InformaticaUmanistica

Basi di Dati

SQL-92

Dettagli e Approfondimenti

UNIVERSITÀ DI PISA

Dettagli e Approfondimenti

- **DDL: Tabelle**
  - valori di default
  - vincoli di riferimento
  - modifiche allo schema
- **DDL: Viste**
  - definizione e uso
- **DDL: Indici**
- **DCL**
  - utenti
  - autorizzazioni
  - schemi esterni
- **DML: Aggiornamenti**
  - inserimenti
- **DML: Interrogazioni**
  - operatori insiemistici
  - SELECT: espressioni
  - FROM: join
  - WHERE: op. like

SQL

- **SQL-92 Intermediate**

- **DDL**
  - “Data Definition Language”
  - definizione degli oggetti dello schema
  - CREATE DATABASE
  - DROP DATABASE
  - CREATE TABLE
  - DROP TABLE

SQL

- **DCL**
  - “Data Control Language”
  - utenti e autorizzazioni
- **DML**
  - “Data Manipulation Language”
  - interrogazioni e aggiornamenti
  - INSERT, DELETE, UPDATE
  - SELECT

DDL: Tabelle

- **Creazione**
  - CREATE TABLE <nome> (<schema>);
- **<schema>**
  - una o più definizioni di attributo
  - zero o più definizioni di vincoli di tabella
- **Definizione di attributo**
  - <nomeattributo> <tipo> [<vincoli di colonna>]

Valori di Default

- **Nella CREATE TABLE e' possibile specificare valori di default per gli attributi**

```
CREATE TABLE Studenti (
     matr integer PRIMARY KEY,
     cognome varchar(20) NOT NULL,
     nome varchar(20) NOT NULL,
     ciclo char(20) DEFAULT 'laurea tr.',
     anno integer NOT NULL DEFAULT 1,
     relatore char(4) REFERENCES Professori(cod));
```

Vincoli di Riferimento

- **Vincoli di riferimento di colonna**
  - <attr> <tipo> REFERENCES <chiave est.>
  - es: **`docente char(4) REFERENCES Professori(cod)`**

- **Vincoli di riferimento di tabella**
  - FOREIGN KEY (<attributi>)
    REFERENCES <chiave est.>
  - es: se la chiave di Professori è nome, cogn. **`FOREIGN KEY (provincia, collegio)`**
    `REFERENCES Collegi(provincia, numero)`

Vincoli di Riferimento

- **Aggiornamenti in cascata**
  - ON {UPDATE | DELETE} <azione>

- **<azione>**
  - RESTRICT (è l'azione standard)
  - CASCADE
  - SET NULL
  - SET DEFAULT
  - es:

```
docente char(4) REFERENCES Professori(cod)
                ON UPDATE SET NULL;
```

Modifiche allo Schema

- **Modifiche ad una tabella già esistente**
  - ALTER TABLE
- **Tre funzioni**
  - ridenominazione della tabella
  - aggiunta, ridenominazione ed eliminazione di attributi
  - aggiunta ed eliminazione di vincoli

Modifiche allo Schema

- **Ridenominazione della tabella**
  - ALTER TABLE <nome> RENAME TO <nuovo nome>;
  - es: **`ALTER TABLE Professori RENAME TO Docenti;`**

- **Modifiche agli attributi**
  - ALTER TABLE <nome> RENAME COLUMN <vecchio attributo> TO <nuovo attributo>;
  - ALTER TABLE <nome> ADD COLUMN <nuovo attributo> <tipo>;
  - ALTER TABLE <nome> DROP COLUMN <nome attributo>;

Modifiche allo Schema

- **Esempio**
  - `ALTER TABLE Studenti ADD COLUMN dataNascita DATE;`
  - `ALTER TABLE Studenti ADD COLUMN luogoNascita VARCHAR(20);`
  - `ALTER TABLE Studenti ADD COLUMN reddito DECIMAL(8,2);`
  - `ALTER TABLE Studenti RENAME COLUMN dataNascita TO dataDiNascita;`
  - `ALTER TABLE Studenti DROP COLUMN dataDiNascita;`

