The SQL language

NGUYEN Hong Phuong
phuongnh@soict.hut.edu.vn

1

Contents

2

Data types in PostGreSQL

Name	Alias	Description
bigint	int8	signed eight-byte integer
bigserial	serial8	autoincrementing eight-byte integer
bit[(n)]		fixed length bit string
bit varying[(n)]	varbit	variable length bit string
boolean	bool	logical boolean (true/false)
character varying[(n)]	varchar[(n)]	variable length character string
character[(n)]	char[(n)]	fixed length character string

3

Data types in PostGreSQL (cont)

Name	Alias	Description
date		calendar date (year, month, day)
double precision	float8	double precison floating point number
integer	int,int4	signed four byte integer
money		currency amount
numeric[(p,s)]	decimal[(p,s)]	exact numeric of selectable precision
real	float4	single precision floating point number
smallint	int2	signed two byte integer

4

Data types in PostGreSQL (cont)

Name	Alias	Description
serial	serial4	autoincrementing four-byte integer
text		variable length character string
time[(p)]{without time zone}		time of day
time[(p)]{with time zone}	timez	time of day, including time zone
timestamp[(p)]{without timezone}		date and time
timestamp[(p)]{with timezone}	timestamptz	date and time, including time zone

5

Creating a new table

```
-- Table: "DeTai"
-- DROP TABLE "DeTai";
CREATE TABLE "DeTai"(
  "DT#" character(4) NOT NULL,
  "TenDT" character varying(50) NOT NULL,
  "Cap" character(12) NOT NULL,
  "KinhPhi" integer,
  CONSTRAINT "KhoaChinhDeTai" PRIMARY KEY ("DT#")
);
ALTER TABLE "DeTai" OWNER TO postgres;
COMMENT ON TABLE "DeTai" IS 'Bảng Đề tài chứa thông tin về tên
      đề tài, cấp quản lý và kinh phí';
```

6

Creating a new table

```
-- Table: "GiangVien"
-- DROP TABLE "GiangVien";
CREATE TABLE "GiangVien"(
  "GV#" character(4) NOT NULL,
  "HoTen" character(30) NOT NULL,
  "DiaChi" character varying(50) NOT NULL,
  "NgaySinh" date NOT NULL,
  CONSTRAINT "KhoaChinhGiangVien" PRIMARY KEY
     ("GV#")
);
ALTER TABLE "GiangVien" OWNER TO postgres;
COMMENT ON TABLE "GiangVien" IS 'Bảng Giảng viên chứa
     thông tin về giảng viên';
```

7

Creating a new table

```
-- Table: "ThamGia"
-- DROP TABLE "ThamGia";
CREATE TABLE "ThamGia"(
  "GV#" character(4) NOT NULL,
  "DT#" character(4) NOT NULL,
  "SoGio" smallint,
  CONSTRAINT "KhoaChinhThamGia" PRIMARY KEY ("GV#",
     "DT#"),
  CONSTRAINT "KhoaNgoai1" FOREIGN KEY ("GV#")
     REFERENCES "GiangVien" ("GV#") MATCH SIMPLE
     ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT "KhoaNgoai2" FOREIGN KEY ("DT#")
     REFERENCES "DeTai" ("DT#") MATCH SIMPLE
     ON UPDATE CASCADE ON DELETE CASCADE
);
ALTER TABLE "ThamGia" OWNER TO postgres;
```

8

Another simple example

```
CREATE TABLE weather (
city varchar(80),
temp_lo int, -- low temperature
temp_hi int, -- high temperature
prcp real, -- precipitation
date date
);
```

9

```
CREATE TABLE cities (
name varchar(80),
location point
);
```

10

Populating a Table With Rows

- The INSERT statement is used to populate a table with rows:
  - INSERT INTO weather VALUES ('San Francisco', 46, 50, 0.25, '1994-11-27');
  - INSERT INTO cities VALUES ('San Francisco', '(-194.0, 53.0)');
  - INSERT INTO weather (city, temp_lo, temp_hi, prcp, date) VALUES ('San Francisco', 43, 57, 0.0, '1994-11-29');
  - INSERT INTO weather (date, city, temp_hi, temp_lo) VALUES ('1994-11-29', 'Hayward', 54, 37);

11

Querying a Table

- To retrieve data from a table, the table is queried. An SQL SELECT statement is used to do this.
- to retrieve all the rows of table weather, type:
  - SELECT * FROM weather;
  - SELECT city, temp_lo, temp_hi, prcp, date FROM weather;
  - SELECT city, (temp_hi+temp_lo)/2 AS temp_avg, date FROM weather;

12

- SELECT * FROM weather WHERE city = 'San Francisco' AND prcp > 0.0;
- SELECT * FROM weather ORDER BY city;
- SELECT * FROM weather ORDER BY city, temp_lo;
- SELECT DISTINCT city FROM weather;
- SELECT DISTINCT city FROM weather ORDER BY city;

13

Joins Between Tables

SELECT * FROM weather, cities WHERE city = name;

SELECT city, temp_lo, temp_hi, prcp, date, location FROM weather, cities WHERE city = name;

SELECT weather.city, weather.temp_lo, weather.temp_hi, weather.prcp, weather.date, cities.location FROM weather, cities WHERE cities.name = weather.city;

14

- Join queries of the kind seen thus far can also be written in this alternative form:
  - SELECT * FROM weather INNER JOIN cities ON (weather.city = cities.name);
  - SELECT * FROM weather LEFT OUTER JOIN cities ON (weather.city = cities.name);

15

- We can also join a table against itself. This is called a self join.

```
SELECT W1.city, W1.temp_lo AS low,
W1.temp_hi AS high, W2.city, W2.temp_lo
AS low, W2.temp_hi AS high
FROM weather W1, weather W2
WHERE W1.temp_lo < W2.temp_lo AND
W1.temp_hi > W2.temp_hi;
```

16

```
SELECT *
FROM weather w, cities c
WHERE w.city = c.name;
```

17

Aggregate Functions

- PostgreSQL supports aggregate functions.
- An aggregate function computes a single result from multiple input rows.
- there are aggregates to compute the count, sum, avg (average), max (maximum) and min (minimum) over a set of rows
  - SELECT max(temp_lo) FROM weather;

18

- SELECT city FROM weather WHERE temp_lo = max(temp_lo); --WRONG
- SELECT city FROM weather WHERE temp_lo = (SELECT max(temp_lo) FROM weather);

- Aggregates are also very useful in combination with GROUP BY clauses.
  - SELECT city, max(temp_lo) FROM weather GROUP BY city;

19

```
SELECT city, max(temp_lo)
FROM weather
GROUP BY city
HAVING max(temp_lo) < 40;
```

```
SELECT city, max(temp_lo)
FROM weather
WHERE city LIKE 'S%'
GROUP BY city
HAVING max(temp_lo) < 40;
```

20

Updates

- You can update existing rows using the UPDATE command
  - UPDATE weather SET temp_hi = temp_hi - 2, temp_lo = temp_lo – 2 WHERE date > '1994-11-28';

21

Deletions

- Rows can be removed from a table using the DELETE command
  - DELETE FROM weather WHERE city = 'Hayward';
- Without a qualification, DELETE will remove all rows from the given table, leaving it empty
  - DELETE FROM tablename;

22