

Trigger 1

ATLETA(CodAtleta, NomeSquadra)
ARRIVO_ATLETI(CodAtleta, TempoImpiegato)
ARRIVO_SQUADRE(NomeSquadra, NumeroAtletiArrivati)
CLASSIFICA(Posizione, CodAtleta, TempoImpiegato)

Si scrivano i trigger necessari per aggiornare le tabelle *ARRIVO_SQUADRE* e *CLASSIFICA*, in caso di inserimento di un record nella tabella *ARRIVO_ATLETI*. Per l'aggiornamento della tabella *ARRIVO_SQUADRE* si consideri anche il caso in cui una squadra non sia ancora presente nella tabella. Per l'aggiornamento della tabella *CLASSIFICA* si consideri, che il campo *TempoImpiegato* possa assumere lo stesso valore per due atleti diversi.

```
CREATE OR REPLACE TRIGGER Update_Classifica
AFTER INSERT ON ARRIVO_ATLETI
FOR EACH ROW
DECLARE
    X NUMBER;
    TEMPO NUMBER;
    POSATLETA NUMBER;
    PARIMERITO BOOLEAN;
BEGIN
    -- Calcolo la posizione dell'atleta nella classifica
    SELECT MAX(Posizione), MAX(TempoImpiegato) INTO X, TEMPO
    FROM CLASSIFICA
    WHERE (TempoImpiegato <= :NEW.TempoImpiegato);
    IF (X IS NULL)
    THEN
        -- L'atleta è in prima posizione e/o la tabella CLASSIFICA è vuota
        POSATLETA := 1;
        PARIMERITO := FALSE;
    ELSE
        -- Verifichiamo se c'è un parimerito
        IF (Tempo < :NEW.TempoImpiegato)
        THEN
            POSATLETA := X+1;
            PARIMERITO := FALSE;
        ELSE
            POSATLETA := X;
            PARIMERITO := TRUE;
        END IF;
    END IF;

    INSERT INTO CLASSIFICA(Posizione, CodAtleta, TempoImpiegato)
    VALUES (POSATLETA, :NEW.CodAtleta, :NEW.TempoImpiegato);

    IF (PARIMERITO = FALSE)
    THEN
        UPDATE CLASSIFICA
        SET Posizione = Posizione+1
        WHERE (TempoImpiegato > :NEW.TempoImpiegato);
    END IF;
END;
```

```

CREATE OR REPLACE TRIGGER Update_Arrivo_Squadre
AFTER INSERT ON ARRIVO_ATLETI
FOR EACH ROW
DECLARE
    SQUADRA CHAR(10);
    X NUMBER;
BEGIN
    -- Leggo la squadra di appartenenza dell'atleta
    SELECT NomeSquadra INTO SQUADRA
    FROM ATLETA
    WHERE CodAtleta = :NEW.CodAtleta;

    -- Verifico la presenza della squadra nella tabella ARRIVO_SQUADRE
    SELECT COUNT(*) INTO X
    FROM ARRIVO_SQUADRE
    WHERE NomeSquadra = SQUADRA;

    IF (X = 0)
    THEN
        -- È il primo atleta di quella squadra arrivato
        INSERT INTO ARRIVO_SQUADRE (NomeSquadra, NumeroAtletiArrivati)
        VALUES (SQUADRA, 1);
    ELSE
        UPDATE ARRIVO_SQUADRE
        SET NumeroAtletiArrivati = NumeroAtletiArrivati+1
        WHERE NomeSquadra = SQUADRA;
    END IF;
END;

```

Trigger 2a

RICHIESTA_NOLEGGIO(CodRich, DataRich, CodSocio, DataInizioNoleggio, DataFineNoleggio, Modello)

RINUNCIA_NOLEGGIO(CodRin, DataRin, CodSocio, DataInizioNoleggio, CodB)

NOLEGGIO(CodB, DataInizio, CodSocio, DataFine)

NOLEGGIO_IN_ATTESA(CodRich, DataRich, CodSocio, DataInizioNoleggio, DataFineNoleggio, Modello)

BARCA(CodB, NomeB, Modello, Marca, AnnoFabbr)

Si scriva un trigger che gestisca *una richiesta di noleggio*.

Occorre verificare se esiste una barca del modello prescelto non in prestito nel periodo richiesto. In caso affermativo, si definisce un nuovo noleggio nella tabella NOLEGGIO. Nel caso in cui più barche siano disponibili, si noleggia quella di fabbricazione più recente (per semplicità si supponga che ve ne sia sempre una sola).

