

Содержание

Об авторе	16
Введение	17
Цель книги	17
Для кого предназначена эта книга	18
Пиктограммы, используемые в книге	18
Что дальше	19
Ждем ваших отзывов!	20
Часть 1. Знакомство с SQL	21
Глава 1. Основы реляционных баз данных	23
Обработка данных	24
Что такое база данных	25
Размер и сложность базы данных	26
Что такое СУБД	26
Плоские файлы	27
Модели баз данных	29
Реляционная модель	30
Компоненты реляционной базы данных	30
Отношения	31
Представления	32
Схемы, домены и ограничения	35
Объектная модель бросает вызов реляционной	36
Объектно-реляционная модель	37
Вопросы проектирования баз данных	38
Глава 2. Основы SQL	39
Что такое SQL	40
Немного истории	41
Инструкции SQL	43
Ключевые слова	44
Типы данных	45
Целочисленные типы	45
Числа с плавающей запятой	48
Символьные строки	50
Двоичные строки	52
Логические значения	53
Значения даты и времени	53

Интервалы	55
Тип XML	56
Тип ROW	58
Типы коллекций	59
Типы REF	61
Пользовательские типы	62
Перечень типов данных	65
Пустые значения	67
Ограничения	68
Использование SQL в архитектуре “клиент/сервер”	68
Сервер	69
Клиент	70
Использование SQL в Интернете и локальной сети	71
Глава 3. Компоненты SQL	73
Язык определения данных	74
Когда “Просто сделай это!” — не лучший совет	74
Создание таблиц	75
Создание представлений	77
Объединение таблиц в схемы	84
Заказ по каталогу	85
Инструкции DDL	85
Язык манипулирования данными	87
Выражения	88
Предикаты	91
Логический оператор	92
Итоговые функции	93
Подзапросы	95
Язык управления данными	96
Транзакции	96
Пользователи и привилегии доступа	97
Ограничения ссылочной целостности угрожают вашим данным	100
Делегирование ответственности за безопасность	102
Часть 2. Использование SQL для создания баз данных	105
Глава 4. Создание простой базы данных	107
Создание простой базы данных с помощью инструмента быстрой разработки	108
Что хранить в базе данных	108
Создание таблицы базы данных	109
Изменение структуры таблицы	116
Создание индекса	117

Удаление таблицы	120
Создание таблицы POWER средствами SQL	121
Создание SQL-запросов в Microsoft Access	122
Создание таблицы	124
Создание индекса	128
Изменение структуры таблицы	129
Удаление таблицы	129
Удаление индекса	130
Переносимость	130
Глава 5. Создание многотабличной базы данных	131
Проектирование базы данных	132
Шаг 1: определение объектов	132
Шаг 2: идентификация таблиц и столбцов	132
Шаг 3: точное определение таблиц	134
Домены, символьные наборы, схемы сортировки и трансляции	138
Ускорение работы базы данных с помощью ключей	138
Работа с индексами	141
Что такое индекс	142
Зачем нужен индекс	143
Поддержка индекса	144
Обеспечение целостности данных	145
Логическая целостность	146
Доменная целостность	147
Ссылочная целостность	147
Когда кажется, будто все безопасно	151
Потенциальные проблемы	152
Ограничения	154
Нормализация базы данных	157
Аномалии изменения и нормальные формы	158
Первая нормальная форма	160
Вторая нормальная форма	161
Третья нормальная форма	163
Доменно-ключевая нормальная форма (ДКНФ)	163
Денормализация	165
Часть 3. Хранение и извлечение данных	167
Глава 6. Манипулирование содержимым базы данных	169
Извлечение данных	170
Создание представлений	172
Создание представлений из таблиц	173
Создание представления с условием отбора	174

Создание представления с модифицированным атрибутом	175
Обновление представлений	176
Добавление новых данных	177
Добавление данных в виде отдельных записей	178
Добавление данных только в выбранные столбцы	179
Добавление группы строк в таблицу	180
Обновление существующих данных	183
Перенос данных	186
Удаление устаревших данных	188
Глава 7. Обработка темпоральных данных	191
Моменты и периоды времени	192
Таблицы с периодами прикладного времени	194
Назначение первичных ключей в таблицах с периодами прикладного времени	196
Применение ограничений ссылочной целостности к таблицам с периодами прикладного времени	197
Создание запросов к таблицам с периодами прикладного времени	198
Работа с системно-версионными таблицами	199
Назначение первичных ключей в системно-версионных таблицах	202
Применение ограничений ссылочной целостности к системно-версионным таблицам	202
Создание запросов к системно-версионным таблицам	203
Отслеживание данных с помощью битемпоральных таблиц	204
Форматирование и анализ значений даты и времени	205
Глава 8. Обработка значений	207
Значения	208
Записи	208
Литералы	209
Переменные	210
Специальные переменные	212
Ссылки на столбцы	213
Выражения	215
Строковые выражения	215
Числовые выражения	216
Выражения со значениями даты/времени	216
Интервальные выражения	217
Условные выражения	218
Функции	218
Статистические вычисления с помощью итоговых функций	218
Функции преобразований	222
Табличные функции	236

