### Лекция 10

1. Запросы DDL
2. Запросы DML

### Запросы DDL (Data Definition Language)

Создание таблиц в БД происходит с использованием команды **CREATE TABLE.**

Простейший синтаксис следующий:

CREATE TABLE Таблица (Поле1 тип1 (размер1), …)

Возможные ограничения:

**- NULL (NOT NULL)** – условие, что значение в поле может или не может принимать NULL-значения.

**- UNIQUE** (можно задать для набора полей) – условие, что значение в поле уникально.

CREATE TABLE T (id1 int NOT NULL, id2 int UNIQUE) – в поле id1 нельзя вставлять NULL-значения, значение в id2 – уникально (NULL вставлять можно, но только однократно).

CREATE TABLE T (id1 int, id2 int, UNIQUE (id1, id2)) – уникальность комбинации полей id1-id2.

**- CHECK** – ограничение на значение.

CREATE TABLE T (id int CHECK (id > 5))

Ограничение означает, что в таблицу нельзя вставлять значения id <= 5.

Пример ограничения на несколько столбцов:

CREATE TABLE T (id1 int, id2 int, CONSTRAINT val CHECK (id1 > 5 OR id2 < 10))

**- Ключи:**

На одно поле:

CREATE TABLE T (id int primary key, id2 int, …)

На набор полей

CREATE TABLE T (id1 int, id2 int, id3 int, primary key (id1, id2))

- Дефолтные значения полей:

Синтаксис в MS SQL Server:

CREATE TABLE T (id1 int, id2 int CONSTRAINT id\_value DEFAULT (15))

По умолчанию, в поле id2 при вставке данных будет вставляться значение 15, если не указано иное.

- Так же есть возможность заполнять поля в таблицах на основе вставляемых данных:

CREATE TABLE T (id1 float, id2 float, id\_avg as (id1 + id2)/2)

**- Внешние ключи**

Нужны для поддержания целостности данных. Например, если покупатель не создан еще в таблице Покупатели, то такой покупатель не должен появляться в таблице Документы. И наоборот, если Покупатель используется в таблице Документы, то в зависимости от прописанных действий на удаление, строки с покупателем либо удалятся во всех таблицах, либо будет сообщение о невозможности удаления, либо покупатель в связанной таблице примет NULL значение. Аналогично при обновлении.

Это ограничение определяется на одном или более атрибутах в одной таблице (referencing table) и указывает на первичный ключ или ограничение типа unique в другой (referenced table). Обратите внимание, что ссылаться можно и на саму таблицу.

Синтаксис:

CREATE TABLE Покупатели (Покупатель\_ID int primary key)

CREATE TABLE Документы (ндок int primary key, Покупатель\_ID int REFERENCES Покупатели(Покупатель\_ID))

Или:

CREATE TABLE Документы (ндок int primary key, Покупатель\_ID int, FOREIGN KEY (Покупатель\_ID) REFERENCES Покупатели(Покупатель\_ID))

Это примеры только NO ACTION внешних ключей. Посмотрите расширенный синтаксис для назначения действий в MSDN.

**Удаление таблицы** осуществляется с помощью команды **DROP TABLE.**

DROP TABLE Таблица

Для проверки существования объекта в БД используется функция Object\_id

IF OBJECT\_ID('Покупатели', 'U') IS NOT NULL

DROP TABLE Покупатели;

Object\_id принимает на вход имя и тип объекта и возвращает внутренний идентификатор объекта, если объект с таким именем и типом существует, иначе возвращает NULL. В данном случае, если функция что-то вернула, то значит объект уже существует, и мы принимаем решение его удалить.

Если таблица уже была создана, то для изменения структуры таблицы используется команда **ALTER TABLE.**

ALTER TABLE имя таблицы ADD COLUMN имя поля тип поля (размер)

ALTER TABLE имя таблицы DROP COLUMN имя поля

Добавлять ограничения тоже можно после создания таблицы при помощи ALTER TABLE.

