SQL

Зачем это всё?

Содержание

- Реляционные БД
- CREATE TABLE, TRUNCATE, DROP
- INSERT INTO, UPDATE, DELETE
- SELECT
- GROUP BY, агрегация
- JOIN
- Оконные функции
- Транзакции
- SQL и Python

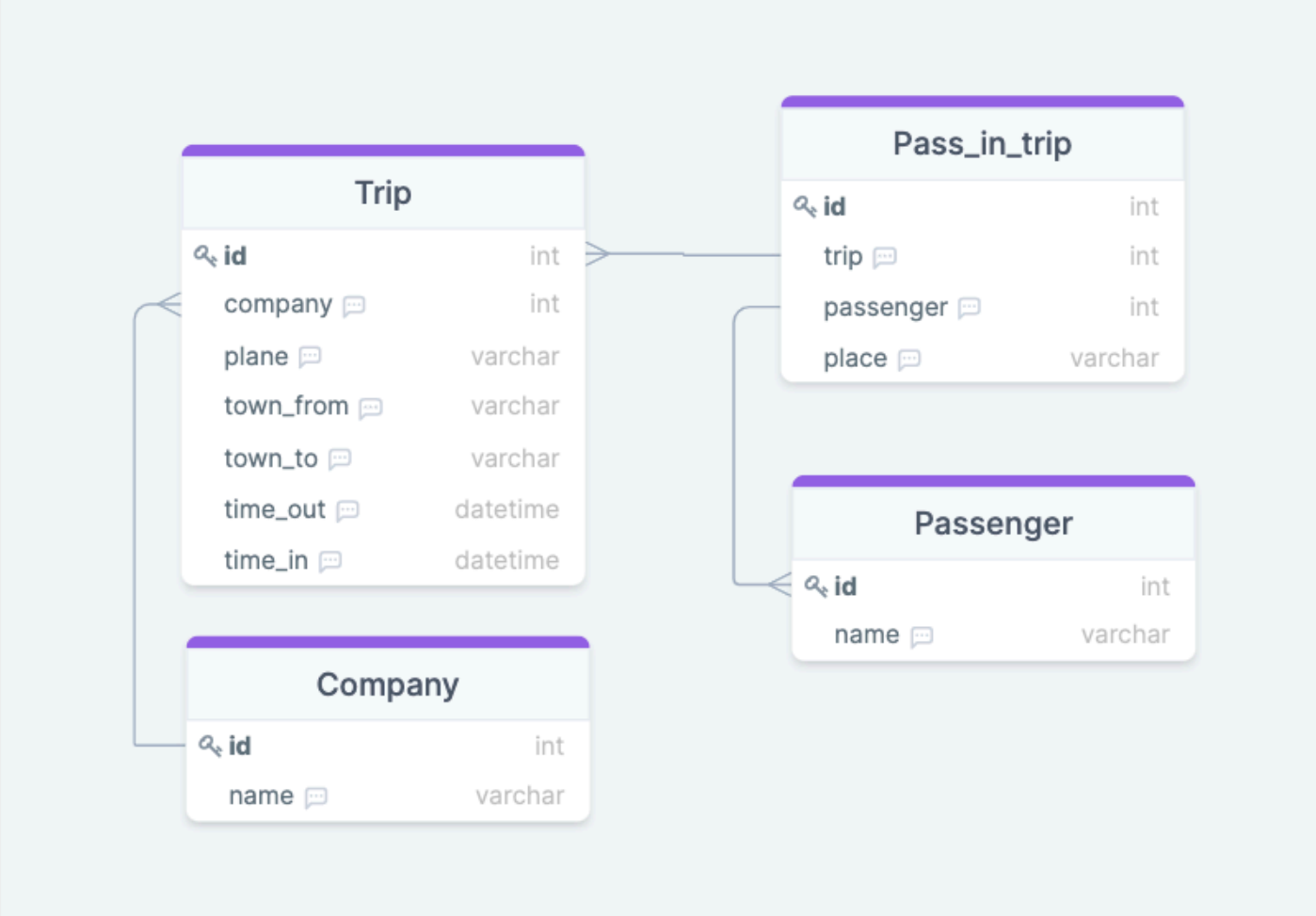
Содержание

- Реляционные БД
- CREATE TABLE, TRUNCATE, DROP
- INSERT INTO, UPDATE, DELETE
- SELECT
- GROUP BY, агрегация
- JOIN
- Оконные функции
- Транзакции
- SQL и Python

