ФГБОУ ВПО «Воронежский государственный
технический университет»

Кафедра систем информационной безопасности

337-2014

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 6–9 по дисциплинам
«Основы построения защищенных СУБД»,
«Безопасность систем баз данных»
для студентов специальностей 090301
«Компьютерная безопасность»,
090303 «Информационная безопасность
автоматизированных систем»
очной формы обучения

Воронеж 2015

Составитель канд. техн. наук Д. Г. Плотников

УДК 004.056.5

Методические указания к лабораторным работам № 6–9 по дисциплинам «Основы построения защищенных СУБД», «Безопасность систем баз данных» для студентов специальностей 090301 «Компьютерная безопасность», 090303 «Информационная безопасность автоматизированных систем» очной формы обучения / ФГБОУ ВПО «Воронежский государственный технический университет»; сост. Д. Г. Плотников. – Воронеж, 2015. 65 с.

Выполняя лабораторные работы, студенты получают знания и навыки по современным базам данных и системам управления базами данных, в частности для построения систем баз данных, управления данными с помощью языка SQL и других средств современных СУБД. В издании рассматриваются реляционные базы данных и системы управления базами данных.

Методические указания подготовлены в электронном виде в текстовом редакторе MS Word 2013 и содержатся в файле Плотников_ЛР_БД_6-9.pdf.

Табл. 7. Ил. 8. Библиогр.: 7 назв.

Рецензент д-р техн. наук, проф. А.Г. Остапенко

Ответственный за выпуск зав. кафедрой д-р техн. наук, проф. А.Г. Остапенко

Издается по решению редакционно-издательского совета Воронежского государственного технического университета

Лабораторная работа № 6
Управление транзакциями

Цель работы

Освоение способов управления транзакциями в среде MS SQL Server.

Темы для предварительной проработки

- Свойства транзакций и управление транзакциями.
- Уровни изоляции.

Подготовка к работе

Подготовить SQL-скрипты для выполнения проверок изолированности транзакций. Ваши скрипты должны работать с одной из таблиц, созданных в лабораторной работе №2.

Выполнение работы

- Запустить Microsoft Query Analyzer и соединиться с базой данных. Открыть второе окно для ввода текста запросов (Ctrl+N в первом окне).
- Установить в обоих сеансах уровень изоляции READ UNCOMMITTED . Выполнить сценарии проверки:
  - потерянных изменений,
  - грязного чтения.
- Записать протокол выполнения сценариев.
- Установить в обоих сеансах уровень изоляции READ COMMITTED. Выполнить сценарии проверки:
  - грязного чтения.
  - неповторяющегося чтения.
- Записать протокол выполнения сценариев.
- Установить в обоих сеансах уровень изоляции REPEATABLE READ. Выполнить сценарии проверки:

- неповторяющегося чтения,
- фантома.

• Записать протокол выполнения сценариев.

• Установить в обоих сеансах уровень изоляции SERIALIZABLE. Выполнить сценария проверки фантома. Записать протокол выполнения сценария.

• Закончить работу с Microsoft Query Analyzer.

Содержание отчета

• Сценарий и протокол его выполнения в среде Microsoft Query Analyzer.

• Краткие выводы о навыках, приобретенных в ходе выполнения работы.

Теоретические сведения

Уровни изоляции

Стандарт SQL/92 определяет уровни изоляции транзакций в многопользовательской системе через отсутствие таких аномалий доступа к базе данных, которые могут в конечном итоге угрожать целостности данных. В стандарте различаются следующие аномалии:

- **Потерянные изменения**. Транзакция T1 читает данные. Транзакция T2 читает те же данные. Транзакция T1 на основании прочитанного значения вычисляет новое значение данных, записывает его в базу данных и завершается. Транзакция T2 на основании прочитанного значения вычисляет новое значение данных, записывает его в базу данных и завершается. В результате значение, записанное транзакцией T2, "затрет" значение, записанное транзакцией T1.
- **Грязное чтение**. Транзакция T1 изменяет некоторые данные, но еще не завершается. Транзакция T2 читает эти же данные (с изменениями, внесенными транзакцией T1) и принимает на их основе какие-то

решения. Транзакция T1 выполняет откат. В результате решение, принятое транзакцией T2 основано на неверных данных.

- **Неповторяющееся чтение**. Транзакция T1 в ходе своего выполнения несколько раз читает одни и те же данные. Транзакция T2 в интервалах между чтениями транзакцией T1 изменяет эти данные и фиксируется. В результате оказывается, что чтения одних и тех же данных в транзакции T1 дает разные результаты.
- **Фантом**. Транзакция T1 в ходе своего выполнения несколько раз выбирает множество строк по одним и тем же критериям. Транзакция T2 в интервалах между выборками транзакции T1 добавляет или удаляет строки или изменяет столбцы некоторых строк, используемых в критерии выборки, и фиксируется. В результате оказывается, что одни и те же выборки в транзакции T1 выбирают разные множество строк.

Промышленные СУБД в том или ином объеме выполняют требования стандарта по дифференциации уровней изоляции, но при формально одном и том же уровне изоляции поведение транзакций может существенно различаться в разных СУБД.

Определение уровней изоляции в стандарте и в рассматриваемых нами СУБД сведено в табл. 1:

Таблица 1

Определение уровней изоляции

Уровни изоляции SQL/92	**АНОМАЛИИ**				**Microsoft SQL Server**	**DB2**	**Oracle**
	Потерянные изменения	Грязное чтение	Неповто-ряющееся чтение	Фантом			
READ UNCOMMITTED	нет	да	да	да	READ UNCOMMITTED	UNCOMMITTED READ	-
READ COMMITTED	нет	нет	да	да	READ COMMITTED	CURSOR STABILITY	READ COMMITTED
REPEATABLE READ	нет	нет	нет	да	REPEATABLE READ	READ STABILITY	-
SERIALIZABLE	нет	нет	нет	нет	SERIALIZABLE	REPEATABLE READ	SERIALIZABLE

Сценарии

Мы приводим сценарии проверки нежелательных ситуаций на примере таблицы EXAMPLE (табл. 2), структура и содержимое которой приведены ниже. Вам предстоит разработать подобные сценарии, работающие с одной из таблиц, созданных Вами в работе № 2.

Таблица 2

Таблица EXAMPLE

id INTEGER	**dat INTEGER**
1	100
2	110
3	120
4	130
5	140
6	150
7	160
8	170
9	180
10	190
11	200

Ниже приводятся сценарии проверок (табл. 3). Сценарии должны выполняться пошагово, что приводит к тому, что транзакции T1 и T2 выполняются параллельно в разных сеансах. Мы подразумеваем, что после выполнения каждого сценария мы восстанавливаем исходное содержимое таблицы EXAMPLE.

Таблица 3

Сценарии проверок

Шаг	Транзакция Т1	Транзакция Т2
1. Потерянные изменения		
1	BEGIN TRANSACTION	BEGIN TRANSACTION
2	UPDATE example SET dat=101 WHERE id=1	
3		UPDATE example SET dat=102 WHERE id=1
4		COMMIT
5	COMMIT	
Если потерянные изменения допускаются, то сценарий выполнится без ошибок и блокировок. В базе данных сохранится изменение, сделанное на шаге 2.		
2. Грязное чтение		
1	BEGIN TRANSACTION	BEGIN TRANSACTION
2	SELECT * FROM example WHERE id=1	
3		UPDATE example SET dat=101 WHERE id=1
4	SELECT * FROM example WHERE id=1	
5		ROLLBACK
6	SELECT * FROM example WHERE id=1	
Если допускается незавершенное чтение, то сценарий выполнится без ошибок и блокировок. На шаге 2 будут выбраны значения (1,100). На шаге 3 -(1,101). На шаге 4 - (1,100).		

Продолжение табл. 3

Шаг	Транзакция T1	Транзакция T2
3. Неповторяющееся чтение		
1	BEGIN TRANSACTION	BEGIN TRANSACTION
2	SELECT * FROM example WHERE id=1	
3	[COMMIT]	UPDATE example SET dat=101 WHERE id=1
4		COMMIT
5	SELECT * FROM example WHERE id=1	
6	COMMIT	
Если допускается неповторяющееся чтение, то сценарий выполнится без ошибок и блокировок. Операцию COMMIT на шаге 3 выполнять не придется. На шаге 2 будут выбраны значения (1,100). На шаге 3 - (1,101).		
4. Фантом (пример для READ UNCOMMITTED)		
1	BEGIN TRANSACTION	BEGIN TRANSACTION
2	SELECT * FROM example WHERE dat>180	
3	[COMMIT]	INSERT INTO example VALUES(12,210)
4		COMMIT
5	SELECT * FROM example WHERE dat>180	
6	COMMIT	
Если допускается фантом, то сценарий выполнится без ошибок и блокировок. Операцию COMMIT на шаге 3 выполнять не придется. На шаге 2 будут выбраны значения (10,190), (11,200). На шаге 3 - (10,190), (11,200), (12,210).		

Окончание табл. 3

Шаг	Транзакция T1	Транзакция T2
5. Тупик (пример для REPEATABLE READ)		
1	SELECT id from exampe WHERE dat=120	
2		SELECT id from exampe WHERE dat=130
3	UPDATE example SET id=3 WHERE dat=130	
4		UPDATE example SET id=4 WHERE dat=120
Если система не обнаруживает и не устраняет тупиков, то после выполнения шага 4 транзакции должны взаимно заблокироваться.		

Инструментальные средства Microsoft SQL Server

Для выполнения сценариев проверки изолированности следует открыть два окна внутри Microsoft Query Analyzer (либо два экземпляра Microsoft Query Analyzer)

Для того чтобы набор операторов выполнялся внутри транзакции, следует заключить их между строчками BEGIN TRANSACTION и COMMIT:

```
BEGIN TRANSACTION
...
...
COMMIT
```

Установка уровня изоляции в Microsoft Query Analyzer выполняется командой:

SET TRANSACTION ISOLATION LEVEL *уровень_изоляции*
уровень_изоляции может принимать значения: READ UNCOMMITTED / READ COMMITTED / REPEATABLE READ / SERIALIZABLE

Команду, выполняющую изменение уровня изоляции, следует разместить первой в скрипте.

