UNIVERSITÀ DEGLI STUDI DI NAPOLI
FEDERICO II

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

CORSO DI LAUREA IN INFORMATICA

Progettazione di una base di dati per la gestione di un repository di software e delle attività di testing ad esso correlate.

Giuliano Galloppi N86001508
Federico Maglione N86001405

Anno Accademico 2015/2016

Professori: Adriano Peron
Alessandro De Luca

Indice

1. Analisi del problema

1.1 Descrizione del problema:

La base di dati qui presentata permette la gestione di un repository di software e delle attività di testing ad esso correlate.

Un ***repository*** è un ambiente di un sistema informativo, in cui vengono gestiti i metadati, attraverso tabelle relazionali; l'insieme di tabelle, regole e motori di calcolo tramite cui si gestiscono i metadati prende il nome di metabase. È un ambiente che può essere implementato attraverso numerose piattaforme hardware e sistemi di gestione dei database.

Il database contiene i descrittori di progetti software, la loro strutturazione in moduli o packages. I sorgenti SW non sono contenuti in un database ma sono contenuti in un file system. Il sistema permette l'associazione tra i descrittori delle strutture dei vari livelli ed i file che li contengono depositati nel file system. Per ciascun progetto esistono diverse versioni del progetto (release). Per ogni progetto nel repository vengono ospitati anche i casi di test (da considerarsi a tutti gli effetti dei sorgenti di codice) usati per validare un progetto. Nel database viene tenuta traccia di tutte le esecuzioni dei casi di test e del loro esito. In particolare lo stesso caso di test può essere eseguito più volte sulla stessa release del progetto e su release diverse dello stesso progetto. Per ogni esecuzione di un caso di test viene tenuta traccia nel database di tutte le strutture che sono state interessate dall'esecuzione (il livello più basso da considerare è quello del metodo).

1.2 Descrizione della progettazione:

La progettazione della base di dati è suddivisa in due punti:

- La progettazione concettuale , dove vengono proposti i due tipi di class diagram, con le relative tabelle e associazioni che danno un'idea del funzionamento della costruzione del repository che è stato presentato e di come sarà presentata la base di dati su di esso. A seguire è dato un dizionario dei dati relativo alle tabelle del progetto, che descrivono le funzionalità dei vari attributi ed un documento dei vincoli dove sono descritte normative per la funzionalità del repository.

