori colonne, che permettono di visualizzare informazioni riassuntive aggiuntive, quali valori minimo, medio, massimo e deviazione standard.

Esistono, infine, ulteriori tabelle temporanee create da Oracle che mantengono delle istantanee della stessa tipologia di dati, come la *DBA_HIST_SYSMETRIC_SUMMARY* e la *DBA_HIST_SYSMETRIC_HISTORY*.

Per tutte queste motivazioni precedentemente elencate, è stata individuata la *V$SYSMETRIC* come una buona soluzione per le attività che ci si è prefissati di portare a termine.

Nello specifico, quindi, sono state eseguite delle attività di Data Aggregation, rilevazione di valori anomali e analisi di correlazione tra le varie metriche prese in considerazione.

2.1 Data Collection & Data Ingestion

È stata realizzata una raccolta dati di due mesi circa, tramite la realizzazione di un job, ovvero un programma schedulato in un certo intervallo di tempo, che permette di storicizzare informazioni. Per essere eseguito deve, innanzitutto, essere creato e, successivamente, fatto eseguire (attraverso un comando di run); al termine del suo utilizzo è necessario determinarne lo stop.

Sono state memorizzate le informazioni riguardanti i 155 parametri contenuti nella *V$SYSMETRIC*. Sono stati analizzati in totale 78503 record, uno per ogni minuto, come si può notare dalla colonna *BEGIN_TIME*. Le altre metriche presenti descrivono diversi aspetti dell'attività del carico di lavoro del Database; due esempi sono il numero di chiamate al secondo e le letture fisiche per transazione, che aiutano il DBA (Database Administrator) a monitorare il Database e stimare l'efficacia con cui opera in un dato momento.

Nella seguente figura (Figura 2.1) si può vedere un estratto della tabella utilizzata.

	BEGIN_TIME	Background CPU Usage Per Sec	Background Time Per Sec	Temp Space Used	User Transaction Per Sec	Physical Reads Per Sec	Physical Writes Per Sec	Physical Writes Direct Per Sec	Total Parse Count Per Sec	Hard Parse Count Per Sec	...	Host CPU Utilization (%)
0	2022-01-18 23:58:26	84.065819	1.095944	3213885440	153.016030	1040.935383	122.178152	7.073211	2349.562056	2.892084	...	8.064783
1	2022-01-18 23:59:27	104.864552	1.956922	3366977536	163.268391	12549.899227	1819.499496	1214.544844	2266.644273	5.710447	...	12.556629
2	2022-01-19 00:00:26	152.542080	3.487549	3218079744	194.770703	9606.647971	1721.972946	1330.616958	2512.322666	5.658199	...	13.654166
3	2022-01-19 00:01:27	101.212733	2.007515	3222274048	198.974790	3311.882353	245.193277	10.000000	2485.495798	5.327731	...	8.965264
4	2022-01-19 00:02:26	88.825833	1.444276	3222274048	182.702792	3201.800760	185.412192	10.110689	2342.276557	4.642326	...	10.500874
...	...	...	...	...	...	...	...	...	...	...	...	...
78498	2022-03-14 23:54:26	77.012123	0.877022	3198156800	145.311984	8120.699901	132.271377	12.528887	1253.482998	1.469132	...	10.644811
78499	2022-03-14 23:55:27	97.831205	1.447217	3200253952	148.984729	9069.676120	418.996476	292.280584	874.811210	1.459977	...	13.716103
78500	2022-03-14 23:56:26	95.406727	1.048594	3199205376	144.422043	5383.719758	107.963710	11.290323	841.616263	1.041667	...	12.075410
78501	2022-03-14 23:57:26	96.163667	1.032587	3199205376	146.558838	5573.279419	118.814986	6.981350	857.138142	1.287341	...	11.547821
78502	2022-03-14 23:58:26	91.309681	1.004597	3199205376	127.644190	4055.893728	74.037330	4.943669	693.425256	1.059358	...	10.271063

78503 rows × 156 columns

Figura 2.1: Estratto *V$SYSMETRIC*

2.2 Framework di partenza

Su questi tipi di dati, in precedenza, sono già stati utilizzati algoritmi di Anomaly Detection in Real Time di tipo univariato (o monovariato), che prevedono la selezione delle variabili singolarmente, cioè una ad una senza metterle in relazione fra di loro. È stato utilizzato un apprendimento di tipo semi-supervisionato, il quale si colloca a metà tra l'apprendimento supervisionato, che utilizza solo dati etichettati, e l'apprendimento non supervisionato, che utilizza solo dati non etichettati; in questo modo è migliorata significativamente la qualità dell'apprendimento.

Ciò è stato realizzato attraverso un algoritmo di Isolation Forest [11], caratterizzato dalla creazione di un modello, che verrà utilizzato su un nuovo dataset per associare ogni punto ad un valore numerico: -1 se si tratta di un valore anomalo, 1 se normale. È una procedura costruita attorno alla teoria degli alberi decisionali e delle foreste

casuali. Quando viene presentato un set di dati, l'algoritmo li divide in due parti in base a un valore di soglia casuale. Questo processo continua in maniera ricorsiva fino all'isolamento di ogni elemento del dataset iniziale. Una volta analizzati tutti i dati, vengono filtrati i punti che hanno richiesto meno passaggi, rispetto agli altri, per essere isolati. È meno probabile che i campioni che scendono più in profondità nell'albero siano outliers poiché hanno richiesto più tagli per essere isolati, a differenza di quelli che finiscono sui rami più corti più facilmente separabili.

Nella seguente immagine (Figura 2.2) si può vedere un semplice schema di questo algoritmo.

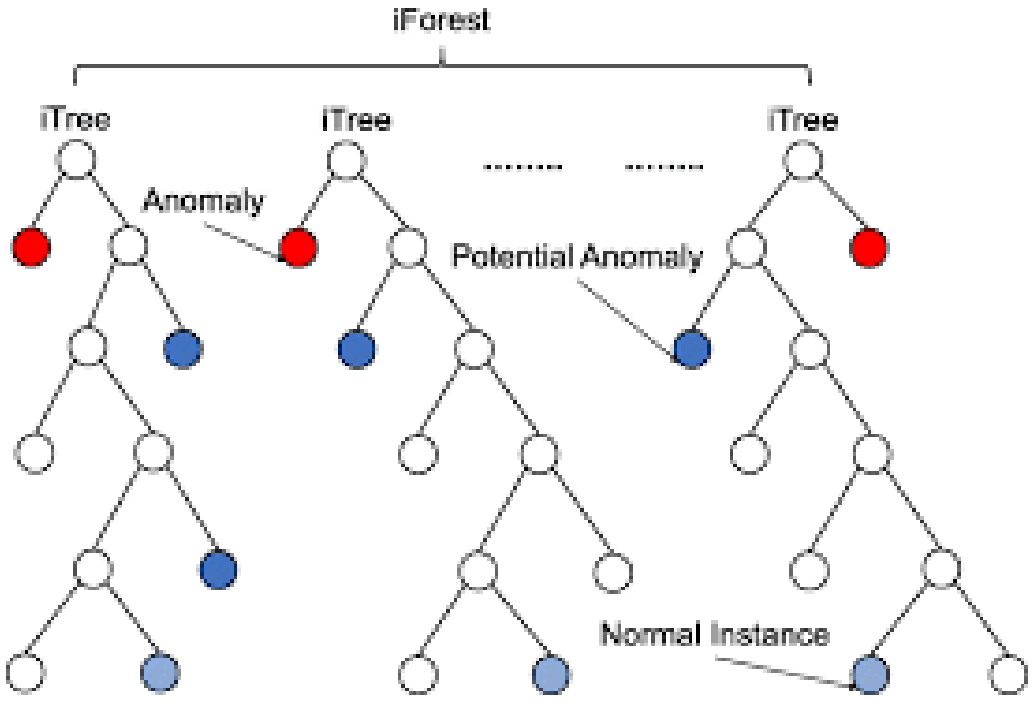

Figura 2.2: Isolation Forest

Questo elaborato si pone l'obiettivo di estendere il framework di partenza e gli algoritmi di Anomaly Detection utilizzati, introducendo nuove metodologie da applicare su differenti tipi di dati.

2.3 Anomaly Detection

Con questa base di partenza, a seguito di un accurato studio del contesto e un approfondimento sugli argomenti relativi allo sviluppo di questo elaborato, è stata effettuata un'analisi funzionale sulle metriche, sia individualmente che a livello aggregato. Si assuma, per semplicità per le prossime considerazioni, data l'elevata quantità di dati presi in analisi, che si stia lavorando con variabili causali che seguono una distribuzione normale.

Per una prima visualizzazione generale del Data Frame è stato ritenuto necessario visualizzare delle prime informazioni su alcune statistiche descrittive, calcolate in maniera complessiva per ogni singola colonna:

- **Valore minimo**
- **Valore massimo**
- **Media**
- **Deviazione standard**
- **25° percentile**
- **50° percentile**
- **75° percentile**

Si precisa che con il termine percentile si indica quel valore che delimita una certa porzione di dati (es: 25%) rispetto al totale e che, in particolare, il secondo preso in considerazione corrisponde alla mediana.

Un estratto di queste informazioni può essere visionato nella seguente figura (Figura 2.3).

	Background CPU Usage Per Sec	Background Time Per Sec	Temp Space Used	User Transaction Per Sec	Physical Reads Per Sec	Physical Writes Per Sec	Physical Writes Direct Per Sec	Total Parse Count Per Sec
count	78503.000000	78503.000000	7.850300e+04	78503.000000	78503.000000	78503.000000	78503.000000	78503.000000
mean	278.229127	4.816808	1.911122e+09	817.411751	29548.345584	2320.876704	1429.916671	3592.357884
std	130.970667	2.533204	1.253955e+09	1088.934449	24886.021946	4185.444152	3966.341930	2420.039618
min	13.006934	0.000000	0.000000e+00	4.000664	0.000000	0.462963	0.000000	0.000000
25%	175.249377	2.746063	8.042578e+08	346.434837	10480.828301	536.345606	40.319492	1677.936508
50%	268.719463	4.719455	1.712325e+09	640.020027	25204.959631	978.011696	142.011247	2792.399207
75%	361.274917	6.550524	2.985296e+09	887.474304	41068.523320	2181.815144	729.339839	5412.917346
max	1152.764322	21.728394	4.293919e+09	32791.684865	414613.468013	86166.521957	84955.284689	12244.275954

Figura 2.3: Esempio di output statistiche descrittive

Con una prima analisi macroscopica si può capire quale sia il dominio di riferimento dei dati e come essi siano distribuiti. Due valori troppo diversi di media e mediana potrebbero indicare un forte sbilanciamento della distribuzione dei dati, della presenza di una grande quantità di record anomali o di pochi outliers ma con valori molto estremi (troppo alti o troppo bassi).

Questo è stato il punto di partenza per una successiva analisi di Outliers Detection. Si è scelto di standardizzare la distribuzione di ogni metrica, al fine di vedere più facilmente quali osservazioni si trovassero sulle code. Per fare ciò, la differenza tra il valore di ogni singola osservazione e la relativa media del parametro preso in analisi è stata divisa per la deviazione standard, secondo la formula: [12]

$$z_i = \frac{(x_i - \mu)}{\sigma} \tag{2.1}$$

In questo modo si ottiene una distribuzione gaussiana, con media nulla, centrata sull'origine degli assi cartesiani. Avendo scelto un intervallo di confidenza bilaterale del 95%, sono stati selezionati dalle tavole della normale standard il limite superiore e inferiore dell'intervallo. Risulta, quindi, una probabilità di errore del 5% che questo range di valori non contenga il parametro stimato della popolazione. Infine, tutti i

valori maggiori di 1.96 e minori di -1.96 presenti nelle code sono stati evidenziati e valutati.

