Cofinanziato dall'Unione europea

REGIONE PIEMONTE

FinTech Software Developer- 2023/2025

Basi di dati SQL

Docente: Roi Davide Simone

Titolo argomento: Architettura Postgresql – Creazione di Tabelle e Viste

Roi Davide
Dispense

Intervento realizzato da
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

OGGETTI: TABELLA vs VISTA
Tabella: raccoglie/contiene i dati
Vista: visualizza i dati

Creazione di Tabelle

Roi Davide Dispense

```
-- Creare una tabella con il nome "nome_tabella"
CREATE TABLE nome_tabella (
 nome_colonna tipo_colonna [ clausola_default ] [ vincoli_di_colonna ] ,
 nome_colonna tipo_colonna [ clausola_default ] [ vincoli_di_colonna ] ,
 nome_colonna tipo_colonna [ clausola_default ] [ vincoli_di_colonna ] ,
Ecc..
[ , [ vincolo_di tabella] ... ]
);

[ clausola_default ] = DEFAULT { valore | NULL }
[ vincoli_di_colonna ] =
NOT NULL--indica che la colonna non puo' assumere il valore NULL.
PRIMARY KEY --indica che la colonna è la chiave primaria della tabella.
REFERENCES nome_tabella (nome_colonna) –indica che la colonna è chiave secondaria con la colonna nome_colonna
della tabella padre: nome_tabella
   [ ON DELETE { CASCADE | SET DEFAULT | SET NULL } ]
   [ ON UPDATE { CASCADE | SET DEFAULT | SET NULL } ]
[ TABLESPACE tablespace_name ]
```

Identifica in modo univoco, una o più colonne
NB: unique è come la PK ma evita casi con + info uguali

vincolo di colonna ("obblighi")
NULL
NOT NULL: comporta che sia presente un dato(non può essere nullo)

Tipi di Dati in Postgresql

Roi Davide Dispense

In PostgreSQL, ogni colonna di una tabella è associata ad un tipo di dato, che definisce il tipo di valore che può essere memorizzato in quella colonna.

Alcuni dei principali tipi di dati utilizzabili in PostgreSQL:

Intero: il tipo di dato "**integer**" consente di memorizzare numeri interi, senza parte decimale. In PostgreSQL, il tipo "bigint" consente di memorizzare numeri interi molto grandi. se 3.7 = 4

accettano | numeri | negativi

Serial: il tipo di dato "**serial**" consente di memorizzare numeri interi (è a tutti gli effetti un "**integer**" autoincrementale, gestito automaticamente dal DBMS di postgres con valore di DEFAULT e TRIGGER automatizzati). valore automatizzato, temporale..?

Decimale: il tipo di dato "**numeric**" consente di memorizzare numeri con una parte intera e una parte decimale di lunghezza variabile. In PostgreSQL, il tipo "**decimal**" è un sinonimo del tipo "numeric". se 3.7 resta e accetta interi,

Stringa: il tipo di dato "**varchar**" consente di memorizzare stringhe di lunghezza variabile, mentre il tipo "**char**" consente di memorizzare stringhe di lunghezza fissa. In PostgreSQL, il tipo "**text**" è un sinonimo del tipo "varchar" e consente di memorizzare stringhe di lunghezza variabile molto grandi.

Data e ora: il tipo di dato "**date**" consente di memorizzare date, mentre il tipo "**timestamp**" consente di memorizzare date e ore. fino al millisecondo Il tipo "**timestamptz**" consente di memorizzare date e ore con fuso orario. al giorno d'oggi i computer inseriscono molteplici dati in millisecondi, quindi non sono considerabili univoci

Booleano: il tipo di dato "**boolean**" consente di memorizzare valori booleani (vero o falso).

Array: il tipo di dato "**array**" consente di memorizzare un insieme di valori di uno stesso tipo in una sola colonna. più dati in fila ...

JSON: il tipo di dato "**json**" consente di memorizzare dati in formato JSON.

UUID: il tipo di dato "**uuid**" consente di memorizzare identificatori univoci universali.

