Università degli Studi di Padova

Dipartimento di Matematica

Laurea Triennale in Informatica

Basi di Dati - Laboratorio 4

Progettazione e realizzazione di un Database

Massimiliano de Leoni	deleoni@math.unipd.it
Alessandro Padella	alessandro.padella@phd.unipd.it
Samuel Cognolato	samuel.cognolato@studenti.unipd.it

Indice

1 Realizzazione della base di dati

Per questo laboratorio utilizzeremo una base di dati che memorizza le informazioni relative ad un file system. In Figura 1 è rappresentato lo schema ER della base di dati che andremo a creare. Il primo passo per la realizzazione di una base di dati è la progettazione. In questa fase definiamo uno schema ER che illustra le informazioni contenute in ogni tabella e le relazioni che intercorrono tra le varie tabelle.

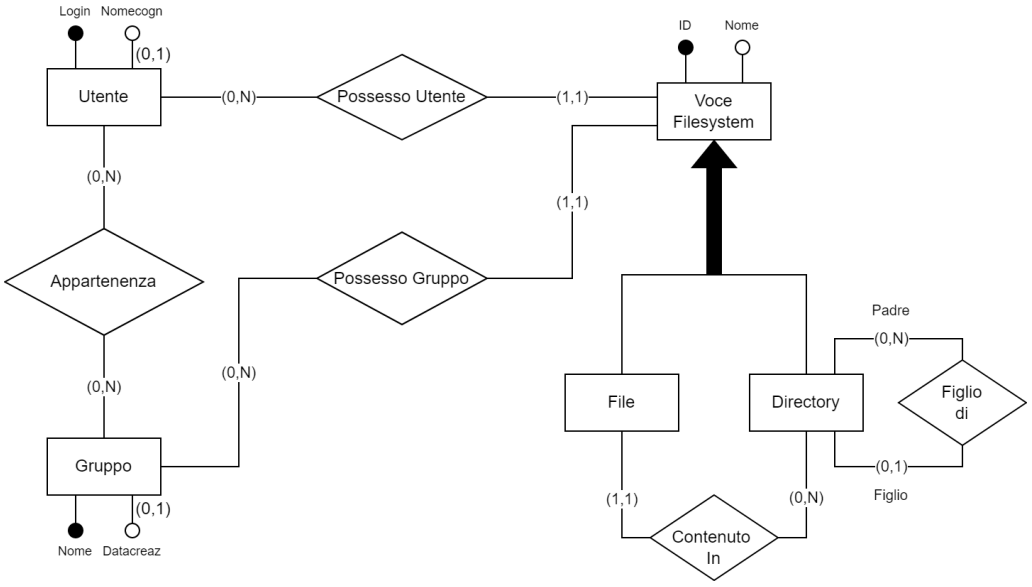

Figura 1: Schema ER

I file sono organizzati in una struttura ad albero, la radice è l'unico file a non avere padre. Per la gestione dei dati facciamo le seguenti assunzioni:

- La cancellazione di un utente determina la cancellazione di tutti i file dei quali è proprietario;
- I gruppi non vengono mai cancellati;
- La cancellazione di una directory comporta la cancellazione dei file in essa contenuti;
- Il `nome` dei file e il `nomecogn` (nome e cognome) degli utenti occupano al massimo 20 caratteri;
- Il `nome` dei gruppi e `login` degli utenti occupano al massimo 8 caratteri.

1.1 Realizzazione dello schema in SQL

Occorre realizzare lo schema in SQL tenendo conto, nella definizione delle chiavi esterne, delle assunzioni elencate precedentemente. Il primo passo per poter tradurre lo schema ER in un database relazionale è quello di eliminare la generalizzazione di `Voce Filesystem`. In Figura 2 è illustrato lo schema ER ristrutturato. Si è scelto di utilizzare 2 entità separate per rappresentare i file e le directory, poiché, in questo laboratorio, considereremo le directory come dei file speciali. Alternativamente sarebbe stato possibile raggruppare nell'entità padre entrambi i figli ed introdurre un nuovo campo dati binario che distinguesse i file ordinari da quelli che rappresentano una directory.

