**База данных**

**База данных** – это хранилище информации о реальных объектах, событиях или явлениях организованная таким образом, чтобы обеспечить удобные операции добавления, удаления, обновления и чтения данных.

**СУБД(DBMS)–** Система Управления Базами Данных – это набор программ и утилит, предназначенных для взаимодействия с БД, которые кроме основных операций добавления, удаления, чтения данных, позволяет выполнять с БД такие операции как:

* Обеспечение санкцианированного доступа к данным (защита данных)
* Обеспечение резервного копирования
* Обеспечение удаленного доступа

Когда речь идет о написании приложения взаимодействующего с БД реально подразумевается, что такое приложение будет взаимодействовать с СУБД, которая в свою очередь и будет взаимодействует с БД. Соответственно, такая СУБД должна быть установлена на компьютере на котором будет работать разрабатываемое приложение.

*Реляционные СУБД/БД*

**Реляционная БД –** это БД, в которой данные представлены в виде таблицы, а все операции над этими данными сводятся к операциям над этими таблицами. В БД может быть множество таблиц, у каждой таблицы есть имя. Имя таблицы уникально. Каждая таблица состоит из строк и столбцов. У каждого столбца в таблице есть имя и тип данных. Создание таблицы в реляционных БД сводится к перечислению названий и типов данных столбцов. Столбцы еще называют **полями**. Строки таблицы хранят непосредственно данные. Каждая строка представляет реальный объект или событие или явление. Строки таблицы еще называют **записями**.На пересечении строки и столбца находится одно значение. Бывают ситуации, когда хранимые в таблице элементы совпадают по все, либо почти по все значениям. Возникает проблема в их идентификации. Для решения этой проблемы предназначем специальный столбец, который называется первичным ключем. Первичный ключ это столбец в таблице, значение в котором для каждой строки уникальны и не могут быть равны NULL (**NULL**–специально значение в реляционных базах даных, которое обозначает отсутствие данных).

*12 правил Эдварда Кодда*

*12 правил, которым должна соответствовать* ***реляционная*** *СУБД*

1. ***Правило информации*** – вся информация в БД должна быть представлена исключительно на логическом уровне и в виде значений, хранящихся в таблицах.
2. ***Правило гарантированного доступа*** – доступ к каждому элементу данных должен осуществляться с помощью комбинации имени таблицы, имени столбца и значения первичного ключа.
3. ***Правило поддержки недействительных значений*** – отсутствующие или пропущенные данные должны быть представлены в виде специального значения (NULL), которое не является нулем, не является пустой строкой. И это значение может учавствовать в выборке данных.
4. ***Правило динамического каталога*** – любая реляционная БД должна уметь себя описывать, то есть содержать информацию о своей структуре. И эта информация должна быть представлена в виде таблице, которая хранится в этой же БД. Такие таблицы называют системными таблицами.
5. ***Правило исчерпывающего подязыка данных*** – в реляционных БД может быть несколько языков данных, но должен быть один высокоуровненвый язык данных, с помощью которого можно было бы выполнить все операции над БД.
6. ***Правило обновления представлений(View)***– если представления теоретически можно обновить, то оно долнжо быть обновляемым.

***View***– это виртуальная таблица, которая формируется на основе реальных таблиц в момент обращения к ней.

1. **Правило добавления, обновления и удаления** – гласит о том, что в реляционных СУБД должна быть возможность добавлять, обновлять и удалять множество строк одной командой.
2. **Правило независимости физических данных** – гласит, что прикладные программы и утилиты для работы с данными должны на логическом уровне оставаться без изменений при любых изменениях способов хранения данных или методов доступа к ним.
3. **Правило независимости логических данных** – гласит, что прикладные программы и утилиты для работы с данными должны на логическом уровнео оставаться нетронутыми при внесении в таблицы изменений, которые не приводят к потере данных.
4. **Правило независимости условий целосности –** СУБД должна предоставлять пользователю самостоятельно накладывать на БД условия целосности. И эти условия целосности должны храниться в системных таблицах.

**Целосность –** это согласованность данных между таблицами

1. **Правило независимости распространения** – гласит, что СУБД должна обеспечивать возможность работы с распределенным данными, то есть данными, находящимися на других компьютерах.
2. **Правило единственности** – если в СУБД есть низкоуровневый язык данных, то должна отсутствовать возможность с помощью этого языка обойти условия целосности на уровне языка высокого уровня.