Nella seguente figura si può vedere una rappresentazione grafica della curva Gaussiana appena discussa (Figura 2.4).

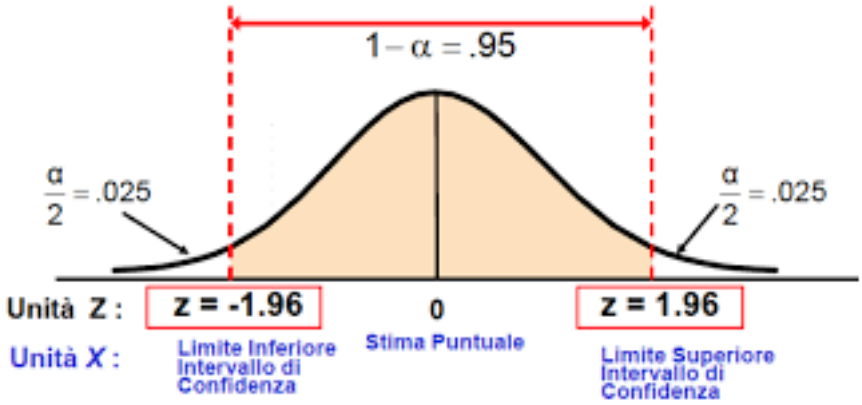

Figura 2.4: Normale standard con intervallo di confidenza bilaterale

I ragionamenti effettuati sono stati fatti anche in riferimento al significato della metrica, in quanto per alcune un valore anomalo potrebbe essere considerato tale se troppo alto, mentre per altre se troppo basso. Inoltre, un aspetto interessante scaturito da ciò, è il fatto che una serie di valori estremi contigui potrebbe essere messo in relazione a performance non soddisfacenti del Database.

Il metodo di Isolation Forest semi-supervisionato, utilizzato in precedenza, aveva la necessità di effettuare più volte il training del modello, ovvero una fase di addestramento predittiva, in modo che il risultato potesse essere utilizzato su nuovi dati. Con l'idea proposta e sviluppata in questo elaborato, invece, si supera questo limite in quanto, una volta impostato un intervallo di confidenza adeguato, ciò che ne segue è automatico.

2.4 Metric Segmentation

A seguito di questa prima fase esplorativa dei dati, l'attenzione di questa trattazione si è spostata sulle attività di segmentazione della totalità delle metriche presenti.

Ciò è stato realizzato per affiancare, ad una visione più generale e complessiva, una più riassuntiva capace di riordinare e dividere le metriche per affinità.

Per raggiungere questo scopo è stata effettuata un'operazione manuale di preselezione degli attributi più significativi e utili (features selection).

Successivamente è stata effettuata una segmentazione delle metriche in base alle categorie di appartenenza. Sono quindi stati creati dei gruppi omogenei, sia dal punto di vista del significato che dalle logiche sottostanti al funzionamento dell'Oracle Database.

Si riportano di seguito i gruppi che sono stati individuati, ordinati casualmente, a cui è stata assegnata un'etichetta che li potesse raggruppare e includerne la maggior parte:

- **Gruppo 1** → CPU
- **Gruppo 2** → Reads/Writes
- **Gruppo 3** → SGA
- **Gruppo 4** → Hard/Soft Parsing
- **Gruppo 5** → Index/Table Scans
- **Gruppo 6** → Database Block
- **Gruppo 7** → Logons

Ogni sottoinsieme identificato è caratterizzato da una quantità di metriche contenute al suo interno differente. Quello più numeroso è quello relativo a processi di lettura e scrittura (Input/Output) con 31 elementi, seguono i gruppi etichettati come SGA e CPU, rispettivamente con 13 e 12 item ciascuno; al termine di questo elenco si trovano Index/Table Scans, Database Block, Hard/Soft Parsing e Logons, dai 10 ai 7 componenti.

Si riportano di seguito le tabelle contenenti alcuni dei gruppi più rappresentativi, le cui tematiche sono già state affrontate lungo il corso di questo elaborato.

Gruppo 3 (SGA)
Shared Pool Free %
Row Cache Miss Ratio
Library Cache Miss Ratio
Buffer Cache Hit Ratio
Cursor Cache Hit Ratio
Row Cache Hit Ratio
PGA Cache Hit %
Library Cache Hit Ratio
Redo Allocation Hit Ratio
Redo Generated Per Txn
Redo Writes Per Txn
Redo Generated Per Sec
Redo Writes Per Sec

Gruppo 4 (Hard/Soft Parsing)
Total Parse Count Per Sec
Total Parse Count Per Txn
Hard Parse Count Per Sec
Hard Parse Count Per Txn
Parse Failure Count Per Sec
Parse Failure Count Per Txn
Execute Without Parse
Soft Parse Ratio

Gruppo 5 (Index/Table Scans)
Long Table Scans Per Txn
Full Index Scans Per Txn
Full Index Scans Per Sec
Total Index Scans Per Sec
Total Index Scans Per Txn
Long Table Scans Per Sec
Total Table Scans Per Txn
Total Table Scans Per Sec
Total Sorts Per User Call
Total Table Scans Per User Call

2.5 Correlation Analysis

Successivamente, in relazione anche alle attività precedentemente descritte, il focus della trattazione si è spostato sull'analisi della possibile esistenza di legami tra metriche prese singolarmente o in maniera aggregata.

Un'analisi di correlazione può essere effettuata senza aver predeterminato delle sottocategorie, ma produce una notevole quantità di risultati difficile da poter analizzare nel dettaglio, essendo caratterizzata da un'eccessivo numero di informazioni. Per questo motivo si è scelto di effettuare queste considerazioni anche in relazione alla segmentazione appena effettuata.

Inoltre risulta di grande valore l'apporto che questo studio può fornire sotto un duplice aspetto: possibilità di ridurre la ridondanza di modello, in quanto si possono prevedere e anticipare le anomalie tra metriche correlate tra loro, e fornire maggiori informazioni funzionali alla figura del consulente, permettendogli di utilizzare un set di strumenti più ampio per rilevare e indagare gli insights nascosti.

Si specifica che sono state effettuate le medesime procedure, che verranno di seguito presentate, per ognuno dei 7 gruppi identificati.

È stato deciso di calcolare il coefficiente di correlazione di Pearson tra tutte le combinazioni di coppie di attributi, in quanto avendo posto l'ipotesi di normalità delle distribuzioni è stato preferito rispetto a quello di Spearman, che si sarebbe adattato meglio nel caso in cui ci fossero stati un elevato numero di outliers.

A livello statistico, è un indice che esprime un'eventuale relazione di linearità tra due variabili A e B, e si ottiene tramite la seguente formula:

$$\rho_{AB} = \frac{cov(A, B)}{\sigma_A \cdot \sigma_B} \tag{2.2}$$

Ha un valore compreso nell'intervallo [-1, +1], in cui gli estremi e il punto centrale hanno i seguenti significati:

- **+1** → perfetta correlazione lineare positiva
- **0** → assenza di correlazione lineare
- **-1** → perfetta correlazione lineare negativa

Nel contesto di business in esame, quindi, si è visto in maniera operativa come rispecchiasse una buona correlazione dal punto di vista funzionale. È stato fissato un valore soglia di 0.7, scelto per un duplice motivo: è stato considerato un valore interessante a seguito della consultazione della documentazione in merito e valutato, dal punto di vista pratico e sperimentale, come un buon delimitatore tra quelle relazioni che potevano essere considerate più "interessanti" e quelle meno.

Si precisa che per questa analisi appena descritta sono stati utilizzati tutti i dati a disposizione, per avere un campione più numeroso e rappresentativo ed essere il più accurati possibile, e che lo stesso ragionamento è stato fatto per valori superiori a 0.7 e inferiori a -0.7.

2.5.1 Matrice di correlazione

La matrice di correlazione è una tabella quadrata, ossia con lo stesso numero di righe e colonne, che mostra i coefficienti di correlazione tra coppie di variabili. Si tratta, in definitiva, di un prospetto che permette di valutare, nell'insieme, il grado di interdipendenza di una serie di grandezze. [13]

Le intestazioni di riga e colonna contengono i nomi delle variabili esaminate. Le celle all'interno della tabella, invece, mostrano i coefficienti di correlazione tra le varie metriche.

Al di sopra e al di sotto della diagonale abbiamo le coppie dei coefficienti di correlazione. Poiché il grado di dipendenza tra le variabili A e B è il medesimo, se si inverte l'ordine, le due parti della tabella avranno gli stessi valori.

La matrice, pertanto, può essere letta considerando solo una delle due metà. Per tale ragione la matrice è definita simmetrica.

Successivamente, è stata quindi creata una matrice di correlazione 155x155, contenente, nello stesso ordine su righe e colonne, tutte le metriche prese in analisi. [14]

Di seguito viene riportato un piccolo estratto della stessa (Figura 2.5).

	Background CPU Usage Per Sec	Background Time Per Sec	Temp Space Used	User Transaction Per Sec	Physical Reads Per Sec	Physical Writes Per Sec	Physical Writes Direct Per Sec	Total Parse Count Per Sec	Hard Parse Count Per Sec	Cursor Cache Hit Ratio	User Rollback UndoRec Applied Per Sec
Background CPU Usage Per Sec	1.000000	0.915771	0.127405	0.335558	0.454489	0.113484	0.047794	0.518326	0.268998	0.000283	0.080910
Background Time Per Sec	0.915771	1.000000	0.116969	0.379027	0.462206	0.136903	0.062402	0.565450	0.283064	-0.000294	0.083133
Temp Space Used	0.127405	0.116969	1.000000	0.027895	0.083068	0.056811	0.043311	0.060891	-0.000761	0.002939	0.021181
User Transaction Per Sec	0.335558	0.379027	0.027895	1.000000	0.148948	0.104452	0.055801	0.069484	0.044936	-0.000845	0.065401
Physical Reads Per Sec	0.454489	0.462206	0.083068	0.148948	1.000000	0.351952	0.314817	0.279108	0.183249	-0.001097	0.046480
Physical Writes Per Sec	0.113484	0.136903	0.056811	0.104452	0.351952	1.000000	0.973034	-0.123995	0.005797	-0.002110	0.079621
Physical Writes Direct Per Sec	0.047794	0.062402	0.043311	0.055801	0.314817	0.973034	1.000000	-0.130660	-0.001678	-0.001613	0.060970
Total Parse Count Per Sec	0.518326	0.565450	0.060891	0.069484	0.279108	-0.123995	-0.130660	1.000000	0.360965	-0.010008	-0.062636
Hard Parse Count Per Sec	0.268998	0.283064	-0.000761	0.044936	0.183249	0.005797	-0.001678	0.360965	1.000000	-0.001454	-0.003750

Figura 2.5: Estratto Matrice di Correlazione

Questa rappresentazione, godendo della proprietà di simmetria, è ridondante, in quanto possiede tutti 1 sulla diagonale e i due triangoli che si creano sono uno il trasposto dell'altro. È stato quindi sufficiente considerare solo una porzione di questa tabella, i cui valori sono stati riordinati in ordine decrescente, per una migliore comprensione.

Si è scelto di visualizzare il risultato graficamente grazie ad una heatmap, una rappresentazione grafica dei dati dove i singoli valori contenuti in una matrice sono rappresentati da colori. Si precisa che, grazie a Plotly, una libreria grafica open source per Python, è stato possibile realizzare una mappa di calore interattiva, che è possibile consultare sia in maniera più generale che particolare.

In questo caso (Figura 2.6), attraverso una scala cromatica dal blu (corrispondente al valore -1) al giallo (corrispondente al valore 1), vengono espresse le relazioni esistenti tra un sottoinsieme di 5 metriche.