Se invece nessuna barca è disponibile nel periodo richiesto, si inserisce la richiesta tra le richieste in attesa di essere processate (tabella NOLEGGIO_IN_ATTESA).

```
CREATE OR REPLACE TRIGGER Richiesta_Noleggio
AFTER INSERT ON RICHIESTA_NOLEGGIO
FOR EACH ROW
DECLARE
    X NUMBER;
    Y NUMBER;
BEGIN
    /* Conto le barche di quel modello nel periodo che non bloccano la richiesta di
       noleggio inserita */
    SELECT COUNT(*) INTO X
    FROM BARCA
    WHERE Modello = :NEW.Modello AND CodB NOT IN
        (SELECT CodB
         FROM NOLEGGIO
         WHERE DataInizio <= :New.DataFineNoleggio AND
              DataFine >= :NEW.DataInizioNoleggio);

    IF (X >= 1)
    THEN
        /* Esiste almeno una barca che può essere noleggiata, devo noleggiare quella
           più nuova */
        SELECT CodB INTO Y
        FROM BARCA
        WHERE Modello = :NEW.Modello AND AnnoFabbr =
            (SELECT MAX(AnnoFabbr)
             FROM BARCA
             WHERE Modello = :NEW.Modello AND
                  CodB NOT IN (SELECT CodB
                              FROM NOLEGGIO
                              WHERE DataInizio <= :New.DataFineNoleggio AND
                                   DataFine >= :NEW.DataInizioNoleggio));

        AND CodB NOT IN (SELECT CodB
                         FROM NOLEGGIO
                         WHERE DataInizio <= :New.DataFineNoleggio AND
                              DataFine >= :NEW.DataInizioNoleggio);

        INSERT INTO NOLEGGIO (CodB, DataInizio, CodSocio, DataFine)
        VALUES (Y, :NEW.DataInizioNoleggio, :New.CodSocio, :NEW.DataFineNoleggio);
    ELSE
        -- Non ci sono barche da poter noleggiare
        INSERT INTO NOLEGGIO_IN_ATTESA (CodRich, DataRich, CodS, DataInizioNoleggio,
                                         DataFineNoleggio, Modello)
        VALUES (:NEW.CodRich, :NEW.DataRich, :NEW.CodS, :NEW.DataInizioNoleggio,
                :NEW.DataFineNoleggio, :NEW.Modello);
    END IF;
END;
```

Trigger 2b

Si scriva un trigger che gestisca *una rinuncia al noleggio*.

Si deve eliminare il noleggio dalla tabella NOLEGGIO. Si deve inoltre verificare se esistono richieste di noleggio in attesa nel periodo indicato e per il modello reso libero. In caso affermativo, si sceglie la richiesta in attesa da più tempo (per semplicità si supponga che ve ne sia sempre una sola), si definisce un nuovo noleggio per quella barca e si elimina la richiesta in attesa dalla tabella NOLEGGIO_IN_ATTESA.

```
CREATE OR REPLACE TRIGGER Rinuncia_Noleggjo
FOR EACH ROW
DECLARE
    DF DATE;
    M CHAR (10);
    A NUMBER;
    B NUMBER;
    C DATE;
    D DATE;
    Y NUMBER;
BEGIN
    /* Per verificare se esiste un noleggio in attesa che può essere soddisfatto nel
       periodo della rinuncia, debbo conoscere la data di fine del noleggio per cui c'è
       la rinuncia. Questa informazione viene persa se la verifica la faccio dopo
       l'eliminazione del noleggio dalla tabella noleggio, quindi la salvo
       preventivamente in una variabile */
    SELECT DataFine INTO DF
    FROM NOLEGGIO
    WHERE CodB = :NEW.CodB AND DataInizio = :NEW.DataInizioNoleggio;

    -- Ora posso cancellare il noleggio, di cui ho già preso la data di fine
    DELETE FROM NOLEGGIO
    WHERE CodB = :NEW.CodB AND DataInizio = :NEW.DataInizioNoleggio;

    -- Selezioniamo il modello della barca resa disponibile
    SELECT Modello INTO M
    FROM BARCA
    WHERE CodB = :NEW.CodB;

    /* Vediamo se esistono richieste di noleggio in attesa per il periodo indicato e per
       il modello reso libero */
    SELECT COUNT(*) INTO Y
    FROM NOLEGGIO_IN_ATTESA N
    WHERE N.DataInizioNoleggio >= :NEW.DataInizioNoleggio AND
          N.DataFineNoleggio <= DF AND Modello = M;

    IF (Y >= 1)
    THEN
        /* Almeno una richiesta di noleggio può essere soddisfatta, scelgo quella in
           attesa da più tempo */
        SELECT CodRich INTO A, CodSocio INTO B, DataInizioNoleggio INTO C,
               DataFineNoleggio INTO D
        FROM NOLEGGIO_IN_ATTESA N
        WHERE N.DataInizioNoleggio >= :NEW.DataInizioNoleggio AND
              N.DataFineNoleggio <= DF AND Modello = M AND
              N.DataRich = (SELECT MIN(DataRichiesta)
                           FROM NOLEGGIO_IN_ATTESA N2
                           WHERE N2.DataInizioNoleggio >= :NEW.DataInizioNoleggio AND
                                N2.DataFineNoleggio <= DF AND Modello = M);

        -- Inserisco un nuovo noleggio
        INSERT INTO NOLEGGIO (CodB, DataInizio, CodSocio, DataFine)
        VALUES (:NEW.CodB, C, B, D)

        -- che non è più in attesa, quindi lo levo
        DELETE FROM NOLEGGIO_IN_ATTESA
        WHERE CodRich = A;
    END IF;
END;
```

Trigger 3a

SERRA (CodSerra, Locazione, NumSensori)
SENSORE (CodSensore, GrandezzaMisurata, CodSerra, OffsetCritico)
LOG_EVENTI (CodE, TimeStamp, TipoEvento, CodSensore, Valore)
MISURA (CodSensore, TimeStamp, Valore)
SINTESI_GIORNO (Data, CodSensore, MediaValore)
NOTIFICA (CodN, CodSerra, Locazione, Messaggio)

