ALTER TABLE

ALTER TABLE

- Что делать если в таблицу надо добавить/удалить столбец после того как таблица уже создана и заполнена данными?
- Или указать, что какое-то поле - внешний ключ? Или что оно должно быть уникальным?
- В этом случае нельзя просто удалить и создать таблицу заново, т.к. будет потеря данных
- Чтобы изменить существующую таблицу используется команда ALTER TABLE
- Аналогично есть ALTER DATABASE, чтобы изменить существующую БД и т.д.
- https://www.w3schools.com/sql/sql_alter.asp
- https://dev.mysql.com/doc/refman/5.7/en/alter-table.html

ALTER TABLE

- **Добавление столбца**:
- ALTER TABLE product
  ADD count INT; # описание столбца, как в CREATE TABLE

- **Удаление столбца:**
- ALTER TABLE product
  DROP count; # достаточно указать имя столбца

ALTER TABLE

- **Добавление ограничения про внешний ключ**:
- ALTER TABLE product
  ADD FOREIGN KEY (supplierId) REFERENCES suppliers(id);

- **Удаление ограничения про внешний ключ:**
- ALTER TABLE product
  DROP FOREIGN KEY fk_name;
  # fk_name – это имя ограничения на внешний ключ.
  # Оно было дано автоматически при добавлении
  # внешнего ключа, и надо подставить туда правильное имя

ALTER TABLE

- Также есть возможность добавлять/удалять другие ограничения – UNIQUE, PRIMARY KEY, DEFAULT
- В некоторых СУБД есть возможность переименовывать столбец и/или менять тип столбца

JOIN таблицы с собой

JOIN таблицы с собой

- JOIN можно применять и чтобы соединять таблицу с самой собой. Это самый обычный JOIN
- Это может быть полезно когда в таблице есть внешний ключ, который ссылается на эту же таблицу
- ```
  CREATE TABLE employee
  (
      id INT AUTO_INCREMENT PRIMARY KEY,
      name VARCHAR(255),
      salary INT NOT NULL,
      chief_id INT NULL,
      FOREIGN KEY (chief_id) REFERENCES employee(id)
      # внешний ключ на саму же таблицу
  );
  ```
  ```

JOIN таблицы с собой

- Это позволяет решать такие задачи:
- Например, найти сотрудников, у которых зарплата больше, чем у руководителя и др.
- ```
  SELECT e.name, e.salary, chief.name AS  chiefName
  FROM employee AS e
  INNER JOIN employee AS chief
      ON e.chief_id = chief.id
  WHERE e.salary > chief.salary;
  ```

- В ON мы соединили сотрудника со своим руководителем, и дальше можем использовать это при фильтрации и при перечислении столбцов в SELECT
  ```

EXISTS, NOT EXISTS.
UNION.
ANY, ALL

EXISTS / NOT EXISTS

- В качестве условия можно использовать оператор EXISTS, в который можно передать подзапрос
- Условие будет истинно, если подзапрос выдаст хотя бы одну строку
- ```
  SELECT co.name
  FROM country AS co
  WHERE EXISTS (SELECT * FROM city
                WHERE countryCode = co.code
                  AND population >= 2000000);
  ```

- NOT EXISTS – обратный оператор к EXISTS
- Условие будет истинно, если подзапрос выдал 0 строк
  ```

UNION

- Иногда данные нужно собрать в единый набор по нескольким разным таблицам
- Например, у нас есть таблица покупателей и таблица поставщиков
- В обеих есть поле city с именем города и поле country с названием страны
- И нам понадобился перечень всех городов и стран из обеих таблиц, хочется получить его одним запросом
- https://www.w3schools.com/sql/sql_union.asp

UNION

- Таблица покупателей

id	customerName	city	country
1	Иван Иванов	Москва	Россия
2	Петр Петров	Берлин	Германия

- Таблица поставщиков

id	supplierName	city	country
1	Intel	Нью-Йорк	США
2	Бирюса	Москва	Россия

UNION

- Перечень всех городов из обеих таблиц:
- SELECT city, country FROM customers
  UNION
  SELECT city, country FROM suppliers
  ORDER BY city;
- Важный момент – UNION откидывает полностью совпадающие строки
- Здесь в результат Москва не попала дважды

city	country
Берлин	Германия
Нью-Йорк	США
Москва	Россия

UNION ALL

- Если нужны дубликаты, можно использовать UNION ALL:
- SELECT city, country FROM customers
  UNION ALL
  SELECT city, country FROM suppliers
  ORDER BY city;

city	country
Берлин	Германия
Нью-Йорк	США
Москва	Россия
Москва	Россия

ANY и ALL

- https://www.w3schools.com/sql/sql_any_all.asp

CASE

CASE

- Иногда при выводе результатов хочется добавить условную логику – в зависимости от условия выводить разные значения
- Например, у нас есть таблица городов, и хочется распределить их на 3 категории, в зависимости от численности:
  - До 100000 – малые города
  - 100000-1000000 – средние города
  - 1000000 и более – города-миллионники
- В этом может помочь оператор CASE
- https://msdn.microsoft.com/ru-ru/library/ms181765(v=sql.120).aspx

CASE

