Вопросы к ГОС. ЭКЗАМЕНУ

Базы данных

1. Нормализация баз данных. 1,2,3 нормальные формы, определения и способ

приведения.

Нормализация — это способ организации данных. В нормализованной базе нет повторяющихся данных, с ней проще работать и можно менять её структуру для разных задач.

**Первая нормальная форма**

В базе данных не должно быть дубликатов и составных данных.

Далее картинка неправильно/правильно



**Вторая нормальная форма**

Если упростить: у каждой записи в базе данных должен быть первичный ключ. Первичный ключ — это элемент записи, который не повторяется в других записях.

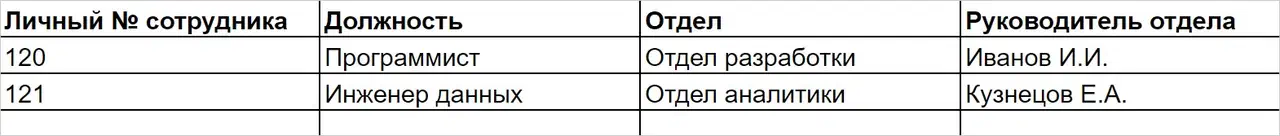
Как надо:



**Третья нормальная форма**

В записи не должно быть столбцов с неключевыми значениями, которые зависят от других неключевых значений.

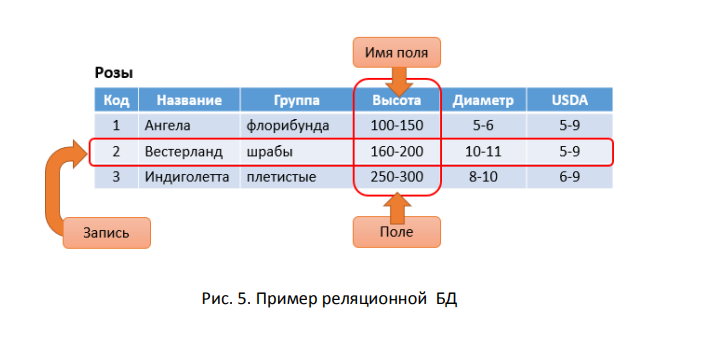
Данные о руководителях отделов нужно вынести в другую таблицу. Тогда в основной таблице не будет транзитивных связей, и она будет соответствовать третьей нормальной форме. Картинка КАК НЕ НАДО:



2. Таблицы, индексы, поля, типы полей, операции над полями.

Таблица — это совокупность связанных данных, хранящихся в структурированном виде в базе данных.





Поля бывают разных типов: числовой, символьный, логический, дата/время

С полями базы данных можно производить следующие операции

* описание (указание имени, типа и длины поля);
* редактирование (изменение имени, типа и длины поля);
* манипуляция (добавление, перемещение и удаление полей).

3. Связи между таблицами, ключи.

**Один к одному**

В этом типе связи каждая запись в одной таблице соответствует только одной записи в другой таблице, и наоборот. Пример: таблица "Пользователь" может иметь связь один-к-одному с таблицей "Профиль", где каждому пользователю соответствует только один профиль, и каждому профилю соответствует только один пользователь

**Один ко многим**

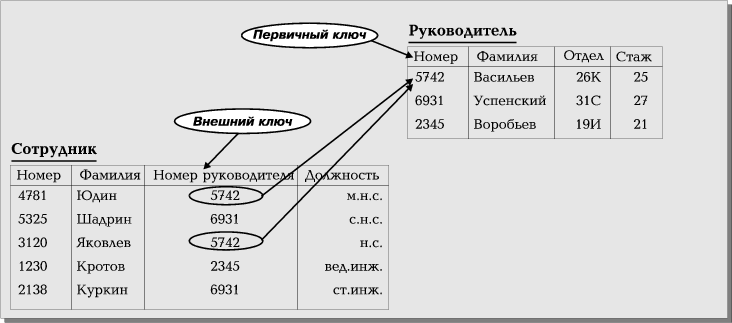
В этом типе связи каждая запись в одной таблице может иметь несколько соответствующих записей в другой таблице, но каждая запись во второй таблице соответствует только одной записи в первой таблице. Пример: таблица "Категория" может иметь связь один-ко-многим с таблицей "Продукты", где каждой категории соответствуют множество продуктов, но каждый продукт относится только к одной категории.