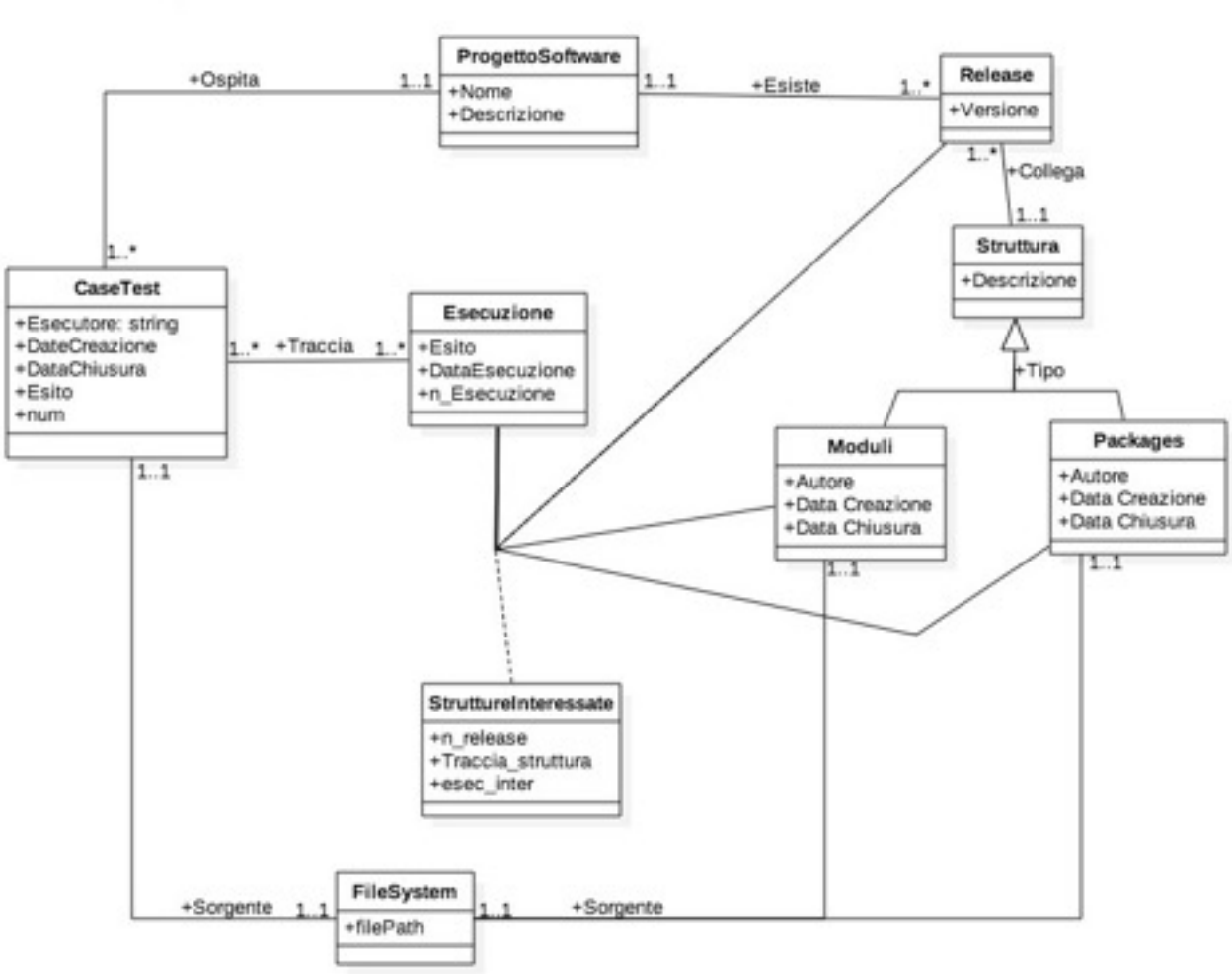
- La progettazione logica descrive lo schema relazionale nel quale sono introdotte le tabelle con i rispettivi attributi , chiavi primarie e vincoli di integrità referenziale (foreign key) .

Successivamente è presente un'implementazione logica in SQL dove le tabelle vengono implementate coi vari vincoli e trigger.

Infine vi è un Test Case che descrive un esempio della base di dati e di query esemplificative che permettono di reperire alcune informazioni dalla base di dati.