В процессе выполнения задания в каждом из окон следует выделять только строчки, присутствующие на данном шаге, и нажимать на кнопку «выполнить». В процессе работы после нажатия на эту кнопку выполнение дальнейших команд может быть заблокировано (становится активна только кнопка «остановить текущую операцию»). В этом случае результат выполнения зависит от параллельной транзакции, и может быть вычислен только при её завершении (COMMIT/ROLLBACK).

Инструментальные средства DB2

Для выполнения сценариев проверки изолированности следует запустить два сеанса интерактивного SQL. Можно запустить в два сеанса DB2 Command Center, однако, Command Center - приложение довольно ресурсоемкое, и его выполнение в двух экземплярах на компьютере недостаточной мощности может оказаться затруднительным, поэтому мы предлагаем во втором сеансе запустить DB2 Command Line Processor. Command Line Processor - приложение для интерактивного выполнения операторов SQL из командной строки.

По умолчанию в средах интерактивного SQL DB2 режим `AUTOCOMMIT` включен. Чтобы его отключить следует ввести команду:

UPDATE COMMAND OPTIONS USING c OFF

Установка уровня изоляции в DB2 выполняется командой:

CHANGE ISOLATION TO *уровень_изоляции*

уровень_изоляции может принимать значения: UR / CS / RS / RR

Изменение уровня изоляции может выполняться только при отсутствии соединения с базой данных. Соединение с базой данных выполняется командой:

CONNECT TO имя базы

разрыв соединения - оператором

CONNECT RESET

Инструментальные средства Oracle

По умолчанию в среде SQL*Plus режим AUTOCOMMIT выключен.

Установка уровня изоляции в Oracle выполняется командой:

ALTER SESSION SET ISOLATION_LEVEL = *уровень_изоляции*

уровень_изоляции может принимать значения READ COMMITTED / SERIALIZABLE

Контрольные вопросы

1. В чем различия имени входа (логина) и пользователя?
2. Рассказать о ролях уровня сервера.
3. Рассказать о ролях уровня базы данных.
4. Можно ли создать свою роль уровня сервера?
5. Для чего нужны роли?
6. Что такое схема?

7. Рассказать о роли уровня базы данных public.
8. Рассказать про директивы GRANT, DENY и REVOKE.
9. Как разрешить пользователю предоставлять разрешение другим пользователям?
10. Как добавить нового пользователя в текущую базу данных?
11. Как создать новый логин?

Дополнительные вопросы

1. Исправить ошибки в обязательной части.
2. Сменить владельца базы данных.
3. Сменить пароль для имени входа.
4. Сменить базу данных по умолчанию для имени входа.
5. Определить роль с заданными правами.

Лабораторная работа № 7
Создание и использование триггеров

Цель работы

Освоение способов создания триггеров в среде Microsoft SQL Server.

Темы для предварительной проработки

- Функции триггеров.
- Синтаксис создания триггеров
- Использование триггеров в целях аудита

Подготовка к работе

- Подготовить план создания триггера для контроля ограничений целостности, заданных в Вашем варианте индивидуального задания. Использовать диалекты как Oracle, так и DB2.
- Сформулировать 2-4 запроса на добавление / изменение / удаление данных в таблицу, для которой будет назначен триггер. Составить SQL-скрипты для выполнения этих запросов.

Выполнение работы

- Запустить Microsoft Enterprise Manager и Microsoft Query Analyzer, соединиться с локальной базой данных.
- В среде Microsoft Enterprise Manager создать триггер контроля ограничений целостности. В среде Microsoft Query Analyzer выполнить скрипты, подготовленные для проверки триггера и сохранить протокол их выполнения.
- Вывести содержимое таблиц, показывающее действие триггеров.
- Закончить работу с Microsoft SQL Server.

Содержание отчета

- Протокол работы в среде Microsoft Query Analyzer.
- Краткие выводы о навыках, приобретенных в ходе выполнения работы.

Теоретические сведения

Триггеры - средство, обеспечивающее автоматическое выполнение некоторых действий при каждой модификации таблицы.

Триггер характеризуется следующими классификационными признаками, которые должны быть заданы при его создании:

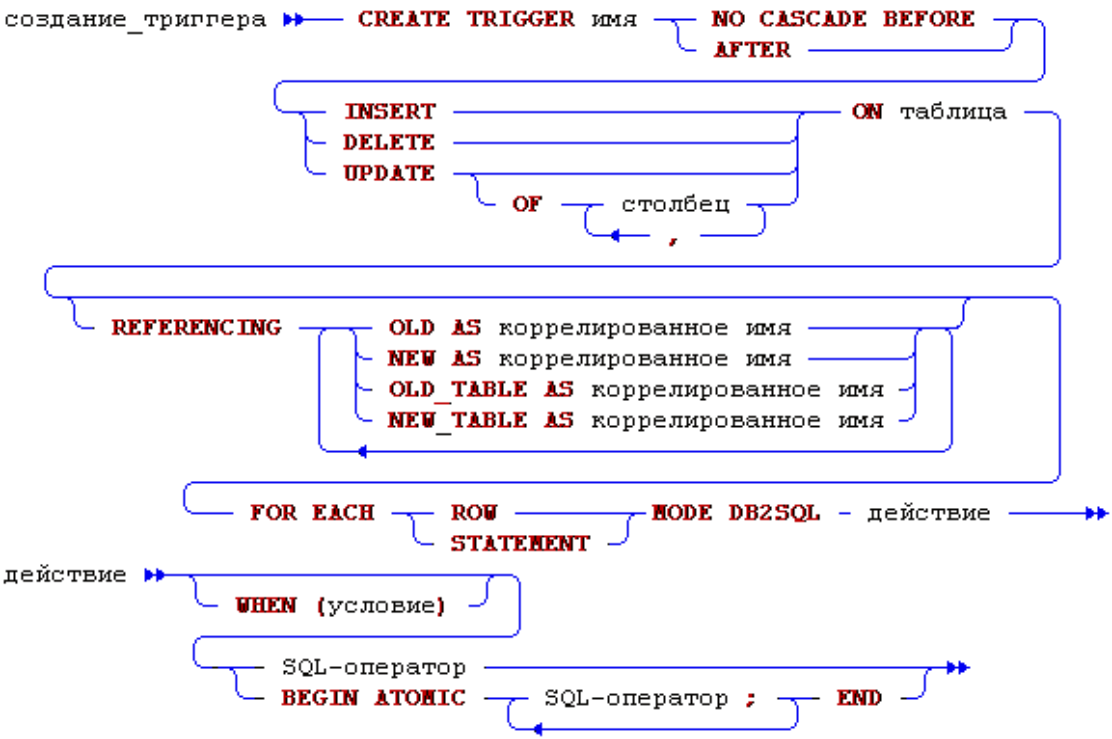
- условие активизации - действие над таблицей, которое вызывает запуск триггера - такими действиями являются операции INSERT, DELETE, UPDATE;
- время активизации - выполнение триггера до или после выполнения операции над таблицей;
- область действия - выполнение триггера либо один раз для каждого оператора модификации таблицы, либо для каждой строки, изменяемой / удаляемой / вставляемой в таблицу.

Таким образом, для каждой таблицы может быть создано до 12 типов триггеров. На самом деле, триггеров может быть и больше, так как может быть создано и несколько триггеров одного типа. В таком случае однотипные триггеры выполняются в порядке их создания.

При обеспечении в общем-то одинаковой функциональности, синтаксис и некоторые детали применения триггеров различаются в DB2, Oracle и MSSQL, поэтому мы рассматриваем их раздельно.

Триггеры: DB2

Триггер создается оператором языка SQL CREATE TRIGGER:



NO CASCADE BEFORE / AFTER - определяет время активизации триггера. NO CASCADE указывает, что если действие триггера включает в себя модификацию других таблиц, для которых в свою очередь определены триггеры, то эти вторичные триггеры не выполняются. (В текущей версии эта опция избыточна, так как в BEFORE-триггере не допускаются операторы модификации).

INSERT/ DELETE / UPDATE - условие активизации, для UPDATE триггер может запускаться при обновлении только заданных столбцов.

ON определяет имя базовой таблицы (не представления!), для которой создается триггер.

REFERENCING - определяет коррелированные имена, по которым действие триггера может обращаться к столбцам

изменяемых строк (промежуточным переменным) и к изменяемой таблице:

- OLD - имя, представляющее изменяемую строку до изменения (по умолчанию - old);
- NEW - имя, представляющее изменяемую строку до изменения (по умолчанию - new);
- OLD_TABLE - имя, представляющее изменяемую таблицу до изменения;
- NEW_TABLE - имя, представляющее изменяемую таблицу до изменения.

Существует довольно большое число правил, регламентирующих применение коррелированных имен (см. DB2 SQL Reference)

FOR EACH ROW / STATEMENT - область действия триггера.

MODE DB2SQL - обязательный параметр, не имеющий вариаций в текущих версиях.

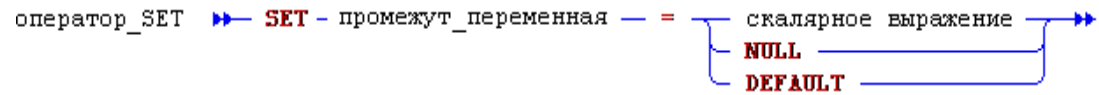
WHEN - определяет условие, при котором выполняются последующие операторы. Условие здесь - такое же условие, какое употребляется во фразе WHERE запроса.

Действия триггера могут включать в себя следующие SQL-операторы:

- для BEFORE-триггера: SELECT, SET, SIGNAL;
- для AFTER-
  триггера: SELECT, SET, SIGNAL, INSERT, DELETE, UP DATE.

Если в действиях триггера выполняются несколько операторов, то они берутся в операторные скобки BEGIN ATOMIC ... END и разделяются символом "точка с запятой".