Si scriva il trigger che gestisce l'arrivo di un nuovo evento di tipo misura (TipoEvento = 'M').
Per questo tipo di eventi, occorre, inserire una nuova misura nella tabella MISURA. Inoltre, occorre verificare se si è verificata una situazione critica nella serra dove si trova il sensore che ha effettuato la misura. Una situazione è critica se più di metà dei sensori presenti nella serra considerata hanno un valore eccessivo per l'ultima misura effettuata in ordine di tempo. Il valore di una misura è eccessivo se è maggiore della misura media del giorno per quel sensore di una quantità superiore a OffsetCritico. La misura media del giorno è memorizzata nella tabella SINTESI_GIORNO per ogni sensore. Questa tabella è mantenuta automaticamente aggiornata da un trigger non considerato nella presente applicazione. Per estrarre la data dall'attributo TimeStamp, si utilizzi una generica funzione DATE(TimeStamp).
Se si verifica una situazione critica, occorre inserire una richiesta di notifica nella tabella notifica. Il codice identificativo univoco CodN è un contatore che deve essere incrementato ogni volta che è inserita una nuova notifica.

```
CREATE OR REPLACE TRIGGER Nuova_Misura
AFTER INSERT ON LOG_EVENTI
FOR EACH ROW
WHEN (:NEW.TipoEvento = 'M')
DECLARE
    CS NUMBER; NS NUMBER; NC NUMBER; CN NUMBER;
    LOC VARCHAR(10);
BEGIN
    INSERT INTO MISURA(CodSensore, TimeStamp, Valore)
    VALUES (:NEW.CodSensore, :NEW.TimeStamp, :NEW.Valore);

    -- Leggo le informazioni sulla serra in cui si trova il sensore
    SELECT S.CodSerra, S.Locazione, S.NumSensori INTO CS, LOC, NS
    FROM SERRA S, SENSORE SS
    WHERE S.CodSerra = SS.CodSerra AND SS.CodSensore = :NEW.CodSensore;

    /* Conto i sensori in quella serra che hanno rilevato una criticità (un valore
       eccessivo per l'ultima misura) */
    SELECT COUNT(*) INTO NC
    FROM MISURA M, SINTESI_GIORNO SG, SENSORE SS
    WHERE M.CodSensore = SG.CodSensore AND M.CodSensore = SS.CodSensore AND
          Data = DATE(M.TimeStamp) AND Valore > MediaValori+OffsetCritico AND
          CodSerra = CS AND DATE(M.TimeStamp) = DATE(:NEW.TimeStamp) AND
          - Il giorno deve essere lo stesso di quello inserito
          M.TimeStamp = (SELECT MAX(TimeStamp)
                        FROM MISURA M1
                        WHERE M1.CodSensore = M.CodSensore);

    IF (NC > NS/2)
    THEN
        SELECT MAX(CodN) INTO CN
        FROM NOTIFICA;

        IF (CN IS NULL)
        THEN
            CN := 1
        ELSE
            CN := CN + 1
        END IF;

        INSERT INTO NOTIFICA(CodN, CodSerra, Locazione, Messaggio)
        VALUES (CN, CS, LOC, 'Situazione Critica');
    END IF;
END;
```

Trigger 3b

Si scriva il trigger per la verifica di correttezza (ed eventuale correzione) di un nuovo evento di tipo misura (TipoEvento = 'M') registrato nella tabella LOG_EVENTI.

Per i sensori per cui la grandezza misurata è la temperatura (GrandezzaMisurata = 'Temperatura'), il valore misurato non può essere inferiore a -50. Se lo è, si deve assegnare il valore -50 alla temperatura per l'evento registrato in LOG_EVENTI.

```
CREATE OR REPLACE TRIGGER Correggi_Valore
BEFORE INSERT on LOG_EVENTI
FOR EACH ROW
WHEN :NEW.TipoEvento = 'M'
DECLARE
    GRANDEZZA VARCHAR(10);
BEGIN
    SELECT GrandezzaMisurata INTO GRANDEZZA
    FROM SENSORE WHERE CodSensore = :NEW.CodSensore;

    IF (GRANDEZZA = 'Temperatura' AND :NEW.Valore < -50)
    THEN
        :NEW.Valore = -50
    END IF;
END;
```

Trigger 3c

Scrivere il trigger che implementi il seguente vincolo:

Una serra non può avere più di 200 sensori con valore OffsetCritico > 50

```
CREATE OR REPLACE TRIGGER VerificaSerra
AFTER INSERT OR UPDATE OF OffsetCritico OR UPDATE OF CodSerra ON SENSORE
DECLARE
    X NUMBER;
BEGIN
    -- Contiamo il numero di serre che violano il vincolo
    SELECT COUNT(*) INTO X
    FROM SERRA
    WHERE CodSerra IN (SELECT CodSerra
                        FROM SENSORE
                        WHERE OffsetCritico > 50
                        GROUP BY CodSerra
                        HAVING COUNT(*) > 200);

    IF (X <> 0)
    THEN
        RAISE_APPLICATION_ERROR(-20500, 'Vincolo Violato');
    END IF;
END;
```