Modifiche allo Schema

- **Semantica dell'aggiunta di colonne**
  - la nuova colonna viene aggiunta allo schema
  - a tutte le ennuple viene aggiunto NULL

- **Semantica dell'eliminazione di colonne**
  - la colonna viene eliminata dallo schema, assieme agli eventuali vincoli relativi
  - viene effettuata la proiezione delle ennuple

Modifiche allo Schema

- **Aggiunta di vincoli**
  - ALTER TABLE <nome> ADD CONSTRAINT <nome vincolo> <def. vincolo di tabella>;
  - l'istanza deve rispettare il vincolo

- **Esempio**
  -
```
ALTER TABLE Studenti ADD CONSTRAINT cf
  UNIQUE (nome, cognome,
          dataNascita, luogoNascita);
```

Modifiche allo Schema

- **Eliminazione di vincoli**
  - ALTER TABLE <nome> DROP CONSTRAINT <nome vincolo>;

- **Esempio**
  - **`ALTER TABLE Studenti DROP CONSTRAINT cf;`**

- **Attenzione:**
  - molti DBMS non supportano nè DROP COLUMN, né DROP CONSTRAIN

DDL: Viste

- **Viste**
  - tabelle “virtuali”
  - definite attraverso un’interrogazione
  - possono essere utilizzate come tabelle reali

- **Due funzioni fondamentali**
  - creazione degli schemi esterni (privatezza dei dati o ristrutturazioni)
  - semplificazione di interrogazioni ricorrenti

DDL: Viste

- **Creazione di viste**
  - CREATE VIEW <nome> AS <SELECT>;
- **Semantica**
  - la vista viene ricalcolata sulla base della sua definizione ogni volta che viene usata
- **Eliminazione di viste**
  - DROP VIEW <nome>;

DDL: Viste

- **Privatezza: esami senza voti**

```
CREATE VIEW EsamiSenzaVoti AS
     SELECT studente, corso
     FROM Esami;

SELECT * FROM EsamiSenzaVoti;

SELECT *
 FROM Studenti, EsamiSenzaVoti
 WHERE matr=studente;

DROP VIEW EsamiSenzaVoti;
```

DDL: Viste

- **Compiti ricorrenti: professori e numeri**

```
CREATE VIEW ProfessoriNumeri AS
    SELECT codice, nome, cognome, numero
    FROM Professori JOIN Numeri
         ON cod=professore;
```

```
SELECT *
 FROM ProfessoriNumeri
 ORDER BY cognome, nome;
```

DDL: Viste

- **Differenza tra tabelle e viste**
  - le tabelle sono materializzate nella base di dati, le viste no (sono derivate dalle tabelle)
  - schema di una vista = attributi e tipi della select
  - istanza di una vista = risultato della select
  - le tabelle sono aggiornabili, le viste no
  - le viste sono sempre aggiornate e consistenti
  - non hanno impatto sulle prestazioni

DDL: Indici

- **Aggiunta di indici**
  - CREATE [UNIQUE] INDEX <nome> ON <tabella>(<attributi>);
  - UNIQUE: impone un vincolo di chiave sugli attributi

- **Esempio:**
  - **`CREATE INDEX annociclo ON Studenti(anno, ciclo);`**

DDL: Indici

- **Annotazioni**
  - indici secondari
  - In alcuni sistemi (es: PgSQL) anche il tipo di indice (BTREE, HASH, ecc.)
  - attenzione all'impatto sulle prestazioni

- **Eliminazione di indici**
  - DROP INDEX <nome>;

- **Esempio**
  - **`DROP INDEX annociclo;`**

DCL “Data Control Language”

- **Due funzioni principali**
  - creazione ed eliminazione di utenti
  - concessione e revoca di autorizzazioni
  - sintassi non standard
- **Utenti**
  - CREATE USER, DROP USER
- **Autorizzazioni**
  - GRANT, REVOKE

Creazione ed Eliminazione di Utenti

- **Esempio: in PgSQL**
  - CREATE USER <nome>
    [WITH [PASSWORD '<password>']
    [CREATEDB] [CREATEUSER]];
  - es: **`CREATE USER pguser WITH PASSWORD'pguser' CREATEDB;`**
  - inserimenti nella tabella pg_shadow