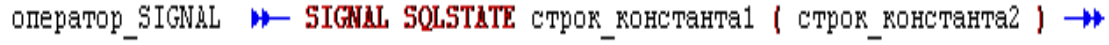
Оператор SET применяется только в триггерах и устанавливает значения промежуточных переменных:



Выражение в операторе SET подчиняется правилам, описанным для скалярного выражения в синтаксисе запроса. Выражение не может содержать агрегатных функций, кроме тех случаев, когда они употребляются в скалярном подзапросе. В нем могут употребляться промежуточные переменные NEW и OLD.

Обратите внимание на то, что в составе выражения присваивания можно применять CASE-выражения - это создает возможность выполнять присваивание по некоторой логике.

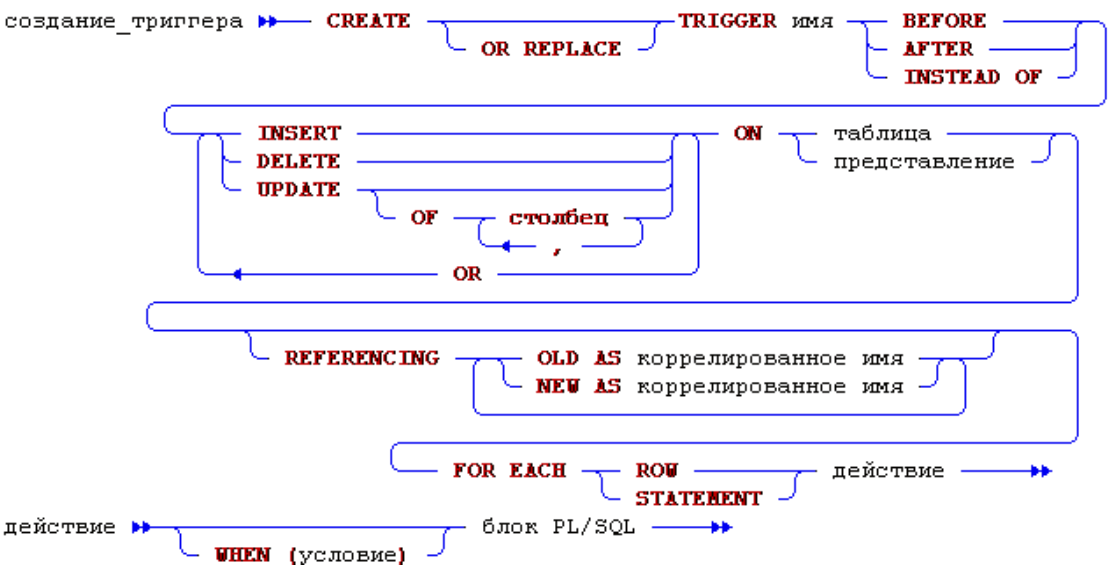
Оператор SIGNAL применяется только в триггерах и вызывает аварийное завершение триггера и оператора, его вызвавшего:



Строковая константа 1 содержит значение, устанавливаемое в SQLSTATE. Ее размер должен быть 5 символов, в ней допускаются только цифры и большие латинские буквы. Обычно значения, формируемые программистом, начинаются с символа '7'.

Триггеры: Oracle

Триггер создается оператором языка SQL CREATE TRIGGER:



Отличия триггера Oracle от триггера DB2 (многие из них видны из самого синтаксиса оператора) состоят в следующем:

- Оператор CREATE позволяет заменить триггер (в DB2 для этого нужно уничтожить триггер и создать его заново).
- Триггер можно создавать и для представления (это связано прежде всего с более широкими возможностями изменения представлений в Oracle).
- Введено новое условие активизации INSTEAD OF (триггер выполняется вместо оператора); оно применяется только для представлений и позволяет изменять базовые таблицы вместо неизменяемого представления.
- Для триггера могут быть назначены несколько условий активизации (через операцию OR). В этом случае действие триггера может "узнать" по какому условию запущен триггер, проверяя значения предикатов INSERTING, DELETEING, UPDATING.
- В условии фразы WHEN не допускаются запросы.

- Собственно выполняемые действия триггера задаются в виде блока на языке PL/SQL.

Триггер, выполняемый после (AFTER) выполнения оператора, модифицирующего некоторую таблицу, не может содержать операторы выборки (SELECT) из этой таблицы.

Триггеры: MSSQL

Триггер создается оператором языка SQL CREATE TRIGGER (приведен его минимальный синтаксис):

```
CREATE TRIGGER имя_триггера
ON { таблица | вид }
{
   { { FOR | AFTER | INSTEAD OF } { [DELETE] [,] [INSERT]
[,] [UPDATE] }
      AS
      [ { IF UPDATE ( столбец )
         [ { AND | OR } UPDATE ( столбец ) ]
            [ ...n ]
      ]
      оператор_SQL [ ...n ]
   }
}
```

К особенностям триггера MSSQL относится:

- Триггер можно создавать и для представления.
- Условие активизации BEFORE представлено условием активизации FOR.
- Введено новое условие активизации INSTEAD OF (триггер выполняется вместо оператора).
- Для триггера могут быть назначены несколько условий активизации через запятую (INSERT, DELETE, UPDAT).
- Проверить, подвергалось ли изменению значение конкретного столбца, можно с помощью предиката UPDATE (столбец).

- Собственно выполняемые действия триггера задаются в виде блока на языке Transact-SQL.
- В теле триггера можно обращаться к данным из двух псевдотаблиц inserted и deleted, которые имеют такую же структуру, как и родительская таблица триггера, и содержат новые и старые значения модифицированных строк соответственно.
- В MSSQL триггеры поддерживают только одну область видимости - выполнение триггера происходит только один раз для каждого оператора модификации таблицы.

Контрольные вопросы

1. В чем разница между sys и INFORMATION_SCHEMA?
2. Как обеспечить максимальную переносимость скриптов, использующих метаданные, на другие SQL-ориентированные СУБД?

Дополнительные вопросы

1. Исправить ошибки в подготовленных выборках.
2. Выбрать имена баз данных и файлы, соответствующие им на диске.
3. Выбрать названия ролей и имена пользователей, входящих в эти роли.
4. Вывести текст запроса, создающего какое-либо системное представление.
5. Составить различные другие выборки по заданию преподавателей.

Лабораторная работа № 8
Выборка метаданных

Цель работы

Ознакомление со способами хранения служебной информации в СУБД Microsoft SQL Server; освоение способов выборки метаданных.

Темы для предварительной проработки

- Состав и структура представлений словаря / системного каталога.

Подготовка к работе

- По приведенной справочной информации изучить Состав и структура представлений словаря Microsoft SQL Server.
- Выполнить следующие запросы на выборку сведений о таблицах, представлениях, триггерах, созданных в процессе выполнения лабораторного практикума:

1. выбрать имена всех таблиц, созданных назначенным пользователем базы данных (например, пользователем s33 для базы d33)
2. выбрать имя таблицы, имя столбца таблицы, признак того, допускает ли данный столбец null-значения, название типа данных столбца таблицы, размер этого типа данных - для всех таблиц, созданных назначенным пользователем базы данных и всех их столбцов
3. выбрать название ограничения целостности (первичные и внешние ключи), имя таблицы, в которой оно находится, признак того, что это за ограничение ('PK' для первичного ключа и 'F' для

внешнего) - для всех ограничений целостности, созданных назначенным пользователем базы данных

4. выбрать название внешнего ключа, имя таблицы, содержащей внешний ключ, имя таблицы, содержащей его родительский ключ - для всех внешних ключей, созданных назначенным пользователем базы данных
5. выбрать название представления, SQL-запрос, создающий это представление - для всех представлений, созданных назначенным пользователем базы данных
6. выбрать название триггера, имя таблицы, для которой определен триггер - для всех триггеров, созданных назначенным пользователем базы данных

- Составить SQL-скрипт для выполнения созданных запросов.

Выполнение работы

- Запустить Microsoft Query Analyzer, соединиться с локальной базой данных.
- В среде Microsoft Query Analyzer выполнить подготовленные скрипты.
- Сохранить результаты выполнения.
- Закончить работу с Microsoft Query Analyzer.

Содержание отчета

- Протокол работы в среде Microsoft SQL Server.
- Краткие выводы о навыках, приобретенных в ходе выполнения работы.

Теоретические сведения

Системный каталог и словарь данных

Каждая СУБД сохраняет метаданные (данные о данных) - детальную информацию обо всех объектах системы. Примерами таких объектов могут служить таблицы, представления, ограничения целостности, триггеры, правила безопасности и т.д. В разных СУБД применяются даже разные названия для метаданных - системный каталог в DB2 или словарь данных (Oracle). Однако общим свойством всех современных реляционных СУБД является то, что каталог/словарь сам состоит из таблиц. В результате пользователь может обращаться к метаданным так же, как и к своим данным - используя оператор SQL SELECT. Изменения же в каталоге/словаре производятся автоматически при выполнении пользователем операторов SQL, изменяющих состояние объектов базы данных.

Точнее, пользователь «видит» не сами таблицы каталога/словаря, а созданные на их базе представления, которые он, конечно же, не может изменять.

Состав и структура каталога/словаря очень различна для различных СУБД.

В Oracle представления словаря данных в большинстве своем доступны любому пользователю с привилегией `CREATE SESSION`. Существуют три группы представлений системного каталога:

- представления с префиксом DBA_ содержат информацию обо всех объектах базы данных (эти представления доступны только пользователям с привилегией SELECT_ANY_TABLE;
- представления с префиксом ALL_ содержат информацию обо всех объектах, доступных данному пользователю;
- представления с префиксом USER_ содержат информацию обо всех объектах, владельцем которых является данный пользователь.

В DB2 представления системного каталога определены в схеме SYSCAT и привилегия выборки из них предоставлена группе PUBLIC.