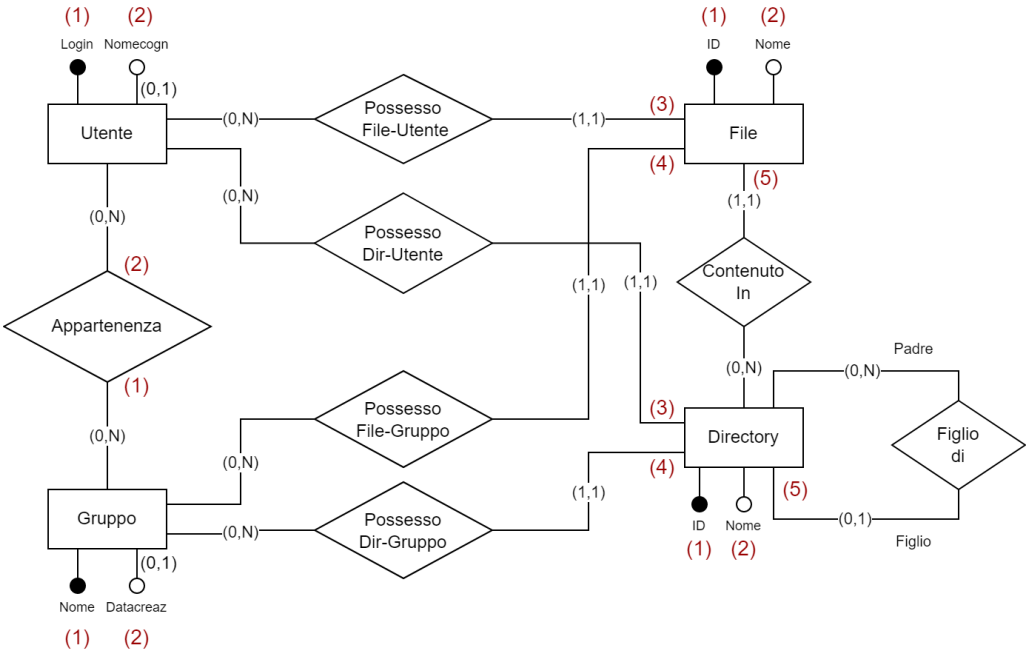

Figura 2: Schema ER ristrutturato

I numeri in rosso bordeaux rappresentano l'ordine in cui inserire i nomi dei campi per ogni entità/relazione nello schema logico, al fine di avere concordanza con le soluzioni e coerenza fra le esercitazioni.
A questo punto, è possibile creare lo schema logico, che è lasciato per esercizio, insieme alla creazione delle tabelle in PostgreSQL.

1.2 Popolamento del database

Una volta creato lo schema si popolino le tabelle utilizzando il codice `filesystem.sql`, la struttra creata per files e directory, sarà la seguente:

ID	Nome	Utente	Gruppo	Padre
1	Radice	root	admin	NULL
11	Var	root	admin	1
111	Mail	mail	mail	11
112	SubM	root	admin	11
12	tmp	root	admin	1
123	SubT	root	admin	12
13	home	root	admin	1
131	rossi	rossi	user	13
132	verdi	verdi	user	13

ID	Nome	Utente	Gruppo	Padre
1111	rossi.mbx	rossi	mail	111
1112	verdi.mbx	verdi	mail	111
121	tmp0.txt	rossi	user	12
122	tmp1.txt	verdi	user	12
1311	slide.txt	rossi	user	131
1312	progetto.pdf	rossi	user	131
1321	eserc1.sql	rossi	user	132

Si noti che `filesystem.sql` popola solamente le tabelle, ma non le definisce.

1.2.1 Osservazioni

- Il primo campo indica l'`ID` del file. Il popolamento della base di dati può risultare semplificato se si nota che l'`ID` della directory padre si può ottenere da quello del figlio togliendo l'ultima cifra (tranne che per la radice);
- Il secondo campo rappresenta il nome. La struttura rispecchia la struttura dell'albero delle directory. Ad esempio: la directory `Var` contiene la directory `Mail` che a sua volta contiene i file `rossi.mbx` e `verdi.mbx`;
- Il terzo campo è l'`Utente` che possiede il file;
- Il quarto campo è il `Gruppo` che possiede il file.