Trigger 4

Sono date le relazioni seguenti (le chiavi primarie sono sottolineate, gli attributi opzionali hanno l'asterisco)

STUDENTE(Matricola, NomeStudente, AnnoImmatricolazione, CorsoLaurea)
CORSO(CodCorso, NomeCorso, NumeroCrediti)
ESAMI_SOSTENUTI(CodCorso, Matricola, Data, Voto)
GRADUATORIA_STUDENTI(Matricola, Punteggio)
BORSE_STUDIO_ASSEGNATE(CodBorsa, Matricola, NumeroOre)
DOMANDA_INSERIMENTO_GRADUATORIA(Matricola, DataDomanda)
OFFERTA_BORSA_STUDIO(CodBorsa, CodCorso, NumeroOre)
NOTIFICA_INFORMAZIONI(CodN, CodBorsa, Matricola*, Messaggio)

Si scrivano i trigger necessari per gestire le seguenti attività per l'assegnazione automatica di borse di studio per attività di supporto alla didattica. Gli studenti interessati a usufruire di borse di studio sono inseriti in una graduatoria. Quando viene offerta una borsa di studio, si seleziona dalla graduatoria lo studente a cui assegnarla. Ad uno stesso studente possono essere assegnate anche più borse di studio, ma complessivamente lo studente non può svolgere più di 150 ore sulle borse di studio assegnate. Di seguito sono descritte nel dettaglio le modalità e i vincoli per ciascuna attività.

Inserimento di uno studente in graduatoria. Lo studente presenta la domanda per l'assegnazione di borse di studio. La domanda viene accettata se lo studente non è già presente in graduatoria (tabella GRADUATORIA) ed ha acquisito almeno 120 crediti sugli esami superati. Se almeno uno dei requisiti non è soddisfatto, la domanda viene annullata. Altrimenti si deve aggiornare la graduatoria, assegnando allo studente un punteggio dato dal prodotto della media dei voti per gli esami superati (votazione maggiore o uguale a 18), per il numero di anni di iscrizione dello studente al corso di laurea (si assuma che l'anno corrente sia fornito dalla variabile SYSDATE).

```
CREATE OR REPLACE TRIGGER Inserisci_in_graduatoria
AFTER INSERT ON DOMANDA_INSERIMENTO_GRADUATORIA
FOR EACH ROW
DECLARE
    N NUMBER; N_CREDITI NUMBER;
    MEDIA NUMBER; ANNO_IMMATR NUMBER; ANNI_ISCR NUMBER;
BEGIN
    -- Vediamo se lo studente è già in graduatoria
    SELECT COUNT(*) INTO N
    FROM GRADUATORIA_STUDENTI
    WHERE MATRICOLA = :NEW.MATRICOLA;

    IF (N = 0)
    THEN
        /* Lo studente non è in graduatoria.
        Calcoliamo il numero dei crediti totali degli esami
        che ha superato e la media*/
        SELECT SUM(NumeroCrediti), AVG(VOTO) INTO N_CREDITI, MEDIA
        FROM CORSO C, ESAMI_SOSTENUTI E
        WHERE C.CODCORSO = E.CODCORSO AND
              E.MATRICOLA = :NEW.MATRICOLA AND
              VOTO >= 18;

        IF (N_CREDITI >= 120)
        THEN
            -- Vediamo il numero di anni di iscrizione dello studente
            SELECT ANNOIMMATRICOLAZIONE INTO ANNO_IMMATR
            FROM STUDENTE
            WHERE MATRICOLA = :NEW.MATRICOLA;

            ANNI_ISCR := SYSDATE - ANNO_IMMATR;

            INSERT INTO GRADUATORIA_STUDENTI (MATRICOLA, PUNTEGGIO)
            VALUES (:NEW.MATRICOLA, MEDIA*ANNI_ISCR);
        END IF;
    ELSE
        RAISE_APPLICATION_ERROR(-20001, 'Vincolo violato');
    END IF;
END;
```