- **Eliminazione di utenti**
  - DROP USER <nome>;
  - es; **`CREATE USER pguser;`**

Concessione e Revoca di privilegi

- **Autorizzazioni**
  - non tutti gli utenti sono autorizzati ad accedere alle basi di dati
  - tipicamente: l'utente che crea la base di dati e/o le tabelle (il proprietario) è autorizzato a fare tutto
  - gli altri utenti non possono fare niente
  - il proprietario può concedere autorizzazioni ad altri utenti

Concessione e Revoca di privilegi

- **Istruzione GRANT**
  - GRANT <privilegi> ON <risorsa>
    TO <utente> [WITH GRANT OPTION];
  - <privilegi>: SELECT, INSERT, DELETE, UPDATE, REFERENCES, ALL
  - <risorsa>: tabella o vista
  - <utente>: nome oppure PUBLIC
  - WITH GRANT OPTION: l'utente può trasmettere i privilegi con l'istruzione GRANT

Concessione e Revoca di Privilegi

- **Esempi**
  - **`GRANT ALL ON Studenti TO pguser;`**
  - **`GRANT SELECT ON Professori TO pguser WITH GRANT OPTION;`**
  - **`GRANT SELECT ON EsamiSenzaVoti TO PUBLIC;`**

- **Revoca di privilegi**
  - REVOKE <privilegio> ON <risorsa> FROM <utente>;
  - es: **`REVOKE DELETE ON Studenti FROM pguser;`**

Schemi Esterni

- **Costruzione dello schema logico**
  - l'utente o il DBA creano la base di dati
  - l'utente o il DBA creano lo schema
- **Costruzione degli schemi esterni**
  - l'utente o il DBA definiscono eventuali viste
  - l'utente o il DBA definiscono le autorizzazioni
- **Schema esterno di un utente**
  - insieme delle risorse visibili all'utente

DML: Aggiornamenti

- **Cancellazioni**
  - DELETE FROM <tabella> <WHERE>;
  - es:
```
DELETE FROM Professori WHERE
       qualifica='supplente';
```

- **Modifiche**
  - istruzione UPDATE <tabella> SET <attributo>=<espressione> <WHERE>;
  - es:
```
UPDATE Professori
   SET qualifica='ordinario'
   WHERE cod='VC';
```

Inserimenti

- **Istruzione INSERT**
  - due forme
- **Una ennupla alla volta**
  - vengono specificati i valori della ennupla
- **Più ennuple alla volta**
  - viene specificata una SELECT
  - tutto il risultato della SELECT viene inserito nella tabella

Inserimenti

- **Una ennupla alla volta**
  - INSERT INTO <tabella> VALUES (<valori>);
- **Non è necessario specificare tutti i valori**
  - lista di attributi assieme alla tabella
  - i valori corrispondono ordinatamente agli attr.
  - altri attributi: NULL oppure DEFAULT
- **Esempio**
  - **`INSERT INTO Corsi(cod, titolo)`**
    `VALUES (‘BD’, ‘Basi di Dati’);`

Inserimenti

- **Più ennuple alla volta (“copia” di ennuple)**
  - INSERT INTO <tabella> <SELECT>;
  - tutti o parte degli attributi (corrispondenza)

- **Esempio:**

```
CREATE TABLE CorsiTriennale (
   cod char(3) PRIMARY KEY,
   titolo varchar(20),
   docente char(4)
   REFERENCES Professori(cod)
);
```

```
INSERT INTO CorsiTriennale
SELECT cod, titolo, docente
FROM Corsi
WHERE ciclo=‘laurea tr.’;
```

differenza con le viste

DML: Interrogazioni

- **Interrogazioni, forma generale**
  - più blocchi SELECT correlati da operatori insiemistici, UNION, INTERSECT, EXCEPT

- **Blocco SELECT**
  - SELECT [DISTINCT] <proiezioni e ridenom.>
  - FROM <prodotti cartesiani e join>
  - [WHERE <selezioni>]
  - [ORDER BY <attributi>];