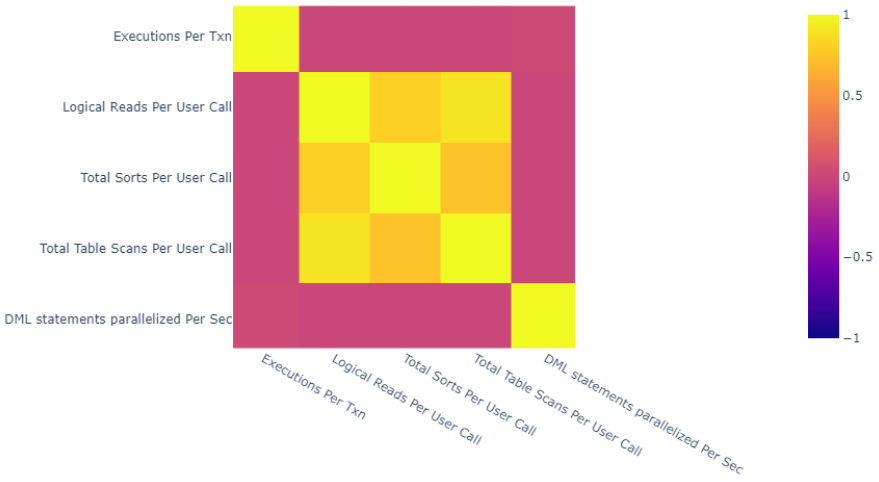

Figura 2.6: Estratto di Heatmap

2.6 Data Visualization

Successivamente, è stato estratto un campione di 10.000 record per ogni metrica. Sono stati realizzati dei line charts, i quali hanno permesso una visualizzazione grafica di un lasso di tempo continuo, che rappresenta circa il 13% del dataset. In questo caso, rispetto alle considerazioni precedenti, è stata scelta solo una limitata porzione dei dati in quanto, nel caso in cui fossero stati scelti un numero maggiore di elementi, i grafici ottenuti avrebbero avuto una rappresentazione poco chiara.

Di seguito vengono riportati alcuni plot (da Figura 2.7 a Figura 2.12) di metriche [15] [16] contenute in ciascuno dei tre gruppi riportati nella sezione precedente (Gruppo 3, 4 e 5).

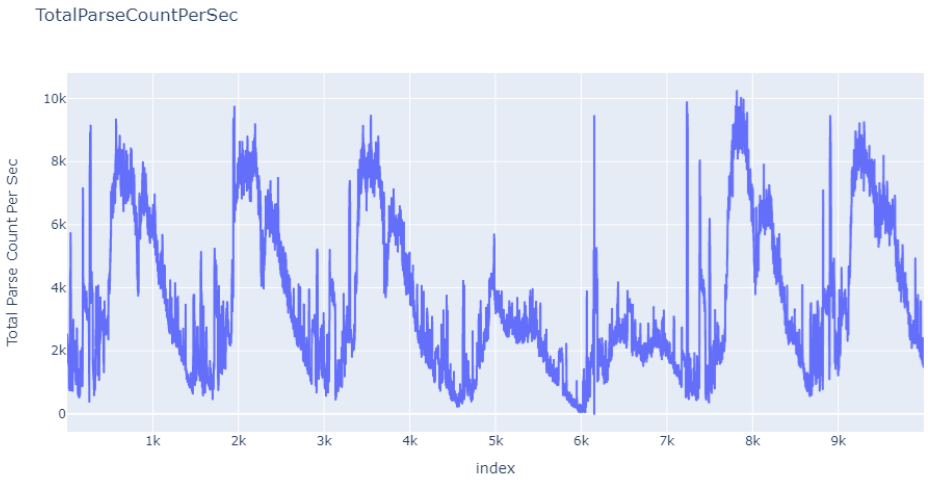

Figura 2.7: Total Parse Count Per Sec

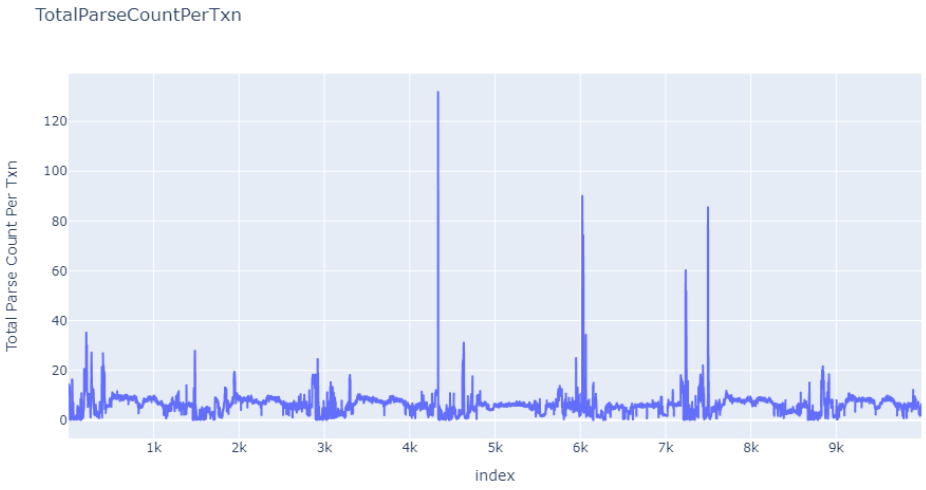

Figura 2.8: Total Parse Count Per Txn

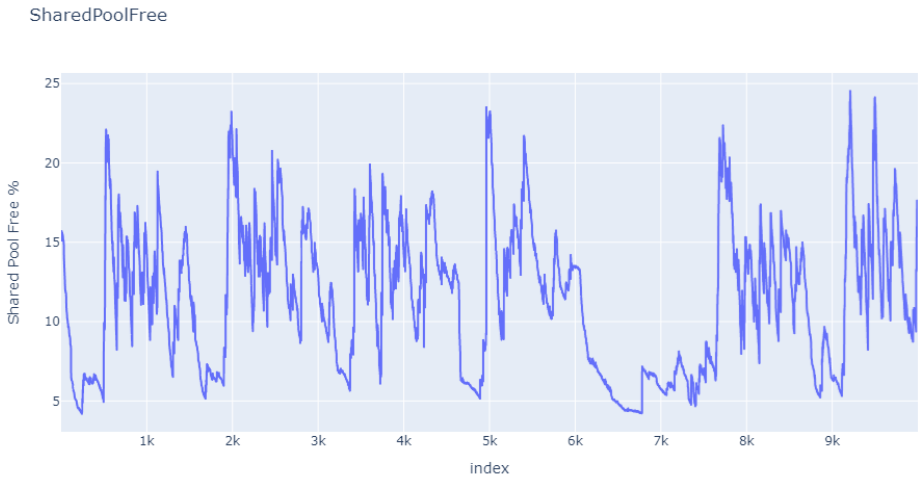

Figura 2.9: Shared Pool Free %

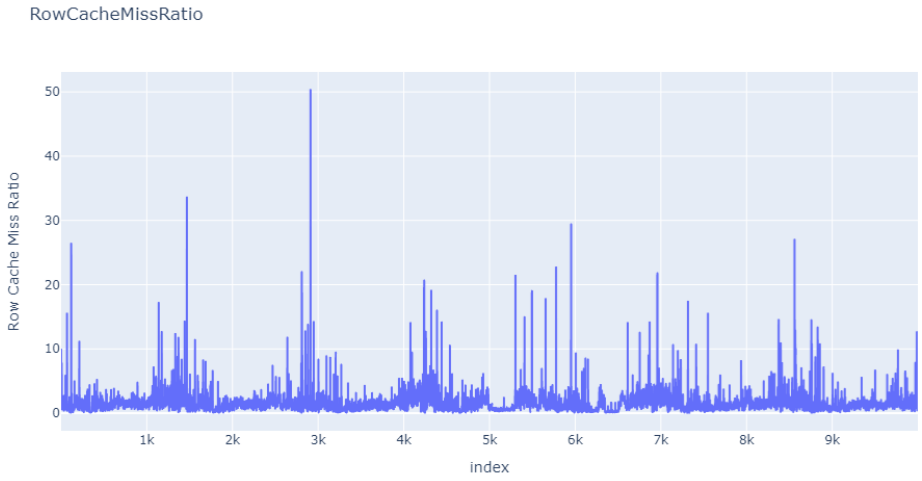

Figura 2.10: Row Cache Miss Ratio

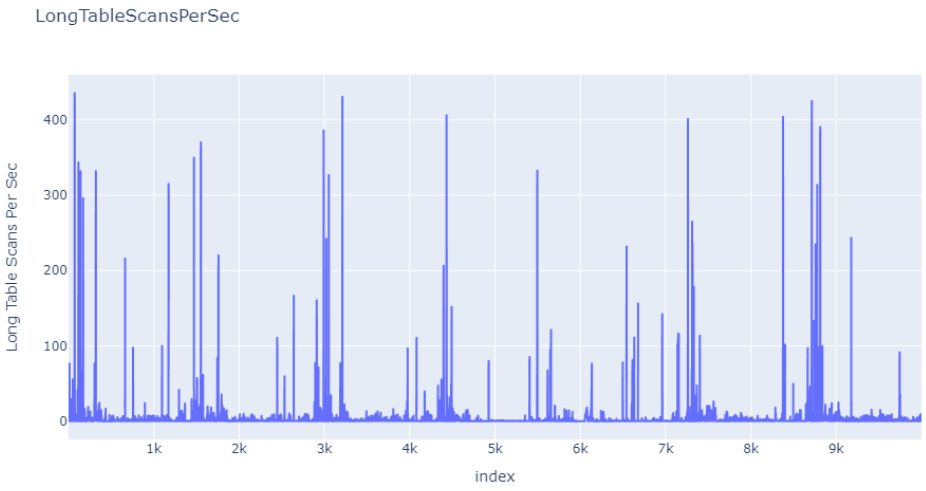

Figura 2.11: Long Table Scans Per Sec

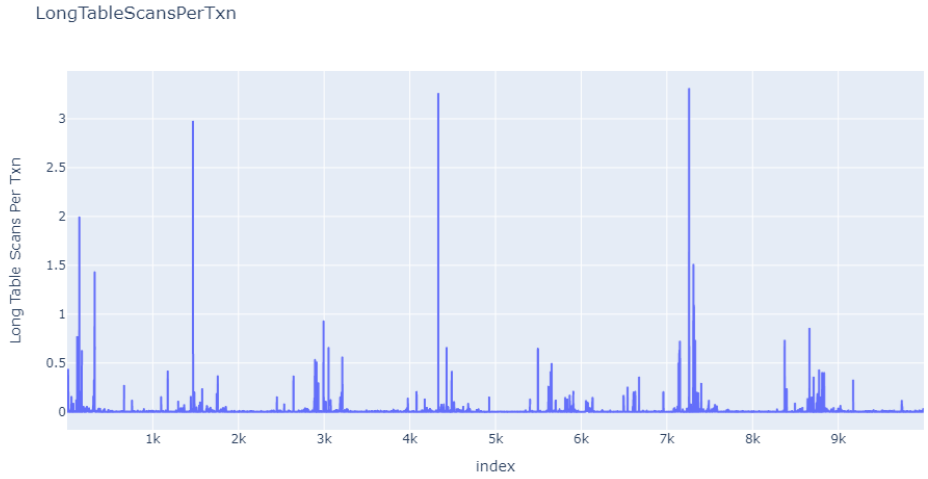

Figura 2.12: Long Table Scans Per Txn

Infine, sono stati uniti i due procedimenti, ovvero, per le coppie di metriche con un coefficiente di correlazione rilevante, sono stati confrontati i due plot e valutata la bontà del calcolo effettuato in precedenza. In particolare, per avere una buona corrispondenza tra la comparazione dei due grafici e il mero valore numerico, è

necessario che ci sia un andamento simile, non solo nei casi più stazionari, ma soprattutto nei punti in cui si rilevano picchi o valli (valori anomali).