Assegnazione di una borsa di studio per un corso. Quando viene offerta una borsa di studio per un corso, si seleziona dalla graduatoria lo studente a cui assegnare la borsa. Viene selezionato lo studente con punteggio più alto tra gli studenti che soddisfano i seguenti requisiti: lo studente ha superato l'esame per il corso per cui è offerta la borsa di studio, e lo studente complessivamente non svolge più di 150 ore sulle borse di studio assegnate. Si assuma che ci sia sempre al più un solo studente che soddisfi tutti i requisiti. Occorre notificare l'esito dell'operazione, sia che la borsa di studio sia stata assegnata, sia che non sia possibile assegnare la borsa (in questo caso, la matricola sarà NULL). L'attributo CodN è un contatore che viene incrementato ogni volta che viene inserita una nuova notifica. Se la borsa di studio è assegnata, si deve aggiornare la tabella BORSE_STUDIO_ASSEGNATE.

```
CREATE OR REPLACE TRIGGER Assegna_Borsa
AFTER INSERT ON OFFERTA_BORSA_STUDIO
FOR EACH ROW
DECLARE
    VINCITORE NUMBER;
    N NUMBER;
BEGIN
    SELECT E.MATRICOLA INTO VINCITORE
    FROM OFFERTA_BORSA_STUDIO O, GRADUATORIA_STUDENTI G, ESAMI_SOSTENUTI E
    WHERE G.MATRICOLA = E.MATRICOLA AND O.CODCORSO = E.CODCORSO AND
          E.VOTO >= 18 AND O.CODBORSA = :NEW.CODBORSA AND
          PUNTEGGIO = (SELECT MAX(PUNTEGGIO)
                      FROM OFFERTA_BORSA_STUDIO O2, GRADUATORIA_STUDENTI G2, ESAMI_SOSTENUTI E2
                      WHERE G2.MATRICOLA = E2.MATRICOLA AND O2.CODCORSO = E2.CODCORSO AND
                            E2.VOTO >= 18 AND
                            150 - :NEW.NUMEROORE >= (SELECT SUM(NUMEROORE)
                                                         FROM BORSE_STUDIO_ASSEGNATE B
                                                         WHERE B.MATRICOLA = :NEW.MATRICOLA; )
                      GROUP BY G2.MATRICOLA; );

    SELECT MAX(CODN) INTO N
    FROM NOTIFICA_INFORMAZIONI;

    IF(VINCITORE IS NOT NULL)
    THEN
        INSERT INTO BORSE_STUDIO_ASSEGNATE(CODBORSA, MATRICOLA, NUMEROORE)
        VALUES (:NEW.CODBORSA, VINCITORE, :NEW.NUMEROORE);

        INSERT INTO NOTIFICA_INFORMAZIONI(CODN, CODBORSA, MATRICOLA, MESSAGGIO)
        VALUES (N+1, :NEW.CODBORSA, VINCITORE, 'Borsa assegnata');
    ELSE
        INSERT INTO NOTIFICA_INFORMAZIONI(CODN, CODBORSA, MATRICOLA, MESSAGGIO)
        VALUES (N+1, :NEW.CODBORSA, NULL, 'Borsa non assegnata');
    END IF;
END;
```

```

CREATE OR REPLACE TRIGGER Assegna_Borsa
AFTER INSERT ON OFFERTA_BORSA_STUDIO
FOR EACH ROW
DECLARE
    X NUMBER;
    N NUMBER;
    MATR_STUD NUMBER;
BEGIN
    SELECT MAX(PUNTEGGIO) INTO X
    FROM GRADUATORIA_STUDENTI
    WHERE MATRICOLA IN (SELECT MATRICOLA
                        FROM ESAMI_SOSTENUTI
                        WHERE CODCORSO = :NEW.CODCORSO AND VOTO >= 18)
    AND MATRICOLA NOT IN (SELECT MATRICOLA
                        FROM BORSE_STUDIO_ASSEGNATE
                        GROUP BY MATRICOLA
                        HAVING SUM(NUMEROORE) + :NEW.NUMEROORE > 150);

    SELECT MAX(CODN) INTO N
    FROM NOTIFICA;

    IF(N IS NULL)
    THEN
        N = 0;
    END IF;

    IF(X IS NOT NULL)
    THEN
        SELECT MATRICOLA INTO MATR_STUD
        FROM GRADUATORIA_STUDENTI
        WHERE PUNTEGGIO = X
        And MATRICOLA IN (SELECT MATRICOLA
                        FROM ESAMI_SOSTENUTI
                        WHERE CODCORSO = :NEW.CODCORSO AND VOTO >= 18)
        AND MATRICOLA NOT IN (SELECT MATRICOLA
                        FROM BORSE_STUDIO_ASSEGNATE
                        GROUP BY MATRICOLA
                        HAVING SUM(NUMEROORE) + :NEW.NUMEROORE > 150);

        INSERT INTO BORSE_STUDIO_ASSEGNATE(CODBORSA, MATRICOLA, NUMEROORE)
        VALUES(:NEW.CODBORSA, MATR_STUD, :NEW.NUMEROORE);

        INSERT INTO NOTIFICA_INFORMAZIONI(CODN, CODBORSA, MATRICOLA, MESSAGGIO)
        VALUES(N+1, :NEW.CODBORSA, MATR_STUD, 'Borsa assegnata');
    ELSE
        INSERT INTO NOTIFICA_INFORMAZIONI(CODN, CODBORSA, MATRICOLA, MESSAGGIO)
        VALUES(N+1, :NEW.CODBORSA, NULL, 'Borsa non assegnata');
    END IF;
END;

```

Trigger Libreria

Sono date le relazioni seguenti (le chiavi primarie sono sottolineate, gli attributi opzionali hanno l'asterisco):

RICHIESTA_ACQUISTO_LIBRI(CodRichiestaAcquisto, CodCliente, Data, ISBN, NumCopieRichieste)
CATALOGO_LIBRI(ISBN, Titolo, Autore, Prezzo)
MAGAZZINO_LIBRI(ISBN, NumCopieDisponibili)
NOTIFICA_ACQUISTO(CodN, Data, Messaggio, Titolo, Autore, Prezzo)
RICHIESTA_INFORMAZIONI_AUTORE(CodRichiestaInformazioni, CodCliente, Data, Autore)
VENDITE_LIBRI(ISBN, NumCopieVendute)
NOTIFICA_INFORMAZIONI(CodRichiestaInformazioni, CodCliente, ISBN, Titolo, Autore)

Si scrivano i trigger necessari per gestire le seguenti due attività in un sito per la vendita on-line di libri:

(1) richiesta di acquisto di libri e (2) richiesta di informazioni su un autore. Il sito dispone di un catalogo dei libri venduti presso il sito (tabella CATALOGO LIBRI). La prima attività è la gestione della richiesta di acquisto di un libro a catalogo da parte di un cliente (inserimento nella tabella RICHIESTA ACQUISTO LIBRI). Nella richiesta di acquisto il cliente specifica il libro e il numero di copie che desidera acquistare. Se tutte le copie richieste sono disponibili nel magazzino, la richiesta di acquisto può essere completata. In questo caso viene aggiornata la disponibilità nel magazzino per il libro, e si notifica al cliente l'avvenuto acquisto (Messaggio da notificare: "Acquisto concluso").

