Лекция 3.

Язык SQL - продолжение

CREATE
READ (RETRIEVE)
UPDATE
DELETE

Создание таблиц в БД

Создание таблиц

CREATE TABLE *имя* (*столбцы*)

Имя таблицы, список столбцов (тип и размер данных), ограничения

Ограничения - дополнительные условия, накладываемые на один или несколько столбцов.

- **Ограничения первичного ключа.** Гарантируется уникальность на уровне таблицы значений столбца(ов).
- **Ограничения внешнего ключа.** Столбец может содержать только значения из первичного ключа другой таблицы.
- **Ограничения уникальности.** Первичный ключ - частный случай
- **Проверочные ограничения.** Ограничивают допустимые значения для столбца.

Задание связей и ограничений


```
CREATE TABLE `order` (
  `id` INTEGER PRIMARY KEY,
  `summa` NUMERIC NOT NULL,
  `order_date` DATE NOT NULL,
  `customer_id` INTEGER NOT NULL,
  `vendor_id` INTEGER NOT NULL,
  FOREIGN KEY (`vendor_id`) REFERENCES `vendor` (`id`)
  ON DELETE RESTRICT
);

CREATE TABLE `customer` (
  `id` INTEGER PRIMARY KEY,
  `name` VARCHAR(30) NOT NULL,
  `city` VARCHAR(30) NOT NULL,
  `rating` INTEGER NOT NULL CHECK (`rating`>0),
  `vendor_id` INTEGER NULL,
  FOREIGN KEY (`vendor_id`) REFERENCES `vendor` (`id`)
  ON DELETE SET NULL
);
```

Метаданные

В SQLite метаданные хранятся в таблице sqlite_master

Команды .table и .schema

Вставка, изменение и удаление данных в таблицах

Добавление строк

INSERT INTO имя_таблицы [(имя_столбца, ...)] **VALUES** (значение, ...), ...

INSERT INTO имя_таблицы [(имя_столбца, ...)] **SELECT** (...)

```
INSERT INTO `vendor` (`id`, `name`, `city`, `percent`) VALUES
(1001, 'Иванов', 'Саранск', 12),
(1002, 'Петров', 'Москва', 13),
(1003, 'Андреев', 'Кострома', 10),
(1004, 'Сидоров', 'Саранск', 10);
```

INSERT, UPDATE, DELETE

Вставка строк в таблицу

INSERT INTO vendor(name, city, percent) VALUES (“Иванов", "Тверь", 10)

INSERT INTO … SELECT …

Изменение строк

UPDATE vendor SET name = “Шахов” WHERE name = “Иванов”

Удаление строк

DELETE FROM vendor WHERE name = “Шахов”

Выборка данных из таблиц

Структура оператора SELECT

SELECT	Определяет, какие столбцы включить в результирующий набор
FROM	Определяет таблицы, из которых выбираются данные, и которые нужно соединить друг с другом
WHERE	Отсеивает ненужные данные
GROUP BY	Используется для группировки строк по общим значениям столбцов
HAVING	Отсеивает ненужные группированные данные
ORDER BY	Сортирует строки результирующего набора по одному или нескольким столбцам

SELECT … FROM … WHERE … GROUP BY … HAVING … ORDER BY …

В SELECT можно включать:

- Столбцы таблиц
- Литералы (числа строки)
- Вычисляемые выражения
- Вызов встроенных и пользовательских функций

SELECT 10 ‘Просто число’, **name** ‘Имя’, **Length(city)** ‘Длина имени города’, **percent || ‘%’** ‘Процент’ **FROM** Vendor

