## **Проектирование базы данных**

Проектирование базы данных представляет собой ключевой этап в процессе создания информационной системы, так как именно от архитектуры базы зависит стабильность, производительность, масштабируемость и удобство поддержки всей системы. Грамотно спроектированная база данных обеспечивает корректное хранение информации, её целостность, минимизирует избыточность и обеспечивает высокую скорость доступа к данным. В рамках данного проекта была разработана база данных, поддерживающая работу автоматизированной системы подбора кандидатов на должность backend-разработчика.

### **2.3.1 Общая характеристика предметной области**

Предметная область охватывает процессы подбора технических специалистов с упором на backend-разработчиков. Основными действующими лицами являются:

* кандидат - человек, желающий получить работу, обладающий набором профессиональных навыков;
* hr-менеджер - сотрудник компании, создающий вакансии и анализирующий отклики кандидатов;
* информационная система - посредник, позволяющий связывать вакансии с кандидатами и обеспечивающий автоматизированную проверку на соответствие.

Работа системы строится на взаимодействии следующих объектов:

* профиль кандидата содержит информацию о навыках, опыте и предпочтениях;
* вакансия содержит описание позиции и список обязательных навыков;
* отклик содержит результат анализа соответствия навыков кандидата требованиям вакансии;
* тесты позволяют проверить реальные знания кандидата по каждой совпавшей компетенции.

Таким образом, база данных должна моделировать все указанные объекты и отношения между ними.

### **2.3.2 Подход к проектированию базы данных**

Проектирование базы данных выполнялось с учётом следующих принципов:

* абстрагирование - отбор только тех сущностей, которые действительно отражают бизнес-логику;
* нормализация - разбиение данных на логически независимые сущности для исключения дублирования;
* соблюдение ограничений целостности - для недопущения логических ошибок (например, отклик без связанной вакансии);
* минимизация связности - для повышения гибкости и снижения риска каскадных изменений;
* логическое и физическое разделение проектирования - сначала создаётся концептуальная модель, затем она трансформируется в реализацию через ORM;
* использование асинхронного доступа - для масштабируемости и высокой производительности системы.

### **2.3.3 Список сущностей и их функциональное назначение**

В результате анализа предметной области были выделены следующие сущности:

* пользователь - представляет собой любую зарегистрированную единицу в системе (hr или кандидат). Хранит авторизационные и контактные данные.
* кандидат - расширение пользователя, обладающее профилем с навыками и историей откликов.
* hr - расширение пользователя, имеющее доступ к управлению вакансиями.
* вакансия - описание открытой должности, включающее требования, описание компании и перечень нужных навыков.
* навык - отдельная единица профессиональной квалификации. Каждый навык может быть связан с несколькими вакансиями и профилями.
* отклик - действие, инициированное кандидатом, представляющее интерес к вакансии. Хранит процент совпадения, дату отклика, статус.
* тест - набор заданий, автоматически формируемых по каждому навыку при достаточном совпадении между кандидатом и вакансией.
* результат теста - данные о попытке прохождения теста: завершённость, оценка, дата выполнения.
* вопрос - элемент теста, включающий текст вопроса, варианты ответов и правильный ответ.
* профиль кандидата - сущность, хранящая список связанных с кандидатом навыков.
* связь "вакансия - навык" - реализует отношение "многие ко многим" между вакансиями и навыками.
* связь "профиль - навык" - позволяет указать, какие навыки есть у кандидата.
* связь "отклик - тест" - привязывает отклик к отправленным тестам.

### **2.3.4 Структура и типы связей между сущностями**

В проектируемой базе данных реализуются следующие типы связей:

* один к одному - между пользователем и профилем (кандидат или hr);
* один ко многим - между hr и вакансиями, между вакансией и откликами;
* многие ко многим - между вакансиями и навыками, между профилями кандидатов и навыками;
* один ко многим - между откликом и тестами, между тестом и вопросами.

Реализация отношений "многие ко многим" осуществляется посредством промежуточных таблиц, каждая из которых содержит внешние ключи на связываемые сущности и, при необходимости, дополнительные атрибуты.

### **2.3.5 Обоснование выбора атрибутов и типов данных**

Для каждой сущности были определены ключевые атрибуты, удовлетворяющие следующим критериям:

* минимальность - каждый атрибут содержит атомарное значение;
* уникальность - первичные ключи обеспечивают однозначную идентификацию каждой записи;
* согласованность - каждый атрибут имеет чётко определённый тип данных, соответствующий своей природе;
* нормализованность - отсутствует повторение данных между таблицами.

