UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Dipartimento di Ingegneria Elettrica e Tecnologie dell'Informazione Corso di Laurea in Informatica

Insegnamento di Basi di Dati e Sistemi Informativi I Anno accademico 2018/2019

Progettazione e sviluppo di una base di dati relazionale per la gestione di una Repository Software.

Autore:

Ciro de Cristofaro

N86002370

ci.decristofaro@studenti.unina.it

Docenti:

Prof.Adriano Peron

Prof.Alessandro De Luca

INDICE

Descrizione del progetto

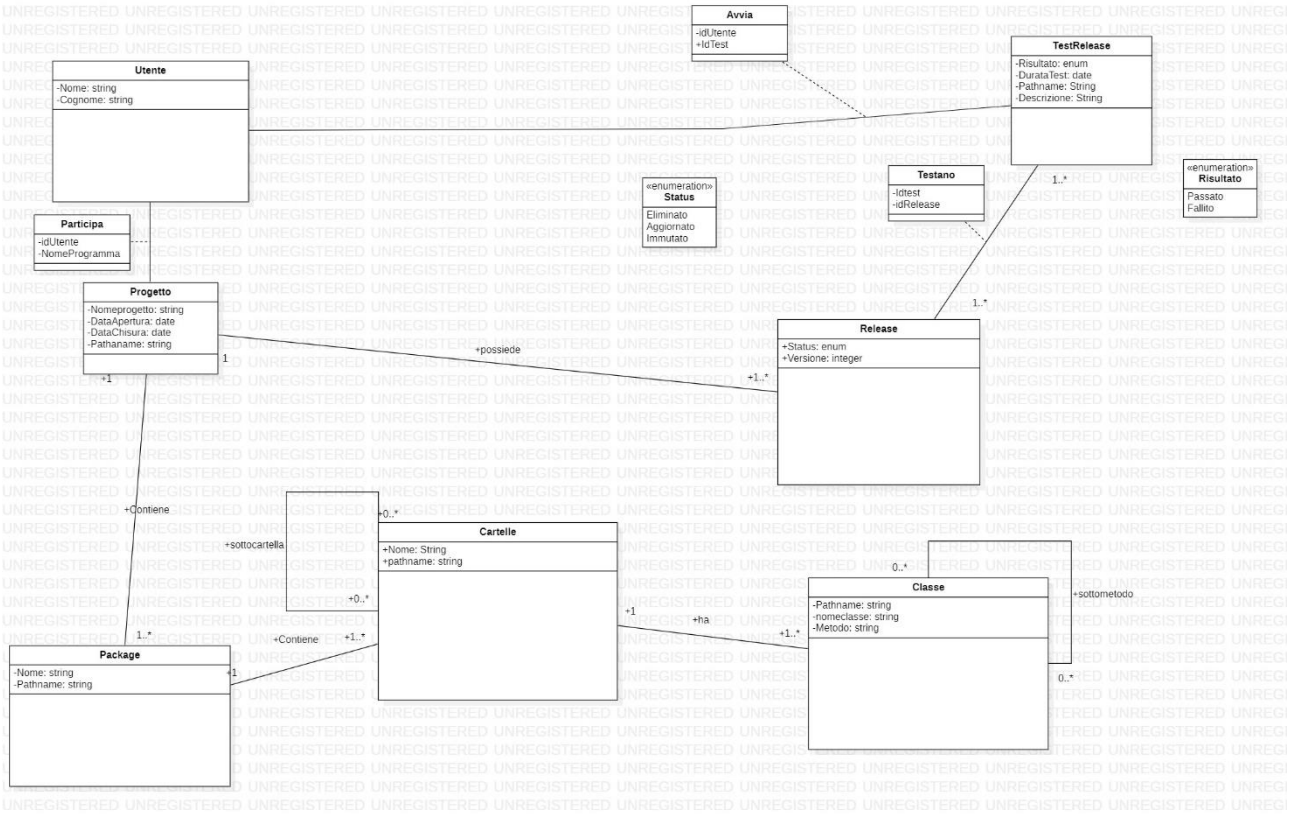
a. Descrizione sintetica

Si progetterà ed implementerà una base di dati relazione che possa essere d'ausilio alla gestione e memorizzazione di un repository di software.
Il database conterrà tutte le informazioni necessarie del ciclo di vita di un progetto software.
Saranno implementate varie funzionalità che prevedono: la creazione dell'utente, del progetto, delle classi, dei package, la chiusura di un progetto o di una classe e i vari Test effettuati.