Глава 9. Использование сложных выражений	239
Условные выражения CASE	240
Использование выражения CASE с условиями отбора	241
Использование выражения CASE со значениями	243
Специальное выражение CASE — NULLIF	245
Еще одно специальное выражение CASE — COALESCE	247
Преобразование типов данных с помощью выражения CAST	248
Использование выражения CAST в SQL	249
Использование выражения CAST при взаимодействии SQL и языка приложения	250
Выражения с записями	251
Глава 10. Выбор нужных данных	253
Уточняющие предложения	254
Предложение FROM	255
Предложение WHERE	256
Предикаты сравнения	257
Предикат BETWEEN	258
Предикаты IN и NOT IN	259
Предикаты LIKE и NOT LIKE	261
Предикат SIMILAR	263
Предикат NULL	263
Предикаты ALL, SOME и ANY	264
Предикат EXISTS	267
Предикат UNIQUE	268
Предикат DISTINCT	268
Предикат OVERLAPS	269
Предикат MATCH	270
Правила ссылочной целостности и предикат MATCH	271
Логические операторы	274
AND	274
OR	275
NOT	276
Предложение GROUP BY	276
Предложение HAVING	279
Предложение ORDER BY	280
Использование инструкции FETCH для ограничения выборки	281
Использование оконных функций для создания результатирующего множества	283
Разделение окна на фрагменты с помощью функции NTILE	284
Навигация в пределах окна	285

Вложение оконных функций	287
Выполнение расчетов по группам записей	288
Распознавание шаблона записи	288
Глава 11. Использование реляционных операторов	291
Оператор UNION	292
Оператор UNION ALL	294
Оператор UNION CORRESPONDING	294
Оператор INTERSECT	295
Оператор EXCEPT	297
Табличные объединения	298
Простое объединение	298
Объединение по равенству	300
Перекрестное объединение	302
Естественное объединение	303
Условное объединение	303
Объединение по именам столбцов	304
Внутреннее объединение	305
Внешнее объединение	306
Объединение со слиянием	310
Предложения ON и WHERE	317
Глава 12. Вложенные запросы	319
Назначение подзапросов	321
Вложенные запросы, возвращающие наборы строк	321
Вложенные запросы, возвращающие одно значение	325
Использование подзапросов вместе с предикатами ALL, SOME и ANY	328
Вложенные запросы как средство проверки на существование	330
Другие коррелированные подзапросы	332
Инструкции UPDATE, DELETE и INSERT	336
Регистрация изменений с помощью конвейерных DML-операций	339
Глава 13. Рекурсивные запросы	341
Что такое рекурсия	341
Хьюстон, у нас проблема	342
Сбой недопустим	342
Что такое рекурсивный запрос	344
Где можно применить рекурсивный запрос	345
Решение “в лоб”	346
Экономия времени с помощью рекурсивного запроса	348
Где еще можно использовать рекурсивные запросы	350

Часть 4. Управление операциями	353
Глава 14. Безопасность базы данных	355
Язык управления данными	356
Уровни доступа пользователей	356
Администратор базы данных	357
Владельцы объектов базы данных	358
Открытый доступ	358
Предоставление полномочий пользователям	359
Роли	360
Вставка данных	361
Просмотр данных	361
Модификация табличных данных	362
Удаление устаревших строк из таблицы	363
Использование ссылок на связанные таблицы	363
Использование доменов	364
Запуск инструкций SQL	366
Предоставление уровневых полномочий	367
Право на предоставление полномочий	368
Отзыв полномочий	369
Совместное использование инструкций GRANT и REVOKE	371
Глава 15. Защита данных	373
Угрозы целостности данных	374
Нестабильность платформы	374
Аппаратный сбой	375
Конкурентный доступ	375
Уменьшение уязвимости данных	378
Использование SQL-транзакций	379
Транзакция по умолчанию	381
Уровни изоляции	381
Неявная инструкция начала транзакции	384
Инструкция SET TRANSACTION	384
Инструкция COMMIT	385
Инструкция ROLLBACK	385
Блокировка объектов базы данных	386
Резервное копирование данных	386
Точки сохранения и субтранзакции	387
Ограничения в транзакциях	389
Предотвращение внедрения зловредного SQL-кода	393
Глава 16. Использование SQL в приложениях	395
SQL в приложении	396
Следите за звездочкой	396