Operatori Insiemistici

- **Eliminano i duplicati**
- **E' possibile mantenere i duplicati**
  - UNION ALL, INTERSECT ALL, EXCEPT ALL
- **Esempio: professori e studenti**

```
SELECT cognome, nome
FROM Professori
UNION
SELECT cognome, nome
FROM Studenti;
```

```
SELECT cognome, nome
FROM Professori
UNION ALL
SELECT cognome, nome
FROM Studenti;
```

Operatori Insiemistici

- **Ordinamenti**
  - le SELECT coinvolte non possono contenere ORDER BY
  - un'unica ORDER BY finale per il risultato

```
SELECT cognome, nome
FROM Professori
ORDER BY cognome
UNION
SELECT cognome, nome
FROM Studenti;
```

```
SELECT cognome, nome
FROM Professori
UNION
SELECT cognome, nome
FROM Studenti
ORDER BY cognome;
```

Clausola SELECT

- **DISTINCT, proiezioni, ridenominazioni**
- **Eventuali funzioni aggregative**
  - COUNT, SUM, MAX, MIN, AVG
  - DISTINCT
- **Esempio: contare le facoltà dei professori**

```
SELECT COUNT(facolta)
FROM Professori
```

risultato scorretto, uguale al numero dei professori

```
SELECT COUNT(DISTINCT facolta))
FROM Professori
```

risultato corretto, uguale al numero di facoltà distinte

Espressioni

- **Possono comparire nella SELECT**
- **Operandi**
  - valori degli attributi
- **Operatori (non standard)**
  - operatori aritmetici +, -, *, % ed altri
  - funzioni matematiche log, exp, sin, …
  - funzioni su stringhe length, substring, …
  - funzioni su date e tempi

Espressioni

- **Esempio: reddito familiare in lire**

```
SELECT cognome, nome, reddito*1936.27
FROM Studenti;
```

- **Esempio: media degli esami dello studente Pasquale Bruno in 110mi**

```
SELECT AVG(voto)/30*110
FROM Studenti JOIN Esami ON cod=studente
WHERE cognome='Bruno' AND
      nome='Pasquale';
```

Clausola FROM

- **Tabelle, prodotti cartesiani, join e alias**
- **Join**
  - R [INNER] JOIN S ON A=B
- **Altre forme di join**
  - R NATURAL JOIN S
  - R LEFT [OUTER] JOIN S ON A=B
  - R RIGHT [OUTER] JOIN S ON A=B
  - R FULL [OUTER] JOIN S ON A=B

Join Esterni

- **Esempio: tutti i professori con i loro eventuali numeri telefonici**

```
SELECT Professori.*, numero
FROM Professori LEFT OUTER JOIN Numeri
     ON cod=professore;
```

Clausola WHERE

- **Condizioni di selezione**
  - condizioni sui valori degli attributi
  - operatori di confronto =, >, <, >=, <=, <>
  - espressioni con operatori e funzioni
  - connettivi booleani AND, OR, NOT

- **Operatori speciali**
  - IS NULL, IS NOT NULL
  - LIKE

Operatore LIKE

- **Operatore LIKE**
  - corrispondenza “parziale” tra valori testuali

- **“Pattern”**
  - sequenza di caratteri e simboli speciali
  - %: una sequenza di 0 o più caratteri
  - _ : un carattere qualsiasi
  - corrisponde ad un insieme di stringhe

Operatore LIKE

- **Esempi:**
  - ‘B%i’: {‘Bianchi’, ‘Belli’, ‘Brutti’, ‘Bi’, …}
  - ‘p_ _ _a’: {‘palla’, ‘pasta’, ‘pista’, …}
  - ‘A_t%’: {‘Antonio’, ‘Artrite’, …}

- **Condizioni**
  - <attributo di tipo testo> LIKE <pattern>
  - vera se il valore dell’attributo appartiene all’insieme di stringhe corrispondenti

Operatore LIKE

- **Esempi**

```
SELECT cognome, nome
FROM Studenti
WHERE cognome LIKE ‘A%’;
```

```
SELECT *
FROM Corsi
WHERE titolo LIKE ‘%programmazione%’;
```