Несмотря на различия, мы попытались свести некоторые общие свойства представлений метаданных в табл. 4. В ней показаны далеко не все представления метаданных и далеко не все столбцы этих представлений. Для Oracle мы ограничились только представлениями группы ALL_. Для DB2 мы опускаем имя схемы - SYSCAT.

Таблица 4

Общие свойства представлений метаданных

Название представления		**Назначение представления**	**Название столбца**		**Назначение столбца**
Oracle	**DB2**		**Oracle**	**DB2**	
ALL_TABLES	TABLES	таблицы	TABLE_NAME	TABNAME	имя таблицы
			OWNER	DEFINER	владелец таблицы
			NUM_ROWS	CARD	количество строк в таблице
ALL_VIEWS	VIEWS	представления	VIEW_NAME	VIEWNAME	имя таблицы
			OWNER	DEFINER	владелец
			TEXT	TEXT	текст запроса из оператора CREATE VIEW
			-	VIEWCHECK	признак WITH CHECK OPTION
ALL_TAB_COLUMNS	COLUMNS	столбцы всех таблиц и представлений	TABLE_NAME	TABNAME	имя таблицы
			COLUMN_NAME	COLNAME	имя столбца
			DATA_TYPE	TYPENAME	тип данных
			DATA_LENGTH	LENGTH	длина данных
			DATA_PRECISION	используется LENGTH	точность

Продолжение табл. 4

			DATA_SCALE	SCALE	число знаков после точки
			DATA_DEFAULT	DEFAULT	значение по умолчанию
			NULLABLE	NULLS	признак допустимости пустого значения
ALL_CONSTRAINTS	TABCONST	табличные ограничения целостности	CONSTRAINT_NAME	CONSTNAME	имя ограничения
			TABLE_NAME	TABNAME	имя таблицы
			OWNER	DEFINER	владелец
			CONSTRAINT_TYPE	TYPE	тип ограничения
	CHECKS	ограничения целостности CHECK		TABNAME	имя таблицы
				DEFINER	владелец
			SEARCH_CONDITION	TEXT	текст условия
	REFERENCES	ссылочные ограничения целостности		TABNAME	имя таблицы
				DEFINER	владелец
				REFTABNAME	имя таблицы, на которую выполнена ссылка

Окончание табл. 4

			R_CONSTRAINT_NAME	REFKEYNAME	имя ключа таблицы, на который выполнена ссылка
			DELETE_RULE	DELETERULE	правила для удаления
			-	UPDATERULE	правила для изменения
ALL_TRIGGERS	TRIGGERS	триггеры	TRIGGER_NAME	TRIGNAME	имя триггера
			OWNER	DEFINER	владелец
			TABLE_NAME	TABNAME	имя таблицы
			TRIGGER_TYPE	TRIGTIME	время активизации
				GRANULARITY	область действия
			TRIGGERING_EVENT	TRIGEVENT	условие активизации
			DESCRIPTION	TEXT	текст оператора CREATE TRIGGER

Поскольку в двух наших концепциях управления доступом различаются принципиально, свести воедино информацию о доступе из системного каталога DB2 и словаря данных Oracle. Поэтому некоторые сведения о метаданных управления доступом даны в двух следующих таблицах (табл. 5 и 6). Обратите внимание на то, что в DB2 информация о пользователях и группах пользователей не отражается в метаданных.

Таблица 5

Метаданные о структуре базы данных - Microsoft SQL Server

Представление	**Назначение представления**	**Название столбца**	**Назначение столбца**
sysobjects	Таблицы, представления, ограничения целостности (первичные и внешние ключи), триггеры и т.д.	name	Название объекта
		id	Идентификатор объекта
		xtype	Тип: C = Ограничение целостности (CHECK) D = Значение по умолчанию (DEFAULT) F = Вторичный ключ (FOREIGN KEY) L = Журнал FN = Скалярная функция P = Хранимая процедура PK = Первичный ключ S = Системная таблица TR = Триггер U = Таблица пользователя UQ = Уникальное ограничение целостности V = Представление X = Внешняя хранимая процедура

Продолжение табл. 5

		uid	Идентификатор пользователя
		parent_obj	Идентификатор родительского объекта (например, идентификатор таблицы, которой принадлежит триггер)
		crdate	Дата создания объекта
syscolumns	Содержит информацию о столбцах таблицы и параметрах хранимых процедур	name	Название столбца или параметра
		id	идентификатор таблицы или хранимой процедуры, с которой связан столбец/параметр
		xtype	идентификатор типа данных (из таблицы systypes)
		colid	идентификатор столбца или параметра
		isnullable	признак того, допускаются ли null-значения (0 - не допускаются, 1 - допускаются)
systypes	Информация о типах данных	name	название типа данных
		xtype	идентификатор типа данных
		length	физическая длина
sysreferences	Информация о ссылочных ограничениях целостности данных (внешних ключах)	constid	идентификатор ссылочного ограничения целостности (ссылка на идентификатор внешнего ключа из таблицы sysobjects)
		fkeyid	идентификатор таблицы, содержащей внешний ключ
		rkeyid	идентификатор таблицы, содержащей его родительский ключ

Окончание табл. 5

syscomments	Содержит текстовые описания объектов (в частности, SQL-скрипты создания триггеров и представлений)	id	идентификатор объекта из таблицы sysobjects, к которому относится данное описание
		text	текст описания
sysusers	Содержит информацию о пользователях базы данных	uid	идентификатор пользователя
		name	имя пользователя

Таблица 6

Метаданные управления доступом - словарь данных Oracle

Название представления	**Назначение представления**	**Название столбца**	**Назначение столбца**
ALL_USERS	пользователи	USERNAME	имя пользователя
		USER_ID	внутренний код пользователя
		CREATED	дата создания
DBA_USERS	пользователи	USERNAME	имя пользователя
		USER_ID	внутренний код пользователя
		CREATED	дата создания
		PASSWORD	пароль
SYSTEM_PRIVILEGE_MAP	список системных привилегий	PRIVILEGE	код привилегии
		NAME	название привилегии
TABLE_PRIVILEGE_MAP	список табличных привилегий	PRIVILEGE	код привилегии
		NAME	код привилегии
DBA_ROLES	роли	ROLE	имя роли
		PASSWORD_REQUIRED	пароль

Продолжение табл. 6

Название представления	Назначение представления	Название столбца	Назначение столбца
ROLE_SYS_PRIVS	системные привилегии ролей	ROLE	имя роли
		PRIVILEGE	привилегия
		ADMIN_OPTION	возможность передачи
ROLE_SYS_PRIVS	табличные привилегии ролей	ROLE	имя роли
		OWNER	владелец таблицы
		TABLE_NAME	имя таблицы
		COLUMN_NAME	имя столбца
		PRIVILEGE	привилегия
		GRANTABLE	возможность передачи
ROLE_ROLE_PRIVS	роли, приданные ролям	ROLE	имя роли
		GRANTED_ROLE	имя приданной роли
		ADMIN_OPTION	возможность передачи

Продолжение табл. 6

Название представления	Назначение представления	Название столбца	Назначение столбца
DBA_ROLE_PRIVS	назначение ролей пользователям или другим ролям	GRANTEE	имя пользователь или роли
		GRANTED_ROLE	назначенная роль
		ADMIN_OPTION	возможность передачи
DBA_SYS_PRIVS	назначение системных привилегий пользователям или ролям	GRANTEE	имя пользователь или роли
		PRIVILEGE	привилегия
		ADMIN_OPTION	возможность передачи
DBA_TAB_PRIVS	назначение табличных привилегий пользователям или ролям	GRANTEE	имя пользователь или роли
		OWNER	владелец таблицы
		TABLE_NAME	имя таблицы
		GRANTOR	имя пользователя, который дал привилегию
		PRIVILEGE	привилегия
		GRANTABLE	возможность передачи

Окончание табл. 6

Название представления	Назначение представления	Название столбца	Назначение столбца
ALL_TAB_PRIVS	привилегии для таблиц, предоставляемые другим пользователям или PUBLIC	GRANTEE	имя пользователь или роли
		OWNER	владелец таблицы
		TABLE_NAME	имя таблицы
		GRANTOR	имя пользователя, который дал привилегию
		PRIVILEGE	привилегия
		GRANTABLE	возможность передачи

Таблица 7

Метаданные управления доступом - системный каталог DB2

Название представления	Назначение представления	Название столбца	Назначение столбца
DBAUTH	назначения привилегий базы данных	GRANTOR	кто дал привилегию
		GRANTEE	кому дана привилегия
		GRANTEETYPE	пользователю/группе

Продолжение табл.7

Название представления	**Назначение представления**	**Название столбца**	**Назначение столбца**
		CREATETABAUTH	привилегия CREATETAB
		BINDADDAUTH	привилегия BINDADD
		CONNECTAUTH	привилегия CONNECT
SCHEMAAUTH	назначения привилегий схемы	GRANTOR	кто дал привилегию
		GRANTEE	кому дана привилегия
		GRANTEETYPE	пользователю/группе
		SCHEMANAME	имя схемы
		ALTERINAUTH	привилегия ALTERIN
		CREATINAUTH	привилегия CREATIN
		DROPINAUTH	привилегия DROPIN
TABAUTH	назначения привилегий таблиц и представлений	GRANTOR	кто дал привилегию
		GRANTEE	кому дана привилегия
		GRANTEETYPE	пользователю/группе
		TABSCHEMA	имя схемы
		TABNAME	имя таблицы

Окончание табл. 7

Название представления	Назначение представления	Название столбца	Назначение столбца
		CONTROLAUTH	привилегия CONTROL
		ALTERAUTH	привилегия ALTER
		DELETEAUTH	привилегия DELETE
		INSERTAUTH	привилегия INSERT
		UPDATEAUTH	привилегия UPDATE
		SELECTAUTH	привилегия SELECT
		REFAUTH	привилегия REFERENCE
		INDEXAUTH	привилегия INDEX
COLAUTH	назначения привилегий для столбцов таблиц и представлений	GRANTOR	кто дал привилегию
		GRANTEE	кому дана привилегия
		GRANTEETYPE	пользователю/группе
		TABSCHEMA	имя схемы
		TABNAME	имя таблицы
		COLNAME	имя столбца
		PRIVTYPE	привилегия: UPDATE/REFERENCE
		GRANTABLE	возможность передачи

Контрольные вопросы

1. Объяснить принцип работы написанного триггера.
2. Какие бывают типы триггеров?
3. Когда может срабатывать триггер?
4. В каком порядке срабатывают триггеры?
5. Можно ли менять порядок срабатывания триггеров?
6. Сработает ли триггер, если оператор, выполненный пользователем, не затрагивает ни одну строку таблицы?