Преимущества и недостатки SQL	397
Сильные и слабые стороны процедурных языков	397
Проблемы, возникающие при совместном использовании SQL с процедурными языками	398
Вставка инструкций SQL в процедурные языки	399
Внедрение SQL-кода	399
Модульный язык	402
Объектно-ориентированные инструменты быстрой разработки	405
Использование SQL в Microsoft Access	406
Часть 5. Практическое использование SQL	409
Глава 17. Доступ к данным с помощью ODBC и JDBC	411
ODBC	412
Интерфейс ODBC	412
Компоненты ODBC	413
ODBC в среде “клиент/сервер”	414
ODBC в Интернете	415
Серверные расширения	415
Клиентские расширения	415
ODBC в локальной сети	417
JDBC	418
Глава 18. Работа с XML-данными	421
Связь между XML и SQL	421
Тип данных XML	422
Когда использовать тип данных XML	423
Когда не стоит использовать тип данных XML	424
Преобразование данных из формата SQL в формат XML и обратно	424
Преобразование наборов символов	424
Преобразование идентификаторов	425
Преобразование типов данных	426
Преобразование таблиц	426
Обработка пустых значений	427
Создание схемы XML	428
Функции SQL для работы с XML-данными	429
XMLDOCUMENT	429
XMLELEMENT	429
XMLFOREST	430
XMLCONCAT	430
XMLAGG	431
XMLCOMMENT	431
XMLPARSE	432

XMLPI	432
XMLQUERY	432
XMLCAST	433
Предикаты	433
DOCUMENT	433
CONTENT	434
XMLEXISTS	434
VALID	434
Преобразование данных XML в таблицы SQL	435
Преобразование нестандартных типов данных в XML	436
Домены	437
Индивидуальные типы UDT	438
Записи	438
Массивы	439
Мультимножества	440
Содружество SQL и XML	441
Глава 19. SQL и JSON	443
Совместное использование JSON и SQL	444
Загрузка и хранение данных JSON в реляционной базе данных	444
Генерирование данных JSON на основе реляционных данных	444
Запрос данных JSON, хранящихся в реляционных таблицах	445
Модель данных SQL/JSON	445
Элементы SQL/JSON	445
Последовательности SQL/JSON	446
Синтаксический анализ JSON	446
Сериализация JSON	447
Функции SQL/JSON	447
Общий синтаксис JSON API	447
Функции запросов	449
Функции конструктора	452
Предикат IS JSON	455
Пустые значения в JSON и SQL	455
Язык XPath в SQL/JSON	455
Дополнительные сведения	456
Часть 6. Расширенные возможности SQL	457
Глава 20. Обработка наборов данных с помощью курсоров	459
Объявление курсора	460
Выражение запроса	461
Предложение ORDER BY	461
Разрешение обновления	463

Чувствительность	464
Перемещаемость	465
Открытие курсора	465
Извлечение данных из отдельных строк	467
Синтаксис	467
Ориентация перемещаемого курсора	468
Позиционные инструкции DELETE и UPDATE	469
Заккрытие курсора	469
Глава 21. Процедурное программирование и хранимые модули	471
Составные инструкции	472
Атомарность	473
Переменные	474
Курсоры	474
Состояния	474
Обработка состояний	475
Необрабатываемые состояния	478
Присваивание	478
Управляющие конструкции	479
Конструкция IF...THEN...ELSE...END IF	479
Конструкция CASE...END CASE	480
Цикл LOOP...END LOOP	481
Инструкция LEAVE	481
Цикл WHILE...DO...END WHILE	482
Цикл REPEAT...UNTIL...END REPEAT	482
Цикл FOR...DO...END FOR	483
Инструкция ITERATE	483
Хранимые процедуры	484
Хранимые функции	485
Полномочия	486
Хранимые модули	487
Глава 22. Обработка ошибок	489
Переменная SQLSTATE	489
Директива WHENEVER	491
Области диагностики	492
Заголовок области диагностики	493
Информационная часть области диагностики	494
Пример нарушения ограничения	497
Добавление новых ограничений в уже созданную таблицу	499
Интерпретация информации, возвращаемой в переменной SQLSTATE	499
Обработка исключений	500

