به نام خدا

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)

دستور کار آزمایشگاه پایگاه داده

جلسه دوم

کارباجداول و کلیدها و نوشتن پرس‌وجوهای ساده

استاد درس

دکتر شهریاری

تابستان ۹۹

مقدمه

در این بخش ابتدا به طور خلاصه درمورد pgsql توضیح داده می‌شود و سپس نگاهی به تفاوت‌های ساختاری دستورها در پرس‌وجوی SQL خواهیم داشت. به پرس‌وجوهای مقدماتی در محیط pgadmin مانند ایجاد و حذف پایگاه‌داده و تعریف کردن و استفاده از قیدو شرایط برای پرس‌وجوها پرداخته می‌شود و درنهایت به تعریف و استفاده از جدول مجازی یا view و مزایای Indexing می‌پردازیم.

PL/pgSQL

PL/pgSQL یک زبان رویه‌ایست که المان‌های زیادی مانند ساختارهای کنترلی، حلقه‌ها و محاسبات پیچیده را می‌تواند به SQL استاندارد اضافه کند و به شما اجازه می‌دهد توابع پیچیده را توسعه دهید و رویه‌ها را در SQL ذخیره کنید که این کار در SQL ساده امکانپذیر نیست.

دلایل دیگه‌ی استفاده از PL/pgSQL:

- برای یادگیری و استفاده بسیار آسان است.
- توابع تعریف شده‌ی کاربر و رویه‌های توسعه یافته را می‌توان مانند توابع از پیش تعریف شده در SQL استفاده کرد.
- توسط آن می‌توان توابع پیچیده را توسعه داد.
- می‌توان بجای ارسال کوئری‌های جدا، برای اجرای شی ذخیره شده در سرور، عبارت را اجرا کرد.

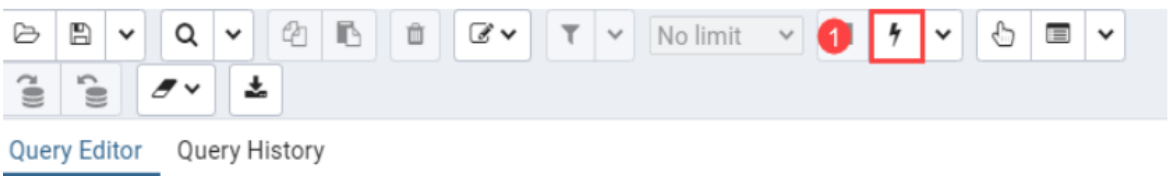
در این زبان که یک زبان بلاک-ساختاری است، توابع و رویه‌های ذخیره شده را به بلاک هایی تقسیم می‌کند.

هر بلاک دارای د. بخش است: اعلامیه و بدنه

وجود بخش اعلامیه برای بخش بدنه اختیاری است اما بخش بدنه الزامی است. هر بلاک با ; بعد از کلمه‌ی END پایان می‌یابد. و بخش بدنه همان بخشیست که کد می‌زنید. بخش بدنه نیز با ; تمام می‌شود.

```
1.  [ <<label>> ]
2. [ declare
3.     declarations ]
4. begin
5.     statements;
6.     ...
7. end [ label ];
```

برای اجرای این زبان در shell از دکمه اجرا استفاده می‌کنیم :



در این زبان ساختاری می‌توان چند زیربلاک را در یک بلاک خارجی جای داد و برای هر زیربلاک یک اسم می‌توان گذاشت.

برای تعریف متغیر از دستور زیر استفاده می‌شود

```
variable_name data_type [:= expression];
```

نکته: اگر مقدار پیشفرض برای متغیر نگذارید NULL به صورت پیشفرض انتخاب می‌شود.

نکته: با دستور pg_sleep() می‌توانید برای متغیر خود زمان شروع و خاتمه بگذارید.