Содержание

- 6 февраля: синтаксис SQL
- 8 февраля: разбираем задачки (~40 мин), транзакции, Python
- 10 февраля: разбираем задачки

Реляционные БД



CREATE TABLE

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255)
);
```

```
DROP TABLE Persons;
```

```
TRUNCATE TABLE Persons;
```

ALTER TABLE

- **ALTER TABLE** Customers **ADD** Email varchar(255);

CustomerID	CustomerName	ContactName	Address	City	PostalCode	Country
89	White Clover Markets	Karl Jablonski	305 - 14th Ave. S. Suite 3B	Seattle	98128	USA
90	Wilman Kala	Matti Karttunen	Keskuskatu 45	Helsinki	21240	Finland
91	Wolski	Zbyszek	ul. Filtrowa 68	Walla	01-012	Poland
92	Cardinal	Tom B. Erichsen	Skagen 21	Stavanger	4006	Norway

INSERT INTO

- **INSERT INTO**
  Customers
  (CustomerName, ContactName, Address, City, PostalCode, Country)
  VALUES
  ('Cardinal', 'Tom B. Erichsen', 'Skagen
  21', 'Stavanger', '4006', 'Norway'),
  ('Cardinal', 'Tom B. Erichsen', 'Skagen
  21', 'Stavanger', '4006', 'Norway');

UPDATE, DELETE

- **UPDATE** Customers
  SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
  WHERE CustomerID = 1;
- **DELETE FROM** Customers
  WHERE CustomerID = 1 or ContactName = 'John'

SELECT

- **SELECT** CustomerName, City **FROM** Customers;
- **SELECT** * **FROM** Customers;
- **SELECT** * **FROM** Customers **ORDER BY** CustomerID;
- **SELECT** * **FROM** Customers **ORDER BY** CustomerID **DESC**;
- **SELECT** * **FROM** Customers **ORDER BY** CustomerID **LIMIT** 10;

SELECT + WHERE

- **SELECT** CustomerName, City **FROM** Customers **WHERE**

  Country = 'USA'

- **SELECT** CustomerName, City **FROM** Customers **WHERE**

  Country = 'USA’ **OR** Country = 'Finland'

- **SELECT** CustomerName, City **FROM** Customers **WHERE**

  Country NOT IN ('USA', 'Finland')

SELECT + WHERE

- **SELECT** CustomerName, City **FROM** Customers **WHERE**

  Country LIKE 'U%’

- **SELECT** * **FROM** Students **WHERE**

  Age **BETWEEN** 3 **AND** 10

Вставляем результат одного запроса в другую таблицу

- **INSERT INTO** *table2*
  SELECT * **FROM** *table1*
  WHERE *condition*;
- **INSERT INTO** *table2* **WITH TRUNCATE**
  SELECT * **FROM** *table1*
  WHERE *condition*;

GROUP BY

- Считаем число заказов по городам:

  SELECT city, **COUNT**(*) **AS** orders
  FROM all_orders_table
  GROUP BY city

  COUNT – агрегационная функция (комбинирует несколько значений в одно)

GROUP BY

- Считаем число заказов по городам:

  SELECT city, **COUNT**(*) **AS** orders
  FROM all_orders_table
  GROUP BY city

  COUNT – агрегационная функция (комбинирует несколько значений в одно)
- Знаете еще такие функции?

GROUP BY

- Считаем число заказов по городам:

  SELECT city, **COUNT**(*) **AS** orders
  FROM all_orders_table
  GROUP BY city

  COUNT – агрегационная функция (комбинирует несколько значений в одно)
- Знаете еще такие функции? MIN, MAX, AVG, SUM, *MEDIAN*