- ```
  SELECT name,
     CASE
        WHEN population < 100000 THEN ‘Малый’
        WHEN population >= 100000 AND population < 1000000
           THEN ‘Средний’
        WHEN population >= 1000000 THEN ‘Миллионник’
     END AS Category
  FROM city;
  ```
- Здесь внутри CASE мы можем написать сколько угодно веток WHEN THEN. Если условие будет TRUE, результатом будет значение из THEN
- В качестве условий могут быть любые логические выражения
- Может быть необязательная ветка ELSE
  ```

Простая версия CASE

- Есть простая версия CASE, которая просто делает проверку поля на равенство
-
```
SELECT requestId,
    CASE status
        WHEN 0 THEN ‘Новая’
        WHEN 1 THEN ‘В работе’
        ELSE ‘Неизвестный статус’
    END AS statusName
FROM requests;
```

Хитрое использование CASE

- Есть еще 2 сценария, когда CASE очень полезен:
  - Использование при сортировке
  - Использование внутри агрегатных функций (чаще всего COUNT) при группировке

Использование CASE при сортировке

- Допустим, вам надо отсортировать по столбцу, допускающему NULL
- ```
  SELECT * FROM country
  ORDER BY IndepYear;
  ```
- По умолчанию строки со значением NULL будут вверху
- Но что если по требованиям надо чтобы строки с NULL были внизу?
- ```
  SELECT * FROM country
  ORDER BY CASE
      WHEN IndepYear IS NULL THEN 1
      ELSE 0
  END;
  ```

Использование CASE при группировке

- У вас есть таблица заявок клиентов

id	status	client
1	Новая	Иван Иванов
2	Выполнена	Иван Иванов
3	Новая	Петр Петров
4	Выполнена	Семен Семенов

- Требуется для каждого клиента вывести количество новых заявок, количество выполненных заявок и общее количество заявок
- Воспользуемся группировкой и COUNT вместе с CASE

Использование CASE при группировке

id	status	client
1	Новая	Иван Иванов
2	Выполнена	Иван Иванов
3	Новая	Петр Петров
4	Выполнена	Семен Семенов

- ```
  SELECT client,
    COUNT(CASE WHEN status = 'Новая' THEN 1 END) AS NewCount,
    COUNT(CASE WHEN status = 'Выполнена' THEN 1 END) AS
  DoneCount,
    COUNT(*) AS TotalCount
  FROM requests
  GROUP BY client;
  ```

client	NewCount	DoneCount	TotalCount
Иван Иванов	1	1	2
Петр Петров	1	0	1
Семен Семенов	0	1	1
  ```

Использование CASE при группировке

- SELECT client,
  COUNT(CASE WHEN status = ‘Новая’ THEN 1 END) AS NewCount,
  COUNT(CASE WHEN status = ‘Выполнена’ THEN 1 END) AS DoneCount,
  COUNT(*) AS TotalCount
  FROM requests
  GROUP BY client;

Это работает и в SUM, AVG и др.

- Если все случаи в CASE не подошли, и нет ELSE, выдается NULL
- Помним, что COUNT выдает количество не NULL значений. Поэтому мы просто занулили ненужные заявки

client	NewCount	DoneCount	TotalCount
Иван Иванов	1	1	2
Петр Петров	1	0	1
Семен Семенов	0	1	1