**SQL – Structured Query Language**

*Категории языка SQL*

1. **DDL** – Data Defination Language – Языкобъявленияданных. **Команды:** *CREATE, ALTER, DROP*
2. **DML –** DataManipulationLanguage – Языеоперированияданными**. Команды:** *INSERT, DELETE, UPDATE, SELECT*
3. **DCL –** DataControlLanguage – Язык управления данными.С помощью этой категории осуществляется назначение пользователей привелегий на выполнение тех или иных операций над БД. **Команды:** *GRANT, REVOKE*

***Домашнее задание***

1. Вывести все книги-новинки дороже 30 рублей
2. Вывести книги, дата издания которых неизвестна
3. Сделать четырехзначным код книги если он не 4
4. Вывести книги категории учебники издательства DiaSoftи Питер. Выполнить сортировку по названию.
5. Вывести названия книг у которых стоимость одной страницы больше 10 коп.

**Агрегирующие функции (функции агрегации)**

* Min – числовой

*SELECT MIN(Price) AS min\_price FROM books*

* Max

*SELECT MAX(Price) AS max\_price FROM books*

* Avg

*SELECT AVG(Price) AS avg\_price FROM books WHERE Izd = “Diasoft”*

* sum
* count

*SELECT COUNT(Date), COUNT(\*) AS date FROM books*

**Спец. Символы для шаблонов**

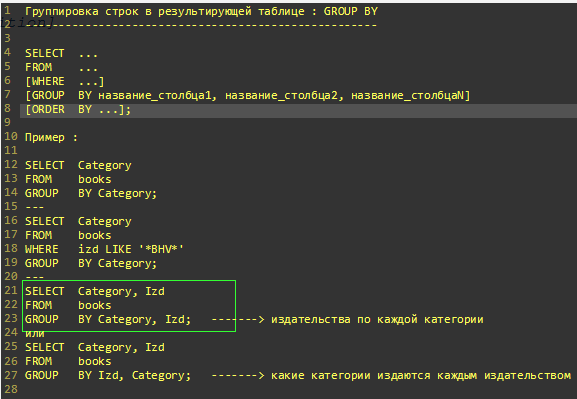
* \* - любой символ
* # - любая цифра
* ?(\_) – в данной позиции находится один любой символ
* [a-z] -в данной поизиции находится символ из диапазона a-z
* [abcdef] – в данной позиции находится символ из множества abcdef
* [!a-z] – в данной позиции находится символ не из диапазона a-z
* [!0-9] – НЕ цифра
* [!abcdef] – НЕ из множества abcdef

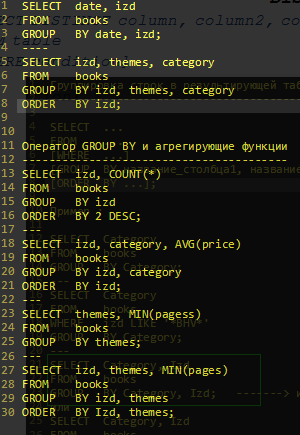
*Исключение повторяющихся строк из результатов таблицы:* ***DISTINCT***

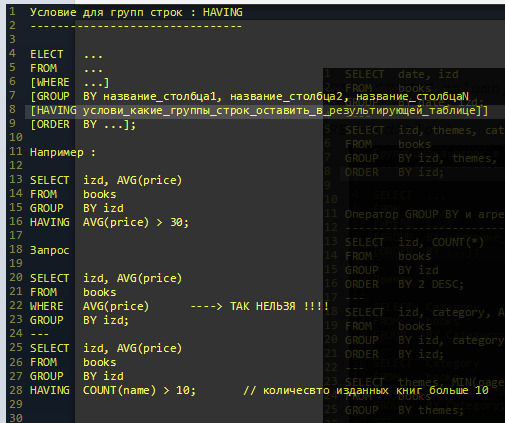
*SELECT DISTINCT column, column2, columnN*

*FROM table*

*[WHERE condition]*

**

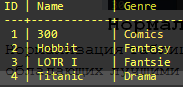
**

**

**Нормализация таблиц. Нормальные формы.**

Нормализация таблиц – это разбитие таблиц на 2 или более таблиц, обладающих лучшими свойствами при удалении, добавлении, обновлении данных. Цель нормализации таблиц добиться такой структуры базы данных, которой бы каждый факт описывался один раз (реальный объект, событие или явление). Это делается не с целью экономии места, а с целью избежать так называемых аномалий данных.