Пример создания ключа. Проверьте, что будет, если для поля Покупатель\_ID не прописать ограничение NOT NULL.

CREATE TABLE Покупатели (Покупатель\_ID int NOT NULL)

INSERT INTO Покупатели VALUES (1), (2)

ALTER TABLE Покупатели

ADD CONSTRAINT PK\_id

PRIMARY KEY (Покупатель\_ID)

DROP TABLE Покупатели

Пример создания внешнего ключа:

CREATE TABLE Сотрудники (empl\_id int primary key, boss\_id int, Surname nvarchar(50))

INSERT INTO Сотрудники

VALUES (1, NULL, 'Первый'), (2, 1, 'Второй')

ALTER TABLE Сотрудники

ADD CONSTRAINT FK\_boss\_empl FOREIGN KEY(boss\_id) REFERENCES Сотрудники(empl\_id);

DROP TABLE Сотрудники

Обратите внимание на то, что в поле boss\_id могут быть NULL значения, хотя в поле empl\_id не имеет NULL значений.

**Индексы:**

Операции поиска-выборки данных из таблиц по значениям их полей могут быть существенно ускорены путем использования индексации данных. Индекс содержит упорядоченный (в алфавитном или числовом порядке) список содержимого проиндексированных столбцов (или группы столбцов) с идентификаторами этих строк.

Определение индекса: *Индекс* - это упорядоченный (буквенный или числовой) список столбцов или групп столбцов в таблице.

Однако при использовании индексов следует иметь в виду, что управление индексом существенно замедляет время выполнения операций, связанных с обновлением данных, так как эти операции требуют перестройки индекса.

Индексы можно создавать как по одному, так и по множеству полей.

CREATE INDEX имя индекса ON имя таблицы (имя столбца1, имя столбца2…)

Можно создавать индекс, который обладает свойством уникальности, то есть значения полей, входящих в индекс – уникальны:

CREATE UNIQUE INDEX имя индекса ON имя таблицы (имя столбца1, имя столбца2…)

Удаление индекса происходит так:

DROP INDEX имя индекса

Как явно пользоваться индексом в запросе:

CREATE TABLE #T (id int, val float)

CREATE INDEX IX\_id ON #T (id)

SELECT \* FROM #T **WITH(INDEX = IX\_id)**

DROP TABLE #T

**Временные таблицы (для MS SQL):**

Пример 1:

DECLARE @Покупатели TABLE (Покупатель\_ID int primary key)

INSERT INTO @Покупатели

SELECT TOP 5 Покупатель\_ID

FROM Покупатели

ORDER BY Название asc

SELECT \*

FROM Документы Док INNER JOIN @Покупатели Покупатели ON

Док.Покупатель\_ID = Покупатели.Покупатель\_ID

Пример 2:

CREATE TABLE #Покупатели (Покупатель\_ID int primary key)

INSERT INTO #Покупатели

SELECT TOP 5 Покупатель\_ID

FROM Покупатели

ORDER BY Название asc

SELECT \*

FROM Документы Док INNER JOIN #Покупатели Покупатели ON

Док.Покупатель\_ID = Покупатели.Покупатель\_ID

DROP TABLE #Покупатели

**Запросы DML (Data Manipulation Language)**

Для изменения данных в таблицах применяются следующие команды:

INSERT (Добавление)

UPDATE (Обновление)

DELETE (Удаление)

Команда **INSERT** производит вставку в таблицу указанных строк.

Если нужно вставить одну строку, определяемую пользователем, в уже существующую таблицу, то синтаксис INSERT следующий:

INSERT INTO Таблица

VALUES (Значение1, Значение2, Значение3, ….)

Значения попадают в таблицу в порядке перечисления, то есть первое попадет в первый столбец и т.д. Кроме того, необходимо учитывать совпадение типов данных по вставляемым полям.