Altrimenti si notifica al cliente il mancato acquisto (Messaggio da notificare: "Acquisto non concluso").

Si consideri che la tabella MAGAZZINO LIBRI contiene, tra i libri disponibili nel catalogo, solo quelli per cui è disponibile almeno una copia nel magazzino. I libri per cui non ci sono copie disponibili non sono presenti in MAGAZZINO LIBRI. L'attributo CodN nella tabella NOTIFICA ACQUISTO è un contatore che deve essere incrementato ogni volta che viene effettuata una nuova notifica in uno stesso giorno.

```
CREATE OR REPLACE TRIGGER Richiesta_Acquisto
AFTER INSERT ON RICHIESTA_ACQUISTO_LIBRI
FOR EACH ROW
DECLARE
    N NUMBER; TITOLO VARCHAR(10); N_MAX NUMBER; N_COPIE NUMBER;
    AUTORE VARCHAR(10); PREZZO NUMBER;
BEGIN
    -- Vediamo se il libro è in magazzino
    SELECT COUNT(*) INTO N
    FROM MAGAZZINO_LIBRI
    WHERE ISBN = :NEW.ISBN;

    IF (N > 0)
    THEN
        -- Questa variabili ci serviranno dopo
        SELECT MAX(CodN) INTO N_MAX
        FROM NOTIFICA_ACQUISTO;

        SELECT C.TITOLO INTO TITOLO, C.AUTORE INTO AUTORE, C.PREZZO INTO PREZZO
        FROM CATALOGO_LIBRI C
        WHERE C.ISBN = :NEW.ISBN;

        -- Vediamo quante copie ci sono in magazzino
        SELECT NumCopieDisponibili INTO N_COPIE
        FROM MAGAZZINO_LIBRI
        WHERE ISBN = :NEW.ISBN;

        IF (N_COPIE >= :NEW.NumCopieDisponibili)
        THEN
            INSERT INTO NOTIFICA_ACQUISTO(CodN, Data, Messaggio, Titolo, Autore, Prezzo)
            VALUES(N_MAX+1, SYS_DATE, 'Acquisto Concluso', TITOLO, AUTORE, PREZZO)
            IF (N_COPIE - :NEW.NumCopieRichieste == 0)
            THEN
                DELETE FROM MAGAZZINO_LIBRI
                WHERE ISBN = :NEW.ISBN;
            ELSE
                UPDATE MAGAZZINO_LIBRI
                SET NumCopieDisponibili = NumCopieDisponibili - N_COPIE
                WHERE ISBN = :NEW.ISBN;
            END IF;
        ELSE
            INSERT INTO NOTIFICA_ACQUISTO(CodN, Data, Messaggio, Titolo, Autore, Prezzo)
            VALUES(N_MAX+1, SYS_DATE, 'Acquisto non Concluso', TITOLO, AUTORE, PREZZO)
        END IF;
    ELSE
        INSERT INTO NOTIFICA_ACQUISTO(CodN, Data, Messaggio, Titolo, Autore, Prezzo)
        VALUES(N_MAX+1, SYS_DATE, 'Acquisto non Concluso', TITOLO, AUTORE, PREZZO)
    END IF;
END;
```