Esempio di scrittura dei tipi di Dati

Roi Davide Dispense

```
CREATE TABLE nome_tabella (
    colonna1 SERIAL,
    colonna2 INTEGER ,
    colonna3 NUMERIC(10,2),
    colonna4 DECIMAL(10,2),
    colonna5 VARCHAR(100),
    colonna6 CHAR(16),
    colonna7 TEXT,
    colonna8 DATE,
    colonna9 TIMESTAMP,
    colonna10 TIMESTAMPTZ,
    colonna11 BOOLEAN,
    colonna12 TEXT[], --array di tipo testo es: ['Maglietta', 'Pantaloni', 'Scarpe']
    colonna13 JSON, --es: {"genere": "fantascienza", "anno": 2020, "descrizione": "Un romanzo distopico sulla lotta per la sopravvivenza nell'era digitale"}
    colonna14 UUID
);
```

Creazione di Tabelle

(esempio reale di tabella cliente)

Roi Davide Dispense

```
-- Creare una tabella: cliente
CREATE TABLE cliente (
    id SERIAL, -- definisce una colonna "id" con un valore univoco auto-incrementale per ogni riga
    nome VARCHAR(50) NOT NULL, -- definisce una colonna "nome" di tipo stringa non nullo
    cognome VARCHAR(50) NOT NULL, -- definisce una colonna "cognome" di tipo stringa non nullo
    data_di_nascita DATE, -- definisce una colonna "data_di_nascita" di tipo data
    codice_fiscale CHAR(16) UNIQUE, -- definisce una colonna «codice_fiscale" di tipo stringa «univoca»
    indirizzo VARCHAR(200), -- definisce una colonna "indirizzo" di tipo stringa
    telefono VARCHAR(20), -- definisce una colonna "telefono" di tipo stringa
    CONSTRAINT const_cliente_pk PRIMARY KEY (id) --definisce esplicitamente come primary key la colonna: id
);
```

intero gestito dal computer, auto increment = serial

Creazione di Tabelle

(dichiarazione implicita ed esplicita delle Primary Key e Foreign Key)

Roi Davide Dispense

```
--Dichiarazione constraint Primary Key implicita su una sola colonna
CREATE TABLE utente (
   codice_fiscale CHAR(16) PRIMARY KEY,
   email VARCHAR(50),
   nome VARCHAR(50),
   cognome VARCHAR(50)
);

--Dichiarazione constraint Primary Key implicita su più colonne
CREATE TABLE cliente (
   codice_fiscale CHAR(16),
   email VARCHAR(50),
   nome VARCHAR(50),
   cognome VARCHAR(50),
   PRIMARY KEY (codice_fiscale, email)
);

--Dichiarazione constraint Primary Key esplicita su più colonne
--(VERSIONE CONSIGLIATA IN QUANTO PIU' COMPLETA)
CREATE TABLE cliente (
   codice_fiscale CHAR(16),
   email VARCHAR(50),
   nome VARCHAR(50),
   cognome VARCHAR(50),
   CONSTRAINT const_cliente_pk PRIMARY KEY (codice_fiscale, email)
);
```

tutti gli oggetti messi sotto al nome, inventato, di costraint non vengono più influenzati da altre cose nel db

```
--Dichiarazione constraint Foreign key implicita su una sola colonna
CREATE TABLE ordine (
   id_ordine INTEGER CONSTRAINT ordine_pk PRIMARY KEY,
   data_ordine TIMESTAMP,
   importo NUMERIC(10,2),
   codice_fiscale_utente CHAR(16) REFERENCES utente(codice_fiscale)
);

--Dichiarazione constraint Foreign key implicita su più colonne
CREATE TABLE ordine (
   id_ordine INTEGER CONSTRAINT ordine_pk PRIMARY KEY,
   data_ordine TIMESTAMP,
   importo NUMERIC(10,2),
   codice_fiscale_cliente CHAR(16),
   email_cliente VARCHAR(50),
   FOREIGN KEY (codice_fiscale_cliente, email_cliente)
      REFERENCES cliente (codice_fiscale, email)
);

--Dichiarazione constraint Foreign key esplicita su più colonne
--(VERSIONE CONSIGLIATA IN QUANTO PIU' COMPLETA)
CREATE TABLE ordine (
   id_ordine INTEGER CONSTRAINT ordine_pk PRIMARY KEY,
   data_ordine TIMESTAMP,
   importo NUMERIC(10,2),
   codice_fiscale_cliente CHAR(16),
   email_cliente VARCHAR(50),
   CONSTRAINT const_ordine_fk FOREIGN KEY (codice_fiscale_cliente, email_cliente)
      REFERENCES cliente (codice_fiscale, email)
);
```

La Chiave esterna è la variabile di dato che mette in relazione tabelle tra loro permette di usare una variabile di una tab come riferimento per il lavoro in un'altra tab
NB: sempre meglio fare + tabelle piccole. occhio: + spazio occupa + costa.