Глава 23. Триггеры	503
Область применения триггеров	503
Создание триггера	504
Триггеры инструкций и строк	505
Когда срабатывает триггер	505
Запускаемая SQL-инструкция	505
Пример определения триггера	506
Срабатывание последовательности триггеров	506
Ссылки на старые и новые значения	507
Срабатывание нескольких триггеров в одной таблице	508
Часть 7. Великолепные десятки	509
Глава 24. Десять самых распространенных ошибок	511
Уверенность в том, что клиенты знают, чего хотят	512
Игнорирование масштаба проекта	512
Учет только технических факторов	512
Отсутствие обратной связи с пользователями	513
Использование только своей любимой среды разработки	513
Использование только своей любимой системной архитектуры	514
Проектирование таблиц базы данных отдельно друг от друга	514
Отказ от консультаций с другими специалистами	514
Игнорирование бета-тестирования	515
Отказ от создания документации	515
Глава 25. Десять советов по извлечению данных	517
Проверяйте структуру базы данных	518
Испытывайте запросы на тестовой базе данных	518
Дважды проверяйте запросы с объединениями	518
Трижды проверяйте запросы с подзапросами	519
Подводите итоги, используя предложение GROUP BY	519
Внимательно относитесь к ограничениям в предложении GROUP BY	519
Используйте круглые скобки с операторами AND, OR и NOT	520
Контролируйте полномочия на извлечение данных	520
Регулярно выполняйте резервное копирование своих баз данных	521
Тщательно обрабатывайте ошибочные состояния	521
Приложение. Ключевые слова ISO/IEC SQL:2016	523
Предметный указатель	527



Глава 17

Доступ к данным с помощью ODBC и JDBC

В ЭТОЙ ГЛАВЕ...

- » Что такое ODBC
- » Компоненты ODBC
- » Использование ODBC в среде “клиент/сервер”
- » Использование ODBC в Интернете
- » Использование ODBC в локальной сети
- » Использование JDBC

С каждым годом компьютеры все активнее и активнее подключаются к сетям, как локальным, так и глобальным. Как следствие, возникла необходимость в обеспечении совместного доступа к базам данных по сети, однако этому препятствует несовместимость системного программного обеспечения и приложений, работающих на разных компьютерах. Важными этапами на пути преодоления такой несовместимости стал стандарт SQL.

К сожалению, стандарт SQL не реализован на практике в чистом виде. Производители СУБД, утверждающие, что их продукты совместимы с международным стандартом SQL, зачастую включают в свои реализации расширения, несовместимые с продуктами *других* производителей. Производители не склонны отказываться от своих расширений, так как покупатели привыкли

к ним и зависят от них. Крупным организациям для совместного использования различных реализаций СУБД необходим другой подход, не требующий от производителей приводить их продукты к “наименьшему общему знаменателю”. Этот подход и реализует открытый интерфейс доступа к базам данных — ODBC (Open DataBase Connectivity).

ODBC

ODBC — это стандартный интерфейс между базой данных и приложением, взаимодействующим с ней. Наличие подобного стандарта позволяет любому приложению на клиентском компьютере получать доступ к любой базе данных на сервере с помощью SQL. Единственное требование заключается в том, чтобы клиентская и серверная части поддерживали стандарт ODBC. ODBC 4.0 — текущая версия стандарта.

Приложение получает доступ к конкретной базе данных, используя специально разработанный для нее *драйвер* (в данном случае драйвер ODBC). Клиентская часть драйвера, работающая напрямую с приложением, должна строго соответствовать стандарту ODBC. Приложению все равно, какая СУБД установлена на сервере. Серверная часть драйвера адаптирована к конкретной базе данных. Благодаря такой архитектуре не только не нужно настраивать приложения на определенную СУБД — им даже не обязательно знать, какая именно СУБД используется. Драйвер скрывает различия между серверами баз данных.

Интерфейс ODBC

Интерфейс ODBC — это унифицированный набор определений, необходимых для организации взаимодействия приложения и базы данных. Каждое такое определение принимается в качестве стандарта. Интерфейс ODBC определяет следующее:

- » библиотеку вызовов функций;
- » стандартный синтаксис SQL;
- » стандартные типы данных SQL;
- » стандартный протокол соединения с базой данных;
- » стандартные коды ошибок.

Вызовы функций ODBC обеспечивают подключение приложения к серверу базы данных, выполнение инструкций SQL и возврат результатов приложению.



СОВЕТ

Чтобы выполнить какое-либо действие с базой данных, в качестве аргумента функции ODBC необходимо указать соответствующую инструкцию SQL. При условии использования стандартного для ODBC синтаксиса SQL результат выполнения этой функции не будет зависеть от того, какая база данных установлена на сервере.

Компоненты ODBC

Интерфейс ODBC состоит из четырех функциональных компонентов, именуемых *уровнями ODBC*. Благодаря каждому из них достигается гибкость ODBC, позволяющая взаимодействовать любым ODBC-совместимым клиентам и серверам. Между пользователем и данными, которые он хочет получить, находятся четыре уровня интерфейса ODBC.

