МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное
образовательное учреждение высшего образования
«Омский государственный технический университет»

О. Б. Малков

ORACLE SQL

Базовая часть

Учебное текстовое электронное издание
локального распространения

Омск
Издательство ОмГТУ
2019

Сведения об издании: 1, 2

ISBN 978-5-8149-2849-8

УДК 004.65(075)
ББК 32.973.26-018.2я73
М19

Рецензенты:

В. А. Мещеряков, д-р техн. наук, проректор по информационным технологиям Сибирского государственного автомобильно-дорожного университета (СибАДИ);

О. Н. Лучко, профессор, зав. кафедрой информатики, математики и естественно-научных дисциплин Омской гуманитарной академии

Малков, О. Б.

М19 Oracle SQL. Базовая часть : учеб. пособие / О. Б. Малков ; Минобрнауки России, ОмГТУ. – Омск : Изд-во ОмГТУ, 2019.

ISBN 978-5-8149-2849-8

Описана практическая работа с СУБД Oracle Database XE 18c. Рассмотрен процесс инсталляции программного обеспечения и создания реляционной базы данных, включая реализацию базовой схемы, представлений и транзакций.

Издание предназначено для студентов, обучающихся по направлениям 09.03.01 «Информатика и вычислительная техника», 09.03.04 «Программная инженерия», 27.03.03 «Системный анализ и управление». Будет полезно студентам других направлений, изучающим дисциплины «Базы данных» и «Системы управления базами данных».

УДК 004.65(075)
ББК 32.973.26-018.2я73

Рекомендовано редакционно-издательским советом Омского государственного технического университета

ISBN 978-5-8149-2849-8

1 электронный оптический диск

Оригинал-макет издания выполнен в Microsoft Office Word 2007/2010
с использованием возможностей Adobe Acrobat Reader.

Минимальные системные требования:

- процессор Intel Pentium 1,3 ГГц и выше;
- оперативная память 256 Мб и более;
- свободное место на жестком диске 260 Мб и более;
- операционная система Microsoft Windows XP/Vista/7/10;
- разрешение экрана 1024×768 и выше;
- акустическая система не требуется;
- дополнительные программные средства Adobe Acrobat Reader 5.0 и выше.

Редактор *К. В. Муковоз*
Компьютерная верстка *Л. Ю. Бутаковой*

Сводный темплан 2019 г.
Подписано к использованию 28.06.19.
Объем 2,97 Мб.

Издательство ОмГТУ
644050, г. Омск, пр. Мира, 11; т. 23-02-12
Эл. почта: info@omgtu.ru

ВВЕДЕНИЕ

Oracle Database – объектно-реляционная система управления базами данных компании Oracle, стабильно занимающая первое место среди реляционных СУБД. Oracle Database обладает выдающимся функционалом, отличной масштабируемостью и беспрецедентной надежностью. С другой стороны, для нее характерны высокая стоимость флагманских версий, требовательность к системным ресурсам, сложность внедрения и обслуживания.

Сделать обучение более доступным позволяют бесплатные Express-версии Oracle Database, популярные в образовательных учреждениях и небольших компаниях, для которых ограничения этих версий не критичны. Тем более в большинстве случаев Oracle Database функционирует одним и тем же образом независимо от операционной системы и избранной версии.

В пособии рассмотрен процесс создания и использования реляционной базы данных средствами Oracle Database. Теоретическая часть пособия содержит общие сведения об используемом программном обеспечении. Практическая часть включает восемь практических занятий, в каждом из которых сжато дается теоретический материал по рассматриваемой теме и приведены конкретные примеры выполнения задания. В качестве сквозного примера используется база данных интернет-магазина, торгующего литературой компьютерной тематики.

Опыт показывает, что при изучении дисциплины «Системы управления базами данных» Oracle Database целесообразно изучать в качестве «второй» СУБД, уже получив навыки работы с такими СУБД, как MySQL или MS SQL Server.

Тема Oracle Database совершенно неподъемна для небольшого учебного пособия, поэтому сразу же изложим принятые в пособии ограничения:

- это не учебник по языку SQL. В пособии не рассматривается синтаксис базовых конструкций языка – эти сведения можно найти в литературе и в Интернете. Акцент сделан на детали, специфичные именно для Oracle Database, и на отличия Oracle SQL от стандарта SQL;
- это не учебник по администрированию Oracle, хотя студент, устанавливающий Express-версию Oracle Database на компьютере, вынужденно становится системным администратором Oracle и должен принимать решения;
- программирование доступа к базе данных на языке PL/SQL здесь не рассматривается. Это целесообразно сделать темой отдельного учебного пособия.

В качестве клиентского приложения используется Oracle SQL Developer. Эта графическая утилита автоматически формирует инструкции SQL, которые можно просмотреть и использовать независимо.

1. СУБД ORACLE DATABASE 18C EXPRESS EDITION

На практических занятиях задания выполняются в среде Oracle Database 18c Express Edition (XE), доступной для загрузки, разработки и развертывания по адресу *https://www.oracle.com*.

1.1. Общие сведения о СУБД Oracle Database 18c XE

Oracle Database XE (Express Edition) – это бесплатная редакция программного обеспечения. Обычно для бесплатных редакций характерны минимальная функциональность, существенные ограничения на используемые ресурсы и малый размер дистрибутива. Тем не менее с выходом Oracle Database 18c XE открылись возможности, ранее неиспользуемые в Express-версиях. Компания Oracle, скептически относившаяся к Express-версиям, изменила стратегию и вместо предоставления минимального функционала включила в редакцию практически все популярные опции полнофункциональной редакции Enterprise Edition, к которым относятся:

- поддержка подключаемых баз данных (Multitenant-архитектура);
- хранение в памяти копии данных в формате, ориентированном на столбцы, для быстрейшего доступа к ним;
- прозрачное сжатие данных в базе данных и в структурах памяти экземпляра;
- поддержка работы с геоданными;
- прозрачное шифрование данных;
- секционирование таблиц и индексов (Partitioning);
- продвинутые аналитические возможности и многое другое.

Oracle Database 18c XE – многомодельная база данных, поддерживающая самые различные данные (реляционные, JSON, XML, Graph, Spatial, Object, Key/Value). С ней могут взаимодействовать приложения, написанные на языках Java, JavaScript, Python, .NET, Go, PHP, C/ C++. Поддержка REST позволяет взаимодействовать с данными через стандартные веб-службы RESTful. Поверх Oracle Database 18c XE может быть развернута популярная платформа разработки приложений Oracle Application Express (APEX) [11].

Ограничения наложены на используемые данной версией СУБД ресурсы (хотя и есть прирост по сравнению с версией Oracle Database 11g XE) [9, 12]:

- максимальный размер базы данных – 12 Гб (против 11 Гб в 11g XE);
- оперативная память – до 2 Гб (против 1 Гб в 11g XE);
- потоки CPU – до 2 (против 1 в 11g XE);
- до трех подключаемых баз данных (Pluggable Databases) (в 11g XE эта функциональность отсутствовала).

Эти возможности делают Oracle Database 18c XE прекрасной средой разработки и создания приложений, управляемых данными. Она может служить платформой для администраторов баз данных, желающих узнать о последних достижениях Oracle. СУБД полностью совместима с другими выпусками Oracle Database и при необходимости легко масштабируется до других выпусков и служб облачных баз данных Oracle.

1.2. Особенности архитектуры Oracle Database 18c XE

Oracle Database 18c, как и ее предшественница Oracle Database 12c, относится к облачным СУБД (буква «c» от слова «cloud» – «облако»). Эти СУБД поддерживают *архитектуру Multitenant*, существенно отличающуюся от архитектуры предшественниц до Oracle Database 11g включительно (буква «g» от слова «grid» – «сетка»). Рассмотрим особенности этой архитектуры [1, 13].

В терминологии Oracle *база данных* (*Database*) – это данные, которые обрабатываются как единое целое. База данных состоит из файлов операционной системы (файлов данных, журнальных, управляющих и пр.). Файлы данных объединены в логические структуры – табличные пространства. В каждой базе данных можно создать множество логических структур – схем, включающих таблицы, представления, процедуры и пр.

Экземпляр Oracle (*Instance*) обеспечивает программные механизмы доступа и управления базой данных. Для доступа к базе данных Oracle пользователи должны обращаться к экземпляру Oracle. Экземпляр может быть запущен независимо от любой базы данных (без ее монтирования или открытия). Экземпляр состоит из процессов, работающих на сервере, и структур в оперативной памяти, обеспечивающих коммуникацию между процессами (*System Global Area, SGA*). Каждый экземпляр имеет уникальное имя – *системный идентификатор* (*SID*).

В дооблачных СУБД Oracle Database один экземпляр мог открыть только одну базу данных (при этом одна база данных могла быть открыта несколькими экземплярами). Новая архитектура Multitenant (мультиарендная) позволяет использовать принцип множества баз данных в составе единой супербазы данных. Согласно официальной терминологии Oracle такая супербаза данных называется мультиарендной контейнерной базой данных (*Container Database, CDB*), а база данных – подключаемой базой данных (*Plugin Database, PDB*). Каждая независимая база данных PDB имеет свой набор схем и табличных пространств, но при этом у них общая память SGA и один набор серверных процессов, что существенно экономит ресурсы сервера (один экземпляр). Старые базы данных (до версии Oracle Database 11g включительно) получили название *не-CDB баз данных*, а старая архитектура стала называться *не-CDB архитектурой*.

Облачные СУБД поддерживают как новую мультиарендную архитектуру, так и прежнюю не-CDB архитектуру. Это позволяет обновить не-CDB базу данных до последних версий и продолжать эксплуатировать ее в этом виде. Однако при желании можно включить не-CDB базу данных в состав CDB как PDB.

1.3. Oracle Database: объекты схемы базы данных

Oracle группирует связанную информацию в логические структуры, называемые *схемами*. Схемы содержат логические структуры, называемые *объектами*. Каждый объект принадлежит одной схеме и его имя включает имя этой схемы. Рассмотрим объекты, которые может содержать схема.

- *Таблицы*. Основные единицы хранения данных в Oracle. Содержат все доступные пользователю данные. Таблица содержит строки, представляющие отдельные записи. Строки состоят из полей, образующих столбцы таблицы.
- *Индексы*. Дополнительные объекты, которые могут улучшить производительность при извлечении данных из таблиц. Индексы создаются на одном или нескольких столбцах таблицы и автоматически сохраняются в базе данных.
- *Представления*. Виртуальные таблицы, объединяющие данные нескольких таблиц. Используют данные таблиц или других представлений.
- *Последовательности*. Объекты, используемые для создания последовательного списка уникальных целых чисел для идентификаторов записей.
- *Синонимы*. Псевдонимы для объектов схемы. Обеспечивают безопасность и удобство работы (скрыть владельца объекта или упростить операторы SQL).
- *Хранимые подпрограммы*. Процедуры и функции, хранимые в базе данных. Могут быть вызваны из клиентских приложений для доступа к базе данных.
- *Триггеры*. Хранимые подпрограммы, которые автоматически запускаются в базе данных при наступлении заданных событий в конкретной таблице или представлении. Могут ограничивать доступ к данным, а также вести журнал.
- *Пакеты*. Хранящиеся в базе данных как единое целое группы связанных подпрограмм и используемых ими явных курсоров и переменных. Могут быть вызваны из клиентских приложений, имеющих доступ к базе данных.

Как правило, объекты базы данных, которые использует приложение, принадлежат одной и той же схеме.

1.4. Oracle Database: администратор базы данных

Пользователь Windows, установивший Oracle, автоматический получает права администратора и может выполнять следующие действия [10]:

- запуск и останов базы данных;
- подключение к базе данных;

- управление сетевыми подключениями;
- управление памятью базы данных;
- управление хранилищами базы данных;
- управление пользователями и безопасностью;
- мониторинг базы данных;
- экспорт и импорт данных и метаданных.

Автоматически созданные учетные записи называются *внутренними учетными записями пользователей*, а соответствующие им схемы – *внутренними схемами*. Одни из них создаются для администрирования базы данных, другие – для работы с некоторыми опциями или модулями базы данных, имеющими свои собственные схемы (например, учетная запись *CTXSYS*, используемая с продуктом Oracle Text, обеспечивающим индексацию интерактивной справки Oracle).

Из внутренних учетных записей для подключения к базе данных следует использовать только учетные записи *SYSTEM* и *SYS*. Они создаются при инсталляции Oracle с паролем, указанным при инсталляции:

- *SYSTEM* – пользовательская учетная запись, обеспечивающая вход в систему для выполнения всех административных функций, кроме запуска и завершения работы базы данных;
- *SYS* – пользовательская учетная запись, дающая неограниченные полномочия на таблицы словаря данных. Базовые таблицы и представления словаря данных хранятся в схеме *SYS*. Они критичны для функционирования Oracle, обрабатываются только СУБД и не должны изменяться пользователем или администратором. Нельзя создавать какие-либо таблицы в схеме *SYS*.

Внимание! Следует избегать подключения к базе данных в качестве пользователя *SYS*. Предпочтительнее для выполнения административных задач использовать учетную запись *SYSTEM*.

Системная привилегия *SYSDBA* назначается только пользователю *SYS* для выполнения высокоуровневых административных задач, таких как запуск и останов базы данных. Если вы хотите войти как пользователь *SYS* через командную строку SQL*Plus, необходимо использовать опцию *AS SYSDBA*.

Существует три способа подключения к базе данных Oracle для выполнения административных задач (административный вход в систему):

- подключение в качестве пользователя *SYSTEM*;
- подключение в качестве пользователя, которому дана роль *DBA*;
- подключение с опцией *SYSDBA*.

1.5. ORACLE DATABASE: ПОЛЬЗОВАТЕЛИ И СЕАНСЫ БАЗЫ ДАННЫХ

Oracle – многопользовательская среда. Чтобы пользователь получил доступ к базе данных, администратор должен создать для него *учетную запись*. Учетные записи с правами администрирования (стандартные) создаются автоматически при установке базы данных. Под ними входят администраторы, чтобы выполнять свои задачи. Каждая база данных Oracle имеет минимум две стандартные учетные записи *SYS* и *SYSTEM*. Пароли для стандартных учетных записей *SYS* и *SYSTEM* задаются при инсталляции СУБД. Oracle содержит также другие стандартные учетные записи и записи внутренних пользователей, с помощью которых поддерживаются различные функции Oracle.

Внимание! Работа с таблицами и другими объектами базы данных конкретной предметной области под учетной записью администратора *крайне нежелательна*, поэтому для доступа к объектам базы данных должны быть созданы конечные пользователи с ограниченным уровнем привилегий.

При создании учетной записи нового пользователя неявно создается схема этого пользователя. *Схема* – это логический контейнер для объектов базы данных, которые создает пользователь. Имя схемы совпадает с именем пользователя и может быть использовано для ссылки на объекты, которыми владеет пользователь (например, *OLEG.ORDERS*).

Создавая нового пользователя, администратор базы данных предоставляет ему определенные *привилегии*.

- *Системные привилегии* дают пользователю полномочия, позволяющие выполнять операции на уровне системы. Это может затрагивать безопасность всей системы, поэтому предоставлять системные привилегии нужно осторожно.
- *Объектные привилегии* позволяют пользователю выполнять операции с конкретным объектом базы данных (таблицей, представлением и др.).

Привилегии могут быть предоставлены пользователю или отозваны у него. Предоставление и отзыв привилегий может выполнять лицо, имеющее соответствующие полномочия.

Именованные группы привилегий называются *ролями*. В отличие от других объектов роли не содержатся в схемах. Администратор создает роли, назначает им системные и объектные привилегии, а потом присваивает роли пользователям. Oracle XE поставляется с несколькими предопределенными ролями:

- *DBA* – позволяет выполнять большинство административных функций; дает все системные привилегии, кроме запуска и останова экземпляра;

- *CONNECT* – позволяет подсоединяться к базе данных;
- *RESOURCE* – предназначена для разработчика приложений, дает право пользователю создавать объекты в своей схеме.