Creazione di Tabelle

(esempio reale di tabelle: cliente e ordine)

```
-- Creare una tabella: cliente
CREATE TABLE cliente (
   id SERIAL, -- definisce una colonna "id" con un valore univoco auto-incrementale per ogni riga
   nome VARCHAR(50) NOT NULL, -- definisce una colonna "nome" di tipo stringa non nullo
   cognome VARCHAR(50) NOT NULL, -- definisce una colonna "cognome" di tipo stringa non nullo
   data_di_nascita DATE, -- definisce una colonna "data_di_nascita" di tipo data
   indirizzo VARCHAR(200), -- definisce una colonna "indirizzo" di tipo stringa
   telefono VARCHAR(20), -- definisce una colonna "telefono" di tipo stringa
   CONSTRAINT const_cliente_pk PRIMARY KEY (id) – definisce esplicitamente come PRIMARY KEY la colonna id
);
-- Creare un vincolo di unicità sulle colonne che identificano il cliente
ALTER TABLE cliente
ADD CONSTRAINT unique_cliente
UNIQUE (nome, cognome, data_di_nascita);

-- Creazione della tabella ordine con cancellazione in cascata su tabella relazionata cliente
--se cancello un record di cliente verranno cancellati automaticamente i record di ordine contenenti il cliente cancellato dalla tabella cliente.
CREATE TABLE ordine (
   id_ordine SERIAL,
   data_ordine TIMESTAMP,
   importo NUMERIC(10,2),
   id_cliente CHAR(16),
   CONSTRAINT const_ordine_pk PRIMARY KEY (id_ordine),
   CONSTRAINT const_ordine_fk FOREIGN KEY (id_cliente)
      REFERENCES cliente (id) ON DELETE CASCADE
);
```

Modifica e cancellazione di Tabelle


```
-- Aggiungi una colonna "stato" di tipo VARCHAR(20) alla tabella "ordine"
ALTER TABLE ordine
ADD COLUMN stato VARCHAR(20);

-- Rimuovi la colonna "importo" dalla tabella "ordine"
ALTER TABLE ordine
DROP COLUMN importo;

-- Modifica la primary key della tabella "ordine" per includere anche la colonna «data_ordine"
ALTER TABLE ordine
DROP CONSTRAINT const_ordine_pk, -- Rimuovi la primary key esistente
ADD CONSTRAINT const_ordine_pk PRIMARY KEY (id_ordine, data_ordine); -- Crea una nuova primary key con 2 colonne

-- Rimuovi la tabella ordine
DROP TABLE ordine;
```

Popolare e modificare record nelle Tabelle


```
-- Inserire i record in una tabella
INSERT INTO nome_tabella
[ ( colonna1, colonna2, ecc…) ]
VALUES
(valore1, valore2, ecc.. );

-- Modificare i record in una tabella
UPDATE nome_tabella
SET colonna1 = nuovo_valore1, ecc…
[ WHERE condizione ];

-- Cancellare i record in una tabella
DELETE FROM nome_tabella
[ WHERE condizione ];
```

Creare e cancellare le Viste

Roi Davide Dispense

```
-- Creare una vista
CREATE VIEW nome_vista AS
SELECT
    nomi_colonne
FROM
    tabelle
WHERE
    condizioni;

-- Modificare una vista
ALTER VIEW nome_vista AS
SELECT nomi_colonne FROM tabelle WHERE condizioni;

-- Cancellare una vista
DROP VIEW nome_vista;
```


Intervento realizzato da ITS Tecnologie dell'Informazione e della Comunicazione

Fonti:

- Documentazione ufficiale di Postgresql
  https://www.postgresql.org/docs/

- SQL online tutorial.org
  https://www.sqltutorial.org/

- SQL online documentazione w3schools
  https://www.w3schools.com/

- Autore: Serena Sensini (2021)
  Titolo: Dasi di Dati - Tecnologie, architetture e linguaggi per database
  Editore: Apogeo
  ISBN: 9788850335534

Fine della Presentazione

Intervento realizzato da
ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE