Фундаментальные концепции и принципы SQL

ГЛАВА

2

С тех пор как был принят первый стандарт языка SQL — SQL-89, — он не изменил своему исходному предназначению: быть стандартизированным, обобщенным, непроцедурным и независимым от производителей языком работы с реляционными базами данных. Он всегда следовал новым тенденциям, и последняя версия стандарта — SQL:2003 — в очередной раз это доказала.

В этой главе…

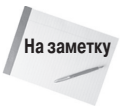
Цели и результаты

В отличие от многих популярных языков программирования, таких как C++, Java, Visual Basic и др., SQL создавался специально для работы с массивами данных и всегда был исключительно непроцедурным. Это значит, что из него были полностью исключены все конструкции управления потоком команд, такие как условные операторы (`IF... THEN`), циклы (`FOR...NEXT`) и т.п.

SQL предназначен для хранения информации, ее обработки и извлечения, и как таковой тесно связан с системами управления базами данных (СУБД). Его существование вне СУБД невозможно. В стандартной клиентской программе обычно требовалось послать запрос к базе данных и затем получить результат в форме либо набора данных, либо индикатора состояния.

В противоположность манипулированию переменными, свойственному стандартному программистскому стилю, вставка, обновление и извлечение данных — это *операции над множествами*. Инструкции SQL выполняют работу с наборами данных и, несмотря на продолжительное время выполнения отдельных операций, с точки зрения программиста, никакого *потока операций* не образуется. Кстати, механизм выполнения запросов SQL может быть внедрен и в процедурные языки, такие как C.

Большей частью программа SQL представляет собой всего одну инструкцию (или их пакет) независимо от того, какой она длины и сложности; эта инструкция либо выполняет всю задачу, либо не выполняет ее вообще.



Чтобы обойти проблему, связанную с дефицитом процедурных средств в SQL, производители СУБД дополнили язык в своих реализациях процедурными расширениями; в результате появились такие версии языка, как PL/SQL в Oracle, Transact-SQL в Microsoft SQL Server и SQL PL в IBM DB2. Последние разработки позволяют использовать языки высокого уровня, такие как Java и Visual Basic, внутри самой базы данных. И эта тенденция постепенно прокладывает свой путь в стандарт SQL (SQL/JRT и ISO/IEC 9075-13:2003).

Развитие SQL плотно переплеталось с парадигмой клиент/серверных вычислений еще задолго до того, как этот термин оказался у всех на слуху. *Клиент* должен знать, как подключиться к серверу, запросить данные и представить их пользователю в удобном формате. *Сервер* же должен "понять" запрос клиента и вернуть соответствующие данные (и кроме того, поддерживать эти данные в таком виде, чтобы обеспечить производительность обработки запросов, а также целостность данных и их защиту).

Сложность реализации на низком уровне (способы перевода запросов на машинный язык и их выполнение) скрыты под покровом простых и понятных инструкций, таких как `SELECT` (отобрать), `INSERT` (вставить) и `UPDATE` (обновить). Всю задачу преобразования этих инструкций в реальные компьютерные команды берет на себя СУБД.

В стандартах SQL определены строительные блоки инструкций SQL, однако совершенно ничего не сказано о том, как они должны быть реализованы. И это полностью развязывает руки создателям СУБД. Во-первых, они сами решают, какие из этих блоков стандарта SQL реализовывать в своих продуктах, а во-вторых, еще и по-разному интерпретируют эти блоки. Расширения, вводимые производителями СУБД, еще более усложняют данный вопрос (хотя порой эти расширения становятся основой для выхода следующей версии стандарта SQL).

Как уже говорилось в главе 1, стандарт ANSI SQL (SQL-89 и SQL-92) ввел три уровня соответствия стандарту ANSI: *минимальный*, *на уровне ядра* и *расширенный*. Следующая версия стандарта — SQL:1999 — заменила эти три уровня двумя: *базовым* и *расширенным*. В базовый уровень, определенный в новом стандарте SQL:2003, включено множество расширенных функций предыдущих стандартов в дополнение к совершенно новым средствам. Все основные производители СУБД выпускают продукты, соответствующие базовому уровню совместимости (правда, при этом не существует единой независимой экспертизы соответствия стандарту).

Документы стандартов ANSI/ISO

Следующие документы формируют основное ядро стандартов ANSI/ISO:

- **ISO/IEC 9075-1:2003/Cor 2:2007**: Information technology, Database languages, SQL, Part 1: Framework (SQL/Framework). (Информационные технологии, языки баз данных, SQL, часть 1: среда (SQL/среда)
- **ISO/IEC 13249-1:2007**: Information technology, Database languages, SQL multimedia and application packages, Part 1: Framework. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 1: среда.)
- **ISO/IEC 9075-2:2003/Cor 2:2007**: Information technology, Database languages, SQL, Part 2: Foundation (SQL/Foundation). (Информационные технологии, языки баз данных, SQL, часть 2: основы (SQL/Основы)
- **ISO/IEC 13249-2:2003**: Information technology, Database languages, SQL multimedia and application packages, Part 2: Full-Text. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 2: полнотекстовый поиск.)