*Аномалии данных*

**

1. Ошибка в названии жанра приводит к появлению нового жанра
2. Мы никогда не узнаем о том, что есть жанр «Драма» до тех пор, пока не добавим фильм этого жанра в базу
3. При удалении последнего фильма жанра «Comics», теряется информация о том, что есть такой жанр

**Нормальные формы** (**НФ**) – некоторые критерии оценивания, насколько правильно выполнилась нормализация таблицы. Для нормализации достаточно 3го уровня НФ, но нужно стремиться к 5ому уровню. Каждая НФ включает в себя предыдущую НФ.

*Нормальные формы*

1. В таблице на пересечении строки и столбца всегда находится одно значение. Согласно правилам Коддалюбая реляционная СУБД уже следит за тем, чтобы в одной ячейке было только одно значение. Значит автоматически приводи к **НФ1**.

Функциональная зависимость – точка Б в таблице функционально зависит от поля А той же таблицы, когда в любой момент времени для каждого из различных значений поля А обязательно существует одно значение поля Б.

Полная функциональная зависимость (относиться только к основным столбцам) – поле Б находится в полной функциональной зависимости от составного поля А, если оно функционально зависит от А и не зависит от любого подмножества А.

1. Если таблица удовлетворяет **НФ1** и все ее поля, не входящие в первичный ключ связаны полной функциональной зависимостью с первичным ключом.
2. Таблица находится в **НФ3**, если она находится в **НФ2** и ни одно из ее ключевых полей не зависит функционально от любого другого не ключевого поля.
3. Нужно избегать ситуаций, когда для одной записи создаются несколько столбцов для хранения информации по сущности для одного типа.
4. Означает, что мы главную таблицу разбили на множество таблиц, состоящих из двух полей: идентификатора и названия. Такие таблицы называют полной декомпозицией. Полная декомпозиция должна позволять нам собирать все таблицы воедино для получения изначальной таблицы без потерь.

После нормализации в главной таблице появляются столбцы, значения которых ссылаются на значения первичного ключа в новых таблицах. Такие столбцы называют **внешними ключами** (**Foreign key**).

**Внешний ключ** – столбец таблицы, значения которого совпадают со значениями первичного ключа в друго таблицы. Внешний ключ играет важную роль для связи между таблицами, т.к. связи между таблицами в нормализированной БД осуществляется последством пары «внешний ключ одной таблицы – первичный ключ другой таблицы».

Таблица, в которой есть внешний ключ, является **дочерней** таблицей по отношению к таблице, на первичный ключ которого ссылается внешний ключ этой таблицы.

Таблица с первичный ключом называется **родительской** таблицей.

*Виды связей между таблицами*

1. **1 : М** – один ко многим (Один жанр ко многим фильмам)
2. **М : 1** – многие к одному (Много книг одной тематики)
3. **М : М** – многие ко многим (Один автор написал множество книг. Любая из этих книг может быть написана несколькими авторами.). Если случилось так, что появилось связ **многие ко многим**, ее нужно приводить к виду **один к одному**.
4. **1 : 1** – один к одному

**Целостность данных** – это согласованность данных между таблицами.

*Правила целостности данных, которые предоставляются СУБД пользователю (согалсно 10-му правилу Кодда)*

* Правила удаления

**RESTRICT (NO ACTION)**  - запрещает удаление строки-предка, если у строки есть строки-потомки. (Строка-предок – это строка родительской таблицы, а строка-потомок – строка дочерней таблицы).

**CASCADE** – при удалении строки-предка автоматически удаляются все строки-потомки

**SET NULL** – при удалении строки-предка значения внешнего ключа в строках-потомках становится равным NULL.

**SET DEFAULT** – при удалении строки-предка значения внешнего ключа в строках-потомках получают значение по умолчанию, которое было задано для столбца внешнего ключа при создании таблицы.

* Правила обновления – относится к случаю, когда менятся значение первичного ключа в строке-предке

**RESTRICT (NO ACTION)**  - запрещает удаление строки-предка, если у строки есть строки-потомки. (Строка-предок – это строка родительской таблицы, а строка-потомок – строка дочерней таблицы).

