

Курсовая работа

**«База данных
кадрового агентства
EXPERIUM»**

студент Ирина Гринцевич
преподаватель Елена Коваленко

г. Москва, 2020 г.

1. Описание проекта

Кадровое агентство занимается поиском и подбором персонала на топ-позиции (уровня руководителей среднего и высшего звена).

База данных Experium используется в одноименной программе для ведения проектов кадрового агентства. Она содержит контакты кандидатов и сотрудников компаний (служит как база контактов), а также историю взаимодействия с кандидатами и компаниями, дает возможность проводить аналитику и оценивать эффективность работы по тому или иному заказу.

2. Таблицы базы данных

Скрипт создания базы данных Experium (experium_script.sql) находится на GitHub.

Наша база данных будет состоять из 16 таблиц:

1. users (люди, или кандидаты),
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. Это люди.

```
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, -- не можем не знать  
  fullname VARCHAR(255), -- полное название компании (необязательно)  
  INDEX companies_name_idx(name)  
) COMMENT = 'Компании';
```

4. companies_profiles – в карточке компании уже 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,  
  name VARCHAR(255) NOT NULL,  
  address VARCHAR(255),  
  phone BIGINT,  
  email VARCHAR(100),  
  website VARCHAR(255),  
  created_at DATETIME DEFAULT NOW(),  
  FOREIGN KEY (company_id) references companies(id)  
) COMMENT = 'Карточка компании';
```