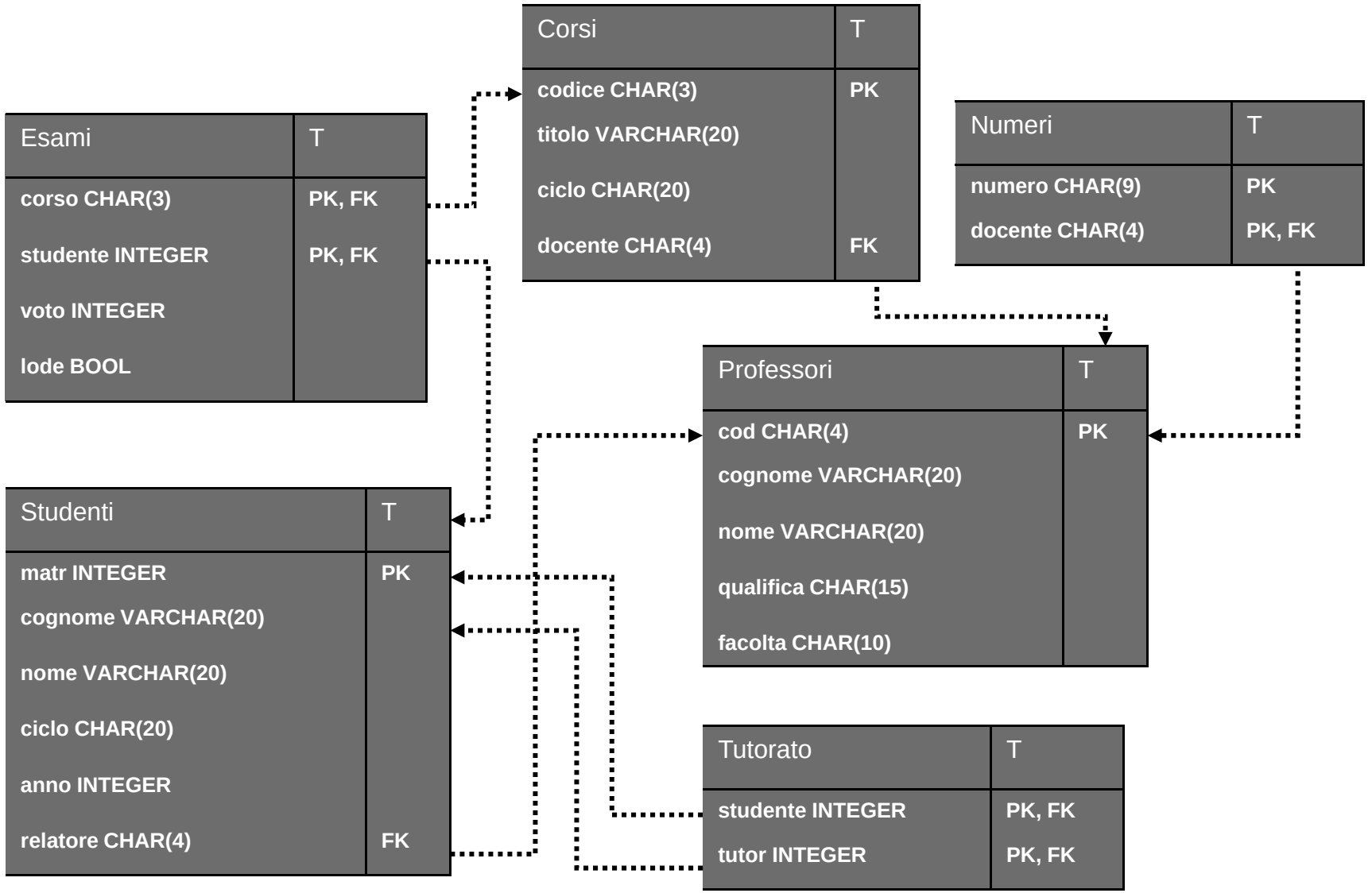
Dettagli e Approfondimenti

- **DDL: Tabelle**
  - valori di default
  - vincoli di riferimento
  - modifiche allo schema
- **DDL: Viste**
  - definizione e uso
- **DDL: Indici**
- **DCL**
  - utenti
  - autorizzazioni
  - schemi esterni
- **DML: Aggiornamenti**
  - inserimenti
- **DML: Interrogazioni**
  - operatori insiemistici
  - SELECT: espressioni
  - FROM: join
  - WHERE: op. like

```
CREATE TABLE Professori (
   cod char(4) PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   qualifica char(15),
   facolta char(10) );

CREATE TABLE Studenti (
   matr integer PRIMARY KEY,
   cognome varchar(20) NOT NULL,
   nome varchar(20) NOT NULL,
   ciclo char(20),
   anno integer,
   relatore char(4)
     REFERENCES Professori(cod)
);
CREATE TABLE Corsi (
   cod char(3) PRIMARY KEY,
   titolo varchar(20) NOT NULL,
   ciclo char(20),
   docente char(4)
     REFERENCES Professori(cod)
);
```

```
CREATE TABLE Tutorato (
   studente integer
     REFERENCES Studenti(matr),
   tutor integer
     REFERENCES Studenti(matr),
     PRIMARY KEY (studente,tutor));
CREATE TABLE Esami (
    studente integer
      REFERENCES Studenti(matr)
      ON DELETE cascade
      ON UPDATE cascade,
    corso char(3)
      REFERENCES Corsi(cod),
    voto integer,
    lode bool,
    CHECK (voto>=18 and voto<=30),
    CHECK (not lode or voto=30),
    PRIMARY KEY (studente, corso));
CREATE TABLE Numeri (
   professore char(4)
     REFERENCES Professori(cod),
   numero char(9),
  PRIMARY KEY (professore,numero));
```


Corsi
T