La seconda attività consiste nel fornire informazioni ai clienti sui libri venduti presso il sito, per supportare i clienti nella scelta del libro da acquistare. Quando un cliente richiede informazioni su un autore (inserimento nella tabella RICHIESTA INFORMAZIONI AUTORE), viene selezionato il libro che ha venduto più copie, tra i libri scritti dall'autore e per cui è disponibile almeno una copia nel magazzino. Si assuma che ci sia sempre al più un unico libro che soddisfa tutti i requisiti. Le informazioni sul libro selezionato sono notificate al cliente (inserimento nella tabella NOTIFICA INFORMAZIONI).

```
CREATE OR REPLACE TRIGGER Richiesta_Informazioni
AFTER INSERT ON RICHIESTA_INFORMAZIONI_AUTORE
FOR EACH ROW
DECLARE
    ID_ISBN VARCHAR(20);
    N NUMBER;
    TITOLO VARCHAR(20);
    AUTORE VARCHAR(20);
BEGIN
    -- Selezioniamo tra i libri dell'autore quello che ha venduto più copie
    SELECT V.ISBN INTO ID_ISBN
    FROM VENDITE_LIBRI V, CATALOGO_LIBRI C, MAGAZZINO_LIBRI M
    WHERE V.ISBN = C.ISBN AND C.Autore = :NEW.Autore AND M.ISBN = V.ISBN AND
    V.NumCopieVendute = (SELECT MAX(NumCopieVendute)
                        FROM VENDITE_LIBRI V1, CATALOGO_LIBRI C1, MAGAZZINO_LIBRI M1
                        WHERE V1.ISBN = C1.ISBN AND C1.Autore = :NEW.Autore AND
                        M1.ISBN = V.ISBN);

    -- Oppure
    SELECT TOP 1 V.ISBN INTO ID_ISBN
    FROM VENDITE_LIBRI V, CATALOGO_LIBRI C, MAGAZZINO_LIBRI M
    WHERE V.ISBN = C.ISBN AND C.Autore = :NEW.Autore AND M.ISBN = V.ISBN
    ORDER BY NumCopieVendute DESC;

    SELECT C.Titolo INTO TITOLO, C.Autore INTO AUTORE
    FROM CATALOGO_LIBRI C
    WHERE C.ISBN = ID_ISBN;

    SELECT MAX(CodRichiestaInformazioni) INTO N
    FROM NOTIFICA_INFORMAZIONI;

    INSERT INTO NOTIFICA_INFORMAZIONI (CodRichiestaInformazioni, CodCliente, ISBN, Titolo, Autore)
    VALUES (N+1, :NEW.CodCliente, ID_ISBN, TITOLO, AUTORE);
END;
```

Trigger Piazzola

Sono date le relazioni seguenti (le chiavi primarie sono sottolineate, gli attributi opzionali sono indicati con *).

PIAZZOLA(CodP, StatoPiazzola, TargaVettura*, CodCliente*, TimeStampIngresso*)
CLIENTI(CodCliente, Nome, Recapito, QuotaAssociativa, DataScadenzaQuotaAssociativa)
RICHIESTA_PIAZZOLA(CodRichiesta, TargaVettura, CodCliente, TimeStampIngresso)
RILASCIO_PIAZZOLA(CodRilascio, TargaVettura, TimeStampUscita)
NOTIFICA(CodN, CodCliente, TargaVettura, IntervalloTempoParcheggio)
RICHIESTA_STATISTICHE(CodStatistica, CodCliente)
NOTIFICA_STATISTICHE(CodStatistica, CodCliente, NumeroParcheggi, NumeroVettureDiverse, DurataMediaParcheggio)

Si vogliono gestire alcune attività presso un parcheggio custodito per autovetture. L'area di parcheggio è organizzata in piazzole, ciascuna identificata da un codice numerico. Le informazioni relative a ciascuna piazzola sono memorizzate nella tabella PIAZZOLA. L'attributo StatoPiazzola indica se la piazzola è in quel momento occupata da una vettura (valore Occupato) oppure è libera (valore Libero). Se la piazzola è occupata da una vettura, la tabella PIAZZOLA contiene la targa della vettura, il codice del cliente e l'istante temporale in cui la vettura è entrata nel parcheggio; gli attributi che memorizzano tali informazioni contengono il valore NULL se la piazzola è libera. Si scrivano i trigger per gestire le seguenti attività in un sistema per la gestione automatica delle piazzole del parcheggio.

(1) Assegnazione di una piazzola ad una vettura. Quando un cliente vuole accedere al parcheggio, viene inserito un nuovo record nella tabella RICHIESTA_PIAZZOLA. L'accesso al parcheggio è permesso solo ai clienti per cui la quota associativa è ancora attiva nell'istante i cui viene effettuata la richiesta. Per tali clienti, si deve selezionare una piazzola libera. Le piazzole sono assegnate per valore di codice piazzola crescente. Pertanto se ci sono più piazzole disponibili, deve essere selezionata quella che ha il valore inferiore di codice identificativo. La tabella PIAZZOLA deve essere aggiornata con le informazioni relative alla vettura che vi è stata assegnata. Se nessuna piazzola è disponibile, o se la quota associativa del cliente che effettua la richiesta è scaduta, la richiesta di assegnazione di una piazzola deve essere annullata.

```
CREATE OR REPLACE TRIGGER RichiestaPiazzola
AFTER INSERT ON RICHIESTA_PIAZZOLA
FOR EACH ROW
DECLARE
    SCADENZA NUMBER;
    N NUMBER;
    CODPMIN NUMBER;
BEGIN
    -- Vediamo quando scade la quota associativa del cliente
    SELECT DataScadenzaQuotaAssociativa INTO SCADENZA
    FROM CLIENTI C
    WHERE C.CodCliente = :NEW.CodCliente;

    IF (SCADENZA < SYS_DATE)
    THEN
        raise_application_error(-12345, 'Quota associativa scaduta');
    ELSE
        -- Vediamo se ci sono piazzole disponibili
        SELECT COUNT(*) INTO N
        FROM PIAZZOLA
        WHERE StatoPiazzola = 'Libero';

        IF (N == 0)
        THEN
            raise_application_error(-12345, 'Nessuna piazzola libera');
        ELSE
            -- Prendiamo la piazzola con codice più piccolo
            SELECT TOP 1 CodP INTO CODPMIN
            FROM PIAZZOLA
            WHERE StatoPiazzola = 'Libero'
            ORDER BY CodP;

            -- Oppure
            SELECT CodP INTO CODPMIN
            FROM PIAZZOLA
            WHERE CodP = (SELECT MIN(CodP)
                        FROM PIAZZOLA
                        WHERE StatoPiazzola = 'Libero');

            UPDATE PIAZZOLA
            SET StatoPiazzola = 'Occupato',
                TargaVettura = :NEW.TargaVettura,
                CodCliente = :NEW.CodCliente,
                TimeStampIngresso = :NEW.TimeStampIngresso
            WHERE CodP = CODPMIN;
        END IF;
    END IF;
END;
```

