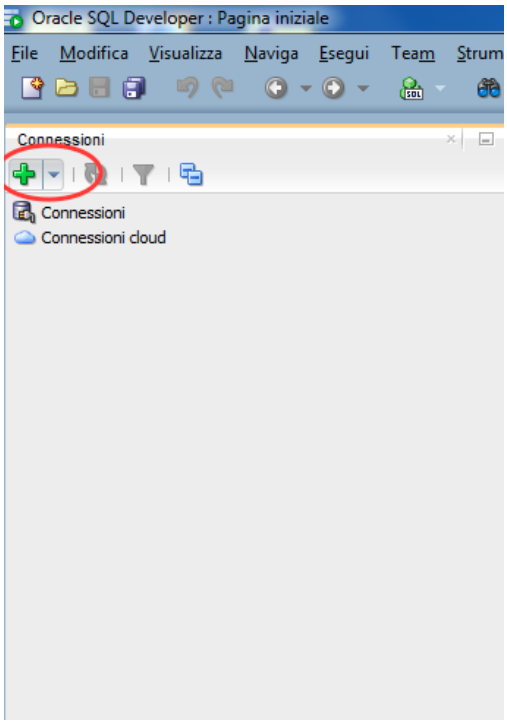
Sistemi di gestione di basi di dati

Esercitazione 2 – Trigger

Connessione alla base di dati

Aprire il programma Oracle SQL Developer
Cliccare su crea nuova connessione:

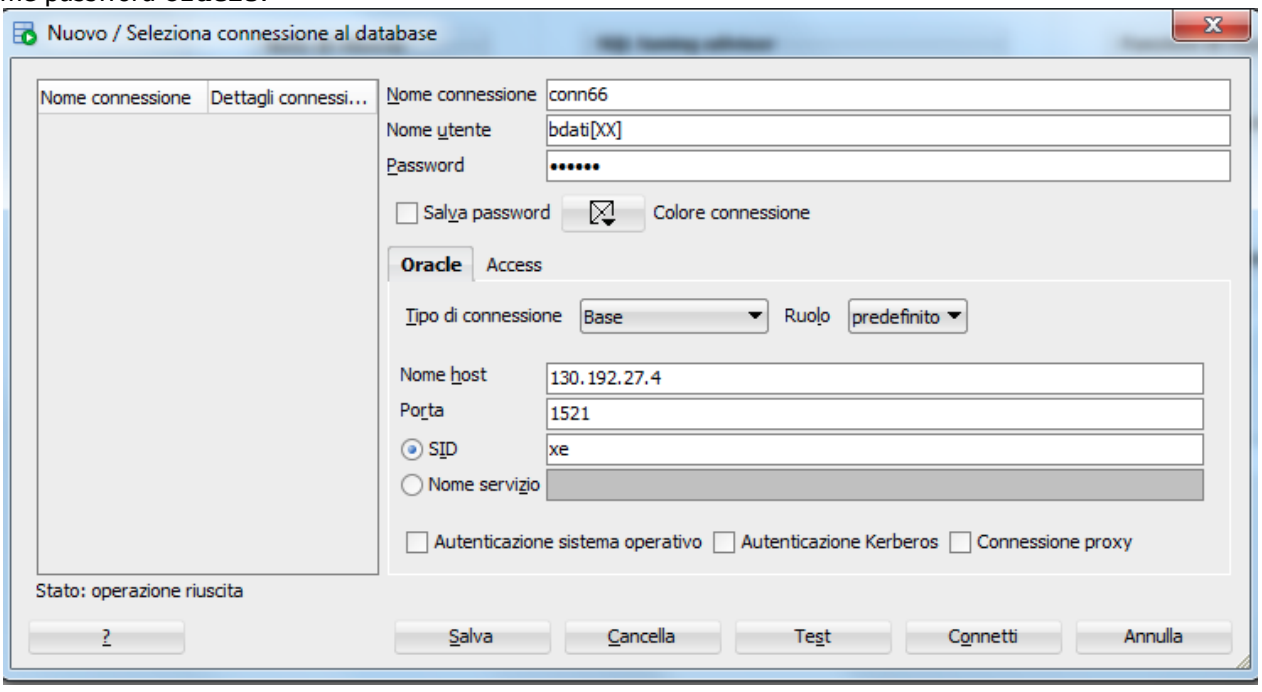


Login

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`**.