Предложение FROM может отсутствовать

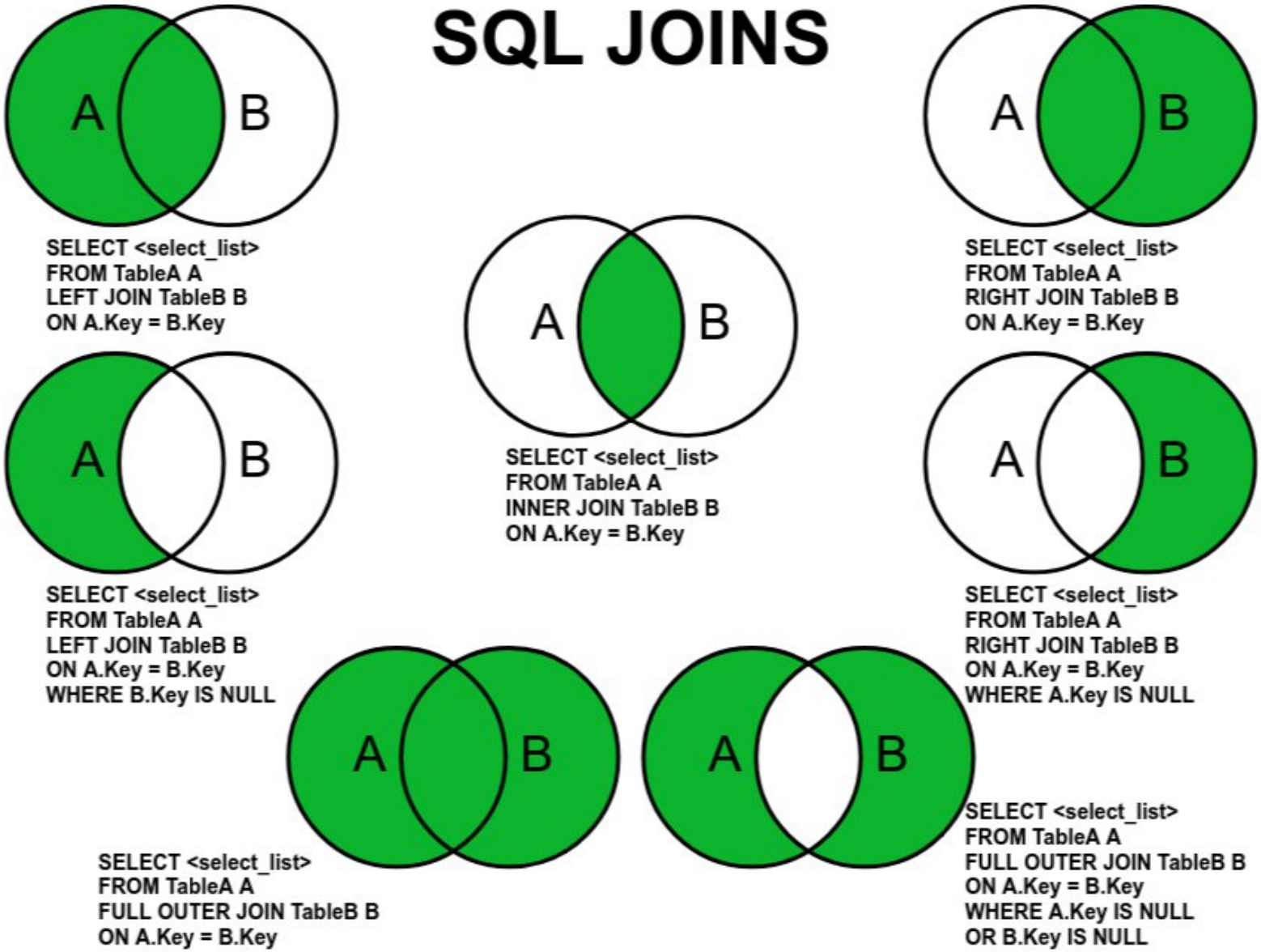
SELECT … **FROM** … WHERE … GROUP BY … HAVING … ORDER BY …

FROM определяет таблицы, используемые запросом, и средства связывания таблиц вместе.