(2) Rilascio di una piazzola precedentemente assegnata ad una vettura. Quando una vettura lascia il parcheggio viene inserito un nuovo record nella tabella RILASCIO PIAZZOLA. Questo evento deve essere gestito aggiornando nella tabella PIAZZOLA tutte le informazioni relative alla piazzola precedentemente occupata dalla vettura. Inoltre si deve notificare al cliente l'intervallo di tempo in cui complessivamente ha occupato il parcheggio (inserimento nella tabella NOTIFICA). L'attributo CodN è un contatore che viene incrementato ogni volta che viene inserita una nuova notifica.

```
CREATE OR REPLACE TRIGGER RilascioPiazzola
AFTER INSERT ON RILASCIO_PIAZZOLA
FOR EACH ROW
DECLARE
    CODPIAZZOLA NUMBER;
    TIMESTAMPINGRESSO NUMBER;
    CODCLIENTE NUMBER;
    TARGA VARCHAR(10);
    N NUMBER;
BEGIN
    -- Mi serve il codice della piazzola in cui la macchina era parcheggiata,
    -- la targa della macchina, il codice cliente e il TimeStamp di ingresso
    SELECT CodP INTO CODPIAZZOLA, TimeStampIngresso INTO TIMESTAMPINGRESSO,
           TargaVettura INTO TARGA, CodCliente INTO CODCLIENTE
    FROM PIAZZOLA
    WHERE TargaVettura = :NEW.TargaVettura;

    -- Aggiorniamo la tabella PIAZZOLA
    UPDATE PIAZZOLA
    SET StatoPiazzola = 'Libero',
        TargaVettura = NULL,
        CodCliente = NULL,
        TimeStampIngresso = NULL
    WHERE CodP = CODPIAZZOLA;

    SELECT MAX(CodN) INTO N
    FROM NOTIFICA;

    INSERT INTO NOTIFICA(CodN, CodClient, TargaVettura, IntervalloTempoParcheggio)
    VALUES(N+1, CODCLIENTE, TARGA, :NEW.TimeStampUscita - TIMESTAMPINGRESSO);
END;
```

(3) Gestione della quota associativa al parcheggio (QuotaAssociativa). Possono utilizzare il parcheggio solo i clienti che sono convenzionati. Il valore della quota associativa può variare in base al cliente, ma non può comunque essere inferiore a 300 euro. Se un valore di quota associativa inferiore a 300 euro è inserito nella tabella CLIENTE, l'attributo QuotaAssociativa deve essere assegnato a 300. Si scriva il trigger per la gestione del vincolo di integrità.

```
CREATE OR REPLACE TRIGGER GestioneQuota
BEFORE INSERT ON CLIENTI
FOR EACH ROW
BEGIN
    IF(:NEW.QuotaAssociativa < 300)
    THEN
        :NEW.QuotaAssociativa = 300;
    END IF;
END;
```

(4) Calcolo statistiche sull'utilizzo parcheggio. Periodicamente sono calcolate delle statistiche relative all'utilizzo del parcheggio da parte dei clienti. L'inserimento di un nuovo record nella tabella RICHIESTA STATISTICHE indica la richiesta delle statistiche per uno specifico cliente. Devono essere calcolati il numero di parcheggi effettuati dal cliente, il numero di vetture diverse utilizzate, e il tempo medio di parcheggio. Tali informazioni devono essere ricavate dalla tabella NOTIFICA. Le statistiche calcolate devono essere inserite nella tabella NOTIFICA STATISTICHE.

```
CREATE OR REPLACE TRIGGER CalcoloStatistiche
AFTER INSERT ON RICHIESTA_STATISTICHE
FOR EACH ROW
DECLARE
    NPARCHEGGI NUMBER;
    NVETTURE NUMBER;
    TEMPOTOT NUMBER;
    N NUMBER;
BEGIN
    SELECT COUNT(*) INTO NPARCHEGGI
    FROM NOTIFICA N
    WHERE N.CodN = :NEW.CodCliente;

    SELECT DISTINCT TargaVettura INTO NVETTURE
    FROM NOTIFICA N
    WHERE N.CodN = :NEW.CodCliente;

    SELECT SUM(IntervalloTempoParcheggio) INTO TEMPOTOT
    FROM NOTIFICA N
    WHERE N.CodN = :NEW.CodCliente;

    SELECT MAX(CodStatistica) INTO N
    FROM NOTIFICA_STATISTICHE;

    INSERT INTO NOTIFICA_STATISTICHE(CodStatistica, CodCliente, NumeroParcheggi,
                                     NumeroVettureDiverse, DurataMediaParcheggio)
    VALUES (N+1, :NEW.CodCliente, NPARCHEGGI, NVETTURE, TEMPOTOT/NPARCHEGGI);
END;
```