Materiale disponibile

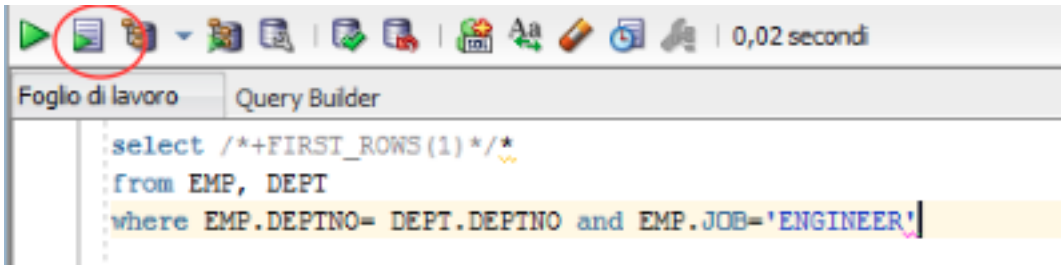
Sono disponibili alcuni script contenenti istruzioni SQL per svolgere le seguenti operazioni:

1. creazione della base di dati
2. creazione dei trigger
3. visualizzazione dei trigger
4. modifica della base dati per attivare i trigger.

Gli script sono disponibili:

- sul sito web del corso http://dbdmg.polito.it/wordpress/teaching/sistemi-per-la-gestione-di-basi-di-dati/#Laboratori-1

Gli script possono essere caricati aprendo File->Apri e selezionando il file .sql e successivamente cliccando il pulsante "Esegui Script"



Comandi utili

Cancellazione di un trigger:

```
drop trigger nome_trigger;
drop trigger "nome_trigger";
```

Sostituzione (aggiornamento) di un trigger esistente (anziché cancellarlo e ricrearlo):

```
CREATE OR REPLACE TRIGGER nomeTrigger …
```

Visualizzazione dei trigger generati:

```
select trigger_name, triggering_event, table_name,
status, description, action_type, trigger_body
from user_triggers;
```

Disabilitazione di un trigger esistente:

```
ALTER TRIGGER triggerName DISABLE;
```

Visualizzazione degli errori dei trigger:

```
select * from USER_ERRORS;
```

Consigli

Per creare le tabelle degli esercizi, sono disponibili gli script script_db_es1.sql, script_db_es2.sql, script_db_es3.sql.

Per la creazione dei trigger, prestare attenzione alla sintassi e ai seguenti dettagli:

- assegnare un nome opportuno alle variabili, evitando parole chiave come MIN, MAX, …
- dichiarare variabili diverse su righe diverse e **non** sulla stessa riga separate da virgola

```
MyVarUno NUMBER;
MyVarDue NUMBER;
MyVarTre VARCHAR2(16);
```

- terminare il trigger con il carattere **/** come nello script `create_triggers.sql` (la creazione del trigger può andare a buon fine anche in assenza del carattere di terminazione);
- terminare le istruzioni con il carattere **;** assegnare nuovi valori alle variabili con **:=**, es.

```
UPDATE tablename
   SET varname=newvalue
   WHERE column=:NEW.attribute;

IF A<3 OR A=3 THEN
   MyVar:='Tre';
ELSE
IF A>3 AND A<5 THEN
   MyVar:='Quattro';
ELSE
   MyVar:='Altro';
END IF;
END IF;
```

Prima di iniziare l'esercitazione si consiglia di eliminare gli eventuali trigger già presenti sull'utente scelto, che potrebbero alterare i risultati degli esercizi proposti.
N.B. Copiando e incollando porzioni di codice direttamente dal testo dell'esercitazione in formato pdf al Browser Web prestare attenzione all'eventuale conversione o codifica errata dei caratteri, che potrebbe generare il seguente errore: *ora-00911: invalid character.*

