ДПО ПК "Аналитик данных"
SQL

Реляционная база данных - это база, в которой данные хранятся с использованием отношений или таблиц, которые в свою очередь организовывают хранение информации, полагаясь на столбцы и строки

Отношения между таблицами задаются за счет таких понятий, как первичный и внешний ключ

Первичный ключ служит для опеределения уникальной записи в рамках своей таблицы

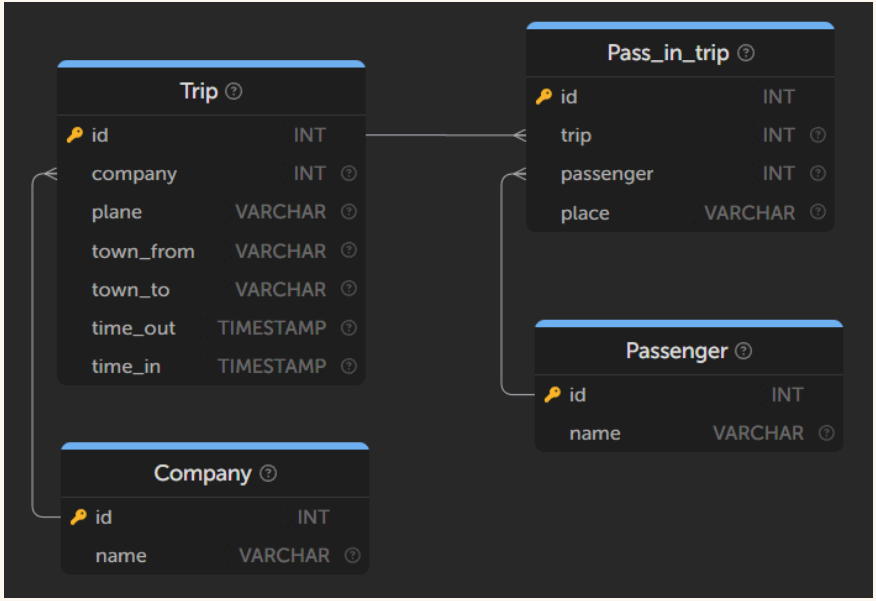
Внешний ключ указывает, что текущая таблица имеет соответствие со строкой из другой таблицы по соответствующему значению ключа

SELECT

Оператор, который позволяет выбрать столбцы для вывода данных в результате запроса

FROM

Оператор, который позволяет указать, из какой таблицы базы данных выбирать записи


Trip
id INT
company INT
plane VARCHAR
town_from VARCHAR
town_to VARCHAR
time_out TIMESTAMP
time_in TIMESTAMP
Pass_in_trip
id INT
trip INT
passenger INT
place VARCHAR
Passenger
id INT
name VARCHAR
Company
id INT
name VARCHAR

```
select *
from Trip;
```

DISTINCT

Ключевое слово, использование которого позволяет отобрать только уникальные значения по заданным столбцам

WHERE

Оператор, который используется для фильтрации первичных данных

```sql
select *
from Trip
where town_to = 'Moscow';
```

ORDER BY

Оператор, который позволяет задать сортировку выводимых данных по указанным столбцам

```sql
select *
from Trip
where town_to = 'Moscow'
order by time_out desc, plane asc;
```

GROUP BY

Оператор, который применяется для указания полей, по которым будет производится группировка для подсчета агрегирующих функций

Агрегирующие функции позволяют посчитать различные показатели поверх первичных данных

Например, SUM, AVG, COUNT, MIN, MAX

```sql
select
    town_from,
    count(*) as rows_cnt,
    count(distinct plane) as plane_cnt,
    min(time_out) as time_out_min,
    max(time_out) as time_out_max
from Trip
group by town_from
order by count(*) desc;
```

HAVING

Оператор, который позволяет отфильтровать сагрегированные данные

```sql
select
    town_from,
    count(*) as rows_cnt,
    count(distinct plane) as plane_cnt,
    min(time_out) as time_out_min,
    max(time_out) as time_out_max
from Trip
group by town_from
having count(*) > 4
order by count(*) desc;
```

LIMIT

Оператор, который используется для отбора определенного количества строк из таблицы

```
select
    *
from Trip
limit 3;
```

Порядок интерпретации SQL запроса на выборку данных:

1. **FROM** — определение таблиц, участвующих в запросе
2. **WHERE** — фильтрация строк на основании заданных условий
3. **GROUP BY** — группировка результатов по указанным столбцам
4. **HAVING** — дополнительное условие фильтрации над группами
5. **SELECT** — выбор нужных столбцов из таблицы
6. **ORDER BY** — сортировка результата по столбцам
7. **LIMIT** — ограничение количества возвращаемых записей

Присоединения позволяют соединить данные из нескольких таблиц по ряду внешних ключей

LEFT JOIN
RIGHT JOIN
INNER JOIN
CROSS JOIN
FULL OUTER JOIN

```sql
select distinct c.name
from Company as c
    left join Trip as t on c.id = t.company
where t.town_to = 'Moscow'
order by c.name asc;
```

Объединения позволяют склеить данные из нескольких таблиц с учетом соответствия столбцов

UNION
UNION ALL

```sql
select distinct c.name
from Company as c
    left join Trip as t on c.id = t.company
where t.town_to = 'Moscow'
union
select 'MyAeroCompany' as name
order by name asc;
```

COMMON TABLE EXPRESSIONS or CTE позволяют придавать структуру запросам к нескольким таблицам и прихранивать промежуточные результаты в оперативной памяти

```sql
with
cte as (
    select distinct c.name
    from Company as c
        left join Trip as t on c.id = t.company
    where t.town_to = 'Moscow'
)
select 'MyAeroCompany' as name
union
select name from cte
order by name asc;
```

SUBQUERIES позволяют использовать вложенные запросы

```sql
select *
from Trip
where plane in (
    select plane
    from Trip
    group by plane
    order by count(*) desc
    limit 1
);
```

WINDOW FUNCTIONS или оконные функции позволяют посчитать агрегирующую метрику поверх данных заданного окна

row_number, rank, dense_rank, lag, lead, sum, count, avg...

```sql
with
cte as (
    select
        *,
        row_number() over(partition by plane order by
            time_out asc) as rn
    from Trip
)
select
    id,
    company,
    plane,
    town_from,
    town_to
from cte
where rn = 1
order by id asc;
```

DDL или операции по определению данных

```sql
CREATE TABLE customers (
    id SERIAL PRIMARY KEY,
    first_name VARCHAR(50),
    last_name VARCHAR(50),
    email VARCHAR(100) UNIQUE NOT NULL
);
```

```
CREATE INDEX idx_email ON customers(email);
```

```
CREATE VIEW customer_emails AS
SELECT id, email FROM customers;
```

```
ALTER TABLE customers ADD COLUMN phone_number VARCHAR(20);
```

```
ALTER TABLE customers DROP COLUMN phone_number;
```

```
DROP TABLE customers CASCADE;
```

```
DROP INDEX idx_email;
```

```
DROP VIEW customer_emails;
```

DML или операции по изменению данных

```
INSERT INTO users(username, email, age) VALUES ('ivanov', 'ivan@example.com', 30);
```

```
INSERT INTO users(username, email, age) VALUES
('sidorov', 'sido@mail.ru', 28),
('kuznetsov', 'ku@ya.ru', 45);
```

```sql
UPDATE users SET age = 31 WHERE username = 'ivanov';
```

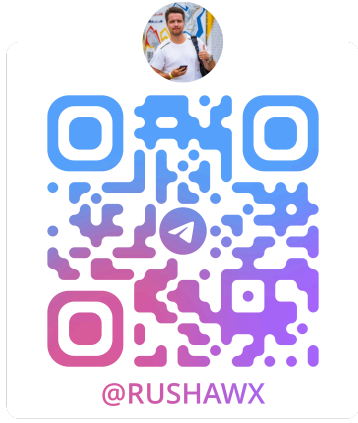
```sql
DELETE FROM users WHERE username = 'petrov';
```

Ресурсы для отработки навыка написания SQL запросов

https://sql-ex.ru/
https://stepik.org/course/63054
https://sql-academy.org/ru
https://leetcode.com/

Хорошая небольшая настольная книжка по SQL
https://www.piter.com/collection/all/product/sql-pocket-guide-4-e-izd

Книги по PostgreSQL
https://postgrespro.ru/education/books


@RUSHAWX