Таблица = набор связанных строк:

- **Постоянные.** CREATE TABLE …
- **Производные.** Возвращаются подзапросом и хранятся в памяти. (SELECT …)
- **Временные.** Изменяемые данные, хранящиеся в памяти. CREATE TEMPORARY TABLE…
- **Виртуальные.** Представления (views). CREATE VIEW …

Связь таблиц внутри FROM



Соединение таблиц

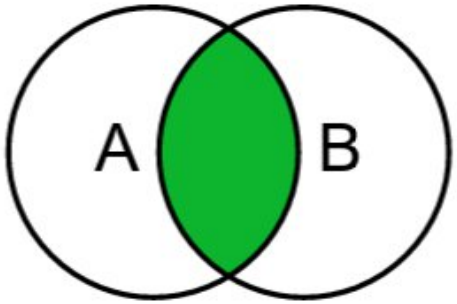
Соединение по равенству (внутреннее соединение)

Условия соединения и фильтрация внутри WHERE

```
SELECT customer.name, vendor.name, vendor.city
FROM vendor, customer
WHERE vendor.city=customer.city AND vendor.percent > 11
```

Условие соединения внутри FROM, условие фильтрации внутри WHERE

```
SELECT customer.name, vendor.name, vendor.city
FROM vendor
      INNER JOIN customer ON vendor.city=customer.city
WHERE vendor.percent > 11
```



Соединение трёх и более таблиц

```
SELECT customer.name 'Покупатель', vendor.name 'Продавец', vendor.city,
`order`.order_date, `order`.summa
FROM vendor
  INNER JOIN customer
    ON vendor.city=customer.city
  INNER JOIN `order`
    ON `order`.customer_id=customer.id;
```

Соединение с результатом подзапроса

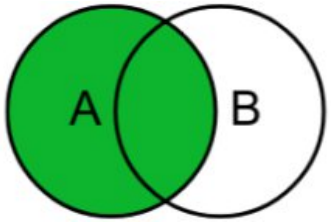
```
SELECT cust_ord.name 'Покупатель', vendor.name 'Продавец', vendor.city,
cust_ord.order_date, cust_ord.summa
FROM vendor
  INNER JOIN
  (SELECT customer.city, customer.name, `order`.order_date, `order`.summa
   FROM customer
     INNER JOIN `order`
       ON `order`.customer_id=customer.id) cust_ord
   ON vendor.city=cust_ord.city;
```

Внешнее соединение таблиц

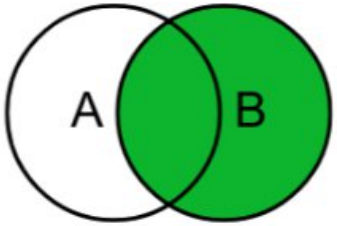
В отличие от внутренних объединений, которые связывают строки двух таблиц, внешние объединения включают в результат также строки, не имеющие пар.

Вывести все пары Продавец-Покупатель (в том числе продавцов, не имеющих покупателей)

```
SELECT vendor.name, customer.name
FROM vendor LEFT OUTER JOIN customer
ON vendor.id = customer.vendor_id
```




```
SELECT vendor.name, customer.name
FROM customer RIGHT OUTER JOIN vendor
ON vendor.id = customer.vendor_id
```



Перекрестное соединение таблиц

CROSS JOIN

Декартово произведение двух множеств (таблиц)

SELECT … FROM … WHERE … **GROUP BY** … **HAVING** … ORDER BY …

Данные можно агрегировать и группировать, тогда мы будем работать с ними на более высоком уровне (с меньшей детализацией), чем они хранятся в базе.

```
SELECT vendor_id FROM `order`;

SELECT vendor_id FROM `order` GROUP BY vendor_id;

SELECT vendor_id, count(*) FROM `order` GROUP BY vendor_id;

SELECT vendor_id, count(*) FROM `order` GROUP BY vendor_id
                                        HAVING count(*) > 1;
```

Подзапросы

Подзапросы

Подзапрос - это запрос, помещённый (в круглых скобках) в другую инструкцию SQL.

- Некоррелированные (вложенные) подзапросы. Выполняются первыми, независимо от содержащего запроса.
- Коррелированные (связанные) подзапросы. Выполняются в составе содержащего запроса, ссылаются на один или несколько столбцов из содержащей инструкции.

Вложенные (некоррелированные) запросы

Внутри HAVING можно применять подзапросы, которые не дают множества значений.

