

# Lab03

martedì 8 novembre 2022 11:50



DSTBD\_La...  
ITA

## Data Science e Tecnologie per le Basi di Dati

### Esercitazione di laboratorio n. 3

#### Viste materializzate e trigger

La finalità di questa esercitazione consiste nel definire delle viste materializzate ritenute utili per rispondere velocemente a frequenti interrogazioni del data warehouse.  
Le viste create devono poi essere mantenute aggiornate opportunamente, gestendo eventuali modifiche apportate sulle tabelle del data warehouse di partenza.

#### 1. Configurazione

Aprire il programma Oracle SQL Developer e creare una connessione al database Oracle come fatto nella prima esercitazione di laboratorio.

Per autenticarsi inserire i seguenti parametri.

- Nome utente: bdati [scegliere un valore compreso tra 1-100]
- Password: orac [scegliere un valore compreso tra 1-100]
- Nome host: 130.192.27.4
- Porta: 1521
- SID: xe

Ad esempio, collegandosi dalla macchina numero 23 del laboratorio, usare come username **bdati23** e come password **orac23**.

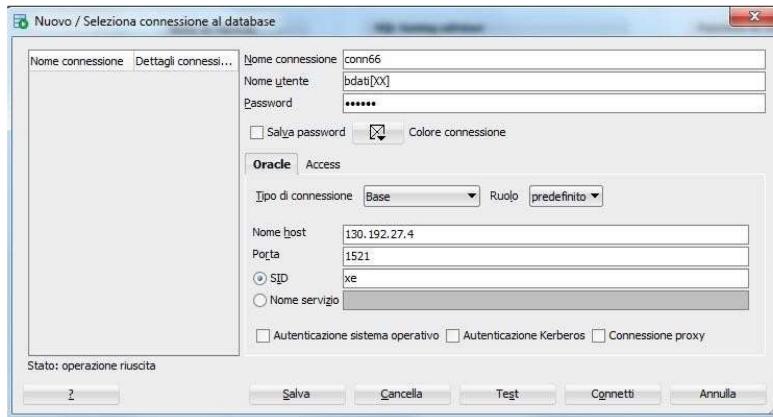


Figura 1 – Connessione al database

Popolare ora il database con le tabelle del data warehouse utilizzato nella prima esercitazione (scaricare lo zip sul sito del corso). Assicurarsi che il proprio database non contenga già le tabelle (FATTI, LUOGO, TARIFFA, TEMPO). Nel caso esistano già è necessario rimuoverle. Importare quindi i file scaricati in formato csv.

## 2. Creazione e aggiornamento della vista materializzata con l'uso di CREATE MATERIALIZED VIEW in ORACLE

### 2.1. Creazione e aggiornamento vista materializzata

**Esercizio 1.1** Partendo dal data warehouse descritto nella prima esercitazione di laboratorio (e il cui schema logico è riportato nella tabella in Figura 2), definire due viste materializzate utili per ridurre i tempi di risposta di almeno tre delle 6 query riportate di seguito.

Tabelle	Descrizione		
<b>DWABD.TEMPO</b> (ID_TEMPO DATA GIORNO MESE ANNO <b>PRIMARY KEY(ID_TEMPO)</b> );	NUMBER DATE VARCHAR VARCHAR NUMBER	NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Dimensione tempo  Cardinalità: 30 tuple
<b>DWABD.TARIFFA</b> (ID_TAR TIPO_TARIFFA <b>PRIMARY KEY(ID_TAR)</b> );	NUMBER VARCHAR	NOT NULL, NOT NULL,	Dimensione tariffa  Cardinalità: 7 tuple
<b>DWABD.LUOGO</b> (ID_LUOGO CITTA PROVINCIA REGIONE <b>PRIMARY KEY(ID_LUOGO)</b> );	NUMBER VARCHAR VARCHAR VARCHAR	NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Dimensione luogo (località)  Cardinalità: 1500 tuple
<b>DWABD.FATTI</b> (ID_TEMPO ID_TAR ID_LUOGO_CHIAMANTE ID_LUOGO_CHIAMATO PREZZO CHIAMATE  <b>PRIMARY KEY(ID_TEMPO, ID_TAR, ID_LUOGO_CHIAMANTE, ID_LUOGO_CHIAMATO)</b> , FOREIGN KEY(ID_TEMPO) REFERENCES TEMPO(ID_TEMPO), FOREIGN KEY(ID_TAR) REFERENCES TARIFFA(ID_TAR), FOREIGN KEY(ID_LUOGO_CHIAMANTE) REFERENCES LUOGO(ID_LUOGO), FOREIGN KEY(ID_LUOGO_CHIAMATO) REFERENCES LUOGO(ID_LUOGO) );	NUMBER NUMBER NUMBER NUMBER NUMBER NUMBER	NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL, NOT NULL,	Tabella dei fatti  Cardinalità: 7809 tuple