**Многие ко многим**

В этом типе связи множество записей в одной таблице может соответствовать множеству записей в другой таблице, и наоборот. Для реализации связи многие-ко-многим обычно используется промежуточная таблица, которая содержит связи между записями двух связанных таблиц. Пример: таблицы "Студенты" и "Курсы" могут иметь связь многие-ко-многим через промежуточную таблицу "Запись", где каждый студент может записаться на несколько курсов, и каждый курс может иметь несколько студентов.

Первичный ключ — особенное поле в SQL-таблице, которое позволяет однозначно идентифицировать каждую запись в ней. Как правило, эти поля используются для хранения уникальных идентификаторов объектов, которые перечислены в таблице, например, это может быть ID клиента или товара.

Внешний ключ нужен для того, чтобы связать две разные SQL-таблицы между собой. Внешний ключ таблицы должен соответствует значению первичного ключа таблицы, с которой он связан.

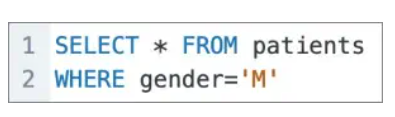


Клиент-серверные системы

1. TransactSQL-оператор Select.

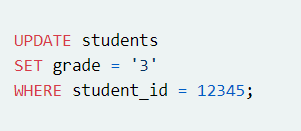
Оператор SQL с помощью которого осуществляется выборка данных.

SELECT \* FROM test\_table

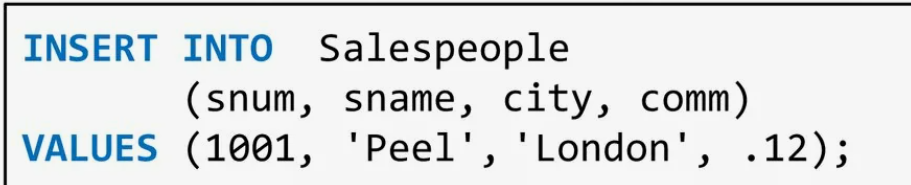


1. TransactSQL-оператор Update, Insert

**Update** - Используется для обновления данных

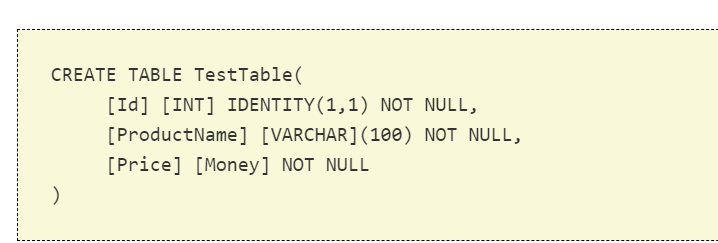


**INSERT** – это инструкция языка T-SQL, которая предназначена для добавления данных в таблицу, т.е. создания новых записей.