Внимание! Использовать эти роли специалисты Oracle не рекомендуют, поскольку их поддержка в будущих версиях не планируется. Следует предоставлять только те привилегии, которые понадобятся конкретному пользователю.

Для работы с Oracle пользователь должен запустить клиентское приложение (например, Oracle SQL Developer) и установить связь с Oracle, используя имя пользователя и его пароль, содержащиеся в учетной записи. *Сеанс базы данных* начинается после установки соединения с базой данных Oracle и заканчивается при отсоединении от нее.

В облачных СУБД Oracle, имеющих контейнер и подключаемые базы данных, структура пользователей изменилась. Теперь пользователи разделены на два типа:

- *общие пользователи* (*Common User*) – создаются на глобальном уровне в CDB и могут управлять как CDB, так и всеми PDB, в которых у них есть привилегии. Имеют одинаковые идентификационные данные и табличное пространство в CDB и в каждой существующей и будущей PDB;
- *локальные пользователи* (*Local User*) – создаются на локальном уровне в PDB и работают только с той подключаемой базой, в которой были созданы. Идентификационные данные ограничены одной PDB.

Общие пользователи, в свою очередь, тоже делятся на два типа:

- *специальные или поставляемые с Oracle* (*Oracle-Supplied*) – к ним относятся пользователи-администраторы *SYS* и *SYSTEM*;
- *создаваемые пользователями* (*User-Created*), имена которых обязательно должны начинаться с *C##* или *c##*.

1.6. Установка и настройка Oracle Database 18c XE

Установка Oracle Database 18c XE особой сложности не представляет, однако необходимо обратить внимание на следующее [12]:

- перед установкой Oracle Database XE 18c следует удалить все существующие базы данных Oracle XE или базы данных с SID XE из целевой системы;
- установщик будет использовать для создания базы данных SID XE (любой SID, кроме SID XE, будет запрещен);
- см. [12], раздел «Exporting and Importing Data between Oracle Database XE 11.2 and 18c», если необходимо переместить данные из XE 11.2 в XE 18c.

Для установки Oracle Database XE необходимо выполнить следующие действия:

1. Войдите в Windows с правами администратора Windows.

2. Если установлена переменная среды *ORACLE_HOME*, удалите ее.

3. Загрузите версию Oracle Database XE для Windows и разархивируйте загруженный zip-файл во временную папку. Запустите *setup.exe*.

4. В окне *Welcome to the InstallShield Wizard for Oracle Database 18c Express Edition* нажмите кнопку *Next*.

5. В окне *License Agreement* ознакомьтесь с соглашением, выберите *I accept the terms in the license agreement* и нажмите кнопку *Next*.

6. Установщик проверяет версию Windows, права администратора и отсутствие на компьютере службы Oracle Database XE. Если какая-то проверка не проходит, отмените установку, устраните проблему и снова запустите установку.

7. В окне *Choose Destination Location* примите значение по умолчанию или нажмите кнопку *Change*, чтобы выбрать другой каталог установки. Не выбирайте каталог, в имени которого есть пробелы. Затем нажмите кнопку *Next*.

8. В окне *Specify Database Passwords* введите и подтвердите единый пароль для учетных записей базы данных *SYS*, *SYSTEM* и *PDBADMIN*. Затем нажмите кнопку *Next*. Пароль должен соответствовать рекомендациям Oracle.

> Внимание! Запомните пароль, вводимый при установке Oracle Database XE. Если он будет утерян, то понадобится повторная установка Oracle Database XE.

9. В окне *Summary* просмотрите параметры установки и, если они удовлетворены, нажмите кнопку *Install*. В противном случае нажмите кнопку *Back*.

10. После завершения установки откроется окно, показанное на рис. 1.

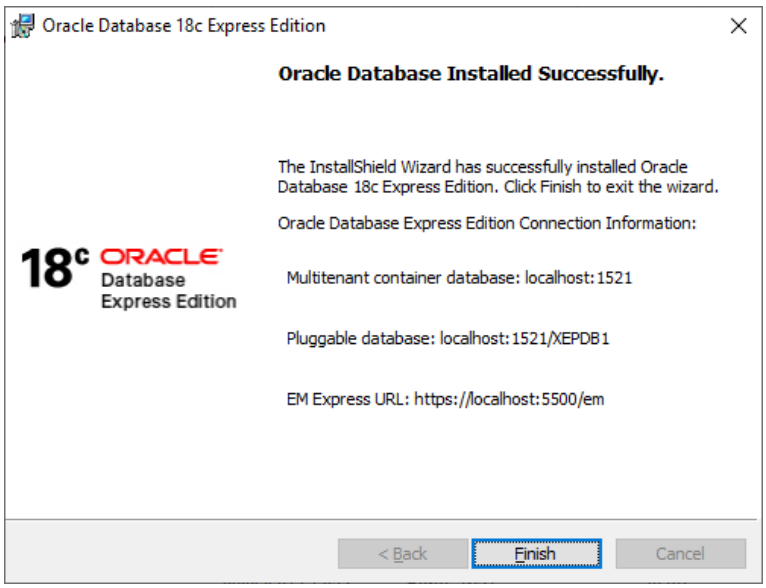


Рис. 1

Запишите строки подключения для баз данных CDB и PDB, а также URL-адрес web-интерфейса EM Express. Нажмите кнопку *Finish*.

В табл. 1 показаны пути к наиболее важным файлам Oracle Database 18c XE.

Таблица 1

Имя и расположение файла	Назначение
<INSTALL_DIR>	Oracle Base. Это корень дерева каталогов Database Oracle XE
<INSTALL_DIR>\dbhomeXE	Oracle Home. В этом каталоге установлена Oracle Database XE. Содержит каталоги исполняемых файлов Oracle Database XE и сетевых файлов
<INSTALL_DIR>\oradata\XE	Файлы базы данных
<INSTALL_DIR>\diag\rdbms\XE\XE\trace	Журналы диагностики. Журнал предупреждений *<INSTALL_DIR>\diag\rdbms\XE\XE\trace\alert_XE.log*
<INSTALL_DIR>\cfgtoollogs	Журналы установки, создания и настройки базы данных. Файл *<INSTALL_DIR>\cfgtoollogs\dbca\xe\xe.log* содержит результаты скрипта создания базы данных
%Program Files%\Oracle\Inventory\logs	Журналы установки программного обеспечения

Каталог *<INSTALL_DIR>* – это каталог установки, выбранный вами. Каталог установки по умолчанию *C:\app\<username>\product\18.0.0*, где *<username>* – имя пользователя Windows, выполняющего установку.

В табл. 2 описаны важные параметры файла откликов. Значение параметра не может быть пустым. Необходимо указать допустимое значение параметров.

Таблица 2

Параметр	Назначение	Значение по умолчанию
INSTALLDIR	Расположение каталога установки. Замените *[USERNAME]* текущим пользователем	*INSTALLDIR=C:\app\[USERNAME]\product\18.0.0*
PASSWORD	Пароль базы данных XE. Сбросьте пароль после завершения установки	*PASSWORD=passwordvalue*
LISTENER_PORT	Порт слушателя	LISTENER_PORT=0 Если порт слушателя равен 0, доступные порты будут выделяться начиная с 1521
EMEXPRESS_PORT	Порт EM Express	EMEXPRESS_PORT=0. Если для порта EM express установлено значение 0, то доступный порт будет автоматически выделяться начиная с 5550
CHAR_SET	Множество символов базы данных	*CHAR_SET=AL32UTF8*

1.7. Пользовательский интерфейс Oracle Database 18c XE

Включает команды системного меню и web-интерфейс Enterprise Manager Express. Доступ к основным функциям Oracle обеспечивает системное меню *Start → Oracle – OraDB18Home1* (рис. 2).

Системное меню Oracle Database 18c XE предоставляет в распоряжение администратора большое количество инструментальных средств, работа с которыми описана в онлайн-документации [10, 11].

Web-интерфейс Enterprise Manager Express обеспечивает выполнение основных операций администрирования базы данных:

- мониторинг ресурсов, используемых базой данных;
- мониторинг хранилищ;
- мониторинг сеансов базы данных;
- просмотр параметров инициализации базы данных;
- настройку параметров безопасности.

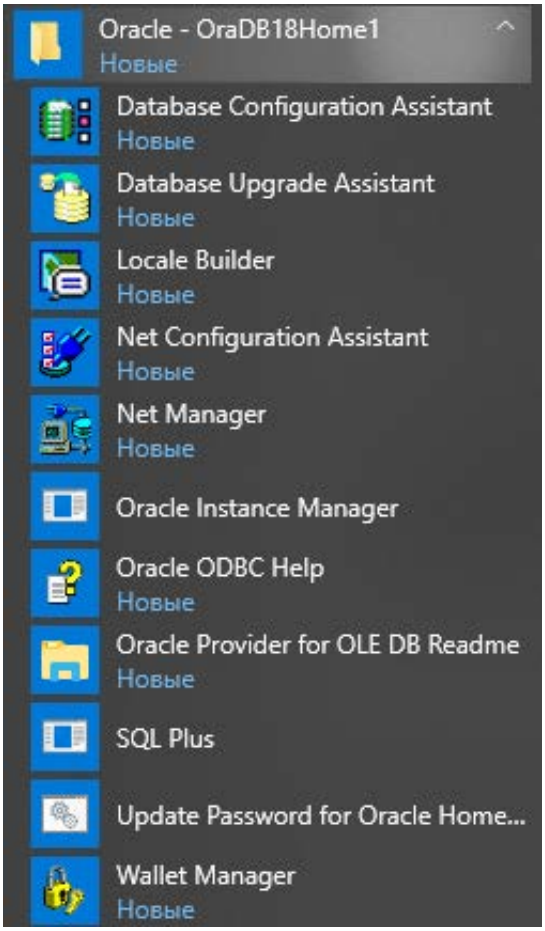


Рис. 2

Детальную информацию по этим операциям уровня базы данных также можно получить в онлайн-документации [10, 11]. Учитывая специфику пособия, оста-

новимся на администрировании базы данных SQL-ориентированными инструментами, к которым относятся:

- SQL*Plus – утилита командной строки (из системного меню);
- Oracle SQL Developer (далее – SQL Developer).

Основным компонентом Oracle является ядро, которое обрабатывает физический перенос данных между памятью и внешним хранилищем, управляет параллелизмом и обеспечивает изоляцию транзакций. Компонентом ядра является оптимизатор, выбирающий наиболее эффективный способ доступа к физической структуре данных и алгоритмы обработки SQL-запросов.

Прикладные программы и пользователи общаются с ядром с помощью *языка структурированных запросов SQL* (*Structured Query Language*), являющегося стандартом взаимодействия с реляционной базой данных. Он представляет собой набор простых инструкций, позволяющих извлекать, обновлять и удалять данные, а также создавать, изменять и удалять объекты баз данных.

Oracle SQL – это практически полностью завершенная реализация стандарта ANSI/ISO/IEC SQL:2011. Корпорация Oracle играет важную роль в стандартизации языка SQL и занимается этим на протяжении многих лет [7].

Выполнение команд языка SQL является единственным способом взаимодействия приложения с Oracle Database. Графические интерфейсы пользователя скрывают от пользователя инструкции SQL, однако под графическим оформлением приложение всегда использует для связи с Oracle язык SQL.

Oracle поставляет процедурное расширение SQL – *PL/SQL* (*Procedural Language/SQL*), добавляющее процедурные элементы (условное управление и циклы). В PL/SQL можно объявлять константы, типы и переменные, создавать блоки, процедуры, функции, пакеты и триггеры, которые хранятся в базе данных.

*Утилита SQL*Plus* – инструмент интерактивных и пакетных запросов. Имеет интерфейс командной строки и устанавливается с каждой инсталляцией Oracle (находится в каталоге *ORACLE_HOME/bin*). Имеет свои команды и окружение. Позволяет ввести и запустить команды SQL*Plus, операторы SQL и PL/SQL, команды операционной системы, решающие следующие задачи:

- создание, выполнение, сохранение и печать результатов запросов;
- определение таблиц и других объектов базы данных;
- создание и выполнение пакетных сценариев;
- администрирование базы данных.

Операторы можно отправлять в интерактивном режиме или в виде сценариев SQL*Plus. При загрузке SQL*Plus выдает приглашение: *SQL*>. В приглашении SQL можно ввести операторы, выполняющие административные задачи (завершение работы базы данных, создание нового пользователя и т. д.), или можно за-

просить, вставить, обновить и удалить данные. Одну инструкцию SQL можно ввести в нескольких строках. Каждый оператор должен заканчиваться точкой с запятой (;). Можно повторно запустить инструкцию, введя косую черту (/).

SQL Developer – интегрированная среда разработки на языках SQL и PL/SQL с возможностью администрирования базы данных, ориентированная на применение в Oracle. Это графическая версия утилиты SQL*Plus. Написана на языке Java, работает на всех платформах, где доступна среда выполнения Java SE. SQL Developer можно использовать в Windows, Linux и Mac OSX. Обеспечивает доступ к базам данных Oracle 9i, 10g, 11g, 12c и 18c, а также некоторым другим (Times Ten, Microsoft Access, MySQL, MS SQL Server).

1.8. Подключение к Oracle Database 18c XE

Локальное подключение с помощью проверки подлинности ОС. Устанавливающий Oracle Database XE пользователь Windows автоматически добавляется в группу операционной системы *ORA_DBA*, что дает ему права *SYSDBA*. Для подключения к базе данных можно использовать следующие команды:

```
cd <oracle_home>\bin
sqlplus / as sysdba
```

В приведенных командах необходимо заменить <*oracle_home*> на путь к *Oracle Home* (см. табл. 1). Эти команды соединяют с корневым контейнером *CDB$ROOT* мультиарендной базы данных (CDB) в качестве пользователя базы данных *SYS*. Этот метод подключения к базе данных работает, даже если не запущена служба слушателя *Net Services*.

Слушатель Net Services и службы по умолчанию. Слушатель базы данных *Net Services* позволяет подключаться к базе данных по протоколу TCP/IP с того же компьютера или с других компьютеров в сети. Конфигурацию слушателя можно просмотреть с помощью следующих команд [12]:

```
cd <oracle_home>\bin
lsnrctl status
```

В результате будут показаны значения следующих параметров:

- порт слушателя;
- список служб, зарегистрированных в слушателе;
- порт Enterprise Manager Express;
- имя файла конфигурации, используемого слушателем;
- имя файла журнала, который вы указываете при подключении к базе данных через слушатель.

Службы по умолчанию, созданные Oracle Database XE, – это *XE* и *XEPDB1*. Служба *XE* подключает к корневому контейнеру базы данных *CDB$ROOT*, а служба *XEPDB1* – к подключаемой базе данных по умолчанию *XEPDB1*, созданной во время установки. Для каждой новой подключаемой базы данных (PDB) будет создана новая служба по умолчанию с тем же именем, что и PDB. Если экземпляр Oracle Database XE выключен, команда *lsnrctl status* не будет отображать службы, к которым можно подключиться.

Подключение к Oracle Database с помощью простого метода подключения. Можно подключиться к базе данных, используя строки [12]:

- мультиарендная контейнерная база данных: *хост[:порт]*;
- подключаемые базы данных: *host[:port] / service_name*.

XEPDB1 – это имя сервиса, определенного для первого pdb-файла, создается по умолчанию. Вы можете заменить *XEPDB1* именем другой PDB, к которой хотите подключиться. Указывать порт необязательно, если слушатель настроен с портом по умолчанию 1521. При использовании другого порта необходимо указать номер порта. Строки подключения для локальных подключений показываются на последнем экране установки. При подключении с удаленного компьютера необходимо указать имя хоста (где установлен XE) вместо *localhost*. Для успешного подключения слушатель базы данных Net Services должен быть запущен на узле базы данных на указанном порту.