Если значения требуется вставить не во все столбцы, и не в том порядке, как они идут в таблице, то в таком случае запрос будет выглядеть так:

INSERT INTO Таблица (Поле2, Поле5, Поле4)

VALUES (Значение1, Значение2, Значение3)

В этом случае, полям, не указанным в списке полей, в которые производится вставка, автоматические присваивается значение NULL (либо какое либо значение по умолчанию, указанное при описании таблицы).

Если возникает необходимость вставить не одну строку, а некоторое множество строк, получаемых, например, с помощью запроса, то синтаксис INSERT такой:

INSERT INTO Таблица

SELECT \*

FROM Таблица2

WHERE ….

или с явным указанием вносимых полей

INSERT INTO Таблица (Поле1, Поле2, Поле3, …

SELECT Значение1, Значение2, Значение3, …

FROM Таблица2

WHERE ….

Значения NULL можно вставить и непосредственно вместо пропускаемого поля:

INSERT INTO Таблица

VALUES (Значение1, NULL, Значение3, …)

или

INSERT INTO Таблица

SELECT Значение1, NULL, Значение3, ….

FROM Таблица2

При любой вставке данных стоит учитывать ограничения, наложенные на таблицу: ключи, уникальность, NULL значения, CHECK ограничения и т.д.

Если нужно вставить несколько строчек VALUES, то достаточно написать (MS SQL с *2005?*)

INSERT INTO Таблица

VALUES (Значение1, Значение2), (Значение3, Значение4) и тд через запятую.

CREATE TABLE #T (id1 float, id2 float CONSTRAINT id\_value DEFAULT (15), id\_avg as (id1 + id2)/2)

INSERT INTO #T (id1, id2) VALUES (1, 2)

INSERT INTO #T VALUES (3, 4)

INSERT INTO #T (id1) VALUES (2)

INSERT INTO #T (id2) VALUES (2)

SELECT \* FROM #T

DROP TABLE #T

|  |  |  |
| --- | --- | --- |
| **id1** | **id2** | **id\_avg** |
| 1 | 2 | 1.5 |
| 3 | 4 | 3.5 |
| 2 | 15 | 8.5 |
| NULL | 2 | NULL |

Команда **DELETE** производит удаление строк из таблицы.

Удаление всех строк из таблицы происходит с помощью запроса со следующим синтаксисом:

DELETE FROM Таблица

При написании запросов на SQL – Server использование DELETE \* (при удалении всех записей из таблицы) нежелательно, так как DELETE при удалении сохраняет каждую удаляемую строку в лог. В этом случае предпочтительнее использование команды TRUNCATE TABLE

TRUNCATE TABLE *ИмяТаблицы,*

так как она производит удаление практически без сохранения лога и немедленно освобождает место, которое было занято удаляемыми записями (методом удалить, создать).

Для удаления нескольких строк, удовлетворяющих определенному условию:

DELETE FROM Таблица

WHERE Поле1 = Значение1

В условии WHERE можно использовать подзапросы.

DELETE FROM Таблица

WHERE Поле1 IN (SELECT Поле3 FROM Таблица2 …)

**Access:**

В предложении FROM подзапроса нельзя ссылаться на таблицу, из которой происходит удаление, но можно ссылаться на строку - текущего кандидата на удаление.

Как указано в Help в Access, инструкцию DELETE можно использовать для удаления записей из таблиц, связанных отношением «один-ко-многим» с другими таблицами.

DELETE Документы.\*

FROM Документы INNER JOIN Покупатели ON

Покупатели.Покупатель\_ID = Документы. Покупатель\_ID

WHERE Покупатели.Наименование like “\*Иванов\*”

Здесь: В таблице Покупатели должен быть заведен ключ «Покупатель\_ID. Таким образом, будет реализована связь один-ко-многим.

