Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический университет имени П. О. Сухого»

Кафедра «Информационные технологии»

ИСПОЛЬЗОВАНИЕ ЯЗЫКА SQL

ПОСОБИЕ
по дисциплинам
«Сетевые технологии и базы данных»,
«Технологии организации хранения
и обработки данных», «Сетевые технологии»
для студентов экономических специальностей
дневной и заочной форм обучения

Электронный аналог печатного издания

Гомель 2007

УДК 004.43(075.8)
ББК 32.973-018.1я73
И88

Рекомендовано к изданию научно-методическим советом факультета автоматизированных и информационных систем ГГТУ им. П. О. Сухого
(*протокол № 9 от 10.08.2006 г.*)

Авторы-составители: *О. Д. Асенчик*, *Е. Г. Стародубцев*

Рецензент: начальник сектора локальных вычислительных сетей вычислительного центра ГГТУ им. П. О. Сухого *С. Б. Сон*

И88 **Использование** языка SQL : пособие по дисциплинам «Сетевые технологии и базы данных», «Технологии организации хранения и обработки данных», «Сетевые технологии» для студентов экон. специальностей днев. и заоч. форм обучения / авт.-сост.: О. Д. Асенчик, Е. Г. Стародубцев. – Гомель : ГГТУ им. П. О. Сухого, 2007. – 21 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: http://gstu.local/lib. – Загл. с титул. экрана.

ISBN 978-985-420-606-6.

Рассматриваются общие понятия реляционной модели данных, описываются операции над реляционными таблицами и их реализация в языке SQL. В качестве «диалекта» SQL выбрана версия языка для ядра баз данных MS Jet – в соответствии с требованием учебных программ для курсов «Сетевые технологии и базы данных», «Технологии организации хранения и обработки данных». Все формальные конструкции синтаксиса SQL, реализующие основные операции обработки данных реляционных таблиц, сопровождаются примерами.

Для студентов экономических специальностей дневной и заочной форм обучения.

УДК 004.43(075.8)
ББК 32.973-018.1я73

ISBN 978-985-420-606-6

1. РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Основой любой базы данных является *модель данных* – совокупность структур данных и операций их обработки. Далее мы будем рассматривать базы данных на основе одной из наиболее распространенных моделей – *реляционной модели данных*. Реляционные модели имеют простую структуру данных, удобное табличное представление и используют аппарат алгебры отношений и реляционного исчисления для обработки данных.

Отношение, или *реляционная таблица*, имеет прямоугольную структуру («плоский файл»). Столбцы реляционной таблицы называются *атрибутами*, *полями* или *доменами*, строки – *записями* или *кортежами отношения*. Реляционная таблица может иметь *ключ* – поле или группу полей, значение которых необходимо для идентификации (однозначного определения) записей. Ключ имеет свойство уникальности значения для каждой записи. Если в таблице имеется несколько ключей, то один из них в объявлении схемы таблицы принимается за *первичный* (*основной*) *ключ*, остальные называются *возможными ключами*. По первичному ключу выполняется автоматическое упорядочивание записей – сортировка записей в порядке возрастания значений ключа. В качестве ключевых полей обычно выбираются поле (поля), наиболее часто используемые для поиска записи.

Работа с данными в реляционных таблицах выполняется на двух уровнях: с данными одной или нескольких таблиц.

1.1. Типовые операции по обработке записей одной реляционной таблицы

Состав операций с реляционными таблицами определяет требования к реляционным языкам. Рассмотрим типовые операции по обработке записей одной реляционной таблицы.

Включить – в таблицу добавляется новая запись, для этого указывается имя таблицы, в которую производится добавление, и значения полей новой записи, обязательно заполнение ключевых полей.

Удалить – из таблицы удаляется одна запись или группа (набор) записей, указывается имя таблицы и первичный ключ удаляемой записи – при удалении одной записи, или условие, выполнение которого необходимо для удаления группы записей.

Обновить – изменяются значения полей указанной записи, задается имя таблицы, идентификатор обновляемой записи или записей, новые значения изменяемых полей.

***Выборка* (*селекция*)** – выбор подмножества записей таблицы по условию отбора в виде логического выражения. Результирующая таблица имеет ту же схему, что и исходная. Эту операцию называют «горизонтальной» выборкой. Частный случай выборки – *пустая выборка*, если записи таблицы не соответствуют условию отбора.

Проекция – указывается подмножество полей исходной таблицы для формирования новой таблицы, имеющей другую схему и, возможно, другой набор записей (могут исключаться повторяющиеся данные). Эта операция называется «вертикальной» выборкой.

