Esercitazione 5

Ottimizzatore di Oracle

Obiettivo dell'esercitazione

Calcolare il piano di esecuzione per alcune query SQL analizzando i seguenti aspetti:

1. metodo di accesso alle tabelle
2. metodo di join
3. ordine in cui vengono eseguite le operazioni
4. uso di indici definiti dall'utente.

Struttura della base di dati

La base di dati di riferimento è composta da 3 tabelle (**EMP**, **DEPT** e **SALGRADE**).
Di seguito viene riportato lo schema delle tabelle ed alcuni record di esempio. Il contenuto delle tabelle riportato è solo di esempio, in quanto nella realtà la base di dati è composta da un numero elevato di record.

*Table **EMP***

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	COZZA MARIA	PROFESSOR	0	09-JUN-81	1181		126
2	ECO LUIGI	PHDSTUDENT	0	09-JUN-81	1360		189
3	CORONA CLARA	PHDSTUDENT	2	09-JUN-81	624		15

*Table **DEPT***

DEPTNO	DNAME
1	INFORMATION
2	CHAIRMANSHIP
3	ENVIRONMENT
4	PHYSICS

*Table **SALGRADE***

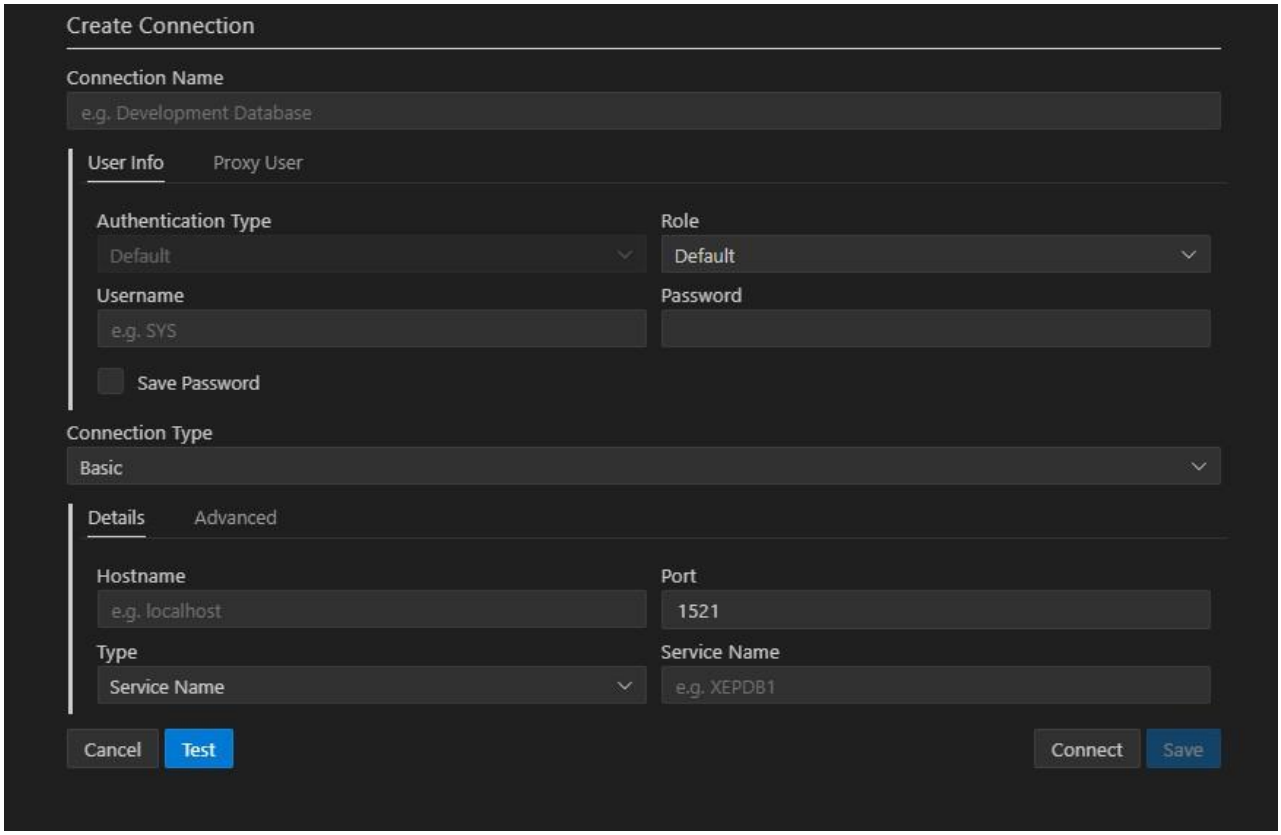
GRADE	LOSAL	HISAL
1	478	1503
2	661	1346
3	489	1358
4	942	1320

