ESERCITAZIONE
5 Agosto 2022

Consegna prevista per le 18:00, una volta terminata la prova bisognerà inviarla a soluzioni.esercizi2022@gmail.com

La prova va svolta necessariamente in autonomia. Buon lavoro!

Teoria

1) **Cosa descrive la Piramide di Anthony? Qual è la sua utilità?**

La piramide di Anthony è un sistema di classificazione per l'organizzazione di imprese e industrie, in cui il lavoro viene distribuito ed organizzato in modo gerarchico e piramidale.
la piramide è costituita da tre macro aree: Attività strategiche (in cima alla piramide), attività tattiche di programmazione e direzione funzionale (al livello intermedio) e attività operative alla base, in cui vi è il personale esecutivo.
Il vantaggio è una semplificazione gestionale, potendo fare riferimento sempre ad un unico responsabile di livello superiore che ha una visione più a lungo termine del lavoro complessivo da svolgere.

2) **Descrivi le differenze tra OLTP ed OLAP.**

OLTP (Online Transaction Processing) è un tipo di elaborazione dei dati che consiste nell'esecuzione di una serie di transazioni che si verificano contemporaneamente.
È indicato in un contesto bancario, di e-commerce, gestionali della pubblica amministrazione ecc.
OLAP, acronimo dell'espressione On-Line Analytical Processing, è invece,
un insieme di tecniche software per l'analisi interattiva e veloce di grandi quantità di dati (reportistica e data mining).

3) **Da quali tabelle è composto il Data Warehouse? Descrivile indicandone le differenze.**

Secondo il Dimensional Fact Model, le tabelle appartengono a due tipologie distinte:

Le tabelle dei fatti, sono tabelle che contengono le misurazioni degli eventi che si desiderano analizzare, e vengono rappresentati al centro diagramma a stella o fiocco di neve.
Le tabelle delle dimensioni forniscono un contesto analitico alle misure, e vengono rappresentati ai rami del diagramma a stella o fiocco di neve.

4) In cosa consiste il Data Mining? Quali sono i vantaggi e gli svantaggi?

Il data mining è l'insieme di tecniche e metodologie che hanno per oggetto l'estrazione di informazioni utili e da grandi quantità di dati.
I vantaggi sono nell'avere la possibilità di utilizzare queste informazioni per indirizzare in maniera mirata degli investimenti, della forza lavoro, o in generale prendere delle decisioni aziendali basandosi sui fatti.
È molto indicato per i settori:
Marketing, Finance / Banking, Manufacturing e Governments
Gli svantaggi sono legati alla privacy (esempio deduzione di dati sensibili da attività nei social online) e alla sicurezza (spionaggio, truffe online ecc.).

5) Descrivi l'architettura del Data Warehouse.

L'architettura di un Data Warehouse è costituita da uno o piu datamart che raggruppa le informazioni di un'unica area e un unico argomento.
Uno dei modelli più utilizzati per la progettazione di un datamart è il Dimensional Fact Model (DFM).

Pratica

1) Scrivi un programma PL/SQL che stampi tutti i numeri primi tra 1 e 50.

```
declare
   numero int := 1;
   x int;
   trovato boolean ;
begin
     while  (numero<=50) loop
       trovato   := false;
       x:=2;
       while  (x<numero) loop

          if (mod(numero,x) =0) then

             trovato := true;
             exit;
          end if;
          x:=x+1;
       end loop ;
       if (not trovato) then
          dbms_output.put_line(numero || ' ');
       end if;
       numero:=numero+1;
   end loop;
end;
```

2) Cosa cambieresti (o cosa trovi di sbagliato) nei seguenti codici? Perchè?

a- FOR i IN 1 .. 100

LOOP calc_totale (i);

IF i > 75

THEN EXIT;

END IF;

END LOOP;

Bastava modificare il ciclo for in 1..75

b- OPEN cursore;

FETCH cursore INTO record;

WHILE cursore%FOUND

LOOP calc_tot (record.salario);

FETCH cursore INTO record;