Например, можно подключиться к корневому контейнеру базы данных с клиентского компьютера с помощью SQL*Plus, используя следующие команды:

```
cd <oracle_home>\bin
sqlplus system@dbhost.example.com:1521
```

К подключаемой базе данных *XEPDB1* по умолчанию можно подключиться с помощью следующих команд:

```
cd <oracle_home>\bin
sqlplus system@dbhost.example.com:1521/XEPDB1
```

Замените *dbhost.example.com* на имя хоста базы данных. При необходимости замените 1521 номером порта, который прослушивает слушатель. Можете заменить *XEPDB1* на имя другой PDB, к которой хотите подключиться.

Чтобы сократить строки подключения или избежать жесткого кодирования имени хоста и порта в коде приложения и сценариях DBA, можно определить на клиентах базы данных псевдоним для строки подключения в файле конфигурации *<oracle_home>\network\admin\tnsnames.ora*.

Создание подключения к базе данных с помощью SQL Developer. Для создания нового подключения к базе данных [10]:

1. В навигаторе подключений SQL Developer щелкните правой кнопкой мыши узел *Connections* и выберите *New Connection*. Откроется диалоговое окно *New / Select Database Connection* (рис. 3).

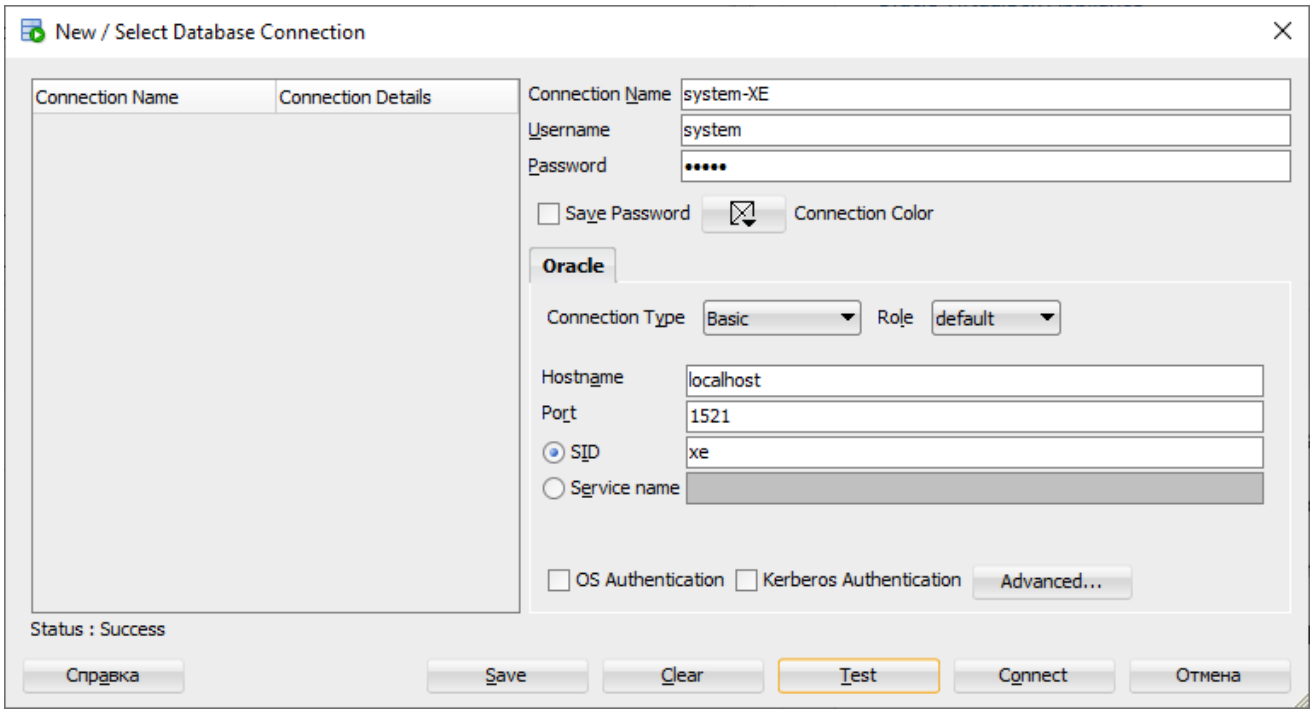


Рис. 3

2. Введите следующую информацию:

- в поле *Connection Name* – имя подключения к базе данных; в поле *Username* – имя пользователя, для которого создается подключение; в поле *Password* – пароль пользователя;
- в поле *Connection Type* выберите тип подключения к базе данных. Типы подключения: *Basic, TNS, LDAP, Advanced, Local/Bequeath*. Состав полей будет меняться в соответствии с выбранным типом соединения. В этом примере описаны поля для базового типа подключения (*Basic)*;
- в поле *Role* выберите *Default* или *SYSDBA* в зависимости от роли;
- в поле *Hostname* введите имя хоста, на котором находится база данных; в поле *Port* – порт для базы данных; в поле *SID* – идентификатор базы данных, если создается соединение с базой данных для не-CDB пользователя или комплексной контейнерной базой данных для CDB-пользователя; в поле *Service name* – имя службы для подключаемой базы данных (PDB), включая имя домена (если подключение предназначено для пользователя PDB).

Внимание! При создании подключения к базе данных не-CDB, CDB или PDB для пользователя с правами администратора (например, *SYS*) обычно указывается *SYSDBA* в поле *Role*.

3. При необходимости нажмите кнопку *Test*, чтобы проверить, что предоставленные данные позволят пользователю подключиться к базе данных.

4. По завершении нажмите кнопку *Connect*, чтобы подключиться к базе данных, или кнопку *Save*, чтобы сохранить подключение. Диалоговое окно создания подключения для пользователя *SYSTEM* показано на рис. 3.

После этого в навигаторе подключений SQL Developer появляется новое подключение *system-XE* (рис. 4).

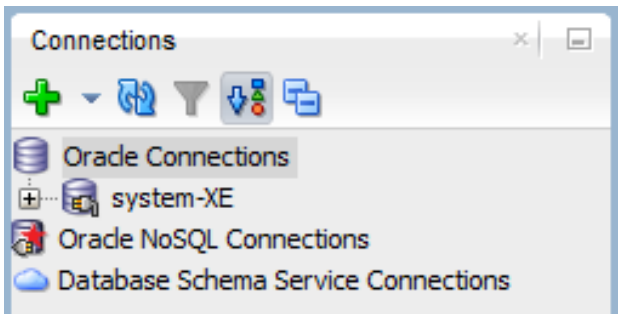


Рис. 4

Теперь, чтобы подключиться к контейнерной базе данных CDB в качестве пользователя *SYSTEM*, достаточно раскрыть узел этого подключения и в открывшемся окне *Connection Information* ввести пароль пользователя *SYSTEM*. Если при создании подключения выбрать опцию *Save Password*, то пароль при каждом подключении к базе данных запрашиваться не будет.

Разорвать соединение с базой данных можно, щелкнув правой кнопкой мыши по имени подключения в навигаторе и выбрав команду *Disconnect*.

1.9. Запуск и останов Oracle Database 18c XE

После установки служба Oracle Database должна быть запущена, а база данных должна быть запущена и открыта перед использованием [12]. По умолчанию при запуске службы Oracle Database запускается и открывается контейнерная база данных, а все подключаемые базы данных должны быть открыты перед использованием. Ниже приведена команда автоматического открытия подключаемых баз данных при запуске службы Oracle.

Запуск и завершение работы с помощью служб Windows. Можно запустить или остановить базу данных с помощью служб Windows Services:

1. В меню *Start* введите *services.msc* в поле поиска и нажмите *Enter*.

2. Найдите службу *OracleServiceXE* в окне *Services*.

3. Щелкните правой кнопкой мыши имя службы и выберите в меню *Start* или *Stop*.

4. Чтобы задать свойства запуска, щелкните правой кнопкой мыши и выберите *Properties*, а затем в диалоговом окне выберите *Automatic*, *Manual* или *Disabled* из списка *Startup type*.

Внимание! Для управления подключениями к базе данных из сети можно запустить или остановить слушатель сетевых служб. Он выполняется как служба Windows с именем *OracleOraDB18Home<n>TNSListener*, где <*n*> – номер, выбранный установщиком Oracle Database XE на основе других домов Oracle, ранее установленных на хосте (для первой установки – *OracleOraDB18Home1TNSListener*).

Запуск и завершение работы с помощью SQL*Plus. Можно завершить работу и запустить базу данных с помощью SQL*Plus. Для завершения работы базы данных выполните следующие команды SQL*Plus:

```
cd <oracle_home>\bin
sqlplus / as sysdba
SQL> SHUTDOWN IMMEDIATE
```

Чтобы запустить базу данных, выполните команды:

```
SQL> STARTUP
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
```

Настройка автоматического открытия подключаемых баз данных. По умолчанию подключаемая база данных (например, XEPDB1) автоматически не открывается и должна быть открыта вручную с помощью кода SQL, приведенного выше.

Подключаемые базы данных можно настроить на автоматическое открытие при открытии контейнерной базы данных, подключившись к контейнерной базе данных с помощью SQL*Plus (как указано выше) и выполнив следующий код SQL:

```
SQL> ALTER PLUGGABLE DATABASE ALL OPEN;
SQL> ALTER PLUGGABLE DATABASE ALL SAVE STATE;
```

Внимание! При создании дополнительных подключаемых баз данных эти команды должны быть выполнены снова.

2. КРАТКО ОБ ORACLE SQL DEVELOPER

В пособии мы используем SQL Developer в качестве графического пользовательского интерфейса для выполнения запросов, изучения объектов, запуска отчетов и выполнения сценариев [14]. Скриншоты получены для версии 18.4.0.376.

2.1. УСТАНОВКА И НАСТРОЙКА SQL DEVELOPER

Продукт входит в состав Oracle Database 11g, 12c и 18c. Его можно бесплатно скачать с сайта Oracle (необходима учетная запись Oracle). Для установки в Windows достаточно разархивировать загруженный zip-файл (например, в корневой каталог *C:*). Будет создана папка *C:\sqldeveloper* с нужными файлами и папками в ней. Для запуска SQL Developer нужно войти в каталог установки и дважды щелкнуть файл *sqldeveloper.exe*.

> Внимание! SQL Developer для Windows не создает ярлыки меню или значки на рабочем столе. При необходимости их можно создать вручную. SQL Developer не создает записей в реестре. Деинсталлировать продукт можно, удалив каталог SQL Developer, созданный при распаковке архива.

Для работы среды необходим комплект Java-разработчика JDK (Java Development Kit). SQL Developer можно скачать с JDK или без него, если JDK уже установлен. Если выбрана загрузка без JDK, то при первом запуске потребуется указать местонахождение JDK и ввести полный путь к файлу *java.exe*.

При первом запуске SQL Developer открывается начальная страница, содержащая ссылки на обнаруженную базу данных, ресурсы (документацию, учебники, форум) и связанные инструменты. Для большинства пользователей вполне достаточна «коробочная» настройка, к которой можно добавить:

- установку местоположения файлов скриптов по умолчанию;
- включение вывода «в полоску».

Чтобы указать расположение по умолчанию файлов скриптов, выберите *Tools → Preferences*. В диалоговом окне настроек выберите *Databases → Worksheet* и введите предпочтительное расположение скриптов в поле *Select default path to look for scripts*, например *c:\scripts*. В этом же окне можно обеспечить удобочитаемый вывод включением опции *Grid in checker board or Zebra pattern*. Теперь каждая вторая строка будет выглядеть темнее.

В SQL Developer онлайн доступна короткая обучающая программа (доступна также в Oracle SQL Developer User's Guide). Программа создает три таблицы, по-

следовательность, представление и процедуру PL/SQL для небольшой базы данных и вставляет данные в таблицы. Для доступа к обучающей программе:

- щелкните *Help*, затем *Table of Contents*;
- раскройте узел *SQL Developer Online Help Release 18.1*;
- выберите *5 SQL Developer Tutorial: Creating Objects for a Small Database*.

2.2. Интерфейс SQL Developer

Пользовательский интерфейс включает навигатор объектов, панель инструментов и рабочий лист (Worksheet).

Навигатор объектов. Позволяет просматривать объекты. Объект активируется щелчком в панели навигатора. Вкладки в окне позволяют получить дополнительные сведения о выбранном объекте (на рис. 5 изображена таблица *Customers*). Данные таблицы отображаются на вкладке *Data*. В окне результатов по умолчанию отображаются первые 50 записей таблицы. На вкладке *SQL* отображаются SQL-запросы создания объекта.

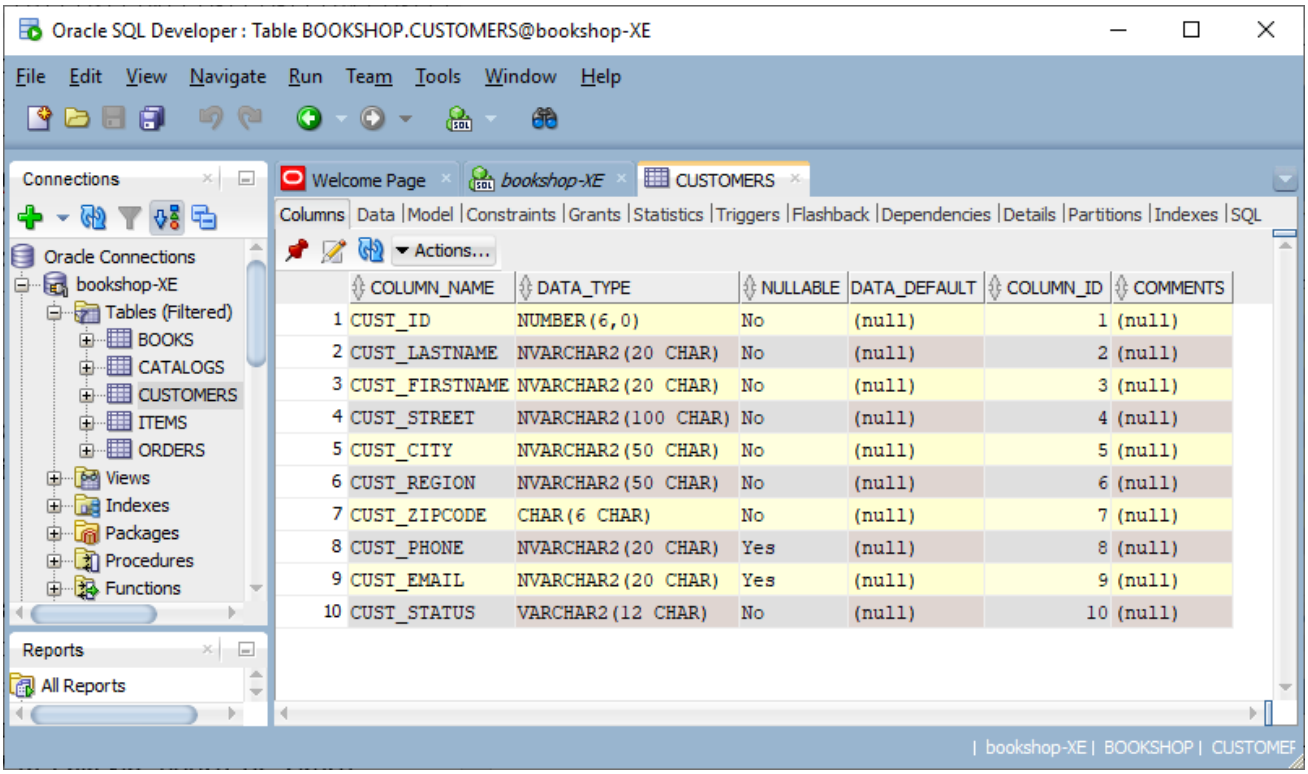
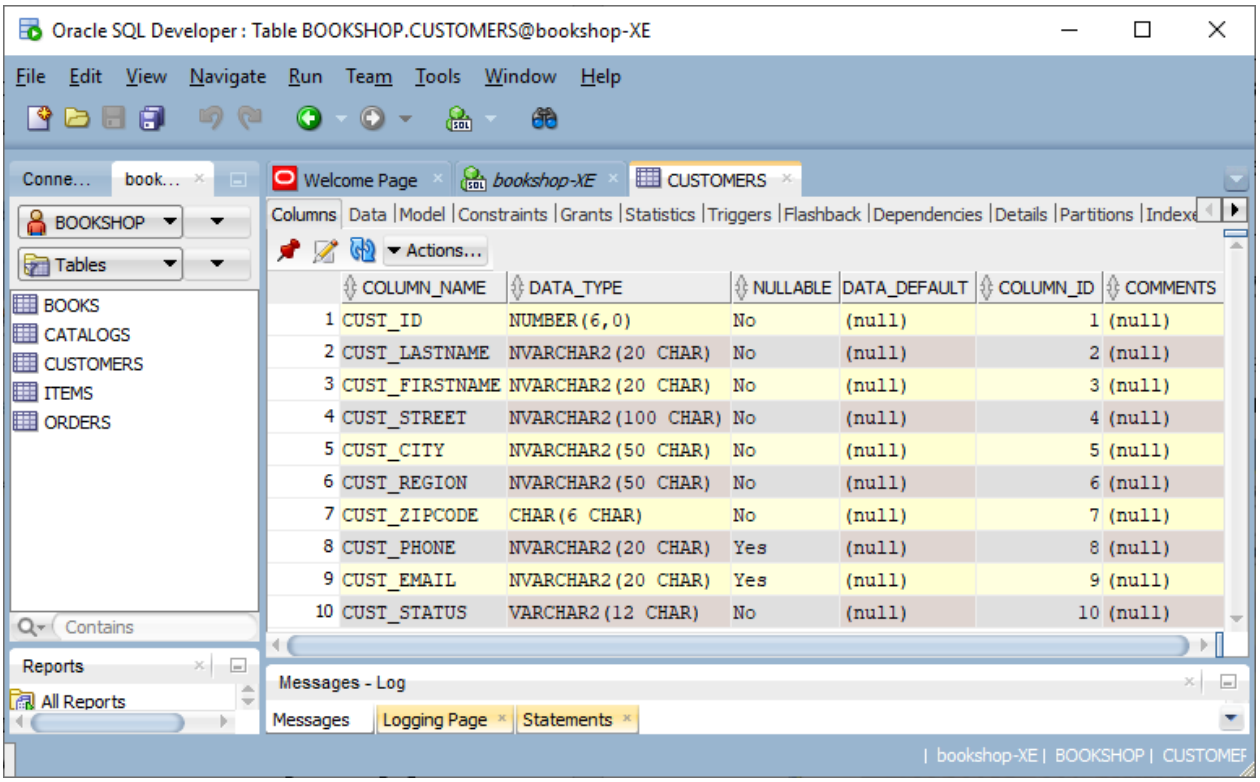