```
SELECT State, AVG(# of friends)
FROM facebook
GROUP BY State
```

facebook

Name	State	# of Friends
Matt	CA	300
Lisa	CA	500
Jeff	CA	600
Sarah	FL	400

final table

State	AVG(# of friends)
CA	0
FL	0

```
SELECT State, City, COUNT(*)
FROM facebook
GROUP BY State, City
```

facebook

Name	State	City	# of Friends
Matt	CA	SF	300
Lisa	CA	SF	500
Jeff	CA	LA	600
Sarah	FL	MIA	400

final table

State	City	COUNT(*)
CA	SF	0
CA	LA	0
FL	MIA	0

GROUP BY + HAVING

- Считаем число заказов по городам:

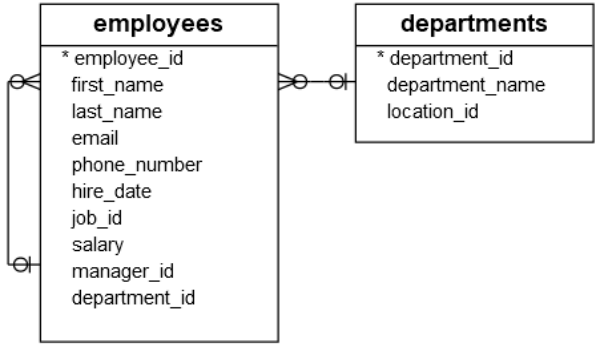
```
SELECT city, COUNT(*) as orders
FROM all_orders_table
GROUP BY city
HAVING LENGTH(city) > 5
```

GROUP BY + HAVING

- Считаем число заказов по городам:

```
SELECT city, COUNT(DISTINCT user_id) AS users
FROM all_orders_table
GROUP BY city
HAVING LENGTH(city) > 5
```

Подзапросы



- Как посчитать среднюю зарплату по отделам (то есть среднее средних)?

Подзапросы

