Дизайн решения

Разработка структуры базы данных с тестовыми данными и примерами SQL-запросов

Подготовлено: Адлер И.А.

Проект: Разработка структуры базы данных для агентства по подбору топ-персонала

Стек: MySQL

Содержание

1. Описание проекта

В качестве проекта я выбрала базу данных для кадрового агентства, которое занимается поиском и подбором персонала на топ-позиции (уровня руководителей среднего и высшего звена).

Компании важно, чтобы БД содержала информацию об истории проектов, контакты кандидатов и сотрудников компаний, историю взаимодействия с кандидатами и компаниями, а также возможность проводить аналитику и оценивать эффективность работы по тому или иному заказу.

Назовем программу, для которой будем разрабатывать структуру БД, - Experium.

2. Таблицы базы данных

Скрипт создания базы данных Experium (***creatingDB_script.sql***) прилагается.

База данных будет включает 16 таблиц:

1. users (кандидаты, сотрудники компаний),
2. users_profiles (профиль человека),

3. companies (компании),
4. companies_profiles (карточка компании),

5. sector (виды отраслей),
6. positions (должности),
7. users_sector (соответствие кандидаты – отрасли),
8. companies_sector (соответствие компании – отрасли),
9. users_companies (список сотрудников компании),

10. documents_type (типы документов),
11. users_documents (документы кандидатов),
12. companies_documents (документы компаний),

13. o_status – типы статуса проекта.
14. orders (заказы от компаний на поиск кандидатов),
15. u_status (статус кандидата в заказе),
16. users_orders (список представленных кандидатов).

3. Описание таблиц

1. Таблица users служит для хранения информации о кандидатах, сотрудниках и контактных лицах компаний.

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    firstname VARCHAR(50),
    lastname VARCHAR(50),
    email VARCHAR(100) UNIQUE,
    phone BIGINT,
    INDEX users_phone_idx(phone),
    INDEX users_email_idx(email),
    INDEX users_firstname_lastname_idx(firstname, lastname)
 ) COMMENT = 'Кандидаты';
```

2. users_profiles – карточка человека, где хранится информация о нем (город проживания, пол, дата рождения и комментарии о человеке).

```
CREATE TABLE users_profiles (
    user_id SERIAL PRIMARY KEY,
    gender CHAR(1),
    birthday DATE,
    hometown VARCHAR(100),
    `comment` text,
    created_at DATETIME DEFAULT NOW(),
    FOREIGN KEY (user_id) references users(id)
 ) COMMENT = 'Карточка человека';
```

3. companies – это таблица с компаниями. Должна содержать как минимум сокращенное название. Возможно также указание полного названия.

```
CREATE TABLE companies (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL, -- обязательно
    fullname VARCHAR(255), -- полное название компании
    INDEX companies_name_idx(name)
 ) COMMENT = 'Компании';
```

4. companies_profiles – в карточке компании содержится более подробная информация (адрес, контакты). Допустимы NULL-значения, т.к. заполнять эту информацию необязательно. По сути, важно только название компании.

```
CREATE TABLE companies_profiles (
    company_id SERIAL PRIMARY KEY,
    email VARCHAR(100),
    phone BIGINT,
    adress VARCHAR(255),
    `comment` text,
    created_at DATETIME DEFAULT NOW(),
```

```
    update_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,
    FOREIGN KEY (company_id) references companies(id)
 ) COMMENT = 'Карточка компании';
```

5. sector – это виды отраслей рынка, например, промышленность, медицина, финансы и так далее.

```
CREATE TABLE sector (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    created_at DATETIME DEFAULT NOW(),
    update_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
 ) COMMENT = 'Отрасль, сектор';
```

6. positions – это должности людей уровня руководителей среднего звена и выше. Массовый подбор линейных сотрудников агентство не осуществляет.