```

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 = 'Отрасль 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. o_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 o_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 «закртыт»). 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. u_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 u_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';
```

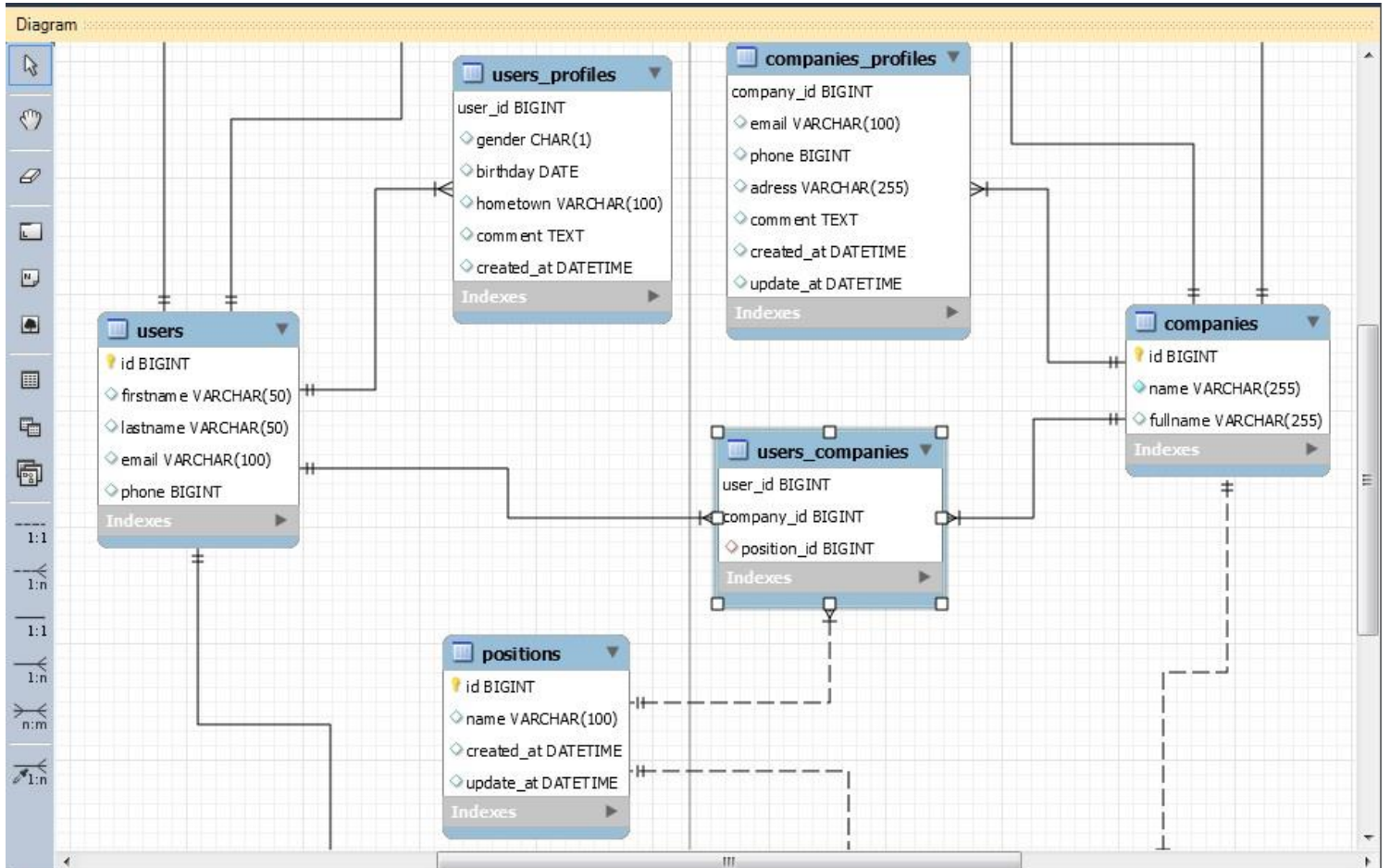
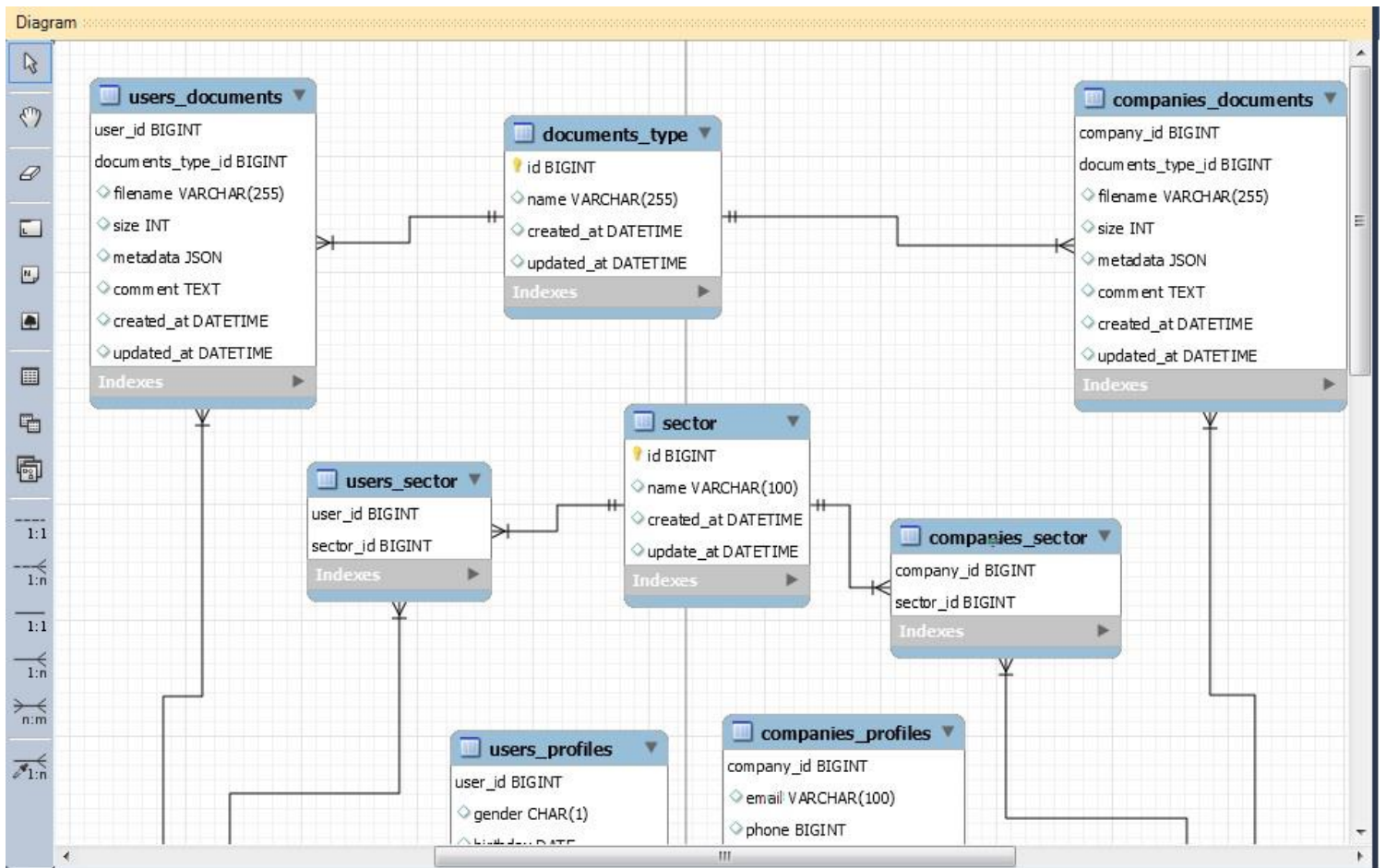