Figura 2 – Tabelle del data warehouse

**QUERY SQL:**

1. Selezionare l'incasso totale per ogni tipo tariffa e per ogni mese dell'anno 2003. Selezionare inoltre l'incasso totale complessivo, l'incasso totale per ogni tipo di tariffa indifferentemente dal mese, e l'incasso totale per ogni mese indifferentemente dal tipo di tariffa.
2. Per ogni mese, selezionare il numero di chiamate totali e l'incasso totale. Utilizzando la funzione RANK(), associare ad ogni mese un numero che identifica la posizione del mese all'interno dei mesi in funzione dell'incasso totale effettuato (1 per il mese che ha incassato di più, 2 per il secondo mese, ecc.)
3. Selezionare per ogni mese dell'anno 2003 il numero di chiamate totali. Utilizzando la funzione RANK(), associare ad ogni mese un numero che identifica la posizione del mese all'interno dei vari mesi dell'anno 2003 in funzione del numero di chiamate totali (1 per il mese con più telefonate, 2 per il secondo mese, ecc.)
4. Separatamente per tariffa, selezionare l'incasso totale ottenuto nel mese di Luglio dell'anno 2003.
5. Selezionare per ogni mese, l'incasso del mese e l'incasso cumulativo dall'inizio dell'anno.
6. Considerare l'anno 2003. Separatamente per tariffa e mese, analizzare: i) l'incasso totale, ii) la percentuale dell'incasso rispetto all'incasso totale considerando tutte le tariffe telefoniche, iii) la percentuale dell'incasso rispetto all'incasso totale considerando tutti i mesi

**Esercizio 1.2** Dopo aver creato le viste materializzate, assicurarsi che queste siano opportunamente aggiornate quando qualche modifica avviene sulla base di dati di partenza. Quali tabelle vanno monitorate per aggiornare di conseguenza le viste create?

**STEP A:** Provare a modificare il contenuto della tabella FATTI nel seguente modo:

```
insert into fatti(id_tempo, id_tar, id_luogo_chiamante, id_luogo_chiamato, prezzo, chiamate)
values(8,1,558,752,150,40000)

insert into fatti(id_tempo, ID_TAR, id_luogo_chiamante, id_luogo_chiamato, prezzo, chiamate)
values(1,6,558,752,100,100)
```

**STEP B:** Verificare adesso il nuovo contenuto delle viste materializzate, aggiornandole con il seguente comando:

```
BEGIN
DBMS_SNAPSHOT.REFRESH('Nome_Vista');
END;
```

Come sono cambiate le due viste?

**Esercizio 1.3 (Opzionale)** Provare a eseguire le query con e senza viste materializzate e verificare che l'output ottenuto sia lo stesso nei due casi.

**Esercizio 1.4 (Opzionale, per chi svolge l'esercitazione sul proprio pc)** Provare ad aggiornare il contenuto delle viste materializzate, creando i materialized view log per tutte le tabelle necessarie, in modo da monitorare i cambiamenti di ciascuna tabella.

Per quali tabelle è necessario creare le apposite strutture di log?

**STEP A:** Per ciascuna tabella in cui lo si ritiene necessario, creare il materialized view log nel seguente modo:

```
CREATE MATERIALIZED VIEW LOG ON TABLE1  
WITH ROWID, SEQUENCE(Attribute_1, Attribute_2, Attribute_3)  
INCLUDING NEW VALUES;
```

**N.B.:** I materialized view log vanno creati **prima** della vista materializzata. Eliminare quindi la materialized view nel caso sia stata già creata al punto precedente.

**STEP B:** Creare la vista materializzata, assicurandosi che l'opzione relativa al metodo di aggiornamento sia settata a "FAST" e che il tipo di aggiornamento sia "ON COMMIT".

**STEP C:** Modificare la tabella dei fatti come indicato nell'esercizio 1.2 e verificare che la modifica sia correttamente propagata sulle viste materializzate.

**N.B.:** I materialized view log **non** possono essere creati con la versione di Oracle presente sui computer del laboratorio.