- » **Приложение.** Это та часть интерфейса ODBC, с которой непосредственно работает пользователь. Естественно, приложения могут быть не только в ODBC-совместимых системах. Однако включение приложения в интерфейс ODBC имеет определенный смысл. Приложение должно взаимодействовать с источником данных при посредничестве ODBC и подключаться с помощью диспетчера драйверов ODBC в строгом соответствии со стандартом ODBC.
- » **Диспетчер драйверов.** Это динамически подключаемая библиотека (DLL), обычно поставляемая компанией Microsoft. Она загружает необходимые драйверы для системных источников данных (возможно, нескольких) и направляет вызовы функций приложений с помощью драйверов к соответствующим источникам данных. Диспетчер драйверов непосредственно обрабатывает некоторые вызовы функций ODBC, а также перехватывает и обрабатывает определенные типы ошибок. Несмотря на то что стандарт ODBC внедрила компания Microsoft, он стал общепринятым (его приняли даже бескомпромиссные сторонники систем с открытым исходным кодом).
- » **Драйвер DLL.** В связи с тем что источники данных могут различаться, причем в некоторых случаях весьма существенно, необходим способ преобразования стандартных вызовов функций ODBC в языковой код конкретного источника данных. Этим и занимается драйвер DLL. Каждый драйвер получает вызовы функций через стандартный интерфейс ODBC и переводит их в код, понятный источнику данных. Как только источник данных готов вернуть результат, драйвер в обратном порядке преобразует его в стандартный для ODBC вид. Драйвер является основным элементом, который позволяет ODBC-совместимому приложению управлять структурой и содержимым ODBC-совместимого источника данных.

» **Источник данных.** Существует множество различных источников данных. Таким источником может быть база данных на основе реляционной СУБД, находящаяся на одном компьютере с приложением, или база данных на удаленном компьютере. В качестве источника данных может выступать файл, хранящийся вне СУБД, доступ к данным которого реализован *индексно-последовательным методом* (indexed sequential access method — ISAM). Такие файлы могут быть расположены как на локальном, так и на удаленном компьютере. Для каждого вида источников данных требуется свой драйвер.

ODBC в среде “клиент/сервер”

В среде “клиент/сервер” интерфейс между клиентской и серверной частями называется API (Application Programming Interface — интерфейс прикладного программирования). API может быть как специальным, так и стандартным. *Специальным* называется интерфейс, разработанный для взаимодействия с конкретной серверной частью. Программным кодом, который формирует этот интерфейс, является драйвер, и в специальной системе он называется собственным драйвером. *Собственный драйвер* оптимизирован для работы с определенной клиентской частью и связанной с ней серверной частью источника данных. В связи с тем что собственные драйверы настроены для работы как с приложением, так и с СУБД, они передают команды и информацию достаточно быстро и с минимальными задержками.



СОВЕТ

Если система “клиент/сервер” рассчитана на доступ к конкретному источнику данных и заведомо не будет использовать другой, лучше воспользоваться собственным драйвером из поставки СУБД. С другой стороны, если системе требуется обеспечивать доступ к данным, хранящимся в различных форматах, использование ODBC API избавит разработчика от выполнения огромного объема ненужной работы.

Драйверы ODBC оптимизированы для работы с определенными источниками данных серверной части, однако все они имеют одинаковый внешний интерфейс с диспетчером драйверов. Любой драйвер, не оптимизированный для работы с конкретным клиентом, скорее всего, проиграет в быстродействии собственному драйверу, который специально разработан для данного клиента. Основным недостатком первого поколения драйверов ODBC была их низкая производительность по сравнению с собственными драйверами. Однако последние измерения показали, что производительность драйверов ODBC 4.0 вполне сравнима с производительностью собственных драйверов.

На сегодняшний день технология достигла того уровня, когда уже не нужно жертвовать производительностью ради преимуществ стандартизации.

ODBC в Интернете

Операции с базами данных в Интернете кое в чем серьезно отличаются от операций с базами данных в среде “клиент/сервер”. Самое заметное отличие, с точки зрения пользователя, заключено в клиентской части системы, содержащей интерфейс пользователя. В среде “клиент/сервер” интерфейс пользователя — это часть приложения, которая связывается с источником данных на сервере посредством ODBC-совместимых инструкций SQL. В веб-среде клиентская часть системы по-прежнему находится на локальном компьютере, но взаимодействует с источником данных на сервере с помощью стандартного протокола HTTP.