Prendendo a coppie i sei grafici riportati sopra, vengono riportati i coefficienti di correlazione, calcolati tra Total Parse Count Per Sec e Total Parse Count Per Txn (2.3), Shared Pool Free % e Row Cache Miss Ratio (2.4), Long Table Scans Per Sec e Long Table Scans Per Txn (2.5).

$$\begin{bmatrix} 1 & 0.27441159 \\ 0.27441159 & 1 \end{bmatrix} \tag{2.3}$$

$$\begin{bmatrix} 1 & -0.0003612 \\ -0.0003612 & 1 \end{bmatrix} \tag{2.4}$$

$$\begin{bmatrix} 1 & 0.95459937 \\ 0.95459937 & 1 \end{bmatrix} \tag{2.5}$$

2.7 Cluster Analysis

2.7.1 Definizione teorica

Per approfondire maggiormente l'argomento in questione, è stato ritenuto necessario affiancare alla precedente procedura di segmentazione l'output di un algoritmo di machine learning non supervisionato.

Si è deciso di adottare questo approccio per due motivazioni in particolare: da un lato confrontare e validare, sebbene si stia prendendo in considerazione due procedimenti ben distinti, la segmentation manuale sviluppata in una sezione precedente di questo capitolo e, d'altra parte, per identificare la possibile esistenza di similarità

di comportamento delle metriche appartenenti ad un gruppo che, apparentemente, potrebbero risultare anche molto distanti tra loro.

L'apprendimento senza supervisione avviene quando i dati di input non sono etichettati o non hanno un corrispondente valore di output e l'algoritmo deve scoprire eventuali relazioni esistenti. Il suo utilizzo è adatto a cercare modelli nascosti nei dati. Ciò è stato portato a termine grazie allo sviluppo di una Cluster Analysis.

La Cluster Analysis, o Clustering, rappresenta l'attività descrittiva nei processi di Data Mining nei Big Data e trova applicazioni in vari settori, dalle scienze sociali al marketing, dalla medicina alla biologia, dalla fisica all'economia [17]. L'obiettivo consiste nel classificare i dati in strutture, in modo tale da semplificarne la comprensione.

La Cluster Analysis (dall'inglese cluster: grappolo, gruppo o sciame) è un metodo statistico per processare i dati, raggruppando gli elementi di un insieme, a seconda delle loro caratteristiche, in classi non assegnate a priori. Serve a mostrare relazioni fra i dati che a prima vista non risultano evidenti, in modo da creare insiemi omogenei utili per ulteriori analisi. È un metodo non supervisionato per ottenere insight dei dati nel mondo dei Big Data.

In statistica è un insieme di strumenti e algoritmi usati per classificare differenti oggetti in gruppi, in modo tale che la somiglianza fra due oggetti sia massima nel caso in cui appartengano allo stesso gruppo e, viceversa, sia minima.

Nella seguente immagine (Figura 2.13) si può vedere una rappresentazione grafica di cosa si intende per Clustering.

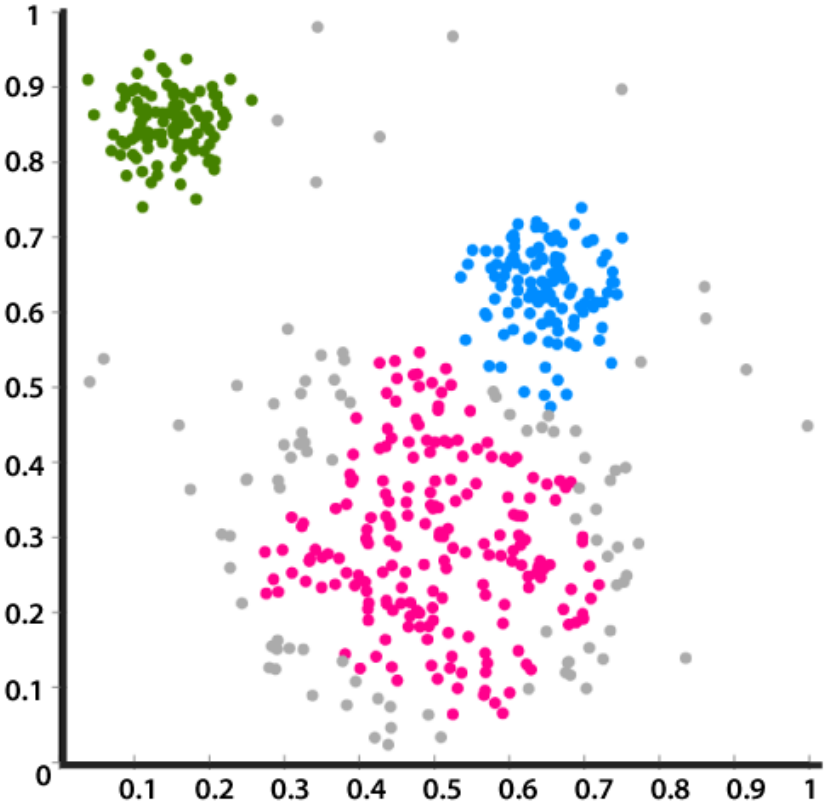

Figura 2.13: Clustering

La Cluster Analysis, quindi, funziona organizzando gli elementi in gruppi o cluster, sulla base della somiglianza e omogeneità fra di loro. A tale scopo vengono sfruttati algoritmi di classificazione sempre più complessi sotto il profilo informatico, ma anche sempre più performanti, al fine di estrarre informazioni utili dai dati, mediante un'accurata e puntuale classificazione.

2.7.2 Hierarchical Clustering

La scelta della tipologia di Clustering da implementare non è stata banale. Ad una valutazione generale su quali algoritmi potessero essere esclusi a priori a causa, ad

esempio, di alcuni parametri di input che, per il contesto analizzato, fossero difficili o non fosse corretto stimare (es: quantità minima di oggetti del cluster o numero di cluster), è seguita una serie di tentativi per poter identificare il migliore.

Al termine di questa fase, il Clustering Gerarchico [18] è stato stabilito come il migliore per alcune caratteristiche intrinseche che verranno illustrate di seguito.

Innanzitutto, bisogna approfondire quali siano le peculiarità di questo algoritmo, del quale ne esistono due versioni:

- **Agglomerativo**, partendo da un cluster differente per ogni elemento si aggrega in maniera iterativa fino al raggiungimento di un unico cluster che contiene tutti i dati.
- **Divisivo**, ovvero l'opposto del precedente.

L'output principale è rappresentato da un dendogramma, una rappresentazione grafica chiara che mostra le relazioni gerarchiche tra i vari cluster. Rappresenta un diagramma ad albero attraverso il quale vengono rappresentati sull'asse delle ascisse i dati in input e su quello delle ordinate la "distanza" tra i cluster.

Una sua illustrazione è riportata di seguito (Figura 2.14).

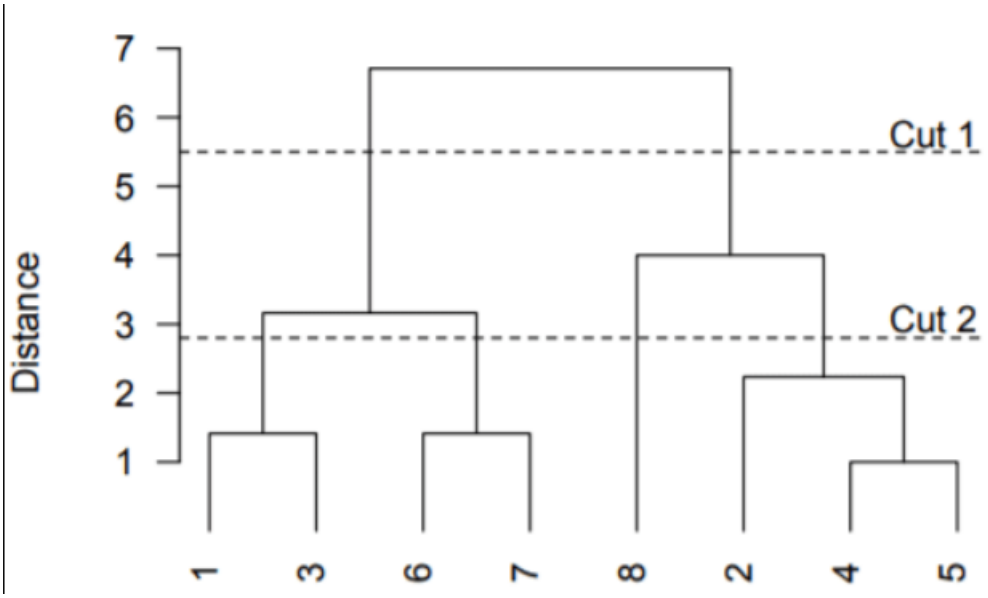

Figura 2.14: Dendogramma Hierachical Clustering

Si può notare che, al variare del parametro distance, possono essere generati diversi livelli di taglio, che corrispondono a gradi di dettaglio differenti e, di conseguenza, configurazioni e numero di cluster variabili.

Ed è proprio questo il vantaggio nell'utilizzo del tipo di algoritmo appena illustrato, che si inserisce perfettamente con l'idea di possedere un livello di aggregazione delle metriche di performance variabile, a seconda del grado di specificità richiesto da chi ha il compito di doverle analizzare. Un focus più agglomerativo è richiesto per lo sviluppo di considerazioni più di carattere macroscopico, mentre scendendo più nello specifico aumentano comportamenti distinti del sempre più elevato numero di cluster.

In particolare, per la parte prettamente operativa, è stata calcolata la distanza euclidea per ogni coppia di metriche, una misura della similarità tra esse, attraverso la seguente formula:

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{n} (x_{ik} - x_{jk})^2} \tag{2.6}$$

Successivamente è stato utilizzato il metodo *linkage* [19] per calcolare la distanza d(s,t) tra i cluster s e t. L'algoritmo ha inizio da una foresta di cluster; quando due cluster s e t appartenenti ad essa sono combinati in un singolo cluster u, s e t sono rimossi dalla foresta e u viene aggiunto. Quando rimane un unico cluster all'interno della foresta, l'algoritmo termina e questo cluster diventa la radice.

Ad ogni iterazione vengono aggiornate:

- una matrice delle distanze d[i, j]
- una matrice delle distanze d(u, v)

All'interno di d[i, j] sono mantenute le distanze tra il cluster i e j nella foresta originale. Nella matrice d(u, v), invece, sono memorizzate le distanze del nuovo cluster u che è stato formato con i rimanenti cluster all'interno della foresta; ciò viene fatto tra tutti i punti i nel cluster u e tutti i punti j nel cluster v, secondo la formula:

$$d(u, v) = max(dist(u[i], v[j])) \tag{2.7}$$

Una volta calcolata la soglia della distanza tra i vari cluster, anche grazie alle informazioni sul grado di dissimilarità tra i gruppi appena ottenuto, è stato possibile raggiungere il risultato finale.

Al fine di completare la procedura, è stato necessario fornire manualmente un'etichetta ad ogni gruppo. Questa operazione non è stata portata a termine per tutti gli insiemi di parametri, in quanto per alcuni non è stato possibile identificare un label di classe appropriato.

Arrivati a questo punto, a differenza del procedimento di segmentazione effettuato in precedenza, si è raggiunto il risultato finale che ha previsto la creazione di 10 cluster, ma ne sono stati identificati solo 8, ovvero:

- **Gruppo 1** → User-level Calls
- **Gruppo 2** → Data Block
- **Gruppo 3** → SGA
- **Gruppo 4** → Reads
- **Gruppo 5** → CPU
- **Gruppo 6** → Transactions
- **Gruppo 7** → Sessions
- **Gruppo 8** → Hard/Soft Parsing

In questo caso, si evidenzia un maggior sbilanciamento nella numerosità dei cluster identificati. Quello relativo alla SGA, infatti, con ben 65 elementi, risulta essere decisamente più ricco rispetto, ad esempio, ai primi due gruppi con rispettivamente 3 e 6 item ciascuno.

