Инструкции языка обработки данных (Инструкции DML)

Инструкции DML предназначены для добавления данных, изменения данных, запроса данных и удаления данных из базы данных. К числу основных инструкций языка DML относятся: SELECT, INSERT, UPDATE, DELETE, MERGE, BULK INSERT.

Извлечение данных: инструкция SELECT

Инструкция SELECT извлекает строки из базы данных и позволяет делать выборку одной или нескольких строк или столбцов из одной или нескольких таблиц. Полный синтаксис инструкции SELECT сложен, однако основные предложения можно вкратце описать следующим образом:

[WITH *общее_табличное_выражение*]
SELECT [DISTINCT | ALL] [TOP *выражение* [PERCENT]] { * | *список_выбора* }
[INTO *новая_таблица*]
[FROM *список_табличных_источников*]
[WHERE *условие_поиска*]
[GROUP BY *group_by_выражение*]
[HAVING *условие_поиска*]
[ORDER BY *order_by_выражение* [ASC | DESC]]

Порядок предложений в инструкции SELECT имеет значение. Любое из необязательных предложений может быть опущено; но если необязательные предложения используются, они должны следовать в определенном порядке. При обработке инструкции SELECT составляющие ее предложения выполняются в следующем порядке:

1. FROM
2. ON
3. JOIN
4. WHERE
5. GROUP BY
6. HAVING
7. SELECT
8. DISTINCT
9. ORDER BY
10. TOP

Предложение FROM указывает таблицу, представление или источник производной таблицы с указанием или без указания псевдонима для использования в инструкции SQL. В инструкции можно использовать до 256 источников таблиц, хотя предел изменяется в зависимости от доступной памяти и сложности других выражений в запросе. Отдельные запросы могут не поддерживать 256 источников таблиц. В качестве источника таблицы может быть указана переменная table.

Примечание. Производительность выполнения запросов может снизиться из-за большого количества таблиц, указанных в запросе. На время компиляции и оптимизации также влияют дополнительные факторы. Они включают в себя наличие индексов и индексированных представлений в каждом табличном источнике и размер списка выбора в инструкции SELECT.

Важным случаем табличного источника является *joined_таблица* - результирующий набор, полученный из двух или более таблиц. Для множественных соединений следует использовать скобки, чтобы изменить естественный порядок соединений.

joined_таблица ::=
левая_таблица CROSS JOIN *правая_таблица* |
левая_таблица [NATURAL] [INNER | { LEFT | RIGHT | FULL [OUTER] } | UNION] JOIN *правая_таблица* [ON *условное_выражение* |
(*joined_таблица*)

CROSS JOIN
: Указывает произведение двух таблиц. Возвращает те же строки, что и соединение без предложения WHERE в старом варианте SQL-92.

JOIN
: Указывает, что данная операция соединения должна произойти между указанными источниками или представлениями таблицы.

INNER

Указывает, что возвращаются все совпадающие пары строк. Несовпадающие строки из обеих таблиц отбрасываются. Если тип соединения не указан, этот тип задается по умолчанию.

FULL [OUTER]

Указывает, что в результирующий набор включаются строки как из левой, так и из правой таблицы, несоответствующие условиям соединения, а выходные столбцы, соответствующие оставшейся таблице, устанавливаются в значение NULL. Этим дополняются все строки, обычно возвращаемые при помощи INNER JOIN.

LEFT [OUTER]

Указывает, что все строки из левой таблицы, не соответствующие условиям соединения, включаются в результирующий набор, а выходные столбцы из оставшейся таблицы устанавливаются в значение NULL в дополнение ко всем строкам, возвращаемым внутренним соединением.

RIGHT [OUTER]

Указывает, что все строки из правой таблицы, не соответствующие условиям соединения, включаются в результирующий набор, а выходные столбцы, соответствующие оставшейся таблице, устанавливаются в значение NULL в дополнение ко всем строкам, возвращаемым внутренним соединением.

ON *условие_поиска*

Задает условие, на котором основывается соединение. Условие может указывать любой предикат, хотя чаще используются столбцы и операторы сравнения, например.

Предложение WHERE определяет условия поиска строк, возвращаемых запросом. Количество предикатов, которое может содержать условие поиска, неограниченно. Дополнительные сведения об условиях поиска и предикатах см. в разделе 5. «Скалярные выражения».

Предложение GROUP BY группирует выбранный набор строк для получения набора сводных строк по значениям одного или нескольких столбцов или выражений. Дополнительные сведения о предложении GROUP BY см. в разделе 14. «GROUP BY запросы».

