Проектирование БД

💡 На этой дополнительной лекции мы разберем структуру для работы на 4 - 5 семинаре. Занятие не сильно сложное, но важное: верно спроектированная БД упрощает нам жизнь. Важно понимание структуры и сегодня мы узнаем про связи между таблицами, спроектируем структуру нашей первой БД

Терминология

Сущность = **таблица**.

Сущность - любой однозначно идентифицируемый конкретный или абстрактный объект, включая события и связи между объектами, информация о котором хранится и обрабатывается в базе данных (БД)

Атрибут = имя столбца

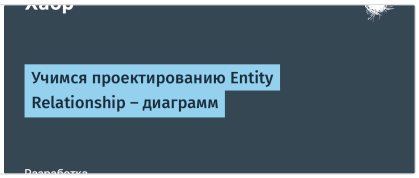
Атрибут базы данных – это имя столбца и содержимое полей под ним в таблице в базе данных.

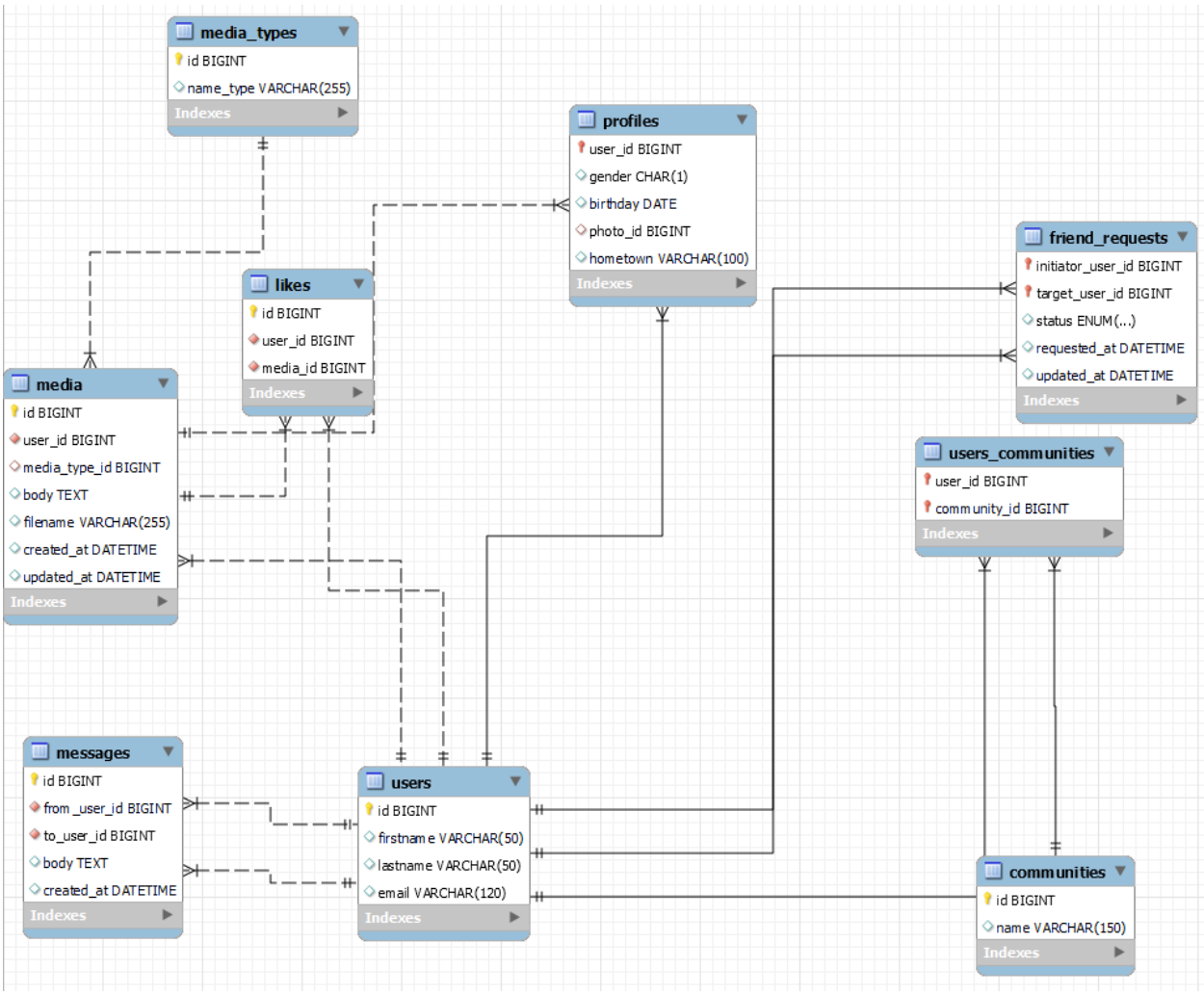
Сегодня мы с вами спроектируем вот такую БД и получим такую диаграмму:

Учимся проектированию Entity Relationship — диаграмм

Здравствуйте. Данная статья посвящена одной из самых популярных, а также и многим знакомой, модели проектирования — ER(Entity Relationship), которая была предложена учёным, в области информатики —...

https://habr.com/ru/post/440556/





Предупреждение

Весь код будет выложен в самом конце. Не всегда запуск будет сразу возможен: возможно, не будет таблицы, на которую ссылается внешний ключ. Обращайте внимание на это.

DDL - язык определения данных

В этой группе операторов мы с вами используем 3 оператора: **CREATE**, **ALTER**, **DROP.**

Итак, начнем с создания БД:

```
CREATE DATABASE IF NOT EXISTS vk;
```

Если такая БД имеется, то повторное создание не произойдет. В случае наличия такой же БД, лучше ее удалять:

```
DROP DATABASE IF EXISTS vk;
```

Далее - выбираем БД для работы:

```
USE  vk;
```

Таблицы

Пользователи

Начнем с создания первой таблицы - `user`. Нашей социальной сетью пользуются конкретные пользователи:

```
-- пользователи
DROP TABLE IF EXISTS user;
CREATE TABLE users
(
    id SERIAL PRIMARY KEY, -- SERIAL = BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
    firstname VARCHAR(50),
    lastname VARCHAR(50) COMMENT 'Фамилия',
    email VARCHAR(120) UNIQUE
);

INSERT INTO user (id, firstname, lastname, email)
VALUES
  (1, 'Reuben', 'Nienow', 'arlo50@example.org'),
  (2, 'Frederik', 'Upton', 'terrence.cartwright@example.org'),
  (3, 'Unique', 'Windler', 'rupert55@example.org'),
  (4, 'Norene', 'West', 'rebekah29@example.net'),
  (5, 'Frederick', 'Effertz', 'von.bridget@example.net'),
  (6, 'Victoria', 'Medhurst', 'sstehr@example.net'),
  (7, 'Austyn', 'Braun', 'itzel.beahan@example.com'),
  (8, 'Jaida', 'Kilback', 'johnathan.wisozk@example.com'),
  (9, 'Mireya', 'Orn', 'missouri87@example.org'),
  (10, 'Jordyn', 'Jerde', 'edach@example.com');
```

`id` - первичный ключ, это может быть большим числом (BIGINT), беззнаковым (UNSIGNED), оно не может быть пустым (NOT NULL), автоинкрементированным (AUTO_INCREMENT), уникальным (UNIQUE). Все это я заменил одним типом: SERIAL.
Справедливо утверждение: `SERIAL = BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE`.

`firstname` - имя человека, представленное строчкой на 50 символов

`lastname` - фамилия человека, представленное строчкой на 50 символов

`email` **-** почта, причем **уникальная,** без повторений (UNIQUE)

Для связи между таблицами мы используем ключ. Какие же бывают связи?

Связи

Связи между таблицами бывают следующих типов:

- Один к одному (One to one)

Данный тип связей встречает не часто. В этом случае объекту одной сущности можно сопоставить только один объект другой сущности. Например, на некоторых сайтах пользователь может иметь только один блог. То есть возникает отношение один пользователь - один блог.

- Один к многим (One to many)

Это наиболее часто встречаемый тип связей. В этом типе связей несколько строк из дочерний таблицы зависят от одной строки в родительской таблице. Например, в одном блоге может быть несколько статей. В этом случае таблица блогов является родительской, а таблица статей - дочерней. То есть один блог - много статей. Или другой пример, в футбольной

команде может играть несколько футболистов. И в то же время один футболист одновременно может играть только в одной команде. То есть одна команда - много футболистов.

- Многие ко многим (Many to many)

При этом типе связей одна строка из таблицы A может быть связана с множеством строк из таблицы B. В свою очередь одна строка из таблицы B может быть связана с множеством строк из таблицы A. Типичный пример - студенты и курсы: один студент может посещать несколько курсов, и соответственно на один курс могут записаться несколько студентов.