1.2. Типовые операции совместной обработки нескольких реляционных таблиц

Операции совместной обработки реляционных таблиц различны для таблиц с одинаковой или отличающейся *схемой данных*. Применительно к *односхемным таблицам*, т. е. к таблицам с одинаковым составом полей, выполняются следующие операции:

1) ***объединение*** – для двух таблиц строится новая таблица той же схемы, содержащая совокупность записей исходных таблиц. Если исходные таблицы содержат записи с одинаковыми значениями первичного ключа, то при объединении таблиц такие записи не дублируются в результирующей таблице. Если записи исходных таблиц имеют различные значения первичного ключа, то результатом объединения будет совокупность всех записей исходных таблиц;

2) ***пересечение*** – для двух таблиц строится новая таблица той же схемы, содержащая общие для них записи (с одинаковыми значениями первичного ключа). Если таких записей нет, результатом пересечения будет пустая таблица;

3) ***вычитание*** – для двух таблиц строится новая таблица той же схемы, содержащая записи первой таблицы, отличные от записей второй таблицы. Если записей, содержащих одинаковое значение первичного ключа, во второй таблице нет, результат вычитания – полный состав записей первой таблицы; если записи второй таблицы содержат все значения первичного ключа записей первой таблицы, результат вычитания – пустая таблица.

Применительно к *разносхемным* таблицам, т. е. таблицам с разным составом полей, выполняются следующие операции:

1) ***декартово произведение*** – две таблицы образуют новую таблицу, которая включает все поля исходных таблиц. В результирующей таблице выводится итог соединения типа «каждый с каждым», при этом могут отсутствовать значения отдельных полей в результирующей записи. Условием совместной обработки разносхемных реляционных таблиц в ряде случаев является наличие общих по типу и значению полей, называемых *внешними ключами* и используемых для задания связей между таблицами;

2) ***соединение*** – две таблицы, имеющие общие поля – внешние ключи, участвуют в создании новой таблицы. Схема новой таблицы строится объединением всех полей исходных таблиц, а результирующие записи формируются по определенным условиям: при выборе из таблиц записей только с одинаковыми значениями внешних ключей – *симметричное соединение*; при выборе всех записей одной из таблиц и только соответствующих им записей другой таблицы – *внешнее соединение*.

2. ЯЗЫК SQL

2.1. Реляционные языки. Общая характеристика языка SQL

Реляционные языки обеспечивают типовые операции по обработке данных реляционных таблиц, позволяют формировать логические условия для операций выборки, проверку целостности (непротиворечивости) данных взаимосвязанных таблиц.

Реляционные языки оперируют с данными как с множествами, применяя к ним основные операции теории множеств. На входе реляционного оператора – множество записей одной или нескольких реляционных таблиц, на выходе – множество записей новой реляционной таблицы. Реляционные языки имеют различный уровень *процедурности* – содержания и последовательности перехода от входных данных к выходным.

Выделяют следующие типы реляционных языков:

1) *dBASE-подобные языки* – приближены к языкам структурного программирования, обеспечивают создание интерфейса пользователя и типовые операции обработки данных. Используются в системах управления базами данных (СУБД) dBASE, Paradox, FoxPro, Clipper и др.

Эти языки занимают промежуточное положение между языками программирования и *языками манипулирования данными* СУБД; обладают выраженной процедурностью обработки данных, т. к. в них явно указывается последовательность действий, ведущих к конечному результату;

2) *графические реляционные языки* – ориентированы на конечных пользователей реляционных баз данных, включая начинающих пользователей. Характеризуются графическим интерфейсом, приемы работы с которым достаточно просты и быстро осваиваются. Типичный представитель – язык QBE (**Q**uery **B**y **E**xample – запрос по образцу), реализованный в различных программных продуктах, например, собственно в составе СУБД (MS Access, MS SQL Server и других), в пакете MS Query, входящем в состав интегрированного пакета программ MS Qffice;

3) *язык структурированных запросов* SQL (**S**tructured **Q**uery **L**anguage) – средство для работы с реляционными базами данных. SQL – *реляционно полный язык*, т. к. в нем реализованы все операции обработки данных реляционных таблиц.

Язык SQL утвержден как стандарт Американским Национальным Институтом Стандартов (ANSI). Однако ведущие производители баз данных, беря стандарт за основу, расширяют его требования и создают свои *диалекты* или *версии* языка SQL. Так, фирмы «Microsoft» и «Oracle» создали свои версии языка, соблюдающие стандарт «в целом», но имеющие ряд особенностей. В результате программный код, написанный на разных версиях SQL, как правило, не является полностью переносимым из одной СУБД в другую.

Для доступа к базам данных СУБД MS Access используется версия языка SQL – *SQL ядра базы данных Microsoft Jet* (MS Jet). MS Jet – часть СУБД (*диспетчер данных*), выполняющая функции загрузки и сохранения данных в пользовательских и системных базах данных. MS Access версии 2002 поддерживает два стандарта SQL, которые соответствуют стандартам ANSI–89 уровень 1 и ANSI–92 уровень 1. Далее мы будем рассматривать традиционные конструкции Jet SQL, которые в основном соответствуют стандарту ANSI–89. Стандарт ANSI–92 уровень 1 ориентирован на связь с СУБД MS SQL Server.

