Day 3 Contents

- Schema
- Transaction
- Users And Privileges
- Views
- Functions

Schema

- A schema is a named collection of tables. A schema can also contain views, indexes, sequences, data types, and functions.

```
CREATE SCHEMA name;
```

Schema

- Example:

```
create schema myschema;
create table myschema.company(
   ID         INT      ,
   NAME       VARCHAR (20),
   AGE        INT          ,
   ADDRESS    CHAR (25) ,
   SALARY     INT
   );
```

Drop Schema

```
//tables should be dropped firstly

DROP SCHEMA myschema;

//drop schema with its tables

DROP SCHEMA myschema CASCADE;
```

Indexs

- Indexes are special lookup tables that the database search engine can use to speed up data retrieval

```
CREATE INDEX index_name ON table_name (cols);

CREATE INDEX MSISDN_index ON SERVICE_USERS (MSISDN);
```

Transactions

- The example of this involves a financial transfer where money from one account is placed into another account. Suppose that Moataz writes a check to ali for $100.00 and ali cashes the check. Moataz’s account should be decremented by $100.00 and ali account incremented by the same amount:

```
UPDATE account SET balance = balance - 100 WHERE name =
‘Moataz';
UPDATE account SET balance = balance + 100 WHERE name =
‘ali';
```

Transactions

- If a crash occurs between the two statements, the operation is incomplete. Depending on which statement executes first, Moataz is $100 short without Islam having been credited, or Islam is given $100 without Motaz having been debited.
- Another use for transactions is to make sure that the rows involved in an operation are not modified by other clients while you're working with them.

Transactions

- A transaction is a set of SQL statements that execute as a unit. Either all the statements execute successfully, or none of them have any effect.
- This is achieved through the use of commit and rollback capabilities. If all of the statements in the transaction succeed, you commit it to record their effects permanently in the database. If an error occurs during the transaction, you roll it back to cancel it. Any statements executed up to that point within the transaction are undone, leaving the database in the state it was in prior to the point at which the transaction began.

Transactions

One way to perform a transaction is to issue a **`BEGIN TRANSACTION`** (or **`BEGIN`**) statement, execute the statements that make up the transaction, and end the transaction with a **`COMMIT`** `or (`**`END TRANSACTION`**`)` statement to make the changes permanent.

If an error occurs during the transaction, cancel it by issuing a **`ROLLBACK`** statement instead to undo the changes.

Transactions

- The following example illustrates this approach. First, create a table to use

```
CREATE TABLE t (name CHAR(20), UNIQUE (name));
```

- The statement creates a table, Next, initiate a transaction with BEGIN TRANSACTION, add a couple of rows to the table, commit the transaction, and then see what the table looks like:

```
BEGIN TRANSACTION;
INSERT INTO t SET name = ‘Islam';
INSERT INTO t SET name = ‘Moataz';
COMMIT;
SELECT * FROM t;
```

Transactions

- if you issue any of the following statements while a transaction is in progress, the server wait until the transaction ended before executing the statement:

```
ALTER TABLE
CREATE INDEX
DROP DATABASE
DROP INDEX
DROP TABLE
RENAME TABLE
TRUNCATE TABLE
```

Transactions

- Postgres enables you to perform a partial rollback of a transaction. To do this, issue a SAVEPOINT statement within the transaction to set a marker. To roll back to just that point in the transaction later, use a ROLLBACK statement that names the savepoint.

```
CREATE TABLE t (name CHAR(20), UNIQUE (name));
CREATE TABLE t (i INT);
BEGIN TRANSACTION;
INSERT INTO t VALUES(1);
SAVEPOINT my_savepoint;
INSERT INTO t VALUES(2);
ROLLBACK TO SAVEPOINT my_savepoint;
INSERT INTO t VALUES(3);
COMMIT;
```

Users & Privileges

- (Required) Add a Linux user (As Database User Name)
- Login to Postgres as super user (postgres)
- Create user in database
- Maintain privileges

Users & Privileges

```
#useradd [options] username          [As Database UserName]

#passwd username

# su – postgres

$ psql

postgres=# CREATE USER username WITH PASSWORD 'password';
```

Privileges

Grant statement

```
GRANT privilege [,...] ON object [,...] TO { PUBLIC |
username }
```

- **privilege** values could be: SELECT, INSERT, UPDATE, DELETE,TRUNCATE,TRIGGER, CREATE,CONNECT, ALL.
- **object**: The name of an object to which to grant access. The possible objects are: table, view.
- PUBLIC: A short form representing all users.
- **username**: The name of a user to whom to grant privileges.