04-ch02.indd 48 07.09.2009 12:42:12

- **ISO 19125-2:2004**: Geographic information, Simple feature access Part 2: SQL option. (Географическая информация, простой доступ к функциям, часть 2: вариант SQL.)
- **ISO/IEC 13249-3:2006**: Information technology, Database languages, SQL multimedia and application packages, Part 3: Spatial. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 3: пространственная информация.)
- **ISO/IEC 9075-3:2003/Cor 1:2005**: Information technology, Database languages, SQL, Part 3: Call-Level Interface (SQL/CLI). (Информационные технологии, языки баз данных, SQL, часть 3: интерфейс уровня вызовов (SQL/CLI).)
- **ISO/IEC 9075-4:2003/Cor 2:2007**: Information technology, Database languages, SQL, Part 4: Persistent Stored Modules (SQL/PSM). (Информационные технологии, языки баз данных, SQL, часть 4: постоянные хранимые модули (SQL/PCM).)
- **ISO/IEC 13249-5:2003**: Information technology, Database languages, SQL multimedia and application packages, Part 5: Still image. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 5: статический образ.)
- **ISO/IEC 13249-6:2006**: Information technology, Database languages, SQL multimedia and application packages, Part 6: Data mining. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 6: раскрытие данных.)
- **ISO/IEC CD 13249-7**: Information technology, Database languages, SQL multimedia and application packages, Part 7: History. (Информационные технологии, языки баз данных, пакеты мультимедиа и приложений, часть 7: история.)
- **ISO/IEC 9075-9:2003/Cor 1:2005**: Information technology, Database languages, SQL, Part 9: Management of External Data (SQL/MED). (Информационные технологии, языки баз данных, SQL, часть 9: управление внешними данными (SQL/MED).)
- **ISO/IEC 9075-10:2003/Cor 2:2007**: Information technology, Database languages, SQL, Part 10: Object Language Bindings (SQL/OLB). (Информационные технологии, языки баз данных, SQL, часть 10: интеграция с объектно-ориентированными языками программирования.)
- **ISO/IEC 9075-11:2003/Cor 2:2007**: Information technology, Database languages, SQL, Part 11: Information and Definition Schemas (SQL/Schemata). (Информационные технологии, языки баз данных, SQL, часть 11: информационные схемы и схемы определений (SQL/Schemata).)
- **ISO/IEC 9075-13:2003/Cor 1:2005**: Information technology, Database languages, SQL, Part 13: SQL Routines and Types Using the Java Programming Language (SQL/JRT). (Информационные технологии, языки баз данных, SQL, часть 13: процедуры и типы SQL, использующие язык программирования Java (SQL/JRT).)
- **ISO/IEC 9075-14:2006**: Information technology, Database languages, SQL, Part 14: XML-Related Specifications (SQL/XML). (Информационные технологии, языки баз данных, SQL, часть 14: спецификации, связанные с XML (SQL/XML).)
- **ISO/IEC 9579:2000**: Information technology, Remote database access for SQL with security enhancement. (Информационные технологии, удаленный доступ к базам данных с расширенной системой защиты.)

Все эти документы можно приобрести на сайтах `www.iso.org` и `www.ansi.org`.

SQL — в достаточной мере приспосабливающийся язык. Следующие его элементы оставлены на откуп производителям СУБД.

- **Семантические и синтаксические различия**.
- **Открытие баз данных для обработки**. Интерфейсы ODBC, OLEDB, JDBC и прочие не являются частью стандарта SQL, хотя в стандарте определен интерфейс уровня вызовов (CLI) (в документе ISI/IEC 9075-3:2003/Cor 1:2005).

04-ch02.indd 49 07.09.2009 12:42:12

- **Реализации динамического и внедренного SQL могут отличаться у разных производителей**.
- **Порядок сопоставления**. Способ представления результатов сортированного запроса. Все зависит от того, какие символы применяются: ASCII или EBCDIC. (Несмотря на то что стандарт UNICODE исключил эту проблему, не стоит сбрасывать со счетов гигантское количество старых данных и приложений.)
- **Разные расширения типов данных**. Пользовательские структуры, составные типы данных, объекты и т.д.
- **Различия в таблицах каталогов баз данных**. Несмотря на то что это нехарактерно для полного уровня соответствия стандарту, производители, работающие на базовом уровне совместимости, лишены стимула к избавлению от своих закрытых структур.

В табл. 2.1 перечислены ключевые области, в которые в стандарте SQL:2003 были внесены изменения или которые были добавлены в стандарт.

Таблица 2.1. Ключевые области SQL:2003

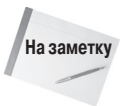
Функция	*Описание*
Интерфейс уровня вызовов (CLI)	Спецификация, определяющая доступ к базе данных посредством набора процедур, которые могут быть вызваны из клиентского приложения (посредством ODBC, JDBC и т.п.)
Информационная схема	Набор представлений для доступа к метаданным конкретной базы данных
Безопасность ролевого уровня	Парадигма системы защиты, определяющая способность точно подстраивать привилегии безопасности, группируя их в логически сопоставимые группы
Рекурсия	Вложенные взаимосвязи, необходимые для иерархических структур модели
Поддержка транзакций и точки сохранения	Способность выполнения групповых изменений данных по принципу “все или ничего”. Под *точкой сохранения* понимается один из методов повышения детализации транзакций, при котором откат может быть выполнен не к началу транзакции, а к некоторой ее именованной точке
Типы данных SQL `BLOB`, `CLOB`, `BOOLEAN`, `REF`, `ARRAY`, `ROW` и пользовательские типы	Новые типы данных, адаптированные к сложности современных вычислений (см. главу 3)
Типы данных мультимедиа: полнотекстовые, фотографии и пространственные	Новые форматы данных, специально созданные для элементов мультимедиа
SQL/XML	Текущий де-факто стандарт обмена данными. Он также стал стандартом и для офисных пакетов, таких как OpenOffice, Microsoft Office и многих других
SQL/MED	Расширения SQL, предназначенные для управления внешними данными посредством внешних таблиц и типов данных связей
SQL/PCM (постоянный хранимый модуль)	Область, управляемая частными процедурными расширениями производителей СУБД, такими как PL/SQL от Oracle, SQL PL от IBM и Transact-SQL от Microsoft. Определяет стандарты программирования управления потоком команд (конструкции `IF...THEN...ELSE`, циклы и др.)
Триггеры	Определение действий, автоматически выполняемых в ответ на одно из предопределенных событий. Новый стандарт позволяет точно настроить базовую функциональность триггеров
Возможности управления подключениями, сеансами, транзакциями и диагностикой	Инфраструктура, поддерживающая централизованную и распределенную обработку