Passi preliminari per lo svolgimento dell'esercitazione.

Opzione 1. Creare una connessione Oracle

Per creare una connessione è possibile utilizzare **Oracle SQL Developer** nella sua estensione per VS Code. Seguire le istruzioni a questo link.

Accedendo alla tab dell'estensione, premere **New Connection** per visualizzare la seguente schermata.



Connettersi al database con le seguenti credenziali:

- Name: **nome scelto dall'utente** (esempio: lab5)
- Username: **La propria matricola, escludendo la lettera S** (esempio: S123456 -> 123456)
- Password: **DBDMG23**
- Hostname: **mp1.polito.it**
- Port: **8002**
- SID: **FREE**
- Lasciare tutto il resto come *configurazione di default*

Se si riscontrano problemi di connessione, utilizzare l'hotspot personale anziché le reti polito/eduroam.

Se si sceglie questa opzione non è necessario importare i dati manualmente, è sufficiente utilizzare lo script SCHEMA_ALL.sql.

Opzione 2. Oracle FreeSQL

Utilizzare Oracle FreeSQL come indicato nel Laboratorio 1.

Se si sceglie questa opzione, utilizzare la tabella `EMP_sampled.sql` anziché durante l'intera esercitazione.

Materiale disponibile

Sono disponibili alcuni script contenenti istruzioni SQL per svolgere le seguenti operazioni:

1. Generare la base di dati
2. Creare un indice su un campo della base di dati
3. Calcolare le statistiche per la base di dati

Gli script sono disponibili sul sito web del corso, alla sezione "Esercitazioni di Laboratorio" (laboratorio 5) negli archivi *scripts (script generazione basi dati, script SQL utili).*

Gli script possono essere caricati e eseguiti utilizzando il pulsante in alto a Destra.

Per visualizzare le statistiche sugli indici, eseguire lo script *show_indexes.sql* (oppure copiare il contenuto dello script e incollarlo come comando SQL).

Definizione dell'ambiente di analisi

All'inizio della sessione di lavoro è necessario svolgere i seguenti passi:

1. Generare la base di dati su cui lavorare presenti nell'archivio *Lab5_Data.zip*, a seconda del metodo scelto:
   - SQL developer su server con connessione a server PoliTO: lanciare solamente SCHEMA_ALL.sql, che copia le tabelle già disponibili nel sistema e crea gli indici necessari.
   - Connessione a server generico: eseguire gli script *DEPT.sql*, *EMP.sql* e *SALGRADE.sql*
   - Oracle FreeSQL: come prima, ma si ricorda di utilizzare lo script *EMP_sampled.sql* anziché l'intero *EMP.sql*.

2. Calcolare le statistiche delle tabelle usando lo script *comp_statistics_tables.sql* Questa operazione è necessaria per aggiornare le statistiche sulle tabelle (es. numero record, distribuzione attributi, …). Le statistiche generate verranno utilizzate dall'ottimizzatore per creare i piani di esecuzione. È sufficiente un'esecuzione dello script per tutta l'esercitazione.