Privileges

REVOKE statement

```
REVOKE privilege [, ...]
ON object [, ...]
FROM { PUBLIC | username }
```

- **privilege** values could be: SELECT, INSERT, UPDATE, DELETE,… ALL.
- **object**: The name of an object to which to grant access. The possible objects are: table, view.
- PUBLIC: A short form representing all users.
- **username**: The name of a user to whom to grant privileges.

Privileges

```
GRANT ALL ON COMPANY TO user1;

REVOKE ALL ON COMPANY FROM user1;
```

Views

- A view can represent a subset of a real table, selecting certain columns or certain rows from an ordinary table.
- A view can even represent joined tables.
- Views are not real tables, but appear as ordinary tables to SELECT.

Views

- The general form of view creation is:

```
CREATE [TEMPORARY] VIEW view_name AS
SELECT column1, column2.....
FROM table_name
[WHERE condition];
```

- TEMPORARY: the view will be created in the temporary space. Temporary views are automatically dropped at the end of the current session.

Views

- Example:

```
CREATE VIEW vCOMPANY AS

SELECT ID, NAME, AGE

FROM  COMPANY;

SELECT * FROM vCOMPANY;
```

- To drop view

```
DROP VIEW vCOMPANY;
```

Updatable Views

- To create an updateable view, the SELECT statement which defines view has to follow several rules as follow:
  - SELECT statement must not reference to more than one table. It means it **must not** contain more than **one table** in **FROM** clause, other tables in **JOIN** statement, or **UNION** with other tables.
  - SELECT statement **must not** use **GROUP BY** or **HAVING** clause.
  - SELECT statement **must not** use **DISTINCT** in the selection list.
  - SELECT statement **must not** reference to the **view** that is **not updatable**
  - SELECT statement **must not** contain any **expression** (aggregates, functions, computed columns…)

Updatable Views

- Create updatable view:

```
CREATE VIEW vOfficeInfo
AS SELECT officeCode, phone, city
FROM offices
```

- Then you can run update statement

```
UPDATE vOfficeInfo
 SET phone = '+33 14 723 5555'
WHERE officeCode = 4
```

Views

Views Advantages:

- **Simplify complex query.** A view is defined by an SQL statement with join on many underlying tables with complex business logic. The view now can be used to hides the complexity of underlying tables to the end users and external applications. Only simple SQL statement is used to work with view.
- **Limited access data to the specific users.** You may don't want a subset of sensitive data can be retrievable by all users (both human and applications). You can use view to expose what data to which user to limit the access.

Built-in Functions

- PostgreSQL provides a large number of functions
- It can be classified into:
  - Comparison Functions
  - Mathematical Functions
  - String Functions
  - Data Type Formatting Functions
  - Geometric Functions
  - Network Address Functions
  - XML Functions

Mathematical Functions

Function	Description	Example	Result
abs(x)	absolute value	abs(-17.4)	17.4
cbrt	cube root	cbrt(27.0)	3
Ceil /ceiling	smallest integer not less than argument	ceil(42.8)	43
floor	largest integer not greater than argument	floor(42.8)	42
log	base 10 logarithm	log(100.0)	2
mod(y, x)	remainder of y/x	mod(9,4)	1
power	a raised to the power of b	power(9.0, 3.0)	729

Mathematical Functions

Function	Description	Example	Result
round(v numeric, sint)	round to s decimal places	round(42.4382, 2)	42.44
sqrt()	square root	sqrt(2.0)	1.4142135623731

Mathematical Functions

Function	Description
acos(x)	inverse cosine
asin(x)	inverse sine
atan(x)	inverse tangent
atan2(y, x)	inverse tangent of y/x
cos(x)	cosine
cot(x)	cotangent

String Functions

Function	Description	Example	Result
ascii(string)	ASCII code of the first character of the argument.	ascii('x')	120
concat(str,str2,...)	Concatenate all arguments. NULL arguments are ignored.	concat('abcde', 2, NULL, 22)	abcde222
concat_ws(sep, str1,str2 ,,..)	Concatenate all but first arguments with separators. The first parameter is used as a separator. NULL arguments are ignored.	concat_ws(',', 'abcde', 2, NULL, 22)	Abcde,2,22