Любой пользователь, располагающий соответствующим клиентским программным обеспечением (и соответствующими полномочиями), может получить доступ к данным, находящимся в Интернете. Это означает, что можно создать приложение на своем рабочем компьютере, а позже получить доступ к нему со своего мобильного устройства. На рис. 17.1 показаны основные различия между системой “клиент/сервер” и системой, созданной с помощью веб-технологий.

Серверные расширения

В системе, созданной на основе веб-технологий, клиентский компьютер и веб-сервер взаимодействуют с помощью протокола HTTP. *Серверное расширение* — это компонент системы, который преобразует команды, передаваемые по сети, в ODBC-совместимый SQL-код, после чего сервер базы данных связывается с источником данных и выполняет этот код. В обратном направлении источник данных пересылает результат запроса серверу базы данных и далее — серверному расширению, которое преобразует результат запроса к виду, понятному веб-серверу. Затем данные отсылаются клиентскому компьютеру по Интернету и отображаются у пользователя. На рис. 17.2 приведена схема такого взаимодействия.

Клиентские расширения

Такие приложения, как Microsoft Access 2016, предназначены для обработки данных, которые хранятся либо локально на компьютере пользователя, либо на сервере, расположенном в локальной или глобальной сети, либо в облачных хранилищах Интернета. Хранилище, предоставляемое компанией Microsoft,

называется OneDrive. Доступ к приложению в облаке можно также получить с помощью браузера. Браузеры создаются и оптимизируются для организации простого и интуитивно понятного интерфейса доступа к веб-серверам любых типов. Наиболее популярные браузеры, такие как Google Chrome, Mozilla Firefox, Microsoft Internet Explorer и Apple Safari, изначально не предназначались (и не оптимизировались) для использования в качестве клиентской части базы данных. Чтобы добиться требуемого уровня взаимодействия с базой данных в Интернете, необходимы дополнительные функциональные возможности. Для обеспечения этих возможностей были разработаны различные *клиентские расширения*, которые включают элементы управления ActiveX, апплеты Java и сценарии. Расширения взаимодействуют с сервером с помощью протокола HTTP, используя стандартный язык — HTML. Любой HTML-код, получающий доступ к базе данных, перед отправкой источнику данных преобразуется серверным расширением в ODBC-совместимый SQL-код.

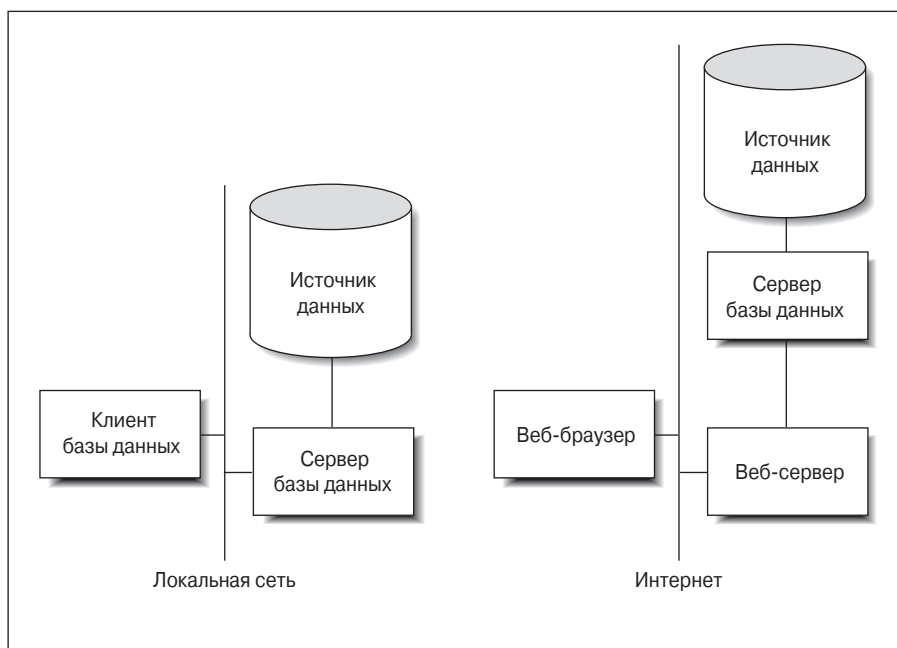


Рис. 17.1. Система "клиент/сервер" в сравнении с веб-ориентированной системой

Элементы управления ActiveX

Элементы управления Microsoft ActiveX работают с Microsoft Internet Explorer — одним из самых популярных браузеров в мире (хотя его позиции значительно пошатнулись с ростом популярности таких браузеров, как Google Chrome и Mozilla Firefox).