```
CREATE TABLE positions (
    id SERIAL PRIMARY KEY,
    name VARCHAR(100),
    created_at DATETIME DEFAULT NOW(),
    update_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
 ) COMMENT = 'Должность';
```

7. users_sector – таблица, объединяющая людей и отрасли, т.к. каждый человек может быть экспертом в нескольких областях, равно как и в одной отрасли может быть много людей. Связь многие-ко-многим реализуем через дополнительную таблицу.

```
CREATE TABLE users_sector (
    user_id BIGINT UNSIGNED NOT NULL,
    sector_id BIGINT UNSIGNED NOT NULL,
    PRIMARY KEY (user_id, sector_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (sector_id) REFERENCES sector(id)
 ) COMMENT = 'Отрась специализации кандидата';
```

8. companies_sector – по аналогии с предыдущей таблицей соединяем компании и отрасли. Например, компания может находиться в отрасли E-commerce и Retail одновременно. Или медицина и фарма. Снова связь многие-ко-многим.

```
CREATE TABLE companies_sector (
    company_id BIGINT UNSIGNED NOT NULL,
    sector_id BIGINT UNSIGNED NOT NULL,
    PRIMARY KEY (company_id, sector_id),
    FOREIGN KEY (company_id) REFERENCES companies(id),
    FOREIGN KEY (sector_id) REFERENCES sector(id)
 ) COMMENT = 'Отрасль компании';
```

9. users_companies – список сотрудников компании. Состоит из id человека (NOT NULL), id компании (NOT NULL) и id должности (возможно NULL-значение).

```
CREATE TABLE users_companies (
    user_id BIGINT UNSIGNED NOT NULL,
    company_id BIGINT UNSIGNED NOT NULL,
    position_id BIGINT UNSIGNED NULL, -- можем не знать должность

    PRIMARY KEY (user_id, company_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (company_id) REFERENCES companies(id),
    FOREIGN KEY (position_id) REFERENCES positions(id)
 ) COMMENT = 'Список сотрудников';
```

10. documents_type – переходим к необходимости прикрепления документов. В карточке человека можно хранить резюме, файл с рекомендациями, оффер и т.д. К компании же можно прикрепить договор, соглашение, акт или что-то еще. Поэтому нам нужна таблица с названиями типов документов.

```
CREATE TABLE documents_type (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
 ) COMMENT = 'Тип документа';
```

11. users_documents – таблица, связывающая людей и документы, т.к. тут связь многие-ко-многим.

```
CREATE TABLE users_documents (
    user_id BIGINT UNSIGNED NOT NULL,
    documents_type_id BIGINT UNSIGNED NOT NULL,
    filename VARCHAR(255),
    size INT,
    metadata JSON,
    `comment` text,
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX (user_id),
    PRIMARY KEY (user_id, documents_type_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (documents_type_id) REFERENCES documents_type(id)
 ) COMMENT = 'Документы кандидата';
```

12. companies_documents – то же самое, как в предыдущем пункте, только в отношении компаний.

```
CREATE TABLE companies_documents (
    documents_type_id BIGINT UNSIGNED NOT NULL,
    company_id BIGINT UNSIGNED NOT NULL,
    documents_type_id BIGINT UNSIGNED NOT NULL,
    filename VARCHAR(255),
    size INT,
    metadata JSON,
    `comment` text,
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX (company_id),
    PRIMARY KEY (company_id, documents_type_id),
    FOREIGN KEY (company_id) REFERENCES companies(id),
    FOREIGN KEY (documents_type_id) REFERENCES documents_type(id)
 ) COMMENT = 'Документы компании';
```

13. order_status – перечисление статусов, в котором может находиться проект. Например, «новый проект», который только получило агентство и еще не запустило в работу. «В работе» - в проекте начинают появляться первые кандидаты. Статус «завершен» должен автоматически присваиваться проекту, в котором выбран финалист. При этом в проекте может быть только один финалист, двух быть не может.