Как правило, используются следующие две основные технологии работы с языком SQL. Во-первых, SQL применяется как *интерактивный язык*, непосредственно «работающий» с базой данных с использованием специальных программ-утилит, которые могут вхо-

дить, например, в состав программного комплекса той или иной СУБД. При этом сразу выводятся результаты выполнения различных команд обработки данных. Во-вторых, SQL используется как *вложенный язык*, команды которого могут быть вставлены в программы, написанные на различных языках разработки прикладных программ, например, Visual Basic, C/C++, C#, Delphi, Java.

Основу языка SQL составляют операторы, которые можно разбить на группы, опираясь на их функциональное назначение. Рассмотрим основные группы часто используемых операторов SQL.

Операторы определения объектов базы данных – *операторы DDL* (**D**ata **D**efinition **L**anguage). Примеры таких операторов: CREATE SCHEMA – создать схему базы данных; DROP SCHEMA – удалить схему базы данных; CREATE TABLE – создать таблицу; ALTER TABLE – изменить таблицу; DROP TABLE – удалить таблицу; CREATE VIEW – создать представление; DROP VIEW – удалить представление.

Операторы манипулирования данными – *операторы DML* (**D**ata **M**anipulation **L**anguage), например: SELECT – выбрать строки из таблицы; INSERT – добавить строки в таблицу; UPDATE – изменить строки в таблице; DELETE – удалить строки из таблицы; COMMIT – зафиксировать изменения; ROLLBACK – отменить изменения.

Операторы защиты и управления доступом, такие как: GRANT – предоставить привилегии; REVOKE – отменить привилегии; CREATE ASSERTION – создать ограничение; DROP ASSERTION – отменить ограничение.

Команды языка SQL являются инструкциями, с помощью которых пользователь обращается к базе данных. Команды состоят из одной или нескольких логических частей, называемых *предложениями*. Предложения начинаются *ключевым словом* – служебным словом, имеющим специальное значение в SQL, и состоят из ключевых слов, определяющих имя предложения, и аргументов.

2.2. Типы данных

Определение типов данных в языке SQL является основной областью, в которой коммерческие программы СУБД и стандарты SQL не всегда совпадают. Типы данных языка SQL ядра базы данных MS Jet включают 13 основных типов данных. Характеристика наиболее часто используемых типов данных приведена в таблице.

Типы данных языка SQL ядра MS Jet

Тип данных	Синоним	Размер	Описание
CHARACTER	TEXT(n), CHAR	2 байта на знак	Текст от 0 до 255 знаков
BIT	BOOLEAN, LOGICAL	1 байт	Значения «Да» (Yes) и «Нет» (No), а также поля, содержащие одно из двух возможных значений
TINYINT	INTEGER1, BYTE	1 байт	Целое значение от 0 до 255
SMALLINT	SHORT, INTEGER2	2 байта	Короткое целое от –32 768 до 32 767
INTEGER	LONG, INT, INTEGER4	4 байта	Длинное целое от –2 147 483 648 до 2 147 483 647
REAL	SINGLE, FLOAT4	4 байта	Число с плавающей точкой и одинарной точностью
FLOAT	DOUBLE, FLOAT8, NUMBER	8 байт	Число с плавающей точкой и двойной точностью
MONEY	CURRENCY	8 байт	Масштабируемое целое от –922 337 203 685 477,5808 до 922 337 203 685 477,5807
DATETIME	DATE, TIME	8 байт	Дата или время; допустимы даты любого года от 100 до 9999

2.3. Операторы языка определения данных

В синтаксических конструкциях на языке SQL, приведенных ниже, будем использовать следующие обозначения:

– знак «звездочка» (*) используется для обозначения «все», т. е. «все случаи, удовлетворяющие определению»;

– в квадратные скобки ([ ]) заключены необязательные элементы синтаксиса (сами квадратные скобки в этом случае в операторе SQL не используются);

– вертикальная черта (|), разделяющая два элемента, указывает на то, что в операторе используется либо один элемент, либо второй (сама вертикальная черта в текст оператора также не включается);

– в фигурные скобки ({ }) заключаются элементы, разделенные вертикальной чертой (иначе невозможно понять, какие именно элементы предложены на выбор);

– троеточие означает, что далее в операторе следует (в зависимости от контекста) либо выражение, либо повторяются элементы, указанные перед тремя точками.

Для создания реляционной таблицы нужно задать состав и свойства её полей: имя, тип и длину (если нужно) каждого поля. Каждая таблица в базе данных должна иметь уникальное имя.