- Il quinto campo è l'ID del padre, NULL solo nel caso di root, ottenuto come descritto nel primo punto.

- I gruppi con le relative date di creazione sono:

Gruppo	**Data**
user	2007-01-02
mail	2006-01-01
admin	2006-02-04
sys	2006-12-25
none	2007-01-01

- Gli utenti presenti con i relativi gruppi di appartenenza sono:

Login	**Nome**	**Gruppi**
root	NULL	user, mail, admin, sys
verdi	Gino Verdi	user, mail
rossi	Anna Rossi	user, mail
mail	NULL	mail
nobody	NULL	

Per l'inserimento dei dati nelle tabelle, si può utilizzare lo script

2 Interrogazioni

Fornire le query SQL per rispondere alle seguenti domande. Ogni domanda è accompagnata dal risultato che si otterrebbe rispetto alla popolazione indicata precedentemente.

1. Indicare `nome` e `id` della directory radice.

`Nome`	`ID`
`Radice`	`1`

2. Creare una vista con il numero di file posseduti da ogni utente ed il suo `login`, e visualizzarne il contenuto.

Login	NumFile
mail	0
nobody	0
root	0
rossi	5
verdi	2

3. Creare una vista con il numero di file posseduti complessivamente dagli utenti per ogni gruppo ed il `nome` del gruppo, e visualizzarne il contenuto. Attenzione: non si chiede il numero di file posseduti direttamente dal gruppo. Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

Nome	NumFile
admin	0
mail	7
sys	0
user	7

4. Elencare i gruppi i cui utenti posseggono, complessivamente, il massimo numero di file. Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

Nome	Count
mail	7
user	7

5. Elencare gli utenti che non appartengono a nessun gruppo o a tutti i gruppi.

Login	NomeCognome
nobody	NULL

6. Aggiungere l'utente `root` al gruppo `none` e riprovare la query precedente.

Login	NomeCognome
nobody	NULL
root	NULL

7. Creare una vista con gli id delle directory vuote (che all'interno non hanno nè file nè altre directory). Indicare quindi `nome` e `id` delle cartelle vuote.

Nome	ID
SubM	112
SubT	123

8. Cancellare le directory vuote e mostrare le directory rimanenti (Ricaricare DB dopo questa query). Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

ID	Nome	Utente	Gruppo	Padre
1	Radice	root	admin	NULL
11	Var	root	admin	1
12	tmp	root	admin	1
13	home	root	admin	1
111	Mail	mail	mail	11
131	rossi	rossi	user	13
132	verdi	verdi	user	13

9. Cancellare l'utente `rossi` e verificare che siano stati cancellati i suoi file mostrando la tabella `File` (Ricaricare DB dopo questa query). Per quale motivo vengono cancellati i suoi file?

ID	Nome	Utente	Gruppo	Padre
1112	verdi.mbx	verdi	mail	111
122	tmp1.txt	verdi	user	12

10. 'E possibile cambiare il nome di login dell'utente `rossi` in `bianchi`? Se sì, fare l'update. Se no, cambiare adeguatamente la definizione delle tabelle dello schema. In entrambi i casi, visualizzare i dati delle cartelle.

ID	Nome	Utente	Gruppo	Padre
1112	verdi.mbx	verdi	mail	111
122	tmp1.txt	verdi	user	12
1111	rossi.mbx	bianchi	mail	111
121	tmp0.txt	bianchi	user	12
1311	slide.txt	bianchi	user	131
1312	progetto.pdf	bianchi	user	131
1321	eserc1.sql	bianchi	user	132