**CASCADE** – при удалении строки-предка автоматически удаляются все строки-потомки

**SET NULL** – при удалении строки-предка значения внешнего ключа в строках-потомках становится равным NULL.

**SET DEFAULT** – при удалении строки-предка значения внешнего ключа в строках-потомках получают значение по умолчанию, которое было задано для столбца внешнего ключа при создании таблицы.

**Ключевое слово JOIN**

Предназначен для связи между таблицами предложением FROM

**Ключевое слово UNION, UNION ALL**

* В объединяемых результирующих таблицах долнжо быть одинаковое количество столбцов и типы данных в этих столбцах должны совпадать.
* UNION исключает повторяющиеся строки из объединенной результирующей таблицы

**Понятие транщакций**

Транзакция позволяет объединить несколько команд SQL одну большую составную команду. Которая выполняется\не выполняется целиком. Любая транзакция обладает свойствами, которые сокращенно называют ACID.

**A** **-** Atomiciti (Атомарность – неделимость – выполняется или не выполняется транзакция вообще)

**C –** Consistency (Связанность – означает как в случае успешного так и в случае аварийного завершения транзакции целостность в БД не нарушается)

**I –** Isolation (Изолированность – что одна транзакция не может взаимодействовать с другими транзакциями)

**D –** Durability (Надежность – что транзакция никак не зависит от внешних факторов (как выключаение света, поломка ПК и т.д.))

Транзакции различают:

* **Явные**

Начинаются при помощи команд: Begin Tran, Begin Transaction

Заканчиваются при помощи команд (подтверждение транзакции): Commit Tran, Commit Word

Отмена транзакции: RollbackTran, Rollback Transaction, Rollback Word

Begin Transaction

INSERT INTO product\_09358 (prodname, price, prodweight)

VALUES ('Pepsi-Cola', 25.00, 1000);

INSERT INTO product\_09358 (prodname, price, prodweight)

VALUES ('Fanta-Cola', 23.00, 1000);

SELECT \*

FROM product\_09358;

RollBack Tran

SELECT \*

FROM product\_09358;

* **Неявные**

Set Implici\_Transactions ON – любая команда будет считаться транзакцией и будет ждать подтверждения

Отменить неявные транзакции можно:

* **Автоматические**

**Системная переменная MS SQL @@ERROR**

DECLARE @a int;

SET @a = 0;

Begin Tran

INSERT INTO product\_09358 (prodname, price, prodweight)

VALUES ('1Pepsi-Cola12', 25.00, 1000);

--SET @a = @a + @@ERROR;

SAVE Transaction Hello;

INSERT INTO product\_09358 (prodname, price, prodweight)

VALUES ('2Fanta-Cola12', 23.00, 1000);

SET @a = @a + @@ERROR;

IF @a = 0

BEGIN

PRINT 'Подтверждение транзакции';

COMMIT Tran;

END

ELSE

Begin

ROLLBACK TRANSACTION Hello;

COMMIT Tran;

END

PRINT @a;

// Виды

CREATE VIEW IzdAndMinYear2

AS

SELECT Press.Id AS Id, MIN(Books.YearPress) AS MinYear

FROM Books, Press

WHERE Books.Id\_Press = press.ID

GROUP BY press.ID;