انواع دستورات در SQL

Data Definition Language : DDL

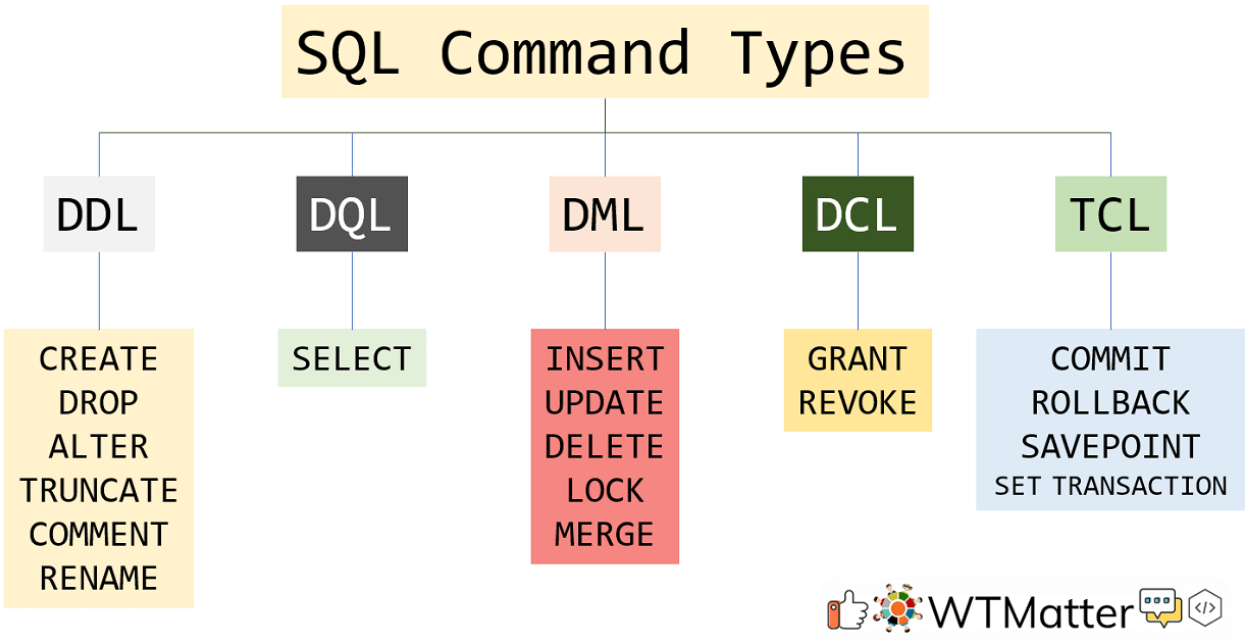
Data Manipulation Language : DML

Data Query Language : DQL

Data Control Language : DCL

Transaction Control Language : TCL

برای انتقال بهتر تفاوت تعاریف بالا به شکل زیر توجه کنید:



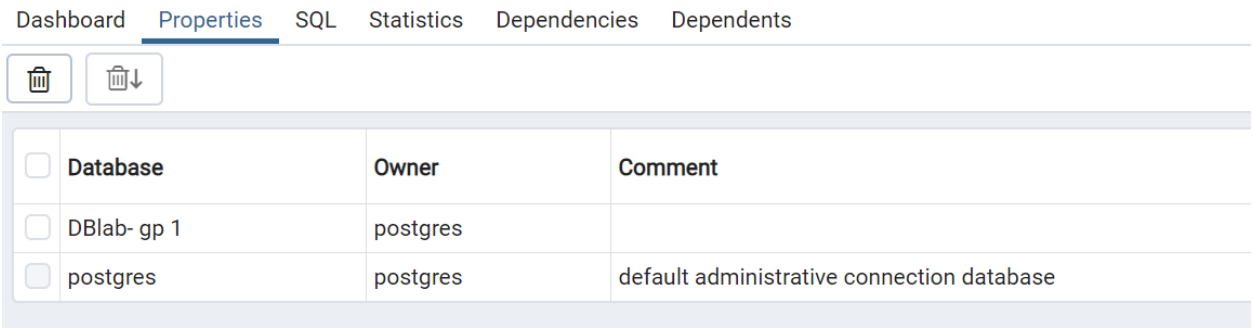
ایجاد و حذف پایگاه‌داده

ابتدا برای قابل ویرایش شدن بخش کد کوئری از نوار بالا گزینه Tools را انتخاب و Query Tool را بزنید.