04-ch02.indd 50 07.09.2009 12:42:12

SQL — живой язык; он продолжает развиваться и приспосабливаться к требованиям современной жизни. Несмотря на тенденции рынка в направлении стандартизации функций и методов обмена данными, производители по понятным причинам стремятся заблокировать пользователей в среде конкретных СУБД, завлекая их нестандартными функциями. Эти расширения порой существенно повышают эффективность выполнения процедур, но одновременно затрудняют, а иногда даже делают невозможным перенос программ SQL на другую платформу СУБД.

Любой производитель СУБД заинтересован в отправке своих новых идей в комитеты AMSI/ISO, которые их тщательно анализируют и, вполне вероятно, включат в следующий стандарт SQL. Язык SQL завтрашнего дня может оказаться совсем не таким, каким мы знаем его сегодня. Среди ждущих своего часа стандартов — расширяемый язык разметки XML и оперативная аналитическая обработка данных OLAP.

Язык XML подробно рассматривается в главе 15.



Все расширения, встроенные в конкретные реализации СУБД, потенциально могут быть отправлены в комитеты стандартизации и включены ими в официальный стандарт SQL.

Первое знакомство с SQL

На протяжении всей книги для примеров будет использоваться учебная база данных обработки заказов фиктивной строительной компании ACME. В приложении Б можно детально познакомиться с ее структурой, а в приложении Е — с процессом ее установки в тройке ведущих СУБД: Oracle 11g, DB2 9.5 и Microsoft SQL Server 2008. В этой главе мы проведем обзорную экскурсию по SQL на примере именно этой базы данных. Мы пытались сделать синтаксис как можно более обобщенным — по крайней мере, приемлемым для любой СУБД, рассматриваемой в этой книге. По этой причине предложенный синтаксис в ряде случаев может оказаться недостаточно эффективным в конкретной реализации СУБД.

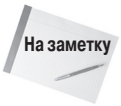
В языке SQL жизненный цикл любого объекта начинается с инструкции `CREATE`. Она используется для создания всех объектов, составляющих базу данных: таблиц, индексов, ограничений и др. Вопросы создания, изменения и уничтожения объектов базы данных подробно рассмотрены в главах 4 и 5.

Предполагая, что база данных и ее схема уже созданы, рассмотрение можно начать с инструкции `CREATE TABLE`. Ее синтаксис практически идентичен во всех трех реализациях СУБД и предполагает указание названий столбцов и их типов данных, что определяет тип информации, которая в дальнейшем будет храниться в таблице. Вот обобщенный пример такой инструкции:

```
CREATE TABLE status
(
  status_id_n INT,
  status_code_s CHAR(2),
  status_desc_s VARCHAR(30)
)
```

Эта инструкция, выполняемая в любой базе данных, приведет к созданию структуры, состоящей из пустой таблицы со столбцами `status_id_n`, `status_code_s` и `status_desc_s` и четко определенными для них типами данных.

04-ch02.indd 51 07.09.2009 12:42:12



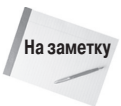
Типы данных будут подробно рассмотрены в главе 3.

Приведенная выше в качестве примера инструкция может быть повторена, однако имена таблиц в каждом случае должны быть разными (в пределах одной схемы). Каждый ее экземпляр будет знаменовать собой создание в базе данных новой таблицы. Таблица может существовать как обособленный объект, не имея никакой зависимости от других объектов базы данных. Однако чтобы получить полную отдачу от *реляционной* природы СУБД, нужно определить *отношения* между родительскими и дочерними объектами. Эти отношения между таблицами определяются с помощью *ограничений* внешнего ключа. Чтобы упростить обзорную экскурсию, мы пока не будем использовать ограничения (а также значения по умолчанию).

Ограничения подробно описаны в главе 4.

Для уничтожения таблицы (равно как и любого другого объекта базы данных) используют инструкцию `DROP`:

```
DROP TABLE status
```



На практике ограничения ссылочной целостности могут избавить программиста от необходимости явного уничтожения таблиц. Из главы 1 вы уже знаете, что именно делает базу данных реляционной в контексте ее целостности. В работе может потребоваться отключение некоторых ограничений или организация их в каскадную структуру. В главах 4 и 5 ссылочной целостности и ограничениям уделяется особое внимание.

База данных — это нечто большее, нежели обычный набор таблиц. Существует гораздо больше объектов, равно как и связанных с ними процессов и структур, поставленных на службу данным; однако в настоящий момент уместно обрисовать только несколько укрупненную, общую картину происходящего в базе данных.

Получение данных и их отправка

После создания таблицы вполне естественно наполнить ее информацией — в конце концов, для чего же создавалась база данных? В SQL определены четыре основные инструкции, которые предназначены для извлечения информации из базы данных и дальнейшей работы с этой информацией (табл. 2.2). Более подробно эти инструкции будут рассмотрены в главах 6–9.

