### Лекция 6.

План**:**

1. Соединение таблиц: INNER JOIN.
2. Запросы на манипулирование данными

Типы связей между таблицами (было на 1-м занятии)

**Один-к-одному** – каждой строке одной таблицы соответствует одна строка (или ни одной) второй таблицы. С другой стороны, каждая строка второй таблицы должна быть связана только с одной строкой первой таблицы.

Пример: таблица «Товар» (поле Товар\_ID – первичный ключ) и таблица «Склад», в которой, например, хранится информация о текущем остатке на складе (поле товар – первичный ключ).

**Один-ко-многим** – каждой строке первой таблицы соответствует ноль, одна или более строк второй таблицы. С другой стороны, каждая строка второй таблицы должна быть связана только с одной строкой первой таблицы.

Пример: таблица «Товары» и таблица «Документы\_данные». Частным случаем связи «один-ко-многим» является связь «один-к-одному».

**Многие-ко-многим** – любой строке первой таблицы может соответствовать ноль, одна или несколько строк второй таблицы. При этом каждая строка второй таблицы может быть связана с одной или более строк первой таблицы. Связь многие-ко-многим реализуется с помощью связывающей таблицы.

Пример: таблица «Документы» и таблица «Товары» по полям товар-ндок через промежуточную таблицу «Документы\_данные», или таблица «Товары» и таблица «Покупатели» через промежуточные таблицы «Документы» и «Документы\_данные» по полям «товар» и «покупатель».

Примерный набор заголовков для базы данных «Продажи»:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Документы** |  | **Документы\_данные** |  | **Товары** |  | **Покупатели** |
| **ндок** |  | **ндок** |  | **Товар\_ID** |  | **Покупатель\_ID** |
| Дата |  | **Товар\_ID** |  | Наименование |  | ФИО |
| Покупатель\_ID |  | Цена |  | Цена |  |  |
|  |  | Колво |  | Остаток |  |  |
|  |  |  |  | Объем |  |  |
|  |  |  |  | Масса |  |  |

Какие недостатки у хранения данных в разных таблицах? Может оказаться так, что из одной таблицы данные удалили (например, удалили строки из «Документы\_данные», которые соответствуют закупке товаров по документу с ндок = 5, но информация о покупке сохранилась в таблице «Документы», если же в таблице «Документы» еще есть поле «Сумма», в котором есть информация о суммарной стоимости покупки, то при подсчете денег в кассе может возникнуть проблема: данные по базе, посчитанные разными способами будет отличаться). Поэтому можно поддерживать ссылочную целостность.

**Ссылочная целостность –** некоторая система правил, используемых в Access/MS SQL для поддержания связей между записями в связанных таблицах, а также обеспечивающих защиту от случайного удаления или изменения связанных данных. Подробнее, когда можно установить целостность данных, можно прочитать в Help для Access/SQL Express (в файле с определениями есть инструкция по созданию схемы данных).

Ссылочная целостность позволяет, например, исключить случаи, когда в документах появляется товар, которого нет в номенклатуре, позволяет при удалении документа удалять и все его данные, при обновлении внешнего ключа каскадно обновлять его во всех таблицах и тд.

**Задание 1:**

В приложении к архиву есть краткая инструкция, как создавать схему данных в Access или MS SQL. Если это Вам интересно (включая поведение таблиц в случае создания разных видов ссылочной целостности).

### Типы соединения таблиц:

**Операции реляционной алгебры:**

1. Объединение, пересечение, разность и декартово произведение
2. Выборка, проекция, соединение, деление

**Выборка:** возвращает отношение, содержащее кортежи из заданного отношения, причем данные удовлетворяют указанным условиям. (SELECT \* WHERE)

**Проекция:** Возвращает отношение, содержащее все кортежи (подкортежи) заданного отношения, которые остались в этом отношении после исключения из него некоторых атрибутов. (SELECT Поле1)

**Произведение:** Возвращает отношение, содержащее все возможные кортежи, которые являются сочетанием двух кортежей, принадлежащих соответственно двум заданным отношениям. (SELECT \* FROM A,B – декартово произведение)

Объединение: **Возвращает отношение, содержащее все кортежи, которые принадлежат либо одному из двух заданных отношений, либо им обоим. (UNION)**

**Пересечение:** Возвращает отношение, содержащее все кортежи, которые принадлежат одновременно двум заданным отношениям.