Профили пользователей

После столбца можно указать комментарий: `COMMENT 'Фамилия'` **дает комментарий к столбцу** `lastname`**.** Каждый пользователь ведет свой профиль в социальной сети. Давайте попробуем создать таблиц у с информацией о профиле человека - `profile`:

```
-- профиль пользователя
DROP TABLE IF EXISTS `profile`;
CREATE TABLE `profiles` (
    user_id SERIAL PRIMARY KEY,
    gender CHAR(1),
    birthday DATE,
    photo_id BIGINT UNSIGNED,
    hometown VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES users(id)  ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (photo_id) REFERENCES media(id)  ON UPDATE CASCADE ON DELETE SET NULL
);

INSERT INTO `profile` (user_id, gender, birthday, photo_id, hometown)
VALUES
  (1, 'f', '1976-11-08', 9, 'Adriannaport'),
  (2, 'f', '2011-12-20', 13, 'North Nettieton'),
  (3, 'm', '1994-06-15', 3, 'Padbergtown'),
  (4, 'f', '1979-11-07', NULL, 'Port Rupertville'),
  (5, 'f', '1976-04-19', 5, 'Spencerfort'),
  (6, 'f', '1983-09-07', 6, 'South Woodrowmouth'),
  (7, 'm', '2014-07-31', NULL, 'South Jeffereyshire'),
  (8, 'f', '1975-03-26', 17, 'Howeside'),
  (9, 'f', '1982-11-01', 9, 'Loweborough'),
  (10, 'm', '1977-05-14', NULL, 'New Nellaside');
```

Как правило, имена столбцов, которые будут ключами, подчиняются структуре:

`id` - только первичный ключ.

`name_id` - внешний ключ на конкретную таблицу. Вместо параметра `name` идет таблицы, на которую мы ссылаемся. Допустим, `user_id` из таблицы `profile` - внешний ключ на столбец `id` из таблицы `profile` . Связь идет от внешнего ключа к первичному.

1-1

Обратные кавычки (` `) должны использоваться для идентификаторов таблиц и столбцов, но необходимы только в том случае, если идентификатор является зарезервированным ключевым словом MySQL или когда идентификатор содержит пробельные символы.

`gender` - пол (”М” или “Ж”).

`birthday` - дата рождения.

`hometown` - город рождения пользователя.

`user_id` - это и первичный ключ, и внешний ключ. Это гарантирует связь 1 - 1.

1 пользователь имеет 1 профиль 🙂 `ON UPDATE CASCADE ON DELETE CASCADE` - если удаляем пользователя, то удаляется и его профиль. То же самое происходит с обновлением.

`photo_id` - это у нас заготовка для внешнего ключа. Создать прямо сейчас ссылку на таблицу `photo` мы не сможем. Нельзя добавлять внешние ключи на таблицы, которые не существуют.

1-М и следующая таблица - сообщения

Все пользователи общаются друг с другом. Можно ввести дополнительно таблицу `message:`

```
-- сообщения
DROP TABLE IF EXISTS message;
CREATE TABLE messages (
    id SERIAL PRIMARY KEY,
    from_user_id BIGINT UNSIGNED NOT NULL,
    to_user_id BIGINT UNSIGNED NOT NULL,
    body TEXT,
    created_at DATETIME DEFAULT NOW(),
    FOREIGN KEY (from_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (to_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO messages  (from_user_id, to_user_id, body, created_at)
VALUES
(1, 2, 'Voluptatem ut quaerat quia. Pariatur esse amet ratione qui quia. In necessitatibus reprehenderit et. Nam accusantium aut qui quae n
(2, 1, 'Sint dolores et debitis est ducimus. Aut et quia beatae minus. Ipsa rerum totam modi sunt sed. Voluptas atque eum et odio ea molest
(3, 1, 'Sed mollitia quo sequi nisi est tenetur at rerum. Sed quibusdam illo ea facilis nemo sequi. Et tempora repudiandae saepe quo.',  DA
(4, 1, 'Quod dicta omnis placeat id et officiis et. Beatae enim aut aliquid neque occaecati odit. Facere eum distinctio assumenda omnis est
(1, 5, 'Voluptas omnis enim quia porro debitis facilis eaque ut. Id inventore non corrupti doloremque consequuntur. Molestiae molestiae del
(1, 6, 'Rerum labore culpa et laboriosam eum totam. Quidem pariatur sit alias. Atque doloribus ratione eum rem dolor vitae saepe.',  DATE_A
(1, 7, 'Perspiciatis temporibus doloribus debitis. Et inventore labore eos modi. Quo temporibus corporis minus. Accusamus aspernatur nihil
(8, 1, 'Suscipit dolore voluptas et sit vero et sint. Rem ut ratione voluptatum assumenda nesciunt ea. Quas qui qui atque ut. Similique et
(9, 3, 'Et quia libero aut vitae minus. Rerum a blanditiis debitis sit nam. Veniam quasi aut autem ratione dolorem. Sunt quo similique dolo
(10, 2, 'Praesentium molestias quia aut odio. Est quis eius ut animi optio molestiae. Amet tempore sequi blanditiis in est.',  DATE_ADD(NOW
(8, 3, 'Molestiae laudantium quibusdam porro est alias placeat assumenda. Ut consequatur rerum officiis exercitationem eveniet. Qui eum max
(8, 1, 'Quo asperiores et id veritatis placeat. Aperiam ut sit exercitationem iste vel nisi fugit quia. Suscipit labore error ducimus quaer
(8, 1, 'Earum sunt quia sed harum modi accusamus. Quia dolor laboriosam asperiores aliquam quia. Sint id quasi et cumque qui minima ut quo.
(4, 1, 'Aut enim sint voluptas saepe. Ut tenetur quos rem earum sint inventore fugiat. Eaque recusandae similique earum laborum.',  DATE_AD
(4, 1, 'Nisi rerum officiis officiis aut ad voluptates autem. Dolor nesciunt eum qui eos dignissimos culpa iste. Atque qui vitae quos odit
(4, 1, 'Consequatur ut et repellat non voluptatem nihil veritatis. Vel deleniti omnis et consequuntur. Et doloribus reprehenderit sed earum
(2, 1, 'Iste deserunt in et et. Corrupti rerum a veritatis harum. Ratione consequatur est ut deserunt dolores.',  DATE_ADD(NOW(), INTERVAL
(8, 1, 'Dicta non inventore autem incidunt accusamus amet distinctio. Aut laborum nam ab maxime. Maxime minima blanditiis et neque. Et labo
(8, 1, 'Amet ad dolorum distinctio excepturi possimus quia. Adipisci veniam porro ipsum ipsum tempora est blanditiis. Magni ut quia eius qu
(8, 1, 'Porro aperiam voluptate quo eos nobis. Qui blanditiis cum id eos. Est sit reprehenderit consequatur eum corporis. Molestias quia qu
```

`from_user_id` - ссылка на таблицу с пользователями `users`. Отправитель сообщения

`to_user_id` - получатель. Он так же проживает в таблице `users`.

`from_user_id` , `to_user_id` не могут быть типом данных `SERIAL` .

`SERIAL` - это и **AUTO_INCREMENT, а автоинкремент встречается только 1 раз в таблице.** Чтобы решить эту проблему, вспомним формулу:

SERIAL = BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE

Берем из нее нужную часть по свойствам - `BIGINT UNSIGNED NOT NULL`

`body` - текст сообщения, он у нас может быть огромным, поэтому, применяется тип `TEXT`

`created_at` - дата отправления сообщения.

1 пользователь может быть несколько раз отправителем или получателем. Так же мы могли и сделать и с профилями: 1 человек может иметь несколько профилей (те же “фейки”).

Пользователи есть, они обмениваются сообщениями, между ними есть связь - это друзья.

Запросы на дружбу

```
-- заявки на дружбу
DROP TABLE IF EXISTS friend_requests;
CREATE TABLE friend_requests (
    initiator_user_id BIGINT UNSIGNED NOT NULL, -- инициатор
    target_user_id BIGINT UNSIGNED NOT NULL, -- и получатель
    `status` ENUM('requested', 'approved', 'unfriended', 'declined'),
    requested_at DATETIME DEFAULT NOW(),
    updated_at DATETIME,
    PRIMARY KEY (initiator_user_id, target_user_id),
    FOREIGN KEY (initiator_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (target_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
INSERT INTO friend_requests (initiator_user_id, target_user_id, `status`, requested_at, updated_at)
VALUES
(1, 10, 'approved', '2023-01-05 06:40:37', '2023-01-05 16:28:19'),
(1, 2, 'requested', '2023-01-06 07:33:23', NULL),
(1, 3, 'approved', '2023-01-07 01:53:07', '2023-01-18 16:22:56'),
(4, 1, 'approved', '2023-01-08 15:57:26', '2023-01-15 18:12:00'),
(5, 2, 'approved', '2023-01-08 18:22:00', '2023-01-14 08:25:00'),
(6, 3, 'unfriended', '2023-01-09 17:07:59', '2023-01-09 17:12:45'),
(7, 1, 'requested', '2023-01-09 06:20:23', NULL),
(8, 6, 'unfriended', '2023-01-10 01:50:03', '2023-01-10 06:50:59'),
(9, 7, 'approved', '2023-01-11 22:52:09', NULL),
(10, 6, 'approved', '2023-01-12 00:32:15', '2023-01-12 10:22:15');
```

`status` обернули в кавычки обратные, так как это зарезервированное слово. Этот столбец хранится в виде списка возможных значений.