Таблица 2.2. Четыре основные инструкции SQL

Инструкция SQL	*Назначение*
`INSERT`	Вставка в таблицу новой строки
`UPDATE`	Изменение в таблице существующих значений
`SELECT`	Извлечение данных из таблицы
`DELETE`	Удаление данных из таблицы

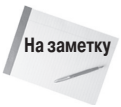
В действительности SQL составляют четыре разных языка: язык определения данных DDL (см. главы 4 и 5), язык обработки данных DML (глава 6), язык запросов к данным DQL (главы 8 и 9) и язык управления данными (глава 14).

04-ch02.indd 52 07.09.2009 12:42:12

К примеру, чтобы добавить новую запись о состоянии в таблицу status, созданную в предыдущем примере, используется следующая инструкция:

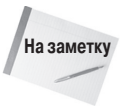
```
INSERT INTO status
  (STATUS_ID_N, STATUS_CODE_S, STATUS_DESC_S)
  VALUES (8,'70','INVOICED')
```

Эта инструкция состоит из ключевого слова INSERT, за которым следует имя таблицы и список ее столбцов, дополняемый списком соответствующих им значений.



В данной инструкции можно перечислять не все поля, а только их подмножество. В приведенном примере недостающим полям будут присвоены значения по умолчанию. (СУБД всегда автоматически заполняет недостающие данные предопределенными значениями.) В инструкции может быть опущен и сам список столбцов, если значения заданы для всех полей и расположены в правильном порядке.

Эта инструкция может быть введена в любой утилите доступа к СУБД (о таких программах речь пойдет в приложении Д). Выполнение инструкции база данных сопровождает откликом о количестве обработанных полей или сообщением об ошибке в случае неудачной операции.



Значения для полей STATUS_CODE_S и STATUS_DESC_S заключены в одинарные кавычки, так как эти столбцы имеют символьный тип данных. Значение числового поля STATUS_ID_N в таких кавычках не нуждается.

Если стоит задача изменения некоторых существующих данных (к примеру, нужно изменить код состояния утверждения, а все остальные данные оставить без изменения), применяется инструкция UPDATE. И в этом случае синтаксис инструкции единообразен во всех трех реализациях СУБД, описываемых в книге: Oracle 11g, IBM DB 9.5 и Microsoft SQL Server 2008.

```
UPDATE status
  SET status_desc_s = 'APPROVED'
  WHERE status_id_n = 8
```

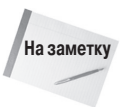
В приведенном примере представлено вездесущее предложение WHERE, ограничивающее число обрабатываемых строк (в данном случае до одного заказчика с идентификатором 8). Если это предложение в запросе опущено, результатом станет замена значения упомянутого поля во всех строках таблицы (в данном случае у всех заказчиков).

То же относится и к удалению данных. Если нужно удалить из базы данных запись об определенном заказчике, можно выполнить следующую инструкцию:

```
DELETE status
  WHERE status id n = 8
```

И в этом случае если из инструкции удалить предложение WHERE, уничтожены будут все записи таблицы (т.е. сведения обо всех заказчиках). Обычно СУБД оснащены встроенным механизмом восстановления удаленных данных, но это совершенно не значит, что не нужно внимательно относиться к операциям, выполняемым над данными.

Инструкция SELECT в своем базовом виде извлекает данные из таблицы или представления. В этой инструкции нужно указать список полей, которые следует включить в *результирующий набор данных*, или заменить его символом звездочки, что указывает на извлечение всех полей.



Представлением называют виртуальную таблицу, заполняемую в момент выполнения встроенного в нее запроса. Представления подробно описываются в главе 4.

04-ch02.indd 53 07.09.2009 12:42:13

В следующем сценарии из представления V_CUSTOMERS_TOTAL извлекаются столбцы CUSTOMER_NAME, ORDER_NUMBER и TOTAL_PRICE.

```
SELECT customer_name,
       order_number, total_price
FROM v_customer_totals
```

```
customer_name            order_number     total_price
-----------------------  ---------------- -----------
WILE BESS COMPANY        523720           7511.00
WILE BESS COMPANY        523721           8390.00
WILE BESS COMPANY        523722           6608.00
WILE BESS COMPANY        523723           11144.00
WILE ELECTROMUSICAL INC. 523726           6608.00
WILE ELECTROMUSICAL INC. 523727           6608.00
WILE ELECTROMUSICAL INC. 523728           6608.00
```

Инструкция SELECT может быть очень сложной и извлекать данные из множества таблиц; дополнительно могут вноситься некоторые коррективы в значения и порядок возвращаемых данных. Подробно инструкция SELECT описана в главах 8 и 9.

Срезы: одни и те же данные под разными углами зрения

Существует множество разных точек зрения на одни и те же данные. SQL предоставляет средства манипуляций возвращаемыми в клиентское приложение данными. Данные можно упорядочить по возрастанию и убыванию по практически любому столбцу таблицы; при возвращении данных можно выполнить некоторые вычисления, а также можно ввести ограничения на отображение возвращаемых данных. Вот несколько примеров, демонстрирующих эти возможности.

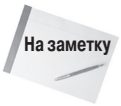
Базовый запрос SELECT возвращает результирующий набор данных, соответствующий заданному критерию, взяв за основу лишь существующие данные. А что делать, если нужно увидеть, к примеру, общий объем продаж, но за вычетом федерального налога? В запрос можно встроить довольно сложные вычисления. Следующий пример основан на представлении V_CUSTOMER_TOTALS, содержащем столбцы CUSTOMER_NAME, ORDER_NUMBER и TOTAL_PRICE.