Дополнительные вопросы

1. Исправить ошибки в работе триггера.
2. Модифицировать триггер каким-либо образом.

Лабораторная работа № 9
Использование индексов и средств оптимизации запросов

Цель работы

Освоение основных методов и средств оптимизации запросов в средах СУБД Microsoft SQL Server.

Темы для предварительной проработки

- Использование индексов для ускорения выполнения запросов.
- Визуальные интерфейсы управления индексами Microsoft Enterprise Manager.
- Средства оптимизации запросов, предоставляемые средой Microsoft SQL Server.

Выполнение работы

1. Подготовить рабочие наборы данных:
   - Запустить Microsoft Query Analyzer, подключиться к пользовательской базе данных.
   - Для того, чтобы использование индексов было целесообразным, необходим проводить индексацию существенных объемов данных в таблицах. Для получения этих наборов данных нужно скопировать таблицы Customers, Orders, Order Details и Products из демонстрационной базы данных Northwind в пользовательскую базу данных с помощью следующих SQL-операторов (выполняемых в пользовательской базе данных):

```
SELECT * INTO Customers FROM Northwind..Customers
DECLARE @count INT
SELECT @count = 10
WHILE @count >= 0
```

```
  BEGIN
    INSERT INTO Customers SELECT * FROM
Northwind..Customers
    SELECT @count = @count - 1
  END

SELECT * INTO Orders FROM Northwind..Orders
SELECT * INTO OrderDetails FROM
Northwind.."Order Details"

SELECT * INTO Products FROM Northwind..Products
SELECT @count = 10
WHILE @count >= 0
  BEGIN
    INSERT INTO Products
(ProductName,SupplierID,CategoryID,QuantityPerUnit,

UnitPrice,UnitsInStock,UnitsOnOrder,

ReorderLevel,Discontinued)
    SELECT ProductName,SupplierID,CategoryID,
          QuantityPerUnit,UnitPrice,UnitsInStock,
            UnitsOnOrder,ReorderLevel,Discontinued
    FROM Northwind..Products
    SELECT @count = @count - 1
  END
```

Обратите внимание, что здесь используется цикл для увеличения количества записей в таблицах `Customers` и `Products`. Хотя это и некорректная операция в плане того, что мы дублируем первичные ключи и другую уникальную информацию, но в данном случае нас интересуют в первую очередь большие объемы информации в таблице, а не строгое следование ограничениям целостности.

Во всех заданиях лабораторной работы будут использоваться скопированные в предыдущем пункте таблицы

Customers, Orders, Order Details и Products в пользовательской базе данных. На рисунке ниже приведена диаграмма структуры и связей этих таблиц:

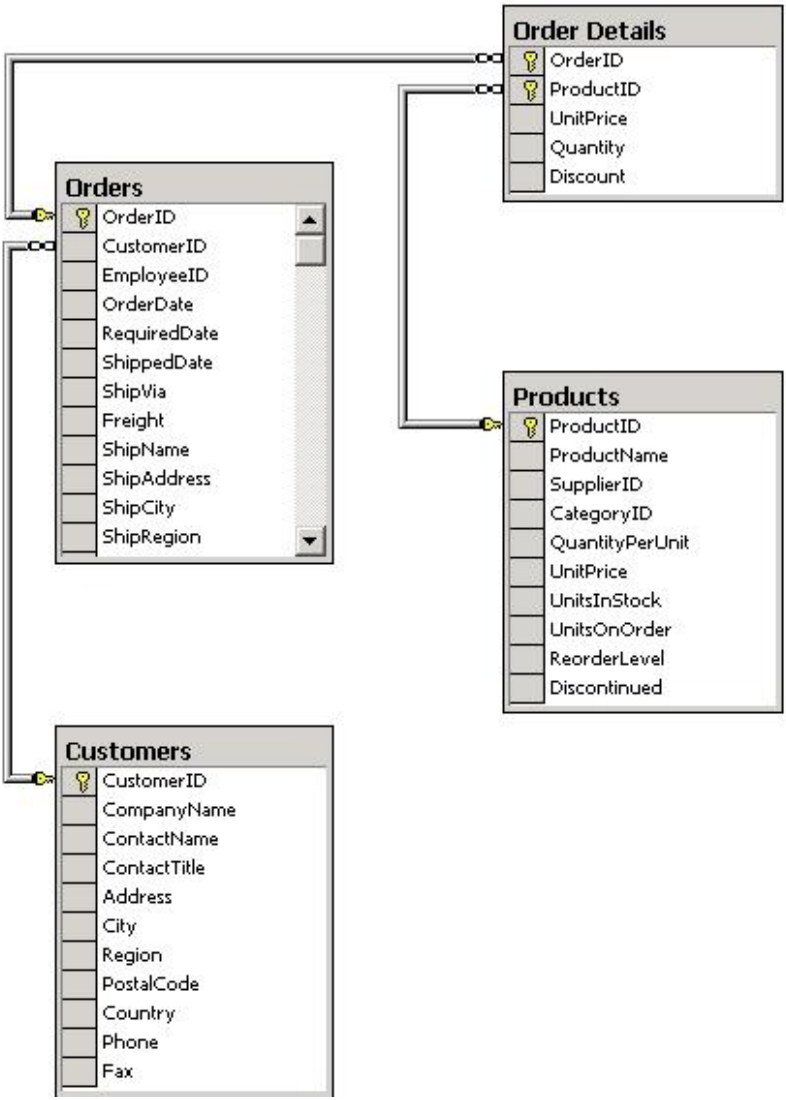


Рис. 1. Структура таблиц Customers, Orders, Order Details и Products

2. Выполнить оптимизацию запроса путем ручного создания индексов:
   - Построить SQL-запрос к таблице Customers с фильтрациями по идентификатору заказчика и нескольким другим полям.
   - Получить план выполнения запроса без использования индексов.
   - Оценить эффективность выполнения запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса в Microsoft Query Analyzer и сохранить эти отчеты в файле (например, report1_before.rpt) для последующего сравнения, поставив курсор в окно с результатами выполнения запроса и воспользовавшись пунктом меню ***File -> Save As...***.
   - Создать несколько индексов по используемым в запросе полям с помощью мастера или с помощью средств управления индексами Microsoft Enterprise Manager
   - Получить план выполнения запроса с использованием индексов и сравнить его с первоначальным планом.
   - Оценить эффективность выполнения оптимизированного запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса, сравнить их с характеристиками до индексирования и сохранить эти отчеты в файле (например, report1_after.rpt).
3. Выполнить оптимизацию запроса с помощью мастера ***Index Tuning Wizard***:
   - Построить SQL-запрос к связанным таблицам Customers и Orders с фильтрациями по нескольким полям этих таблиц.

- Получить план выполнения запроса без использования индексов.
- Оценить эффективность выполнения запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса и сохранить эти отчеты в файле.
- Получить план выполнения запроса с использованием индексов и сравнить его с первоначальным планом.
- Оценить эффективность выполнения оптимизированного запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса, сравнить их с характеристиками до индексирования и сохранить эти отчеты в файле.

4. Выполнить оптимизацию запроса с помощью мастера ***Index Tuning Wizard*** по результатам трассировки запросов с помощью ***Microsoft SQL Profiler***:
   - Построить SQL-запрос ко всем 4-м связанным таблицам Customers, Orders, OrderDetails и Produc ts с фильтрациями по нескольким полям этих таблиц.
   - Получить план выполнения запроса без использования индексов.
   - Оценить эффективность выполнения запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса и сохранить эти отчеты в файле.
   - Получить план выполнения запроса с использованием индексов и сравнить его с первоначальным планом.

- Оценить эффективность выполнения оптимизированного запроса с помощью статистики выполнений операций ввода-вывода и временных характеристик исполнения запроса, сравнить их с характеристиками до индексирования и сохранить эти отчеты в файле.

5. Закончить работу с Microsoft SQL Server.

Содержание отчета

- Протокол работы в среде Microsoft Query Analyzer.
- Краткие выводы о навыках, приобретенных в ходе выполнения работы.

Теоретические сведения

1. Основные принципы построения индексов

Рассмотрим пример того, как индексы могут использоваться для ускорения выборки данных. На рис. 2 показана таблица «Состав школы», содержащая имена и занимаемые должности всех работников школы:

Номер	Имя	Должность
1	Джон	Техник
2	Дэвид	Директор
3	Адам	Водитель автобуса
4	Гэри	Учитель
5	Лиза	Техник
6	Крис	Учитель
7	Дебби	Председатель совета
8	Денис	Заместитель директора
9	Брайан	Техник

Рис. 2. Таблица «Состав школы»

Как поступить, если из этой таблицы необходимо выбрать имена всех работников, занимающих должность техника? Можно прочитать все строки данных таблицы и отобразить имена только тех работников, которые занимают

должность техника. Процедура последовательного считывания всех строк таблицы в целях выполнения запроса называется ***сканированием таблицы***. А теперь создадим индекс для столбца «Должность» таблицы «Состав щколы», или, другими словами, проиндексируем эту таблицу по столбцу «Должность». Результаты этой работы представлены на рис. 3:

Должность	Указатель на строку
Водитель автобуса	3
Директор	2
Заместитель директора	8
Председатель совета	7
Техник	1
Техник	5
Техник	9
Учитель	4
Учитель	6

Рис. 3. Индекс по столбцу «Должность» таблицы «Состав школы»

Показанный на рис. 3 индекс содержит указатель на данные. Воспользуемся этим индексом для выполнения того же запроса. Вместо полного сканирования таблицы «Состав щколы» считывается только первая строка индекса и проверяется должность. Если это не интересующее нас значение «Техник», считывается следующая строка, и так, пока не будет найдена первая строка с требуемым значением. Из найденной строки выбирается указатель на запись, представляющий собой точный последовательный номер соответствующей строки таблицы «Состав щколы». Чтение строк индекса продолжается до тех пор, пока в его строках будет содержаться требуемое значение «Техник», после чего обработка индекса прекращается.