Рис. 5

Пользователь может также исследовать не принадлежащие ему объекты, к которым он имеет доступ. Объекты, к которым имеется доступ, можно увидеть, развернув узел *Other Users* в нижней части списка объектов. Там же отображаются все пользователи базы данных, даже если вы не видите их объектов.

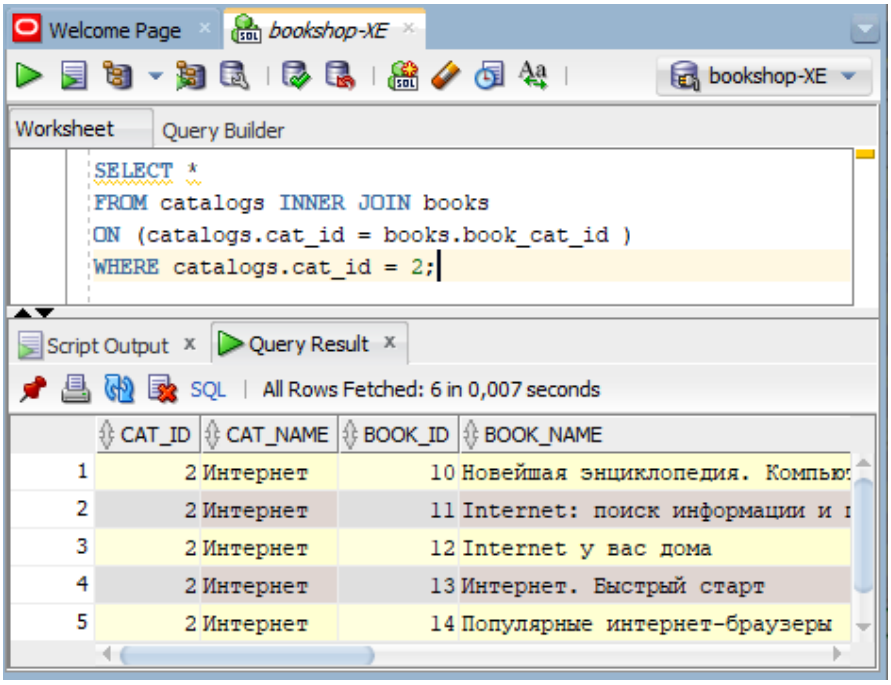
Браузер схем. Делает просмотр объектов базы данных более удобным. Для вызова щелкните правой кнопкой мыши на подключении в навигаторе и выберите *Schema Browser* (рис. 6).



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	CUST_ID	NUMBER(6,0)	No	(null)	1	(null)
2	CUST_LASTNAME	NVARCHAR2(20 CHAR)	No	(null)	2	(null)
3	CUST_FIRSTNAME	NVARCHAR2(20 CHAR)	No	(null)	3	(null)
4	CUST_STREET	NVARCHAR2(100 CHAR)	No	(null)	4	(null)
5	CUST_CITY	NVARCHAR2(50 CHAR)	No	(null)	5	(null)
6	CUST_REGION	NVARCHAR2(50 CHAR)	No	(null)	6	(null)
7	CUST_ZIPCODE	CHAR(6 CHAR)	No	(null)	7	(null)
8	CUST_PHONE	NVARCHAR2(20 CHAR)	Yes	(null)	8	(null)
9	CUST_EMAIL	NVARCHAR2(20 CHAR)	Yes	(null)	9	(null)
10	CUST_STATUS	VARCHAR2(12 CHAR)	No	(null)	10	(null)

Рис. 6

Если SQL-запрос смотрится не слишком наглядно, можно его отформатировать с помощью сочетания клавиш *Ctrl + F7* (рис. 7).



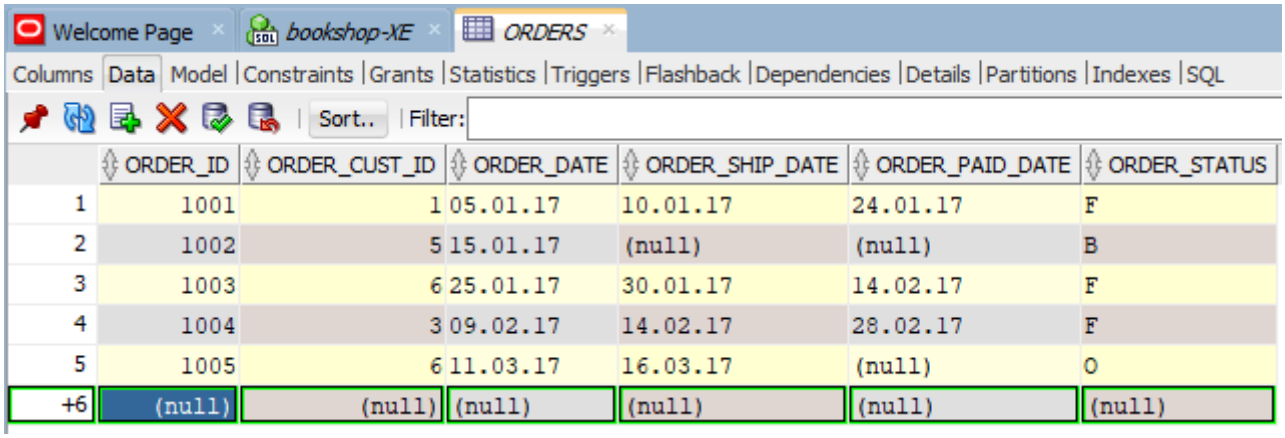
	CAT_ID	CAT_NAME	BOOK_ID	BOOK_NAME
1	2	Интернет	10	Новейшая энциклопедия. Компью
2	2	Интернет	11	Internet: поиск информации и
3	2	Интернет	12	Internet у вас дома
4	2	Интернет	13	Интернет. Быстрый старт
5	2	Интернет	14	Популярные интернет-браузеры

Рис. 7

Рабочий лист (Worksheet). Обеспечивает ввод инструкций для получения и изменения данных. Можно вводить операторы SQL и PL/SQL. Поддерживаются также некоторые команды SQL*Plus, такие как *COLUMN*, *DESCRIBE* и *SPOOL*. Полный список поддерживаемых и неподдерживаемых команд SQL*Plus приведен в интерактивном руководстве *Oracle SQL Developers User's Guide*. Рабочий лист автоматически открывается при подключении к базе данных. Чтобы открыть другой лист или закрыть один из открытых, нажмите на значок *Worksheet SQL* или выберите пункт меню *Tools → SQL Worksheet*.

Внимание! Если на рабочем листе записано несколько операторов, то они должны заканчиваться точкой с запятой (;) или косой чертой (/) в отдельной строке, иначе сеанс вернет сообщение об ошибке *ORA-00933: SQL command not properly ended*.

Для ввода новой записи в таблицу необходимо открыть вкладку *Data* и нажать значок с зеленым плюсом. Активируются значки фиксации и отката (рис. 8). Вводятся данные в поля, где указано *null*, после чего необходимо нажать *COMMIT*. В результате этих действий выполняется оператор *INSERT*.



	ORDER_ID	ORDER_CUST_ID	ORDER_DATE	ORDER_SHIP_DATE	ORDER_PAID_DATE	ORDER_STATUS
1	1001	1	05.01.17	10.01.17	24.01.17	F
2	1002	5	15.01.17	(null)	(null)	B
3	1003	6	25.01.17	30.01.17	14.02.17	F
4	1004	3	09.02.17	14.02.17	28.02.17	F
5	1005	6	11.03.17	16.03.17	(null)	O
+6	(null)	(null)	(null)	(null)	(null)	(null)

Рис. 8

Работа с запросами и скриптами осуществляется следующим образом.

Выполнение одного оператора. Осуществляется командой *Run Statement* (*Ctrl + Enter*, или *F9*, или значок «большой зеленый треугольник»). Если на рабочем листе несколько операторов, то команда *Run Statement* будет работать с оператором под курсором при условии, что предыдущие операторы были завершены символом «;» или «/».

Зарезервированные слова SQL выделяются цветом. Функция автозаполнения предлагает имена таблиц, представлений и столбцов. Если она случайно отключена нажатием *ESC*, ее можно снова включить посредством *Auto-complete → Completion*

Insight или *Ctrl + Space*. Результат выполнения запроса появится в окне *Query Result* (рис. 9). Изменить порядок сортировки данных столбца можно, дважды щелкнув заголовок столбца в окне *Query Result*.

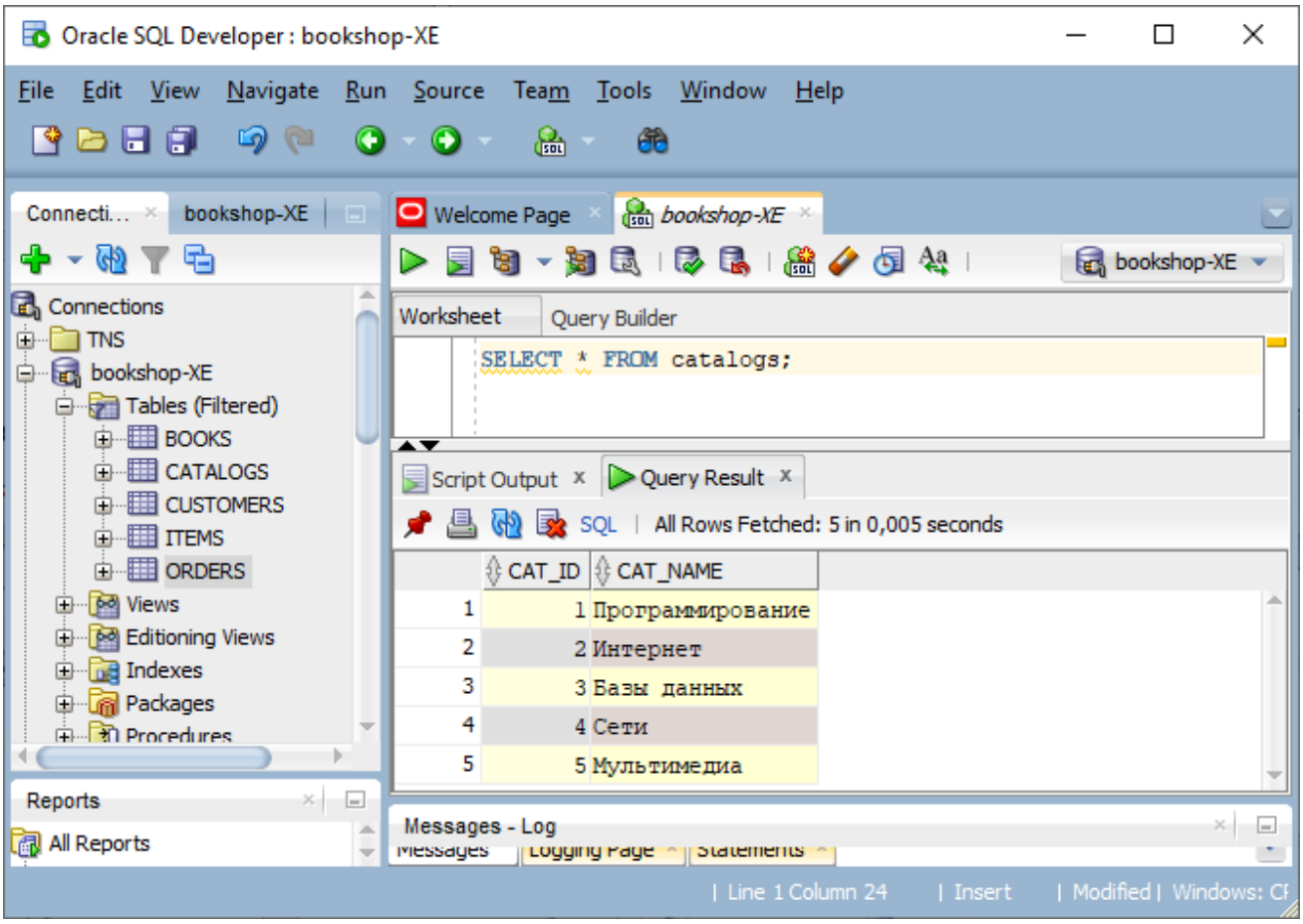


Рис. 9

Выполнение скрипта (сценария). Команда *Run Script* (*F5*) выполняет группу операторов SQL и команд SQL*Plus на рабочем листе, рассматривая их как скрипт. Используется при наличии нескольких инструкций или при необходимости форматирования выходных данных с помощью команд SQL*Plus.

Выходные данные отображаются в окне *Script Output* рядом с окном *Query Result*. Вывод почти идентичен тому, что можно увидеть в SQL*Plus (рис. 10).

Очистить окно *Script Output* для отображения только новых выходных данных можно с помощью кнопки *Clear* (изображение ластика на карандаше).