Предположим, что федеральный налог составляет 8,5%, тогда запрос может выглядеть следующим образом:

```
SELECT customer_name, order_number,
       (total_price-(total_price * 0.085)) net_sale
FROM v customer totals
```

```
customer_name            order_number       net_sale
-----------------------  ------------------ --------
WILE BESS COMPANY        523720             6872.56
WILE BESS COMPANY        523721             7676.85
WILE BESS COMPANY        523722             6046.32
WILE BESS COMPANY        523723             10196.76
WILE ELECTROMUSICAL INC. 523726             6046.32
WILE ELECTROMUSICAL INC. 523727             6046.32
WILE ELECTROMUSICAL INC. 523728             6046.32
```


04-ch02.indd 54 07.09.2009 12:42:13



По умолчанию каждый столбец в возвращаемых результатах имеет свое собственное, изначальное имя. В случае вычисляемых столбцов, как в приведенном выше примере, СУБД использует в качестве его имени все выражение. Чтобы результаты были более понятны, выражение можно дополнить описательным псевдонимом (в данном случае `NET_SALE`).

В результаты, возвращаемые запросом, включены вычисляемые значения — другими словами, исходные данные были трансформированы. SQL предлагает несколько функций, которые могут быть использованы в запросе для преобразования данных после их извлечения, некоторые производители СУБД расширяют возможности преобразования данных собственными функциями.

В главе 10 представлено исчерпывающее описание наиболее популярных функций и особенности их использования в продуктах крупнейших производителей СУБД (Microsoft, IBM и Oracle). В приложении Ж предлагается практически полный список функций, реализованных на сегодняшний день производителями СУБД.

С помощью SQL можно в полной мере управлять отображением результирующих данных. Их можно упорядочить по любому столбцу, как по алфавиту, так и по числовым значениям. Предположим, что нам нужно вывести список всех компаний, упорядочив их по общей сумме конкретных заказов.

```
SELECT customer_name, order_number,
       (total_price-(total_price * 0.085)) net_sale
FROM v_customer_totals
ORDER BY net_sale
```

```
customer_name            order_number   net_sale
------------------------ -------------- --------
WILE ELECTROMUSICAL INC. 523726         6046.32
WILE ELECTROMUSICAL INC. 523727         6046.32
WILE ELECTROMUSICAL INC. 523728         6046.32
WILE BESS COMPANY        523722         6046.32
WILE BESS COMPANY        523720         6872.56
WILE BESS COMPANY        523721         7676.85
WILE BESS COMPANY        523723         10196.76
```

А теперь предположим, что нужно создать такой же отчет, но в его верхней части должны размещаться наибольшие, а не наименьшие заказы. В этом случае упорядочение нужно выполнять не по возрастанию (как положено по умолчанию), а по убыванию. Придется воспользоваться модификатором `DESC`.

```
SELECT customer_name, order_number,
      (total_price-(total_price * 0.085)) net_sale
FROM v_customer_totals
ORDER BY net_sale DESC
```

```
customer_name            order_number   net_sale
------------------------ -------------- --------
WILE BESS COMPANY        523723         10196.76
WILE BESS COMPANY        523721         7676.85
WILE BESS COMPANY        523720         6872.56
WILE BESS COMPANY        523722         6046.32
WILE ELECTROMUSICAL INC. 523726         6046.32
WILE ELECTROMUSICAL INC. 523727         6046.32
WILE ELECTROMUSICAL INC. 523728         6046.32
```


04-ch02.indd 55 07.09.2009 12:42:13

Консолидация

С помощью SQL можно не только преобразовывать данные при извлечении, но и консолидировать их, т.е. заменять группу полей, сгруппированных по какому-либо критерию, основанному на одном столбце или их группе, одним значением. К примеру, таким образом можно получить общий объем продаж. Предположим, что в представлении V_CUSTOMER_TOTALS содержатся сводные данные о заказах каждого из клиентов компании. Требуется узнать общий объем продаж, т.е. сумму всех заказов.

Для этой задачи отлично подойдет встроенная SQL-функция SUM. Все, что нам нужно, — это просуммировать значения полей TOTAL_PRICE представления V_CUSTOMER_TOTALS.

```
SELECT SUM(total_price) net_sale_total
FROM v_customer_totals

net_sale_total
--------------
457000.40
```

Аналогичным образом можно найти и среднюю сумму заказа, на этот раз использовав встроенную функцию AVG.

```
SELECT AVG(total_price) net_sale_average
FROM v_customer_totals

net_sale_average
----------------
8960.792156
```

На практике данный запрос обычно ограничивают конкретным заказчиком или определенным диапазоном дат.

С помощью других предложений, GROUP_BY и HAVING, можно просуммировать значения поля результирующего набора данных NET_SALE по заказчикам, датам, товарам и т.д. Группировка позволяет выполнять консолидацию в каждой из групп. (Все эти предложения более подробно рассмотрены в главе 6, посвященной инструкции SELECT.)

Приведенные выше примеры позволяют получить общее представление о том, что можно сделать в базе данных с помощью SQL.

О функциях SQL, в том числе итоговых, более подробно рассказывается в главе 10.

Защита данных

Несмотря на то что львиная доля средств защиты данных встроена в инфраструктуру СУБД, SQL также не остался от этого вопроса в стороне. Механизм защиты, предлагаемый SQL, многоуровневый, при этом степень детализации большей частью зависит от конкретной реализации СУБД. В сущности, все сводится к управлению правами доступа к различным объектам базы данных и к выполнению конкретных операций. К примеру, некоторому пользователю можно открыть доступ к конкретной таблице или к выполнению определенной инструкции (скажем, SELECT), но запретить добавление данных в таблицу (т.е. использовать инструкцию INSERT).