Come si può notare dalle etichette, nella maggior parte dei casi c'è un collegamento tra la segmentazione, effettuata in precedenza, e questa. È bene evidenzare però, che in questo tipo di analisi emerge un'associazione non solo per categorie, ma anche per tipo di comportamento affine delle metriche.

Si riporta in una tabella, a titolo di esempio, il Gruppo 8:

Gruppo 8 (Hard/Soft Parsing)
Total Parse Count Per Sec
Hard Parse Count Per Sec
User Rollbacks Per Sec
Enqueue Timeouts Per Sec
GC CR Block Received Per Second
Current Open Cursors Count
Network Traffic Volume Per Sec
Parse Failure Count Per Sec
GC Current Block Received Per Second
CR Blocks Created Per Sec
Run Queue Per Sec
User Calls Per Sec
Redo Writes Per Sec
Queries parallelized Per Sec

Capitolo 3

SQL_SHARED_CURSOR

A seguito dell'approfondita digressione nel capitolo precedente sull'analisi di performance del Database Oracle, l'attenzione di questo elaborato si sposta al lavoro svolto riguardante le attività di Anomaly Detection relative ai piani di esecuzione del Database.

In questo capitolo verranno presentati gli studi e le operazioni effettuate sulla *V$SQL_SHARED_CURSOR* che, nonostante i continui aggiornamenti a versioni sempre più recenti, non presenta ancora una visualizzazione delle informazioni contenute in essa sufficientemente chiara e comprensibile.

Come anticipato nel Capitolo 1, verranno indagate più a fondo le tematiche riguardanti i piani di esecuzione. Nello specifico, in questa vista sono contenute tutte le informazioni necessarie per comprendere per quale motivo, per la stessa interrogazione al Database, viene staccato più di un figlio.

Non ci sono state valide alternative da valutare, dato che nella *V$SQL_SHARED_CURSOR* si trova tutto ciò che occorre per indagare sul perché vengono generati molteplici piani di esecuzione e non viene condiviso il cursore. È stato effettuato, quindi, più un lavoro di riorganizzazione dei dati, sul modo in cui essi venivano presentati.

Scendendo più nel dettaglio, dopo una prima fase di approfondimento degli

elementi che verranno illustrati, si è sviluppata una fase di manipolazione dei dati a disposizione, al fine di migliorarne le proprietà di fruibilità e velocità di lettura.

3.1 Data Exploration & Data Manipulation

Oracle esternalizza i motivi di non condivisione di un piano di esecuzione in questa vista, che contiene, oltre ai valori identificativi dell'istruzione *SQL* e del child, 64 possibili motivi per cui un piano di esecuzione non è stato condiviso.

Il primo campo rappresenta l'identificatore della query *SQL*, il secondo invece l'indirizzo del cursore padre, univoco per ogni *SQL_ID*; successivamente si trovano il *CHILD_NUMBER*, un numero progressivo che parte da 0 separatamente per ogni *SQL_ID*, e il *CHILD_ADDRESS*, che rappresenta l'indirizzo del cursore figlio, univoco per ogni coppia *SQL_ID* e *CHILD_NUMBER*. Seguono, infine, un grande numero di attributi che possono essere impostati al valore booleano 'Y' o 'N'.

Si possono vedere alcuni record, presi come esempio, nella seguente figura (Figura 3.1).

	SQL_ID	ADDRESS	CHILD_ADDRESS	CHILD_NUMBER	OPTIMIZER_MISMATCH	BIND_MISMATCH
0	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x8c\xb8W\x00'	0	Y	N
1	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x89\xd5w\x80'	1	Y	N
2	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x89\xa9\xf7\xc0'	2	Y	N
3	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x85Z+\x08'	3	Y	N
4	1gfaj4z5hn1kf	b'\x00\x00\x00\x00\x89\xac\x02\x98'	b'\x00\x00\x00\x00}\xfc\xb6\xc0'	2	Y	N
5	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00\x85\x13,\xb8'	0	N	Y
6	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00h\xf0\x19P'	1	N	Y
7	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00}\xf3r@'	2	N	Y
8	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00\x8c\xd4\xce\x18'	3	N	Y
9	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00\x89\xd2/\x00'	4	N	Y
10	34rznuxy8h2a4	b'\x00\x00\x00\x00\x85\xaa\xd9\x10'	b'\x00\x00\x00\x00\x89H\x8e\xc0'	5	N	Y

Figura 3.1: Estratto *V$SQL_SHARED_CURSOR*

È stato scelto di andare a vedere più da vicino quelli che, nel dataset in possesso, si sono verificati con una frequenza maggiore; per questo motivo, di seguito

vengono riportati i principali in un elenco, aggiungendo la definizione presa dalla documentazione di Oracle, ed una breve analisi successiva.

- **ROLL_INVALID_MISMATCH** (*Marked for rolling invalidation and invalidation window exceeded*)
  Ha a che fare con il cursore e, più nello specifico, con l'invalidazione dello stesso.

  Come si può notare nella seguente immagine (Figura 3.2), il ciclo di vita di un cursore è composto da una fase di dichiarazione, in cui è necessario definire un nome e fornire un'istruzione *SELECT* per il set di risultati che si vuole ottenere; poi, attraverso l'istruzione *FETCH*, è possibile recuperare una riga per volta i risultati e, infine, si provvede alla chiusura e alla deallocazione del cursore.

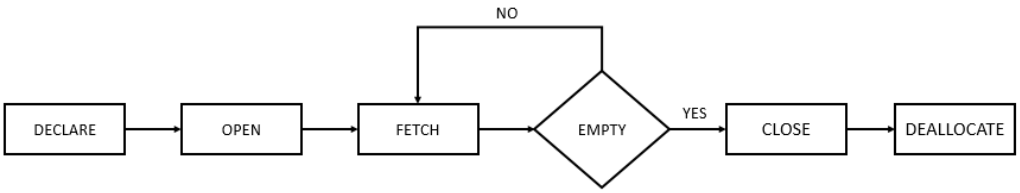

Figura 3.2: Ciclo di vita cursore

Prima della versione 10g, i cursori nel pool condiviso venivano invalidati immediatamente dopo aver raccolto le statistiche sugli oggetti referenziati [20]. Nelle versioni successive, l'invalidazione del cursore viene gestita in maniera diversa. Vengono raccolte le statistiche sulle tabelle a cui fa riferimento il cursore. La volta successiva che la query verrà eseguita per questo cursore, Oracle genererà un numero casuale compreso tra 0 e il valore del parametro *OPTIMIZER INVALIDATION PERIOD* (impostato di default a 18.000 secondi). Il cursore rimarrà valido per il numero di secondi generato casualmente, verrà poi invalidato e creato un nuovo cursore figlio. Ciò, quindi, può accadere in Database che raccolgono statistiche quotidianamente.

- **BIND_MISMATCH** (*The bind metadata does not match the existing child cursor*)

  Quando Oracle compila un nuovo piano di esecuzione per un cursore padre, diverse informazioni vengono archiviate in memoria insieme al cursore figlio sottostante [21]. Questi metadati del cursore contengono, tra gli altri, i parametri dell'ottimizzatore corrente, le statistiche e le identità degli oggetti, il nome, il tipo e la lunghezza delle variabili di collegamento (bind variables).

  Nelle due illustrazioni seguenti (Figure 3.3 e 3.4) è possibile apprezzare la differenza nell'utilizzo di bind variables nella scrittura di una query, in quanto il numero di cursori generati sarà differente.

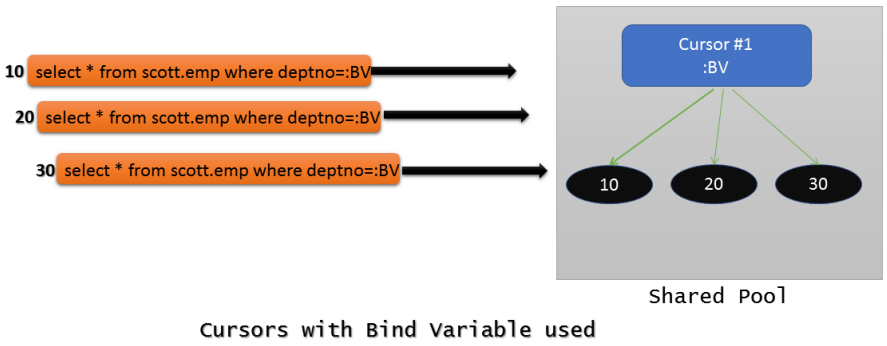

Figura 3.3: Esempio utilizzo bind variables

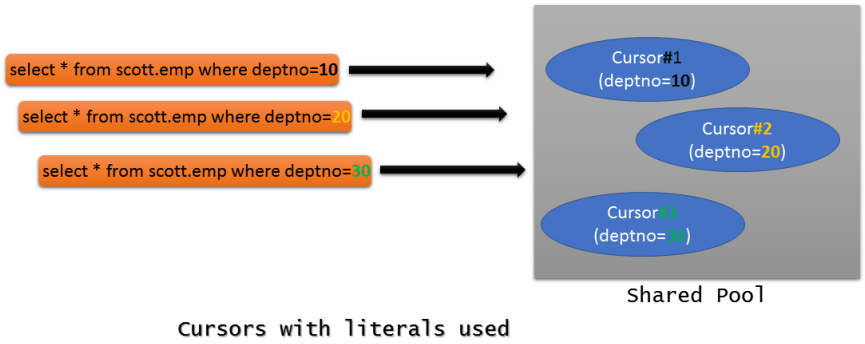

Figura 3.4: Esempio utilizzo literals

In effetti, tutto ciò che interrompe i metadati archiviati invaliderà il cursore e, quindi, provocherà la creazione di un nuovo piano di esecuzione.

- **OPTIMIZER_MISMATCH** (*The optimizer environment does not match the existing child cursor*)
  Nel momento in cui Oracle stacca un nuovo cursore figlio da cursore padre, memorizzerà informazioni che identificano nel modo più completo possibile l'ambiente che caratterizza la compilazione e l'esecuzione di questo piano di esecuzione [22]. I parametri dell'ottimizzatore rappresentano una parte importante di questi metadati del cursore archiviati. Quando lo stesso cursore padre viene avviato di nuovo, il cursore figlio esistente verrà condiviso e riutilizzato, a condizione che tutti i parametri dell'ottimizzatore rimangano intatti. Se, tuttavia, Oracle si rende conto che uno o più valori di quei parametri sono stati modificati, invaliderà il cursore figlio esistente e ne creerà uno nuovo.

- **LANGUAGE_MISMATCH** (*The language handle does not match the existing child cursor*)

  Si verifica a seguito del cambio di lingua (es: *alter session set nls_language = FRENCH*) tra due esecuzioni successive della stessa query [23]. È bene sottolineare che Oracle è sufficientemente intelligente da riconoscere quando la lingua non ha possibilità di incidere sul risultato di un'interrogazione al Database. Ciò significa che non sempre una modifica del linguaggio NLS (Notazione Lineare Strutturata) comporta il cambiamento del piano di esecuzione in quanto, ad esempio, l'ordinamento dei risultati secondo un valore numerico non sortisce nessun effetto.