3 Soluzioni

3.1 Creazione tabelle

```
01 |  -- Creazione tabella Utente
02 |  CREATE TABLE Utente (
03 |      login VARCHAR(8) PRIMARY KEY,
04 |      nomecogn VARCHAR (20)
05 |  );
06 |
07 |  -- Creazione tabella Gruppo
08 |  CREATE TABLE Gruppo (
09 |      nome VARCHAR(8) PRIMARY KEY,
10 |      datacreaz DATE
11 |  );
12 |
13 |  -- Creazione tabella Appartenenza (fra gruppi ed utenti)
14 |  CREATE TABLE Appartenenza (
15 |      gruppo VARCHAR(8) REFERENCES Gruppo(nome),
16 |      utente VARCHAR(8) REFERENCES Utente(login)
17 |      ON DELETE CASCADE,
18 |      PRIMARY KEY (utente, gruppo)
19 |  );
20 |
21 |  -- Creazione tabella Directory
22 |  CREATE TABLE Directory (
23 |      id INT PRIMARY KEY,
24 |      nome VARCHAR(20),
25 |      utente VARCHAR(8) NOT NULL REFERENCES Utente(login)
26 |      ON DELETE CASCADE,
27 |      gruppo VARCHAR(8) NOT NULL REFERENCES Gruppo(nome),
28 |      padre INT REFERENCES Directory(id)
29 |      ON DELETE CASCADE
30 |  );
31 |
32 |  -- Creazione tabella File
33 |  CREATE TABLE File (
34 |      id INT PRIMARY KEY,
35 |      nome VARCHAR(20),
36 |      utente VARCHAR(8) NOT NULL REFERENCES Utente(login)
37 |      ON DELETE CASCADE,
38 |      gruppo VARCHAR(8) NOT NULL REFERENCES Gruppo(nome),
39 |      padre INT NOT NULL REFERENCES Directory(id)
40 |      ON DELETE CASCADE
41 |  );
```

3.2 Interrogazioni

1. Indicare `nome` e `id` della directory radice.

```
01 |  SELECT directory.nome, directory.id
02 |  FROM directory
03 |  WHERE directory.padre is NULL;
```

2. Creare una vista con il numero di file posseduti da ogni utente ed il suo `login`, e visualizzarne il contenuto.

```
01 |  CREATE VIEW numFileXUtente(login, conteggio) AS
02 |  SELECT login, COUNT(file.id)
03 |  FROM utente LEFT JOIN file ON (utente.login = file.utente)
04 |  GROUP BY login;
05 |
06 |  SELECT * FROM numFileXUtente;
```

3. Creare una vista con il numero di file posseduti complessivamente dagli utenti per ogni gruppo ed il `nome` del gruppo, e visualizzarne il contenuto. Attenzione: non si chiede il numero di file posseduti direttamente dal gruppo. Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

```
01 |  CREATE VIEW numFileXGruppo(gruppo, conteggio) AS
02 |  SELECT gruppo.nome, SUM(conteggio)
03 |  FROM numFileXUtente
04 |      JOIN appartenenza ON numFileXUtente.login = appartenenza.utente
05 |      JOIN gruppo ON appartenenza.gruppo = gruppo.nome
06 |  GROUP BY gruppo.nome;
07 |
08 |  SELECT * FROM numFileXGruppo;
```

4. Elencare i gruppi i cui utenti posseggono, complessivamente, il massimo numero di file. Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

```
01 |  CREATE VIEW numFileXGruppo(gruppo, conteggio) AS
02 |  SELECT gruppo.nome, SUM(conteggio)
03 |  FROM numFileXUtente
04 |      JOIN appartenenza ON numFileXUtente.login = appartenenza.utente
05 |      JOIN gruppo ON appartenenza.gruppo = gruppo.nome
06 |  GROUP BY gruppo.nome;
07 |
08 |  SELECT gruppo, conteggio
09 |  FROM numFileXGruppo
10 |  WHERE conteggio = (SELECT MAX(conteggio) FROM numFileXGruppo);
```

5. Elencare gli utenti che non appartengono a nessun gruppo o a tutti i gruppi.

```
01 |  SELECT utente.login, utente.nomecogn
02 |  FROM utente
03 |      LEFT JOIN appartenenza ON utente.login = appartenenza.utente
04 |  GROUP BY utente.login
05 |  HAVING COUNT(appartenenza.gruppo) = 0
06 |      OR COUNT(appartenenza.gruppo) = (SELECT COUNT (*) FROM gruppo);
```