04-ch02.indd 56 07.09.2009 12:42:13

Привилегии

Предполагая, что пользователь JOHN_DOE определен в базе данных, для предоставления ему права использовать инструкцию SELECT в представлении V_CUSTOMER_TOTALS можно воспользоваться следующим оператором.

```
GRANT SELECT
ON v_customer_totals
TO john_doe
```

Для предоставления ему прав использования двух инструкций, SELECT и UPDATE, применяется следующий синтаксис:

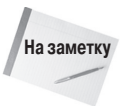
```
GRANT SELECT, UPDATE
ON v_customer_totals
TO john_doe
```

А теперь отзовем привилегию на инструкцию SELECT:

```
REVOKE SELECT
ON v_customer_totals
FROM john_doe
```

И наконец, мы отбираем у него все привилегии:

```
REVOKE ALL
ON v_customer_totals
FROM john doe
```



В конкретных реализациях существуют и другие инструкции управления привилегиями. К примеру, в языке Transact-SQL от Microsoft введена инструкция DENY, позволяющая настроить всеразрешающую инструкцию GRANT. В главе 12 вопрос системы безопасности рассматривается более детально.

Представления

Один из самых популярных механизмов обеспечения защиты информации предлагают представления. *Представление* — это средство ограничения данных, доступных пользователю. Представление можно охарактеризовать как виртуальную таблицу — оно позволяет объединять столбцы из разных таблиц базы данных, ограничивать состав столбцов, доступных для просмотра, и т.д. На самом деле представление не содержит данных — оно извлекает их с помощью прикрепленной инструкции SELECT. В наиболее практичных приложениях извлечение данных из представлений эффективно заменяет собой извлечение данных из таблиц.

К примеру, в представление V_CUSTOMER_TOTALS собрана информация из таблиц CUSTOMER, ORDER_HEADER, ORDER_LINE и PRODUCT, по ходу сгруппированная и консолидированная некоторым образом.

Полный синтаксис создания представления V_CUSTOMER_TOTALS можно найти в приложении Б.

В некоторых представлениях доступ к данным может быть ограничен (к примеру, может быть запрещено использование инструкций UPDATE и INSERT). Представления более подробно описаны в главе 4.

04-ch02.indd 57 07.09.2009 12:42:13

О защите информации можно найти гораздо больше сведений, чем представлено в этой книге. К примеру, во всех трех реализациях обсуждаемых здесь СУБД реализован механизм защиты, основанный на ролях. В нем каждому пользователю назначается некоторая роль (к примеру, бухгалтера), а привилегии предоставляются уже ролям.

Исчерпывающую информацию по вопросам защиты в SQL вы найдете в главе 12.

Доступ к данным из клиентского приложения

Многие клиентские приложения предназначены для работы с данными СУБД. Все они используют для этого SQL, но с помощью двух радикально отличающихся способов.

Внедренный SQL позволяет создавать программы, которые осуществляют доступ к базам данных с помощью инструкций SQL, *внедренных* в какой-либо стандартный язык программирования с поддержкой стандартов ANSI/ISO (такой как C или COBOL). Это значит, что приложение программируется на стандартном языке высокого уровня, а переключение на SQL выполняется, только когда необходимо обратиться к информации в базе данных. Обычно производители СУБД выпускают специальные пакеты функций, упрощающие создание таких приложений.

В *динамическом* SQL есть все то же, что и во внедренном. Главное отличие состоит в том, что динамический запрос SQL не встраивается изначально в программу, а формируется динамически в процессе ее выполнения, при этом запросы передаются в СУБД как обычный текст. Это обеспечивает программистам гибкость, которую встроенный SQL не может предоставить по определению. В динамическом SQL нет необходимости жестко прошивать в программе имена таблиц, столбцов и даже баз данных — все это можно формировать в процессе выполнения.

Внедренный и динамический SQL подробно обсуждаются в главе 16.

Новые разработки

Стандарт SQL:2003 также отражает концепцию, которая эволюционирует уже на протяжении нескольких лет, — *оперативную аналитическую обработку данных* (OLAP). Эту концепцию нельзя назвать новой. Совет OLAP был организован в 1995 году (до этого времени бо́льшая часть аналитики выполнялась вручную). Эта тема вновь стала популярной с появлением *хранилищ данных* (еще одна новая концепция, связанная с базами данных).

OLAP извлекает из статичных данных смысловую информацию, выявляя сложные тенденции и позволяя оценивать данные с разных ракурсов. Накапливаемые данные преобразуются в многомерный объект, подогнанный под различные запросы пользователя. Запросы OLAP позволяют получить ответы на более интеллектуальные вопросы, чем обычные запросы SQL. Примером такого вопроса может быть следующий: “Как отразится на объеме продаж повышение цен на сахар на 10%?”. Такие задачи до появления стандарта SQL:1999 обычно решались с помощью доморощенных приложений. Продукты OLAP дополняют СУБД, и все три основных производителя СУБД (Microsoft, Oracle и IBM) предлагают свои решения. В целом на рынке около трех десятков производителей предлагают свои решения OLAP на базе реляционных баз данных.