```
CREATE TABLE order_status (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
 ) COMMENT = 'Статус проекта';
```

14. orders – это таблица проектов по поиску кандидатов. По сути, это заказ от компании на поиск персонала на какую-либо должность. Содержит свой обязательный id, id компании, а также статус проекта («новый проект», «в работе» и «закрыт»). По умолчанию проекту присваивается первый статус «новый проект».

В отношении этой таблицы могут быть связи многие-ко-многим, поэтому потребуются дополнительные объединяющие таблицы. У компании может быть сразу много заказов на поиск. Одного и того же кандидата можно сразу показывать по нескольким проектам одновременно. Отсюда следуют следующие таблицы.

```
CREATE TABLE orders (
    id SERIAL PRIMARY KEY,
    company_id BIGINT UNSIGNED NOT NULL,
    o_status_id BIGINT UNSIGNED DEFAULT 1,
    position_id BIGINT UNSIGNED NOT NULL,
```

```
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    INDEX (company_id),
    FOREIGN KEY (company_id) REFERENCES companies(id),
    FOREIGN KEY (o_status_id) REFERENCES o_status(id),
    FOREIGN KEY (position_id) REFERENCES positions(id)
 ) COMMENT = 'Заказы';
```

15. user_status – это текущее положение человека в проекте. В начале он «кандидат», если его резюме одобрили, он становится «претендентом» и готовится к очному интервью. Победитель из всех претендентов становится «финалистом» и получает оффер от компании.

Возможно также, что человек будет «исключен» из проекта, при этом для истории мы оставим его в списках с соответствующим статусом и пометками себе на будущее.