Мы не добавляли с вами `id`. Будем использовать составной первичный ключ, состоящий из 2 столбцов. Дело в том, что при добавлении `id` у нас будет много ненужной информации:

id	initiator_user_id	target_user_id	status
1	1	2	requested
2	1	2	requested

Инициатором дружбы является человек под номером (initiator_user_id = 1), допустим, я отправил запрос на дружбу своему знакомому Антону (target = 2).

И случайно попытался добавить ту же заявку, забыв, что запрос уже отправился. Антон видит 2 заявки: первую он принимает, а вторую отклоняет:

id	initiator_user_id	target_user_id	status
1	1	2	approved
2	1	2	declined

Вопрос: кто же мы с Антоном в итоге? Можно посмотреть по времени принятия или отказа от заявки.

Избавимся-ка мы от `id` . Введем с вами первичный ключ, причем составной - состоящий из нескольких столбцов.

```
PRIMARY KEY (initiator_user_id, target_user_id)
```

Что нам это дает? При попытке сделать повторную отправку заявки мы будем получать ошибку, так как ключ - уникальный объект, заявка будет только 1😄

Но! Заявка, отправленная Антоном на дружбу мне(1,2) и моя заявка на дружбу(2,1)*- это **разные** вещи. Да, необычно, но с этим можно будет бороться. Для этого мы сможем с вами ввести триггер, который будет искать “зеркальные” заявки.

```
-- 1, 2  <> 2, 1
```

initiator_user_id	target_user_id	status
1	2	approved
2	1	declined

- в круглых скобках указаны id инициатора и получателя заявки

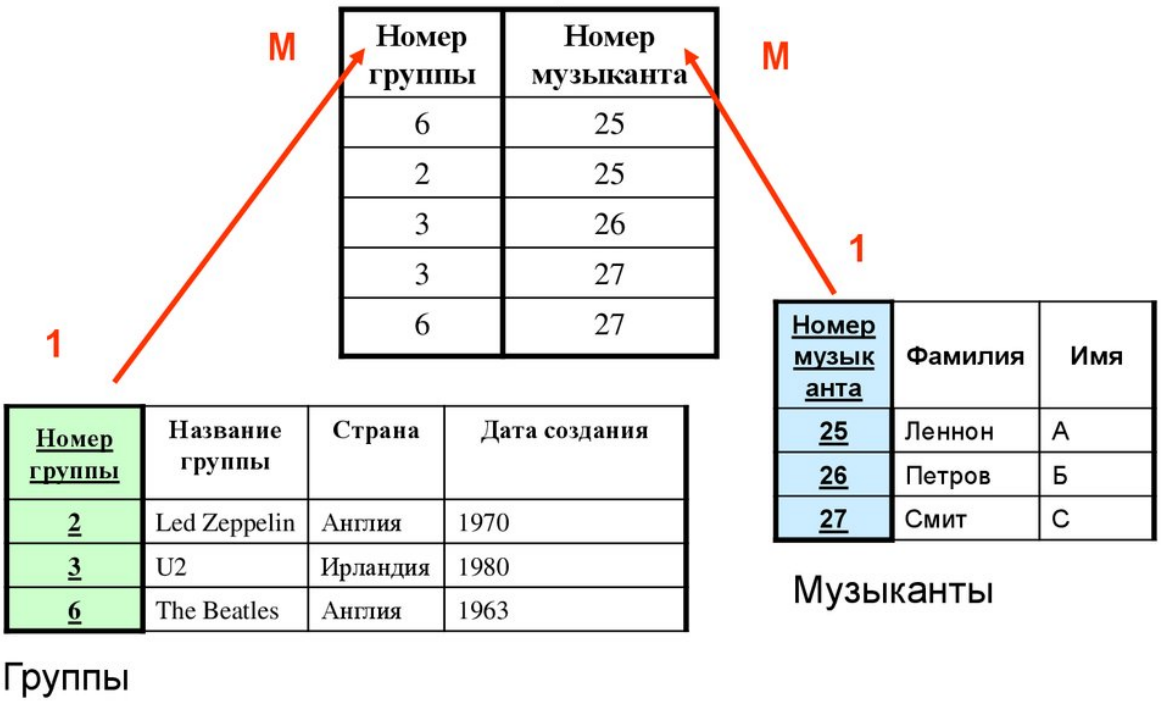
Наши юзеры состоят в сообществах. Имеет смысл создать таблицу под **сообщества**

Связь M-M. Сообщества

Связь **M-M** организована через промежуточную табличку. Пример:

Связь *многие-ко-многим (М –М)*

Таблица 3



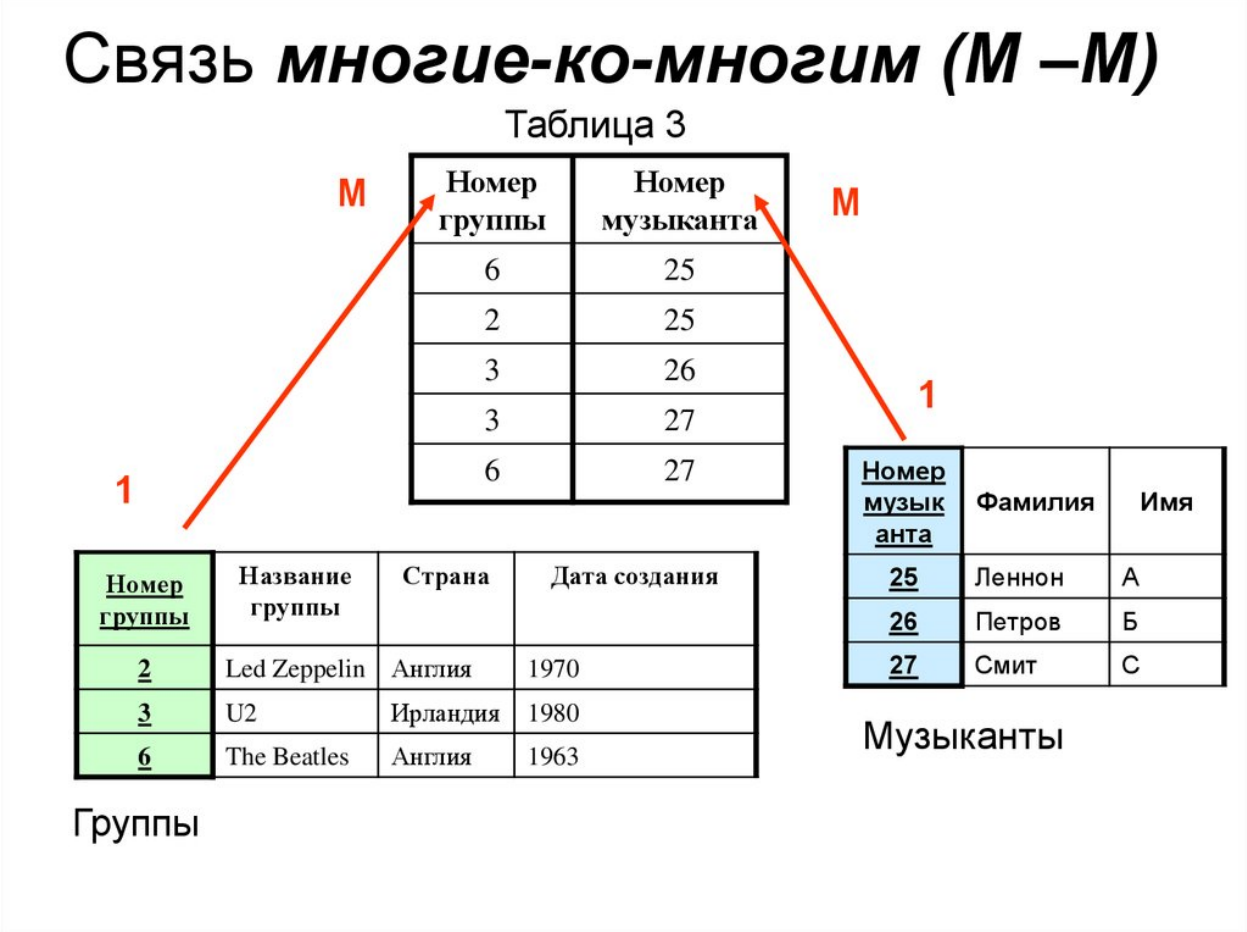
Номер группы	Номер музыканта
6	25
2	25
3	26
3	27
6	27

Номер группы	Название группы	Страна	Дата создания
2	Led Zeppelin	Англия	1970
3	U2	Ирландия	1980
6	The Beatles	Англия	1963