Progettazione Concettuale

In questo capitolo inizia la progettazione della base di dati al livello di astrazione più alto.
Dal risultato dell'analisi dei requisiti che devono essere soddisfatti si avverrà ad uno schema concettuale indipendente dalla struttura dei dati e dall'implementazione fisica. In tale schema concettuale, che verrà rappresentato usando un class diagram UML, si evidenzieranno le entità rilevanti ai fini della rappresentazione dei dati e le relazioni che intercorrono tra esse.

a. Class diagram



b. Class diagram ristrutturato

Al fine di rendere il class diagram idoneo alla traduzione in schemi relazionali e di migliorare l'efficienza dell'implementazione si procede alla ristrutturazione dello stesso. Al termine del procedimento il class diagram non conterrà attributi strutturati, attributi multipli e gerarchie di specializzazione.

Analisi delle ridondanze

Abbiamo deciso di unire le due entità package e cartella in modo di non avere ridondanze

Analisi degli identificativi

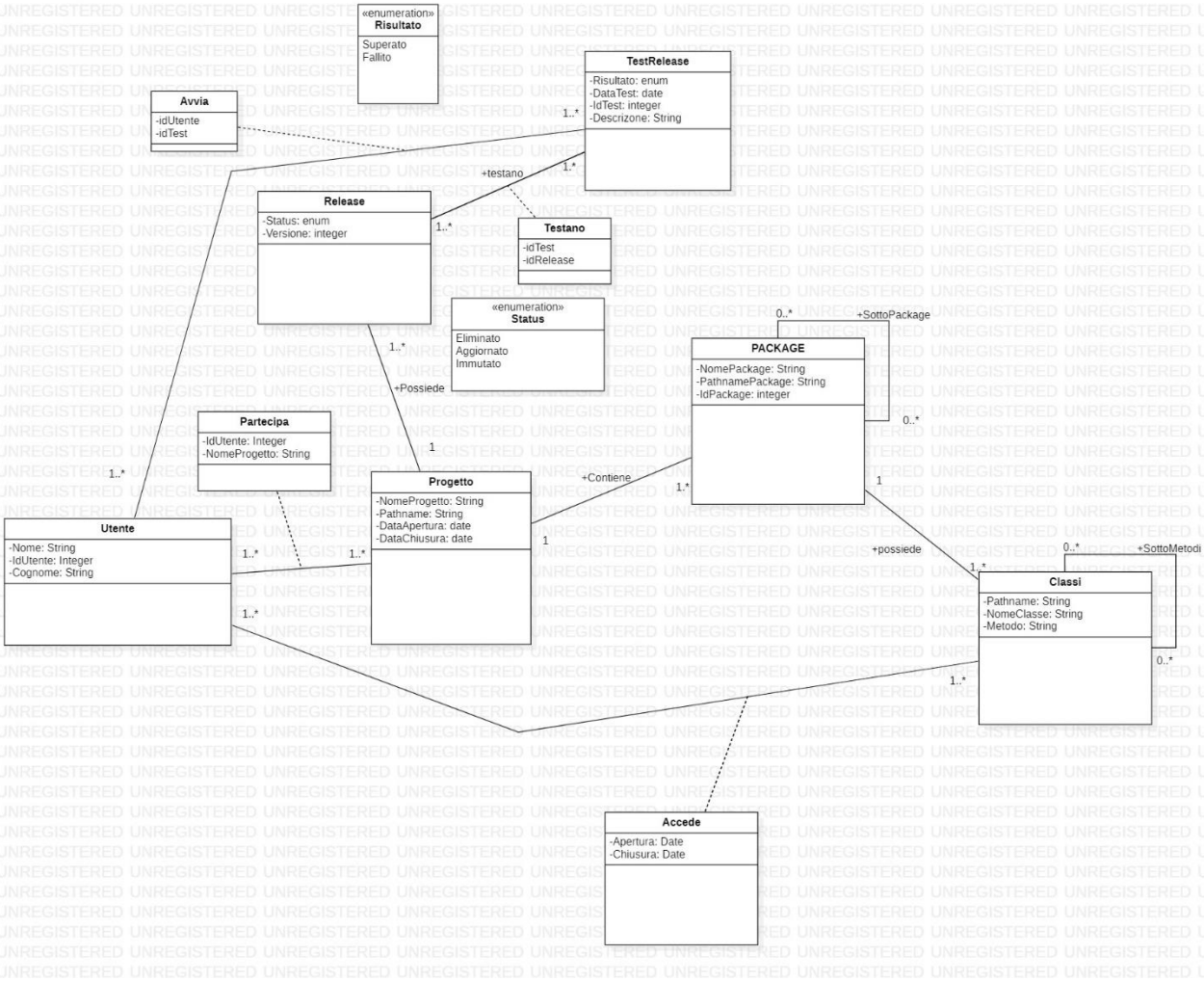
Risulta conveniente ai fini dell'efficienza l'introduzione di chiavi "tecniche" in ogni - o quasi - entità. Tali chiavi tecniche altro non saranno che identificativi numerici che permetteranno di discriminare con maggiore facilità le istanze.