Синтаксис оператора создания новой таблицы:

CREATE TABLE *таблица* (*поле_1 тип* [(*размер*)]
[NOT NULL] [*индекс_1*] [, *поле_2 тип* [(*размер*)] [NOT NULL]
[*индекс_2*] [, . . .]] [, CONSTRAINT *составной_индекс* [,…]]);

где *таблица* – имя создаваемой таблицы; *поле_1*, *поле_2*,…– имена полей таблицы (таблица должна содержать хотя бы одно поле); *тип* – тип данных поля; *размер* – размер поля в символах (только для текстовых и двоичных полей); *индекс_1*, *индекс_2*,… – директивы создания простых индексов (по отдельному полю); *составной_индекс* – директива создания составного индекса (по нескольким полям). Ограничение для поля NOT NULL (не пустое), требует обязательного заполнения поля допустимыми данными при добавлении новых записей в таблицу. Инструкция на языке SQL заканчивается точкой с запятой.

Каждый индекс имеет уникальное в пределах данной таблицы имя. Для создания *простого индекса* используется инструкция (помещается за именем поля, входящего в индекс):

CONSTRAINT *имя_индекса* {PRIMARY KEY | UNIQUE |
REFERENCES *внешняя_таблица* [(*внешнее_поле*)]};

Директива создания *составного индекса* (помещается в любом месте после определения элементов индекса):

CONSTRAINT *имя_индекса* {PRIMARY KEY (*ключевое_поле_1*
[, *ключевое_поле_2* [,…]]) | UNIQUE (*уникальное_поле_1* [,…]]) |
FOREIGN KEY (*ссылка_1* [, *ссылка_2* [,…]]) REFERENCES
внешняя_таблица [(*внешнее_поле_1* [, *внешнее_поле_2*[,…]])]};

В инструкциях использованы следующие служебные слова: UNIQUE – уникальный индекс (в таблице не может быть двух записей, имеющих одно и то же значение полей, входящих в индекс); PRIMARY KEY – первичный ключ таблицы (может состоять из нескольких полей; упорядочивает записи таблицы); FOREIGN KEY – внешний ключ для связи с другими таблицами (может состоять из нескольких полей); REFERENCES – ссылка на внешнюю таблицу.

ПРИМЕР

CREATE TABLE *Студенты* (*Имя* TEXT, *Фамилия* TEXT, *ДатаРождения* DATETIME, CONSTRAINT *Индекс1* UNIQUE (*Имя*, *Фамилия*, *ДатаРождения*));

Создаётся таблица *Студенты* с текстовыми полями *Имя*, *Фамилия*, полем *ДатаРождения* типа дата/время и с составным уникальным индексом с именем *Индекс1*, включающим все поля этой таблицы.

2.4. Операторы языка обработки данных

Инструкция SELECT

Получив эту инструкцию, ядро MS Jet возвращает из базы данных набор запрошенных записей. Инструкция SELECT формирует запрос на выборку (Select Query). Синтаксис инструкции:

SELECT [*предикат*] { * | *таблица*.* | [*таблица*.]*поле_1*
[AS *псевдоним_1*] [, [*таблица*.]*поле_2* [AS *псевдоним*_2] [,...]]}
FROM *выражение* [, ...] [IN *внешняя_база_данных*] [WHERE...] [GROUP BY...] [HAVING...] [ORDER BY...] [WITH OWNERACCESS OPTION];

Описание параметров инструкции:

– *предикат* – задает различные варианты отбора записей. Возможные значения: ALL (возвращаются все значения, удовлетворяющие условиям отбора), DISTINCT (если записи содержат повторяющиеся значения в выбранных для запроса полях, то возвращается только одна из них), DISTINCTROW (если записи полностью повторяются, то возвращается только одна из них), TOP n [PERCENT] (возвращаются либо первые *n* записей, либо первые *n* процентов записей из набора данных). Если предикат не указан, используется значение ALL;

– * или *таблица*.*. Символ «звездочка» (*) указывает на выбор всех полей таблицы. Если в запросе участвует одна таблица, может использоваться упрощенный синтаксис (*), если таблиц несколько – синтаксис *ИмяТаблицы*.* указывает на то, что нужно выбрать все поля из конкретной таблицы *ИмяТаблицы*;

– *таблица* – имя таблицы, данные из которой выбираются;

– *поле_1*, *поле_2* – имена полей, из которых выбираются данные. Если полей несколько, они выбираются в указанном порядке;

– *псевдоним_1*, *псевдоним_2* – новые названия для столбцов (заголовки), которые будут использоваться вместо имен полей;

– *выражение* – имена одной или нескольких таблиц, участвующих в запросе. Например, *Таблица1*, *Таблица2* AS *Псевдоним*. Вместо имени *Таблица2* в последующей части SQL-оператора можно использовать *Псевдоним*. Кроме того, здесь же может быть указана связь между таблицами при помощи операций INNER JOIN (симметричное соединение), LEFT JOIN и RIGHT JOIN (левое и правое соединения);

– *внешняя_база_данных* – если информация запрашивается не из текущей базы данных, сразу за ключевым словом IN нужно указать полный путь и имя файла внешней базы данных, заключенные в одинарные кавычки;

– *предложение* WHERE – вслед за ключевым словом WHERE указывается логическое выражение. Записи из базовых таблиц (запросов) попадают в итоговую выборку в случае истинности этого выражения. Предложение WHERE необязательное, но если оно присутствует, то следует за предложением FROM;