```
CREATE TABLE user_status (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255),
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP
 ) COMMENT = 'Статус кандидата';
```

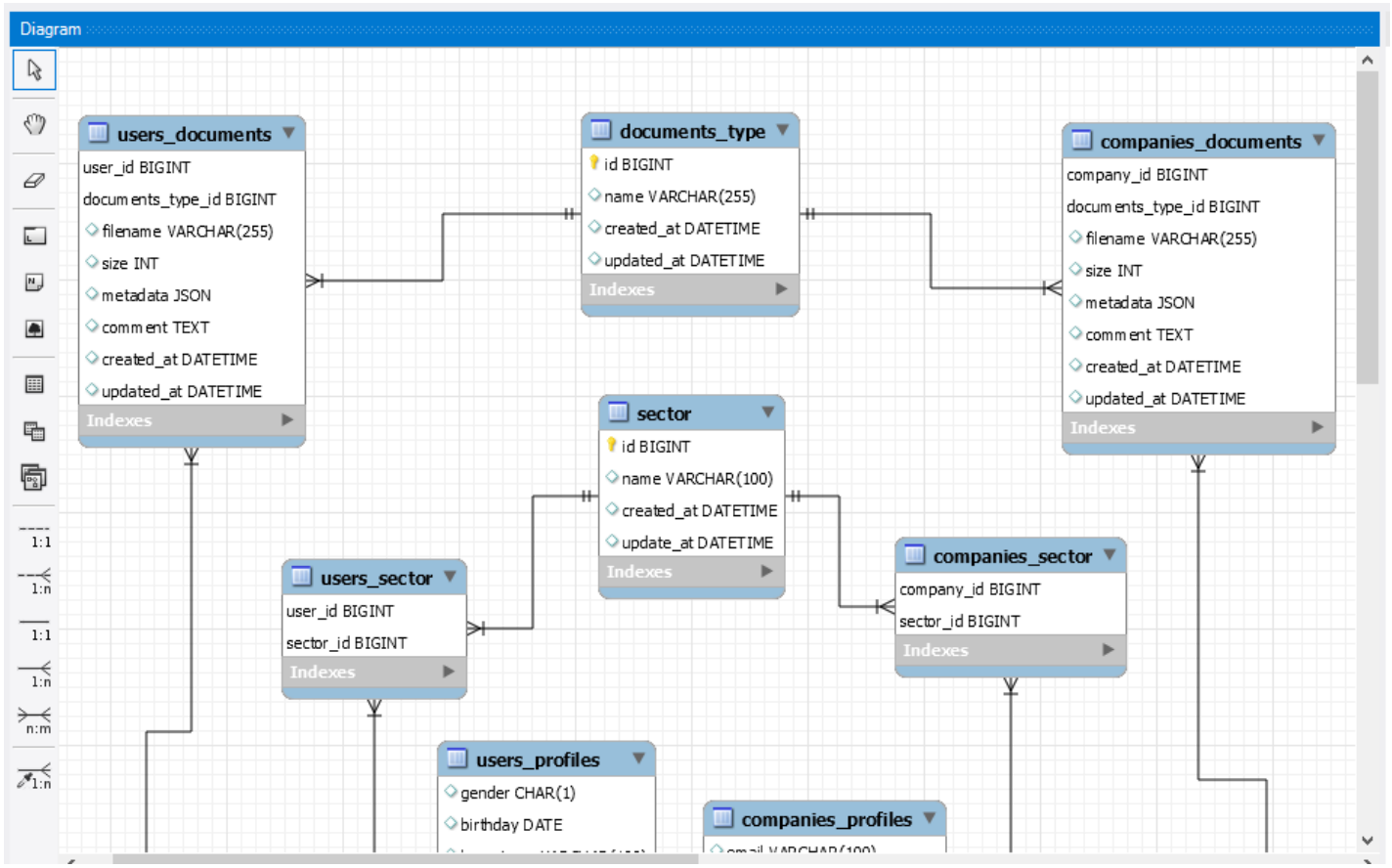
16. users_orders – это список представленных кандидатов по каждому заказу, где отражается статус человека. По умолчанию ставим значение 1 «претендент» (это значит, что его резюме отправлено заказчику. Далее он либо перейдет в статус «кандидат», либо будет исключен из проекта).

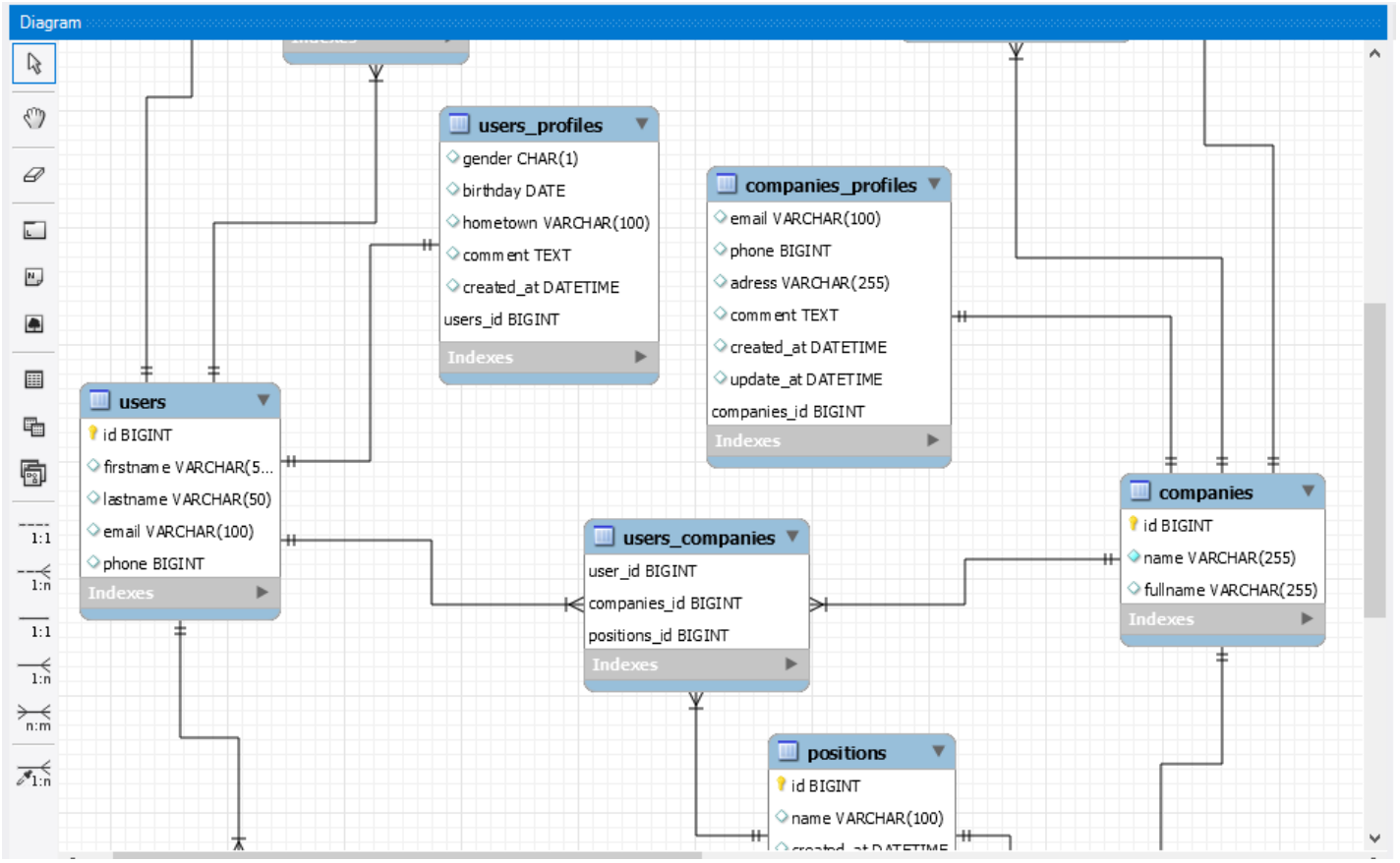