Rimozione degli attributi multipli

Non ci sono attributi multipli

Rimozione delle gerarchie di specializzazione

Non sono presenti gerarchie di specializzazione


«enumeration»
Risultato
Superato
Fallito
TestRelease
-Risultato: enum
-DataTest: date
-IdTest: integer
-Descrizone: String
Avvia
-idUtente
-idTest
1..*
+testano
1..*
Release
-Status: enum
-Versione: integer
1..*
Testano
-idTest
-idRelease
«enumeration»
Status
Eliminato
Aggiornato
Immutato
1..*
+Possiede
0..*
+SottoPackage
PACKAGE
-NomePackage: String
-PathnamePackage: String
-IdPackage: integer
0..*
Partecipa
-IdUtente: Integer
-NomeProgetto: String
1..*
1
+Contiene
1..*
1
Progetto
-NomeProgetto: String
-Pathname: String
-DataApertura: date
-DataChiusura: date
1
Utente
-Nome: String
-IdUtente: Integer
-Cognome: String
1..*
1..*
1..*
+possiede
1..*
0..*
+SottoMetodi
Classi
-Pathname: String
-NomeClasse: String
-Metodo: String
0..*
1..*
Accede
-Apertura: Date
-Chiusura: Date

c. Dizionario dei dati

Dizionario delle classi

Classe	Descrizione	Attributi
Utente	Descrittore di ciascun utente presente nel database	**IDUTENTE**(integer): chiave primaria, identifica univocamente un utente; **NOME**(string):nome dell'utente **COGNOME**(string):cognome dell'utente
PROGETTO	Descrittore dei progetti presente nel database	**NOMEPROGETTO**(string)**:** Chiave primaria, indentifica il nome dei vari progetti **PATHNAME**(string): contiene il percorso in cui verrà salvato il progetto. **DATAAPERTURA**(date):contiene informazioni sulla creazione del progetto **DATACHIUSURA**(date):contiene informazione sulla chiusura del progetto
PACKAGE	Descrittore dei package presente nel database	**IDPACKAGE**(integer): Chiave primaria, identifica i vari package contenuti nei progetti **PATHNAME**(string):contiene il percorso in cui verrà memorizzato il package **NOMEPACKAGE**(string):Nome identificativo dei vari package
CLASSI	Descrittore delle classi presenti nel database	**IDCLASSE**(integer):Chiave primaria ,identifica le varie classi presenti nei progetti **NOMECLASSE**(string):Nome identificativo delle varie classi **METODO**(String): contiene informazioni rilevanti ai metodi di una classe
RELEASE	Descrittore delle release presenti nel database	**STATUS**(status):identifica lo stato delle varie release **VERSIONE**(integer):identifica la versione dei progetti **IDRELEASE**(integer): Chiave primaria,identifica univocamente le varie release

TESTRELEASE	Descrittore dei test presenti nel database	**RISULTATO**(risultato):identifica il risultato di un test effettuato **DATATEST**(Date): contiene la data del test effettuata **IDTEST**(integer):chiave primaria, che indentifica univocamente i vari test **DESCRIZIONE**(string):Contiene la descrizione del test effettuato sulle release

