ERD (Entity-Relationship Diagram) — это графическое представление структуры данных, которое позволяет описать сущности (entities) и их взаимосвязи в информационной системе. ERD помогает моделировать базы данных и понимать связи между данными.

В ERD выделяют три основные компоненты:

1. **Сущности (Entities):** Представляют объекты или понятия, хранящие информацию в базе данных. Сущности обычно имеют атрибуты, описывающие их характеристики.
2. **Связи (Relationships):** Определяют взаимосвязи между сущностями. Связи указывают на то, как сущности связаны друг с другом и какие свойства этих связей.
3. **Атрибуты (Attributes):** Это характеристики сущностей или связей, которые хранят информацию о них.

Для нашего случая все сущности выбраны отлично, все что нужно присутствует.

**Парсинг**

Существует несколько протоколов и методов, которые могут использоваться для парсинга данных из баз данных, в зависимости от типа базы данных и целей парсинга. Вот некоторые из наиболее распространенных методов:

SQL (Structured Query Language): SQL - это стандартный язык для работы с реляционными базами данных. Вы можете использовать SQL для выполнения запросов, выборки, обновления и удаления данных из базы данных. В большинстве языков программирования есть библиотеки или драйверы, которые позволяют выполнять SQL-запросы к базам данных.

ODBC (Open Database Connectivity): ODBC - это стандартный интерфейс для взаимодействия с различными типами баз данных. Он позволяет создавать универсальные приложения, которые могут подключаться к разным базам данных, используя один и тот же код.

JDBC (Java Database Connectivity): JDBC - это Java API для взаимодействия с базами данных с использованием языка программирования Java. Он позволяет Java-приложениям подключаться и взаимодействовать с разными СУБД.

ORM (Object-Relational Mapping): ORM библиотеки, такие как SQLAlchemy для Python или Hibernate для Java, позволяют вам работать с базой данных, используя объекты и классы, вместо явных SQL-запросов. ORM облегчает маппинг объектов приложения на структуру базы данных.

Web API: Некоторые базы данных предоставляют Web API, которые позволяют вам отправлять HTTP-запросы для получения данных. Например, для работы с базами данных NoSQL, такими как MongoDB или Firebase Realtime Database, вы можете использовать соответствующие API.

Специфичные библиотеки и драйверы: Для некоторых баз данных существуют специфичные библиотеки и драйверы, которые облегчают парсинг данных. Например, для работы с PostgreSQL, вы можете использовать библиотеку psycopg2 для Python.

Выбор протокола или метода зависит от типа базы данных, языка программирования, который вы используете, и вашей конкретной задачи парсинга данных.

В нашем случае будет использован SQL

Вот некоторые из основных принципов информационной безопасности, которые следует учитывать при работе с протоколами парсинга SQL:

1. **Аутентификация**: Убедитесь, что только авторизованные пользователи имеют доступ к системе парсинга SQL. Используйте сильные методы аутентификации, такие как пароли, токены или двухфакторную аутентификацию.
2. **Авторизация**: Ограничьте доступ к данным и операциям парсинга SQL на основе ролей и прав доступа. Не предоставляйте излишних полномочий.
3. **Шифрование**: Защитите данные, передаваемые между компонентами системы парсинга SQL, с использованием протоколов шифрования, таких как SSL/TLS.
4. **Защита от инъекций SQL**: Убедитесь, что система парсинга SQL защищена от атак инъекций SQL, таких как SQL-инъекции. Используйте параметризованные запросы и валидацию данных для предотвращения таких атак.
5. **Мониторинг и журналирование**: Ведите журнал действий пользователей и событий парсинга SQL. Это позволит быстро выявить аномальную активность и потенциальные угрозы.
6. **Обновление и патчи**: Регулярно обновляйте программное обеспечение и библиотеки, используемые в системе парсинга SQL, чтобы исправить известные уязвимости.
7. **Фильтрация ввода**: Всегда фильтруйте и валидируйте входные данные, чтобы предотвратить передачу вредоносных запросов через протокол парсинга SQL.
8. **Управление сеансами**: Внимательно управляйте сеансами пользователей и предотвращайте несанкционированный доступ к активным сеансам.
9. **Ограничение ресурсов**: Ограничьте ресурсы, которые могут использовать пользователи, чтобы предотвратить исчерпание ресурсов и атаки типа "отказ в обслуживании".
10. **Обучение и осведомленность**: Обучите персонал, работающий с парсингом SQL, в вопросах информационной безопасности и соблюдении правил безопасности.

В нашем случае соблюдаются все необходимые принципы ИБ, следовательно не возникнет случаев проблем с информационной безопасностью