В закрытом проекте, по крайней мере, один человек из всего списка должен иметь статус финалист. При этом при присвоении человеку статуса «финалист» проект должен автоматически приобретать статус «закрыт».

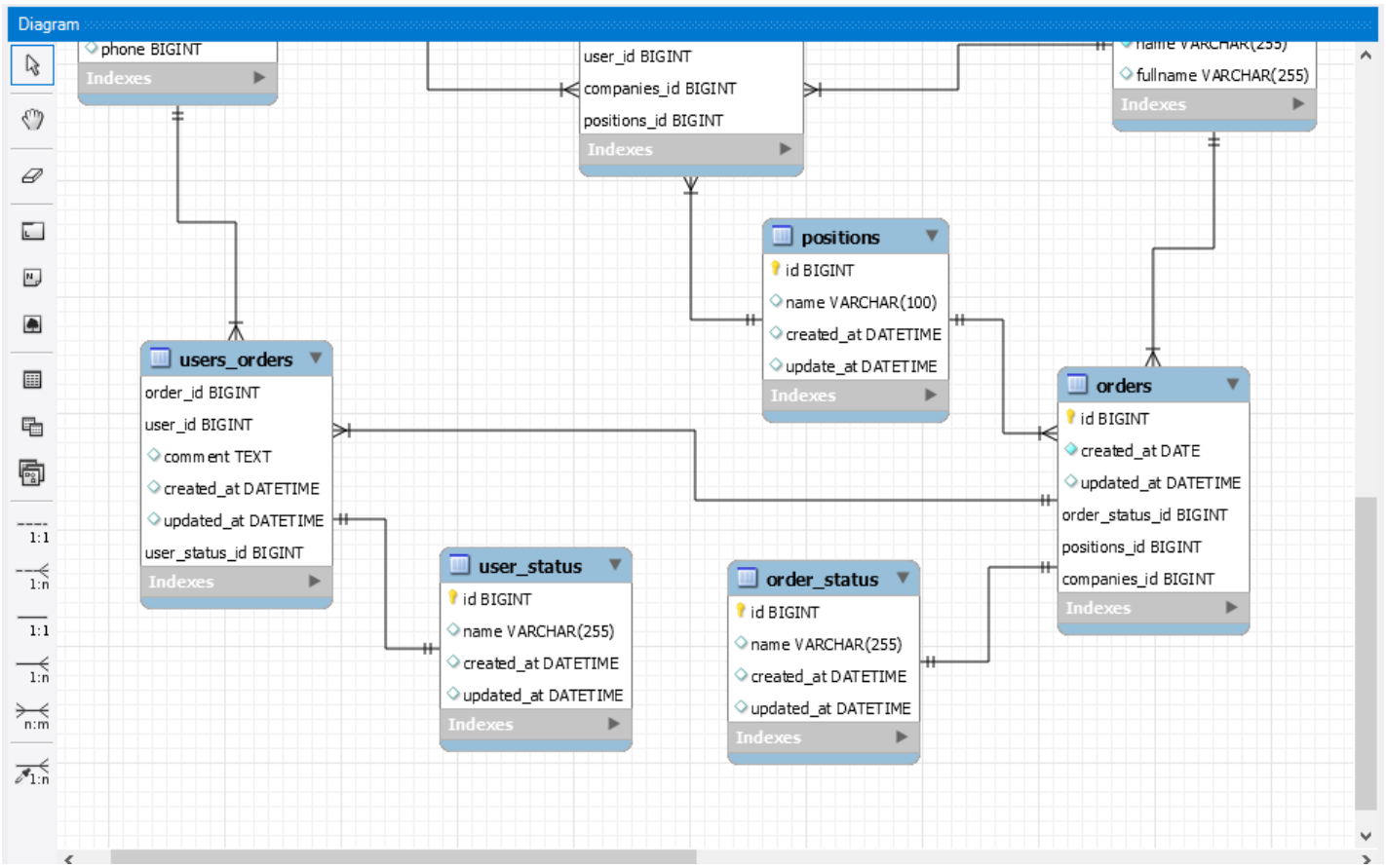
```
CREATE TABLE users_orders (
    order_id BIGINT UNSIGNED NOT NULL,
    user_id BIGINT UNSIGNED NOT NULL,
    u_status_id BIGINT UNSIGNED NOT NULL DEFAULT 1,
    `comment` text,
    created_at DATETIME DEFAULT NOW(),
    updated_at DATETIME DEFAULT CURRENT_TIMESTAMP ON UPDATE
CURRENT_TIMESTAMP,