String Functions

Function	Description	Example	Result
initcap(string)	Convert the first letter of each word to upper case and the rest to lower case.	initcap('hi THOMAS')	Hi Thomas
left(str text, n int)	Return first n characters in the string.	left('abcde', 2)	ab
right(str text, n int)	Return last n characters in the string.	right('abcde', 2)	de

String Functions

Function	Description	Example	Result
repeat(string text, number int)	Repeat string the specified number of times	repeat('Pg', 4)	PgPgPgPg
replace(string text, from text, to text)	Replace all occurrences in string of substring from with substring to	replace('abcdefabcdef', 'cd', 'XX')	abXXefabXXef
reverse(str)	Return reversed string.	reverse('abcde')	edcba

String Functions

Function	Description	Example	Result
strpos(string, substring)	Location of specified substring (substring in string), but note the reversed argument order)	strpos('high', 'ig')	2
substr(string, from [,count])	Extract substring	substr('alphabet', 3, 2)	ph

String Functions

Function	Description	Example	Result
length(string)	Number of characters in string.	length('jose')	4
trim(string text [,characters text])	Remove the longest string containing only characters from characters (a space by default)	trim('zzzytrimxxy', 'xyz')	trim
rtrim(string text [,characters text])	Remove the longest string containing only characters from characters (a space by default) from the end of string	rtrim('trimxxxx', 'x')	trim
ltrim(string text [,characters text])	Remove the longest string containing only characters from characters (a space by default) from the start of string	ltrim('xxxtrim', 'x')	trim
quote_literal(value)	Convert the given value to text and then quote it as a literal.	quote_literal(42.5)	'42.5’

Date/Time Functions

Function	Description	Example	Result
age(timestamp, timestamp)	Subtract arguments, producing a "symbolic" result that uses years and months	age('2001-04-10', '1957-06-13')	43 years 9 mons 27 days
age(timestamp)	Subtract from current_date	age(timestamp '1957-06-13')	43 years 8 mons 3 days

Date/Time Functions

Function	Description	Example	Result
date_part(text, timestamp)	Get subfield,The valid field names are: century, day, hour,microseconds, minute, month, second, timezone, timezone_hour, timezone_minute, week, year.	date_part('hour', '2001-02-16 20:38:40')	20
now()	Current date and time		
to_char(timestamp,text)	convert time stamp to string	to_char(now(), 'HH12:MI:SS')	

Date/Time Functions

Pattern	Description
HH12	hour of day (01-12)
HH24	hour of day (00-23)
MI	minute (00-59)
YYYY	year (4 and more digits)
YYY	last 3 digits of year
YY	last 2 digits of year
Y	last digit of year

Date/Time Functions

Pattern	Description
MONTH	full upper case month name (blank-padded to 9 chars)
Month	full capitalized month name (blank-padded to 9 chars)
month	full lower case month name (blank-padded to 9 chars)
MON	abbreviated upper case month name (3 chars in English, localized lengths vary)
Mon	abbreviated capitalized month name (3 chars in English, localized lengths vary)
mon	abbreviated lower case month name (3 chars in English, localized lengths vary)
MM	month number (01-12)

Date/Time Functions

Pattern	Description
MONTH	full upper case month name (blank-padded to 9 chars)
Month	full capitalized month name (blank-padded to 9 chars)
month	full lower case month name (blank-padded to 9 chars)
MON	abbreviated upper case month name (3 chars in English, localized lengths vary)
Mon	abbreviated capitalized month name (3 chars in English, localized lengths vary)
mon	abbreviated lower case month name (3 chars in English, localized lengths vary)
MM	month number (01-12)

Date/Time Functions

Pattern	Description
DAY	full upper case day name
Day	full capitalized day name
day	full lower case day name
DY	abbreviated upper case day name (3 chars)
Dy	abbreviated capitalized day name
dy	abbreviated lower case day name
DDD	day of year (001-366)
DD	day of month (01-31)
D	day of the week, Sunday(1) to Saturday(7)

Date/Time Functions

Pattern	Description
W	week of month (1-5) (The first week starts on the first day of the month.)
WW	week number of year (1-53) (The first week starts on the first day of the year.)