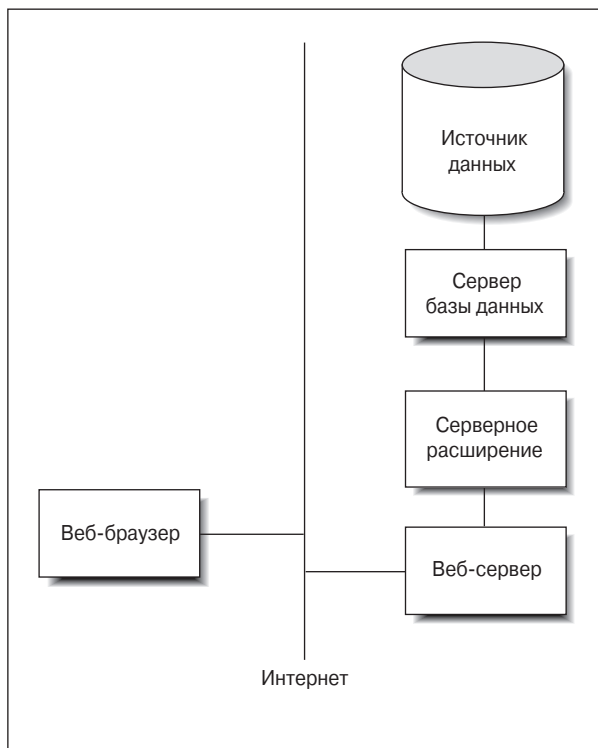


Рис. 17.2. Веб-ориентированная СУБД с серверным расширением

Сценарии

Сценарии — наиболее гибкий инструмент создания клиентских расширений. Использование языка сценариев, такого как JavaScript или Microsoft VBScript, позволяет максимально контролировать происходящее на клиентском компьютере. С помощью сценариев можно проверять достоверность данных, вводимых в поля формы, и отбраковывать неправильно заполненные формы еще на клиентском компьютере. Это экономит ваше время и нагрузку на интернет-канал. Безусловно, контроль данных можно также выполнять на сервере путем применения ограничений к значениям элементов данных. Как и апплеты Java, сценарии встраиваются в веб-страницу и выполняются при ее открытии пользователем.

ODBC в локальной сети

Интранет — это локальная или региональная сеть, работающая как упрощенная версия Интернета. Поскольку вся сеть принадлежит одной

организации, как правило, отсутствует необходимость в применении комплексных мер безопасности, таких как брандмауэры. Все инструменты, разработанные для создания веб-приложений, подходят также для создания приложений локальных сетей. ODBC работает в локальных сетях точно так же, как и в Интернете. При наличии нескольких различных источников данных клиенты, использующие браузеры и соответствующие клиентские и серверные расширения, могут взаимодействовать с этими источниками посредством SQL-кода, передаваемого с помощью HTTP и ODBC. Драйвер ODBC преобразует SQL-код в собственный язык команд базы данных и выполняет его.

JDBC

Интерфейс Java-приложений для взаимодействия с базами данных (Java DataBase Connectivity — JDBC) имеет много общего с ODBC, но в то же время содержит несколько существенных отличий. Одно из отличий явствует из названия. Как и ODBC, JDBC — универсальный интерфейс доступа к базам данных, не зависящий от источника данных на сервере. Различие состоит в том, что клиентское приложение для JDBC может быть написано только на Java, а не на каком-то другом языке программирования (например, C++ или Visual Basic). Еще одно отличие состоит в том, что Java и JDBC от начала и до конца разрабатывались для использования в Интернете.

Java — это язык программирования (похожий на C++), разработанный компанией Sun Microsystems специально для создания клиентских веб-ориентированных программ. После установления соединения между клиентской машиной и сервером соответствующий апплет Java загружается на компьютер клиента, где и выполняется. Апплет, внедренный на веб-страницу, предоставляет функции взаимодействия с базой данных, реализуя гибкий доступ клиента к данным на сервере. На рис. 17.3 показан механизм работы веб-приложения с базой данных с использованием апплета Java, запущенного на клиентской машине.

Апплет — это небольшое приложение, находящееся на сервере. Когда клиент подключается по Интернету к серверу, апплет загружается на клиентский компьютер и запускается. Апплеты Java разработаны таким образом, чтобы запускаться в своей *песочнице*. Песочница — это определенное (изолированное) место в памяти клиентского компьютера, где выполняются апплеты Java. Апплет не может взаимодействовать с чем-нибудь вне песочницы. Такая архитектура позволяет защитить клиентский компьютер от потенциально опасных апплетов, которые могут получить доступ к конфиденциальной информации или нанести серьезный вред данным.