Внимание! Не все поддерживаемые команды форматирования вывода SQL*Plus правильно интерпретируются *Run Script*. Например, команда *COLUMN* не изменяет заголовки столбцов, однако *SET FEEDBACK OFF* работает, как ожидалось.

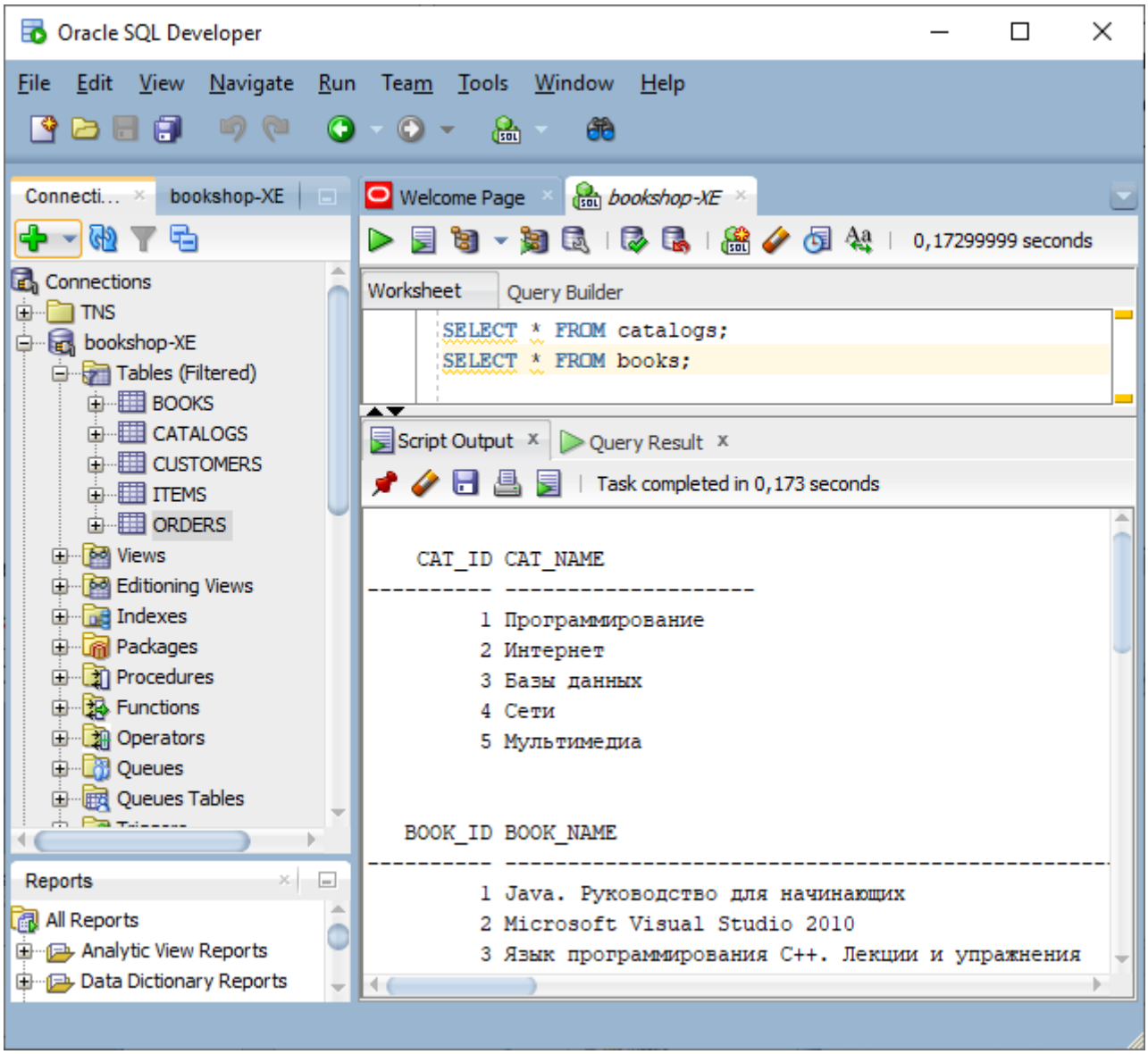


Рис. 10

Сохранение скрипта. Сложные операторы рекомендуется сохранять в скрипте, который можно выполнить позже. После ввода выберите *File → Save (Ctrl + S)* или значок с диском, чтобы открыть диалоговое окно сохранения файла. Открываемый каталог должен быть таким же, как указанный в разделе *Configuration*.

Запуск сохраненного скрипта. Существует два способа загрузки и запуска сохраненного скрипта. Во-первых, можно использовать команду @, применяемую в SQL*Plus. Напечатайте *@select_catalogs_books.sql* в рабочем листе и выберите *Run Script (F5)* (рис. 11).

Второй вариант – выберите *File → Open* и укажите файл скрипта (например, *select_catalogs_books.sql)*. Операторы, содержащиеся в этом файле, будут загружены на рабочий лист.

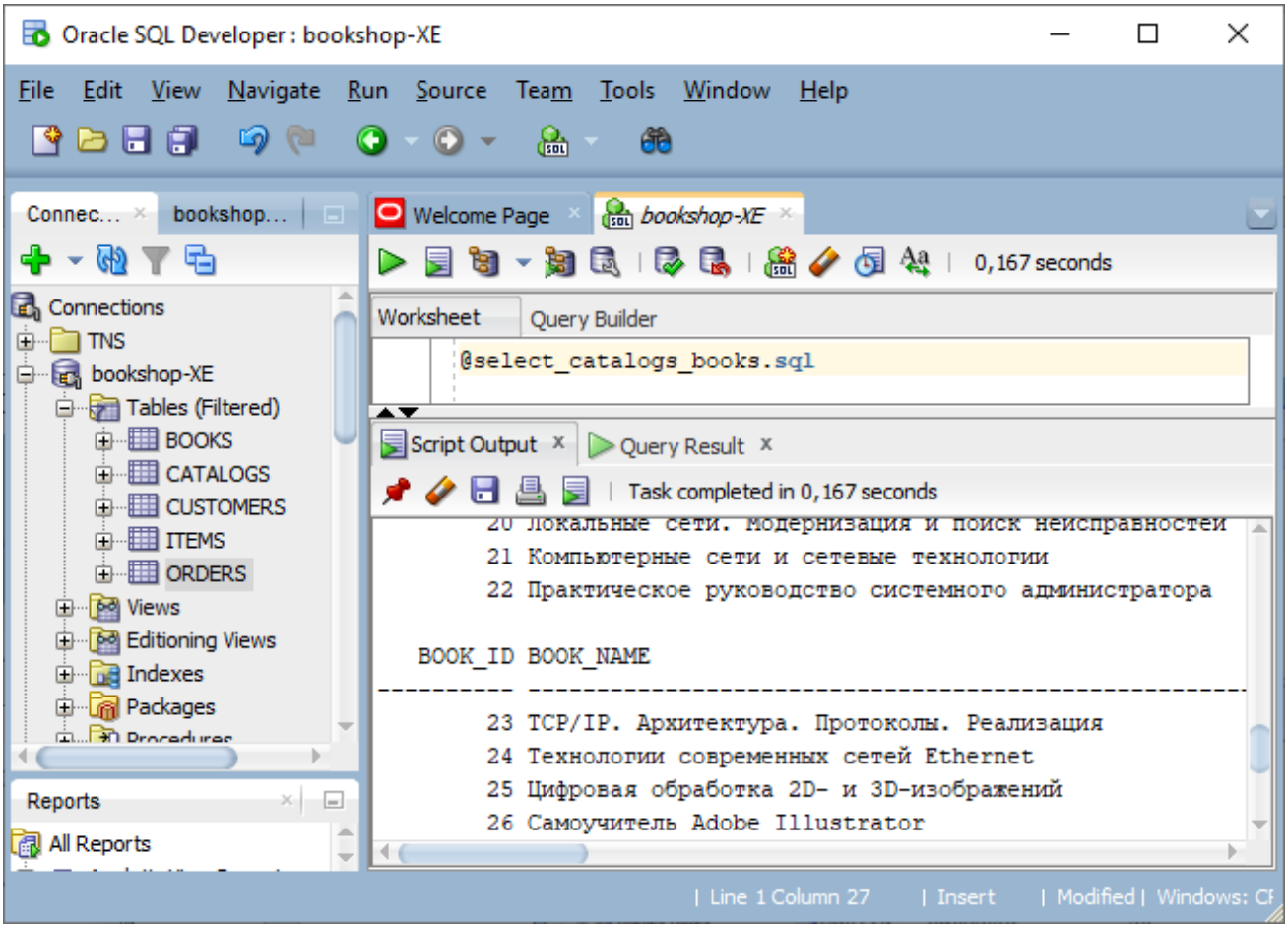


Рис. 11

Выберите подключение, которое хотите использовать, в раскрывающемся списке *Choose db Connection* в правом верхнем углу окна (пока не будет выбрано подключение, остальные кнопки будут оставаться серыми). После того как подключение будет выбрано, нажмите кнопку *Run Script*, чтобы увидеть результат.

Внимание! Если вам не нравятся горячие клавиши, определенные в SQL Developer по умолчанию, их можно изменить, перейдя в *Tools* → *Preferences* → *Shortcut Keys*, где они будут доступны для редактирования.

Экспорт данных в файл. Результаты выполнения оператора с помощью *Run Statement* могут быть экспортированы в файл. SQL Developer поддерживает различные форматы, например, такие как *CSV*, *HTML*, *XML*, *Excel*, *PDF*, *insert*, *sqlloader*. Щелкните правой кнопкой мыши в окне результатов запроса *Query Result*, выберите *Export* и укажите свой выбор (рис. 12). При этом можно даже заранее определить выбор формата по умолчанию с помощью *Tools* → *Preferences* → *Database* → *Utilities* → *Export* .

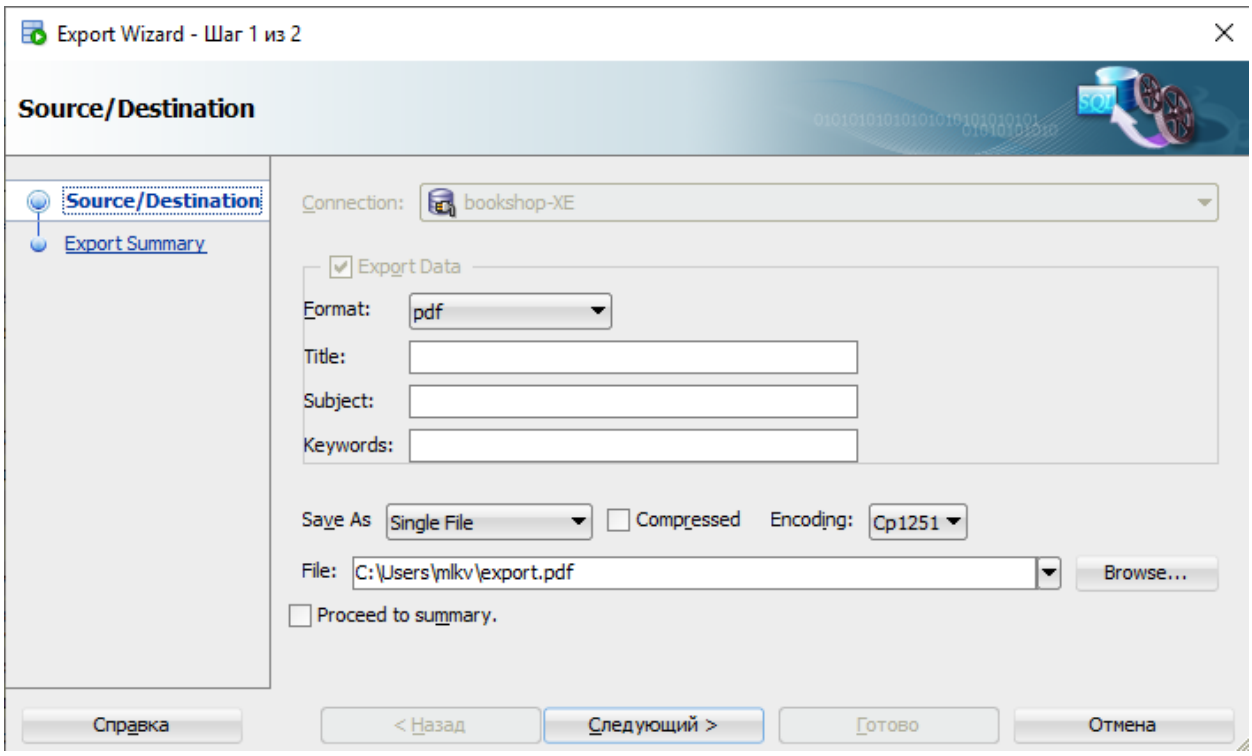


Рис. 12

Пользовательские отчеты. Можно включить оператор в отчет, определяемый пользователем, который запускается из библиотеки отчетов. Щелкните правой кнопкой мыши выходные данные в окне результатов запроса *Query Result* и выберите *Save Grid as Report* (рис. 13).

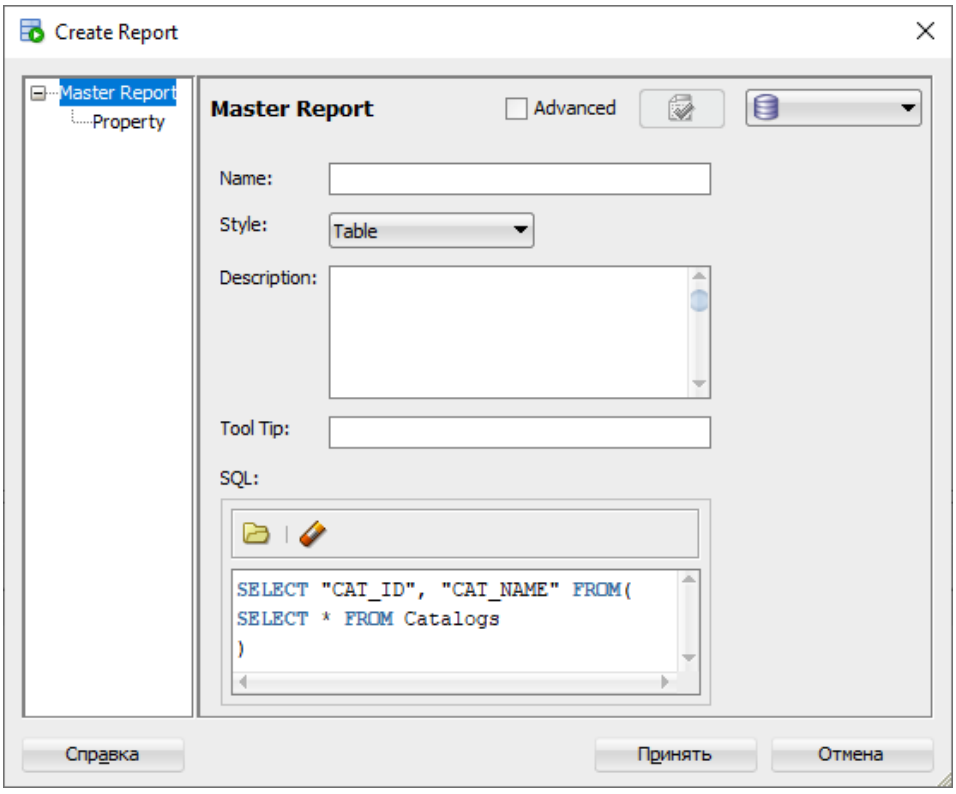


Рис. 13

Сохраните отчет, например, под именем *Report1*. Теперь оператор можно запустить, щелкнув по имени отчета. Откройте меню *View* → *Reports* – в нижней части списка у вас есть *User Defined Reports* (рис. 14). При желании можно запланировать автоматический запуск отчета с помощью графического интерфейса SQL Developer. В Oracle поставляется пакет *dbms_scheduler*, который можно найти в навигаторе под вашим соединением в узле *Scheduler*.