Для понимания того, как подобный алгоритм поиска нужной строки помогает ускорить выполнение запроса, кратко рассмотрим физическую структуру хранимых данных MSSQL.

В MSSQL данные и индексы таблиц хранятся в виде страниц, формат которых показан на рис. 4:

Рис. 4. Формат страницы данных SQL Server

Каждая страница размером 8192 байт включает заголовок, имеющий длину 96 байт. Еще один фрагмент страницы используется для размещения других структур данных, например, информации о переполнении строк. Вся оставшаяся часть страницы (8060 байт) предназначена для размещения данных (т.е. информации таблицы или индекса).

Предположим, что таблица "Состав щколы" содержит и другую информацию, например домашний адрес работника, номер телефона и т.д. Размер одной строки данных таблицы "Состав щколы" в этом случае может составлять приблизительно 2000 байт, тогда как ширина столбца "Должность" составляет 25 байт. Учитывая приведенные значения, разместим показанную на рис. 2 таблицу и показанный на рис. 3 индекс на страницах SQL Server, формат которых представлен на рис. 4. Данное размещение показано на рис. 5:

Физическое расположение страниц с данными таблицы «Состав школы»

Страница 5

	Джон	Техник	.....
	Дэвид	Директор	.....
	Адам	Водитель автобуса	.....
	Гэри	Учитель	.....

Страница 10

	Лиза	Техник	.....
	Крис	Учитель	.....
	Дебби	Председатель совета	.....
	Денис	Заместитель директора	.....

Страница 12

	Брайан	Техник	.....

Физическая страница индекса столбца «Должность» таблицы «Состав школы»

Страница 36

	Водитель автобуса	стр. 5, строка 3
	Директор	стр. 5, строка 2
	Заместитель директора	стр. 10, строка 4
	Председатель совета	стр. 10, строка 3
	Техник	стр. 5, строка 1
	Техник	стр. 10, строка 1
	Техник	стр. 12, строка 1
	Учитель	стр. 5, строка 4
	Учитель	стр. 10, строка 2

Рис. 5. Страницы данных таблицы «Состав школы» и ее индекса в базе данных SQL Server

Исходя из представленной на рис. 5 схемы рассмотрим, как в SQL Server будет выполняться поиск (без использования индекса) всех работников школы, занимающих должность «Заместитель директора». Прежде всего, будет считана страница с номером пять и ее данные будут просмотрены в поисках записей о нужных работниках. На первой странице (номер пять) такие записи найдены не будут. SQL Server считает следующую страницу (номер десять), просмотрит ее содержимое и выведет информацию из четвертой записи. Поскольку системе неизвестно, в скольких записях столбца «Должность» таблицы содержится значение «Заместитель директора», будет считана и просмотрена еще одна, последняя, страница таблицы с номером двенадцать. Итак, SQL Server выполнил полное сканирование таблицы, прочитав все записи с ее данными - в этом примере считывается всего три страницы данных. Ну, а если бы таблица «Состав школы» содержала бы в тысячу раз больше записей - около 9000 (что совсем не много для SQL Server)? Тогда для выборки необходимых данных пришлось бы считать 3000 страниц, причем даже в том случае, если бы в ней существовала только одна удовлетворяющая условию запись.

Рассмотрим, как будет выполняться тот же запрос с использованием индекса. Прежде всего, SQL Server считает страницу с данными индекса и промотрит ее содержимое в поисках значения «Заместитель директора». Искомое значение содержится в третьей строке. Из этой строки выбирается значение указателя, показывающее, что соответствующая запись является четвертой на странице номер десять. SQL Server считывает десятую страницу, выбирает четвертую строку и отображает найденное значение имени. Номер строки на странице сокращенно обозначается RID (Row Identifier). Затем проверяется значение в следующей строке индекса. Поскольку значение в ней отличается от искомого, SQL Server заканчивает обработку запроса. Таким образом, в данном случае для выполнения запроса потребовалось считать только две страницы, а не три, как в случае сканирования таблицы.

А что можно сказать о выполнении обсуждавшегося выше запроса об именах техников? Сканирование таблицы предусматривает чтение всех трех страниц данных. Использование же индекса потребует от SQL Server считать все три страницы данных, плюс еще одну страницу индекса. В итоге считанных страниц получится даже больше, чем при обычном сканировании! В некоторых случаях сканирование таблицы может оказаться эффективнее по сравнению с примененим для поиска индекса. Принятие решения о выборе используемого при поиске индекса или применении метода сканирования в SQL Server возлагается на службу оптимимзации запросов.

Структура индексов в Microsoft SQL Server

Для представления индексов в SQL Server используется схема двоичного дерева, показанная на рис. 5. Двоичные деревья представляют собой многоуровневые динамические поддерживаемые структуры.

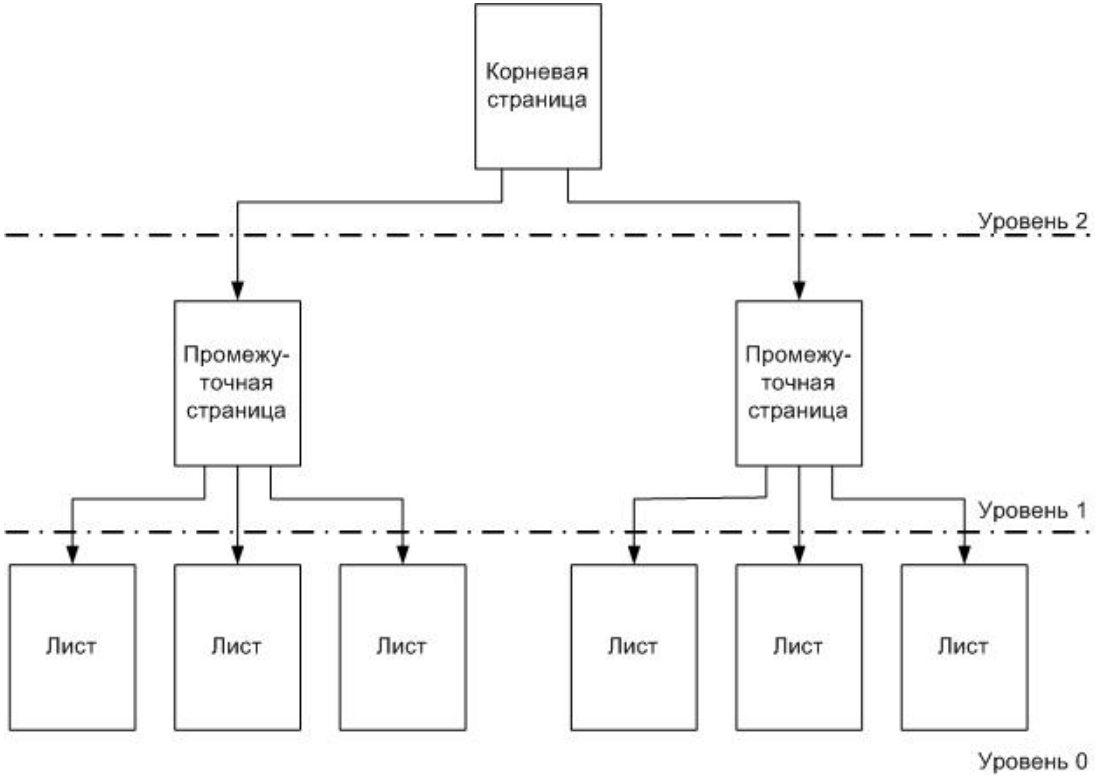


Рис. 6. Схема построения двоичного дерева

Двоичное дерево состоит из верхнего уровня, называемого ***корнем***, нижнего уровня, называемого ***листами*** (это всегда уровень 0), и несколько (от 0 до N) ***промежуточных уровней***. Дерево на рис. 6 включает один промежуточный уровень. В терминах SQL Server каждый из прямоугольников на рис. 6 отображает страницу индекса (или страницу данных). Чем больше уровней используется для представления индекса, тем больше страниц индекса потребуется считать для получения доступа к искомым записям данных (т.е. с увеличением числа уровней производительность обработки индекса уменьшается). В SQL Server поддерживается два различных типа индексов – ***кластерные*** и ***некластерные***.

Кластерный индекс

Кластерный индекс представляет собой двоичное дерево, в котором на нулевом уровне (уровне листов) содержатся страницы актуальных данных таблицы, а физически информация хранится в логическом порядке данного индекса.

При создании кластерного индекса резко возрастает количество дисковых операций ввода-вывода, связанное с переупорядочиванием страниц данных, созданием страниц индекса и удалением освободившихся страниц данных таблицы.

На рис. 7 показан пример кластерного индекса, созданного для столбца «Имя» таблицы «Состав школы». Обратите внимание, что страницы данных являются листовыми страницами этого кластерного индекса, а информация на страницах данных логически упорядочена.