– *предложение* GROUP BY – содержит список полей, которые используются для группировки записей. Порядок следования полей в этом предложении определяет уровни их группировки. В предложении GROUP BY можно указать до 10 полей. Группировку можно производить и для выражений (например, вычисляемых полей);

– *предложение* HAVING – вслед за предложением HAVING указывается логическое выражение, определяющее, какие из уже отобранных и сгруппированных записей будут отображаться в результирующем наборе данных. От предложения WHERE отличается тем, что записи отбираются уже после группировки. Таким образом, предложение WHERE используется для первоначального отбора записей во временный набор данных, а предложение HAVING задает, какие из сгруппированных записей временного набора данных попадут в итоговую выборку. Предложение HAVING можно использовать только в случае использования предложения GROUP BY, и оно должно располагаться непосредственно за ним;

– *предложение* ORDER BY – содержит список полей, по которым производится сортировка в итоговом наборе данных. Наивысший приоритет при сортировке имеет первое поле, указанное в этом предложении. Вслед за именем поля может быть указан порядок сортировки – ASC (по возрастанию) или DESC (по убыванию). Если порядок сортировки не указан, применяется сортировка по возрастанию (ASC). Сортировку можно применять и к выражениям, в частности, к вычисляемым полям;

– *описание* WITH OWNERACCESS OPTION – используется для предоставления пользователю, выполняющему запрос, тех же прав, которыми обладает владелец запроса (Owner);

– *предложение* IN – определяет внешний файл базы данных, с которой устанавливается связь.

ПРИМЕРЫ

SELECT TOP 25 * FROM *Заказы* IN "E:\databases\Борей.mdb";

– выборка данных из таблицы *Заказы*, которая находится во внешнем файле базы данных Борей.mdb, в папке E:\databases. Если предложение IN опустить, то выборка будет производиться из таблицы *Заказы* текущей базы данных (соединение с которой в данный момент активно). Символ «звездочка» (*) перед ключевым словом FROM указывает на то, что из таблицы выбираются все поля. Отсутствие условия выборки (конструкции WHERE) означает, что в итоговую выборку должны попасть все без исключения записи таблицы *Заказы*. Однако размещенный в конструкции SELECT предикат TOP 25 ограничивает количество возвращаемых записей первыми 25 записями.

SELECT *Заказано.*, Заказы.КодКлиента, Заказы.ДатаИсполнения, Заказы.КодЗаказа* FROM *Заказы, Заказано*
WHERE (((*Заказы.ДатаИсполнения*)>=#1/1/1998#) AND
((*Заказы.КодЗаказа*)=[*Заказано*].[*КодЗаказа*])) OR
(((*Заказы.ДатаИсполнения*) Is Null) AND
((*Заказы.КодЗаказа*)=[*Заказано*].[*КодЗаказа*]));

– задается выборка данных из двух таблиц: *Заказы* и *Заказано*. Наименования этих таблиц перечислены через запятую в предложении FROM. Конструкция *Заказано.** предложения SELECT означает, что в итоговую выборку должны попасть все поля из таблицы *Заказано*. Далее через запятую перечислены два поля из таблицы *Заказы*, которые также должны попасть в итоговую выборку. Предложение WHERE содержит сложное логическое выражение. Конструкция *[Заказы].[КодЗаказа]=[Заказано].[КодЗаказа]* задает связь между таблицами по полям *КодЗаказа* обеих таблиц. Остальная часть логического выражения, указанного в предложении WHERE, задает выборку только тех записей из связанных таблиц, для которых заказы были выполнены позднее 01.01.98 г. или не были выполнены вообще (поле *ДатаИсполнения* имеет значение Null).

SELECT *Название, Город, Адрес* FROM *Клиенты*
WHERE (((*Страна*) IN ('Беларусь', 'Россия', 'Мексика')));

В этом примере оператор SQL запрашивает информацию из таблицы *Клиенты*, расположенной в текущей базе данных. В итоговую выборку попадут только поля *Название*, *Город* и *Адрес*. Четвертое поле, *Страна*, используется в предложении WHERE исключительно для ограничения при отборе записей из базовой таблицы и отображено в итоговой выборке не будет. В предложении WHERE используется оператор IN, возвращающий значение Истина (True), если значение поля совпадает с одним из элементов, перечисленных после ключевого слова IN. Список оператора IN берется в круглые скобки, элементы перечисляются через запятую (в бланке запроса – через точку с запятой). Таким образом, в итоговую выборку попадет только информация о клиентах из Беларуси, России и Мексики.

Обратите внимание на разницу в использовании ключевого слова IN в предложениях FROM и WHERE в примерах выше.