### 3. Aggiornamento e gestione viste tramite Trigger

Supponendo che il comando CREATE MATERIALIZED VIEW non sia disponibile, creare ora le viste materializzate definite nell'esercizio precedente seguendo i passi elencati:

**STEP 3.1** Creare la struttura della vista materializzata con la seguente istruzione:

```
CREATE TABLE VM1 (...)
```

**STEP 3.2** Popolare la tabella VM1 con i record necessari tramite la seguente istruzione

```
INSERT INTO VM1 (...)  
( SELECT ...  
... )
```

**Esercizio:** Si scrivano adesso i Trigger necessari per propagare le modifiche (inserimento di un nuovo record) apportate nella tabella FATTI alle viste materializzate create VM1 e VM2.

Verificare il corretto funzionamento dei trigger eseguendo le seguenti operazioni e verificando che VM1 e VM2 vengano aggiornate di conseguenza:

```
insert into fatti(id_tempo, id_tar, id_luogo_chiamante, id_luogo_chiamato, prezzo, chiamate)  
values(8,2,558,752,150, 40000)  
  
insert into fatti(id_tempo, ID_TAR, id_luogo_chiamante, id_luogo_chiamato, prezzo, chiamate)  
values(1,7,558,752,100,100)
```

Su quale delle due tabelle è stato eseguito l'aggiornamento di un record già esistente? Su quale invece è stato inserito un nuovo record?

#### Comandi utili per la gestione dei trigger

- Cancellazione di un trigger:
  - drop trigger nome\_trigger;
- Sostituzione (aggiornamento) di un trigger esistente:
  - CREATE OR REPLACE TRIGGER nome\_trigger;
- Visualizzazione dei trigger generati:
  - select trigger\_name, triggering\_event, table\_name, status, description, action\_type, trigger\_body from user\_triggers;
- Visualizzazione degli errori dei trigger:
  - select \* from USER\_ERRORS;

#### ESERCIZIO 1.1

QUERY:

1.

```
SELECT t.TIPO_TARIFFA, te.MESE, SUM(f.prezzo) as incassoPerTipologiaAnno,  
SUM(SUM(f.PREZZO)) OVER () as incassoTotale,  
SUM(SUM(f.PREZZO)) OVER (PARTITION BY t.TIPO_TARIFFA) as incassoPerTipologia,  
SUM(SUM(f.PREZZO)) OVER (PARTITION BY te.MESE) as incassoAnnuale  
FROM TARIFFA t, TEMPO te, FATTI f  
WHERE f.ID_TAR=t.ID_TAR AND te.ID_TEMPO=f.ID_TEMPO AND ANNO=2003  
GROUP BY t.TIPO_TARIFFA, te.MESE  
ORDER BY t.TIPO_TARIFFA, te.MESE
```

MISURE: SUM(PREZZO)  
TABELLE: TARIFFA, TEMPO, FATTI  
GROUP BY: TIPO\_TARIFFA, MESE  
PREDICATI DI SELEZIONE: ANNO=2003

2.

```
SELECT Mese, Anno, SUM(F.PREZZO) AS incassoTot, SUM(F.CHIAMATE) AS ChiamateTot, DENSE_RANK()  
OVER(ORDER BY SUM(F.PREZZO) DESC) AS rankIncasso  
FROM FATTI F, TEMPO T  
WHERE F.ID_TEMPO=T.ID_TEMPO  
GROUP BY T.MESE, T.ANNO  
ORDER BY rankIncasso, Anno;
```

MISURE: SUM(PREZZO)  
TABELLE: TEMPO, FATTI  
GROUP BY: MESE  
PREDICATI DI SELEZIONE:

3.

```
SELECT T.MESE, SUM(F.CHIAMATE) AS ChiamateMensili, DENSE_RANK() OVER(ORDER BY  
SUM(F.CHIAMATE) DESC) as rankChiamateMensili  
FROM FATTI F, TEMPO T  
WHERE F.ID_TEMPO=T.ID_TEMPO AND T.ANNO=2003  
GROUP BY T.MESE  
ORDER BY rankChiamateMensili
```

MISURE: SUM(CHIAMATE)  
TABELLE: TEMPO, FATTI  
GROUP BY: MESE  
PREDICATI DI SELEZIONE: ANNO=2003

#### VISTA MATERIALIZZATA:

```
CREATE MATERIALIZED VIEW ViewFatti1  
BUILD IMMEDIATE  
REFRESH FAST ON COMMIT AS  
  SELECT TIPO_TARIFFA, MESE, ANNO, SUM(PREZZO)AS PREZZO, SUM(CHIAMATE)AS CHIAMATE  
  FROM TARIFFA, TEMPO, FATTI  
  WHERE TARIFFA.ID_TAR=FATTI.ID_TAR AND TEMPO.ID_TEMPO=FATTI.ID_TEMPO  
  GROUP BY TIPO_TARIFFA,MESE, ANNO
```