Еще одним относительно новым средством, поддерживаемым SQL, является расширяемый язык разметки XML, предназначенный для обмена данными. Его часто называют *самоописательным форматом*, так как в нем данные представлены в иерархической структуре, которая

04-ch02.indd 58 07.09.2009 12:42:13

в совокупности с расширенным языком стилей XSL обеспечивает визуальное представление в браузере или служит средством обмена данными между разными платформами.

Открытый стандарт, не запатентованный ни одной компанией, сегодня используется практически всеми производителями, служа, без преувеличения, универсальным форматом обмена данными. Одно из главных достоинств XML заключается в способности передаваться через протокол HTTP (основной протокол Интернета), что сделало сети закрытого типа устаревшим явлением. Данные XML можно зашифровать и передавать по протоколу SSL; XML стал краеугольным камнем веб-служб, основанных на новой парадигме — архитектуре, ориентированной на службы (*SOA*). Однако эта универсальность имеет свою цену. К примеру, текстовый XML-код медленнее, чем скомпилированный код, так как текст должен каждый раз подвергаться синтаксическому анализу и интерпретироваться. Как бы там ни было, такие расширения, как XPath и XQuery, позволили устранить ряд недостатков. И поскольку тройка крупнейших производителей СУБД, обсуждаемых в настоящей книге, встраивают поддержку XML в свои продукты, он быстро стал популярным способом хранения полуструктурированной информации.

Любая платформа, любое время

SQL отличается от стандартных языков программирования, таких как C/C#, Pascal, VB.NET или Java, поскольку он не может быть использован для создания обособленных приложений. Он не может существовать вне ядра базы данных, которое необходимо для преобразования инструкций в выполняемые машинные команды. В SQL отсутствуют программные конструкции, которыми наполнены все остальные языки, в частности операторы условной логики и циклы, также в нем не используются переменные.

Несмотря на то что этот дефицит может быть восполнен за счет использования функций SQL, процедурных расширений и объектно-ориентированных средств, SQL никогда не претендовал на роль многофункционального языка программирования. В то же время эта слабость является ключом к его повсеместному распространению. Для выполнения инструкций языка необходима некоторая СУБД (подобно тому, как программам на языке Java нужна специфичная для платформы виртуальная машина), но SQL можно справедливо назвать первым платформно-независимым языком.

Любая программа SQL создается путем ввода команд в текстовом формате. После этого команды компилируются в двоичный машинный код и исполняемые файлы различных языков программирования. Основная проблема данного подхода состоит в том, что такая программа для каждой платформы должна быть перекомпилирована. К примеру, программа, скомпилированная для Microsoft Windows, не будет работать в системе UNIX, и наоборот. Решение этой проблемы, предложенное языком Java, все равно требует наличия специфичной для конкретной платформы виртуальной машины JVM.

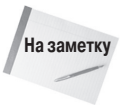
Запросы SQL создаются практически так же, как и программы на других языках, — путем ввода ключевых слов. Однако на этом сходство заканчивается. Программы SQL сохраняются в текстовом файле ASCII, который может быть скопирован без изменений в другую операционную систему, такую как UNIX, Windows, MAC OS, Linux и т.д. Более того, такой файл можно открыть, изменить и повторно сохранить в этой операционной системе с помощью стандартных средств редактирования — при этом не следует принимать в расчет специфичные для конкретной платформы соглашения. Так как программа на языке SQL всегда остается сценарием, ее легко переносить с одной платформы в другую (но не между разными СУБД). В некотором смысле СУБД работает подобно виртуальной машине Java. Даже если инструкции SQL внедряются в язык высокого уровня, такой как C, они остаются не более чем текстом. Так же как веб-страницы на языке HTML, представляющие собой набор символов ASCII, интерпретируются

04-ch02.indd 59 07.09.2009 12:42:13

для отображения веб-браузером, программы SQL для выполнения должны быть отправлены в некоторую СУБД, преобразующую текстовые инструкции в машинный код. Всем этим программам нужны для выполнения свои специфичные интерпретаторы.

Одной из главных платформно-независимых особенностей языка SQL является набор базовых типов данных: в какой бы операционной системе ни выполнялась программа — UNIX или Windows, 32- или 64-разрядной, — размер и структура резервируемых блоков хранения информации остаются неизменными. Тип данных `INTEGER` всегда будет занимать 4 байта, а `DOUBLE` — 8 байт (правда, существуют и дополнительные типы данных, свойственные конкретным реализациям СУБД). В противоположность этому, типы данных обычных языков программирования в разных системах резервируют разные объемы памяти (к примеру, тип `INTEGER` в 32-разрядных системах занимает 4 байта, а в 64-разрядных — восемь), так как при этом используются разные компиляторы. (О типах данных более подробно будет рассказано в главе 3.)

В некотором смысле принцип независимости от платформы SQL сходен с используемым в языке Java: у первого в роли интерпретатора текстовых команд выступает СУБД, а у второго — виртуальная машина JVM. Так, программа, набранная в системе Windows с помощью программы Блокнот, может быть выполнена в СУБД Oracle, установленной в операционной системе UNIX или Linux.



