UNIVERSITÀ DEGLI STUDI DI UDINE
HIC SUNT FUTURA

DIPARTIMENTO DI SCIENZE MATEMATICHE, INFORMATICHE E FISICHE

Tesi di Laurea in
Informatica

Progettazione e implementazione di una base di dati relazionale per il percorso gravidanza e parto nell'Ospedale di Udine

Candidato
Andrea Salvador

Relatore
Prof. Angelo Montanari

Correlatori
Prof. Lorenza Driul
Dott. Andrea Brunello
Dott. Nicola Saccomanno

Anno accademico 2024/2025

Contatti dell'Istituto
Dipartimento di Scienze Matematiche, Informatiche e Fisiche
Università degli Studi di Udine
Via delle Scienze, 206
33100 Udine — Italia
+39 0432 558400
https://www.dmif.uniud.it/

Indice

1

Progettazione fisica

La fase di progettazione logica (Sezione **??**) produce uno schema logico, formalizzato nel modello relazionale, che descrive l'insieme di tabelle e di vincoli su di esse che contengono i dati di interesse. Questo schema è interrogabile, ovvero è possibile formulare delle interrogazioni (anche dette *query* o richieste) sulla base di dati; le interrogazioni si formulano attraverso linguaggi formali specifici per il modello relazionale come algebra relazionale e calcolo relazionale.

Il linguaggio SQL (*Structured Query Language*) è un linguaggio di interrogazione per basi di dati relazionali [1], affermatosi come standard *de facto* (grazie alle numerose implementazioni nei sistemi distribuiti commercialmente) e anche *de iure*[1]. SQL è composto di diverse parti: è possibile esprimere sia la definizione delle tabelle (DDL, *Data Definition Language*) sia le operazioni di inserimento, modifica, cancellazione e interrogazione (DML, *Data Manipulation Language*), insieme anche ai vincoli di integrità e ai privilegi degli utenti che possono accedere alla base di dati.

In questo capitolo costruiamo fisicamente la base di dati progettata nei capitoli precedenti attraverso diversi *script* in linguaggio SQL. Il DBMS che utilizziamo è PostgreSQL, uno tra i più diffusi DBMS relazionali.

1.1 Definizione dei domini

Alcune tabelle dello schema logico presentano attributi con vincoli sul dominio o con un insieme finito di valori possibili. In SQL è possibile definire domini, utilizzabili poi allo stesso modo di quelli predefiniti del linguaggio, che sintetizzano i vincoli da imporre ed evitano ripetizioni e quindi possibili inconsistenze. Elenchiamo di seguito i domini definiti per lo schema relazionale della nostra base di dati (Listato 1.1).

- Nella tabella `paziente` si utilizza come identificatore il codice fiscale italiano, ovvero una stringa alfanumerica di 16 caratteri.
- Nella tabella `gravidanza` l'attributo `pma_tipo` contiene l'informazione relativa al tipo di procreazione medicalmente assistita, che può avere valore `iui`, `fivet` o `icsi`.
- Nella tabella `visita` l'attributo `categoria_visita`, relativo al tipo di visita, può avere valore `primo_trimestre`, `secondo_trimestre`, `biometrica`, `altro_tipo`.

[1] Il primo standard ANSI e ISO per il linguaggio SQL è del 1986 (SQL-86) [1]. Successivamente sono stati pubblicati diversi aggiornamenti fino alla versione attuale, SQL:2023 (ISO/IEC 9075:2023).

```
-- Dominio codice_fiscale per persona
create domain codice_fiscale as char(16);
-- Dominio pma_tipo_enum per +gravidanza
create domain pma_tipo_enum as varchar
check (value in ('iui','fivet','icsi'));
-- Dominio categoria_visita_enum per visita
create domain categoria_visita_enum as varchar
check (value in ('primo_trimestre','secondo_trimestre','biometrica',
  'altro_tipo'));
-- Dominio stato_crescita_enum per visita
create domain stato_crescita enum as varchar
check (value in ('regolare','fgr','sga'));
```

Listato 1.1: Definizione dei domini.

```
create table paziente (
  cf codice_fiscale,
  primary key (cf),
  nome varchar not null,
  cognome varchar not null,
  data_nascita date not null
);
```

Listato 1.2: Definizione della tabella `paziente`.

1.2 Definizione delle tabelle

1.2.1 Area della gravidanza

La tabella `paziente` (Listato 1.2) utilizza il dominio `codice_fiscale` definito precedentemente.

Nella tabella `gravidanza` (Listato 1.3) è stato introdotto come chiave primaria un attributo `id`, da intendersi come numero progressivo; un'implementazione tipica e di "basso livello" degli identificatori di questo tipo prevede di dichiarare l'attributo come intero e chiave primaria, forzando l'utente a determinare il valore da assegnare a `id` ad ogni inserimento, solitamente calcolando il massimo dei valori `id` presenti nella tabella e aggiungendo 1. In PostgreSQL [2] è possibile ovviare a tale ostacolo definendo il campo `id` come `serial`: non è più necessario indicare né calcolare manualmente il valore `id` da assegnare perché viene determinato all'occorrenza, permettendo alle *query* di inserimento di esserne trasparenti. Nell'implementazione, `serial` è *zucchero sintattico* che si traduce nella definizione di una `sequence` di `integer` (a cui vengono dedicati 4 byte) ed esistono le varianti `smallserial` (2 byte, corrispondente a `smallint`) e `bigserial` (8 byte, corrispondente a `bigint`). Quando un valore dichiarato come `serial` viene usato come chiave esterna in altre tabelle deve essere trattato come il tipo `integer` corrispondente.