#### PRIMA DEVO GENERARE I MATERIALIZED VIEW LOG

```
CREATE MATERIALIZED VIEW LOG ON FATTI  
WITH SEQUENCE, ROWID  
(ID_TAR, ID_TEMPO, PREZZO, CHIAMATE)  
INCLUDING NEW VALUES;
```

```
CREATE MATERIALIZED VIEW LOG ON TARIFFA  
WITH SEQUENCE, ROWID  
(ID_TAR, TIPO_TARIFFA)  
INCLUDING NEW VALUES;
```

```
CREATE MATERIALIZED VIEW LOG ON TEMPO  
WITH SEQUENCE, ROWID  
(ID_TEMPO, MESE, ANNO)  
INCLUDING NEW VALUES;
```

#### QUERY PER LA VISTA

1.

```
SELECT TIPO_TARIFFA, MESE, SUM(prezzo) as incassoPerTipologiaAnno,  
SUM(SUM(PREZZO)) OVER () as incassoTotale,  
SUM(SUM(PREZZO)) OVER (PARTITION BY TIPO_TARIFFA) as incassoPerTipologia,  
SUM(SUM(PREZZO)) OVER (PARTITION BY MESE) as incassoAnnuale  
FROM viewfatti1  
WHERE ANNO=2003  
GROUP BY TIPO_TARIFFA, MESE  
ORDER BY TIPO_TARIFFA, MESE
```

2.

```
SELECT MESE, ANNO, SUM(PREZZO) AS incassoTot, SUM(CHIAMATE) AS ChiamateTot, DENSE_RANK()  
OVER(ORDER BY SUM(PREZZO) DESC) AS rankIncasso  
FROM viewfatti1  
GROUP BY MESE,ANNO  
ORDER BY rankIncasso, Anno;
```

3.

```
SELECT MESE, SUM(CHIAMATE) AS ChiamateMensili, DENSE_RANK() OVER(ORDER BY SUM(CHIAMATE)  
DESC) as rankChiamateMensili  
FROM viewfatti1  
WHERE ANNO=2003  
GROUP BY MESE  
ORDER BY rankChiamateMensili
```

#### QUERY:

4.

```
SELECT TIPO_TARIFFA,MESE, SUM(F.Prezzo) AS IncassoMensile  
FROM FATTI F, TEMPO T, TARIFFA TAR  
WHERE F.ID_TEMPO=T.ID_TEMPO AND F.ID_TAR=ATAR.ID_TAR AND T.ANNO=2003 AND T.MESE='7-2003'  
GROUP BY TIPO_TARIFFA, MESE  
ORDER BY TIPO_TARIFFA
```

MISURE: SUM(PREZZO)  
TABELLE: TARIFFA, TEMPO, FATTI  
GROUP BY: TIPO\_TARIFFA, MESE  
PREDICATI DI SELEZIONE: ANNO=2003 AND MESE='07-2003'

5.

```
SELECT T.MESE,SUM(F.PREZZO) AS IncassoMensile,  
DENSE_RANK() OVER (PARTITION BY T.ANNO
```

MISURE: SUM(PREZZO)  
TABELLE: TEMPO, FATTI

5.

```
SELECT T.MESE,SUM(F.PREZZO) AS IncassoMensile,
SUM(SUM(F.PREZZO)) OVER ( PARTITION BY T.ANNO
                           ORDER BY T.MESE
                           ROWS UNBOUNDED PRECEDING) as IncassoCumulativo
FROM FATTI F, TEMPO T
WHERE F.ID_TEMPO=T.ID_TEMPO
GROUP BY T.MESE, T.ANNO
```

MISURE: SUM(PREZZO)  
TABELLE: TEMPO, FATTI  
GROUP BY: MESE, ANNO

6.

```
SELECT TIPO_TARIFFA, MESE, SUM(F.PREZZO) AS IncassoTot,
100*((SUM(F.PREZZO))/(SUM(SUM(F.PREZZO))OVER(PARTITION BY TIPO_TARIFFA))) as percTipo,
100*((SUM(F.PREZZO))/(SUM(SUM(F.PREZZO))OVER(PARTITION BY MESE))) as percMese
FROM FATTI F, TEMPO T, TARIFFA TA
WHERE F.ID_TEMPO=T.ID_TEMPO AND F.ID_TAR=TA.ID_TAR AND T.ANNO=2003
GROUP BY TIPO_TARIFFA, MESE
```