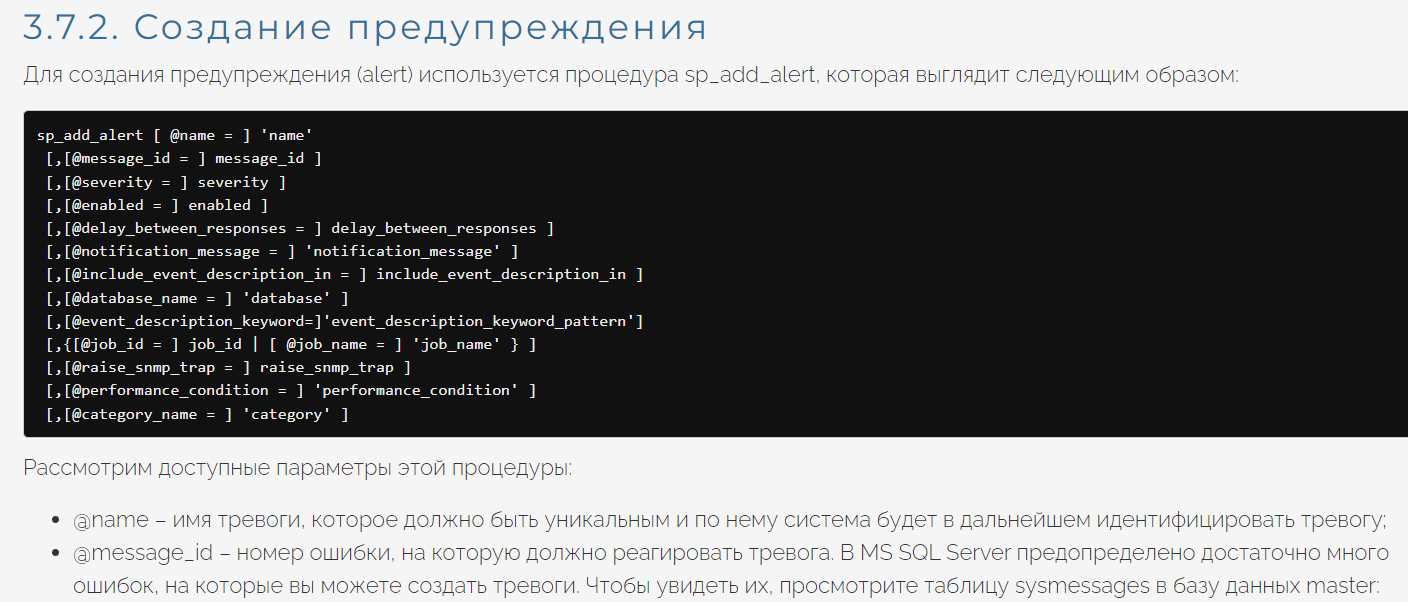


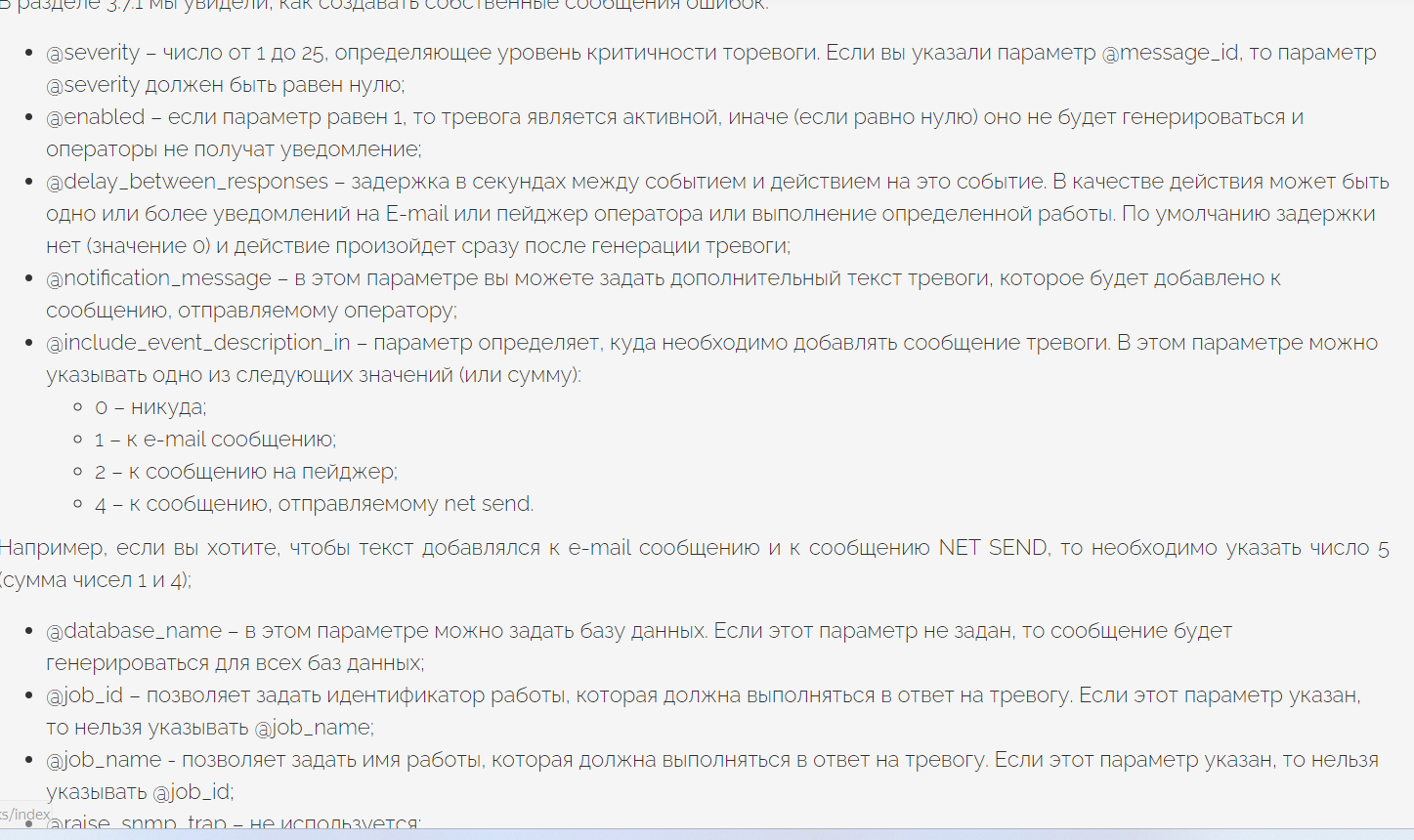
1. TransactSQL-оператор Create, Alert

Инструкция CREATE TABLE создает новую таблицу базы данных со всеми соответствующими столбцами требуемого типа данных.



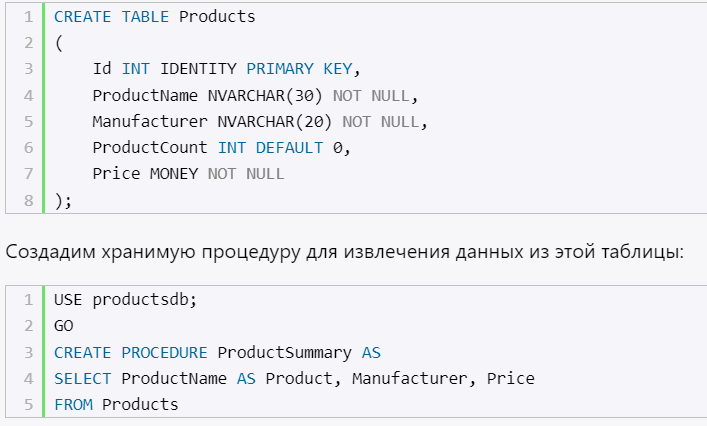
Предупреждения (Alert или я еще люблю говорить Тревога) позволяют контролировать работу вашего сервера. Вы можете создавать тревоги на определенные события сервера (ошибки данных, ошибки запросов, ошибки или удачные выполнения работ и т.д.) и направлять сообщения определенным операторам.





1. TransactSQL хранимые процедуры, создание, применение.

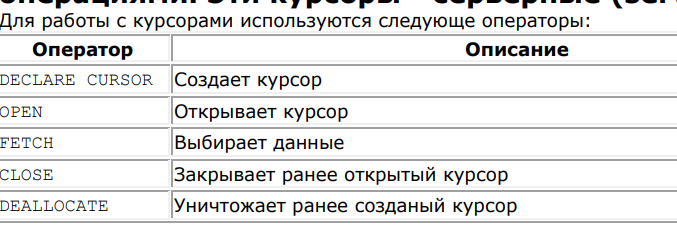
**Хранимые процедуры** – это объекты базы данных, в которых заложен алгоритм в виде набора SQL инструкций. Иными словами, можно сказать, что хранимые процедуры – это программы внутри базы данных. Хранимые процедуры используются для сохранения на сервере повторно используемого кода. Создается при помощи CREATE PROCEDURE



1. Cursor, Trigger создание и применение.