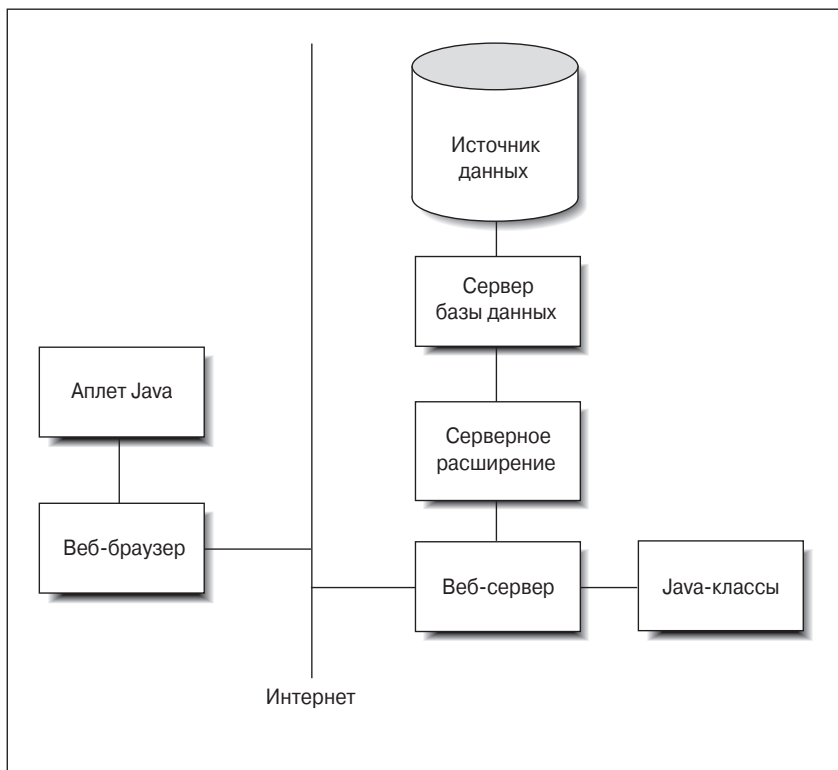


Рис. 17.3. Веб-приложение, предназначенное для работы с базой данных с использованием аплета Java

Главное достоинство апплетов Java заключается в том, что они постоянно обновляются. Поскольку апплеты загружаются с сервера при каждом использовании (а не остаются постоянно на машине клиента), клиент всегда снабжен последней версией аплета, доступной на момент его запуска.



СОВЕТ

Если вы отвечаете за поддержку сервера своей организации, то вам никогда не придется волноваться о совместимости с любым из клиентов при обновлении программного обеспечения сервера. Просто убедитесь в том, что загружаемый апплет Java совместим с новой конфигурацией сервера, и если все веб-браузеры поддерживают апплеты Java, то они автоматически также станут совместимыми с сервером. Java — это полнофункциональный язык программирования, позволяющий создавать надежные приложения доступа к базам данных в любых конфигурациях “клиент/сервер”. При таком использовании доступ Java-приложений к базам данных посредством JDBC подобен доступу к данным приложений C++ посредством ODBC. В то же

время при использовании в Интернете (или в локальной сети) работа приложений Java в корне отличается от работы приложений C++.

Когда система функционирует в Интернете, условия ее работы отличаются от условий в среде “клиент/сервер”. Клиентская часть приложения, которая работает с Интернетом, — это браузер с минимальными вычислительными возможностями. Эти возможности должны быть расширены, чтобы переложить на клиента часть работы с базой данных, и такое расширение функций обеспечивают апплеты Java.



ВНИМАНИЕ!

Загрузка данных с неизвестного сервера связана с рядом потенциальных опасностей. Если же вы загружаете Java-апплет, то уровень опасности резко снижается, хотя и не до нуля. Стоит с осторожностью относиться к сомнительному серверу и не позволять исполняемому коду беспрепятственно “заходить” на ваш компьютер.

Как и ODBC, JDBC передает SQL-инструкции от клиентской части приложения (апплета), запускаемого на компьютере клиента, к источнику данных на сервере. Также JDBC служит для передачи результатов выполнения запросов или сообщений об ошибках от источника данных назад приложению. Выгода от использования JDBC заключается в том, что разработчик апплетов может использовать стандартный интерфейс JDBC, не заботясь о том, какая база данных находится на сервере. JDBC выполняет все преобразования, необходимые для корректного двухстороннего взаимодействия. Несмотря на то что интерфейс JDBC предназначен для работы в веб-среде, он может также работать в среде “клиент/сервер”, обеспечивая посредничество для взаимодействия приложения, написанного на языке Java, с серверной частью базы данных.