In alternativa

```
01 |  SELECT utente.login, utente.nomecogn
02 |  FROM utente JOIN appartenenza ON login = utente
03 |  GROUP BY login
04 |  HAVING COUNT(*) = (SELECT COUNT(*) FROM gruppo)
05 |  UNION
06 |  SELECT utente.login, utente.nomecogn
07 |  FROM utente
08 |  WHERE login NOT IN (SELECT utente FROM appartenenza)
```

6. Aggiungere l'utente `root` al gruppo `none` e riprovare la query precedente.

```
INSERT INTO appartenenza(gruppo, utente) VALUES ('none', 'root');
```

   Ripetere una delle soluzioni del punto precedente.

7. Creare una vista con gli id delle directory vuote (che all'interno non hanno nè file nè altre directory). Indicare quindi `nome` e `id` delle cartelle vuote.

```
CREATE VIEW emptydirs(id) AS
SELECT directory.id
FROM directory
EXCEPT(
    SELECT directory.padre
    FROM directory
    UNION
    SELECT file.padre
    FROM file
    );
```

   In alternativa, facendo attenzione a rimuovere il valore `NULL`, si può utilizzare `NOT IN`:

```
CREATE VIEW emptydirs(id) AS
SELECT directory.id
FROM directory
WHERE directory.id NOT IN(
    SELECT directory.padre
    FROM directory
    UNION
    SELECT file.padre
    FROM file
    EXCEPT
    SELECT NULL
    );
```

   Ed infine si visualizza:

```
SELECT directory.nome, directory.id
FROM directory, emptydirs
WHERE directory.id = emptydirs.id;
```

8. Cancellare le directory vuote e mostrare le directory rimanenti (Ricaricare DB dopo questa query). Suggerimento: utilizzare la vista ottenuta nell'esercizio precedente.

```
DELETE FROM directory
WHERE directory.id IN (SELECT id FROM emptydirs);

SELECT directory.*
FROM directory;
```

9. Cancellare l'utente `rossi` e verificare che siano stati cancellati i suoi file mostrando la tabella `File` (Ricaricare DB dopo questa query). Per quale motivo vengono cancellati i suoi file?

```
01 | DELETE FROM utenti
02 | WHERE login = 'rossi';
03 |
04 | SELECT * FROM file;
```

10. 'E possibile cambiare il nome di login dell'utente `rossi` in `bianchi`? Se sì, fare l'update. Se no, cambiare adeguatamente la definizione delle tabelle dello schema. In entrambi i casi, visualizzare i dati delle cartelle.

    Non è possibile cambiare il nome degli utenti con la corrente definizione delle tabelle, in quanto non è definito un evento nel caso di update. Occorre quindi definire le tabelle aggiungendo la regola `ON UPDATE CASCADE` sui constraint di chiave esterna che si riferiscono ad `Utente(login)`. In alternativa è possibile rimuovere e riaggiungere i constraint di chiave esterna nel seguente modo:

```
01 | -- Rimozione constraint di FK
02 | ALTER TABLE appartenenza DROP CONSTRAINT appartenenza_utente_fkey;
03 | ALTER TABLE directory DROP CONSTRAINT directory_utente_fkey;
04 | ALTER TABLE file DROP CONSTRAINT file_utente_fkey;
05 |
06 | -- Riaggiunta constraint constraint di FK
07 | ALTER TABLE appartenenza ADD CONSTRAINT appartenenza_utente_fkey
08 |     FOREIGN KEY (utente) REFERENCES Utente(login)
09 |     ON DELETE CASCADE ON UPDATE CASCADE;
10 | ALTER TABLE directory ADD CONSTRAINT directory_utente_fkey
11 |     FOREIGN KEY (utente) REFERENCES Utente(login)
12 |     ON DELETE CASCADE ON UPDATE CASCADE;
13 | ALTER TABLE file ADD CONSTRAINT file_utente_fkey
14 |     FOREIGN KEY (utente) REFERENCES Utente(login)
15 |     ON DELETE CASCADE ON UPDATE CASCADE;
```