**Курсор** — ссылка на контекстную область памяти. Используются курсоры в хранимых процедур.

Курсоры позволяют обрабатывать данные для каждой возвращаемой строки отдельно, не пользуясь множественными, традиционными для SQL операциями. Эти курсоры – серверные



Курсор объявляется оператором DECLARE:

DECLARE cursor\_name [INSENSITIVE] [SCROLL]

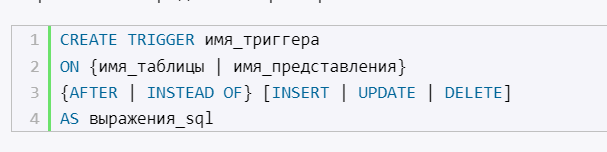
CURSOR

FOR select\_statement

[FOR {READ ONLY | UPDATE [OF column\_list]}]

**Tриггер** - это специальный тип хранимой процедуры, которая автоматически вызывается, когда данные в определенной таблице добавляются, удаляются или изменяются с помощью SQL-предложений INSERT, DELETE или UPDATE. В зависимости от того, какое из событий, связанных с изменением данных, инициирует запуск триггера, он называется insert trigger, delete trigger или update trigger.

Для создания триггера применяется выражение CREATE TRIGGER



После оператора и имени триггера необходимо указать в каком случае будет срабатывать триггер. Возможно 6 вариантов:

BEFORE INSERT

BEFORE UPDATE

BEFORE DELETE

AFTER INSERT

AFTER UPDATE

AFTER DELETE

То есть триггер срабатывает либо до, дибо после вставки, обновления, удаления данных из БД в SQL.