codice CHAR(3)
PK
titolo VARCHAR(20)
ciclo CHAR(20)
docente CHAR(4)
FK
Esami
T
corso CHAR(3)
PK, FK
studente INTEGER
PK, FK
voto INTEGER
lode BOOL
Numeri
T
numero CHAR(9)
PK
docente CHAR(4)
PK, FK
Professori
T
cod CHAR(4)
PK
cognome VARCHAR(20)
nome VARCHAR(20)
qualifica CHAR(15)
facolta CHAR(10)
Studenti
T
matr INTEGER
PK
cognome VARCHAR(20)
nome VARCHAR(20)
ciclo CHAR(20)
anno INTEGER
relatore CHAR(4)
FK
Tutorato
T
studente INTEGER
PK, FK
tutor INTEGER
PK, FK

Professori

cod	cognome	nome	qualifica	facolta
FT	**Totti**	**Francesco**	**ordinario**	**Ingegneria**
CV	**Vieri**	**Christian**	**associato**	**Scienze**
ADP	**Del Piero**	**Alessandro**	**supplente**	**null**

Studenti

matr	cognome	nome	ciclo	anno	relatore
111	**Rossi**	**Mario**	**laurea tr.**	**1**	**null**
222	**Neri**	**Paolo**	**laurea tr.**	**2**	**null**
333	**Rossi**	**Maria**	**laurea tr.**	**1**	**null**
444	**Pinco**	**Palla**	**laurea tr.**	**3**	**FT**
77777	**Bruno**	**Pasquale**	**laurea sp.**	**1**	**FT**
88888	**Pinco**	**Pietro**	**laurea sp.**	**1**	**CV**

Corsi

cod	titolo	ciclo	docente
PR1	**Programmazione I**	**laurea tr.**	**FT**
ASD	**Algoritmi e Str. Dati**	**laurea tr.**	**CV**
INFT	**Informatica Teorica**	**laurea sp.**	**ADP**

Tutorato

studente	tutor
111	77777
222	77777
333	88888
444	88888

Numeri

professore	numero
FT	0971205145
FT	347123456
VC	0971205227
ADP	0971205363
ADP	338123456

Esami

studente	corso	voto	lode
111	PR1	27	false
222	ASD	30	true
111	INFT	24	false
77777	PR1	21	false
77777	ASD	20	false
88888	ASD	28	false
88888	PR1	30	false
88888	INFT	30	true