    PRIMARY KEY (order_id, user_id),
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (u_status_id) REFERENCES u_status(id)
 ) COMMENT = 'Список представленных кандидатов';
```

4. ERDiagram

Представлено в файле - model_experium.mwb. Скриншоты:







5. Наполнение таблиц

С помощью сервиса filldb.info сгенерируем тестовые данные для таблиц. Данные можно импортировать в таблицы из разных форматов, например, csv:

- на примере таблицы users:

```
LOAD DATA LOCAL INFILE 'E:/csv/users.csv'
INTO TABLE users
FIELDS TERMINATED BY ','
ENCLOSED BY ' " '
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

При импорте из файла часто возникают ошибки, связанные с правами и безопасностью (3948, 1290). Один из вариантов решения ошибки 3948:

```
ERROR 3948 (42000): Loading local data is disabled; this must be enabled on both the client and server sides
mysql> SET GLOBAL local_infile=1;
Query OK, 0 rows affected (0.00 sec)

mysql> LOAD DATA local INFILE 'E:/csv/users.csv'
    -> INTO TABLE users
    -> FIELDS TERMINATED BY ','
    -> ENCLOSED BY '"'
    -> LINES TERMINATED BY '\n'
    -> IGNORE 1 ROWS;
Query OK, 100 rows affected (0.32 sec)
Records: 100  Deleted: 0  Skipped: 0  Warnings: 0

mysql>
```

Полная загрузка всех таблиц представлена в файле ***insert-values.sql***.