برای ایجاد پایگاه‌داده از کد SQL زیر استفاده می‌کنیم:

```
1.  CREATE DATABASE DB_NAME
```

از تب Properties میتوانید پایگاه‌داده‌هایی که ایجاد کرده‌اید را ببینید



در ستون سمت چپ پنجره شاخه پایگاه‌داده‌ای که ایجاد کرده‌اید را انتخاب کنید و در سمت راست تب SQL را باز کنید و کد ایجاد پایگاه‌داده را میبینید.

درصورتی که میخواهید پایگاه‌داده جدید ایجاد کنید در همین قسمت اضافه می‌کنید.

برای حذف پایگاه‌داده مدنظر :

```
1.  DROP DATABASE database_name;
```

نکته: برای حذف پایگاه‌داده نباید در همان پایگاه‌داده باشید. میتوانید ابتدا یک پایگاه‌داده دیگر ایجاد کنید و از آنجا به حذف پایگاه‌داده قبلی بپردازید.

ممکنه پایگاه‌داده مدنظر شما وجودنداشته باشد یا قبلا پاک شده باشد دراین صورت :

```
1.  DROP DATABASE IF EXISTS some_database;
```

ایجاد و تغییر جداول پایگاه‌داده

در زیرشاخه Schemas از پایگاه‌داده انتخابی؛ وارد Public و سپس Tables بشوید. اگر شاخه جدول هایی که جلسه گذشته ساختید را انتخاب کنید و از سمت راست پنجره تب آخر که پایگاه‌داده باز شده فعلی است را بزنید و کد مد نظر را وارد کنید.

برای ایجاد جدول جدید:

```
CREATE [IF NOT EXISTS] TABLE table_name (
    column_name TYPE [column_constraint],
    [table_constraint,]
);
```

در بخش Column Type شرایط و ویژگی های ستون های مدنظر را وارد می‌کنید. ابتدا باید کلید اصلی جدول را مشخص کنید. کلید هر جدول باید یکتا باشد.

برای مثال برای ایجاد جدول معلمان با کلید اصلی id با نوع int و ستون های ویژگی First_name و last_name و Subject و Grade_level را انتخاب می‌کنیم.

```
CREATE TABLE teachers (
    id INT PRIMARY KEY,
    first_name VARCHAR,
    last_name VARCHAR,
    subject VARCHAR,
    grade_level int
);
```

برای حذف جدول :

```
DROP TABLE table_name;
```

اگر جدولی که می‌خواهید حذف کنید کلید خارجی داشته باشد؛ دستور زیر تمام جداول زیرمجموعه جدول فعلی را نیز حذف خواهد کرد.

```
DROP TABLE table_name CASCADE;
```

برای اضافه یا حذف کردن ستون و یا ایجاد تغییر در ستون و ساختار یک جدولی که قبلا ایجاد کردیم از ALTER TABLE استفاده می‌کنیم:

```
ALTER TABLE table_name action;
```

به جای table_name اسم جدول و به جای action عملی که میخواهیم اعمال شود را جایگزین می‌کنیم.

چند مثال ساده:

برای اضافه کردن ستون با ویژگی‌های مدنظر:

```
ALTER TABLE table_name
ADD COLUMN column_name datatype column_constraint;
```

برای حذف ستون از جدول موردنظر :

```
ALTER TABLE table_name
```

```
2.  DROP COLUMN column_name ;
```

برای تغییرنام ستونی که قبلا ایجاد شده :

```
1.  ALTER TABLE table_name
2.  RENAME COLUMN column_name
3.  TO new_column_name;
```

برای تغییر یک گزینه پیشفرض در ستون جدول و حذف قبلی:

```
1.  ALTER TABLE table_name
2.  ALTER COLUMN column_name
3.  [SET DEFAULT value | DROP DEFAULT];
```

برای اضافه کردن شرایط و ویژگی:

```
1.  ALTER TABLE table_name
2.  ADD CONSTRAINT constraint_name constraint_definition;
```

برای تغییر نام یک جدول به نام دیگر:

```
1.  ALTER TABLE table_name
2.  RENAME TO new_table_name;
```

برای تغییر نام ستون به یک نام دیگر:

```
1.  ALTER TABLE table_name
2.  RENAME COLUMN title TO table_name_title;
```

برای ساختن مقدار خالی برای پیشفرض برای ستون یک ستون

```
1.  ALTER TABLE tabel_name
2.  ALTER COLUMN new_tabel_name
3.  SET DEFAULT '_blank';
```

تعریف کلید اصلی و خارجی و استفاده از آن‌ها

کلید اصلی

کلید اصلی (PK) در هر جدول، یک ستون یا گروهی از ستون هاست که برای مشخص کردن یک سطر به صورت یکتا استفاده می‌شود.