Documento delle associazioni

Nome	Descrizione	Classi Coinvolte
PARTECIPA	Esprime la relazione tra gli utenti e progetto	**Utente[1..*] ruolo partecipa:** uno o più utenti possono partecipare a vari progetti **Progetto [1..*] ruolo partecipa:** ad un progetto possono partecipare uno o più utenti
Contiene	Esprime la relazione tra progetto e package	**Progetto[1..1] ruolo contiene:** un progetto può contenere uno specifico package **Package[1..*] ruolo contiene:** un progetto può contenere uno o più package
SottoPackage	Esprime la relazione tra package e se stesso	**Package[0..*] ruolo sottopackage:** un package può contenere al suo interno un altro package
Possiede	Esprime la relazione tra package e classi	**Package[1..1] ruolo possiede:** indica le classi contenute **Classi[1..*] ruolo èpossiede:** indica in quale package è contenuta la classe
Accede	Esprime la relazione tra utente e classi	**Utente[1..*] ruolo accede:** indica quali utenti possono accedere a determinate classi **Classi[1..*] ruolo accede:** indica le classi accessibili dagli utenti
Possiede	Esprime la relazione tra Progetto e Release	**Progetto[1..1]ruolo possiede:** indica una precisa release a che progetto si riferisce **Release[1..*] ruolo possiede:** indica le release dei progetti
Testano	Esprime la relazione tra Release e TestRelease	**Release[1..*]ruolo testano:** indica la release coinvolta nel test **TestRelease[1..*]ruolo testa:** indica il test usato per testare la release
Avvia	Esprime la relazione tra Utente e TestRelease	**Utente[1..*] ruolo avvia:** indica quale utente ha avviato un test **TestRelease[1..*] ruolo avvia:** indica i test avviati

Progettazione logica

In questo capitolo si tratterà la traduzione dello schema concettuale in uno schema logico. Negli schemi relazionali che seguiranno le chiavi primarie sono indicate con una singola sottolineatura mentre le chiavi estere con una doppia sottolineatura

Traduzione in schema relazionali

Schema logico

Schema	**Chiavi esterne**
UTENTE(idutente,Nome,Cognome)	Nessuna
PARTECIPA(idutente,nomeprogetto)	Idutente ->Utente.idutente Nomeprogetto- >Progetto.nomeprogetto
Progetto(nomeprogetto,pathname,dataapertura, datachiusura)	nessuna
Package(idpackage,nomepackage,pathname,nomeprogetto)	Nomeprogetto - >Progetto.nomeprogetto
Sottopackage(idpackage,sottopackage)	Idpackage->Package.idpackage Sottopackage- >Package.idpackage
Classi(idclassi,pathname,nomeclasse,metodo,idpackage)	Idpackage->Package.idpackage
Sottometodo(idclasse,metodo)	Idclasse-> Classi.idpackage
Accede(idutente,idclasse,dataapertura,datachiusura)	Idutente->Utente.idutente Idclasse->Classi.idClasse
Release(idRelease,versione,status,nomeprogetto)	Nomeprogetto- >Progetto.nomeprogetto
Testano(idtest,idrelease)	Idtest -> TestRelease.idtest Idrelease -> Release.idrelease
TestRelease(idtest,risultato,datatest,descrizione)	Nessuna
Avvia(idutente,idtest)	Idutente->utente.idutente Idtest->TestRelease.idtest

Progettazione Fisica

La base di dati verrà implementata nel sistema per la gestione di dati Oracle xe 11g

a. Definizione delle tabelle

Seguono le definizioni delle tabelle estratte dallo script di creazione del database

Definizione della tabella UTENTE

```
--  Table UTENTE
------------------------------------------------------------

  CREATE TABLE   "UTENTE"
   (    "IDUTENTE" NUMBER NOT NULL PRIMARY KEY,
    "NOME" VARCHAR2(20 BYTE) NOT NULL,
    "COGNOME" VARCHAR2(40 BYTE) NOT NULL
   );
```