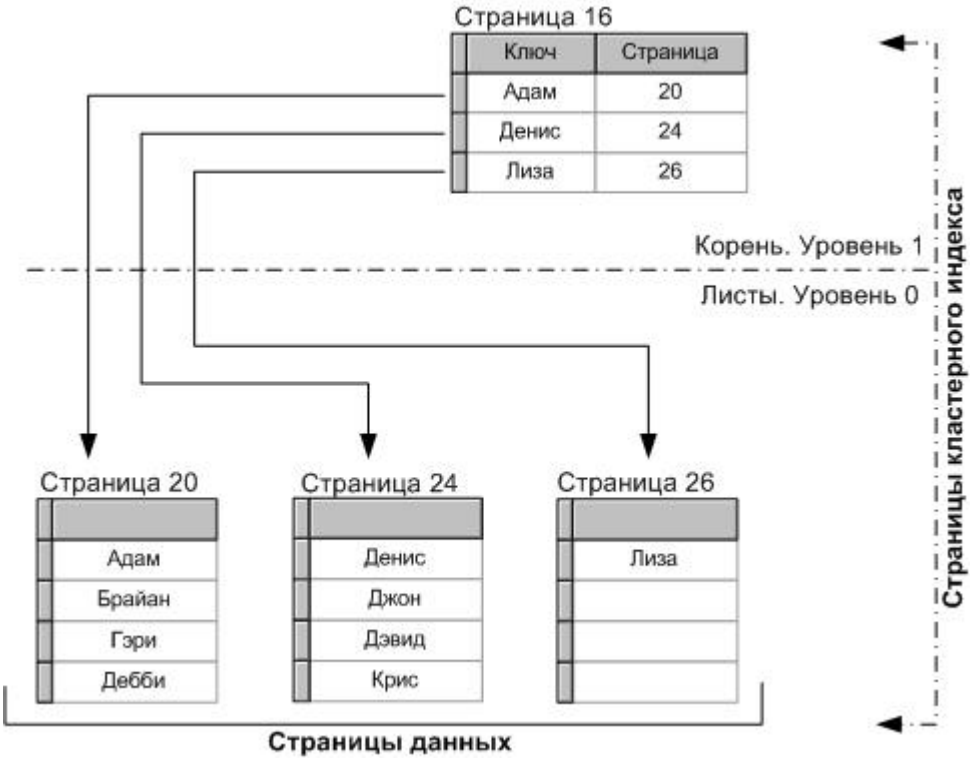


Рис. 7. Кластерный индекс для столбца «Имя» таблицы «Состав школы»

Поскольку записи на страницах данных физически располагаются в требуемом порядке, для каждой таблицы может существовать только один кластерный индекс.

В SQL Server страницы индексов состоят из заголовка страницы, после которого располагаются собственно строки индекса. Строки индекса состоят из ключевого значения и указателя на страницу индекса или строку данных таблицы (листовой уровень кластерного индекса). Страницы индекса последовательно связываются друг с другом с помощью двунаправленных ссылок.

Некластерные индексы

В случае ***некластерных индексов*** страницы листового уровня содержат не актуальные данные таблицы (как в случае кластерного индекса), а указатель на строку данных, включающий номер страницы данных и порядковый номер записи на странице. Некластерный индекс не требует физического переупорядочивания строк данных таблицы.

На рис. 8 показан пример некластерного индекса для столбца «Имя» таблицы «Состав школы».

Обратите внимание, что некластерные индексы всегда имеют на один уровень больше кластерных, поэтому после достижения уровня листов дополнительно потребуется выполнить чтение страницы данных. Если таблица имеет кластерный индекс, указатели строк некластерных индексов будут ссылаться на уровень листов кластерного индекса. Если таблица не имеет кластерного индекса, указатель строк представляет собой RID, создаваемый на основе идентификатора файла, номера страницы и номера записи на странице.

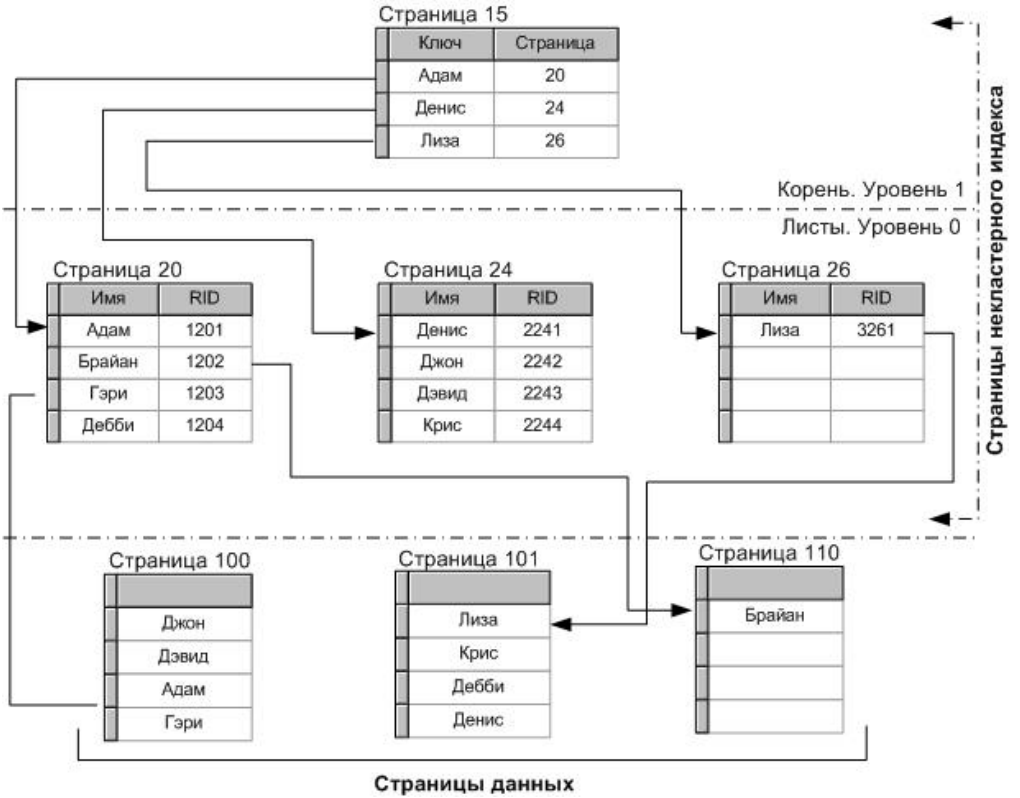


Рис. 8. Некластерный индекс для столбца «Имя» таблицы «Состав школы»

Рекомендуемая стратегия использования индексов

Выбор используемых индексов определяется структурой таблиц и типом запросов, с помощью которых будет выполняться поиск данных в этих таблицах . Прежде счем приступать к созданию индексов, надо убедиться, что индексируемые столбцы входят в критерии поиска запросов или помещаются в таблицы по каким-либо другим соображениям, например для предотвращения дублирования данных. В последующих разделах обсуждаются некоторые аспекты стратегии использования индексов.

Что следует индексировать

Ниже приведено несколько критериев, которыми можно руководствоваться при определении, какие именно столбцы таблицы следует индексировать:

- Столбцы, используемые для объединения таблиц
- Столбцы, используемые для ограничения диапазона данных, которые анализируются при выполнении запросов
- Столбцы, используемые в директивах ORDER BY и GROUP BY запросов
- Столбцы, используемые в функциях суммирования и подведения итогов

Что не следует индексировать

Ниже приведены случаи, когда индексацию не следует использовать вовсе или использовать в крайне ограниченных масштабах:

- Таблицы содержат незначительное количество строк
- Столбцы имеют слабо выраженную селективность (т.е. очень широкий диапазон значений)
- Значения в столбцах имеют очень большую длину (не рекомендуется индексировать столбцы с значениями длиннее 25 байт)
- Столбцы при построении запросов не используются

Кластерные и некластерные индексы

Как уже упоминалось, для каждой таблицы может существовать только один кластерный индекс. Ниже перечислены случаи, в которых применять кластерные индексы целесообразно:

- Столбцы используются в широком диапазоне запросов
- Столбцы используются в директивах ORDER BY и GROUP BY запросов
- Столбцы используются для объединения таблиц
- Используются запросы, возвращающие большой результирующий набор данных

Некластерные индексы целесообразно применять в ситуациях, перечисленных ниже:

- Столбцы используются в функциях суммирования и группирования
- Столбцы имеют внешние ключи
- Используются запросы, возвращающие небольшие результирующие наборы данных
- Доступ к информации часто осуществляется с помощью некоторого столбца, используемого в условиях объединения таблиц или в директивах ORDER BY и GROUP BY запросов

Контрольные вопросы

1. Что такое кластерный индекс? В чем его отличие от некластерного?
2. Можно ли создать неуникальный кластерный индекс?
3. В чем отличие первичного ключа и уникального индекса?
4. В каких случаях имеет смысл создавать индексы? Какие колонки следует включать в индекс и почему?
5. Какие существуют способы внутренней организации индексов?

6. Рассказать о проблеме фрагментации индексов. Как бороться с фрагментацией?

7. Имеет ли значение порядок указания колонок при создании индекса?

8. В чем разница между Index Scan и Index Seek?

Дополнительные вопросы

1. Исправить ошибки в подготовленных выборках.

2. Могут ли индексы ухудшить производительность? Если да, то продемонстрировать это.

3. На что влияет порядок сортировки (ASC\DESC) при создании индекса? Продемонстрировать это.

4. Объяснить и продемонстрировать разницу в скорости выполнения запросов в зависимости от того, входят ли все колонки из списка выборки в некластерный индекс.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Кузнецов, С. Д. Основы баз данных [Текст] / С. Д. Кузнецов. – М.: Бином. Лаборатория знаний, Интернет-университет информационных технологий, 2007.

2. Microsoft SQL Server 2005 Books Online. [Электронный ресурс] – Режим доступа: http://msdn.microsoft.com/en-us/library/ms130214%28v=sql.90%29.aspx

3. Гарсиа-Молина, Г. Системы баз данных. Полный курс [Текст] / Г. Гарсиа-Молина, Дж. Ульман, Д. Уидом. – М.: Вильямс, 2004.

4. Ульман, Дж. Основы реляционных баз данных [Текст] / Дж. Ульман, Д. Уидом. – М.: Лори, 2006.

5. Коннолли, Т. Базы данных: проектирование, реализация и сопровождение. Теория и практика [Текст] / Т. Коннолли, К. Бегг. – М.: Вильямс, 2003.