SELECT *КодЗаказа*, Sum(*Цена*Количество*) AS [*Стоимость заказа*]
FROM *Заказано* GROUP BY *КодЗаказа*
HAVING (((Sum(*Цена*Количество*))>1000))
ORDER BY Sum(*Цена*Количество*) DESC;

Этот оператор SQL будет осуществлять операцию выборки с группировкой. Из таблицы *Заказано* текущей базы данных отбираются поле *КодЗаказа* и произведение значений полей *Цена* и *Количество*. Поскольку в операторе отсутствует предложение WHERE, то во временную выборку попадут значения указанных полей из всех записей таблицы. Предложение GROUP BY содержит поле *КодЗаказа* – именно по этому полю будет выполняться группировка отобранной информации. Для остальных полей, указанных в предложении SELECT, должны быть указаны какие-либо статистические функции. В данном примере для вычисляемого поля – произведения значений полей *Цена* и *Количество* – указана групповая операция суммирования (Sum).

В процессе группировки будут суммироваться значения выражения *Цена*Количество*, соответствующие каждому значению поля *КодЗаказа*. Таким образом, во временном наборе сгруппированных данных будут фигурировать записи, каждая из которых содержит пару значений: код заказа и полная стоимость заказа. Поскольку предложение HAVING содержит условие, то в итоговую выборку попадут

только те записи из сгруппированной выборки, которые удовлетворяют заданному условию (то есть только те записи, для которых стоимость заказа больше 1000). Итоговая выборка записей будет отсортирована по убыванию стоимости заказа, поскольку в предложении ORDER BY после соответствующего выражения указано ключевое слово DESC. Обратите внимание на то, что сортировка задана для выражения.

Для вычисляемого поля, соответствующего стоимости заказа, указан псевдоним *Стоимость заказа*. Этот псевдоним в дальнейшем можно использовать: в качестве имени поля запроса (например, при обращении к нему из модуля на языке Visual Basic); в качестве заголовка столбца в итоговой выборке. Так как псевдоним содержит символ «пробел», то он заключен в квадратные скобки.

Операция JOIN

Операции JOIN – *внутреннего объединения* (INNER JOIN), *внешнего объединения* (LEFT JOIN и RIGHT JOIN) – служат для создания связей между таблицами. Эти операции могут использоваться в любом предложении FROM (или, в случае запроса на обновление, в предложении UPDATE). Операция INNER JOIN объединяет записи из двух таблиц, если связующие поля одного типа и содержат одинаковые значения, остальные записи в выборке не участвуют. Операция LEFT JOIN используется для создания *левого внешнего объединения*. Левое внешнее объединение включает все записи из первой (левой) таблицы (даже если нет совпадающих значений для записей из второй (правой) таблицы) и только соответствующие по значению связующего поля записи правой таблицы.

Операция RIGHT JOIN используется для создания *правого внешнего объединения*. Правое внешнее объединение включает все записи из второй (правой) таблицы (даже если нет совпадающих значений с записями из первой (левой) таблицы) и только соответствующие по значению связующего поля записи левой таблицы.

Например, операцию LEFT JOIN можно использовать со связанными таблицами *Отделы* (левой) и *Сотрудники* (правой) для отбора всех отделов, включая те, в которых нет сотрудников. Чтобы отобрать всех сотрудников, включая не приписанных ни к какому отделу, можно использовать операцию RIGHT JOIN.

Синтаксис:

таблица_1 [INNER | LEFT | RIGHT] JOIN *таблица_2*
ON *таблица_1.поле_1 оператор таблица_2.поле_2*;

Описание параметров: *таблица_1*, *таблица_2* – имена таблиц, записи которых подлежат объединению; *поле_1*, *поле_2* – имена полей, используемых для связи; *оператор* – любой из операторов сравнения =, <, >, <=, >= или <> (не равно). В обычных ситуациях применяется оператор =, а остальные операторы сравнения служат для создания сложных связей.

Операция JOIN может содержать несколько предложений ON. В этом случае может использоваться такой синтаксис:

SELECT *поля* FROM *таблица_1* INNER JOIN *таблица_2*
ON *таблица_1.поле_1 оператор таблица_2.поле_1* AND
ON *таблица_1.поле_2 оператор таблица_2.поле_2*) OR
ON *таблица_1.поле_3 оператор таблица_2.поле_3*)];

Кроме того, операции JOIN могут быть вложенными:

SELECT *поля* FROM *таблица_1* INNER JOIN
(*таблица_2* INNER JOJN [( ]*таблица_3*
[INNER JOIN [( ]*таблица_X* [INNER JOIN ...)]
ON *таблица_3.поле_3* оператор *таблица_X.поле_X*)]
ON *таблица_2.поле_2* оператор *таблица_3.поле_3*)
ON *таблица_1.поле_1* оператор *таблица_2.поле_2*;

Операции LEFT JOIN и RIGHT JOIN могут быть вложены в операцию INNER JOIN, однако обратное вложение недопустимо.