Предложение HAVING определяет условие поиска для группы. Дополнительные сведения о предложении HAVING см. в разделе 14. «GROUP BY запросы».

Выражение SELECT указывает столбцы, возвращаемые запросом. Выражение SELECT может содержать следующие аргументы:

ALL

Указывает, что в результирующем наборе могут появиться повторяющиеся строки. ALL является значением по умолчанию.

DISTINCT

Указывает, что в результирующем наборе могут появиться только уникальные строки. Значения NULL считаются равными для ключевого слова DISTINCT.

TOP (*выражение*) [PERCENT]

Указывает на то, что только заданное число или процент строк будет возвращен из результирующего набора запроса. Аргумент *выражение* может быть либо числом, либо процентом строк. В целях обратной совместимости использование TOP *выражение* без скобок в инструкциях SELECT поддерживается, но не рекомендуется.

список_выбора

Столбцы, выбираемые для результирующего набора. Список выбора представляет собой серию выражений, отделяемых запятыми. Максимальное число выражений, которое можно задать в списке выбора — 4 096.

элемент_выбора ::=

скалярное_выражение [[AS] *имя_столбца*] | *имя_таблицы* . *

В целях избежания неоднозначности ссылок, которые могут возникнуть, если в двух таблицах из предложения FROM содержатся столбцы с одинаковыми именами, следует указывать квалификатор для аргумента *имя_столбца*. Например таблицы SalesOrderHeader и SalesOrderDetail в базе данных AdventureWorks содержат столбцы с именем ModifiedDate. Если в запросе соединяются две таблицы, то данные о дате изменения из таблицы SalesOrderDetail могут быть заданы в списке выбора как SalesOrderDetail.ModifiedDate. Выражение в списке выбора может быть константой, функцией, любым сочетанием имен столбцов, констант и функций, соединенных оператором (операторами) или вложенным запросом.

*

Указывает на то, что все столбцы из всех таблиц и представлений в предложении FROM должны быть возвращены. Столбцы возвращаются таблицей или представлением, как указано в предложении FROM, и в порядке, в котором они находятся в таблице или представлении.

Предложение INTO в инструкции SELECT создает новую таблицу в файловой группе по умолчанию и вставляет в нее результирующие строки из запроса.

Предложение ORDER BY указывает порядок сортировки для столбцов, возвращаемых инструкцией SELECT. Предложение ORDER BY не может применяться в представлениях, встроенных функциях, производных таблицах и вложенных запросах, если не указано предложение TOP. Предложение ORDER BY может содержать следующие аргументы:

order_by_выражение

Указывает столбец, по которому должна выполняться сортировка. Столбец сортировки может быть указан с помощью имени или псевдонима столбца или неотрицательного целого числа, представляющего позицию имени или псевдонима в списке выбора. Имена и псевдонимы столбцов могут быть дополнены именем таблицы или представления. В SQL Server уточненные имена и псевдонимы столбцов связываются со столбцами, перечисленными в предложении FROM. Если в выражении *order_by_выражение* отсутствует квалификатор, то значение должно быть уникальным во всех столбцах, перечисленных в инструкции SELECT. Можно указать несколько столбцов сортировки. Последовательность столбцов сортировки в предложении ORDER BY определяет организацию упорядоченного результирующего набора. В предложение ORDER BY могут входить элементы, которых нет в списке выборки. Однако если указана конструкция SELECT DISTINCT, или инструкция содержит предложение GROUP BY, или если инструкция SELECT содержит оператор UNION, то столбцы сортировки должны присутствовать в списке выборки. Кроме того, если в инструкцию SELECT входит оператор UNION, то имена и псевдонимы столбцов должны быть из числа уточненных в первом списке выбора. Столбцы типа ntext, text, image или xml не могут быть использованы в предложении ORDER BY.

COLLATE {*collation_имя*}

Указывает, что операция ORDER BY должна выполняться в соответствии с параметрами сортировки, указанными в аргументе *collation_имя*, но не в соответствии с параметрами сортировки столбца, определенных в таблице или представлении. Значение *collation_имя* может быть именем параметров сортировки Windows или именем параметров сортировки SQL. Дополнительные сведения см. в MSDN в разделах «Настройка параметров сортировки в программе установки» и «Использование параметров сортировки SQL Server». Аргумент COLLATE применяется только к столбцам данных типа char, varchar, nchar и nvarchar.

ASC

Указывает, что значения в указанном столбце должны сортироваться по возрастанию, от меньших значений к большим значениям.

DESC

Указывает, что значения в указанном столбце должны сортироваться по убыванию, от больших значений к меньшим.

Примечания.