**Разность:** Возвращает отношение, содержащее все кортежи, которые принадлежат первому из двух заданных отношений и не принадлежат второму.

**Соединение:** Возвращает отношение, содержащее все возможные кортежи, которые представляют собой комбинацию атрибутов двух кортежей, принадлежащих двум заданным отношениям, при условии, что в этих двух комбинируемых кортежах присутствуют одинаковые значения в одном или нескольких общих для исходных отношений атрибутах (причем эти общие значения в результирующем кортеже появляются один раз, а не дважды). (SELECT FROM INNER JOIN)

**Деление:** Для заданных двух унарных отношений и одного бинарного возвращает отношение, содержащее все кортежи из первого унарного отношения, которые содержатся также в бинарном отношении и соответствуют всем кортежам во втором унарном отношении.

**Реализация в SQL:**

***Замечание:***

В запросах обращаться можно не только к таблицам, но и другим запросам на выборку данных. Обращение происходит точно также.

Что делать, если нужно использовать данные из нескольких таблиц? Таблицы нужно соединять, объединять и тд.

**1. Операции соединения. Типы соединений:**

Соединение таблиц в MS Access производится с помощью операций: JOIN (декартово произведение, CROSS JOIN, «,»), INNER JOIN, LEFT JOIN, RIGHT JOIN (а также FULL JOIN в MS SQL Server).

**INNER JOIN:**

Синтаксис:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A INNER JOIN B ON A.поле = B.поле

С помощью такого запроса из декартового произведения кортежей множества A и B в результат выборки попадут только те, у которых значения в указанных кортежах совпадают.

В принципе, результат работы INNER JOIN аналогичен работе JOIN с WHERE:

SELECT A.\*, B.\* (указывать атрибуты для вывода)

FROM A, B

WHERE A.поле = B.поле

Но, во-первых, читаемость запроса, в котором соединено много таблиц, ухудшается, во-вторых, в таком синтаксисе нельзя использовать различные хинты для ускорения запросов (о различных хинтах можно почитать в MSDN по ключевому слову «hints»).

Пример:

Две таблицы:

Документы и Покупатели:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** |  | **ID** | **Покупатель** |
| 1 | 10 |  | 10 | Иванов |
| 2 | 11 |  | 15 | Сидоров |

Результат выполнения запроса

SELECT \*

FROM Документы, Покупатели

Будет

|  |  |  |  |
| --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** | **ID** | **Покупатели** |
| 1 | 10 | 10 | Иванов |
| 2 | 11 | 10 | Иванов |
| 1 | 10 | 15 | Сидоров |
| 2 | 11 | 15 | Сидоров |

SELECT \*

FROM Документы INNER JOIN Покупатели ON

Документы.Покупатель\_ID = Покупатель.ID

|  |  |  |  |
| --- | --- | --- | --- |
| **ндок** | **Покупатель\_ID** | **ID** | **Покупатель** |
| 1 | 10 | 10 | Иванов |

Примечание:

1. Соединять можно не только 2 таблицы, но и больше, описывая соединение по нужным полям. Пример для 3-х:

SELECT A.\*, B.\*, C.\*

FROM A

INNER JOIN B ON

A.поле1 = B.поле1 AND

A.поле2 = B.поле2

INNER JOIN C ON

A.поле3 = C.поле3

WHERE C.поле4 = 1234

1. Соединение можно проводить не только по условию равенства, но и по условию <, >, <> и тд. Кроме того, можно проводить различные операции со сравниваемыми значениями:

INNER JOIN B ON A.поле1 = B.поле1 + 1 (если поле1 - числовое )

1. Соединение по нескольким полям также необязательно проводить только с помощью AND, можно использовать, например, OR.