1. С# элементы управления для доступа к базам данных.????????????

Элементы управления источниками данных включают любые элементы управления, реализующие интерфейс IDataSource. SqlDataSource (существуют также другие источники данных: ObjectDataSource, XmlDataSource, SiteMapDataSource и LinqDataSource). Этот элемент позволяет подключаться к любому источнику данных, который имеет поставщика данных ADO.NET. Сюда относятся SQL Server, Oracle и OLE DB или ODBC.

Элемент управления SqlDataSource позволяет разработчику получать доступ к данным, хранящимся в любой реляционной базе данных, поддерживающей ADO.NET. Он может использовать поставщик System.Data.SqlClient для доступа к базе данных SQL Server, поставщик System.Data.OleDb, поставщик System.Data.Odbc или поставщик System.Data.OracleClient для доступа к Oracle. Таким образом, SqlDataSource, безусловно, используется не только для доступа к данным в базе данных SQL Server.

Чтобы использовать SqlDataSource, необходимо просто указать значение для свойства ConnectionString и указать команду SQL или хранимую процедуру. Элемент управления SqlDataSource отвечает за работу с базовой архитектурой ADO.NET. Он открывает подключение, запрашивает источник данных или выполняет хранимую процедуру, возвращает данные, а затем закрывает подключение для вас.

1. Трехзвенная архитектура приложения, использующего базу данных

**Трёхуровневая архитектура** — это архитектурная модель программного комплекса, предполагающая наличие в нём трёх типов компонентов (уровней, звеньев):

* Клиентские приложения — с ними работают пользователи.
* Серверы приложений — с ними работают клиентские приложения.
* Серверы баз данных — с ними работают серверы приложений.



1. Транзакции- определение, назначение и применение.

**Транзакция** — это архив для запросов к базе. Он защищает ваши данные благодаря принципу «всё, или ничего».

Представьте, что вы решили послать другу 10 файликов в мессенджере. Какие есть варианты:

Кинуть каждый файлик отдельно.

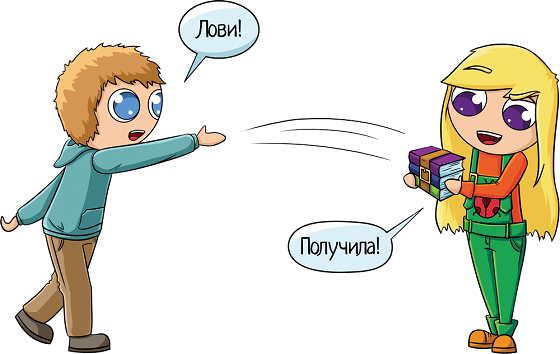
Сложить их в архив и отправить архив.

Вроде бы разницы особой нет. Но что, если что-то пойдет не так? Соединение оборвется на середине, сервер уйдет в ребут или просто выдаст ошибку...

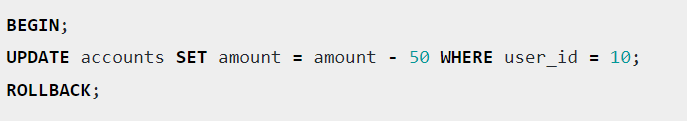
В первом случае ваш друг получит 9 файлов, но не получит один.



Во втором не получит ничего. Нет промежуточных состояний. Или получил всё, или не получил ничего. Но зато если произошла ошибка, вы снова перешлете сообщение. И друг получит все файлики разом, не придется проверять «не потерялся ли кто».



Если нужно, транзакцию можно откатить самостоятельно. Для этого необходимо выполнить запрос ROLLBACK до COMMIT. Это нужно, когда выполняются запросы из кода приложения.



В информатике есть набор требований к транзакционной системе, которые гарантируют ее надежность — ACID. К ним относятся:

* Atomicity (Атомарность) - Любая транзакция не может быть частично завершена — она либо выполнена, либо нет.
* Consistency (Согласованность) - Завершившаяся транзакция должна сохранять согласованность базы данных.
* Isolation (Изолированность) - Когда транзакция выполняется, параллельные транзакции не должны оказывать влияния на ее результат.
* Durability (Устойчивость) - Изменения, которые сделаны успешно завершенной транзакцией, должны остаться сохраненными после возвращения системы в работу.