1. Значения NULL рассматриваются как минимально возможные значения.
2. Число элементов в предложении ORDER BY не ограничивается. Однако существует ограничение в 8 060 байт для размера строки промежуточных рабочих таблиц, необходимых для операций сортировки. Это ограничивает общий размер столбцов, указываемый в предложении ORDER BY.

WITH *обобщенное_табличное_выражение* задает временно именованный результирующий набор, называемый общим табличным выражением (ОТВ). Он получается при выполнении простого запроса и определяется в области выполнения одиночной инструкции SELECT, INSERT, UPDATE, MERGE или DELETE. Это предложение может использоваться также в инструкции CREATE VIEW как часть определяющей ее инструкции SELECT. Общее табличное выражение может включать ссылки на само себя. Такое выражение называется рекурсивным обобщенным табличным выражением.

Примеры простейших запросов

```
SELECT 2*2 AS "2x2"
SELECT 'Hello, World!' AS Field
SELECT AVG(Qty) AS [Средний размер поставки] FROM SPJ
```

Примеры запросов для демонстрационной базы данных AdventureWorks

A. Использование простого предложения FROM. В следующем примере извлекаются столбцы TerritoryID и Name из таблицы SalesTerritory в образце базы данных `AdventureWorks`.

```
SELECT TerritoryID, Name
FROM Sales.SalesTerritory
ORDER BY TerritoryID ;
```

B. Использование синтаксиса SQL-92 для CROSS JOIN. В следующем примере возвращается векторное произведение двух таблиц `Employee` и `Department`. Возвращается список всех возможных сочетаний строк `EmployeeID` и все строки имен `Department` .

```
SELECT e.EmployeeID, d.Name AS Department
FROM HumanResources.Employee e
CROSS JOIN HumanResources.Department d
ORDER BY e.EmployeeID, d.Name;
```

Г. Использование синтаксиса SQL-92 для FULL OUTER JOIN. В следующем примере возвращается имя продукта и любые соответствующие заказы на продажу в таблице `SalesOrderDetail`. В примере также возвращаются все заказы на продажу, продукты для которых не представлены в таблице `Product`, и все продукты с заказом на продажу, отличные от тех, которые представлены в таблице `Product`.

```
-- The OUTER keyword following the FULL keyword is optional.
SELECT p.Name, sod.SalesOrderID
FROM Production.Product p
FULL OUTER JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID
WHERE p.ProductID IS NULL
OR sod.ProductID IS NULL
ORDER BY p.Name ;
```

Д. Использование синтаксиса SQL-92 для LEFT OUTER JOIN. Следующий пример соединяет две таблицы по столбцу `ProductID` и сохраняет несовпадающие строки из левой таблицы. Таблица `Product` сопоставляется с таблицей `SalesOrderDetail` по столбцам `ProductID` в каждой таблице. В результирующем наборе отображаются все продукты, как входящие, так и не входящие в заказы.

```
SELECT p.Name, sod.SalesOrderID
FROM Production.Product p
LEFT OUTER JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID
ORDER BY p.Name ;
```

Е. Использование синтаксиса SQL-92 для INNER JOIN. Следующий пример возвращает все имена продуктов и все идентификаторы заказов на продажу.

```
-- By default, SQL Server performs an INNER JOIN if only the JOIN
-- keyword is specified.
SELECT p.Name, sod.SalesOrderID
FROM Production.Product p
INNER JOIN Sales.SalesOrderDetail sod
ON p.ProductID = sod.ProductID
ORDER BY p.Name ;
```

Ж. Использование синтаксиса SQL-92 для RIGHT OUTER JOIN. Следующий пример соединяет две таблицы по столбцу TerritoryID и сохраняет несовпадающие строки из правой таблицы. Таблица SalesTerritory сопоставляется с таблицей SalesPerson по столбцу TerritoryID каждой таблицы. В результирующем наборе отображаются все представители отдела продаж независимо от того, назначена им или нет обслуживаемая территория.

```
SELECT st.Name AS Territory, sp.SalesPersonID
FROM Sales.SalesTerritory st
RIGHT OUTER JOIN Sales.SalesPerson sp
ON st.TerritoryID = sp.TerritoryID ;
```

И. Использование производной таблицы. Следующий пример использует производную таблицу, инструкцию `SELECT` после предложения `FROM`, для возврата имен и фамилий сотрудников и городов, в которых они проживают.

```
SELECT RTRIM(c.FirstName) + ' ' + LTRIM(c.LastName) AS Name, d.City
FROM Person.Contact c INNER JOIN HumanResources.Employee e
ON c.ContactID = e.ContactID
INNER JOIN
   (SELECT ea.AddressID, ea.EmployeeID, a.City
    FROM Person.Address a INNER JOIN HumanResources.EmployeeAddress ea
    ON a.AddressID = ea.AddressID) AS d
ON e.EmployeeID = d.EmployeeID
ORDER BY c.LastName, c.FirstName;
```

Добавление новых строк: инструкция INSERT

Инструкция INSERT добавляет одну или несколько новых строк в таблицу или представление.