EXIT WHEN record.salario > 100000;

END LOOP;

CLOSE cursore;

EXIT WHEN record.salario > 100000;
Andrebbe messo subito dopo, per evitare di generare errore
WHILE cursore%FOUND

c- FOR contatore IN lim_inf .. lim_sup

LOOP

IF contatore > lim_inf * 2

THEN

lim_sup := lim_inf;

END IF;

END LOOP;

L'istruzione lim_sup := lim_inf; non farà uscire anticipatamente dal ciclo for

d- DECLARE

CURSOR cursore IS

SELECT salario

FROM tabella;

```
record cursore%ROWTYPE
BEGIN
OPEN cursore;
LOOP FETCH cursore INTO record;
EXIT WHEN cursore%NOTFOUND;
calc_tot (record.salario);
END LOOP;
CLOSE cursore;
END;
```

record cursore%ROWTYPE manca il ';'

3) Riscrivi questo codice in modo tale da ottenere lo stesso risultato senza utilizzare un LOOP.

```
FOR i IN 1 .. 2
LOOP
IF i = 1
THEN
dai_bonus (impiegato_id, 2000000);
ELSIF i = 2
THEN
dai_bonus (presidente_id, 5000000);
END IF;
END LOOP;
```

Sostituire tutto il codice con

Exec dai_bonus (impiegato_id, 2000000);
Exec dai_bonus (presidente_id, 5000000);

4) Indica quale attributo dei cursori viene utilizzato in ognuna di queste occasioni:

a) Se il FETCH non riporta nessun record dal cursore.

%NOTFOUND

b) Se si vuole mostrare a video il numero di record riportati.

%ROWCOUNT

c) Se il cursore rimane aperto.

%ISOPEN

d) Se il FETCH trova il record desiderato.

%found

5) Riscrivi il seguente codice utilizzando un costrutto diverso dall'IF che snellisca il programma e lo renda di più facile lettura.

```
IF salario < 10000
THEN bonus := 2000;
ELSE
IF salario < 20000 THEN bonus := 1500;
ELSE
IF salario < 40000 THEN bonus := 1000;
ELSE bonus := 500;
END IF;
END IF;
END IF;
```

Riscrivere il codice con:

```
CASE
        when salario  < 10000 then bonus := 2000;
        when salario < 20000  then bonus := 1500;
        when salario < 40000 then bonus := 1000;
        else
                bonus := 500;
end case;
```

6) Quale statement rimuoveresti da questo blocco di codice? Perché?

```
DECLARE
CURSOR cursore IS
SELECT nome, cognome, numero
FROM tabella
WHERE stipendio < 2500;
record cursore%ROWTYPE;

BEGIN
FOR record IN cursore
```

```
LOOP dai_aumento (record.numero, 10000);
END LOOP;
END;
```

Rimuoverei La riga cursore%ROWTYPE;

7) Dopo l'esecuzione di questo blocco di codice PL/SQL sarà sollevata un'eccezione predefinita. Sapresti indicare quale e perché?

```
DECLARE
stringa VARCHAR2(5);
BEGIN
stringa := 'Steven';
END;
```

character string buffer too small
sostituire il 5 con minimo 7
stringa VARCHAR2(7);

8) Quanti e quali statement mancano dal seguente blocco di codice?

```
DECLARE
CURSOR cursore
IS
SELECT * FROM tabella;
BEGIN
OPEN cursore;
FETCH cursore INTO record;
END;
```

Le modifiche minime solo

per una corretta

esecuzione sono:

Aggiungere

record tabella%ROWTYPE; prima di begin

Aggiungere close cursore; prima di end;