Definizione della tabella PARTECIPA

```
-- Table PARTECIPA
------------------------------------------------------

  CREATE TABLE   "PARTECIPA"
   (    "IDUTENTE" NUMBER NOT NULL,
    "NOMEPROGETTO" VARCHAR2(200 BYTE) NOT NULL,
    CONSTRAINT "IDUTENTEPARTECIPA_FK1" FOREIGN KEY ("IDUTENTE") REFERENCES   "UTENTE" ("IDUTENTE"),
    CONSTRAINT "NOMEPROGETTOPARTECIPA_FK2" FOREIGN KEY ("NOMEPROGETTO")  REFERENCES   "PROGETTO" ("NOMEPROGETTO")
   );
------------------------------------------------------
```

Definizione della tabella PROGETTO

```
--  Table PROGETTO
------------------------------------------------------------

  CREATE TABLE   "PROGETTO"
   (    "NOMEPROGETTO" VARCHAR2(200 BYTE) NOT NULL PRIMARY KEY,
    "PATHNAME" VARCHAR2(200 BYTE) NOT NULL,
    "DATAAPERTURA" DATE NOT NULL,
    "DATACHIUSURA" DATE
   );
```

Definizione della tabella PACKAGE

```
--   Table PACKAGE
-------------------------------------------------------

  CREATE TABLE   "PACKAGE"
   (    "NOMEPACKAGE" VARCHAR2(200 BYTE) NOT NULL,
    "PATHNAME" VARCHAR2(200 BYTE) NOT NULL,
    "NOMEPROGETTO" VARCHAR2(200 BYTE) NOT NULL,
    "IDPACKAGE" NUMBER NOT NULL PRIMARY KEY,
    CONSTRAINT "PACKAGE_FK1" FOREIGN KEY ("NOMEPROGETTO") REFERENCES   "PROGETTO" ("NOMEPROGETTO"),
    CONSTRAINT "NOMEPACKAGE_UN" UNIQUE("NOMEPACKAGE")
   );
-------------------------------------------------------
```

Definizione della tabella SOTTOPACKAGE

```
--------------------------------------------------------
--    Table SOTTOPACKAGE
--------------------------------------------------------

  CREATE TABLE   "SOTTOPACKAGE"
   (    "IDPACKAGE" NUMBER ,
    "SOTTOPACKAGE" NUMBER ,
    CONSTRAINT "IDPACKAGEPRINCIPALE_FK1" FOREIGN KEY ("IDPACKAGE")  REFERENCES   "PACKAGE" ("IDPACKAGE"),
    CONSTRAINT "SOTTOPACKAGE_FK2" FOREIGN KEY ("SOTTOPACKAGE")   REFERENCES   "PACKAGE" ("IDPACKAGE")
   );
```

Definizione della tabella CLASSI

```
--------------------------------------------------------
--      Table CLASSI
--------------------------------------------------------

  CREATE TABLE   "CLASSI"
   (    "PATHNAME" VARCHAR2(200 BYTE) NOT NULL,
    "NOMECLASSE" VARCHAR2(200 BYTE) NOT NULL,
    "METODO" VARCHAR2(200 BYTE) NOT NULL,
    "IDPACKAGE" NUMBER NOT NULL,
    "IDCLASSE" NUMBER NOT NULL PRIMARY KEY,
    CONSTRAINT "NOMECLASSE_UNQ" UNIQUE("NOMECLASSE"),
    CONSTRAINT "CLASSI_FK1" FOREIGN KEY ("IDPACKAGE") REFERENCES   "PACKAGE" ("IDPACKAGE")
   );
```

Definizione della tabella SOTTOMETODO

```
--------------------------------------------------------
--      Table SOTTOMETODO
--------------------------------------------------------

  CREATE TABLE   "SOTTOMETODO"
   (    "IDCLASSE" NUMBER ,
    "SOTTOMETODO" VARCHAR2(200 BYTE),
    CONSTRAINT "SOTTOMETODO_FK1" FOREIGN KEY ("IDCLASSE") REFERENCES   "CLASSI" ("IDCLASSE")
   );
--------------------------------------------------------
```

Definizione della tabella ACCEDE

```
--------------------------------------------------------
--      Table ACCEDE
--------------------------------------------------------

  CREATE TABLE   "ACCEDE"
   (    "IDUTENTE" NUMBER,
    "DATAAPERTURA" DATE,
    "DATACHIUSURA" DATE,
    "IDCLASSE" NUMBER,
    CONSTRAINT "IDCLASSEACCEDE_FK" FOREIGN KEY ("IDCLASSE")  REFERENCES   "CLASSI" ("IDCLASSE"),
    CONSTRAINT "IDUTENTEACCEDE_FK" FOREIGN KEY ("IDUTENTE") REFERENCES   "UTENTE" ("IDUTENTE")
   );
```