[1] Primary key

شما یک PK با شرایط و ویژگی‌های مخصوص آن تعریف می‌کنید. در واقع یک PK ترکیبی از شرط NOT_NULL و UNIQUE بودن باید باشد. هر یک جدول تنها باید یک PK داشته باشد. به ازای هر PK در جدول PostgreSQL یک درخت بی روی آن ستون یا گروهی از ستون ها برای فهرست گذاری و تعریف PK می‌سازد.

نحوه تعریف PK در جدول را در صفحه بعد میبینیم:

```
1.  CREATE TABLE table (
2.      column_1 datatype PRIMARY KEY
3.      column_2 data_type
4.      ...
5.  );
```

اگر PK شامل بیشتر از یک ستون باشد:

```
1.  CREATE TABLE table (
2.      column_1 data_type ,
3.      column_2 data_type,
4.      ...
5.      PRIMARY KEY (column_1, column_2)
6.  );
```

نکته: اگر شما برای جدول خود PK تعریف نکنید خود Postgres یک ستون را به جای کلیداصلی جدول انتخاب می‌کند و آن ستون را table-name_pkey نام گذاری می‌کند.

اگر میخواهید در جدول های ساخته شده قبلی خود یک ستون دیگررا هم به PK اضافه کنید:

```
1.  ALTER TABLE table_name ADD PRIMARY KEY (column_1, column_2);
```

اگر قبلا برای جدولتان PK نذاشته‌اید با دستور زیر PK بگذارید:

```
1.  ALTER TABLE table_name
2.  ADD PRIMARY KEY (table_name_pk);
```

اگر میخواهید یک ستون شمارنده در جدولتان داشته باشید، فیلد AUTO_INCREMENT برای شما این کار را میکند.

نکته: Auto_increment به صورت پیشفرض از ۱ شروع می‌کند.

ممکن است شما بخواهید ستون PK شما این ویژگی را داشته باشد در این صورت:

```
1.  CREATE TABLE table_name(
2.      ID int NOT NULL AUTO_INCREMENT,
3.      ...
4.      PRIMARY KEY (ID)
5.      );
```

شما می‌توانید با دستور زیر عدد شروع شمارش را تغییر دهید:

```
1.  ALTER TABLE Persons AUTO_INCREMENT=100;
```

آنگاه دیگر نیازی نیست در زمان اضافه کردن یک رکورد جدید به جدول مقدار ID را تعریف کنید.

اگر میخواهید ویژگی Auto_increment را تغییر دهید از IDENTITY به صورت زیر استفاده کنید:

```
1.  ID int IDENTITY(1,1)  PRIMARY KEY;
```

عدد اول مقدار شروع شمارش و عدد دوم مقدار اضافه شدن به عدد قبلی به ازای اضافه شدن به هررکورد است.

کلید خارجی

کلید خارجی (FK) در هر جدول به یک یا چند ستون گفته می‌شود که به یک PK از یک جدول دیگر ارجاع می‌دهد.

جدول ای که شامل کلید خارجی است ، جدول ارجاع یا جدول فرزند نامیده می شود. و جداول ارجاع شده توسط کلید خارجی ، جدول ارجاع شده یا جدول والدین نامیده می شود. یک جدول باتوجه به روابط آن با جداول دیگر می تواند چندین کلید خارجی داشته باشد که با Constraints شرایط آن را مشخص می‌کنیم:

CONSTRAINT [fk_name]

FOREIGN KEY(fk_columns)

REFERENCES parent_table(parent_key_columns)

ON DELETE [delete_action]

ON UPDATE [update_action]

شرایطی که برای کلید خارجی می‌توان انجام داد:

- SET NULL
- SET DEFAULT
- RESTRICT
- NO ACTION
- CASCADE

SET NULL : در زمان پاک کردن به صورت آبشاری به صورت پیشفرض کلید خارجی ستون در سطری که ارجاع به فرزند در زمان ارجاع دادن به آن سطر توسط جدول والدین پاک شده راNULL می‌کند.