- **OPTIMIZER_MODE_MISMATCH**

  Questo parametro può alterare radicalmente le caratteristiche del carico di lavoro di esecuzione di una query *SQL* e può essere modificato in maniera simile al precedente. [24] I differenti valori che può assumere sono uno tra le quattro seguenti:

  1. **All rows**, impostazione di default che restituisce il set di risultati completo riducendo al minimo le risorse di elaborazione
  2. **First_rows_n** (con n $\bar{1}$, 10, 100, 1000) ritorna le prime n righe della query
  3. **Rule** utilizza un insieme di regole
  4. **Choose** basato sui costi dove sono disponibili le statistiche

  Per quanto riguarda le prime due si evidenzia una differenza fondamentale nei piani di esecuzione generati, ovvero la presenza di Full Table Scans o di Index Range Scans. Entrambi i metodi vengono utilizzati quando è necessario recuperare tutti i dati, come dal 90% al 100%. Inoltre, se la query non ha una clausola *WHERE* e la tabella non ha un indice cluster, verrà utilizzato il primo,

viceversa il secondo, che permette di avere performance migliori in termini di velocità di esecuzione.

- **BIND_LENGTH_UPGRADEABLE** (*Bind length(s) required for the current cursor are longer than the bind length(s) used to build the child cursor*)
  Quando viene creato un cursore figlio, Oracle sembra pre-allocare uno spazio per la variabile di binding interna, in base a come essa è definita. Se la variabile di bind esistente viene ridefinita e lo spazio è significativamente più grande, verrà staccato un nuovo cursore figlio [25].

- **USE_FEEDBACK_STATS** (*A hard parse is forced so that the optimizer can reoptimize the query with improved cardinality estimates*)
  Nel momento in cui l'ottimizzatore genera un piano di esecuzione, la presenza di statistiche mancanti o obsolete, predicati o operatori complessi può attivare l'ottimizzatore per monitorare la cardinalità delle operazioni nel piano. Una volta completata l'esecuzione, se vi è una differenza significativa tra le cardinalità stimate ed effettive, i valori effettivi vengono archiviati nella SGA, per un uso successivo. Alla successiva esecuzione l'istruzione viene ottimizzata nuovamente utilizzando le cardinalità memorizzate, consentendo così un miglioramento del piano. Il feedback sulla cardinalità è specifico dell'istruzione e va perso se l'istanza viene riavviata o l'istruzione non è più disponibile nel pool condiviso; nelle ultime versioni di Oracle Database questo parametro è stato rinominato come feedback statistico.

Rispetto alle versioni di Oracle Database precedenti è stato aggiunto un attributo *REASON*, strutturato secondo un formato *XML*, nel quale sono contenuti i parametri settati a 'Y' che hanno causato la generazione di un nuovo child. Essendo però la

lettura poco agevole, è stato deciso di effettuare una procedura di manipolazione dei dati, in modo da modificare le informazioni per renderle più organizzate e leggibili.

È stato quindi deciso di creare una tabella più snella e veloce da consultare, contenente alcuni parametri identificativi del record e una colonna riassuntiva, che contiene un elenco dei booleani impostati a 'Y', grazie alla seguente query.

```
01 | SELECT SQL_ID, ADDRESS, CHILD_ADDRESS, CHILD_NUMBER, (
       REASON1 || REASON2 ... REASON63) AS REASON
02 | FROM (SELECT SQL_ID, ADDRESS, CHILD_ADDRESS, CHILD_NUMBER,
03 |           CASE WHEN ANYDATA_TRANSFORMATION = 'Y' THEN '
       ANYDATA_TRANSFORMATION' END REASON1,
04 |           CASE WHEN AUTH_CHECK_MISMATCH = 'Y' THEN '
       AUTH_CHECK_MISMATCH' END REASON2,
05 |           CASE WHEN BIND_EQUIV_FAILURE ='Y' THEN '
       BIND_EQUIV_FAILURE' END REASON3,
06 |           CASE WHEN BIND_LENGTH_UPGRADEABLE ='Y' THEN '
       BIND_LENGTH_UPGRADEABLE' END REASON4,
07 |           CASE WHEN BIND_MISMATCH ='Y' THEN ' BIND_MISMATCH'
       END REASON5
08 |           ...
09 |           CASE WHEN USE_FEEDBACK_STATS ='Y' THEN '
       USE_FEEDBACK_STATS' END REASON62,
10 |           CASE WHEN USER_BIND_PEEK_MISMATCH ='Y' THEN '
                     USER_BIND_PEEK_MISMATCH' END REASON63
11 |       FROM V$SQL_SHARED_CURSOR
12 |       WHERE 1=1
13 |           AND ANYDATA_TRANSFORMATION ='Y' OR
       AUTH_CHECK_MISMATCH ='Y' OR BIND_EQUIV_FAILURE ='Y' OR
       BIND_LENGTH_UPGRADEABLE ='Y' OR BIND_MISMATCH ='Y' OR
14 |           ...
```

```
15 |            OR USE_FEEDBACK_STATS ='Y' OR
         USER_BIND_PEEK_MISMATCH ='Y');
```

In questo modo si è ottenuto il seguente risultato finale più compatto (Figura 3.5).

	SQL_ID	ADDRESS	CHILD_ADDRESS	CHILD_NUMBER	REASON
45	94yj0phdyh01q	b'\x00\x00\x00\x00y\x01\x07\x90'	b'\x00\x00\x00\x00\x84\xc1\x95\xa8'	45	ROLL_INVALID_MISMATCH
46	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x8c\xb8W\x00'	0	OPTIMIZER_MISMATCH
47	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x89\xd5w\x80'	1	OPTIMIZER_MISMATCH
48	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x89\xa9\xf7\xc0'	2	OPTIMIZER_MISMATCH
49	gngtvs38t0060	b'\x00\x00\x00\x00\x8c\xb8^X'	b'\x00\x00\x00\x00\x85Z+\x08'	3	OPTIMIZER_MISMATCH ROLL_INVALID_MISMATCH
50	bqfx5q2jas08u	b'\x00\x00\x00\x00\x85\x01tx'	b'\x00\x00\x00\x00\x85\x01\x0bx'	0	ROLL_INVALID_MISMATCH USE_FEEDBACK_STATS
51	bqfx5q2jas08u	b'\x00\x00\x00\x00\x85\x01tx'	b'\x00\x00\x00\x00\x84\xff,\xc0'	1	ROLL_INVALID_MISMATCH
52	bqfx5q2jas08u	b'\x00\x00\x00\x00\x85\x01tx'	b'\x00\x00\x00\x00\x89\xd1\x04\x10'	2	ROLL_INVALID_MISMATCH
53	bqfx5q2jas08u	b'\x00\x00\x00\x00\x85\x01tx'	b'\x00\x00\x00\x00\x81\xeb\xf4\x10'	3	ROLL_INVALID_MISMATCH
54	bqfx5q2jas08u	b'\x00\x00\x00\x00\x85\x01tx'	b'\x00\x00\x00\x00\x80\xed\xe0X'	4	ROLL_INVALID_MISMATCH
55	3wqb0gax640fv	b'\x00\x00\x00\x00\x85ed@'	b'\x00\x00\x00\x00\x8c\xf8\xdc\xc0'	0	USE_FEEDBACK_STATS

Figura 3.5: Estratto risultato finale

Capitolo 4

ACTIVE_SESSION_HISTORY

In questo capitolo verrà presentato il lavoro svolto sull'ultima vista di sistema presa in analisi, la *V$ACTIVE_SESSION_HISTORY* che contiene uno storico delle sessioni attive, sia quelle in corso che quelle già terminate [26].

Una sessione Oracle è un'entità logica che viene generata nel momento in cui viene effettuato un login nel Database. Una singola connessione può avere 0, 1 o più sessioni stabilite su di essa.

Quando l'istanza si interrompe, la memoria dell'istanza svanisce con essa. Pertanto, Oracle Database archivia le informazioni in una tabella per renderle persistenti. Un campione su dieci viene mantenuto su disco ed è visibile in una vista denominata *DBA_HIST_ACTIVE_SESS_HISTORY*. Il suo utilizzo è simile all'utilizzo della *V$ACTIVE_SESSION_HISTORY*, ma è bene evidenziare che la frequenza di campionamento è ridotta; la storicizzazione dei dati ogni 10 secondi determina sicuramente una perdita di informazioni che, in questa trattazione, non è considerata accettabile.

Molte delle informazioni contenute in questa tabella temporanea sono presenti anche nella *V$SESSION*, che è però sprovvista della colonna *SAMPLE_TIME*, ovvero il tempo di campionamento, che sarà fondamentale nello sviluppo di questo

elaborato.

Per questi motivi si intende proseguire con la *V$ACTIVE_SESSION_HISTORY* in un solco già tracciato, da una parte, dell'Anomaly Detection e dall'altra dall'analisi sulle performance del Database.

Il processo di rilevazione degli outlier è il punto di arrivo e non quello di partenza, come è stato per il capitolo precedente: se prima era una caratteristica intrinseca dei dati in input, ora è possibile apprezzarne i risultati solo a seguito di molteplici elaborazioni e di una rappresentazione visiva dello studio eseguito sulle performance.

Inoltre, si sottolinea un forte collegamento con la *V$SQL_SHARED_CURSOR*, essendo due viste complementari per lo scopo di questa trattazione. Una volta identificati i valori anomali con la *V$ACTIVE_SESSION_HISTORY*, sarà possibile investigare e conoscerne le motivazioni come sviluppato nel Capitolo 3.

Indagando più nello specifico per quanto concerne la parte operativa, il risultato che si è voluto raggiungere in quest'analisi è una visualizzazione grafica del tempo di esecuzione (elapsed time), per ogni istruzione *SQL*, al trascorrere del tempo, ovvero in timestamp successivi. L'obiettivo ultimo è stato quello di identificare eventuali valori anomali, secondo la metodologia già utilizzata nel Capitolo 2 di questa trattazione.

4.1 Data Exploration & Data Manipulation

In questa vista sono visualizzate le attività della sessione campionata nel Database. Contiene snapshots di sessioni di Database attive con una frequenza di campionamento di una volta al secondo.

Una sessione attiva è una sessione in attesa sulla CPU o su qualsiasi evento che non appartiene alla classe di attesa (Idle) al momento del campionamento. Le informazioni vengono scritte in un buffer circolare nella SGA, quindi maggiore è l'attività del Database, minore è il tempo per cui le informazioni rimarranno disponibili; le grandezze sono, dunque, inversamente proporzionali tra loro.

La *V$ACTIVE_SESSION_HISTORY* è, quindi, essenzialmente una tabella dei fatti, che può essere collegata a una serie di dimensioni, al fine di fornire statistiche specifiche per una grande varietà di elementi:

- **Istruzioni SQL**
- **Piani di esecuzione**
- **Oggetti**
- **Eventi di attesa**
- **Sessioni**
- **Moduli**
- **Azioni**
- **Identificatori client**
- **Servizi**

Questo lo rende un modo incredibilmente flessibile per identificare cosa stanno facendo o hanno fatto le sessioni attive.

Una sua rappresentazione si può vedere nella seguente illustrazione (Figura 4.1).

	SAMPLE_ID	SAMPLE_TIME	SAMPLE_TIME_UTC	USECS_PER_ROW	IS_AWR_SAMPLE	SESSION_ID	SESSION_SERIAL#	SESSION_TYPE	FLAGS	USER_ID
0	222265903	2022-06-21 04:37:40.864	2022-06-21 02:37:40	1002217	N	13150	59091	BACKGROUND	16	824
1	222265902	2022-06-21 04:37:39.862	2022-06-21 02:37:39	1002098	N	13150	59091	BACKGROUND	16	824
2	222265901	2022-06-21 04:37:38.860	2022-06-21 02:37:38	1001908	N	13150	59091	BACKGROUND	16	824
3	222265900	2022-06-21 04:37:37.858	2022-06-21 02:37:37	1002134	N	13150	59091	BACKGROUND	16	824
4	222265899	2022-06-21 04:37:36.855	2022-06-21 02:37:36	1002256	N	13150	59091	BACKGROUND	16	824
5	222537113	2022-06-24 08:09:12.075	2022-06-24 06:09:12	1002419	N	23227	26143	FOREGROUND	16	1040
6	222537112	2022-06-24 08:09:11.072	2022-06-24 06:09:11	1003242	N	23227	26143	FOREGROUND	16	1040
7	222537111	2022-06-24 08:09:10.069	2022-06-24 06:09:10	1002032	N	23227	26143	FOREGROUND	16	1040
8	222549813	2022-06-24 11:41:24.732	2022-06-24 09:41:24	1002226	N	158	12009	FOREGROUND	16	1040
9	222549812	2022-06-24 11:41:23.730	2022-06-24 09:41:23	1002636	N	158	12009	FOREGROUND	16	1040
10	222549811	2022-06-24 11:41:22.727	2022-06-24 09:41:22	1002279	N	158	12009	FOREGROUND	16	1040

Figura 4.1: Estratto *V$ACTIVE_SESSION_HISTORY*

A seconda dell'attività del database, ciò porterà alla raccolta di molti dati all'interno del buffer di questa vista, nonostante abbia una quantità limitata di spazio.