**Что делать, если необходимо соединить таблицу саму с собой?**

Для таких целей используются псевдонимы (или алиасы). Кроме того, псевдонимы можно использовать и в том случае, если название таблицы, из которой Вы берете данные, слишком длинное или сложное для написания. После создания для таблицы псевдонима, в SELECT, WHERE к полям этой таблицы следует обращаться через псевдоним.

SELECT NewNameA.\*, NewNameB.\* (указывать атрибуты для вывода)

FROM A as NewNameA INNER JOIN B as NewNameB ON NewNameA.поле = NewNameB.поле

WHERE NewNameA.поле2 = 2345

|  |  |  |
| --- | --- | --- |
| **Документы** |  | **Документы\_данные** |
| **ндок** |  | **ндок** |
| Дата |  | **Товар\_ID** |
| Покупатель\_ID |  | Колво |
| Сумма |  | Цена |

**Данные в таблицах следующие:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ндок | Дата | Покупатель\_ID | Сумма |  | ндок | Товар\_ID | Колво | Цена |
| 1 | 01.11.15 | 1 | 170 |  | 1 | 1 | 3 | 20 |
| 2 | 02.11.15 | 2 | 235 |  | 1 | 2 | 2 | 55 |
| 3 | 02.11.15 | 2 | 50 |  | 2 | 3 | 3 | 25 |
|  |  |  |  |  | 2 | 4 | 4 | 40 |
|  |  |  |  |  | 3 | 2 | 1 | 50 |

**Задача 3:**

Что выведется в результате выполнения запроса:

SELECT Документы.ндок, SUM(Сумма)

FROM Документы INNER JOIN Документы\_данные as Данные ON

Документы.ндок = Данные.ндок

GROUP BY Документы.ндок

Запустите запрос на этих данных, подумайте, понимаете ли Вы этот результат?

Чтобы стало совсем понятно, почему именно такой ответ, то подумайте, группировка и суммирование работают на результатах виртуальной таблицы, полученной в результате соединения таблиц Документы и Документы\_данные.

Запустите запрос

SELECT \*

FROM Документы INNER JOIN Документы\_данные as Данные ON

Документы.ндок = Данные.ндок

Сколько строк в результате? Сколько раз встречается каждый номер документа и сумма из таблицы Документы?

**Задача 4:**

У Вас есть таблица: Обороты: **Покупатель**, Сумма. Нужно для каждого покупателя вывести его рейтинг. Покупатель с максимальной суммой имеет максимальный рейтинг и тд. Как написать такой запрос?

|  |  |
| --- | --- |
| **Покупатель\_ID** | **Сумма** |
| 1 | 100 |
| 2 | 50 |
| 3 | 40 |
| 4 | 30 |

SELECT T1.Покупатель\_ID, T1.Сумма, COUNT(T2.Покупатель\_ID) Рейтинг

FROM Обороты T1 INNER JOIN Обороты T2 ON

T1.Сумма <= T2.Сумма

GROUP BY T1.Покупатель\_ID, T1.Сумма

Подумайте, что выведет запрос, если покупатель 3 тоже закупился на 50 уе?

Чуть позже разберем специальные функции, которые позволяют нумеровать строки. Пока наша цель - научиться решать такие задачи простейшими доступными средствами.

### Запросы DDL (Data Definition Language)

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

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))

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

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

- 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 будет вставляться значение 15, если не указано иное.

- Так же есть возможность заполнять поля в таблицах на основе вставленных 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 удаляться во всех таблицах, либо будет сообщение о невозможности удаления, либо покупатель в связанной таблице примет NULL значение. Аналогично при обновлении.

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))

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

DROP TABLE Таблица

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

**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

**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 тд. 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 тд.

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.

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

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

DELETE \* *(\* - не указывается для запросов на SQL Server)*

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 без сохранения лога и немедленно освобождает место, которое было занято удаляемыми записями.

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 тд).

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