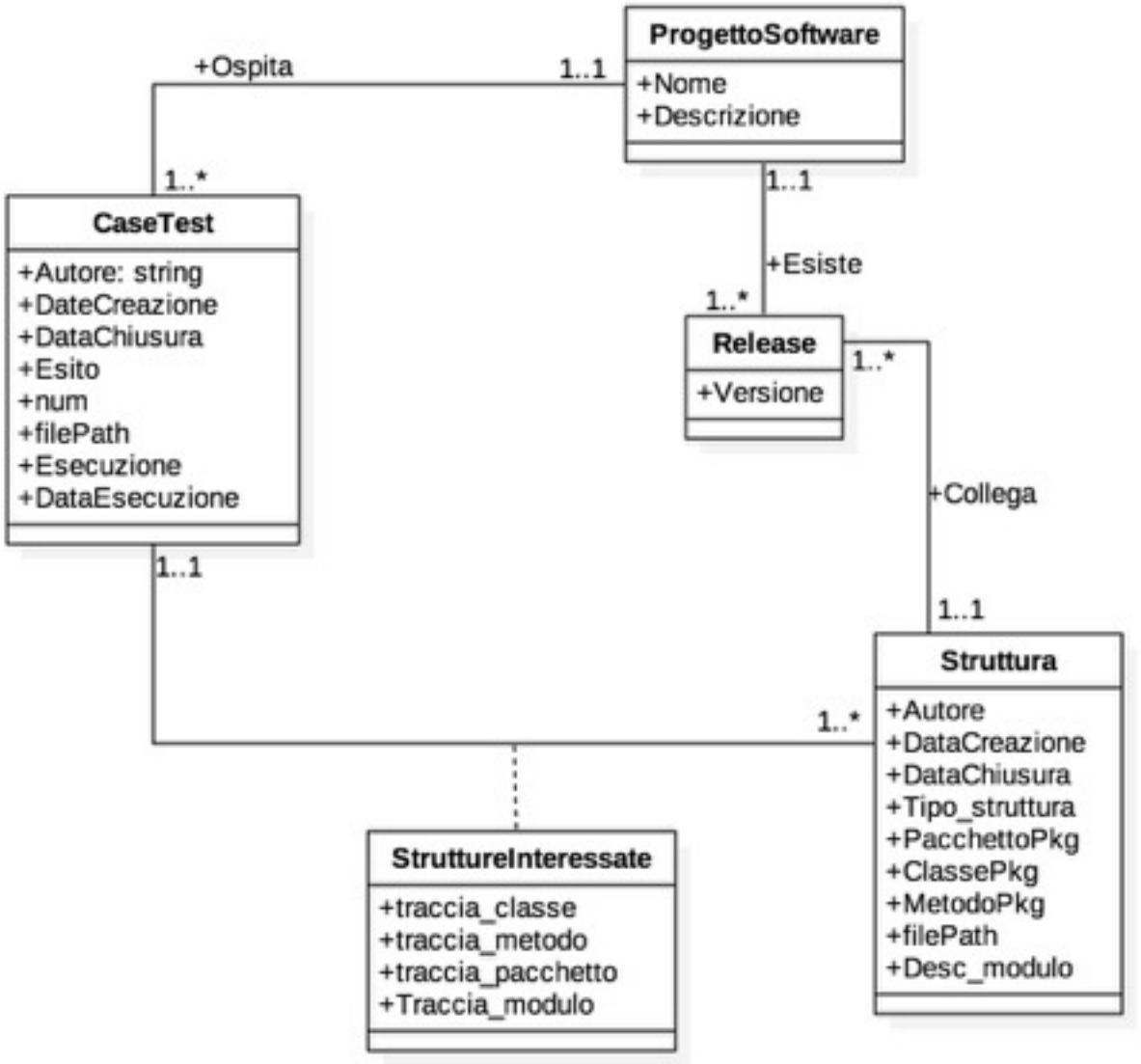
2. Progettazione Concettuale

2.1 Class Diagram



2.2 Class Diagram Ristrutturato

Su questa ristrutturazione del Class Diagram sono state sostituite le generalizzazioni raggruppandole in un unica classe come nei casi dell' entità “Moduli” e “Packages” , esse sono state unificate in un unica entità “Struttura” con l'aggiunta di un attributo “Tipo_struttura” che specifica se la struttura in questione è un modulo o un pacchetto, l'entità “filePath” è diventata attributo delle entità “CaseTest” e “Struttura” , invece l'entità “Esecuzione” è stata accorpata all'entità “CaseTest”.



2.3 Dizionario dei dati

Entità	***Descrizione***	***Attributi***
ProgettoSoftware	E' il repository dei progetti, contiene il nome e descrizione di ogni progetto.	**Nome**: *string* - Nome del progetto. **Descrizione:** *string* - Descrizione del progetto.
Release	Contiene le release di ogni progetto.	**Versione:** *real* - numero di versione del progetto.
CaseTest	Casi di test utilizzati per validare il progetto.	**Autore:** *string* - Persona che ha eseguito il test. **DataCreazione:** *Date* - Data di creazione del test. **DataChiusura:** *Date* - Data di chiusura del test. **Esito:** *char* - Esito del test, resituisce *'P'* se ha successo, *'N'* altrimenti. **num:** *int* - numero del caso di test. **filePath:** *string* - percorso dei sorgenti del caso di test. **Esecuzione:** *int* - numero di esecuzione del caso di test. **DataEsecuzione:** *Date* - data d'esecuzione del case test.

Struttura	Raccoglie la strutturazione di ogni release del progetto software, ogni release è strutturata in moduli o packages, se in packages è riportata la loro strutturazione in altri packages, classi o metodi. Essa riporta tutte le altre relative informazioni sulla struttura.	**Autore:** *string* - autore del pacchetto o modulo. **DataCreazione:** *Date* - Data di creazione della struttura. **DataChiusura:** *Date* - Data di chiusura della struttura. **Tipo_struttura:** *string* - Tipo della struttura , può assumere solo i valori “M” cioè “modulo” oppure “P” cioè “package”. **PacchettoPkg:** *string* - Nome del/dei pacchetto/i contenuto/i nella struttura “packages”. **ClassePkg:** *string* - Nome della/e classe/i contenuta/e nel pacchetto “PacchettoPkg” nella struttura “packages”. **MetodoPkg:** *string* - Nome del/dei metodo/i contenuto/i nel pacchetto “PacchettoPkg” , nella classe “ClassePkg” della struttura “packages”. **filePath:** *string* - percorso dei file sorgenti delle strutture. **Desc_modulo:** *string* - descrizione del modulo usato.