Синтаксис DELETE на **MS SQL** сервер описывается следующим примером:

DELETE FROM Документы

FROM Документы INNER JOIN Покупатели ON

Покупатели.Покупатели\_ID = Документы.Код\_клиента

WHERE Покупатели.Наименование like “\*Иванов\*”

Команда **UPDATE** производит обновление указанных данных.

Как уже заметили, команды DELETE и INSERT удаляют и вставляют строки целиком. Если же нужно изменить (удалить) значение только в некоторых полях нужной строки, то в таком случае следует пользоваться командой UPDATE.

Синтаксис UPDATE такой:

UPDATE Таблица

SET Поле1 = Значение1;

Выражение после команды SET может быть, например, таким:

Поле1 = Поле1 + Значение1 (Поле2\*Значение1 и т.д.).

Для удаления значения в определенном поле можно выполнить такой запрос:

UPDATE Таблица

SET Поле1 =NULL;

Также можно присваивать новое значение не всем строкам таблицы, а определенным по нужному пользователю условию.

UPDATE Таблица

SET Поле1 = Значение1

WHERE Поле2 = Значение2;

Кроме того, модифицировать можно не одно поле за раз, а несколько. Поля в таком случае перечисляются через «,».

UPDATE Таблица

SET Поле1 = Значение1, Поле2 = Значение2, Поле3 = Значение3

Подумайте и проверьте, что будет в результате выполнения запроса. В таблице T есть 2 поля id1 и id2.

UPDATE T

SET id1 = id2, id2 = id1

Что делать в случае, если данные необходимо изменить на основании данных (или связей) с другими таблицами?

**Access:**

(пусть в таблице Документы есть поле Оплата, проставим его равным 0, если покупатель носит фамилию Иванов).

UPDATE Документы INNER JOIN Покупатели ON

Документы.Покупатель\_ID = Покупатели.Покупатель\_ID

SET Документы.Оплата = 0

WHERE Покупатели.Название like “\*Иванов\*”

MS SQL:

UPDATE Документы

SET Оплата = 0

FROM Документы INNER JOIN Покупатели ON

Документы.Покупатель\_ID = Покупатели.Покупатель\_ID

WHERE Покупатели.Название like “\*Иванов\*”

Пример 2:

У Вас есть таблицы

|  |  |  |
| --- | --- | --- |
| Документы |  | Данные |
| **ндок** |  | **ндок** |
| Сумма |  | **Товар\_ID** |
|  |  | Колво |
|  |  | Цена |

Поле «Сумма» во всех документах на текущий момент равно 0. Нужно проставить его верно. То есть у Вас следующие данные:

|  |  |
| --- | --- |
| ндок | Сумма |
| 1 | 0 |
| 2 | 0 |
| ндок | Товар\_ID | | Колво | Цена |
| 1 | 1 | | 1 | 2 |
| 1 | 2 | | 2 | 1 |
| 2 | 3 | | 3 | 4 |

По данным в таблице Данные нужно проставить поле «Сумма». То есть для первого документа поле сумма = 1\*2 + 2\*1 = 4, а для второго документа Сумма = 3\*4 = 12.

Синтаксис в Access:

UPDATE Документы INNER JOIN Данные ON Документы.ндок = Данные.ндок

SET Документы.Сумма = Документы.Сумма + Данные.Колво\*Цена;

Синтаксис в MS SQL:

Подобный же синтаксис не будет верно работать в MS SQL:

Проверьте, что будет, если Вы напишете запрос к указанным данным следующим образом?

UPDATE Документы

SET Документы.Сумма = Документы.Сумма + Данные.Колво\*Цена

FROM Документы INNER JOIN Данные ON Документы.ндок = Данные.ндок

Правильным способом здесь будет - использование подзапроса.

UPDATE Документы

SET Документы.Сумма = Данные.Сумма

FROM Документы INNER JOIN

(

SELECT ндок, SUM(Колво\*Цена) as Сумма

FROM Данные

GROUP BY ндок

) ON Документы.ндок = Данные.ндок