Найти количество покупателей с рейтингом, превышающим среднее значение для покупателей из Саранска

```
SELECT rating, COUNT(DISTINCT id) FROM customer
GROUP BY rating
HAVING rating > (SELECT AVG(rating)
FROM customer WHERE city="Саранск")
```

Связанные (коррелированные) подзапросы

Найти всех покупателей, сделавших заказы 10.12.2016

```
SELECT * FROM customer a
WHERE “2016-12-10” IN
   (SELECT order_date FROM order b WHERE a.id = b.customer_id)
```

Внутренний запрос должен выполняться отдельно для каждой строки внешнего запроса.

1. Выбирается текущая строка-кандидат из таблицы, указанной во внешнем запросе.
2. Значение этой строки сохраняется в псевдониме из FROM внешнего запроса.
3. Выполняется подзапрос. Каждый раз, когда встречается псевдоним для внешнего запроса, его значение применяется к текущей строке-кандидату (внешней ссылке).
4. Оценивается предикат внешнего запроса на основе подзапроса, выполненного на шаге 3 (будет ли строка-кандидат включена в выходные данные).
5. Процедура повторяется для следующей строки-кандидата.

Связанные (коррелированные) подзапросы

1. **Найдем имена и номера всех продавцов, имеющих более одного покупателя**

```
SELECT id, name
FROM vendor main
WHERE 1 < (SELECT COUNT(*) FROM customer WHERE vendor_id=main.id)
```

2. **Найдем все заказы, сумма в которых превышает среднюю сумму заказа для данного покупателя**

```
SELECT *
FROM order a
WHERE summa > (SELECT AVG(summa) FROM `order` b
      WHERE a.customer_id = b.customer_id)
```

Подзапросы

Могут использоваться:

- В разделе WHERE операторов SELECT, UPDATE, DELETE.
- В разделе SET оператора UPDATE.
- В разделе VALUES оператора INSERT. Должны быть некоррелированными скалярными.
- В разделе FROM оператора SELECT. Должны быть некоррелированными.
- В разделе HAVING оператора SELECT … FROM … GROUP BY …
- В разделе ORDER BY оператора SELECT. Подзапрос должен быть скалярным.

Подзапросы в условии WHERE

Подзапрос возвращает:

- **Одна строка, один столбец.** Скалярный подзапрос, можно использовать в условиях равенства.
- **Несколько строк, один столбец.** В проверках на включение в множество.
- **Несколько столбцов.** В проверках на включение в множество.

Подзапросы в разделе FROM

Подзапрос выступает в качестве источника данных (формирует временную таблицу)

Пример. Для всех покупателей определить общее количество и сумму купленного товара, в зависимости от суммы отнести покупателя к одной из трех групп: «Мало» (до 100 рублей), «Средне» (от 100 до 500 рублей), «Много» (более 500 рублей).

Подзапросы в разделе ORDER BY

Вывести фамилии покупателей, отсортированные по количеству их покупок.

```
SELECT c.name
FROM customer c
ORDER BY
(SELECT count(*) FROM `order` ord
   WHERE c.id=ord.customer_id) DESC;
```

Подзапросы в инструкции INSERT

Внести в таблицу order запись о покупке, если известны фамилии продавца и покупателя.

```
INSERT INTO `order` (summa, order_date, customer_id, vendor_id)
VALUES (200.25, '2021-09-21', (SELECT id FROM customer WHERE
name='Ли'), (SELECT id FROM vendor WHERE name='Петров'));
```

Подзапросы - итоги

- Либо не зависят от содержащего запроса, либо ссылаются на столбцы из содержащей инструкции.
- Используются в условиях WHERE в операторах сравнения и операторах IN, NOT IN, EXISTS, NOT EXISTS.
- Могут применяться в инструкциях SELECT, INSERT, UPDATE, DELETE.
- Могут создавать результирующие наборы, которые можно соединять в запросе с другими таблицами и подзапросами.
- Могут использоваться для генерации значений для заполнения таблицы или столбцов в результирующем наборе запроса.
- Используются в предложениях SELECT, FROM, WHERE, HAVING и ORDER BY.