Avendo scelto di raccogliere dati ad ogni secondo, è possibile che la stessa query (stesso execution time) venga rilevata più di una volta; in quel caso per identificare il tempo di esecuzione è necessario calcolare la differenza tra massimo e minimo sample time. Nell'eventualità in cui per una coppia di *SQL_ID* e *SQL_EXEC_START* ci sia un solo record, l'elapsed time sarà nullo.

La query ideata per questo scopo è stata la seguente:

```
01 | SELECT SQL_ID, SQL_EXEC_START,
02 |      extract (day from (MAX(SAMPLE_TIME)-MIN(SAMPLE_TIME)))
       *86400
03 |      + extract (hour from (MAX(SAMPLE_TIME)-MIN(SAMPLE_TIME)))
       *3600
04 |      + extract (minute from (MAX(SAMPLE_TIME)-MIN(SAMPLE_TIME)
       ))*60
```

```
05 |      + extract (second from (MAX(SAMPLE_TIME)-MIN(SAMPLE_TIME)
       )) AS ELAPSED_TIME
06 | FROM OPA.V_ASH_HIST
07 | WHERE SQL_ID IN
08 |      (SELECT DISTINCT A.SQL_ID
09 |      FROM (SELECT SQL_ID, COUNT(*) AS CNT_1
10 |           FROM OPA.V_ASH_HIST
11 |           GROUP BY SQL_ID) A
12 |           , (SELECT SQL_ID, COUNT(*) AS CNT_2
13 |           FROM OPA.V_ASH_HIST A
14 |           WHERE A.SQL_EXEC_START is not null
15 |           GROUP BY SQL_ID) B
16 |      WHERE A.SQL_ID = B.SQL_ID
17 |      AND A.CNT_1 = B.CNT_2)
18 | GROUP BY SQL_ID, SQL_EXEC_START
19 | ORDER BY SQL_ID, SQL_EXEC_START ASC
```

Nella seguente immagine (Figura 4.2) si può vedere un esempio del dataset ottenuto, a seguito della modifica dei dati effettuata.

	SQL_ID	SQL_EXEC_START	ELAPSED_TIME
0	3uvgvq7x533zb	2022-06-21 07:19:05	0.000
1	1nr3g021qty72	2022-06-21 06:50:32	0.000
2	ar16uy6gz9189	2022-06-21 06:16:43	0.000
3	dpq6xgt45d91p	2022-06-21 06:15:28	0.000
4	8835b6xt5yywq	2022-06-21 06:08:51	2.004
5	42kxmgu3td4vp	2022-06-21 06:06:06	0.000
6	3uvgvq7x533zb	2022-06-21 06:05:33	0.000
7	1qjnb3j6ja9p4	2022-06-21 06:00:09	4.014
8	gxn33x09rjz5b	2022-06-21 06:00:02	4.013
9	34rmzaznkvpaz	2022-06-21 06:00:03	0.000
10	cxhbbzdfw5445	2022-06-21 05:45:44	0.000
11	g3qd21aq68fd6	2022-06-21 05:37:55	0.000
12	dztqawjzk7q5n	2022-06-21 05:30:01	1.002

Figura 4.2: Estratto rielaborazione *V$ACTIVE_SESSION_HISTORY*

4.2 Data Visualization

Una volta selezionati solo quei valori "interessanti", cioè escluse quelle query con un plot caratterizzato da meno di 20 valori, per le quali non si può apprezzare un trend nel tempo, o con un grafico piatto che è rappresentato solo da una serie di punti con ordinata pari a zero, è stato possibile ottenere una prima visualizzazione grafica soddisfacente.

Infine, in aggiunta a questa prima analisi, è stato sviluppato un modello di Anomaly Detection, già utilizzato precedentemente in questa trattazione (si faccia riferimento al Capitolo 2), basato sul metodo z-scoring.

In questo caso, però, è stato scelto di eseguire un test monolaterale destro, in quanto era importante evidenziare quando il tempo di esecuzione di una query

aumentasse. Per avere un risultato più robusto sono state effettuate molteplici prove con diverse percentuali (95%, 96%, 97%, 98% e 99%), per valutare se e come si modificassero gli esiti.

Al fine di identificare solamente gli outliers più evidenti, ed avere un punto di vista più critico e restrittivo, si è scelto di utilizzare una percentuale pari al 98%. In questo caso sono stati considerati anomali quei valori maggiori della soglia $z = 2.065$. Con questi parametri, sono considerati accettabili, una volta normalizzati secondo i valori della normale standard calcolati, quei valori inferiori alla soglia.

Il codice completo utilizzato per il plotting finale dei risultati ottenuti è il seguente.

```
01 | fig = go.Figure()
02 | inizio = 0
03 | fine = 3
04 | lunghezza = excel_agg['SQL_ID'].nunique()
05 | ripetizioni = 0
06 |
07 | if lunghezza/3 == lunghezza//3:
08 |     ripetizioni = lunghezza//3
09 | else:
10 |     ripetizioni = lunghezza//3 + 1
11 | for contatore in range (0,ripetizioni):
12 |     array = excel_agg['SQL_ID'].unique()[inizio:fine]
13 |     if (array.size == 1):
14 |         fig = make_subplots(rows=1, cols=1, subplot_titles= [
       array[0]])
15 |     elif (array.size == 2):
16 |         fig = make_subplots(rows=2, cols=1, subplot_titles= [
       array[0], array[1]])
17 |     else:
```

```
18 |          fig = make_subplots(rows=3, cols=1, subplot_titles= [
       array[0], array[1], array[2]])
19 |      inizio = fine
20 |      fine = fine + 3
21 |      riga = 0
22 |      stringa = ""
23 |      color = ["blue","red","green"]
24 |      for indice in array:
25 |          riga += 1
26 |          fig.add_trace(go.Scatter(x = excel_agg.loc[excel_agg[
       'SQL_ID'] == indice]['SQL_EXEC_START'], y = excel_agg.loc[
       excel_agg['SQL_ID'] == indice]['ELAPSED_TIME'], line =
       dict(color=color[riga-1]), visible=True), row = riga, col
       = 1)
27 |          fig.add_trace(go.Scatter(x = excel_agg.loc[excel_agg[
       'SQL_ID'] == indice][excel_agg.loc[excel_agg['SQL_ID'] ==
       indice]['OUTLIER'] == True]['SQL_EXEC_START'], y =
       excel_agg.loc[excel_agg['SQL_ID'] == indice][excel_agg.loc
       [excel_agg['SQL_ID'] == indice]['OUTLIER'] == True]['
       ELAPSED_TIME'], mode="markers", marker_size = 10,
       marker_color = ["black"]), row = riga, col = 1)
28 |          stringa = stringa + "_" + indice
29 |      fig.show()
30 |      plotly.offline.plot(fig, filename ='C:/Users/Mediamente/
       Prove Jupyter/V$ACTIVE_SESSION_HISTORY/'+stringa+'.html',
       auto_open=False)
31 |      fig.data = []
```

Da queste righe di codice si può capire che sono state create delle figure con al proprio interno tre tracce di tre colori diversi (verde rosso e blu), uno per ogni stringa identificativa di una query.

I grafici ottenuti sono poi stati modificati andando ad evidenziare i punti che rappresentavano dei valori anomali con dei pallini neri, aggiungendo un'ulteriore traccia su di essi. Questo ha permesso di usufruire di un modo molto rapido per una visualizzazione grafica della situazione in esame.

Infine, è stata prevista la possibilità di visualizzare direttamente i plot ottenuti e di scaricare il relativo file *HTML* per una più comoda e semplice visualizzazione futura.

Nelle seguenti illustrazioni si possono apprezzare alcune raffigurazioni dei risultati ottenuti, che rappresentano l'andamento nel corso del tempo del tempo di esecuzione di sei *SQL_ID* differenti (da Figura 4.3 a Figura 4.8).

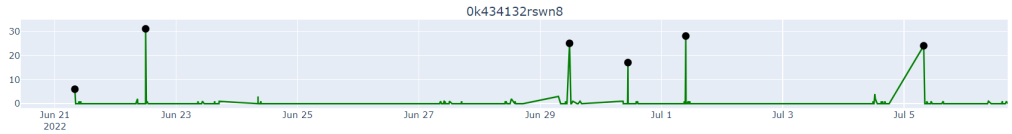

Figura 4.3: Plot 1

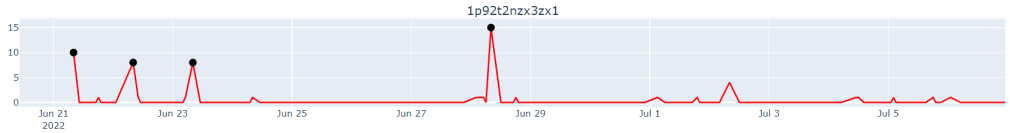

Figura 4.4: Plot 2

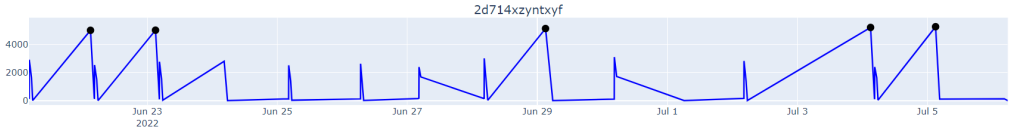

Figura 4.5: Plot 3

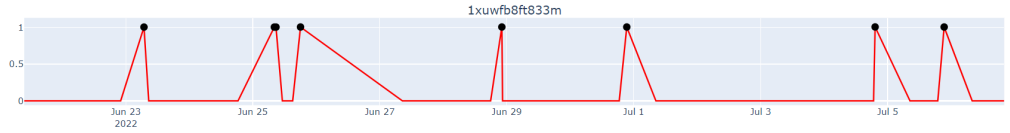

Figura 4.6: Plot 4

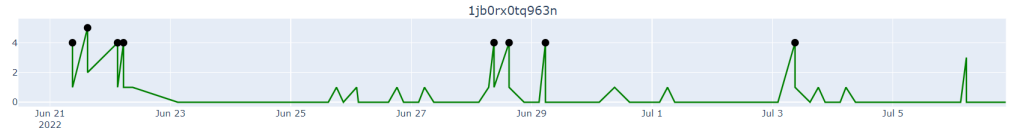

Figura 4.7: Plot 5

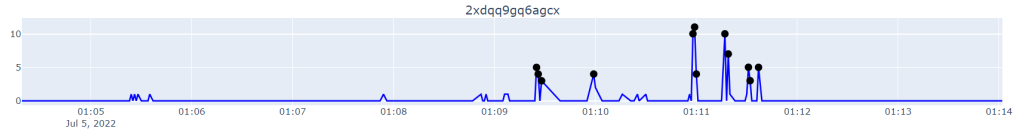

Figura 4.8: Plot 6

Capitolo 5

CONCLUSIONI

In questo capitolo conclusivo saranno esposti i risultati ottenuti a seguito degli obiettivi iniziali che erano stati fissati, che hanno reso necessario l'inizio di questo lavoro di tesi. Verrà poi posta la giusta attenzione verso ciò che di buono è stato realizzato e come questo possa essere proiettato all'interno di una realtà aziendale ben strutturata.