StruttureInteressate <associazione>	Rappresenta tutte le strutture che sono state interessate dall'esecuzione di un caso di test.	**traccia_classe:** *string* - traccia dell'eventuale classe interessa-*ta dall'esecuzione.* **traccia_metodo:** *string* - traccia dell'eventuale metodo interessato dall'esecuzione. **traccia_pacchetto:** *string* - traccia dell'eventuale pacchetto interessato dall'esecuzione. **Traccia_modulo:** *string* - traccia della descrizione dell'eventuale modulo interessato dall'esecuzione.

2.4 Documento dei vincoli

1. “Tipo_Struttura” dell’entità **Struttura** può assumere solo come valori “M” che sta per “modulo”o “P” che sta per “package” , tutti gli altri verranno rifiutati.

2. L’attributo “PacchettoPkg” assume valore NULL se “Tipo_Struttura” non ha valore “P”.

3. L’attributo “ClassePkg” assume valore NULL se “Tipo_Struttura” non ha valore “P”.

4. L’attributo “MetodoPkg” assume valore NULL se “Tipo_Struttura” non ha valore “P”.

5. L’attributo “Desc_modulo” assume valore NULL se “Tipo_Struttura” non ha valore “M”.

6. I metodi relativi ad una classe , per ogni classe , vanno inseriti tra parentesi quadre per essere distinti e nell’ordine delle relative classi di appartenenza.

7. L’attributo “Esito” dell’entità **CaseTest** può assumere solo come valori “P” che sta per “positivo” oppure “N” che sta per negativo, tutti gli altri verranno rifiutati.

8. Quando è inserito il pacchetto “PacchettoPkg” dell’entità “Struttura” si attiva un’eccezione che inserisce il valore nell’ entità “StruttureInteressate” nell ‘attributo “traccia_pacchetto” per renderlo reperibile dall’esecuzione associata ad esso.

9. Quando è inserita la classe “ClassePkg” dell’entità “Struttura” si attiva un’eccezione che inserisce il valore nell’ entità “StruttureInteressate” nell’attributo “traccia_classe” per renderlo reperibile dall’esecuzione associata ad esso.

10. Quando sono inseriti i metodi “MetodoPkg” dell’entità “Struttura” si attiva un’eccezione che inserisce il valore nell’ entità “StruttureInteressate” nell’attributo “traccia_metodo” per renderlo reperibile dall’esecuzione associata ad esso.

11. Quando è inserita la descrizione del modulo “Desc_modulo” dell’entità “Struttura” si attiva un’eccezione che inserisce il valore nell’ entità “StruttureInteressate” nell’attributo “Traccia_modulo” per renderlo reperibile dall’esecuzione associata ad esso.

3. Progettazione logica

3.1 Schema relazionale

ProgettoSoftware (IdProgetto,Nome, Descrizione).

Release(IdRelease , Versione , IdProgetto, EsternaStruttura).

Struttura (IdStruttura, Autore, DataCreazione, DataChiusura, Tipo_struttura, PacchettoPkg, ClassePkg, MetodoPkg, filePath, Desc_modulo, Estest).