Группы

Номер музыканта	Фамилия	Имя
25	Леннон	А
26	Петров	Б
27	Смит	С

Музыканты



Установим тип связи М-М у пользователей и сообществ, которыми они пользуются (в 1 сообществе состоит несколько пользователей и 1 пользователь может состоять во многих сообществах). Таблица сообщества имеет вид:

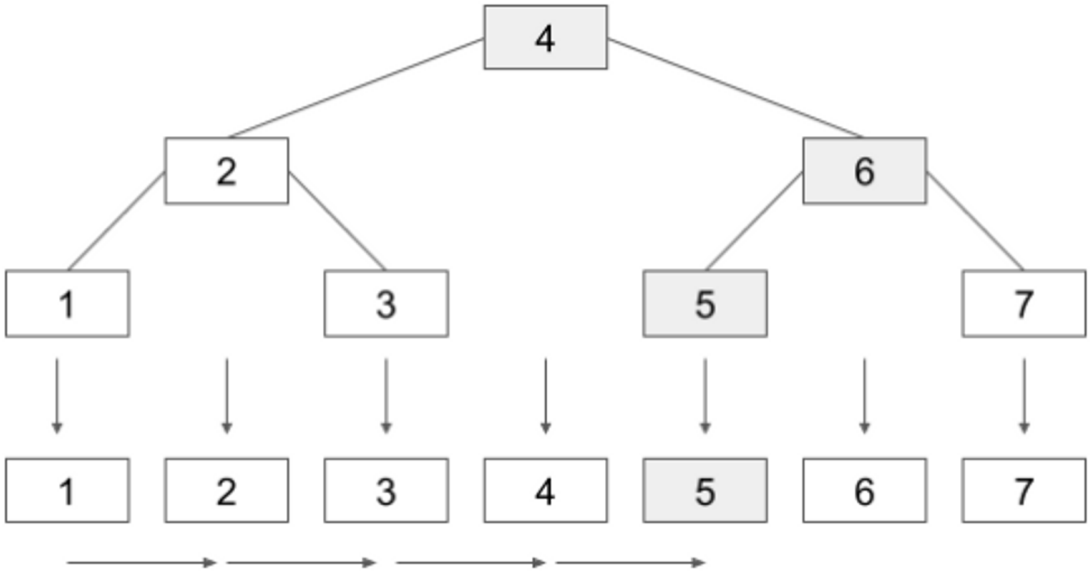
```
DROP TABLE IF EXISTS communities;
CREATE TABLE communities
(
    id SERIAL PRIMARY KEY,
    name VARCHAR(150),
    INDEX communities_name_idx(name)
);
INSERT INTO `communities` (name)
VALUES ('atque'), ('beatae'), ('est'), ('eum'), ('hic'), ('nemo'), ('quis'), ('rerum'), ('tempora'), ('voluptas');
```

INDEX = **Индексы**:

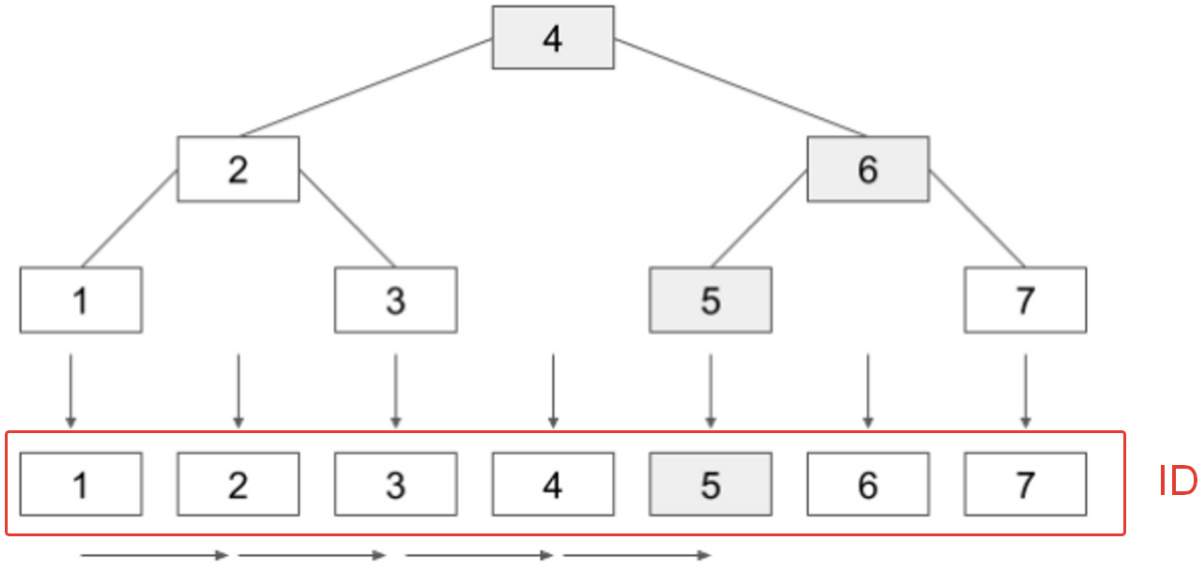
👨‍🎓 Индексы и ключи

Индексы ускоряют работу нашего кода :P

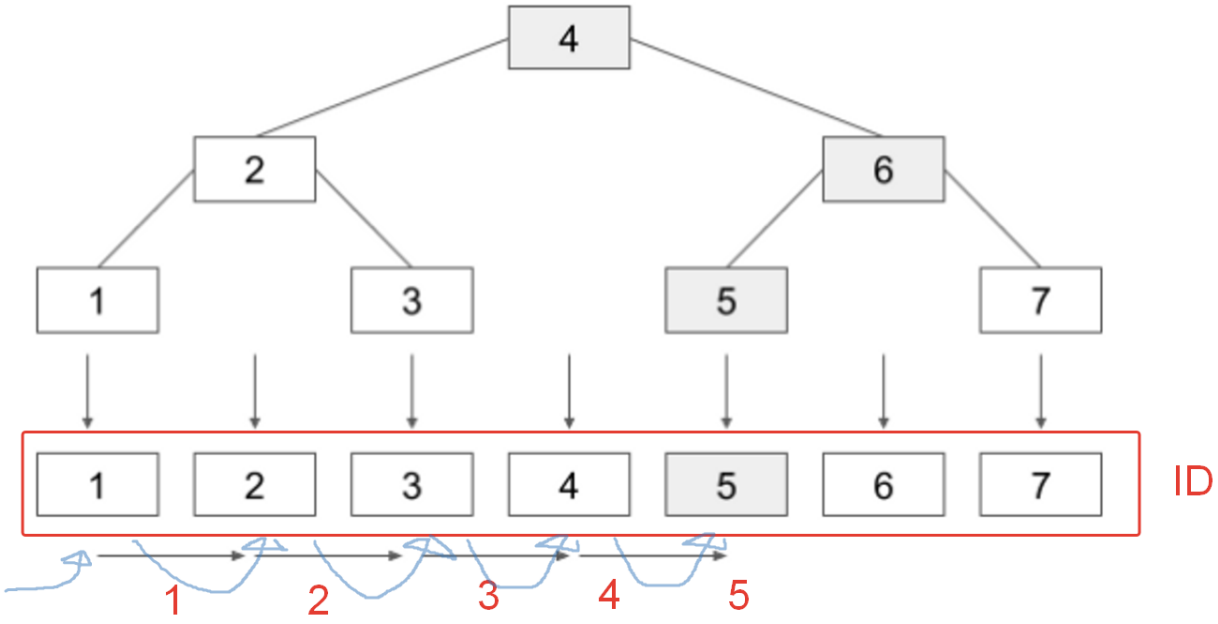
Рассмотрим кратко суть нашей оптимизации с помощью индексов с помощью бинарного дерева:



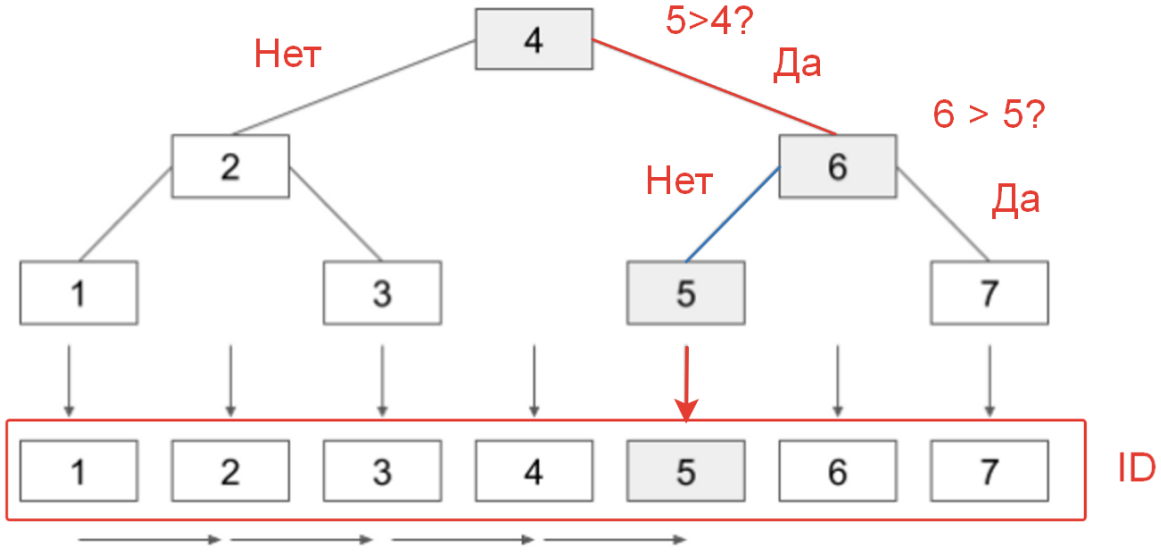
Страшновато, но бояться не нужно. Самая нижняя строчка - это id записи, которую нам нужно найти. Предположим, мы ищем запись id = 5:



Пусть нижняя строчка не имеет индекса. Чтобы найти id = 5, мы обошли все id от 1 до 5. Потребовалось **5 итераций**. А если потребуется id = 100, мы сделаем целых 100 итераций.



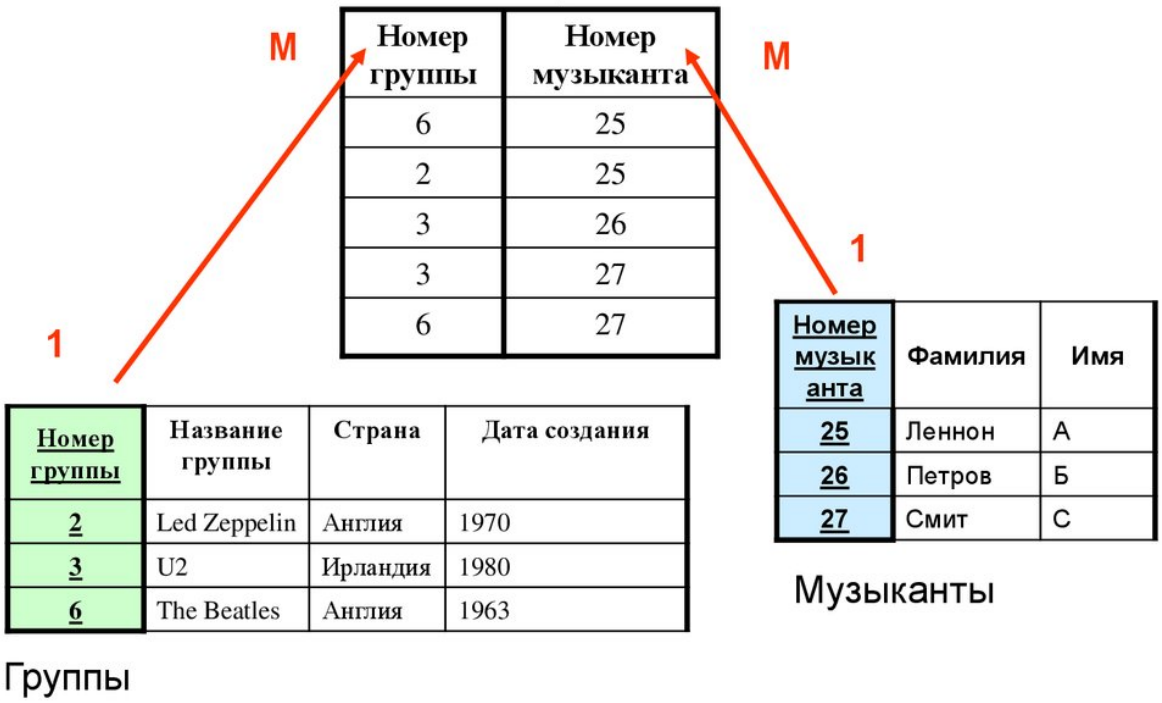
Все, что находится выше красной строчки - индексы. Мы с вами наблюдаем так называемое “дерево индексов”.
Как происходит поиск по **индексам** id = 5?



Как видите, к id = 5 мы подошли гораздо быстрее 😊 Индексы позволяют быстрее перемещаться по данным за счет деления отрезка на части.
Попробуем связать пользователей и сообщества. Для этого вводится дополнительная таблица, имя которой представляет название двух таблиц, связанных друг с другом. Получается, что мы делим связь М-М на две связи: табличка **сообщества связана с дополнительной таблицей через связь 1-М и пользователи связаны с этой таблицей связью 1-М:**

Связь *многие-ко-многим (М –М)*

Таблица 3



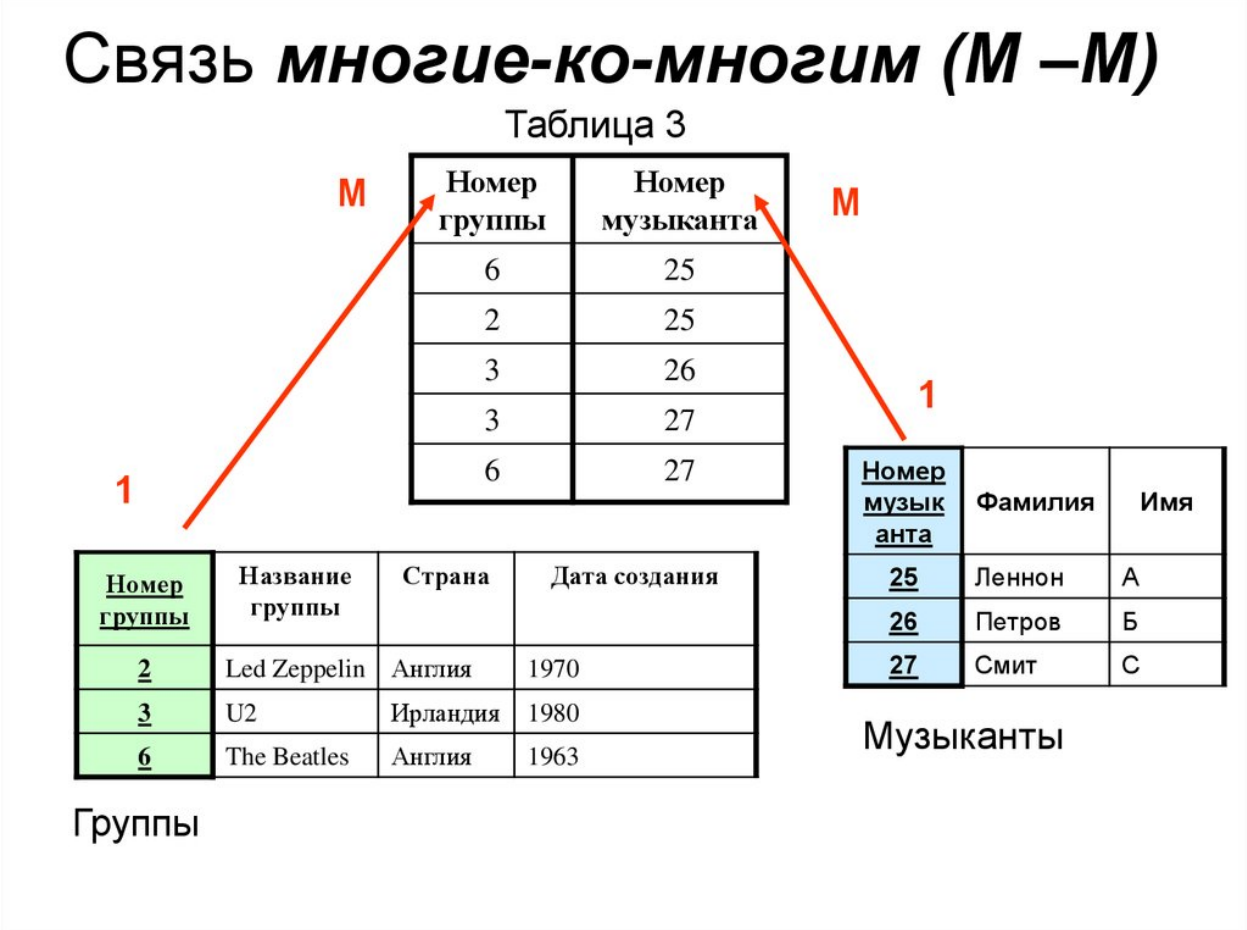
Номер группы	Номер музыканта
6	25
2	25
3	26
3	27
6	27

Номер группы	Название группы	Страна	Дата создания
2	Led Zeppelin	Англия	1970
3	U2	Ирландия	1980
6	The Beatles	Англия	1963