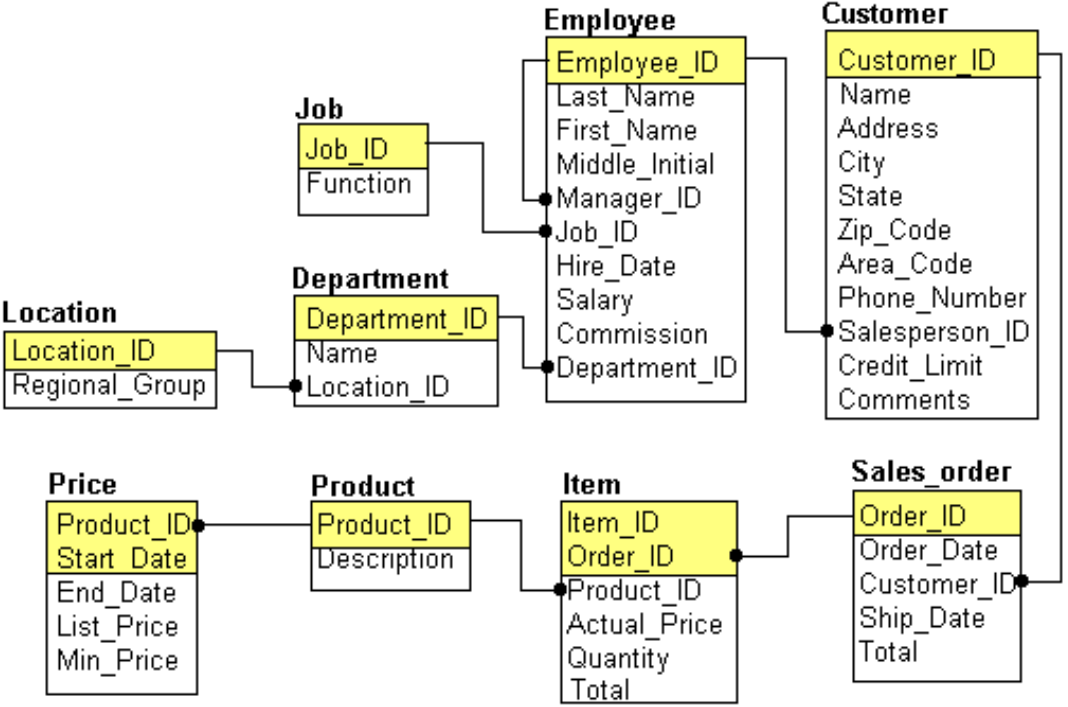
6. Станек У. Р. Microsoft SQL Server 2005. Справочник администратора [Текст] / У. Р. Станек. – М.: Русская редакция, 2008.

7. Нильсен, П. SQL Server 2005. Библия пользователя [Текст] /П. Нильсен. – М.: Вильямс, 2008.

.

ПРИЛОЖЕНИЕ 1

Концептуальная схема учебной базы данных



ПРИЛОЖЕНИЕ 2

Таблицы базы данных

N пп	Имя столбца	Тип данных Oracle	Тип данных DB2(MSSQL)	Комментарий
Таблица EMPLOYEE - сотрудники фирмы				
1	employee_id	NUMBER(4,0)	SMALLINT	Код сотрудника
2	last_name	VARCHAR2(15)	VARCHAR(15)	Фамилия
3	first_name	VARCHAR2(15)	VARCHAR(15)	Имя
4	middle_initial	VARCHAR2(1)	VARCHAR(1)	Средний инициал
5	manager_id	NUMBER(4,0)	SMALLINT	Код начальника
6	job_id	NUMBER(3,0)	SMALLINT	Код должности
7	hire_date	DATE	DATE(DATETIME)	Дата поступления в фирму
8	salary	NUMBER(7,2)	NUMERIC(7,2)	Зарплата
9	commission	NUMBER(7,2)	NUMERIC(7,2)	Комиссионные
10	department_id	NUMBER(2,0)	SMALLINT	Код отдела
CREATE TABLE EMPLOYEE (employee_id SMALLINT NOT NULL PRIMARY KEY,				

```
last_name VARCHAR(15),
first_name VARCHAR(15),
middle_initial VARCHAR(1),
manager_id SMALLINT,
job_id SMALLINT,
hire_date DATETIME,
salary NUMERIC(7,2),
commission NUMERIC(7,2),
department_id SMALLINT  );
```

Таблица DEPARTMENT - отделы фирмы

1	department_id	NUMBER(2,0)	SMALLINT	Код отдела
2	name	VARCHAR2(14)	VARCHAR(14)	Название отдела
3	location_id	NUMBER(3,0)	SMALLINT	Код места размещения

```
CREATE TABLE DEPARTMENT (
 department_id SMALLINT NOT NULL PRIMARY KEY,
 name VARCHAR(14),
 location_id SMALLINT );
```

Таблица LOCATION - места размещения отделов				
1	location_id	NUMBER(3,0)	SMALLINT	Код места размещения
2	regional_group	VARCHAR2(20)	VARCHAR(20)	Город

```
CREATE TABLE LOCATION (
  location_id SMALLINT NOT NULL PRIMARY KEY,
  regional_group VARCHAR(20) );
```

Таблица JOB - должности в фирме				
1	job_id	NUMBER(3,0)	SMALLINT	Код должности
2	function	VARCHAR2(30)	VARCHAR(30)	Название должности

```
CREATE TABLE JOB  (
  job_id SMALLINT NOT NULL PRIMARY KEY,
  [function] VARCHAR(30) );
```

Таблица CUSTOMER - фирмы-покупатели				
1	customer_id	NUMBER(6,0)	INTEGER	Код покупателя
2	name	VARCHAR2(45)	VARCHAR(45)	Название покупателя
3	address	VARCHAR2(40)	VARCHAR(40)	Адрес
4	city	VARCHAR2(30)	VARCHAR(30)	Город
5	state	VARCHAR2(2)	VARCHAR(2)	Штат
6	zip_code	VARCHAR2(9)	VARCHAR(9)	Почтовый код
7	area_code	NUMBER(3,0)	SMALLINT	Код региона
8	phone_number	NUMBER(7,0)	INTEGER	Телефон
9	salesperson_id	NUMBER(4,0)	SMALLINT	Код сотрудника-продавца, обслуживающего данного покупателя
10	credit_limit	NUMBER(9,2)	NUMERIC(9,2)	Кредит для покупателя
11	comments	LONG	VARCHAR(500)	Примечания

```
CREATE TABLE CUSTOMER (
  customer_id INTEGER NOT NULL PRIMARY KEY,
  name VARCHAR(45),
  address VARCHAR(40),
```

```
  city VARCHAR(30),
  state VARCHAR(2),
  zip_code VARCHAR(9),
  area_code SMALLINT,
  phone_number INTEGER,
  salesperson_id SMALLINT,
  credit_limit NUMERIC(9,2),
  comments VARCHAR(500) );
```

Таблица SALES_ORDER - договоры о продаже

1	order_id	NUMBER(4,0)	SMALLINT	Код договора
2	order_date	DATE	DATE(DATETIME)	Дата договора
3	customer_id	NUMBER(6,0)	INTEGER	Код покупателя
4	ship_date	DATE	DATE(DATETIME)	Дата поставки
5	total	NUMBER(8,2)	NUMERIC(8,2)	Общая сумма договора

```
CREATE TABLE SALES_ORDER (
  order_id SMALLINT NOT NULL PRIMARY KEY,
  order_date DATETIME,
  customer_id INTEGER,
  ship_date DATETIME,
  total NUMERIC(8,2) );
```

Таблица ITEM - акты продаж				
1	order_id	NUMBER(4,0)	SMALLINT	Код договора, в состав которого входит акт
2	item_id	NUMBER(4,0)	SMALLINT	Код акта
3	product_id	NUMBER(6,0)	INTEGER	Код продукта
4	actual_price	NUMBER(8,2)	NUMERIC(8,2)	Цена продажи
5	quantity	NUMBER(8,0)	INTEGER	Количество
6	total	NUMBER(8,2)	NUMERIC(8,2)	Общая сумма

```
CREATE TABLE ITEM (
  order_id SMALLINT NOT NULL,
  item_id SMALLINT NOT NULL,
  product_id INTEGER,
  actual_price NUMERIC(8,2),
  quantity INTEGER,
  total NUMERIC(8,2),
PRIMARY KEY (order_id,item_id)
```

Таблица PRODUCT - товары				
1	product_id	NUMBER(6,0)	INTEGER	Код продукта
2	description	VARCHAR(30)	VARCHAR(30)	Название продукта

```
CREATE TABLE PRODUCT (
  product_id INTEGER NOT NULL PRIMARY KEY,
  description VARCHAR(30) );
```

Таблица PRICE – цены				
1	product_id	NUMBER(6,0)	INTEGER	Код продукта
2	list_price	NUMBER(8,2)	NUMERIC(8,2)	Объявленная цена
3	min_price	NUMBER(8,2)	NUMERIC(8,2)	Минимально возможная цена
4	start_date	DATE	DATE(DATETIME)	Дата установления цены
5	end_date	DATE	DATE(DATETIME)	Дата отмены цены

```
CREATE TABLE PRICE (
  product_id INTEGER NOT NULL,
  list_price NUMERIC(8,2),
  min_price NUMERIC(8,2),
  start_date DATETIME NOT NULL,
  end_date DATETIME,
  PRIMARY KEY (product_id,start_date)
);
```

СОДЕРЖАНИЕ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторным работам № 6–9 по дисциплинам
«Основы построения защищенных СУБД»,
«Безопасность систем баз данных»
для студентов специальностей 090301
«Компьютерная безопасность»,
090303 «Информационная безопасность
автоматизированных систем»
очной формы обучения

Составитель
Плотников Денис Геннадьевич

В авторской редакции

Подписано к изданию 20.04.2015
Уч.- изд. л. 4,0

ФГБОУВПО «Воронежский государственный
технический университет»
394026 Воронеж, Московский просп., 14