Esercizio #1

Si consideri la seguente base di dati

IMP (EMPNO, DEPTNO, ENAME, JOB, SAL)

DIP (DEPTNO, DNAME, LOC, MINSAL, MAXSAL)

Nel caso un dipartimento passi dal ruolo (attributo DNAME) di 'ACCOUNTING' al ruolo di 'SALES', il salario per tutti gli impiegati del dipartimento viene incrementato di 100. Definire un trigger che implementi tale funzionalità.

Lo svolgimento dell'esercizio è strutturato nei seguenti passi:

1. Creare la base dati utilizzando lo script `create_db_es1.sql`
2. Creare il trigger, eventualmente mediante uno script.
3. Verificare il contenuto delle tabelle IMP e DIP.
4. Modificare il nome del dipartimento 'ACCOUNTING':
   `UPDATE DIP set DNAME = 'SALES' where DNAME='ACCOUNTING';`
5. Verificare il contenuto delle tabelle IMP e DIP.

Durante l'esercitazione dovranno essere eseguiti i seguenti passi:

- scrivere il trigger;
- verificare l'output generato (dettagli del risultato via Web, come nella figura di esempio a destra) in cui si osserva il risultato ottenuto

Esercizio #2

Si consideri la seguente base di dati relativa all'emissione di biglietti (TICKETS) per voli aerei (FLIGHTS), per i quali gli utenti possono essere in possesso di tessere promozionali (CARDS). Ogni tessera acquista un credito (CREDITS) in miglia percorse corrispondenti al biglietto comprato con la tessera stessa. I possessori della tessera possono accedere a diverse fasce di sconti in base al totale delle miglia percorse con i voli dei biglietti comprati, in particolare fino a 30 mila miglia totali lo stato della tessera è SILVER, da 30 a 50 mila è GOLD e oltre a 50 mila è PREMIUM. I passaggi di stato scatenano una notifica (NOTIFY) al possessore della tessera.

CARDS(CARDNO, NAME, STATUS)

FLIGHTS(FLIGHTID, DEPARTURETIME, DEPARTURECITY, ARRIVALCITY, MILES)

TICKETS(TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)

CREDITS(TICKETID, CARDNO, MILES)

NOTIFY(CARDNO, NOTIFYNO, NOTIFYDATE, OLDSTATUS, NEWSTATUS, TOTALMILES)

Si scriva il trigger necessario per gestire l'emissione di un nuovo biglietto, aggiornando opportunamento la base dati, come descritto in dettaglio di seguito. Quando il biglietto è emesso, se è associato al numero di tessera di un cliente (CARDNO IS NOT NULL), è necessario aggiornare le informazioni sulle miglia percorse. In particolare come prima operazione occorre aggiornare la tabella **CREDITS** con le informazioni appropriate. È necessario verificare se il cliente ha cambiato STATUS e aggiornarlo di conseguenza in **CARDS**. Lo status iniziale è SILVER e resta tale fino a quando il cliente non accumula un numero di miglia superiore a 30000. Quando le miglia totali accumulate sono comprese tra 30000 e 50000, lo STATUS viene aggiornato a GOLD. Se superano 50000 miglia percorse diviene PREMIUM (livello massimo). Se avviene una variazione di STATUS del cliente, occorre inserire nella tabella **NOTIFY** l'informazione necessaria per notificare al cliente la variazione di stato. Si noti che l'attributo NOTIFYNO è un contatore che deve essere incrementato di 1 ogni volta che si inserisce una nuova notifica per lo stesso cliente.

È consigliabile scrivere un trigger per il primo passo, provarlo e poi aggiornarlo per soddisfare i requisiti del secondo passo, e infine integrarlo con le richieste del terzo ed ultimo passo.