Mettendo assieme le analisi effettuate separatamente in ogni capitolo, si possono identificare due binari separati, da cui si sono sviluppate le relative analisi in merito.

Il Capitolo 2 si concentra maggiormente sulla valutazione delle performance di un Database, a seguito del continuo controllo di parametri rilevanti, del loro raggruppamento per determinare macro gruppi con comportamenti affini e la rilevazione di valori fuori dalla norma.

I successivi, ed ultimi due, di questa trattazione, invece, permettono di capire quando una query genera un nuovo piano di esecuzione, e di conseguenza viene staccato un nuovo child; questo può portare, sotto un punto di vista pratico, ad una possibile anomalia, il cui rilevamento è determinato dal monitoraggio del tempo di esecuzione del piano. Successivamente, valuta se questo abbia generato un peggioramento sul piano delle performance, in termini di tempo, e infine ne determina

le motivazioni alla base.

Mi sembra doveroso porre nuovamente la giusta attenzione alla tematica del Real Time che, nonostante non se ne parli nella parte più operativa di questo elaborato, risulta il perno centrale e il vero valore aggiunto di tutto il lavoro svolto. Non tutte le aziende fanno studi di ricerca in questa direzione, ma si concentrano ancora sul Near Real Time, che prevede l'accettazione dell'elaborazione di una certa attività anche nell'ordine di grandezza dei minuti anziché dei secondi. Un passaggio da una visione all'altra comporterebbe un notevole miglioramento dell'efficienza operativa in molte realtà esistenti.

Risulta di notevole importanza ciò che è stato realizzato sotto diversi punti di vista. Senza dubbio, i primi utilizzatori del frutto del mio lavoro sarebbero i dipendenti di Mediamente Consulting stessi. Confrontandomi continuamente con loro è scaturita più volte la necessità di un modello che facilitasse e velocizzasse il loro compito di consulenti e li potesse aiutare nei momenti di decision making. Si passerebbe dunque da un metodo di lavoro reattivo ad uno proattivo, con il chiaro intento di prevenire potenziali eventi futuri, in modo da pianificare anticipatamente le corrette azioni da svolgere e intervenire più velocemente.

Inoltre, in maniera indiretta ma non meno importante, i clienti di questa società, che potrebbero interfacciarsi con un servizio più rapido ed efficiente, aumenterebbero così il loro grado di soddisfazione per il lavoro di assistenza.

Nel complesso gli obiettivi proposti sono stati raggiunti da questo progetto, infatti il risultato di Anomaly Detection in Real Time genera una procedura soddisfacente e di alto valore, sia dal punto di vista aziendale che dal punto di vista del cliente. In aggiunta, attraverso la misurazione delle metriche di performance di un Database, è possibile mantenere un continuo monitoraggio del funzionamento dello stesso, notare eventuali criticità ed, in tal caso, agire in maniera più tempestiva rispetto a prima.

Al termine del lavoro svolto, per quanto riguarda gli aspetti più tecnici dei risultati

raggiunti, possono essere valutati i maggiori punti di forza. Questi sono sicuramente legati al buon livello di dati di partenza, all'alta flessibilità e riproducibilità delle procedure utilizzate e alla velocità di visualizzazione delle problematiche. Infatti, avendo previsto una struttura flessibile e replicabile, viene aumentata notevolmente l'importanza intrinseca dell'opera; sarà infatti possibile riutilizzare questo progetto e modellarlo in base alle esigenze richieste. Nonostante le attività operative che interessano questo lavoro di tesi fossero già svolte manualmente dal team di dipendenti appartenenti alla Business Unit infrastrutturale, ritengo che il vero punto di forza sia la rapidità di visualizzazione di determinati contenuti.

Queste sono tutte analisi molto preziose che saranno, con alta probabilità, eseguite in un prossimo futuro. Infatti, tengo a sottolineare che il progetto non è stato concluso con questa tesi, ma c'è la volontà e l'intenzione di portarlo avanti e implementarlo nel framework esistente per agevolare le analisi di routine quotidiane.

Si può anche considerare, come fonte di potenziale ulteriore analisi e miglioramento, l'analisi sopra illustrata come un fattore di forza del lavoro svolto in questo progetto di ricerca. Infatti, i risultati ottenuti possono essere visti come una componente di un disegno molto più ampio inserito all'interno della società di consulenza.

Next step Si è pensato ad alcune possibili migliorie da implementare a livello pratico che potrebbero rendere questo elaborato maggiormente completo e robusto. Bisogna tenere in considerazione la facoltà di estendere tutte le analisi effettuate anche ad altre possibili viste di sistema che Oracle fornisce, alcune delle quali sono state citate lungo il corso della trattazione. Inoltre è possibile affiancare ai modelli di Anomaly Detection utilizzati altre soluzioni che potrebbero essere ritenute più adatte o maggiormente performanti.

Risultano poi fondamentali e imprescindibili dal resto il fatto, ad esempio, che il modello debba essere inserito nel framework esistente e che si possa generare un cruscotto dedicato all'Anomaly Detection, particolare ancora non presente. Nello

specifico potrebbe essere introdotta l'intuizione sviluppata nel contesto del Clustering Gerarchico e il relativo sviluppo del dendogramma, integrando i mezzi essenziali per la visualizzazione delle metriche secondo un livello di aggregazione variabile, potendo modificare la granularità dei dati a proprio piacimento.

Bibliografia

[1] Veronica Basaglia. *Real Time Analytics: cosa sono e perché sono utili*. [Online; in data 11 Febbraio 2020]. 2020. URL: https://www.naxa.ws/real-time-analytics/ (cit. a p. vi).

[2] Digital Coach. *Real Time Analytics: cos'è e come funziona*. 2022. URL: https://www.digital-coach.com/it/real-time-analytics/ (cit. a p. vii).

[3] Irene Di Deo. *Big Data: i Real Time Analytics dalla teoria alla pratica*. [Online; in data 19 Novembre 2018]. 2018. URL: https://blog.osservatori.net/it_it/real-time-analytics (cit. a p. vii).

[4] Oracle Corporation. *Oracle Database Documentation*. 2022. URL: https://docs.oracle.com/en/database/oracle/oracle-database/ (cit. a p. 1).

[5] Saverio Mola. *L'architettura di Oracle*. [Online; in data 16 Ottobre 2006]. 2006. URL: https://www.html.it/pag/16910/l-architettura-di-oracle/ (cit. a p. 1).

[6] Abu Fazal Abbas. *Oracle 12c: SEED (PDB$SEED) pluggable database is in unusable state? Here's how you can recover/recreate it*. [Online; in data 21 Marzo 2016]. 2016. URL: https://blog.toadworld.com/oracle-12c-seed-pdbseed-pluggable-database-is-in-unusable-state-heres-how-you-can-recover/recreate-it (cit. a p. 2).

[7] Vito Lavecchia. *Che cosa sono le proprietà ACID dei DBMS in informatica?* URL: https://vitolavecchia.altervista.org/che-cosa-sono-le-proprieta-acid-dei-dbms-in-informatica/ (cit. a p. 6).

[8] Donald Burleson. *Oracle Database Metric v$ Views*. 2017. URL: http://www.dba-oracle.com/oracle10g_tuning/t_Oracle_10g_sysmetric_v$_views.htm (cit. a p. 19).

[9] Donald Burleson. *Oracle v$sysmetric_history*. 2017. URL: http://www.dba-oracle.com/oracle10g_tuning/t_10g_v$sysmetric_history.htm (cit. a p. 19).

[10] Donald Burleson. *v$sysmetric and v$sysmetric_history*. 2017. URL: http://www.dba-oracle.com/oracle10g_tuning/t_v$sysmetric_history_scripts.htm (cit. a p. 20).

[11] Muhammad Asad Iqbal Khan. *Anomaly Detection with Isolation Forest and Kernel Density Estimation*. [Online; in data 1 Gennaio 2022]. 2022. URL: https://machinelearningmastery.com/anomaly-detection-with-isolation-forest-and-kernel-density-estimation/ (cit. a p. 21).

[12] Philip Wilkinson. *Univariate Outlier Detection in Python*. [Online; in data 12 Agosto 2021]. 2021. URL: https://towardsdatascience.com/univariate-outlier-detection-in-python-40b621295bc5 (cit. a p. 24).

[13] Domenico Soriano. *Che cosa è la matrice di correlazione? come si costruisce in Python?* [Online; in data 20 Settembre 2021]. 2021. URL: https://www.domsoria.com/2021/09/che-cosa-e-la-matrice-di-correlazione-come-si-costruisce-in-python/ (cit. a p. 30).

[14] Giacomo Saver. *Matrice di correlazione: cos'è e come si usa per investire*. [Online; in data 27 Maggio 2015]. 2015. URL: https://www.segretibancari.com/matrice-di-correlazione/ (cit. a p. 30).

[15] Melissa Bierly. *12 Python Data Visualization Libraries to Explore for Business Analysis*. [Online; in data 22 Ottobre 2021]. 2021. URL: https://mode.com/blog/python-data-visualization-libraries/#Plotly (cit. a p. 32).

[16] Plotly. *Line Charts in Python*. 2022. URL: https://plotly.com/python/line-charts/ (cit. a p. 32).

[17] Mirella Castigli. *Cluster analysis: cos'è, come funziona ed esempi*. [Online; in data 22 Ottobre 2021]. 2021. URL: https://www.bigdata4innovation.it/data-science/data-mining/cluster-analysis/ (cit. a p. 37).

[18] Cristiano Casadei. *Clustering gerarchico*. [Online; in data 10 Luglio 2019]. 2019. URL: https://www.developersmaggioli.it/blog/clustering-gerarchico/ (cit. a p. 39).

[19] The SciPy community. *scipy.cluster.hierarchy.linkage*. 2022. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html (cit. a p. 41).

[20] DIANAROBETE. *Why The Child Cursor Is Not Reused: ROLL_INVALID_MISMATCH*. [Online; in data 5 Settembre 2018]. 2018. URL: http://dbaparadise.com/2018/09/why-the-child-cursor-is-not-reused-roll_invalid_mismatch/ (cit. a p. 46).

[21] Mohamed Houri. *Why my execution plan has not been shared - Part III*. [Online; in data 25 Settembre 2016]. 2016. URL: https://blog.toadworld.com/why-my-execution-plan-has-not-been-shared-part-iii (cit. a p. 47).

[22] Mohamed Houri. *Why my execution plan has not been shared - Part IV*. [Online; in data 5 Maggio 2017]. 2017. URL: https://blog.toadworld.com/2017/05/05/why-my-execution-plan-has-not-been-shared-part-iv (cit. a p. 48).

[23] Mohamed Houri. *Why my execution plan has not been shared - Part V*. [Online; in data 13 Giugno 2017]. 2017. URL: `https://blog.toadworld.com/2017/06/13/why-my-execution-plan-has-not-been-shared-part-v` (cit. a p. 49).

[24] Ian Fergusson. *The Oracle Optimizer: Mode Considerations*. [Online; in data Ottobre 2015]. 2015. URL: `https://www.cintra.com/blog-posts/the-oracle-optimizer-mode-considerations/` (cit. a p. 49).

[25] Craig Shallahamer. *How To Create LOTS Of Oracle Database Library Cache Child Cursors*. [Online; in data 17 Settembre 2012]. 2012. URL: `https://blog.orapub.com/20120917/how-to-create-lots-of-oracle-database-library-cache-child-cursors.html` (cit. a p. 50).

[26] Donald Burleson. *v$active_session_history*. 2017. URL: `http://www.dba-oracle.com/oracle10g_tuning/t_v$active_session_history.htm` (cit. a p. 53).