16. users_orders – это список представленных кандидатов по каждому заказу, где отражается статус человека. По умолчанию ставим значение 1 «претендент» (это значит, что его резюме отправлено заказчику. Далее он либо перейдет в статус «кандидат», либо будет исключен из проекта).

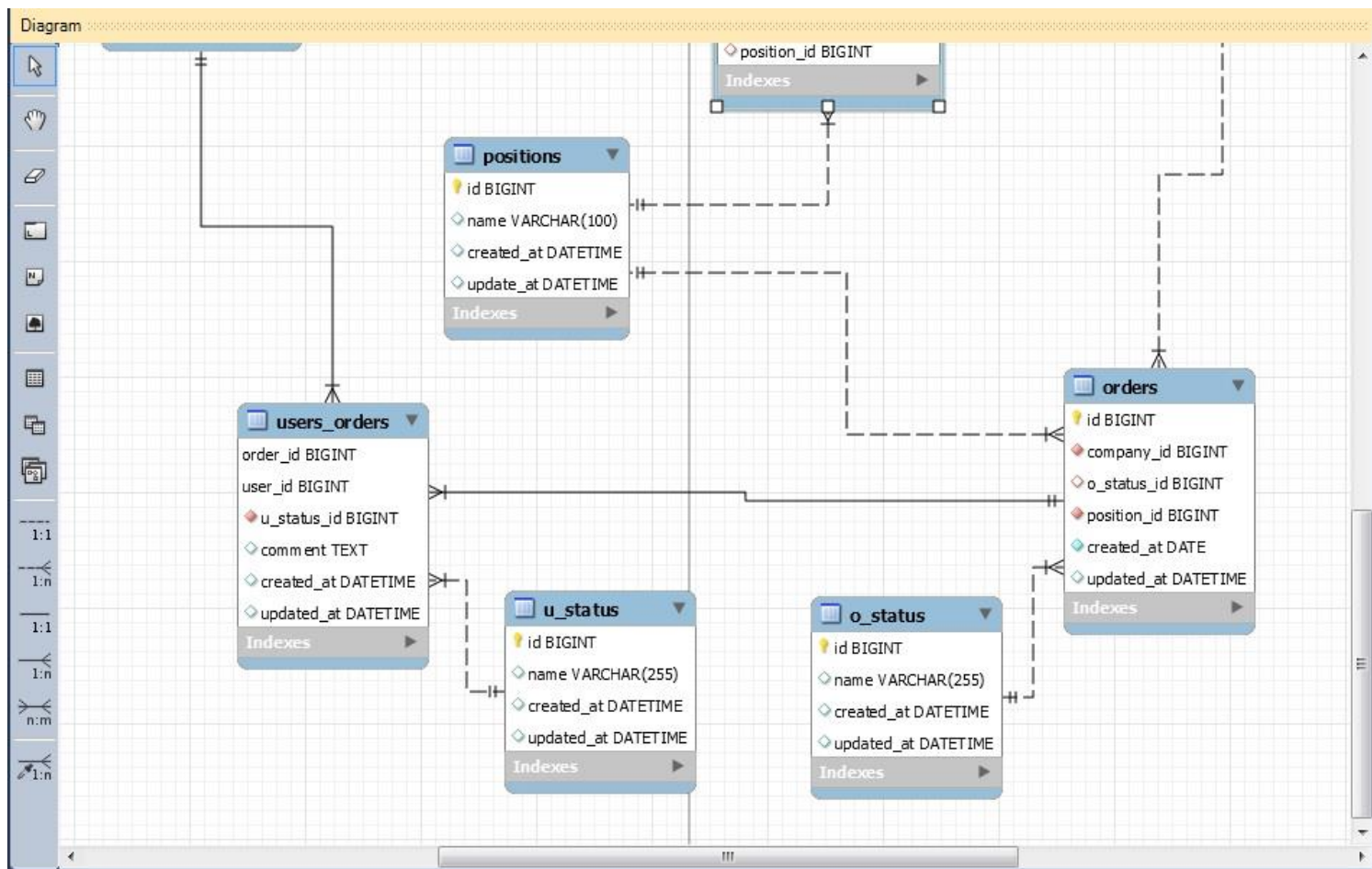
В закрытом проекте, по крайней мере, один человек из всего списка должен иметь статус финалист. При этом при присвоении человеку статуса «финалист» проект должен автоматически приобретать статус «закрыт».