MISURE: SUM(PREZZO)  
TABELLE: TARIFFA, TEMPO, FATTI  
GROUP BY: TIPO\_TARIFFA, MESE  
PREDICATI DI SELEZIONE: ANNO=2003

#### VISTA MATERIALIZZATA: (identica alla precedente)

```
CREATE MATERIALIZED VIEW ViewFatti1
BUILD IMMEDIATE
REFRESH FAST ON COMMIT AS
  SELECT TIPO_TARIFFA, MESE, ANNO, SUM(PREZZO)AS PREZZO, SUM(CHIAMATE)AS CHIAMATE
  FROM TARIFFA, TEMPO, FATTI
  WHERE TARIFFA.ID_TAR=FATTI.ID_TAR AND TEMPO.ID_TEMPO=FATTI.ID_TEMPO
  GROUP BY TIPO_TARIFFA,MESE, ANNO
```

#### QUERY PER LA VISTA

4.

```
SELECT TIPO_TARIFFA,MESE, SUM(Prezzo) AS IncassoMensile
FROM viewfatti1
WHERE ANNO=2003 AND MESE='7-2003'
GROUP BY TIPO_TARIFFA, MESE
ORDER BY TIPO_TARIFFA
```

5.

```
SELECT MESE,SUM(PREZZO) AS IncassoMensile,
SUM(SUM(PREZZO)) OVER ( PARTITION BY ANNO
                           ORDER BY MESE
                           ROWS UNBOUNDED PRECEDING) as IncassoCumulativo
FROM viewfatti1
GROUP BY MESE, ANNO
```

6.

```
SELECT TIPO_TARIFFA, MESE, SUM(PREZZO) AS IncassoTot,
100*((SUM(PREZZO))/(SUM(SUM(PREZZO))OVER(PARTITION BY TIPO_TARIFFA))) as percTipo,
100*((SUM(PREZZO))/(SUM(SUM(PREZZO))OVER(PARTITION BY MESE))) as percMese
FROM viewfatti1
WHERE ANNO=2003
GROUP BY TIPO_TARIFFA, MESE
```

#### Esercizio 3

```
CREATE TABLE VM1(
TipoTariffa VARCHAR(26),
Mese DATE CHECK (Mese IS NOT NULL),
Anno INTEGER CHECK(Anno IS NOT NULL),
PrezzoTot INTEGER CHECK(PrezzoTot IS NOT NULL AND PrezzoTot>0),
ChiamateTot INTEGER CHECK (ChiamateTot IS NOT NULL AND ChiamateTot>0)
PRIMARY KEY(TipoTariffa, Mese)
)

INSERT INTO VM1 (TipoTariffa, Mese, Anno, PrezzoTot, ChiamateTot)
(
  SELECT TIPO_TARIFFA, MESE, ANNO, SUM(PREZZO), SUM(CHIAMATE)
  FROM TARIFFA TA, TEMPO T, FATTI F
  WHERE TA.ID_TAR=F.ID_TAR AND T.ID_TEMPO=F.ID_TEMPO
  GROUP BY TIPO_TARIFFA, MESE, ANNO
)
```

#### CREAZIONE DEI TRIGGER PER PROPAGARE LE MODIFICHE DELLA TABELLA FATTI

```
CREATE TRIGGER RefreshVM1
AFTER INSERT ON FATTI
FOR EACH ROW
DECLARE
```

```

varTipoTariffa VARCHAR2(26);
varMese VARCHAR2(26);
varAnno INTEGER;
N INT;

BEGIN
-- leggere le tabelle dimensionali per recuperare i valori dell'identificatore della vista
materializzata
-- Servizio, Nazionalità, Semestre, Regione

SELECT TIPO_TARIFFA INTO varTipoTariffa
FROM TARIFFA
WHERE ID_TAR=:NEW.ID_TAR;

SELECT MESE,ANNO INTO varMese, varAnno
FROM TEMPO
WHERE ID_TEMPO=:NEW.ID_TEMPO;

-- Verifico se esiste una tupla in 'ViewFatti1' associata ai valori di

SELECT COUNT(*) INTO N
FROM VM1
WHERE TIPOTARIFFA=varTipoTariffa AND MESE=varMese AND ANNO=varAnno;

IF(N>0) THEN
  UPDATE VM1
  SET PrezzoTot=PrezzoTot+:NEW.PREZZO,
  ChiamateTot=ChiamateTot+:NEW.CHIAMATE
  WHERE TIPOTARIFFA=varTipoTariffa AND MESE=varMese;

ELSE
  INSERT INTO VM1 (TipoTariffa, Mese, Anno, PrezzoTot, ChiamateTot)
  VALUES(varTipoTariffa, varMese, varAnno, :NEW.PREZZO, :NEW.CHIAMATE);

END IF;
END;

```