СУБД Oracle 11g доступна для следующих платформ:

- IBM AIX 5L версий 5.2 и 5.3 (64-разрядные);
- HP OpenVMS Alpha 8.2 и Itanium 8.2-1;
- HP Tru64 UNIX V5.1B;
- HP-UX Itanium 11i v1 (11.11), 11i V2 (11.23) и PA RISC (64-bit);
- IBM z/OS (OS/390) V1.4 и z/OSe V1.4 или выше;
- IBM zSeries Linux: Red Hat Enterprise Linux (RHEL) 4.0 (Update 2 или новее) и SUSE Linux Enterprise Server (SLES) 9 (Service Pack 2 или выше);
- Linux (Itanium, POWER, x86 и x86-64): Red Hat Enterprise Linux AS/ES 3.0 (Update 4 или новее); Red Hat Linux 4.0 (Update 1 или новее); SUSE Linux Enterprise Server 8.0 с установленным пакетом SP4 или выше; SUSE Linux Enterprise Server 9.0 с установленным пакетом SP2 или выше и Asianux 1.0/2.0;
- Microsoft Windows 2000 с установленным пакетом SP1 или выше, все редакции, включая Terminal Services и Microsoft Windows 2000 MultiLanguage Edition (MLE); все редакции Windows Server 2003; Windows XP Professional; редакции Business, Enterprise и Ultimate системы Windows Vista;
- Sun Solaris Operating System: 64-разрядная SPARC (Solaris 8 Update 7 или новее; Solaris 9 Update 6 или новее; Solaris 10), а также x86 и x86-64 (Solaris 10).

Различные редакции IBM DB2 9.5 поддерживаются следующими ОС:

- Microsoft Windows: Windows XP Professional и Winndows XP Professional x64; Windows 2003 редакций Standard, Enterprise и Datacenter (как 32-, так и 64-разрядные); Windows Vista; VMWare и Microsoft Virtual Server 2005;
- Linux: Novell Open Enterprise Server 9, Red Hat Enterprise (RHEL) 4 и SUSE Linux Enterprise Server (SLES) 9 и 10;
- IBM AIX: версии 5.2 и 5.3;
- HP-UX: 11iv2 (11.23.0505) операционные системы с пакетом Base Quality (QPKBASE) от мая 2005 года, пакетом Applications Quality (QPAPPS) от мая 2005 года, и заплаткой PHNE 32606 (требуется 64-разрядное серверное ядро HP-UX);

04-ch02.indd 60 07.09.2009 12:42:13

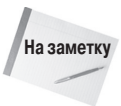
- Sun Solaris: Solaris 10 (UltraSPARC и x86-64 EM64 или AMD64); или Solaris 9 (только UltraSPARC,требуется 64-разрядное ядро и заплатки 111711-12 и 11712-12)

СУБД Microsoft SQL Server 2008 (различные редакции) доступна для системы Microsoft Windows следующих версий:

- Windows Server 2003 64-Bit x64 SP1;
- Windows Server 2003 64-Bit x64 Enterprise Edition SP1;
- Windows Server 2003 64-Bit Itanium SP1;
- Windows Server 2003 64-Bit Itanium Enterprise Edition SP1;
- Windows XP Professional 64-Bit x64 SP2;
- Windows Vista 64-Bit x64 Business Edition;
- Windows Vista 64-Bit x64 Enterprise Edition.

Как уже отмечалось, между разными диалектами SQL существует множество различий. К примеру, запрос, написанный для Microsoft SQL Server 2008, может не быть выполнен в IBM DB2, и наоборот. В то же время запрос, написанный для Oracle, установленной в системе Windows, не потребует изменений, чтобы быть выполненным в Oracle, установленной в UNIX или Linux. Это значит, что, если в запросе использованы только ключевые слова и средства, общие для всех трех СУБД, он в неизменном виде может быть выполнен в любой из этих СУБД.

Существуют десятки других СУБД, написанных для самых разнообразных платформ, такие как Teradata, Sybase, Ingres, Informix, Empress, MySQL, mSQL, PostgreSQL, LEAP RDBMS, FirstBase, Ocelot, Progress, Typhoon, SQL/DS, Daffodil DB, Cloudscape, Compaq Non-Stop SQL MX и SQL/MP, Linter RDBMS SQL, Interbase, Universe и GNU SQL Server. Продолжают появляться и новые разработки.



Несмотря на то что подавляющее большинство производителей стремятся сделать свои продукты совместимыми со стандартами ANSI, небольшой сегмент рынка занимают СУБД, использующие язык, отличный от SQL.

Каждая из этих баз данных имеет свой диалект SQL, однако подавляющее большинство из них совместимы со стандартом ANSI. Базовые выражения SQL практически без изменений могут быть выполнены в любой из этих СУБД, так что знание языка SQL можно применить вкаждой их этих систем.

Резюме

Язык SQL нельзя назвать обычным многофункциональным языком программирования — он создавался специально для работы с наборами взаимосвязанных данных.

Существует стандарт SQL, в целом поддерживаемый всеми крупными производителями СУБД. Несмотря на то что в каждой из СУБД имеется свой диалект этого языка, SQL можно назвать своеобразным эсперанто современного мира баз данных.

С помощью SQL можно создавать и удалять множество разнообразных объектов базы данных; также он позволяет вставлять, обновлять и удалять данные.

В клиентских приложениях используют два различных подхода к работе с базами данных: внедренный SQL и динамический SQL. Каждый из этих подходов нашел свое применение, в то же время синтаксис выражений SQL остается без изменений.

OLAP — относительно новое направление в SQL, сопутствующее стремительно развивающейся аналитической обработке бизнес-информации, необходимой для поддержки производственных процессов и принятия решений.

04-ch02.indd 61 07.09.2009 12:42:13

XML вводился как стандартный формат обмена данными и был поддержан практически всеми производителями программного обеспечения. Некоторые реализации СУБД (в частности, ведущая тройка, обсуждаемая в настоящей книге) уже имеют встроенную поддержку XML — открытого стандарта описания структурированных данных. XML становится стандартом де-факто для обмена данными между предприятиями, особенно через каналы Интернета.

04-ch02.indd 62 07.09.2009 12:42:14