3. Controllare che non siano presenti altri indici riguardanti le 3 tabelle (EMP, DEPT, SALGRADE), oltre a quelli di sistema (il cui nome inizia con SYS). In particolare, usare lo script *show_indexes.sql.*

4. Cancellare gli indici (non di sistema) trovati al passo 3: **DROP INDEX *nomeindice*;**

Calcolo del piano di esecuzione per una query

Per ottenere il piano di esecuzione di una data query è necessario eseguire le seguenti operazioni:

- **Aggiungere il prefisso "EXPLAIN PLAN FOR"**
- **Inserire la query da analizzare, formattata in singola riga**
- **Aggiungere la query "SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);"**

Esempio:

```
EXPLAIN PLAN FOR SELECT * FROM EMP;
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);
```

Verrà mostrato il piano di esecuzione con il seguente formato:

```
PLAN_TABLE_OUTPUT
------------------------------------------------------------------------------------
Plan hash value: 615168685

-------------------------------------------------------------------------------
| Id  | Operation          | Name | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |      |  5068 |   326K|   141   (0)| 00:00:01 |
|*  1 |  HASH JOIN         |      |  5068 |   326K|   141   (0)| 00:00:01 |
|   2 |   TABLE ACCESS FULL| DEPT |   507 | 10647 |     3   (0)| 00:00:01 |
|*  3 |   TABLE ACCESS FULL| EMP  |  5078 |   223K|   138   (0)| 00:00:01 |
-------------------------------------------------------------------------------
```

OPERATION: operazione da eseguire sulle tabelle/indici.
ROWS: dimensione tabella/indice.
COST: costo dell'operazione, valore stimato proporzionale alle risorse utilizzate (CPU, I/O, memoria). Il costo si riferisce ad uno specifico nodo dell'albero algebrico ed è **cumulativo**. Ciò significa che il costo di un nodo include, oltre al costo dell'operazione considerata, i costi di tutti i suoi nodi figli.

Nella foto di esempio viene eseguita una hash join tra le due tabelle DEPT ed EMP. Il costo dell'accesso alla tabella EMP è 138, quello della tabella DEPT è 3. Il nodo di hash join tra le due tabelle ha costo cumulativo pari a 141 (138+3). La select ha costo cumulativo 141 perché non introduce operazioni aggiuntive dopo la hash join.

Tipi di operazioni che si possono trovare nel piano di esecuzione:

- **JOIN, GROUP BY, TABLE ACCESS, INDEX SCAN**

- **Access Predicates**: indicano una o più condizioni che devono soddisfare i record per poter essere selezionati. Vengono usati su dati **indicizzati** (ordinati). Permettono di specificare il **range (start, stop)** di record ordinati che soddisfa la condizione.
  E' possibile trovare questo nodo in una join (specifica le condizioni di join, come nell'immagine di esempio) o nelle operazioni di filtro sugli attributi di una tabella (condizioni WHERE).
- **Filter Predicates**: indicano una o più condizioni che devono soddisfare i record per poter essere selezionati. A differenza degli access predicates l'operazione di filter viene eseguita man mano che si scorrono i record ordinati. Se sono presenti sia access sia filter predicates, i primi specificano un range di record indicizzati, i secondi permettono di filtrare in quel range.
  E' possibile trovare questo nodo nelle operazioni di filtro sugli attributi di una tabella (condizioni WHERE).

Comandi utili

- Per leggere quali campi compongono una tabella: **DESCRIBE NomeTabella;**

- Per creare un indice su un campo di una tabella:
  CREATE INDEX NomeIndice ON NomeTabella(NomeCampo);

- Per aggiornare le statistiche relative ad un indice esistente: **ANALYZE INDEX NomeIndice COMPUTE STATISTICS;**

- Per rimuovere un indice:
  DROP INDEX NomeIndice;

- Per visualizzare l'elenco degli indici relativi ad una tabella:
  SELECT INDEX_NAME FROM USER_INDEXES
  WHERE table_name='nome tabella in maiuscolo';

- Per visualizzare le statistiche relative agli indici:
  SELECT USER_INDEXES.INDEX_NAME as INDEX_NAME, INDEX_TYPE,
  USER_INDEXES.TABLE_NAME, COLUMN_NAME||'('||COLUMN_POSITION||')' as
  COLUMN_NAME, BLEVEL, LEAF_BLOCKS, DISTINCT_KEYS,
  AVG_LEAF_BLOCKS_PER_KEY, AVG_DATA_BLOCKS_PER_KEY,
  CLUSTERING_FACTOR
  FROM user_indexes, user_ind_columns
  WHERE user_indexes.index_name=user_ind_columns.index_name and
  user_indexes.table_name=user_ind_columns.table_name;