Упрощенный синтаксис

инструкция_insert ::=
INSERT [INTO] *имя_таблицы_или_представления* [(*список_имен_столбцов*)] *выражение_запроса* |
DEFAULT VALUES }

Распространенные случаи инструкции INSERT

Случай 1: INSERT *имя_таблицы_или_представления* (*список_имен_столбцов*)
VALUES (*список-константных-выражений*)

Случай 2: INSERT *имя_таблицы_или_представления*
SELECT *список_выбора*
FROM *список_таблиц*
WHERE *условное_выражение*

Замечание. Значением константного выражения может быть DEFAULT или NULL.

Примеры (для демонстрационной базы данных AdventureWorks2012)

```
INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
VALUES (1,'Orlando','N.','Gee');

INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
VALUES (12,'Johnny','A.','Capino'),
(16,'Christopher','R.','Beck'),
(18,'David','J.','Liu');

INSERT INTO dbo.demoCustomer (CustomerID, FirstName, MiddleName, LastName)
SELECT BusinessEntityID, FirstName, MiddleName, LastName
FROM Person.Person
WHERE BusinessEntityID BETWEEN 19 AND 35;
```

Изменение существующих данных: инструкция UPDATE

Инструкция UPDATE изменяет существующие данные в таблице или представлении.

Упрощенный синтаксис

инструкция_update ::=
UPDATE *имя_таблицы_или_представления*
SET *список_set_фраз*
[WHERE *условное_выражение*]

set_фраза ::=
имя_столбца = { *выражение* | NULL | DEFAULT }

Распространенные случаи инструкции UPDATE

Случай 1: UPDATE *имя_таблицы* SET *имя_столбца* = *выражение*
WHERE *условное_выражение*

Случай 2: UPDATE *имя_таблицы* SET *имя_столбца* = *выражение*
FROM *имя_таблицы*
WHERE *условное_выражение*

Примеры (для демонстрационной базы данных pubs)

```
update publishers set city = 'Atlanta', state = 'GA'

update publishers set pub_name = null

update authors set state = 'PC', city = 'Big Bad Bay City'
where state = 'CA' and city = 'Oakland'

update titleauthor set title_id = titles.title_id
from titleauthor, titles, authors
where titles.title = 'The Psychology of Computer Cooking'
      and authors.au_id = titleauthor.au_id
      and au_lname = 'Stringer'

update titles set ytd_sales = ytd_sales + qty
from titles, sales
where titles.title_id = sales.title_id
      and sales.ord_date in (select max(sales.ord_date) from sales)
```

По поводу этих примеров следует сделать следующее пояснение. FROM-предложение в инструкции UPDATE определяет, что для формулирования критериев операции обновления используется таблица, представление или производный источник таблицы. Если обновляемый объект тот же самый, что и объект в предложении FROM, и в предложении FROM имеется только одна ссылка на этот объект, псевдоним объекта указывать не обязательно. Если обновляемый объект встречается в предложении FROM несколько раз, только одна ссылка на этот объект не должна указывать псевдоним таблицы. Все остальные ссылки на объект в предложении FROM должны включать псевдоним объекта. Представление с триггером INSTEAD OF UPDATE не может быть целью инструкции UPDATE с предложением FROM.

Удаление данных: инструкция DELETE

Инструкция DELETE едаляет строки из таблиц и представлений.

Упрощенный синтаксис

инструкция_delete ::=
DELETE [FROM] *имя_таблицы_или_представления*
[FROM *список_табличных_источников*]
[WHERE *условное_выражение*]

Примеры (для демонстрационной базы данных pubs)

```
delete publishers where pub_name = 'Jardin, Inc.'

delete titles from authors, titles, titleauthor
where titles.title_id = titleauthor.title_id
      and authors.au_id = titleauthor.au_id
      and city = 'Big Bad Bay City'
```

Для второй инструкции DELETE будет напечатано следующее сообщение:

```
Сообщение 547, уровень 16, состояние 0, строка 1
Конфликт инструкции DELETE с ограничением REFERENCE "FK__titleauth__title__0BC6C43E". Конфликт произошел в базе данных "pubs", таблица "dbo.titleauthor", column 'title_id'.
```

Выполнение данной инструкции было прервано.