CaseTest(Autore, DataCreazione, DataChiusura, Esito, num, filePath, Esecuzione, DataEsecuzione, IdProgetto)

StruttureInteressate (esec_inter, IdStruttura,traccia_classe, traccia_metodo, traccia_pacchetto, Traccia_modulo)

3.2 Implementazione logica in SQL

```
CREATE TABLE ProgettoSoftware
( IdProgetto        varchar(50)     NOT NULL,
Nome                char(25)        NOT NULL,
Descrizione         varchar(100),
PRIMARY KEY (IdProgetto) );
```

- **i vincoli 'rif_prog' e 'rif_release' garantiscono che quando un progetto è cancellato si cancellino i dati ad esso correlati.**

```
CREATE TABLE Release
( IdRelease         varchar(50)         NOT NULL,
Versione            float(25)           NOT NULL,
IdProgetto          varchar(50)         NOT NULL,
EsternaStruttura    varchar(50)         NOT NULL,
PRIMARY KEY(IdRelease),
CONSTRAINT rif_prog FOREIGN KEY(IdProgetto) REFERENCES ProgettoSoftware(IdProgetto)
                                    ON DELETE CASCADE    ON UPDATE CASCADE  );
CONSTRAINT rif_release FOREIGN KEY (EsternaStruttura) REFERENCES Struttura(IdStruttura)
                                    ON DELETE CASCADE    ON UPDATE CASCADE );
```

- **Il vincolo 'tipo' fa si che il valore dell'attributo 'Tipo_struttura' abbia solo valori 'M' cioè 'modulo' oppure 'P' cioè 'package'.**
- **I vincoli 'st_fk' e 'esec_fk' garantiscono che quando una struttura è cancellata si cancellino i dati ad essa correlati.**

```
CREATE TABLE Struttura
( IdStruttura       varchar(50)     NOT NULL,
Autore              char(25),
DataCreazione       DATE,
DataChiusura        DATE,
Tipo_struttura      char(1)         NOT NULL,
PacchettoPkg        varchar(100)    DEFAULT NULL,
ClassePkg           varchar(100)    DEFAULT NULL,
MetodoPkg           varchar(100)    DEFAULT NULL,
filePath            varchar(100)    UNIQUE          NOT NULL,
```

```
Desc_modulo          varchar(100)          DEFAULT NULL,
Estest               int                   NOT NULL,
PRIMARY KEY(IdStruttura),

CONSTRAINT st_fk FOREIGN KEY (IdRelease) REFERENCES Release (IdRelease)
                                   ON DELETE CASCADE      ON UPDATE CASCADE,
CONSTRAINT esec_fk FOREIGN KEY (Estest) REFERENCES CaseTest (Esecuzione)
                                   ON DELETE CASCADE      ON UPDATE CASCADE,
CONSTRAINT tipo CHECK (Tipo_struttura IN ('M' , 'P')) );
```

- **Il vincolo 'esi' fa si che l'attributo 'Esito' possa assumere solo valori come 'P' che sta per positivo oppure 'N' che sta per negativo.**
- **Il vincolo 'test_fk' garantisce che quando un case test è cancellato si cancellino i dati ad esso correlati.**

```
CREATE TABLE CaseTest (
Esecuzione           int                   NOT NULL,
Autore               char(25),
DataCreazione        DATE,
DataChiusura         DATE,
Esito                char(1)               NOT NULL,
num                  int                   NOT NULL,
filePath             varchar(100)          UNIQUE          NOT NULL,
DataEsecuzione       DATE,
IdProgetto           varchar(50)           NOT NULL,

PRIMARY KEY(Esecuzione),
CONSTRAINT test_fk FOREIGN KEY (IdProgetto) REFERENCES ProgettoSoftware (IdProgetto)
                                   ON DELETE CASCADE      ON UPDATE CASCADE ),
CONSTRAINT esi CHECK (Esito IN ('P' , 'N') );
```

- **I vincoli 'intere' e 'st_fk' garantiscono che quando una struttura interessata è cancellata si cancellino i dati ad essa correlati.**

```
CREATE TABLE StruttureInteressate (
esec_inter           int                   NOT NULL,
```

```
IdStruttura          varchar(50)        NOT NULL,
traccia_classe       varchar(100)       DEFAULT NULL,
traccia_metodo       varchar(100)       DEFAULT NULL,
traccia_pacchetto    varchar(100)       DEFAULT NULL,
Traccia_modulo       varchar(100)       DEFAULT NULL,
CONSTRAINT intere FOREIGN KEY (esec_inter) REFERENCES CaseTest (Esecuzione)
                                 ON DELETE CASCADE      ON UPDATE CASCADE,
CONSTRAINT st_fk FOREIGN KEY (IdStruttura) REFERENCES Struttura (IdStruttura)
                                 ON DELETE CASCADE      ON UPDATE CASCADE,
);
```

3.3 Implementazione dei vincoli

I vincoli dall' 1 al 7 sono implementati nelle tabelle.