ПРИМЕРЫ

SELECT *Типы.Категория*, *Товары.Марка*, *Товары.Цена*
FROM *Типы* INNER JOIN *Товары* ON
Типы.КодТипа = Товары.КодТипа WHERE (((*Товары.Цена*)>1000));

Инструкция объединяет таблицы *Типы* и *Товары* по полю *КодТипа*. После ее выполнения будет выведен набор записей с тремя столбцами *Категория*, *Марка*, *Цена*, причем значение поле *Цена* будет больше 1000.
SELECT *Заказы.КодКлиента*, *Клиенты.Название* FROM *Клиенты* LEFT JOIN *Заказы* ON *Клиенты.КодКлиента = Заказы.КодКлиента* WHERE (((*Заказы.КодКлиента*) Is Null));

Приведенный SQL-оператор иллюстрирует часто используемый вариант применения внешнего объединения (в данном случае LEFT JOIN). В этом примере отбираются все записи из таблицы *Клиенты* и только те записи из таблицы *Заказы*, для которых есть связанные поля в таблице *Клиенты*. Записи, соответствующие отсутствующим для клиентов заказам, будут содержать в поле *Заказы.КодКлиента* значение Null. Согласно условию предложения WHERE, в итоговую выборку попадут наименования только тех клиентов, для которых отсутствуют соответствующие записи в таблице *Заказы*.

Такого типа операторы с внешним объединением таблиц служат для поиска «потерянных» записей в таблицах, связанных отношением *один-ко-многим*. В последнем примере разыскиваются клиенты, для которых не оформлено ни одного заказа. Если левое объединение (LEFT JOIN) заменить правым (RIGHT JOIN), то разыскиваться будут заказы, для которых в таблице *Клиенты* нет соответствующих записей. Такая ситуация невозможна, если между таблицами задана связь внутреннего типа (INNER JOIN) на уровне базы данных и задана поддержка *ссылочной целостности данных* (referential integrity).

Инструкция UPDATE

Инструкция UPDATE лежит в основе запросов на обновление (Update Query). При помощи запроса этого типа можно изменить значения указанных полей заданной таблицы на основе заданного набора условий. Синтаксис:

UPDATE *таблица* SET *поле=новоеЗначение* WHERE *условиеОтбора*;

Описание параметров: *таблица* – имя таблицы, поля которой подлежат обновлению; *поле=новоеЗначение* – список конструкций *поле=новоеЗначение*, следующий за предложением SET, задает новые значения, присваиваемые указанным полям таблицы; *условиеОтбора*– условие, на основании которого производится обновление полей указанной таблицы. Синтаксис предложения WHERE аналогичен рассмотренному выше для инструкции SELECT.

Обычно инструкция UPDATE используется для обновления большого количества полей и большого количества записей, в том числе, когда записи находятся в разных таблицах (в запросах на обновление можно применять различные типы объединений (JOIN), хотя и с большой осторожностью).

ПРИМЕРЫ

UPDATE *Товары* SET *ПоставкиПрекращены* = False
WHERE (((*ПоставкиПрекращены*)=True));

Приведенный в этом примере SQL-оператор служит для обновления поля *ПоставкиПрекращены* таблицы *Товары*. Имя таблицы указано в конструкции UPDATE, а конструкция SET содержит список выражений вида: *поле=выражение*. Аргумент *поле* ссылается на обновляемое поле (*ПоставкиПрекращены*), а значение аргумента *выражение* (False) будет присвоено указанному слева от знака равенства полю. В выражении могут использоваться имена полей из таблиц, участвующих в запросе. В приведенном примере обновляемое поле одно – поле логического (Yes/No) типа *ПоставкиПрекращены*. Этому полю будет присвоено значение Ложь (False) для всех записей таблицы *Товары*, удовлетворяющих условиям конструкции WHERE.

В данном случае конструкция WHERE содержит всего одно условие. В результате работы запроса будут обновлены только те записи, в которых поле *ПоставкиПрекращены* имеет значение Истина (True), остальные записи будут оставлены без изменения. Итог работы запроса, основанного на рассматриваемой в этом примере строке SQL-оператора, не зависит от конструкции WHERE. Все записи таблицы *Товары* в поле *ПоставкиПрекращены* будут иметь значение Ложь (False) независимо от того, присутствует ли конструкция WHERE в строке запроса. Однако в случае отсутствия предложения WHERE запрос будет работать намного медленнее, поскольку обновляться будут все без исключения записи. Для таблиц, содержащих большое количество записей, разница в скорости выполнения таких запросов может быть достаточно существенной.