[۲] Foreign key

SET DEFAULT : مقدار را روی پیشفرض تنظیم کند.

RESTRICT و NO ACTION : عملیات حذف و یا بروزرسانی را رد میکند.

CASCADE: وقتی که می‌خواهیم تمام سطرهای جداول فرزند به تبع سطهای والدین به صورت آبشاری حذف یا بروزرسانی کنیم از این عبارت استفاده می‌کنیم.

پرس‌وجوهای ساده

برای تعریف قید و ویژگی از دستور زیر استفاده می‌کنیم

```
1.  CREATE TABLE table_name (
2.      column1 datatype constraint,
3.      column2 datatype constraint,
4.      column3 datatype constraint,
5.      ....
6.  );
```

برای قیود هم یکسری شرایط تعریف شده که به توضیح آن ها میپردازیم:

- NOT NULL :

اجازه نمی‌دهد که هیچ ستونی در آن جدول شامل مقدار خالی/پوچ باشد.

- UNIQUE:

اجازه نمی‌دهد که هیچ خانه‌ای در یک ستون مقدار تکراری داشته باشد.

- PRIMARY KEY:

این شرط یعنی باید هردو شرط قبلی یعنی NOTNULLو UNIQUE را داشته باشد تا به PK در جدول تبدیل شود.

- FOREIGN KEY:

باید در جدول ارجاع داده شده یک صفت یکتا در سطر/خانه باشد.

- CHECK:

در وارد کردن مقادیر ستون‌ها یک شرطی را چک می‌کند.

- DEFAULT:

برای زمانی که مقداری را بعنوان ورودی مقداردهی ستون نمی‌دهیم، یک مقدار پیشفرض برای تعریف می‌کند.

- INDEX:

برای ساختن و بازیابی داده از پایگاه‌داده با سرعت زیاد

دستورات پایه‌ای و تعریف View

SELECT:

این عبارت بیشترین استفاده و همزمان بیشترین پیچیدگی و تنوع را درمیان عبارات پرس‌وجو را دارد. به همین دلیل برای راحتی در یادگیری می‌تواند به پرس‌وجوهای کوچکتر بدل شود.

عباراتی که می‌توان در ساختار SELECT استفاده کرد:

DISTINCT- ORDER BY- WHERE- LIMIT or FETCH- GROUP BY- HAVING- JOIN- UNION- INTERSECT- EXCEPT-

ساده ترین فرم استفاده از SELECT، FROM است.وقتی می‌خواهیم پرس‌وجویی از یک جدول بخصوص را بازیابی کنیم:

```
1. SELECT  select_list  FROM   table_name
```

اگر می‌خواهید از لیستی از ستون ها لیست مدنظر را بازیابی کنید، کاما (،) بگذارید .

اگر می‌خواهید تمام داده از یک ستون را بازیابی کنید بجای نوشتن اسم تمامی ستون‌ها (*) بگذارید.

نکته: توجه کنید که زبان SQL، case-insensitive است.

INSERT:

دستور INSERT INTO برای اضافه کردن مقادیر در ستون‌های خاص از جداول است.

```
1. INSERT INTO table_name (column1, column2, column3,...)
2. VALUES (value1, value2, value3, ...);
```

اگر برای تمام ستون‌ها میخواهید مقادیر اضافه کنید:

```
1. INSERT INTO table_name
2. VALUES (value1, value2, value3, ...);
3.
```

UPDATE:

برای بروزرسانی ساختار در جدول یا ستون یا یک خانه‌ی خاص به‌کار میرود.

```
1.  UPDATE table_name
2. SET column1 = value1, column2 = value2, ...
3. WHERE condition
```

نکته: اگر در بخش WHERE شرط مورد نظر و در بخش SET ستون مورد نظر را ننویسید این دستور تمام رکوردهای آن جدول را بروزرسانی خواهد کرد.