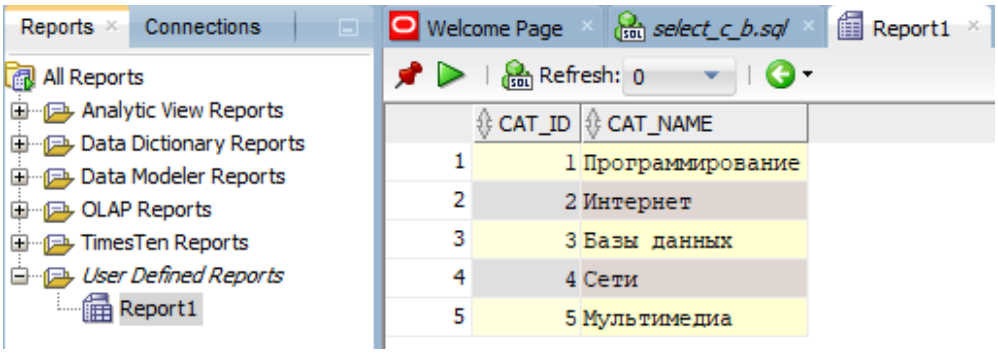


Рис. 14

2.3. Оптимизация запросов SQL

При наличии медленно выполняемых запросов можно детально исследовать их выполнение с помощью двух инструментов: плана выполнения – *Explain Plan (F10)* и автотрассировки – *Autotrace (F6)*.

Разница между ними в том, что *Explain Plan* просто отображает план выполнения инструкции, взятой из кэша базы данных, тогда как *Autotrace* фактически выполняет инструкцию, чтобы можно было отображать информацию о фактическом плане выполнения, а также статистику текущего выполнения (извлеченные строки, блоки, считанные с диска или кэша, и т. д.). Переход по *Tools* → *Preferences* → *Database* → *Autotrace-ExplainPlan* позволяет выбрать, какие столбцы и статистика будут отображаться при использовании этих инструментов. На рис. 15 показан результат применения инструмента *Explain Plan* для объединения двух таблиц. На рис. 16 показан результат применения *Autotrace (F6)*. После объяснения плана следует статистика о ходе исполнения инструкции. При первом выполнении эти статистические данные скрыты. Чтобы увидеть статистику, нужно потянуть черные стрелки, находящиеся внизу слева, вверх.

Нажав на кнопку с маленькой красной булавкой, можно сохранить этот первый вывод автотрассировки, изменить инструкцию SQL и повторно применить *Autotrace*, чтобы получить второе объяснение плана и статистику. Затем необходимо сравнить их, щелкнув правой кнопкой мыши на второй вкладке *Autotrace* и выбрав *Compare with Autotrace*. Отличия в выполнении инструкций будут отмечены красным цветом.

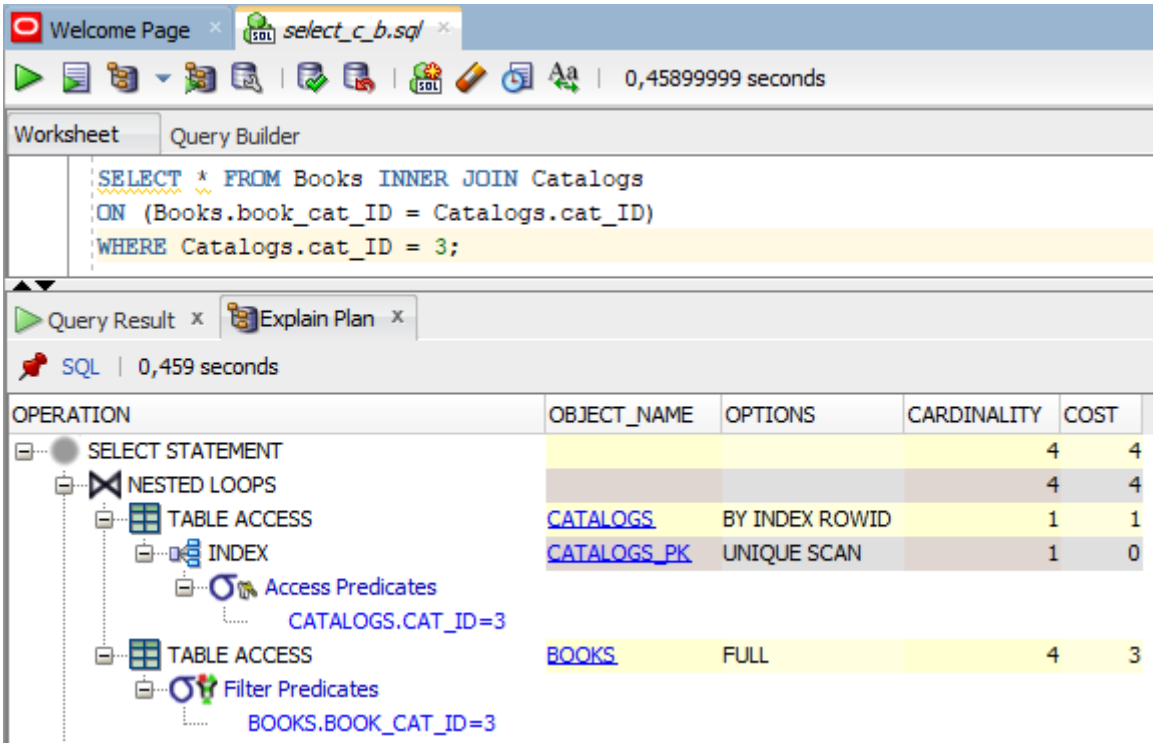


Рис. 15

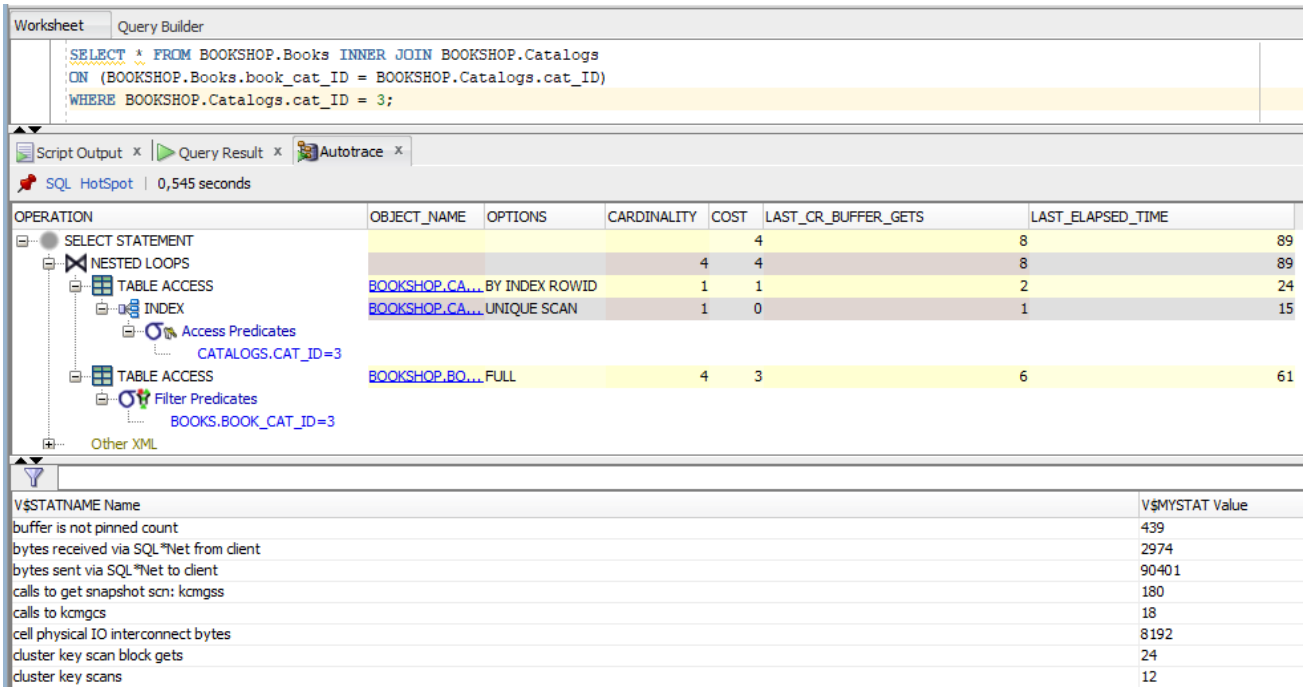


Рис. 16

2.4. Работа с кодом PL/SQL

SQL Developer позволяет также разрабатывать код PL/SQL, используя мастера и редакторы, выполняя компиляцию и отладку. Редактор для написания функций или процедур выглядит, как показано на рис. 17, и открывается через навигатор слева.

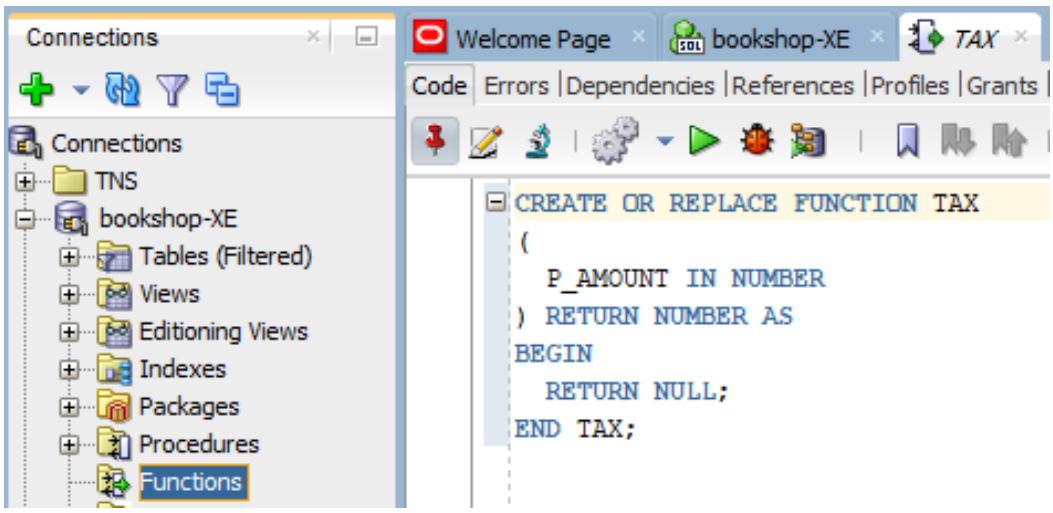


Рис. 17

Щелкните правой кнопкой мыши узел типа объекта, например функции, выберите *New Function*, заполните поля мастера ввода и нажмите кнопку *OK*. Вы находитесь в редакторе PL/SQL, где можно выполнять редактирование, компилирование, установку точек останова и отладку. Как видно из рис. 17, функция только что запущена. Необходимо заполнить содержимое исполняемой части функции вместо заглушки *RETURN NULL* и изменить возвращаемое значение.

Запуск кода PL/SQL для тестирования. Выполните компилирование функции командой *Compile*. При запуске кода с помощью значка с зеленой стрелкой *(Run)* SQL Developer предлагает анонимный блок для выполнения функции. Введите выбранное значение для теста и нажмите кнопку *OK* (рис. 18).

Отладка кода для поиска ошибок. Даже если код компилируется без ошибок, он может давать неправильные результаты. Выяснить причины этого помогает отладка. При отладке код сначала компилируется для отладки с помощью параметра компиляции под двумя маленькими серыми колесами. Скомпилированная версия функции теперь содержит как исполняемый код, так и исходный код для справочных целей во время выполнения.

Затем устанавливают точки останова (*breakpoints*) в разных местах кода (отображаются в виде красных точек на полях). После этого запускают код нажатием левой кнопки мыши на значок в виде маленькой красной божьей коровки. Когда вновь отобразится автоматически предоставляемый анонимный блок для выполнения кода, введите выбранные тестовые значения и нажмите кнопку *OK*.

Теперь SQL Developer предоставляет дополнительную панель отладки в верхней части для навигации по коду во время отладки. В нижней части есть несколько вкладок для исследования стека переменных и точек останова, а также для настройки слежения за особо важными переменными – это полезно, если в коде много переменных.

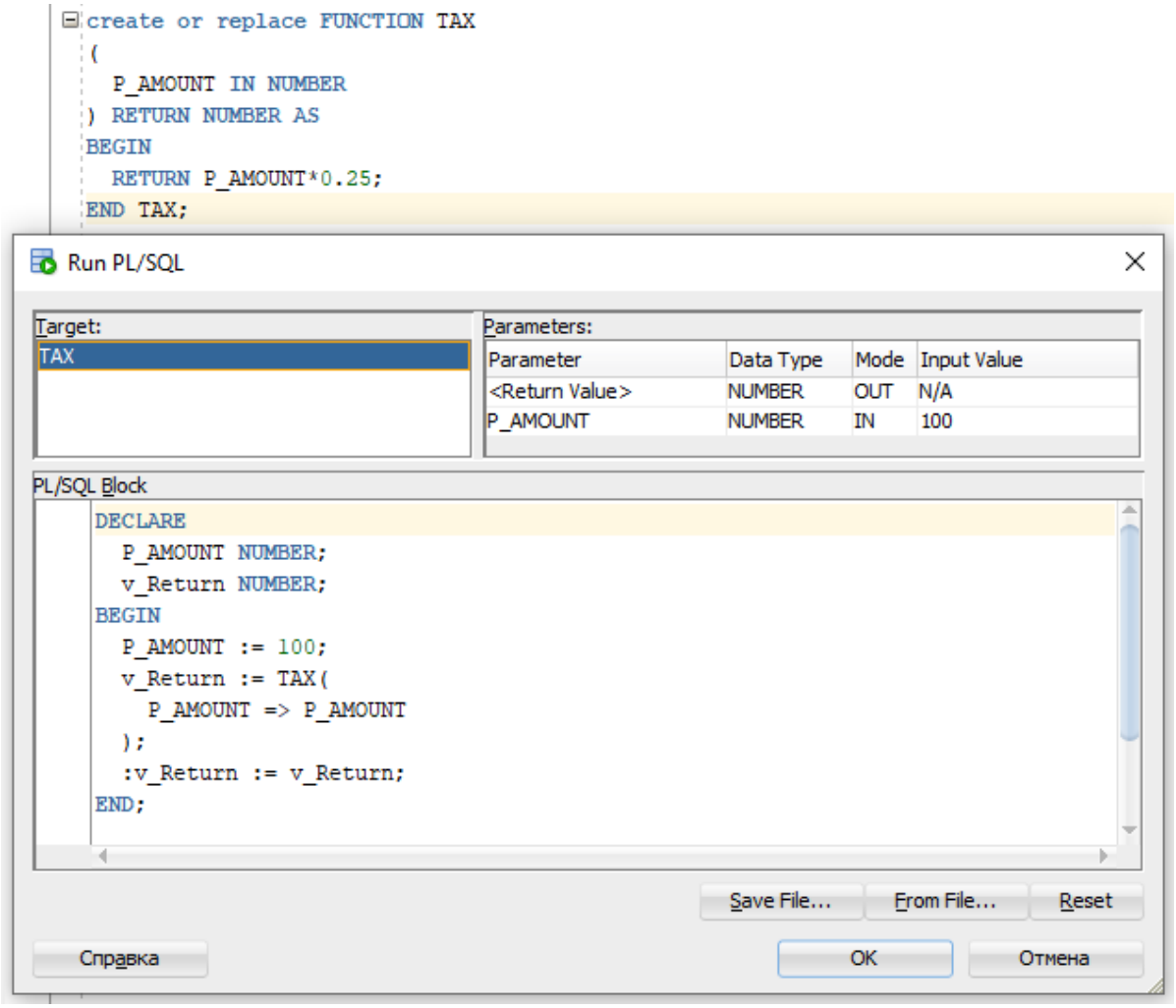


Рис. 18

Четвертый значок (красная стрелка, указывающая на страницу кода) означает *Step Into*. Эта опция обеспечивает пошаговое выполнение кода. Текущий оператор идентифицируется с помощью красной стрелки на полях. Пошаговое выполнение позволяет без спешки исследовать все этапы жизни переменных.