```
SELECT
        ROUND(AVG(average_salary), 0)
 FROM (
        SELECT
                AVG(salary) average_salary
        FROM employees
        GROUP BY department_id
) AS department_salary
```

JOIN

OrderID	CustomerID	OrderDate
10308	2	1996-09-18
10309	37	1996-09-19
10310	77	1996-09-20

CustomerID	CustomerName	ContactName	Country
1	Alfreds Futterkiste	Maria Anders	Germany
2	Ana Trujillo Emparedados y helados	Ana Trujillo	Mexico
3	Antonio Moreno Taquería	Antonio Moreno	Mexico

JOIN

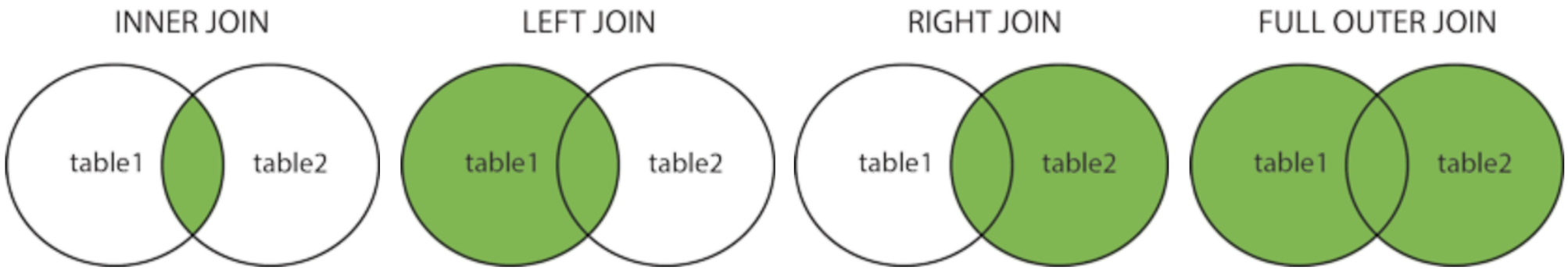
- **SELECT** Orders.OrderID, Customers.CustomerName, Orders.OrderDate
  FROM Orders
  INNER JOIN Customers
  ON Orders.CustomerID = Customers.CustomerID;

OrderID	CustomerName	OrderDate
10308	Ana Trujillo Emparedados y helados	9/18/1996
10365	Antonio Moreno Taquería	11/27/1996
10383	Around the Horn	12/16/1996
10355	Around the Horn	11/15/1996
10278	Berglunds snabbköp	8/12/1996

SQL JOINS Explained

ZMWolski

JOIN


INNER JOIN
table1
table2
LEFT JOIN
table1
table2
RIGHT JOIN
table1
table2
FULL OUTER JOIN
table1
table2

А как работают джойны?

- Inner loops join
- Hash join
- Merge join

Inner loops join

- Построчно сравниваем строки таблиц, выводим совпавшие

Inner loops join

- Построчно сравниваем строки таблиц, выводим совпавшие
- O(NM), N, M – кол-во строк в одной и другой таблице

Merge join

- Вспоминаем merge sort

Merge join

- Вспоминаем merge sort
- Допустим, что колонки отсортированы по ключу join’а

Merge join

- Вспоминаем merge sort
- Допустим, что колонки отсортированы по ключу join’а
- Получаем join за линейное (O(N + M)) время

Merge join

- Вспоминаем merge sort
- Допустим, что колонки отсортированы по ключу join’а
- Получаем join за линейное (O(N + M)) время
- Но сортированность есть не всегда и она не бесплатная

Hash join

- Пусть M <= N, M – число строк в первой таблице
- Собираем хеш-таблицу по первой таблице за O(M)
- Ищем строки из второй таблице в хеш-таблице
- Получаем сложность O(M + N), но нужна дополнительная память

Hash join

- Пусть M <= N, M – число строк в первой таблице
- Собираем хеш-таблицу по первой таблице за O(M)
- Ищем строки из второй таблице в хеш-таблице
- Получаем сложность O(M + N), но нужна дополнительная память
- Хороший алгоритм, если одна из таблиц очень маленькая

Running total mileage visual		
Day	**Miles Driving**	**Running Total**
Jan. 1	60	
Jan. 2	80	
Jan. 3	10	
Jan. 4	150	

Оконные функции

```
SELECT
        Day,
        MilesDriving,
        SUM(MilesDriving) OVER(ORDER BY Day) AS RunningTotal
FROM table;
```

Оконные функции

Running Average Example		
Day	**Daily Revenue**	**3 Day Average**
1	39	
2	528	
3	39	
4	86	
5	86	
6	351	

Оконные функции

```
SELECT
        Day
        DailyRevenue,
AVG(Daily Revenue) OVER(ORDER BY 'Day' ROWS 2 PRECEDING) AS 3DayAverage
FROM table;
```

Оконные функции

- Хотим посчитать среднее время между заказами для каждого пользователя
- Как это сделать?

Оконные функции

- Хотим посчитать среднее время между заказами для каждого пользователя
- Как это сделать?
- Нужно для каждой строчки находить предыдущую

Оконные функции

```
select
   user_id,
   order_datetime,
   lag(order_datetime) as prev_order_datetime over (partition by user_id order by order_datetime)
from orders
```

А еще есть условные выражения

Почитайте про них сами:
https://www.postgresql.org/docs/9.4/functions-conditional.html

Ещё почитать

- https://www.w3schools.com/sql/
- https://proglib.io/p/sql-for-20-minutes/
- https://www.postgresql.org/docs/9.1/tutorial-window.html
- https://habr.com/ru/post/268983/

И немного мотивации:

- https://www.linkedin.com/pulse/why-you-should-learn-sql-brewster-knowlton/
- https://www.dataquest.io/blog/why-sql-is-the-most-important-language-to-learn/