:VIEW

VIEW چیزی بیش از یک عبارت SQL نیست که در یک پایگاه داده با یک اسم مرتبط ذخیره می شود. یک VIEW در واقع نمایش مجازی یک جدول از پیش تعریف شده در SQL است.یک VIEW می تواند شامل تمام ردیف های یک جدول باشد یا سطرها را از یک جدول انتخاب کنید. VIEW را می توان از یک یا بسیاری از جداول ایجاد کرد که بستگی به سؤال نوشتاری SQL برای ایجاد یک VIEW دارد.VIEW هارا می‌توانید ایجاد یا حذف یا شامل تغییر کنید. همچنین مانند SQL از JOINیا WHERE برای آن استفاده کنید.

VIEW ، که نوعی جداول مجازی است ، به کاربران امکان می دهد موارد زیر را انجام دهند :

- ساختار داده ها به روشی که کاربران یا طبقات کاربران به صورت طبیعی یا بصری بیابند.

نکته: یک View همیشه داده‌های به روز شده (up-to-date) را نشان می‌دهد! موتور یک پایگاه داده هر زمانی که کاربر در خواست view کند داده‌ها را بازسازی می‌نماید.

- دسترسی به داده ها را به گونه ای محدود کنید که کاربر بتواند دقیقاً مورد نیاز خود را ببیند و (گاهی) دقیقاً مورد نیاز خود را تغییر دهد.
- خلاصه داده ها از جداول مختلف که می تواند برای تولید گزارش استفاده شود.
- VIEW را اینگونه می‌سازیم:

```
1.  CREATE VIEW view_name AS
2. SELECT column1, column2.....
3. FROM table_name
4. WHERE [condition];
```

نکته: می‌توانید چندین جدول را در عبارت SELECT خود به همان روش استفاده کنید همانطور که از آنها در یک سؤال معمولی SQL SELECT استفاده می‌کنید.

انواع عملگرها در SQL

عملگرها در SQL در قسمت WHERE برای ایجاد شرط یا مقایسه و کاربرد عملگرهای دیگر استفاده می‌شود.

۱. عملگرهای حسابی(Arithmetic operators)
۲. عملگرهای مقایسه‌ای(Comparison operators)

۳. عملگرهای منطقی(Logical operators)

۴. عملگرهای نفی(used to negate conditions)

عملگرهای حسابی:

+(جمع)
-(تفریق)
***(ضرب)**
/(تقسیم)
٪(باقیمانده)

عملگرهای مقایسه‌ای:

=
!=
<>
>
<
>=
<=
!<
!>

عملگرهای منطقی:

ALL
AND
ANY
BETWEEN
EXISTS
IN
LIKE
NOT
OR
IS NULL
UNIQUE

تمرین

۱-جداول ساخته شده جلسه قبل را با دستور ALTER ویرایش کنید و یک دستور دلخواه اضافه کنید.

۲-یک مثال برای یک مجموعه و چگونگی تعریف کلید اصلی و کلید خارجی شرح دهید و بگویید چرا این ویژگی را برای آن مجموعه مناسب کلید اصلی می‌دانید.

۳-سه دستور ساده با استفاده از INSERT و SELECT و UPDATE بزنید و بفرستید.

۴-یک VIEW از چند ستون از جدول ساخته شده‌ی خود بسازید و آن را نمایش دهید.

۵-به انتخاب خود از هرکدام از انواع عملگرها(حسابی،منطقی،مقایسه‌ای) ۲ عملگر را انتخاب و کاربرد آن را به صورت خلاصه توضیح دهید.

نکات مهم هنگام تمرین:

- کلیه مستند تکلیف‌ها (کد و گزارش و خروجی ها) باید در قالب یک فایل فشرده با نام **DBlab_#studentnumber_#assignmentnumber** (عدد آخر شماره سری تمرین است) در سامانه **courses** اپلود شوند.
- درصورت دستی نوشتن تکالیف تشریحی، ناخوانا بودن آن‌ها باعث کسر نمره می‌شود.
- درصورت بروز اشکال درمورد هر تمرین تا ۲۴ ساعت قبل از اتمام زمان ارسال، به آدرس **databaselab99.aut@gamil.com** ایمیل بزنید**.**
- تاخیر در ارسال تکالیف به ازای هرروز مشمول ٪۱۰کسر نمره خواهد شد.
- درصورت کشف تقلب براساس مقررات آموزشی برخورد خواهد شد.
- حداکثر تعداد صفحات باید ۱۰ صفحه باشد.

موفق باشید