Группы

Номер музыканта	Фамилия	Имя
25	Леннон	А
26	Петров	Б
27	Смит	С

Музыканты




```
-- пользователи сообщества
DROP TABLE IF EXISTS users_communities;
CREATE TABLE users_communities
(
    user_id BIGINT UNSIGNED NOT NULL,
    community_id BIGINT UNSIGNED NOT NULL,
    PRIMARY KEY (user_id, community_id),
    -- чтобы не было 2 записей о пользователе и сообществе
    FOREIGN KEY (user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (community_id) REFERENCES communities(id) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO users_communities (user_id, community_id)
VALUES
(1, 1), (1, 2), (1, 3),
(2, 1), (2, 2),
(3, 1), (3, 2), (3, 10), (3, 5), (3, 8),
(4, 1), (4, 2), (4, 3), (4, 8),
(5, 1), (5, 2), (5, 3), (5, 6), (5, 8), (5, 10),
(6, 1), (6, 2), (6, 3), (6, 6),
(7, 1), (7, 2), (7, 3), (7, 8), (7, 7), (7, 6),
(8, 1), (8, 2), (8, 3), (8, 5), (8, 7), (8, 9),
(9, 1), (9, 2),
(10, 1), (10, 10);
```

У нас будет составной первичный ключ, чтобы не было 2 одинаковых пользователей у сообщества 😄

Медиа

Здесь из интересного:

- `user_id` - ссылка на таблицу user, внешний ключ
- `media_type_id` - тип медиа
- `filename` может храниться в разных типах. Если указать тип `BLOB`, то файлы будут храниться вместе с вашей базой. Если же указать путь (абсолютный или относительный), то можно поместить его в тип данных `VARCHAR`. Для маленьких файлов предпочтительнее `BLOB` (допустим, какие-то маленькие иконки, память под БД не такая и большая). Возможно, имеет смысл выделить данные о файле в какое-то поле. Для этого можно будет реализовать столбец `metadata` , тип данных которого - это пара “ключ : значение” или `JSON.` (С версии 5.7 в СУБД MySQL доступен)
- `created_at` и `updated_at` - время создания и изменения объекта
- `FOREIGN KEY(media_type_id) REFERENCES media_types(id) ON UPDATE CASCADE ON DELETE SET NULL` - в данной ситуации мы при удалении оставляем значение `NULL`. Почему? Давайте рассмотрим ситуацию: при удалении типа медиа, не надо удалять все медиа. Удалив тип медиа (фотографии), не нужно удалять все фотки из таблицы `media`

```
-- медиа
DROP TABLE IF EXISTS media;
CREATE TABLE media(
    id SERIAL PRIMARY KEY,
    user_id BIGINT UNSIGNED NOT NULL,
    media_type_id BIGINT UNSIGNED,
    body text,
    filename VARCHAR(255),
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (media_type_id) REFERENCES media_types(id) ON UPDATE CASCADE ON DELETE SET NULL
);

INSERT INTO media (media_type_id, user_id, filename, body) VALUES
(3, 1, 'a.avi', 'Est aut aliquid est. Soluta eveniet doloribus ipsam ut id assumenda quam. Consequuntur et velit aperiam error corporis rem
(2, 2, '1.mp3', 'Laborum dolorem magni unde vero sit hic. Pariatur quia voluptas magnam. Blanditiis dolorem et molestias veniam facere prov
(1, 3, 'sapiente.png', 'A quam ea ullam reiciendis vel et reiciendis. Delectus iure mollitia est. Eum dolores dolores sed officia vitae qui
(3, 4, 'rerum.mp4', 'Velit commodi in veniam occaecati. Tenetur temporibus ullam enim officiis aut illum. In maiores velit nobis soluta mol
(1, 5, 'user5-profile-image.bmp', 'Est eaque est qui et omnis rerum. Expedita porro nesciunt inventore consequatur quos. Inventore aut reru
(1, 6, 'quasi.jpg', 'Atque et nam nulla et aliquid. Vel repellendus aut natus odit quibusdam quis atque et. Laudantium minima velit dolores
(3, 7, 'non.mp4', 'Nihil ut et reprehenderit et. Numquam veniam quis impedit. Voluptas saepe rerum illo vel omnis. Aliquam illum ut quo sae
(4, 8, 'new_01012023.docx', 'Sint quod fugit molestiae dolore repellendus est. Quis corporis necessitatibus commodi placeat temporibus dolo
(1, 9, 'ullam.jpg', 'Optio sed aperiam veniam eum. Rerum placeat soluta iusto perspiciatis quibusdam dolorem eos. Ea pariatur optio est ut.
(2, 10, '2.mp3', 'Facere nostrum facilis aperiam quisquam dolor. Minima omnis est nam.'),
(3, 1, 'non.mp4', 'A omnis ratione non. Et velit sed incidunt corporis ut rerum nemo. Ut pariatur tempora ea incidunt praesentium ut. Eligе
(4, 1, 'new_07012023.html', 'Quo minus harum debitis debitis quis sunt. Dolores suscipit placeat dolorum non voluptate et. Non eos odio ess
(1, 1, '1.jpg', NULL),
(2, 1, 'godt.mp3', 'Nobis est sed blanditiis assumenda incidunt explicabo. Facere rem assumenda odio explicabo ad enim repellat quia. Dolor
(3, 1, 'path1.avi', 'Tempora ad et aspernatur laborum ut dolor et. Exercitationem quaerat corporis placeat et.'),
(4, 1, 'new_07012023.html', 'Accusantium est ea fuga omnis mollitia. Dolores officia et consequatur iste est quo. Ullam laborum qui ut quo
(1, 1, 'map.bmp', 'Quasi itaque atque ut aliquam debitis. Qui consequuntur maiores sit ad perspiciatis quaerat assumenda repudiandae. Neces
(2, 1, '7.acc', 'Deserunt voluptatem quia voluptatem sit. Qui omnis distinctio optio voluptatem veniam atque dolorem. Repellat laboriosam i
(3, 1, 'path2.avi', 'Fugiat consequuntur voluptatem odit omnis. Quia aut voluptate officia rerum. Cumque voluptatem eaque dolorum voluptas
(4, 2, 'new_10012023.html', 'Autem dolore beatae aut corporis fugit ratione. Ex beatae qui at. Est deleniti asperiores temporibus perferend
(1, 8, 'poster.jpg', 'Provident eligendi animi quidem qui ipsum. Accusamus dolor sint est qui magnam. Omnis enim quis dolore magni facilis
(2, 10, 'music_all.mp3', 'Animi ut labore dolore atque consequuntur maxime fugit. Sunt et et facere sint. Recusandae numquam accusamus a. N
(3, 3, 'kino.avi', 'Sint officiis a ipsa quaerat rerum ea totam. Aut perferendis deleniti error ipsa ducimus ipsam. Ipsum a sunt quis place
(4, 9, 'new_117012023.doc', 'Et accusamus a et adipisci dolore. Blanditiis sit vitae dolorem voluptas temporibus numquam ab. Aut temporibus
(1, 10, 'main_photo.jpg', 'Consequuntur animi sed ea perferendis ad magnam. Aut libero alias sequi iste qui est. Ut quo dignissimos quibusd
(2, 1, 'ariya.mp3', 'Accusantium sit praesentium voluptatem molestias architecto. Excepturi doloremque ab eligendi eum ullam voluptas qui.
(3, 1, 'film.mp4', NULL),
(4, 1, 'news.html', 'A culpa nostrum et quis. Id ipsum ipsum deserunt earum eaque aut earum. Blanditiis et commodi voluptas.'),
(1, 3, 'non.jpg', 'Cumque quod provident provident fugit cumque numquam. Fuga totam delectus a aut quaerat nemo dignissimos eos. Ratione ha
(1, 8, 'et.jpg', 'Provident quia fuga et consequatur reprehenderit repellat. Et aut cum nostrum ut beatae animi aut alias.');
```

Медиа - понятие обширное (фото, музыка, посты вашего друга в вк). Чтобы узнать, с чем мы имеем дело, создадим таблицу, где укажем типы медиафайлов.

Тип медиа

```
-- типы медиа
DROP TABLE IF EXISTS media_types;
CREATE TABLE media_types(
```

```
    id SERIAL PRIMARY KEY,
    name_type VARCHAR(255)
);

INSERT INTO media_types (name_type)
VALUES  ('Photo'), ('Music'), ('Video'), ('Post');
```

Лайки или зачем еще выкладывать медиа 🙂

```
-- лайки медиа
DROP TABLE IF EXISTS likes;
CREATE TABLE likes(
    id SERIAL PRIMARY KEY,
    user_id BIGINT UNSIGNED NOT NULL,
    media_id BIGINT UNSIGNED NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)  ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (media_id) REFERENCES media(id)  ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO likes (user_id, media_id) VALUES
(1, 1), (2, 1), (3, 1), (4, 1), (5, 2), (6, 2), (7, 2), (8, 8), (1, 9), (10, 15), (7, 11), (5, 12), (6, 13), (1, 14), (1, 15), (1, 16), (1,
```

Указываем, какой пользователь лайкает и самое главное - какую фотографию.