Definizione della tabella RELEASE

```
--------------------------------------------------------
--     Table RELEASE
--------------------------------------------------------

  CREATE TABLE   "RELEASE"
   (    "STATUS" VARCHAR2(200 BYTE),
    "NOMEPROGETTO" VARCHAR2(200 BYTE) NOT NULL ,
    "IDRELEASE" NUMBER NOT NULL PRIMARY KEY,
    "VERSIONE" NUMBER NOT NULL,
    CONSTRAINT "STATUS_ENUM" CHECK (STATUS IN ('ELIMINATO','AGGIORNATO','IMMUTATO')),
    CONSTRAINT "NOMEPROGETTORELEASE_FK" FOREIGN KEY ("NOMEPROGETTO") REFERENCES   "PROGETTO" ("NOMEPROGETTO")

   );
--------------------------------------------------------
```

Definizione della tabella TESTANO

```
--------------------------------------------------------
--     Table TESTANO
--------------------------------------------------------

  CREATE TABLE   "TESTANO"
   (    "IDRELEASE" NUMBER NOT NULL,
    "IDTEST" NUMBER NOT NULL,
    CONSTRAINT "IDRELEASETESTANO_FK" FOREIGN KEY ("IDRELEASE") REFERENCES   "RELEASE" ("IDRELEASE"),
    CONSTRAINT "IDTESTTESTANO_FK" FOREIGN KEY ("IDTEST")  REFERENCES   "TESTRELEASE" ("IDTEST")
   );
--------------------------------------------------------
```

Definizione della tabella TESTRELEASE

```
--      Table TESTRELEASE
--------------------------------------------------------

  CREATE TABLE   "TESTRELEASE"
   (    "IDTEST" NUMBER NOT NULL PRIMARY KEY,
    "DATATEST" DATE NOT NULL,
    "DESCRIZIONE" VARCHAR2(2000 BYTE) NOT NULL,
    "RISULTATO" VARCHAR2(20 BYTE) NOT NULL,
    "PATHNAME" VARCHAR2(200 BYTE) NOT NULL
   );
--------------------------------------------------------
```

Definizione della tabella AVVIA

```
--------------------------------------------------------
--     Table AVVIA
--------------------------------------------------------

  CREATE TABLE   "AVVIA"
   (    "IDUTENTE" NUMBER NOT NULL,
    "IDTEST" NUMBER NOT NULL,
    CONSTRAINT "IDTESTAVVIA_FK" FOREIGN KEY ("IDTEST") REFERENCES   "TESTRELEASE" ("IDTEST"),
    CONSTRAINT "IDUTENTEAVVIA_FK" FOREIGN KEY ("IDUTENTE") REFERENCES   "UTENTE" ("IDUTENTE")
   );
--------------------------------------------------------
```

b. Definizione dei vincoli e/o dei trigger

INSERIMENTOINVIES

Quando viene inserita la data di chiusura di un progetto, verranno aggiornate tutte le date di chiusura di ogni classe

```
--------------------------------------------------------
--     Trigger INSERIMENTOINVIES
--------------------------------------------------------

  CREATE OR REPLACE TRIGGER   "INSERIMENTOINVIES"
INSTEAD of  INSERT OR UPDATE ON VIEWPROGETTO
FOR EACH ROW
DECLARE
NOME progetto.nomeprogetto%TYPE;
date1 progetto.datachiusura%TYPE;
BEGIN
IF(inserting) THEN
INSERT INTO progetto(nomeprogetto,pathname,dataapertura)
VALUES(:new.nomeprogetto,:new.pathname,:new.dataapertura);
INSERT INTO release(STATUS,NOMEPROGETTO,IDRELEASE,VERSIONE)
VALUES(NULL,:new.nomeprogetto,IDRELEASEINC.nextval,1);
END IF;
IF(UPDATING) THEN

update progetto
set progetto.datachiusura=:new.datachiusura
where progetto.nomeprogetto=:old.nomeprogetto;
SELECT nomeprogetto,datachiusura INTO nome,date1
FROM  VIEWPROGETTO
WHERE   :old.datachiusura is NUll and nomeprogetto=:old.nomeprogetto;
UPDATE ACCEDE
SET accede.datachiusura=date1
WHERE accede.datachiusura is null and idclasse in (Select cl.idclasse from partecipa join progetto p on partecipa.nomeprogetto=p.nomeprogetto
 join package pack on p.nomeprogetto=pack.nomeprogetto join classi cl on cl.idpackage=pack.idpackage
where partecipa.nomeprogetto=nome);



END IF;

END;
```