9) All'interno dei seguenti blocchi di codice sono contenuti 8 errori. Sapresti indicarli tutti?

```
CREATE OR REPLACE PACKAGE c_package AS
```

```
  PROCEDURE aggiungiCliente(c_id clienti.id%type,
  c_nome clienti.Nome%type
  c_età clienti.età%type,
  c_indir clienti.indirizzo%type,
  c_sal clienti.salario%type);


  PROCEDURE canc_cliente(c_id  clienti.id TYPE);


  PROCEDURE listaCliente;

END;

CREATE OR REPLACE PACKAGE c_package AS
  PROCEDURE aggiungiCliente(c_id clienti.id%type,
    c_nome clienti.Nome%type,
    c_età clienti.età%type,
    c_indir clienti.indirizzo%type,
    c_sal  clienti.salario%type))
  IS
  BEGIN
    INSERT clienti (id,nome,età,indirizzo,salario)
      VALUES(c_id, c_nome, c_età, c_indir, c_sal);
  END aggiungiCliente;

  PROCEDURE canc_cliente(c_id clienti.id%type) IS
  BEGIN
```

```
    DELETE FROM clienti
    WHERE id = c_id;
  END canc_cliente;

  PROCEDURE listaClientela IS
  CURSOR c_clienti is
    SELECT  nome FROM clienti;
  TYPE c_lista TABLE OF clienti.Nome%type;
  nome_lista c_lista := c_lista();
  counter integer :=0;
  BEGIN
    FOR n IN c_clienti LOOP
    counter := counter +1;
    nome_lista.extend;
    nome_lista(counter) := n.nome;
    dbms_output.put_line('Cliente(' ||counter|| ')'||nome_lista(counter));

  END listaCliente;

        END c_package;
```

1.

Gli errori sono:
c_nome clienti.Nome%type manca la virgola

2.

PROCEDURE canc_cliente(c_id clienti.id TYPE);

Costituire con

PROCEDURE canc_cliente(c_id clienti.id%type);

3.

Dopo il primo End sostutuire

CREATE OR REPLACE PACKAGE c_package AS

Con

CREATE OR REPLACE PACKAGE BODY c_package AS

4.

Togliere la parentesi di chiusura in

```
  PROCEDURE aggiungiCliente(c_id clienti.id%type,
    c_nome clienti.Nome%type,
    c_età clienti.età%type,
```

c_indir clienti.indirizzo%type,

c_sal clienti.salario%type))

Con
PROCEDURE aggiungiCliente(c_id clienti.id%type, c_nome clienti.Nome%type,

c_età clienti.età%type, c_indir clienti.indirizzo%type, c_sal clienti.salario%type)

5.
Inserire la clausola into in
INSERT clienti (id,nome,età,indirizzo,salario)

VALUES(c_id, c_nome, c_età, c_indir, c_sal);

Con
INSERT into clienti (id,nome,età,indirizzo,salario) VALUES(c_id, c_nome, c_età, c_indir, c_sal);

6.
Aggiungere commit prima di
END canc_cliente;

7.Inserire:
Declare
Begin
dopo
PROCEDURE listaClientela IS

8.
Inserire prima di :
END listaCliente;
End loop;

10) In ognuno dei seguenti esempi di codice modulare, identifica i cambiamenti che apporteresti per migliorarne la struttura, la performance e la funzionalità.

a- CREATE OR REPLACE FUNCTION funzione (var_funz IN VARCHAR2)

RETURN VARCHAR2

IS

BEGIN

IF var_funz = 'C' THEN RETURN 'CLOSED';

ELSIF var_funz = 'O' THEN RETURN 'OPEN';

ELSIF var_funz = 'A' THEN RETURN 'ACTIVE';

ELSIF var_funz = 'I' THEN RETURN 'INACTIVE';

END IF;

END;

Utilizzare un case when al posto dei vari if

b- CREATE OR REPLACE FUNCTION funzione (var_funz IN VARCHAR2, altra_var_funz OUT DATE)

RETURN VARCHAR2

IS

BEGIN

... /* stessa funzione di prima */

END;

Togliere il parametro altra_var_funz OUT DATE in quanto non utilizzato

c- CREATE OR REPLACE FUNCTION nome_compagnia (compagnia_id_in IN compagnia.compagnia_id%TYPE)

RETURN VARCHAR2

IS

c_nome compagnia.compagnia_id%TYPE;

e_trovata EXCEPTION;