Весь код:

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/ed99f11a-a5b6-4c6a-8e0b-4b33b70f2af9/vk_db.sql

```
DROP DATABASE IF EXISTS lesson_4;
CREATE DATABASE lesson_4;
USE lesson_4;

-- пользователи
DROP TABLE IF EXISTS users;
CREATE TABLE users (
  id SERIAL PRIMARY KEY, -- SERIAL = BIGINT UNSIGNED NOT NULL AUTO_INCREMENT UNIQUE
    firstname VARCHAR(50),
    lastname VARCHAR(50) COMMENT 'Фамилия',
    email VARCHAR(120) UNIQUE
);

INSERT INTO users (id, firstname, lastname, email) VALUES
(1, 'Reuben', 'Nienow', 'arlo50@example.org'),
(2, 'Frederik', 'Upton', 'terrence.cartwright@example.org'),
(3, 'Unique', 'Windler', 'rupert55@example.org'),
(4, 'Norene', 'West', 'rebekah29@example.net'),
(5, 'Frederick', 'Effertz', 'von.bridget@example.net'),
(6, 'Victoria', 'Medhurst', 'sstehr@example.net'),
(7, 'Austyn', 'Braun', 'itzel.beahan@example.com'),
(8, 'Jaida', 'Kilback', 'johnathan.wisozk@example.com'),
(9, 'Mireya', 'Orn', 'missouri87@example.org'),
(10, 'Jordyn', 'Jerde', 'edach@example.com');

-- сообщения
DROP TABLE IF EXISTS messages;
CREATE TABLE messages (
  id SERIAL PRIMARY KEY,
  from_user_id BIGINT UNSIGNED NOT NULL,
    to_user_id BIGINT UNSIGNED NOT NULL,
    body TEXT,
    created_at DATETIME DEFAULT NOW(),
    FOREIGN KEY (from_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (to_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE
);
```

```
INSERT INTO messages  (from_user_id, to_user_id, body, created_at) VALUES
(1, 2, 'Voluptatem ut quaerat quia. Pariatur esse amet ratione qui quia. In necessitatibus reprehenderit et. Nam accusantium aut qui quae n
(2, 1, 'Sint dolores et debitis est ducimus. Aut et quia beatae minus. Ipsa rerum totam modi sunt sed. Voluptas atque eum et odio ea molest
(3, 1, 'Sed mollitia quo sequi nisi est tenetur at rerum. Sed quibusdam illo ea facilis nemo sequi. Et tempora repudiandae saepe quo.',  DA
(4, 1, 'Quod dicta omnis placeat id et officiis et. Beatae enim aut aliquid neque occaecati odit. Facere eum distinctio assumenda omnis est
(1, 5, 'Voluptas omnis enim quia porro debitis facilis eaque ut. Id inventore non corrupti doloremque consequuntur. Molestiae molestiae del
(1, 6, 'Rerum labore culpa et laboriosam eum totam. Quidem pariatur sit alias. Atque doloribus ratione eum rem dolor vitae saepe.',  DATE_A
(1, 7, 'Perspiciatis temporibus doloribus debitis. Et inventore labore eos modi. Quo temporibus corporis minus. Accusamus aspernatur nihil
(8, 1, 'Suscipit dolore voluptas et sit vero et sint. Rem ut ratione voluptatum assumenda nesciunt ea. Quas qui qui atque ut. Similique et
(9, 3, 'Et quia libero aut vitae minus. Rerum a blanditiis debitis sit nam. Veniam quasi aut autem ratione dolorem. Sunt quo similique dolo
(10, 2, 'Praesentium molestias quia aut odio. Est quis eius ut animi optio molestiae. Amet tempore sequi blanditiis in est.',  DATE_ADD(NOW
(8, 3, 'Molestiae laudantium quibusdam porro est alias placeat assumenda. Ut consequatur rerum officiis exercitationem eveniet. Qui eum max
(8, 1, 'Quo asperiores et id veritatis placeat. Aperiam ut sit exercitationem iste vel nisi fugit quia. Suscipit labore error ducimus quaer
(8, 1, 'Earum sunt quia sed harum modi accusamus. Quia dolor laboriosam asperiores aliquam quia. Sint id quasi et cumque qui minima ut quo.
(4, 1, 'Aut enim sint voluptas saepe. Ut tenetur quos rem earum sint inventore fugiat. Eaque recusandae similique earum laborum.',  DATE_AD
(4, 1, 'Nisi rerum officiis officiis aut ad voluptates autem. Dolor nesciunt eum qui eos dignissimos culpa iste. Atque qui vitae quos odit
(4, 1, 'Consequatur ut et repellat non voluptatem nihil veritatis. Vel deleniti omnis et consequuntur. Et doloribus reprehenderit sed earum
(2, 1, 'Iste deserunt in et et. Corrupti rerum a veritatis harum. Ratione consequatur est ut deserunt dolores.',  DATE_ADD(NOW(), INTERVAL
(8, 1, 'Dicta non inventore autem incidunt accusamus amet distinctio. Aut laborum nam ab maxime. Maxime minima blanditiis et neque. Et labo
(8, 1, 'Amet ad dolorum distinctio excepturi possimus quia. Adipisci veniam porro ipsum ipsum tempora est blanditiis. Magni ut quia eius qu
(8, 1, 'Porro aperiam voluptate quo eos nobis. Qui blanditiis cum id eos. Est sit reprehenderit consequatur eum corporis. Molestias quia qu

-- заявки на дружбу
DROP TABLE IF EXISTS friend_requests;
CREATE TABLE friend_requests (
  initiator_user_id BIGINT UNSIGNED NOT NULL,
    target_user_id BIGINT UNSIGNED NOT NULL,
    `status` ENUM('requested', 'approved', 'unfriended', 'declined'),
  requested_at DATETIME DEFAULT NOW(),
  updated_at DATETIME,
    PRIMARY KEY (initiator_user_id, target_user_id),
    FOREIGN KEY (initiator_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (target_user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO friend_requests (initiator_user_id, target_user_id, `status`, requested_at, updated_at)
VALUES
(1, 10, 'approved', '2023-01-05 06:40:37', '2023-01-05 16:28:19'),
(1, 2, 'requested', '2023-01-06 07:33:23', NULL),
(1, 3, 'approved', '2023-01-07 01:53:07', '2023-01-18 16:22:56'),
(4, 1, 'approved', '2023-01-08 15:57:26', '2023-01-15 18:12:00'),
(5, 2, 'approved', '2023-01-08 18:22:00', '2023-01-14 08:25:00'),
(6, 3, 'unfriended', '2023-01-09 17:07:59', '2023-01-09 17:12:45'),
(7, 1, 'requested', '2023-01-09 06:20:23', NULL),
(8, 6, 'unfriended', '2023-01-10 01:50:03', '2023-01-10 06:50:59'),
(9, 7, 'approved', '2023-01-11 22:52:09', NULL),
(10, 6, 'approved', '2023-01-12 00:32:15', '2023-01-12 10:22:15');

-- сообщества
DROP TABLE IF EXISTS communities;
CREATE TABLE communities(
  id SERIAL PRIMARY KEY,
  name VARCHAR(150),
    INDEX communities_name_idx(name)
);

INSERT INTO `communities` (name)
VALUES ('atque'), ('beatae'), ('est'), ('eum'), ('hic'), ('nemo'), ('quis'), ('rerum'), ('tempora'), ('voluptas');

-- пользователи сообщества
DROP TABLE IF EXISTS users_communities;
CREATE TABLE users_communities(
  user_id BIGINT UNSIGNED NOT NULL,
  community_id BIGINT UNSIGNED NOT NULL,
    PRIMARY KEY (user_id, community_id),
    FOREIGN KEY (user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (community_id) REFERENCES communities(id) ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO users_communities (user_id, community_id)
VALUES
(1, 1), (1, 2), (1, 3),
(2, 1), (2, 2),
(3, 1), (3, 2), (3, 10), (3, 5), (3, 8),
(4, 1), (4, 2), (4, 3), (4, 8),
(5, 1), (5, 2), (5, 3), (5, 6), (5, 8), (5, 10),
(6, 1), (6, 2), (6, 3), (6, 6),
(7, 1), (7, 2), (7, 3), (7, 8), (7, 7), (7, 6),
(8, 1), (8, 2), (8, 3), (8, 5), (8, 7), (8, 9),
(9, 1), (9, 2),
```

```
(10, 1), (10, 10);

-- типы медиа
DROP TABLE IF EXISTS media_types;
CREATE TABLE media_types(
  id SERIAL PRIMARY KEY,
    name_type VARCHAR(255)
);

INSERT INTO media_types (name_type)
VALUES  ('Photo'), ('Music'), ('Video'), ('Post');


-- медиа
DROP TABLE IF EXISTS media;
CREATE TABLE media(
  id SERIAL PRIMARY KEY,
    user_id BIGINT UNSIGNED NOT NULL,
    media_type_id BIGINT UNSIGNED,
    body text,
    filename VARCHAR(255),
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME ON UPDATE CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (media_type_id) REFERENCES media_types(id) ON UPDATE CASCADE ON DELETE SET NULL
);

INSERT INTO media (media_type_id, user_id, filename, body) VALUES
(3, 1, 'a.avi', 'Est aut aliquid est. Soluta eveniet doloribus ipsam ut id assumenda quam. Consequuntur et velit aperiam error corporis rem
(2, 2, '1.mp3', 'Laborum dolorem magni unde vero sit hic. Pariatur quia voluptas magnam. Blanditiis dolorem et molestias veniam facere prov
(1, 3, 'sapiente.png', 'A quam ea ullam reiciendis vel et reiciendis. Delectus iure mollitia est. Eum dolores dolores sed officia vitae qui
(3, 4, 'rerum.mp4', 'Velit commodi in veniam occaecati. Tenetur temporibus ullam enim officiis aut illum. In maiores velit nobis soluta mol
(1, 5, 'user5-profile-image.bmp', 'Est eaque est qui et omnis rerum. Expedita porro nesciunt inventore consequatur quos. Inventore aut reru
(1, 6, 'quasi.jpg', 'Atque et nam nulla et aliquid. Vel repellendus aut natus odit quibusdam quis atque et. Laudantium minima velit dolores
(3, 7, 'non.mp4', 'Nihil ut et reprehenderit et. Numquam veniam quis impedit. Voluptas saepe rerum illo vel omnis. Aliquam illum ut quo sae
(4, 8, 'new_01012023.docx', 'Sint quod fugit molestiae dolore repellendus est. Quis corporis necessitatibus commodi placeat temporibus dolo
(1, 9, 'ullam.jpg', 'Optio sed aperiam veniam eum. Rerum placeat soluta iusto perspiciatis quibusdam dolorem eos. Ea pariatur optio est ut.
(2, 10, '2.mp3', 'Facere nostrum facilis aperiam quisquam dolor. Minima omnis est nam.'),
(3, 1, 'non.mp4', 'A omnis ratione non. Et velit sed incidunt corporis ut rerum nemo. Ut pariatur tempora ea incidunt praesentium ut. Eligе
(4, 1, 'new_07012023.html', 'Quo minus harum debitis debitis quis sunt. Dolores suscipit placeat dolorum non voluptate et. Non eos odio ess
(1, 1, '1.jpg', NULL),
(2, 1, 'godt.mp3', 'Nobis est sed blanditiis assumenda incidunt explicabo. Facere rem assumenda odio explicabo ad enim repellat quia. Dolor
(3, 1, 'path1.avi', 'Tempora ad et aspernatur laborum ut dolor et. Exercitationem quaerat corporis placeat et.'),
(4, 1, 'new_07012023.html', 'Accusantium est ea fuga omnis mollitia. Dolores officia et consequatur iste est quo. Ullam laborum qui ut quo
(1, 1, 'map.bmp', 'Quasi itaque atque ut aliquam debitis. Qui consequuntur maiores sit ad perspiciatis quaerat assumenda repudiandae. Neces
(2, 1, '7.acc', 'Deserunt voluptatem quia voluptatem sit. Qui omnis distinctio optio voluptatem veniam atque dolorem. Repellat laboriosam i
(3, 1, 'path2.avi', 'Fugiat consequuntur voluptatem odit omnis. Quia aut voluptate officia rerum. Cumque voluptatem eaque dolorum voluptas
(4, 2, 'new_10012023.html', 'Autem dolore beatae aut corporis fugit ratione. Ex beatae qui at. Est deleniti asperiores temporibus perferend
(1, 8, 'poster.jpg', 'Provident eligendi animi quidem qui ipsum. Accusamus dolor sint est qui magnam. Omnis enim quis dolore magni facilis
(2, 10, 'music_all.mp3', 'Animi ut labore dolore atque consequuntur maxime fugit. Sunt et et facere sint. Recusandae numquam accusamus a. N
(3, 3, 'kino.avi', 'Sint officiis a ipsa quaerat rerum ea totam. Aut perferendis deleniti error ipsa ducimus ipsam. Ipsum a sunt quis place
(4, 9, 'new_117012023.doc', 'Et accusamus a et adipisci dolore. Blanditiis sit vitae dolorem voluptas temporibus numquam ab. Aut temporibus
(1, 10, 'main_photo.jpg', 'Consequuntur animi sed ea perferendis ad magnam. Aut libero alias sequi iste qui est. Ut quo dignissimos quibusd
(2, 1, 'ariya.mp3', 'Accusantium sit praesentium voluptatem molestias architecto. Excepturi doloremque ab eligendi eum ullam voluptas qui.
(3, 1, 'film.mp4', NULL),
(4, 1, 'news.html', 'A culpa nostrum et quis. Id ipsum ipsum deserunt earum eaque aut earum. Blanditiis et commodi voluptas.'),
(1, 3, 'non.jpg', 'Cumque quod provident provident fugit cumque numquam. Fuga totam delectus a aut quaerat nemo dignissimos eos. Ratione ha
(1, 8, 'et.jpg', 'Provident quia fuga et consequatur reprehenderit repellat. Et aut cum nostrum ut beatae animi aut alias.');

-- лайки медиа
DROP TABLE IF EXISTS likes;
CREATE TABLE likes(
  id SERIAL PRIMARY KEY,
    user_id BIGINT UNSIGNED NOT NULL,
    media_id BIGINT UNSIGNED NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id)  ON UPDATE CASCADE ON DELETE CASCADE,
    FOREIGN KEY (media_id) REFERENCES media(id)  ON UPDATE CASCADE ON DELETE CASCADE
);

INSERT INTO likes (user_id, media_id) VALUES
(1, 1), (2, 1), (3, 1), (4, 1), (5, 2), (6, 2), (7, 2), (8, 8), (1, 9), (10, 15), (7, 11), (5, 12), (6, 13), (1, 14), (1, 15), (1, 16), (1,

-- профиль пользователя
DROP TABLE IF EXISTS `profiles`;
CREATE TABLE `profiles` (
  user_id SERIAL PRIMARY KEY,
    gender CHAR(1),
    birthday DATE,
  photo_id BIGINT UNSIGNED,
    hometown VARCHAR(100),
    FOREIGN KEY (user_id) REFERENCES users(id)  ON UPDATE CASCADE ON DELETE CASCADE,
```

```
    FOREIGN KEY (photo_id) REFERENCES media(id)  ON UPDATE CASCADE ON DELETE SET NULL
);
INSERT INTO `profiles` (user_id, gender, birthday, photo_id, hometown) VALUES
(1, 'f', '1976-11-08', 9, 'Adriannaport'),
(2, 'f', '2011-12-20', 13, 'North Nettieton'),
(3, 'm', '1994-06-15', 3, 'Padbergtown'),
(4, 'f', '1979-11-07', NULL, 'Port Rupertville'),
(5, 'f', '1976-04-19', 5, 'Spencerfort'),
(6, 'f', '1983-09-07', 6, 'South Woodrowmouth'),
(7, 'm', '2014-07-31', NULL, 'South Jeffereyshire'),
(8, 'f', '1975-03-26', 17, 'Howeside'),
(9, 'f', '1982-11-01', 9, 'Loweborough'),
(10, 'm', '1977-05-14', NULL, 'New Nellaside');
```