Vincoli 8,9,10,11.

- **CREATE OR REPLACE TRIGGER** vincoli

```
AFTER INSERT ON STRUTTURA
FOR EACH ROW
BEGIN
    INSERT INTO STRUTTUREINTERESSATE(IDSTRUTTURA, TRACCIA_CLASSE,
TRACCIA_METODO, TRACCIA_PACCHETTO, TRACCIA_MODULO, ESEC_INTER)
VALUES(:NEW.IDSTRUTTURA, :NEW.CLASSEPKG,:NEW.METODOPKG, :NEW.PACCHET
TOPKG, :NEW.DESC_MODULO, :NEW.ESTEST);
 END;
```

- **CREATE OR REPLACE TRIGGER** VINCOLO2

```
AFTER UPDATE ON STRUTTURA
FOR EACH ROW
 BEGIN
    UPDATE STRUTTUREINTERESSATE SI
    SET ESEC_INTER = :NEW.ESTEST , TRACCIA_CLASSE = :NEW.CLASSEPKG ,
        TRACCIA_METODO = :NEW.METODOPKG ,TRACCIA_MODULO
= :NEW.DESC_MODULO,  TRACCIA_PACCHETTO = :NEW.PACCHETTOPKG
          WHERE :OLD.IDSTRUTTURA = SI.IDSTRUTTURA;
 END;
```

4 Test Case

4.1 Esempio di popolamento della Base di Dati

INSERT INTO ProgettoSoftware
VALUES (“0” , “Prog1”, “primo progetto”)

INSERT INTO Release
VALUES (“1” , 1.0 , “0”)

INSERT INTO Struttura
VALUES (“01” , “Mario Rossi” , 01/02/2015 , 02/02/2015 , “P” , “Primo” , “Numero, Casa , Automobile” , “[Ottieni_numero],[get_address, get_phone, get_info], [get_targa]” , “C:\Users \Desktop”, “1”, NULL)

INSERT INTO CaseTest
VALUES (1,”Luca Bianchi” , 03/04/2015 , 03/04/2015 , “N” , 1, “C:\Users\Desktop\Test” , 1, 03/04/2015 , “0”)

INSERT INTO CaseTest
VALUES (1,”Luca Bianchi” , 03/04/2015 , 03/04/2015 , “P” , 1, “C:\Users\Desktop\Test” , 2, 04/04/2015 , “0”)

INSERT INTO CaseTest
VALUES (2,”Luca Bianchi” , 03/04/2015 , 03/04/2015 , “P” , 2, “C:\Users\Desktop\Test” , 1, 04/04/2015 , “0”)

INSERT INTO StruttureInteressate (esec_inter, IdStruttura)
VALUES (2, “01”)

4.2 Esempio di Query Esemplificative

- ***Visualizzare descrizione e release di ogni progetto***

```
SELECT    P.Descrizione , R.Versione , P.IdProgetto
FROM      ProgettoSoftware AS P, Release AS R
WHERE     P.IdProgetto=R.IdProgetto
```

- ***Visualizzare strutture di un determinato autore e la release più recente per ogni tipo di struttura***

```
SELECT Tipo_struttura, PacchettoPkg , ClassePkg , MetodoPkg , Desc_modulo, MAX(R.Versione)
FROM    Struttura NATURAL JOIN Release AS R
WHERE  Autore='Gennaro Esposito'
GROUP BY Tipo_struttura
```

- ***Lista degli esiti dei CaseTest con le strutture interessate per l'esecuzione di esso in ordine della data di chiusura degli stessi***

```
SELECT Struttura.DataChiusura, Esito, Tipo_struttura,traccia_classe, traccia_metodo,
traccia_pacchetto, Traccia_modulo
FROM StruttureInteressate AS s NATURAL JOIN Struttura, CaseTest AS c
WHERE c.IdTest = s.esec_inter
ORDER BY Struttura.DataChiusura
```

- ***Visualizzare il numero totale di case test raggruppati per progetto***

```
SELECT Count(IdTest), IdProgetto
FROM ProgettoSoftware NATURAL JOIN ProgettoSoftware
GROUP BY IdProgetto
```