SELECT Press.Name, Books.Name, Books.YearPress

FROM Press, Books, IzdAndMinYear2

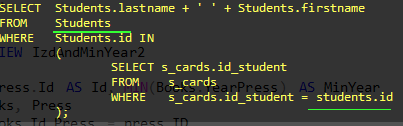
WHERE Press.Id = Books.Id\_Press

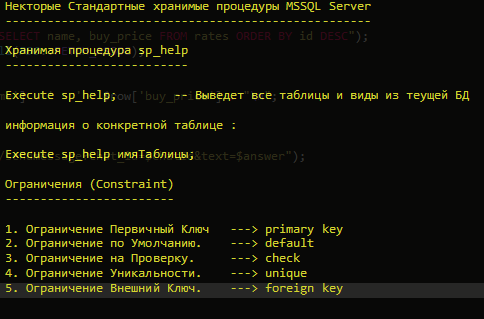
AND Press.Id = IzdAndMinYear2.Id

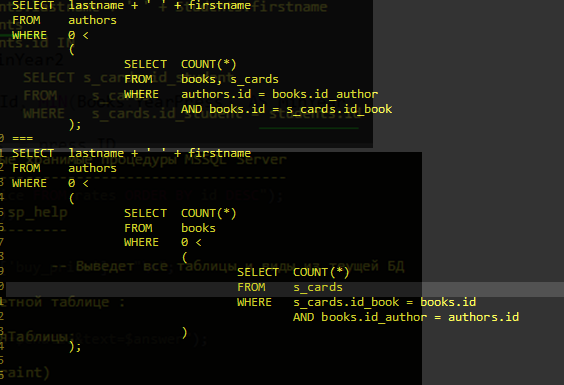
AND IzdAndMinYear2.MinYear = Books.YearPress

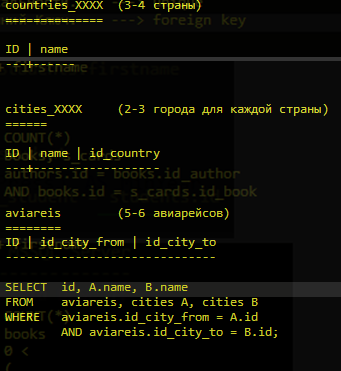
**Коррелирующие (связанные) подзапросы**

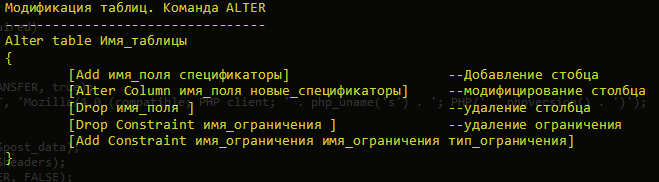
**Коррелирующие запросы** – связанный подзапрос использует таблицу верхнего запроса. Коррелирующий запрос выполняется для каждой строки верхнего запроса.











CREATE TABLE blabla\_09358 (

Id INT NOT NULL,

Name VARCHAR(64),

price numeric(8, 2),

weight INT CONSTRAINT my\_DEF\_09358 DEFAULT '50',

CONSTRAINT my\_CHECK\_09358 CHECK (price > 5.00),

CONSTRAINT my\_PK\_09358 PRIMARY KEY (Id),

CONSTRAINT my\_UNIQ\_09358 UNIQUE (Name)

);

exec sp\_help blabla\_09358

DROP TABLE blabla\_09358

INSERT INTO blabla\_09358 (id, name, price, weight) VALUES (1, 'a4ha', 8, 2)

CREATE TABLE genres\_09358

(

Id INT NOT NULL PRIMARY KEY IDENTITY(1,1),

Name VARCHAR(64)

);

CREATE TABLE films\_09358

(

Id INT NOT NULL PRIMARY KEY IDENTITY(1,1),

Name VARCHAR(128),

Year INT CHECK(Year > 1900),

Id\_Genre INT,

CONSTRAINT FK\_GENRE\_09358 FOREIGN KEY (Id\_Genre)

REFERENCES genres\_09358 (id)

ON DELETE NO ACTION

ON UPDATE NO ACTION

);

INSERT INTO genres\_09358 (name) VALUES ('Фантастика');

INSERT INTO genres\_09358 (name) VALUES ('Комедия');

INSERT INTO genres\_09358 (name) VALUES ('Драма');

INSERT INTO genres\_09358 (name) VALUES ('Блокбастер');

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 1', 1);

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 2', 2);

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 3', 4);

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 4', 4);

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 5', 3);

INSERT INTO films\_09358 (name, Id\_genre) VALUES ('Фильм 6', 1);

ALTER TABLE films\_09358 ADD rating REAL;

ALTER TABLE films\_09358 DROP COLUMN rating;

SELECT \* FROM films\_09358;

ALTER TABLE films\_09358 ADD CONSTRAINT my\_YEAR\_09358 CHECK (Year > 2000)

INSERT INTO films\_09358 (name, year, Id\_genre) VALUES ('Фильм 6', 2001, 1);

exec sp\_help films\_09358

**Хранимые процедуры**

Хранимые процедуры – это подпрограмма БД, которая доступна для вызова. В хранимые процедуры удобно помещать некий SQL-код, который часто используется.