```
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

Если я правильно поняла, то это модель таблиц со связями и отношениями. Представлено в файле - model_experium.mwb. Скриншоты:





5. Наполнение таблиц

Представлено в файле experium.insert-values.sql

Для наполнения можно использовать сервисы по генерации данных, но наполнение данной базы специфично, поэтому я генерировала данные сама. С именами людей и названиями компаний все понятно. У нас будет 60 человек и 20 компаний, 14 секторов рынка, 12 должностей, заказы со статусами выполнения: (новый проект, в работе, завершен), а также статус кандидата в проекте:

мы отправили резюме заказчику – статус «кандидат»,
резюме понравилось, его пригласили на интервью – статус «претендент»,
резюме не понравилось – статус «исключен» из проекта,
из всех претендентов одному сделали оффер – статус «финалист».

6. Скрипты характерных выборок и т.д.

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

Мы

1. посмотрим полный список всех проектов с датами;
2. посмотрим, сколько новых проектов и проектов в работе у нас в 2020 году;
3. посчитаем количество пользователей из разных городов;

4. посчитаем количество пользователей всего и количество заполненных профилей, найдем пользователей с незаполненными профилями;
5. посмотрим, какие компании имеют сотрудников согласно нашей базе, поищем тех, кто имеют одинаковую должность в компании;
6. посмотрим, компании из какого сектора у нас не охвачены.

б) Представления 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.

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, как ни странно, по нему в среднем нужно представить 5 кандидатов.

б) Хранимые процедуры, триггеры 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.

Важно! В скрипте я использую DELIMITER // в начале скрипта и не использую DELIMITER ; в конце, т.к. такой скрипт выдает ошибку и потребовалось несколько дней, прежде, чем выяснить ее причину. Видимо, Workbench автоматически меняет // на ;

Итак, 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.

Можно посмотреть текущую ситуацию за 2020 год. Меняем соответствующие переменные (только год). Получаем, что пока закрыт всего 1 проект, нужно еще 9.

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

рождения. Это мы один-в-один разбирали на уроке, поэтому подробно на них останавливаться не буду. Перехожу сразу к дополнительным сложным триггерам.

Файл add_triggers.sql

3. В списке кандидатов, которых мы показываем заказчику по проекту, содержится информация о статусе человека u_status_id (кандидат или претендент). Как только заказчик выбрал кого-то, то мы меняем статус соответствующего человека на «финалист». Это значит, что проект закрыт. Поэтому должен сработать триггер, который автоматически изменит статус o_status_id соответствующего проекта в таблице orders на 3, то есть, «завершен».

На примере заказа №6:

```
DELIMITER //
DROP TRIGGER IF EXISTS trg1//
CREATE TRIGGER trg1 after UPDATE ON users_orders
FOR EACH ROW
begin
    IF new.u_status_id = 3 THEN
        UPDATE orders
        SET o_status_id = 3 WHERE id = 6;
    END IF;
END//
```

И здесь мы сталкиваемся с проблемой. Нам нужно, чтобы триггер работал на всех заказах, а не только на 6. Пример триггера со вложенным запросом:

```
DELIMITER //
DROP TRIGGER IF EXISTS trg1//
CREATE TRIGGER trg1 after UPDATE ON users_orders
FOR EACH ROW
begin
    IF new.u_status_id = 3 THEN
        UPDATE orders
        SET o_status_id = 3 WHERE id = (SELECT order_id FROM users_orders
WHERE new.u_status_id = 3);
    END IF;
END//
```


Но Mysql выдает ошибку, что вложенный запрос возвращает больше, чем одну строку. Вариант с DISTINCT и GROUP BY не работает.

Требуется дополнительное значительное время, чтобы с этим разобраться. Поэтому это выходит за рамки курсовой работы.

4. И второй сложный триггер, который должен отклонять присвоение второму кандидату в проекте статус 3 "финалист". В проекте может быть только один финалист.

```
DELIMITER //
DROP TRIGGER IF EXISTS trg2//
CREATE TRIGGER trg2 BEFORE UPDATE ON users_orders
FOR EACH ROW
BEGIN
    IF new.u_status_id = 3 AND new.u_status_id = u_status_id THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = ' В проекте уже
        есть финалист, проект закрыт';
    END IF;

END //
```

Не пропускает второе условие после AND:

```
IF new.u_status_id = 3 AND new.u_status_id = u_status_id
```

Говорит, что не найден столбец u_status_id. В какой-то книжке прочитала, что триггеры задают и с выражениями FROM и WHERE. Но в mysql не проходит. Будем разбираться.

7. Предложения по доработке и улучшению базы

К сожалению, срок не позволяет проработать базу, как положено. Базой можно еще заниматься месяц. Предложения по доработке следующие:

- 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.

Но это уже другая история!