Примеры:

* у пользователя: идентификатор (integer), электронная почта (varchar), дата регистрации (timestamp), роль (enum);
* у вакансии: идентификатор (integer), название (varchar), описание (text), идентификатор hr (foreign key);
* у отклика: идентификатор (integer), процент совпадения (float), статус (enum), внешний ключ на кандидата и вакансию;
* у навыка: уникальное название (varchar);
* у теста: идентификатор (integer), связан с откликом и навыком, статус выполнения (boolean), оценка (float);
* у вопроса: текст вопроса (text), список опций (jsonb), индекс правильного ответа (integer).

В таблице 3 описаны сущности и их атрибуты.

Таблица 4 - Сущности и атрибуты базы данных

|  |  |  |
| --- | --- | --- |
| Имя сущности | Описание | Тип |
| Пользователь | id пользователя | primary key |
|  | email | текстовый |
|  | hashed\_password | текстовый |
|  | роль | текстовый |
| Кандидат | id профиля кандидата | primary key |
|  | id пользователя | внешний ключ |
| HR | id профиля HR | primary key |
|  | id пользователя | внешний ключ |
| Вакансия | id вакансии | primary key |
|  | заголовок | текстовый |
|  | компания | текстовый |
|  | описание | текстовый |
|  | id HR-профиля | внешний ключ |
| Навык | id навыка | primary key |
|  | название навыка | текстовый |
| Навыки вакансии | id вакансии | внешний ключ |
|  | id навыка | внешний ключ |
| Навыки кандидата | id профиля кандидата | внешний ключ |
|  | id навыка | внешний ключ |
| Отклик | id отклика | primary key |
|  | id профиля кандидата | внешний ключ |
|  | id вакансии | внешний ключ |
|  | статус | текстовый |
|  | процент совпадения | числовой |
| Тест отклика | id теста | primary key |
|  | id отклика | внешний ключ |
|  | id навыка | внешний ключ |
|  | завершён | логический |
|  | оценка | числовой |
| Вопрос | id вопроса | primary key |
|  | id навыка | внешний ключ |
|  | текст вопроса | текстовый |
|  | варианты ответов | json |
|  | индекс правильного ответа | целочисленный |

### **2.3.6 Хранение вложенных структур**

Особенностью проектируемой базы данных является необходимость хранения списков (варианты ответов в вопросах). Для этого в PostgreSQL используется тип данных jsonb, позволяющий:

* сохранять списки строк и структур в виде одного поля;
* осуществлять фильтрацию, индексирование и извлечение значений;
* обеспечивать гибкость хранения и передачи данных в API.

### **2.3.7 Обеспечение целостности данных**

Целостность базы данных обеспечивается следующими механизмами:

* первичные ключи на каждой сущности;
* внешние ключи между связанными таблицами;
* каскадные удаления в ассоциативных таблицах;
* ограничения уникальности;
* ограничения not null на обязательных полях;
* использование перечислений (enum) для полей статуса, ролей и категорий.

### **2.3.8 Масштабируемость и производительность**

Архитектура базы данных спроектирована с учётом следующих требований к масштабируемости:

* использование индексов на часто запрашиваемые поля (например, email, внешние ключи);
* нормализованная модель, позволяющая расширять систему без дублирования данных;
* асинхронный доступ к базе, позволяющий обрабатывать множество параллельных запросов;
* кэширование данных на уровне приложения для уменьшения нагрузки на БД;
* потенциальная возможность горизонтального масштабирования базы за счёт использования реплик.

### **2.3.9 Интеграция с backend и API**

Все модели базы данных реализованы с использованием SQLAlchemy ORM и сопровождаются Pydantic-схемами. Это обеспечивает:

* строгую типизацию данных;
* автоматическую валидацию при получении и отправке данных через API;
* удобную сериализацию и десериализацию;
* однозначную связь между логической моделью и структурой таблиц.

### **2.3.10 Перспективы расширения модели данных**

Концепция проектирования предусматривает возможность расширения базы данных без необходимости полного рефакторинга. Например:

* добавление сущностей "комментарии", "собеседования", "приглашения";
* хранение резюме кандидатов;
* внедрение версионирования вакансий;
* автоматическое логирование всех действий пользователей.

Такая гибкость достигается за счёт соблюдения принципов нормализации и слабой связности компонентов.

В результате проектирования была создана полнофункциональная база данных, удовлетворяющая всем требованиям предметной области. Она обеспечивает корректное моделирование бизнес-процессов, устойчивость к ошибкам, высокую производительность и гибкость. База данных является надёжной основой для функционирования всей информационной системы, а её структура обеспечивает лёгкость сопровождения и масштабирования проекта.