Используемые источники

1. https://dev.mysql.com/doc/refman/5.7/en/group-by-functions-and-modifiers.html
2. Линн Бейли. Head First. Изучаем SQL. — СПб.: Питер, 2012. — 592 с.
3. Грофф, Джеймс Р., Вайнберг, Пол Н., Оппель, Эндрю Дж. SQL: полное руководство, 3-е изд. : Пер. с англ. — М.: ООО "И.Д. Вильямс", 2015. — 960 с.
4. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. — Пер. с англ. — СПб.: Символ-Плюс, 2010. — 480 с.
5. Кузнецов М.В., Симдянов И.В. MySQL на примерах. — СПб.: БХВ-Петербург, 2007. — 592с.
6. Кузнецов М.В., Симдянов И.В. MySQL 5. — СПб.: БХВ-Петербург, 2006. — 1024с.
7. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом "Вильямс", 2005. — 1328 с.
8. Карвин Б. Программирование баз данных SQL. Типичные ошибки и их устранение. — Рид Групп, 2011. — 336 с.
9. https://dev.mysql.com/doc/refman/5.7/en/union.html.
10. https://dev.mysql.com/doc/refman/5.7/en/subqueries.html
11. https://dev.mysql.com/doc/refman/5.7/en/join.html

Книги:

- “Изучаем SQL”, книга Бейли Л.
- Алан Бьюли "Изучаем SQL" (2007)
- Энтони Молинаро "SQL. Сборник рецептов" (2009)