Создание хранимой процедуры:

Create Procedure имя\_хранимой\_процедуры

AS

Запрос\_SQL

Синтаксис вызова хранимой процедуры:

Exec имя\_ХП;

Execute имя\_ХП;

Синтаксис ХП, которая принимает параметры:

Create Procedure имя\_ХП

@аргумент1 тип\_данных[=значение\_по\_умолчанию][Output]

@аргумент1 тип\_данных[=значение\_по\_умолчанию][ Output]

…

@аргументN тип\_данных[=значение\_по\_умолчанию][ Output]

AS

Запрос\_SQL

CREATE PROCEDURE BooksThemes\_09358

AS

SELECT Books.Name AS Book, Themes.Name AS Theme

FROM Books, Themes

WHERE Books.Id\_Themes= Themes.Id

ORDER BY Books.Name;

GO

execute BooksThemes\_09358

GO

CREATE PROCEDURE BooksForTheme\_09358

@themename varchar(64)

WITH ENCRYPTION

AS

SELECT Books.Name AS Book, Themes.Name AS Theme

FROM Books, Themes

WHERE Books.Id\_Themes = Themes.Id AND themes.NAME LIKE @themename

ORDER BY Books.Name;

GO

DROP PROCEDURE BooksForTheme\_09358

execute BooksForTheme\_09358 '%';

execute BooksForTheme\_09358 'П%';

execute sp\_helptext BooksForTheme\_09358

GO

CREATE PROCEDURE StudCountFromFaculty\_09358

@facultyname varchar(64),

@cnt INT OUTPUT

AS

SELECT @cnt = COUNT(\*)

FROM S\_Cards, Students, Groups, Faculties

WHERE S\_Cards.Id\_Student = Students.Id

AND Students.Id\_group = groups.id

AND Groups.id\_faculty = Faculties.Id

AND Faculties.Name LIKE @facultyname;

GO

DECLARE @result int;

EXECUTE StudCountFromFaculty\_09358 'Программирования', @result OUTPUT;

PRINT 'Количество: ' + str(@result);

--Количество книг, которые брали и студенты и преподаватели по заданной тематике

CREATE PROCEDURE CountBooksByThemes\_09358

@themename varchar(64)

AS

DECLARE @cntStud int;

DECLARE @cntteach int;

DECLARE @cnt int;

SELECT @cntStud = COUNT(\*)

FROM S\_Cards, Books, Themes

WHERE S\_Cards.Id\_Book = Books.Id

AND Books.Id\_themes = Themes.Id

AND Themes.Name LIKE @themename;

SELECT @cntteach = COUNT(\*)

FROM T\_Cards, Books, Themes

WHERE T\_Cards.Id\_Books = Books.Id

AND Themes.Name LIKE @themename;

SET @cnt = @cntStud + @cntteach;

RETURN @cnt;

GO

DECLARE @result INT;

exec @result = CountBooksByThemes\_09358 'Базы данных';

PRINT @result

WHILE (condition)

BEGIN

…

END

**Хранимые функции**

* **Скалярные –** возвращают одно значение
* **Однозапросные –** функции, которые базируются на одном запросе
* **Многозапросные –** функции, которые базируются на нескольких запросах

***Синтаксис***

Синтаксис для всех видов функций:

CREATE FUNSTION ииимя\_функции

(

@имя\_параметра1 тип\_данных [=значение\_по\_умолчанию],

@имя\_параметра2 тип\_данных [=значение\_по\_умолчанию],

…

)

RETURNS возвращаемый\_тип\_данных

[WITH ENCRYPTION]

AS

BEGIN

ТЕЛО

…

RETURN значение

END

Синтаксис вызвоа функции:

SELECT имя\_владельца\_функции.имя\_функции (параметры);

**Триггер**

Триггер – это подпрограмма БД, которая запускается по событию. К событиям относят добавления, удаление, обновление данных в таблице. Триггер ставится на определенную таблицу и на определенное событие, также есть возможность ставить событие на определенное время. В разных СУБД предлагаются разные варианты срабатывания триггеров.

Предназначение:

* Решение задач, связанные с целостностью данных, которые нельзя решить с помощью обычных правил целостности. Например: при взятии книги в библиотеке уменьшить кол-во оставшихся книг в библиотеке. И наоборот, при возврате книги увеличить количество этой книги в библиотеке. Все книги розданы – заблокировать операцию выдачи.