VIEWPROGETTO

Vista utilizzata stampa e inserimento tramite il trigger

```
--------------------------------------------------------
--     View VIEWPROGETTO
--------------------------------------------------------

  CREATE OR REPLACE  VIEW   "VIEWPROGETTO" ("NOMEPROGETTO", "PATHNAME", "DATAAPERTURA", "DATACHIUSURA") AS
  SELECT "NOMEPROGETTO","PATHNAME","DATAAPERTURA","DATACHIUSURA"
FROM PROGETTO  P;
```

Sequence

```
--------------------------------------------------------
--     Sequence IDCLASSEINC incremento idclasse
--------------------------------------------------------

   CREATE SEQUENCE    "IDCLASSEINC"  MINVALUE 1 MAXVALUE 999999 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE ;
--------------------------------------------------------
--     Sequence IDPACKAGEINC incremento idpackage
--------------------------------------------------------

   CREATE SEQUENCE    "IDPACKAGEINC"  MINVALUE 1 MAXVALUE 999999 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE ;
--------------------------------------------------------
--     Sequence IDRELEASEINC incremento idrelease
--------------------------------------------------------

   CREATE SEQUENCE    "IDRELEASEINC"  MINVALUE 1 MAXVALUE 99999 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE ;
--------------------------------------------------------
--     Sequence IDTESTINC incremento idtest
--------------------------------------------------------

   CREATE SEQUENCE    "IDTESTINC"  MINVALUE 1 MAXVALUE 99999 INCREMENT BY 1 START WITH 1 NOCACHE ORDER  NOCYCLE ;
--------------------------------------------------------
--     Sequence IDUTENTEINC incremento idutente
--------------------------------------------------------

   CREATE SEQUENCE    "IDUTENTEINC"  MINVALUE 1 MAXVALUE 999999 INCREMENT BY 1 START WITH 1 NOCACHE  ORDER  NOCYCLE ;
```

TRIGGER AGGIORNAMENTO INDICI

```
--------------------------------------------------------
--     Trigger IDRELEASE
--------------------------------------------------------
create or replace TRIGGER IDRELEASE
    BEFORE INSERT ON RELEASE
    for each row
    BEGIN
    :new.IDRELEASE:= NVL(:new.IDRELEASE,IDRELEASEINC.nextval);
    END;
```

```
--     Trigger IDCLASSE
--------------------------------------------------------

create or replace TRIGGER IDCLASSE
    BEFORE INSERT ON CLASSI
    for each row
    BEGIN
    :new.IDCLASSE:= NVL(:new.IDCLASSE,IDCLASSEINC.nextval);
    END;
```

```
--------------------------------------------------------
--     Trigger idPackage
--------------------------------------------------------
create or replace TRIGGER idPackage
    BEFORE INSERT ON PACKAGE
    for each row
    BEGIN
    :new.IDPACKAGE:= NVL(:new.IDPACKAGE,IDPACKAGEINC.nextval);
    END;
--------------------------------------------------------
```

```
--------------------------------------------------------
--     Trigger IDTEST
--------------------------------------------------------
create or replace TRIGGER IDTEST
    BEFORE INSERT ON TESTRELEASE
    for each row
    BEGIN
    :new.IDTEST:= NVL(:new.IDTEST,IDTESTINC.nextval);
    END;
```

```
--------------------------------------------------------
--     Trigger IDutente
--------------------------------------------------------
create or replace TRIGGER   inserimentoUtente
BEFORE INSERT ON utente
for each row
begin
:new.IDUTENTE:= NVL(:new.IDUTENTE,IDUTENTEINC.nextval);
END;
```