Примерно такова же последовательность действий при создании и отладке хранимых процедур. Мастера для создания хранимых пакетов и триггеров также довольно просты в использовании.

2.5. Data Modeler

Моделирование базы данных, а также обратное проектирование (реверсный инжиниринг) доступны через включенный в SQL Developer инструмент Data Modeler (его использование показано ниже в практическом занятии 3).

Мы рассмотрели основные функции SQL Developer. Получить дополнительные сведения можно на домашней странице разработчика:

http://www.oracle.com/technology/products/database/sql_developer/index.html.

3. ОБЗОР ЯЗЫКА ORACLE SQL

Язык SQL используют обычно двумя различными способами: интерактивным или встроенным. Интерактивный режим предполагает ввод инструкций SQL с клавиатуры и отображение результатов на экране. Использование встроенного SQL предполагает включение инструкций SQL в программу на другом языке программирования (например, Java или C). Практические занятия посвящены интерактивному использованию SQL [3, 4, 8].

3.1. Категории инструкций SQL

Понятие «запрос» в SQL гораздо шире простого извлечения данных. Чаще всего запросы SQL делят на следующие четыре категории:

- определение данных DDL (Data Definition Language);
- манипулирование данными DML (Data Manipulation Language);
- поиск данных (часто относят к категории DML);
- управление доступом к данным DCL (Data Control Language).

Инструкции DDL позволяют создавать, изменять и удалять объекты базы данных (таблицы, представления, индексы, ограничения, синонимы, последовательности и пр.). Они начинаются с одного из трех ключевых слов:

- *CREATE* – создать новый объект базы данных;
- *ALTER* – изменить структуру существующего объекта базы данных;
- *DROP* – удалить объект базы данных.

Например, с помощью инструкции *CREATE VIEW* можно создавать представление. С помощью инструкции *ALTER TABLE* можно изменить структуру таблицы (например, добавив, переименовав или удалив столбец). С помощью инструкции *DROP INDEX* можно удалить индекс.

Если инструкции DDL позволяют изменять *структуру* базы данных, то *инструкции манипулирования данными DML* позволяют изменять *содержимое* базы данных. Эти инструкции начинаются со следующих ключевых слов:

- *INSERT* – добавить строки в таблицу;
- *UPDATE* – изменить значения столбцов существующих строк;
- *DELETE* – удалить строки из таблицы.

Внимание! Для загрузки больших объемов данных в базу данных Oracle используются специально разработанные для этой цели инструменты – *Data Pump* в Oracle Database 10g и выше, *Export* и *Import* в предыдущих релизах Oracle и SQL*Loader.

Инструкции манипулирования данными всегда рассматриваются как часть *транзакции*: изменения базы данных, вызванные ими, приводят к состоянию ожидания до фиксации или отката транзакции. Никто, кроме самой транзакции, не видит вносимых транзакцией изменений до ее фиксации. Это правило имеет место независимо от того, сколько инструкций DML включает транзакция. Для управления транзакциями используются инструкции:

- *COMMIT* – подтвердить изменения, сделанные текущей транзакцией;
- *ROLLBACK* – отменить изменения, сделанные текущей транзакцией, и восстановить исходную ситуацию.

Фиксация транзакции может происходить неявно, без запроса *COMMIT*. Неявно фиксируют текущую транзакцию все инструкции DDL (*CREATE*, *ALTER*, *DROP* и др.). СУБД Oracle рассматривает их как транзакции с одной инструкцией и фиксирует немедленно. Таким образом, если последствия манипулирования с данными можно отменить инструкцией *ROLLBACK*, то инструкции DDL отменить невозможно.

Внимание! Инструкция *DROP* удаляет таблицу. Инструкция *DELETE* очищает таблицу. Инструкция *TRUNCATE* эффективно удаляет все строки в таблице, но выполнить откат для отмены *TRUNCATE* нельзя. Поэтому *TRUNCATE* считается инструкцией DDL.

Единственной *инструкцией поиска данных* в базе данных является *SELECT*. Она действует на уровне таблиц и в результате всегда создает таблицу. Если запрос возвращает одну строку или не возвращает строки, то результатом остается таблица (с одной строкой или пустая).

Инструкции DCL позволяют организовать защиту данных и ограничить доступ к данным. Управление доступом к базе данных происходит через авторизацию пользователя путем предоставления пользователям базы данных логина и пароля, а также некоторых привилегий на уровне базы данных. Наиболее важными инструкциями здесь являются:

- *CREATE USER* – определение новых пользователей базы данных;
- *ALTER USER* – изменение свойств (привилегий и паролей) существующих пользователей базы данных;
- *DROP USER* – удаление пользователя из базы данных.

Более точный доступ к данным можно реализовать, предоставив пользователям определенные привилегии:

- *системные привилегии* дают права на выполнение действий, не связанных с конкретными объектами. Например, привилегия *CREATE SESSION* позволяет

начать сеанс работы с базой данных. Oracle поддерживает около 190 различных системных привилегий;

- *объектные привилегии* дают право доступа к конкретному объекту базы данных (например, таблице) определенным способом. В табл. 3 перечислены наиболее важные объектные привилегии Oracle.

Таблица 3

Объектная привилегия	Разрешенное действие
ALTER	Изменение структуры таблицы (с помощью *ALTER TABLE*)
DELETE	Удаление строк
EXECUTE	Выполнение хранимых функций или процедур
FLASHBACK	Возврат назад во времени (с *FLASHBACK TABLE*)
INDEX	Создание индексов в таблице
INSERT	Вставка новых строк
REFERENCES	Создание ограничений внешнего ключа для таблицы
SELECT	Запрос к таблице или представлению
UPDATE	Изменение значений столбцов существующих строк

СУБД Oracle позволяет группировать привилегии в *роли*. Роли делают управление пользователями намного более простым и гибким. Роли имеют несколько полезных и мощных свойств:

- они динамичны, изменения роли автоматически затрагивают всех пользователей, которым была предоставлена эта роль;
- роли могут быть включены или отключены во время сеанса;
- можно защитить роли с помощью пароля, в этом случае только пользователи, знающие пароль роли, смогут включить роль;
- важнейшим преимуществом ролей является их управляемость.

Для администрирования привилегий и ролей используют инструкции DCL:

- *GRANT* – предоставить пользователям или ролям определенные привилегии или роли;
- *REVOKE* – отменить определенные привилегии или роли у пользователей или ролей.

Кроме двух стандартных инструкций *GRANT* и *REVOKE*, Oracle поддерживает несколько дополнительных инструкций, касающихся безопасности и доступа к данным, например, влияющих на блокировку СУБД, реализующих аудит и настраивающих более детальную авторизацию пользователя.

Внимание! Создание и удаление пользователей, предоставление и отзыв системных привилегий входит в круг обязанностей администраторов базы данных.

3.2. Основные понятия и терминология Oracle SQL

Рассмотрим следующие понятия Oracle SQL:

- константы (литералы);
- переменные;
- операторы, операнды, условия и выражения;
- функции;
- имена объектов базы данных;
- комментарии;
- зарезервированные слова.

Константы (литералы). Могут быть числовыми (числа) и алфавитно-цифровыми (строки). Алфавитно-цифровые константы указываются в кавычках (апострофах). Они чувствительны к регистру, хотя в целом язык SQL к регистру нечувствителен. Числа могут участвовать в арифметических и логических операциях. Единственная операция, разрешенная для строк, – конкатенация.

При записи чисел можно явно указать, что SQL должен интерпретировать числовые значения как числа с плавающей запятой, добавив суффиксы *f* или *d* для указания одиночной или двойной точности. Правильная интерпретация десятичных дробей и разделителей групп в числах зависит от значения параметра сеанса (NLS_NUMERIC_CHARACTERS).

Даты и временные интервалы обычно представляются в виде алфавитно-цифровых констант, но необходимо «помочь» СУБД правильно интерпретировать строки как константы даты или временного интервала. Возможны три варианта задания констант даты и времени:

- указать их как алфавитно-цифровые константы (строки) и положиться на неявную интерпретацию и преобразование Oracle. Это опасно, если фактический параметр формата для этого сеанса отличается от формата строки;
- указать их как алфавитно-цифровые константы (строки) и использовать функцию преобразования *CAST* или *TO_DATE*, чтобы явно указать, как должны интерпретироваться строки;
- указать их как алфавитно-цифровые константы (строки), используя префиксы *DATE*, *TIMESTAMP* или *INTERVAL*. При использовании *INTERVAL* также требуется суффикс для указания измерения, например *DAY*, *MONTH* или *YEAR*.

Примеры числовых констант: *27 1.53 3.124F 456.*

Примеры алфавитно-цифровых констант: *'DONald' '456.'*

Примеры констант даты и времени*: DATE '2004-02-09'.*

TIMESTAMP '2018-09-06 11.42.59.00000' INTERVAL '2' SECOND.

INTERVAL '1-3' YEAR TO MONTH.

Переменные. Могут принимать различные значения или вообще не иметь значения. Переменная всегда имеет имя, на которое можно ссылаться. Oracle SQL поддерживает два типа переменных:

- *переменные с именами столбцов* – имя столбца не изменяется, но его значение обычно меняется от строки к строке при сканировании таблицы;
- *системные переменные (псевдостолбцы)* – не имеют ничего общего с таблицами, но играют важную роль в SQL (табл. 4).

Таблица 4

Переменная	Описание
SYSDATE	Текущая системная дата в базе данных
CURRENT_DATE	Текущая дата на стороне клиентского приложения
SYSTIMESTAMP	Дата и точное время системы с указанием часового пояса
LOCALTIMESTAMP	Дата и точное время системы с указанием часового пояса на стороне клиентского приложения
USER	Имя, используемое для подключения к базе данных

Разница между датами (и временем) на стороне базы данных и клиентского приложения актуальна при удаленном подключении к базе данных.

Операторы, операнды, условия и выражения. Операторы Oracle SQL разделены на четыре категории по типу данных операндов:

- арифметические операторы;
- алфавитно-цифровой оператор;
- операторы сравнения;
- логические операторы.

Арифметические операторы. Язык Oracle SQL поддерживает четыре арифметических оператора: сложение (+), вычитание (–), умножение (*), деление (/). Операторы применяют к числовым значениям, но есть исключения:

- вычитание для двух значений *DATE* даст разницу между этими датами, выраженную в днях;
- сложение значений *DATE* и *INTERVAL* приведет к другой дате;
- при сложении значений *DATE* и *NUMBER* число будет интерпретироваться как интервал, выраженный в днях.

Алфавитно-цифровой оператор. Oracle SQL предлагает только один алфавитно-цифровой оператор, позволяющий объединить строковые выражения, – оператор конкатенации (||). В свою очередь, это компенсируется большим числом алфавитно-цифровых функций.

Операторы сравнения. Позволяют задавать условия. В Oracle SQL доступны следующие операторы сравнения: меньше чем (<), больше чем (>), равно (=), меньше или равно (<=), больше или равно (>=), не равно (<> или !=).

Выражения с операторами сравнения также называются предикатами или логическими выражениями. Эти выражения имеют результатом *TRUE* или *FALSE*, иногда – *UNKNOWN*, например, при наличии строк с недостающей информацией.

Логические операторы. В Oracle SQL имеются три логических (булевых) оператора, операндами которых являются условия: *AND* – логическое И, *OR* – логическое ИЛИ, *NOT* – логическое отрицание.

Выражения. Правильно построенные строки, содержащие переменные, константы, операторы или функции. Как и константы, всегда имеют определенный тип данных:

- числовые выражения, например: *2+6 543* (константа – самое простое выражение);
- алфавитно-цифровые выражения, например: *NAME || ',' || SURNAME*;
- логические выражения, например: *12*SAL > 50000 AND COM >= 2000*;
- выражения типа даты и времени, например: *BDATE+INTERVAL '7' YEAR*.

При записи сложных выражений необходимо учитывать приоритет выполнения оператора: арифметические операторы имеют приоритет перед операторами сравнения, а операторы сравнения имеют приоритет перед логическими операторами. Однако настоятельно рекомендуется в сложных выражениях для задания порядка действий использовать скобки.

Функции. Oracle расширил возможности стандарта SQL, касающиеся функций. Функции SQL имеют имя, за которым в скобках следует один или несколько аргументов в виде списка, разделенного запятыми.

В Oracle SQL имеется шесть категорий функций, выделенных по типу их операндов:

- числовые функции (например, *ABS( ), SQRT( ), POWER( )* и др.);
- алфавитно-цифровые функции (например, *CONCAT( ), LOWER( )* и др.);
- групповые функции (например, *AVG( ), COUNT( ), MAX( ), MIN( )* и др.);
- функции даты и времени (например, *LAST_DAY( )* и др.);
- функции преобразования (например, *TO_CHAR( ), TO_NUMBER( )* и др.);
- прочие функции (например, *NVL( ), COALESCE( ), DECODE( )* и др.).

Oracle позволяет также создавать собственные функции SQL с помощью языков PL/SQL или Java.

Имена объектов базы данных. Все объекты базы данных должны иметь имена. При этом настоятельно рекомендуется ограничиться использованием букв A-Z, цифр от 0 до 9 и при необходимости символа подчеркивания.

Внимание! В Oracle имена объектов не учитывают регистр – внутренне все имена объектов базы данных преобразуются в верхний регистр независимо от того, как они вводятся.

Имена объектов базы данных могут использовать цифры, но начинаются всегда с буквы. Максимальная длина имен – 30 символов.

Имена объектов базы данных должны быть уникальны в пределах своего пространства имен. Разные пользователи базы данных могут использовать одни и те же имена для своих объектов, потому что комбинация *имя владельца / имя объекта* позволяет однозначно определить объект в базе данных.

При желании можно создавать имена объектов базы данных, используя любые символы, а также учитывая регистр. Такие имена заключаются в двойные кавычки. Использовать этот прием не рекомендуется, поскольку имена снова нужно будет заключать в двойные кавычки в каждой интерактивной инструкции SQL, выполняемой для этих объектов. Тем не менее этот прием используется в ряде приложений (утилит) для предотвращения конфликтов с зарезервированными словами, включая возможные зарезервированные слова будущих версий СУБД.

Комментарии. Комментарии к инструкциям SQL уточняют их цели или повышают обслуживаемость. Иначе говоря, пользователь может добавить свой текст, который формально не относится к SQL-выражениям и должен быть проигнорирован СУБД Oracle. Комментарии можно добавить двумя способами: между /* и */ или после двух последовательных знаков минус. Комментарии после двух знаков минус неявно заканчиваются символом новой строки; комментарии между /* и */ могут охватывать несколько строк. Пример:

/ this text will be considered a comment,*
*so the Oracle DBMS will ignore it ... */*
-- and this text too, until the end of this line.

Комментарии также можно добавить в объекты базы данных с помощью инструкции *COMMENT*.

Зарезервированные слова. Как и любой язык, SQL содержит список зарезервированных слов. Их нельзя использовать, например, как имена объектов базы данных. Если же по какой-то причине необходимо использовать зарезервированное слово в качестве имени объекта, то его следует заключить в двойные кавычки, как описано ранее в пункте «Имена объектов базы данных».