Синтаксис:

CREATE TRIGGER имя\_триггера

ON имя\_таблицы

FOR {AFTER | INSTEAD OF} {INSERT | UPDATE | DELETE}

[WITH ENCRYPTION]

AS Тело

**Аутентификация и авторизация в MS SQL Server**

Аутентификация - проверка возможности подключения к ресурсу

Авторизация - это проверка на то, имеет ли пользователь разрешение на использование этого ресурса

Для создания подключения через аутентификацию Windows:

Execute sp\_grantlogin ‘Домен\Логин’

Для добавления подключения при помощи SQL Server аутентификации:

Execute sp\_addlogin [@loginame=]’логин’, [@passwd=]’пароль’, [[@defdb=]’имя\_базы\_данных’]];

Для удаления подключения через SQL сервер аутентификацию:

Execute sp\_droplogin ‘имя\_учетной\_записи’

Для смены пароля:

Execute sp\_password [@old=]’старый\_пароль’, [@new=]’новый\_пароль’, [@loginame=]’логин’

Изменить БД по умолчанию:

Execute sp\_defaultdb [@loginame=]’логин’, [@defdb=]’имя\_базы\_по\_умолчанию’

**Пользователи БД**

Добавить пользователя БД:

Execute sp\_grantdbaccess [@loginame=]’имя\_учетной\_записи’ [, [@name\_in\_db=]’имя\_пользователя\_в\_БД’]

Например:

Execute sp\_grantdbaccess @loginame=’TERMINATOR’, @passwd=’123456’, @defdb=’sp2822’;

USE sp2822;

EXECUTE SP\_GRANTDBACCESS @loginame=’TERMINATOR’;

‘имя\_учетной\_записи’ – это логин, под которым пользователь подключается к SQL Server

Для получения списка всех пользователей текущей БД:

EXECUTE SP\_HELPUSER;

EXECUTE SP\_HELPUSER [[@name\_in\_db=]’имя\_пользователя\_в\_БД’]

Тот, кто создал БД является ее владельцем: dbo(data base owner)

Для смены владельца:

EXECUTE SP\_CHANGEDBOWNER [@loginame=]’имя\_нового\_владельца’

Удалить пользователя БД:

EXECUTE SP\_REVOKEDBACCESS [@name\_in\_db]’имя\_пользователя\_в\_БД’

**Использование ролей**

- Роли уровня сервера

Sysadmin – является собирательной ролью (включает в себя все нижние)

Serveradmin – пользователи, которые являются администраторами, но не имеют доступа БД (управление сервером)

Setupadmin – могут определять хранимые процедуры, которые будут запускаться при старте сервера

Securityadmin – пользователи этой группы управляют учетными записями, а также определяет права доступа к БД и могут изменять пароли пользователей, кроме сисадминов

Processadmin – роль опеделяет возможность следить за процессами, которые происходят в SQL Server, есть возможность удалять зависшие запросы

Dbcreator – пользователи с этой ролью могут создавать БД и являются их владельцами, также пользователи этой группы могут выполнять резервное копирование, а также восстанавливать данные из резервного копирования

Bulkadmin – пользователи этой группы имеют возможность выполнять массовую вставку данных

Diskadmin – имеет права на управление файлами, подключение устройств для резервного копирования

- Роли уровня базы данных

Db\_owner – владелец БД

Db\_accessadmin – управляет учетными записями БД

Db\_securityadmin – роль дает права на администрировании защиты БД. Может создавать роли, назначать юзеров в эту роль, настраивать доступ к этой роли

Db\_ddladmin – могут создавать, удалять, модифицировать объекты БД

Db\_backupoperator – могут производить резервное копирование этой БД

Db\_datareader – могут считывать данные из другой таблицы БД, а также из другого вида или функции

Db\_datawriter – могут добавлять, удалять и обновлять данные в текущей БД

Db\_denydatareader – запрещает чтение данных

Db\_denydatawriter – запрещает запись данных

Для добавления пользоваетеля в роль:

EXECUTE sp\_addsrvrolemember [@loginame=]'логин\_пользователя’, [@rolename=]’имя\_роли\_уровня\_сервера’

Для удаления пользователя из роли:

EXECUTE sp\_dropsrvrolemember [@loginame=]’логин\_пользоваеля’, [@rolename=]’имя\_роли\_уровня\_сервера’