Nella definizione della tabella `gravidanza` (Listato 1.3) vengono formalizzati i vincoli di unicità da imporre su tuple di attributi, che corrispondono alle chiavi candidate dello schema concettuale, esposti nella Sezione **??**. Aggiungendo ulteriori vincoli a `gravidanza` (Listato 1.4) si possono soddisfare l'integrità referenziale verso la tabella `paziente` e il vincolo relativo agli attributi `pma_tipo` e `pma_ovodonazione`, riportati sempre nella Sezione **??**. Per il vincolo rimanente, relativo alle proprietà di crescenza dei valori di parità, si definisce un *trigger* apposito (Sezione **??**).

La tabella `malattia` (Listato 1.5) e la tabella `malattia_gravidanza` (Listato 1.6) seguono le defi-

```
create table gravidanza (
  id serial,
  primary key (id),
  paziente_cf codice_fiscale not null,
  data_primo_ingresso date not null,
  unique (paziente_cf, data_primo_ingresso),
  figli_nati_vivi integer not null,
  aborti_avuti integer not null,
  figli_nati_pretermine integer not null,
  figli_nati_a_termine integer not null,
  unique (paziente_cf, figli_nati_vivi, aborti_avuti, figli_nati_pretermine,
    figli_nati_a_termine),
  parita varchar not null,
  eta_concepimento integer,
  esito boolean,
  pma_tipo pma_tipo_enum,
  pma_ovodonazione boolean
  pregresso_gdm boolean not null,
  pregressa_pih boolean not null,
  pregressa_tireopatia boolean not null,
  pregressa_preeclampsia boolean not null,
  data_prevista_parto date,
  ultima_mestruazione_ecografica date,
  ultima_mestruazione_anamnestica date,
  annotazioni text
);
```

Listato 1.3: Definizione della tabella `gravidanza`.

```
-- Vincolo di chiave esterna
alter table gravidanza
add foreign key (paziente_cf)
references paziente
on update cascade on delete cascade;
-- Vincolo sugli attributi relativi alla PMA
alter table gravidanza
add constraint gravidanza_pma_vincolo
check ((pma_tipo is null and pma_ovodonazione is null)
  or (pma_tipo is not null and pma_ovodonazione is not null));
```

Listato 1.4: Definizione di vincoli aggiuntivi nella tabella `gravidanza`.

```
create table malattia (
  codice varchar,
  primary key (codice),
  nome varchar not null
);
```

Listato 1.5: Definizione della tabella `malattia`.

```
create table malattia_gravidanza (
  gravidanza_id integer,
  malattia_codice varchar,
  primary key (gravidanza_id, malattia_codice),
  terapia text
);
```

Listato 1.6: Definizione della tabella `malattia_gravidanza`.

nizioni riportate nello schema logico. Sulla seconda vengono imposti i vincoli di integrità referenziale (Listato 1.7)

1.2.2 Area delle visite

La tabella `visita` (Listato 1.8) utilizza i domini `categoria_visita_enum` e `stato_crescita_enum` definiti precedentemente; fatta eccezione per il vincolo di chiave esterna (Listato 1.9) i diversi vincoli imposti su questa tabella (Sezione **??**) vengono implementati attraverso *trigger* (**??**).

1.3 Definizione dei vincoli

1.4 Definizione dei trigger

```
alter table malattia_gravidanza
-- Vincolo di chiave esterna verso gravidanza
add foreign key (gravidanza_id)
references gravidanza.id
on update cascade on delete cascade,
-- Vincolo di chiave esterna verso malattia
add foreign key (malattia_codice)
references malattia.codice
on update cascade on delete cascade;
```

Listato 1.7: Definizione dei vincoli di chiave esterna nella tabella `malattia_gravidanza`.

```
create table visita (
  gravidanza_id integer,
  data date,
  primary key (gravidanza_id, data)
  epoca_gestazionale integer not null,
  eta integer not null,
  categoria_visita categoria_visita_enum not null,
  pressione_arteriosa_materna integer,
  anomalie_morfologiche_fetali text,
  prescrizione_asa boolean,
  decorso text,
  fuma boolean,
  premorfologica_indicata boolean,
  stato_crescita stato_crescita_enum,
  utpi float,
  altezza float,
  peso float,
  annotazioni text
);
```

Listato 1.8: Definizione della tabella `visita`.

```
alter table visita
add foreign key (gravidanza_id)
references gravidanza.id
on update cascade on delete cascade;
```

Listato 1.9: Definizione del vincolo di chiave esterna nella tabella `visita`.

2

Funzionalità della base di dati

2.1 Esempi di query

Bibliografia

[1] A. Silberschatz, H.F. Korth, e S. Sudarshan. *Database System Concepts*. McGraw-Hill, 2011.

[2] The PostgreSQL Global Development Group. *PostgreSQL 17.5 Documentation*, 2025.