Тестовые данные содержат:
- 60 человек,
- 20 компаний,
- 14 секторов рынка,
- 12 должностей,
- 4 статуса проекта (новый проект, в работе, отменен заказчиком, завершен),
- 4 статуса кандидата в проекте (кандидат, претендент, исключен, финалист).

6. Запросы SQL:

а) **Скрипты характерных выборок**
представлены в файле - join.sql

1. посмотрим полный список всех проектов с датами:

```
SELECT o.id AS '№ заказа', c.name AS 'Компания', p.name AS
        'Наименование позиции заказа', os.name AS 'Статус проекта', o.created_at
        AS 'Дата начала проекта' FROM orders o
            JOIN positions p
                ON o.position_id = p.id
            JOIN order_status os
                ON o.order_status_id = os.id
            JOIN companies c
                ON o.company_id = c.id
        ORDER BY o.id;
```

2. посмотрим, сколько новых проектов и проектов в работе у нас в 2020 году:

```
SELECT os.name AS 'Тип проекта', COUNT(*) AS 'Всего в 2020' FROM orders o
        JOIN order_status os ON o.order_status_id = os.id
            WHERE order_status_id IN (SELECT os.id FROM order_status os
                WHERE os.name IN ('новый проект', 'в работе'))
            AND o.created_at BETWEEN '2020-01-01' AND '2020-12-31'
        GROUP BY order_status_id;
```

3. посчитаем количество пользователей из разных городов:

```
SELECT hometown AS 'Город', COUNT(*) AS 'кол-во' FROM users_profiles
  GROUP BY hometown;
```

4. посчитаем количество пользователей всего и количество заполненных профилей, найдем пользователей с незаполненными профилями:

```
SELECT 'Пользователей всего' AS ' ', COUNT(*) FROM users
        UNION
SELECT 'Заполненных профилей', COUNT(*) FROM users_profiles;
```

```
SELECT * FROM users WHERE users.id NOT IN
  (SELECT up.user_id FROM users_profiles up, users u WHERE up.user_id = u.id);
```

5. посмотрим, какие компании имеют сотрудников согласно нашей базе, поищем тех, кто имеют одинаковую должность в компании:

```
SELECT c.id, c.name FROM companies c
        WHERE c.id IN (SELECT us.company_id FROM users_companies us)
        ORDER BY c.id;

SELECT c.name AS 'компания', c.fullname AS 'полное название',
        p.name AS 'должность', COUNT(p.name) AS 'кол-во' FROM companies c
                JOIN users_companies uc
                        ON c.id = uc.company_id
                JOIN positions p
                        ON uc.position_id = p.id
        WHERE uc.company_id = (SELECT c.id FROM companies c WHERE c.name = 'ПГК')
        GROUP BY p.name;
```

7. посмотрим, компании из какого сектора у нас не охвачены:

```
SELECT s.name FROM sector s
        WHERE NOT s.id IN (SELECT cs.sector_id FROM companies_sector cs);
```

б) **Представления**

представлены в файле - view.sql

Представление v1 будет показывать соответствие компаний, проектов и статус их завершения как в типовой выборке №1 из предыдущего задания, но с отличиями. В данном представлении мы отсортируем данные так, чтобы можно было быстро найти заказы на одинаковые должности в одном статусе (например, завершен). Это нам нужно для представления №2.

```
CREATE VIEW v1 AS SELECT o.id AS '№ заказа', c.name AS 'Компания',
  p.name AS 'Наименование позиции заказа', os.name AS 'Статус проекта',
  o.created_at AS 'Дата начала проекта'

        FROM orders o
                JOIN positions p
                        ON o.position_id = p.id
                JOIN order_status os
                        ON o.order_status_id = os.id
                JOIN companies c
                        ON o.company_id = c.id
  ORDER BY o.order_status_id DESC, p.id;

SELECT * FROM v1;
```