BEGIN

SELECT nome INTO c_nome

FROM compagnia

WHERE compagnia_id = compagnia_id_in;

RAISE e_trovata;

EXCEPTION

WHEN NO_DATA_FOUND

THEN

RETURN NULL;

WHEN e_trovata

THEN

RETURN c_nome;

END;

Sostutuire
c_nome compagnia.compagnia_id%TYPE; e_trovata EXCEPTION;
Con
c_nome compagnia.compagnia_nome%TYPE; e_trovata EXCEPTION;

d- CREATE OR REPLACE PROCEDURE procedura (compagnia_id IN NUMBER)

IS

bil_rimanente NUMBER := vend_mens (compagnia_id);

```
BEGIN
FOR ind_mens IN 1 .. 12
LOOP
IF bil_rimanente <= 0
THEN
RETURN 0;
ELSE
bil_rimanente := debito (compagnia_id, ind_mens);
END IF;
END LOOP;
END;
```

Sostutuire
RETURN 0;
Con exit;

11) All'interno del seguente blocco di codice sono contenuti 5 errori. Sapresti indicarli tutti?

```
CREATE OR REPLACE TRIGGER t_trigger
 IS
  INSERT OR
  UPDATE OF salario, dipartimento_id
  DELETE
 ON impiegati
BEGIN
 CASE V_selettore
  WHEN INSERTING THEN
   DBMS_OUTPUT.PUT_LINE('Inserimento');
  WHEN UPDATING('salario') THEN
   DBMS_OUTPUT_PUT_LINE('Aggiorno salario');
  WHEN UPDATING('dipartimento_id') THEN
   DBMS_OUTPUT.PUT_LINE('Aggiorno ID dipartimento');
  WHEN DELETING THEN
  DBMS_OUTPUT.PUT_LINE('Cancello...');
END;
```

1. ***sostutuire***

UPDATE OF salario, dipartimento_id DELETE

Con
UPDATE OF salario, dipartimento_id or DELETE

2. ***Scrivere***

END CASE ;
dopo
DBMS_OUTPUT.PUT_LINE('Cancello...');

3. ***Scrivere***

BEFORE oppure AFTER oppure INSTEAD OF
Dopo
CREATE OR REPLACE TRIGGER t_trigger
4.
DBMS_OUTPUT_PUT_LINE sostituire con
DBMS_OUTPUT.PUT_LINE
5.Sostituire
CASE V_selettore
Con CASE

12) Completa la seguente tabella:

Tipo di Collezione	**a)Numero di elementi inseribili al suo interno (limitati/non limitati)**	**b)Tipo/i di dato attribuibili all'indice**	**c)Dense o Sparse**	**d)Possono essere salvati in una colonna del DB (si/no)**	**e)Elementi al suo interno possono essere cancellati indistintamente (si/no)**
1)Array Associativo (index-by)	illimitati	BINARY_INTEGER, PLS_INTEGER, or VARCHAR2	dense	no	Si – qualsiasi posizione
2)Nested Table	illimitati	int	sparse	si	si – qualsiasi posizione
3)VArray	limitati	int	sparse	no	no

13) Immagina di lavorare su di un database di una banca. Un cliente deve trasferire dei soldi da un conto ad un altro e tu hai il compito di creare un programma che supervisioni il tutto. I tre passaggi da tenere in conto sono:

1- Decremento del primo conto

2- Aumento del secondo conto

3- Registrazione della transazione nel registro

Quale tipo di query utilizzeresti per ognuno di questi passaggi? Tenendo presente che, se una di queste azioni non dovesse andare a buon fine bisognerebbe ricominciare da capo, dove inseriresti le parole chiave della Transazione?

```
Begin
        <inizio>
        SAVEPOINT punto_di_salvataggio;
        Decremento_del_primo_conto();
        Aumento_del_secondo_conto();
        Commit;
 exception when others then
     dbms_output.put_line('Errore in fase di salvataggio');
           rollback TO SAVEPOINT punto_di_salvataggio;
        Goto inizio;
End;
```