- Per visualizzare le statistiche relative alle tabelle:
  SELECT TABLE_NAME, NUM_ROWS, BLOCKS, EMPTY_BLOCKS, AVG_SPACE,
  CHAIN_CNT, AVG_ROW_LEN
  FROM USER_TABLES;
- Per visualizzare le statistiche sugli attributi delle tabelle:

```
SELECT COLUMN_NAME, NUM_DISTINCT, NUM_NULLS, NUM_BUCKETS,
DENSITY FROM USER_TAB_COL_STATISTICS
WHERE TABLE_NAME = 'NomeTabella' ORDER BY COLUMN_NAME;
```

- Per visualizzare gli istogrammi (distribuzione valori degli attributi):
  ```
  SELECT *
  FROM USER_HISTOGRAMS;
  ```

Query da analizzare

Le seguenti query dovranno essere analizzate durante l'esercitazione, eseguendo i passi:

1. espressione algebrica ad albero della query
2. piano di esecuzione di Oracle della query originale senza strutture secondarie
3. Per le query da #4 a #6, scegliere una o più strutture fisiche accessorie per migliorare le prestazioni dell'interrogazione.

Richiamo alla struttura delle tabelle

EMP (EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
DEPT (DEPTNO, DNAME, LOC)
SALGRADE (GRADE, LOSAL, HISAL)

Query #1

```
SELECT *
FROM emp, dept
WHERE emp.deptno = dept.deptno AND emp.job = 'ENGINEER';
```

Cambiare l'obiettivo di ottimizzazione dalla modalità ALL ROWS (best throughput) alla modalità FIRST_ROWS (best response time) attraverso l'uso di hint **(/*+ FIRST_ROWS(n) */)**. n è una variabile numerica intera che può assumere valori maggiori o uguali a 1. Assegnare diversi valori ad n e verificare come variano il piano d'esecuzione e i costi delle diverse operazioni.

ALL ROWS: piano esecuzione ottimizzato in modo da minimizzare il tempo di esecuzione di tutta la query.

FIRST_ROWS(n): piano esecuzione ottimizzato in modo da minimizzare il tempo di esecuzione per i primi n record del risultato.

```
SELECT /*+ FIRST_ROWS(n) */ *
FROM emp, dept
WHERE emp.deptno = dept.deptno AND emp.job = 'ENGINEER';
```

Query #2

Confrontare i costi di hash join e nested loop usando l'hint USE / NO_USE_HASH

```
SELECT /*+ NO_USE_HASH(e d) */ d.deptno, AVG(e.sal)
```

```
FROM emp e, dept d
WHERE d.deptno =
e.deptno GROUP BY
d.deptno;
```

Query #3

Disabilitare il metodo hash join mediante l'uso di hint (/*+ NO_USE_HASH(e d) */)

```
SELECT /*+ NO_USE_HASH(e d) */ ename, job, sal, dname
FROM emp e, dept d
WHERE e.deptno =
d.deptno AND NOT EXISTS
    (SELECT * FROM salgrade WHERE e.sal = hisal);
```

Query #4

Si definiscano una o più strutture secondarie (indici) che permettano l'ottimizzazione della seguente query. Si analizzi con particolare attenzione il cambiamento nel piano di esecuzione creando due indici sugli attributi interessati dall'interrogazione.

```
select avg(e.sal) from
emp e
where e.deptno < 10 and e.sal > 100 and e.sal < 200;
```

Query #5

Si definiscano una o più strutture secondarie (indici) che permettano l'ottimizzazione della seguente query:

```
select dname from
dept where deptno in (select
deptno
                    from emp
                    where job = 'PHILOSOPHER');
```

Query #6

Si definiscano una o più strutture secondarie (indici) che permettano l'ottimizzazione della seguente query (rimuovere eventuali indici già esistenti per confrontare le performance della query con e senza indici):

```
select e1.ename, e1.empno, e1.sal, e2.ename, e2.empno, e2.sal from emp
e1, emp e2
where e1.ename <> e2.ename and e1.sal < e2.sal and
e1.job = 'PHILOSOPHER' and e2.job = 'ENGINEER';
```