Представление v2 будет показывать средние показатели по закрываемости вакансии. Например, мы несколько раз закрывали вакансию «Коммерческий директор», исходя из статистики можно посчитать, что в среднем надо показать заказчику 3,5 человека для того, чтобы компания кого-то выбрала. Директора по логистике найти сложнее, по нему в среднем нужно представить 5 кандидатов.

```
CREATE VIEW v2 AS

 (SELECT '1 компания' AS companies, COUNT(*) AS quantity FROM users_orders uo
        WHERE uo.order_id IN (SELECT o.id FROM orders o
                WHERE o.position_id = (SELECT p.id FROM positions p
                        WHERE p.name = 'Коммерческий директор')
        AND o.order_status_id = (SELECT os.id FROM order_status os
                WHERE os.name = 'завершен')
        AND o.company_id = (SELECT c.id FROM companies c WHERE c.name = 'ПГК')))

 UNION

 (SELECT '2 компания', COUNT(*) FROM users_orders uo
        WHERE uo.order_id = (SELECT o.id FROM orders o
                WHERE o.position_id = (SELECT p.id FROM positions p
                        WHERE p.name = 'Коммерческий директор')
        AND o.order_status_id = (SELECT os.id FROM order_status os
                WHERE os.name = 'завершен')
        AND o.company_id = (SELECT c.id FROM companies c
                WHERE c.name = 'УралКалий')));

SELECT * FROM v2;
SELECT AVG(quantity) FROM v2;
```

в) **Хранимые процедуры, триггеры**
представлены в файле - proc-&-triggers.sql

1. В **процедуре №1 (p1)** мы посмотрим общее количество завершенных проектов за каждый год. И сравним полученные данные с годовым планом по закрытию вакансий.

Подставляем в переменные план по закрытию на год @z = 10, начало годового периода @a = '2019-01-01', окончание периода @b = '2019-12-31'. Получаем 7 закрытых проектов за 2019 год, до выполнения плана не дотянули всего тремя проектами.

```
DELIMITER //
SET @z = 10// -- это план по закрытию проектов в год
SET @a = '2019-01-01'// -- это начало года
SET @b = '2019-12-31'// -- конец года

CREATE PROCEDURE p1 (inout value INT) COMMENT ' '
 BEGIN
        SET @x = value;
        SET value = @z - value;
```

```
  END//

SET @y = (SELECT COUNT(*) FROM orders o
        WHERE o.order_status_id = (SELECT os.id FROM order_status os
                WHERE os.name = 'завершен')
        AND o.created_at BETWEEN @a AND @b);

CALL p1(@y);
SELECT @z AS 'план', @x AS 'факт', @y AS 'до выполнения плана';
```

2. **Два триггера** по недопущению изменения даты рождения на дату позднее текущего дня и по недопущению изначального ввода неправильной даты рождения.

```
DELIMITER //
CREATE TRIGGER check_user_age_before_update BEFORE UPDATE ON users_profiles
  FOR EACH ROW
        begin
                IF new.birthday > CURRENT_DATE() THEN
                        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT =
                                'Дата рождения не может быть больше текущей даты';
                END IF;
        END //

DELIMITER //
DROP TRIGGER IF EXISTS check_user_age_before_insert//
CREATE TRIGGER check_user_age_before_insert BEFORE INSERT ON users_profiles
  FOR EACH ROW
        begin
                IF new.birthday > CURRENT_DATE() THEN
                        SET NEW.birthday = CURRENT_DATE();
                END IF;
        END //
```

7. Идеи по доработке и улучшению базы

- добавить в юзеры сотрудников кадрового агентства и назначить ответственных за каждый проект. Можно делать выборки по проектам, которым еще не назначен ответственный, и выбрать соответствующего сотрудника;
- добавить в таблицу заказов столбец с суммой сделки и оценивать эффективность сотрудников исходя из сумм сделок. Вообще результаты в денежном выражении открывают большие возможности для отчетов, планирования и целеполагания.