UPDATE *Заказы* INNER JOIN *Заказано* ON
Заказы.КодЗаказа = *Заказано.КодЗаказа* SET *Заказано.Скидка* = 0
WHERE (((*Заказы-ДатаИсполнения*) Is Null));

В этом примере использованы две таблицы – *Заказы* и *Заказано*. Инструкция UPDATE не может содержать конструкции FROM, поэтому информация об участвующих в запросе таблицах, а также о связях между этими таблицами задается в конструкции UPDATE (информация о связях может быть указана также в конструкции WHERE). В примере полю *Скидка* из таблицы *Заказано* присваивается значение 0 для тех записей, для которых поле *ДатаИсполнения* из связанной таблицы *Заказы* имеет значение Null. Таким образом, если заказ еще не выполнен, то для него снимается скидка. Для задания связи между таблицами *Заказы* и *Заказано* используется обычная для MS Access конструкция INNER JOIN:

UPDATE *Заказы* INNER JOIN *Заказано* ON *Заказы.КодЗаказа* = *Заказано.КодЗаказа* SET *Заказы.ДатаНазначения* = #1/1/1999#, *Заказано.Скидка* =[*Скидка*]*0.9
WHERE (((*Заказы.СтоимостьДоставки*)>30) AND ((*Заказы.ДатаИсполнения*) Is Null));

Запрос, основанный на рассматриваемой строке SQL-оператора, обновляет поля записей, соответствующих заказам, которые еще не выполнены (поле *ДатаИсполнения* имеет значение Null) и стоимость доставки которых больше 30. При выполнении указанных условий полю *ДатаНазначения* из таблицы *Заказы* присваивается значение 01.01.1999, а значение поля *Скидка* из таблицы *Заказано* уменьшается на 10 %.

Инструкция DELETE

Инструкция DELETE лежит в основе запросов на удаление и предназначена для удаления записей из одной или нескольких таблиц, перечисленных в предложении FROM и удовлетворяющих условиям предложения WHERE (если оно указано). Синтаксис:

DELETE [*таблица.*]*, *поле_1* [,...] FROM *таблица*
WHERE *условиеОтбора*;

Описание параметров:

– * (звездочка) – указывает на то, что должны быть удалены все записи из таблиц, перечисленных в предложении FROM. Если перед символом «звездочка» указано имя таблицы, то удалены будут записи из указанной таблицы. Этот параметр не обязателен. Его можно не указывать даже в том случае, когда предложение DELETE не содержит других параметров;

– *таблица* – необязательный параметр. Если в запросе задано несколько связанных между собой таблиц, этот параметр указывает, из какой именно таблицы удаляются записи;

– *поле_1[,...]* – список полей, которые будут использоваться для задания условий отбора в предложении WHERE;

– *условиеОтбора* – условия, при выполнении которых удаляются записи из указанной таблицы (таблиц).

При выполнении запроса, содержащего инструкцию DELETE, удаляются только данные, структура таблицы остается неизменной. Если при создании связей между таблицами в базе данных была зада-

на опция *каскадное удаление*, то удаляются также соответствующие записи из связанных таблиц, находящихся на стороне *многие* отношения *один-ко-многим*. Например, при удалении записи из таблицы *Клиенты* из таблицы *Заказы* без всякого предупреждения будут удалены все записи о заказах, связанных с этим клиентом. Запросы на удаление удаляют записи целиком, а не только отдельные значения полей. Восстановить информацию после запуска на выполнение запроса на удаление невозможно, поэтому рекомендуется предварительно (до запуска на выполнение запроса) сделать резервную копию данных.

ПРИМЕР

DELETE *ПоставкиПрекращены* FROM *Товары*
WHERE (((*ПоставкиПрекращены*)=True));

В предложении DELETE указано поле *ПоставкиПрекращены* из таблицы *Товары*, поскольку оно используется в условии отбора в предложении WHERE. При помощи данного запроса можно удалить из таблицы *Товары* все записи о товарах, поставки которых прекращены.

ЛИТЕРАТУРА

1. Информатика : учебник / под ред. Н. В. Макаровой. – Москва : Финансы и статистика, 2004. – 767 с.

2. Учебные материалы: MS Access [Электронный ресурс]. – Режим доступа : http://matveev.kiev.ua/macceass/.

СОДЕРЖАНИЕ

Учебное электронное издание комбинированного распространения

Учебное издание

ИСПОЛЬЗОВАНИЕ ЯЗЫКА SQL

Пособие
по дисциплинам
«Сетевые технологии и базы данных», «Технологии организации хранения и обработки данных», «Сетевые технологии» для студентов экономических специальностей дневной и заочной форм обучения

Авторы-составители: **Асенчик** Олег Даниилович
Стародубцев Евгений Генрихович

Редактор *Н. В. Гладкова*
Компьютерная верстка *Н. В. Широглазова*

Подписано в печать 02.10.07.
Формат 60x84/16. Бумага офсетная. Гарнитура Таймс.
Цифровая печать. Усл. печ. л. 1,39. Уч.-изд. л. 1,3.
Изд. № 110.
E-mail: ic@gstu.gomel.by
http://www.gstu.gomel.by

Издатель и полиграфическое исполнение:
Издательский центр учреждения образования
«Гомельский государственный технический
университет имени П. О. Сухого».
ЛИ № 02330/0131916 от 30.04.2004 г.
246746, г. Гомель, пр. Октября, 48.

1.