Passo 1

Quando il biglietto è emesso, se è associato al numero di tessera di un cliente (CARDNO IS NOT NULL), è necessario aggiornare le informazioni sulle miglia percorse. In particolare come prima operazione occorre aggiornare la tabella **CREDITS** con le informazioni appropriate.

Passo 2

È necessario verificare se il cliente ha cambiato STATUS e aggiornarlo di conseguenza in **CARDS**.

Lo status iniziale è SILVER e resta tale fino a quando il cliente non accumula un numero di miglia superiore a 30000. Quando le miglia totali accumulate sono comprese tra 30000 e 50000, lo STATUS viene aggiornato a GOLD. Se superano 50000 miglia percorse diviene PREMIUM (livello massimo).

Passo 3

Se avviene una variazione di STATUS del cliente, occorre inserire nella tabella **NOTIFY** l'informazione necessaria per notificare al cliente la variazione di stato. Si noti che l'attributo NOTIFYNO è un contatore che deve essere incrementato di 1 ogni volta che si inserisce una nuova notifica per lo stesso cliente.

Procedimento:

1. Creare la base dati utilizzando lo script `create_db_es2.sql`
2. Creare il trigger affinché soddisfi i requisiti del passo 1
3. Verificare il funzionamento del trigger inserendo due record nella tabella TICKETS
   - un record che abbia CARDNO impostato a NULL, es.
     ```
     INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
     VALUES ('T02', 'RN12K', '01-MAR-07', 'PIPPO', NULL);
     ```
   - un altro record con CARDNO uguale a un valore presente nella tabella CARDS, es.
     ```
     INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
     VALUES ('T03', 'RN12K', '02-APR-07', 'BILL', 50);
     ```
4. Verificare il contenuto delle tabelle e valutarne la correttezza
   ```
   select * from CREDITS;
   ```

```
select * from TICKETS;
```

5. Aggiornare il trigger affinché soddisfi anche i requisiti del passo 2
6. Verificare il funzionamento del trigger inserendo dei record nella tabella TICKETS in modo da far cambiare status a un cliente
   - inserire uno o più biglietti per voli molto lunghi o per clienti che hanno già altri voli, es.

```
INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T04', 'RN12K', '03-MAY-07', 'BILL', 50);
INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T05', 'RN12K', '03-MAY-07', 'BILL', 50);
```

7. Verificare il contenuto delle tabelle e valutarne la correttezza

```
select * from CREDITS;
select * from TICKETS;
select * from CARDS;
```

8. Aggiornare il trigger affinché soddisfi anche i requisiti del passo 3
9. Verificare il funzionamento del trigger inserendo dei record nella tabella TICKETS in modo da far cambiare status a un altro cliente
   - inserire uno o più biglietti per voli molto lunghi o per clienti che hanno già altri voli, es.

```
INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T06', 'RN12K', '03-MAY-07', 'BILL', 50);
INSERT INTO TICKETS (TICKETID, FLIGHTID, FLIGHTDATE, NAME, CARDNO)
VALUES ('T07', 'RN12K', '03-MAY-07', 'BILL', 50);
```

10. Verificare il contenuto delle tabelle e valutarne la correttezza

```
select * from NOTIFY;
select * from CREDITS;
select * from TICKETS;
select * from CARDS;
```

Durante l'esercitazione dovranno essere eseguiti i seguenti passi:

- verificare il risultato di ogni passo controllando i cambiamenti della base dati;
- scrivere il trigger completo.

Esercizio #3

Si consideri la seguente base di dati:

```
IMP ( EMPNO, ENAME, JOB, SAL )
SUMMARY ( JOB, NUM )
```

Verificare che siano state create correttamente le tabelle necessarie effettuando le seguenti query:

```
Select * from IMP;
Select * from SUMMARY
```

Nella tabelle SUMMARY, il campo NUM indica il numero di impiegati in IMP che svolgono uno stesso lavoro. Si scrivano i trigger per mantenere la consistenza tra la tabella IMP e SUMMARY in caso di:

- inserimento di un record in IMP
- aggiornamento del campo JOB in IMP