Некоторые примеры зарезервированных слов: *AND*, *CREATE*, *DROP*, *FROM*, *GRANT*, *HAVING*, *INDEX*, *INSERT*, *MODIFY*, *NOT*, *NULL*, *NUMBER*, *OR*, *ORDER*, *RENAME*, *REVOKE*, *SELECT*, *SYNONYM*, *SYSDATE*, *TABLE*, *UPDATE*, *USER*, *VALUES*, *VIEW* и *WHERE*.

Словарь данных Oracle содержит представление *V$RESERVED_WORDS*, которое можно использовать для проверки имен объектов.

Таблица *Dual*. Часто необходимо выполнять запросы, не относящиеся к конкретному объекту схемы (таблице или представлению). В отличие от MS SQL Server в Oracle нельзя сделать запрос «из ниоткуда» (рис. 19).

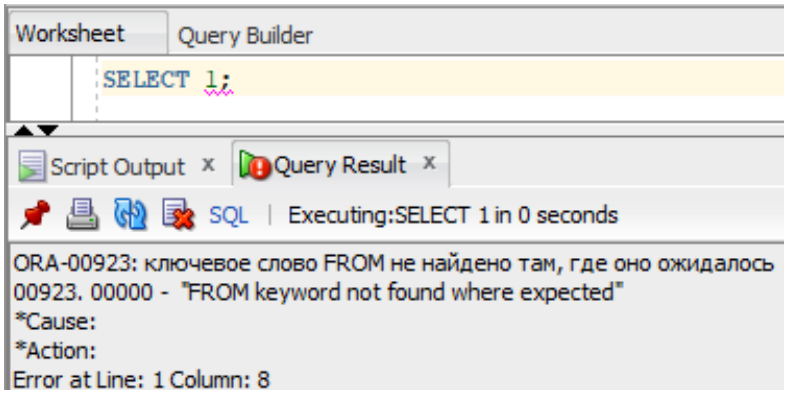


Рис. 19

Oracle предоставляет для этой цели стандартную фиктивную таблицу с именем *DUAL*, содержащую одну строку и один столбец. Начиная с версии 10G, Oracle обрабатывает использование *DUAL* как вызов функции, которая просто оценивает выражение, используемое в списке столбцов.

Ниже показаны два примера использования таблицы *DUAL*, содержащей одну строку (рис. 20).

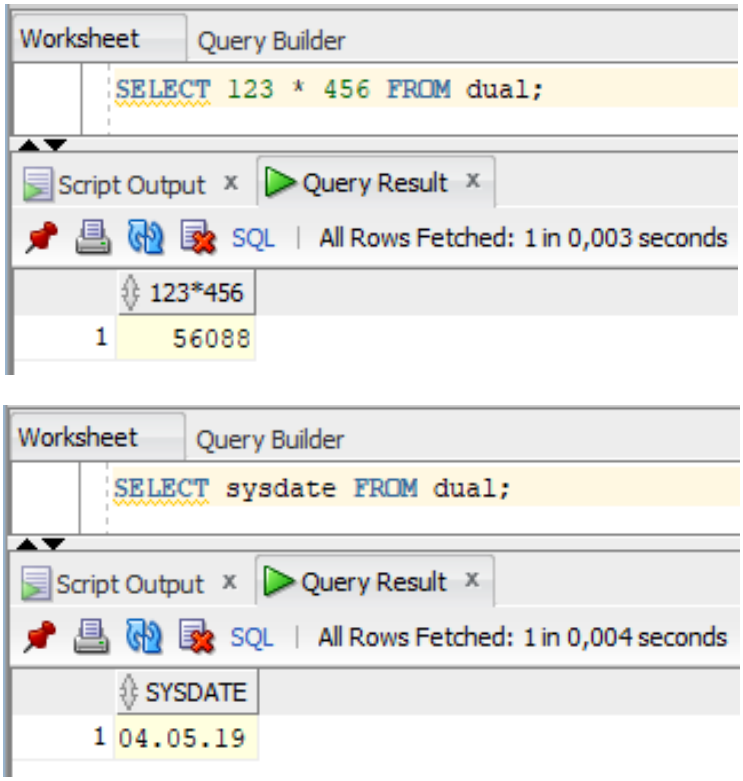


Рис. 20

3.3. Типы данных Oracle SQL

Oracle поддерживает множество стандартных типов данных. Некоторые типы выглядят похожими или даже являются синонимами. Такие типы данных поддерживаются для совместимости Oracle с другими СУБД или со стандартом ANSI/ISO языка SQL (например, *INT* и *INTEGER* являются синонимами *NUMBER(38)*). В табл. 5 представлен обзор наиболее важных типов данных Oracle.

Таблица 5

Тип данных	Описание
CHAR [(n)]	Символьная строка фиксированной длины *n* (по умолчанию 1), использующая набор символов базы данных
VARCHAR | VARCHAR2(n)	Строка переменной длины (максимум *n* символов), использующая набор символов базы данных
NCHAR[(n)]	Символьная строка фиксированной длины *n* (по умолчанию 1), использующая национальный набор символов
NVARCHAR | NVARCHAR2(n)	Строка переменной длины (максимум *n* символов), использующая национальный набор символов
DATE	Дата (между 4712 до н.э. и 9999 н.э.)
TIMESTAMP	Временная отметка с информацией о часовом поясе или без нее
INTERVAL	Интервал дата/время
BLOB	Неструктурированные двоичные данные (Binary Large Object)
CLOB	Большие тексты (Character Large Object), использующие набор символов базы данных
NCLOB	Большие тексты (Character Large Object), использующие национальный набор символов
RAW(n)	Двоичные данные; максимум *n* байт
NUMBER	Хранит любое число, максимальная точность и масштаб 38 знаков
NUMBER(n)	Целое, максимум *n* знаков
NUMBER(n, m)	Всего *n* знаков, максимум *m* знаков справа от запятой
BINARY_FLOAT	32-разрядное число с плавающей запятой
BINARY_DOUBLE	64-разрядное число с плавающей запятой

Чаще всего в столбцах реляционных таблиц размещаются числовые данные, текст и данные, связанные с временем. Соответственно, наиболее важными типами данных Oracle являются *NUMBER, VARCHAR* или *VARCHAR2* и *DATE*.

Тип данных *NUMBER*. Хранит положительные и отрицательные числа, определяет максимальную точность (*n*) и масштаб (*m*). Точность числового типа данных – количество значимых цифр, которые могут быть использованы для его записи. Описание столбца как *NUMBER* в Oracle позволяет использовать до 38 значащих цифр. Примеры использования типа данных *NUMBER* приведены в табл. 6.

Таблица 6

Пример	Описание
NUMBER (4)	Целое число с максимальной длиной в четыре цифры
NUMBER (6, 2)	Число с максимальной точностью до шести цифр, не более двух цифр после запятой
NUMBER (7, -3)	Кратное тысяче и длиной не более семи цифр
NUMBER	Идентично *NUMBER (38, *)*
NUMBER (, 5)*	Идентично *NUMBER (38, 5)*

Символьные типы данных. Используются для хранения строк. Выбор типа определяется необходимой максимальной длиной и использованием Юникода, где некоторые символы состоят из нескольких байтов. Oracle предлагает несколько символьных типов данных. Тип *CHAR* удобен для столбцов, хранящих строки фиксированной длины, тип *VARCHAR2* – для столбцов, хранящих строки переменной длины. Если размер поля типа *CHAR* не задан, то по умолчанию используется значение 1. При использовании типа *VARCHAR2* размер столбца должен указываться обязательно.

Начиная с Oracle 7, типы данных *VARCHAR* и *VARCHAR2* идентичны, но специалисты Oracle с выходом каждой новой версии заявляют, что в будущих выпусках это изменится. При использовании типа *VARCHAR* СУБД сразу же преобразует его в *VARCHAR2*. Поэтому мы рассматриваем только тип *VARCHAR2*.

Если максимальный размер типа данных *VARCHAR2 (32767)* недостаточен для какого-либо столбца, можно использовать тип данных *CLOB* (большой символьный объект). Примеры символьных типов данных приведены в табл. 7.

Таблица 7

Пример	Описание
VARCHAR2 (25)	Символьный, переменной длины до 25 символов
CHAR (4)	Символьный, фиксированной длины 4 символа
CLOB	Символьный, больше максимального размера типа *VARCHAR2*

В табл. 8 перечислены значения максимального размера для описанных выше типов данных, упомянутых до настоящего момента.

Таблица 8

Тип данных	Описание
NUMBER	38 цифр точности
CHAR	2000 байтов
VARCHAR2	4000 или 32767
CLOB	4 Гб

Замечания к табл. 8:

- фактические единицы измерения, используемые для размера типов данных *CHAR* и *VARCHAR2*, зависят от семантики (байты или символы);
- в зависимости от некоторых параметров конфигурации столбцы с типом данных *CLOB* могут содержать данные объемом гораздо больше 4 Гб;
- максимальный размер для типа *VARCHAR2* зависит от параметра базы данных *MAX_STRING_SIZE*: значение *STANDARD* разрешает до 4000 символов, значение *EXTENDED* – 32767 символов; значением по умолчанию для этого параметра является *STANDARD*.

Символьные типы данных с поддержкой национальных языков. Функции поддержки национальных языков *NLS (National Language Support)* Oracle обеспечивают хранение и обработку символьных данных на многих языках. Наборы символов для некоторых языков требуют для хранения кода каждого символа нескольких байтов. Специальные типы данных с поддержкой национальных языков (*NCHAR, NVARCHAR2* и *NCLOB*) являются аналогами рассмотренных выше типов *CHAR, VARCHAR2* и *CLOB*.

Тип данных *DATE*. Базовый тип данных, связанных со временем. По умолчанию значения даты интерпретируются и отображаются в соответствии со стандартным форматом даты, показывающим только день, месяц и последние две цифры года. Можно изменить формат даты по умолчанию для сеанса или использовать функции преобразования в командах SQL, чтобы отображать даты иначе. Oracle разрешает использование дат с 4712 года до н. э. до 9999 года н. э.

> Внимание! Даты Oracle хранятся внутри системы с гораздо большей точностью. Столбцы *DATE* содержат индикацию времени (часы, минуты и секунды), которая может вызвать проблемы при сравнении двух дат. Например, одинаковые даты могут быть разными из-за их невидимых компонентов времени.

Для хранения даты с долями секунды или с часовыми поясами используются типы *TIMESTAMP, TIMESTAMP WITH TIME ZONE* и *TIMESTAMP WITH LOCAL TIME ZONE*. Кроме того, Oracle поддерживает тип *INTERVAL* для сохранения и обработки временных промежутков.

Типы данных *CLOB, BLOB* и другие. Для поддержки мультимедийных приложений с большим объемом контента Oracle предлагает типы данных для крупных объектов *LOB (Large Object)*, позволяющие хранить неструктурированную информацию (текстовые документы, статические изображения, видео и т. д.):

- тип *CLOB (Character Large Objects)* – поле этого типа может хранить крупные символьные объекты текстовых данных (от 8 гигабайт до 128 терабайт);

- тип *BLOB (Binary Large Objects)* – поле может хранить крупные двоичные объекты (графики, видеоклипы, аудиофайлы) (от 8 терабайт до 128 терабайт);
- тип *BFILE* – поле этого типа может хранить файловые указатели на объекты *LOB*, управляемые внешними по отношению к базе данных файловыми системами. Например, столбец *BFILE* может быть списком ссылок (имен файлов) на фотографии, хранящиеся на диске CD-ROM.

Таблица может иметь несколько столбцов *CLOB, BLOB* и *BFILE*, причем в этих столбцах хранятся не сами *LOB*-объекты, а только *указатели* на них. Поле типа *LOB* обычно имеет характеристики хранения, не зависящие от характеристик содержащей его таблицы. Это облегчает выполнение требований к дискам, связанным с *LOB*-объектами. Так, можно обеспечить раздельное хранение основных табличных данных и связанных *LOB*-объектов в разных физических хранилищах.

Oracle продолжает поддерживать типы *LONG* и *LONG RAW*, использовавшиеся ранее для крупных объектов. Однако современные типы данных для *LOB*-объектов обладают многими преимуществами перед этими типами.

3.4. Словарь данных

Чтобы узнать, какие таблицы есть в базе данных, какие у них столбцы, индексируются ли эти столбцы, какие права доступа предоставляются пользователям, необходимо обратиться к *словарю данных*. Словарь данных – это внутреннее служебное администрирование Oracle. Он хранит информацию о данных (метаданные). Словарь автоматически поддерживается Oracle и всегда актуален.

Oracle хранит данные словаря так же, как хранит обычные данные – в таблицах. Преимущество этого подхода в том, что для запроса данных словаря можно использовать язык SQL таким же образом, как и при запросе обычных данных. Иначе говоря, нужно знать только имена таблиц словаря и имена их столбцов.

Доступ к словарю данных – потенциальная угроза безопасности. Поэтому Oracle предлагает системные привилегии и роли для регулирования доступа к словарю. Например, существует роль *SELECT_CATALOG_ROLE*, содержащая все привилегии доступа к словарю данных.

Несмотря на то, что информация хранится в таблицах словаря, большую часть времени вы работаете с его представлениями. Так, словарь данных Oracle содержит представление *DICTIONARY* (*DICT)*, в котором перечислены доступные пользователю представления с кратким описанием их содержимого. На рис. 21 показана сокращенная версия результатов запроса для пользователя *SYSTEM*.

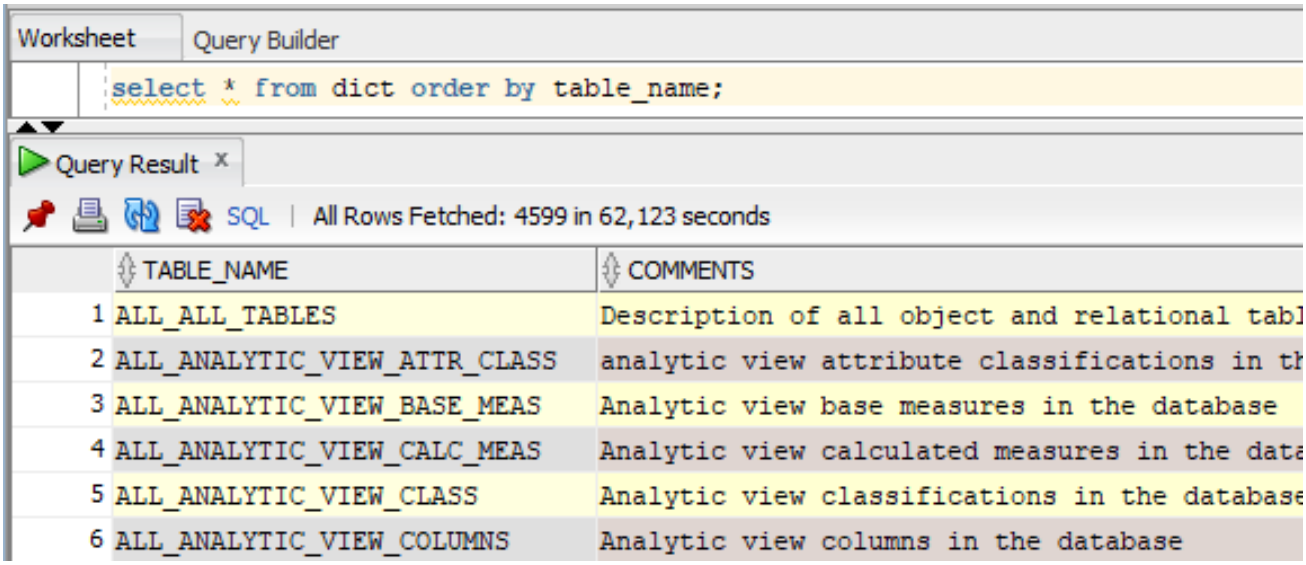


Рис. 21

NLS-параметры текущей сессии (поддержка национального языка) можно получить, выполнив запрос к представлению *NLS_SESSION_PARAMETERS* (рис. 22).

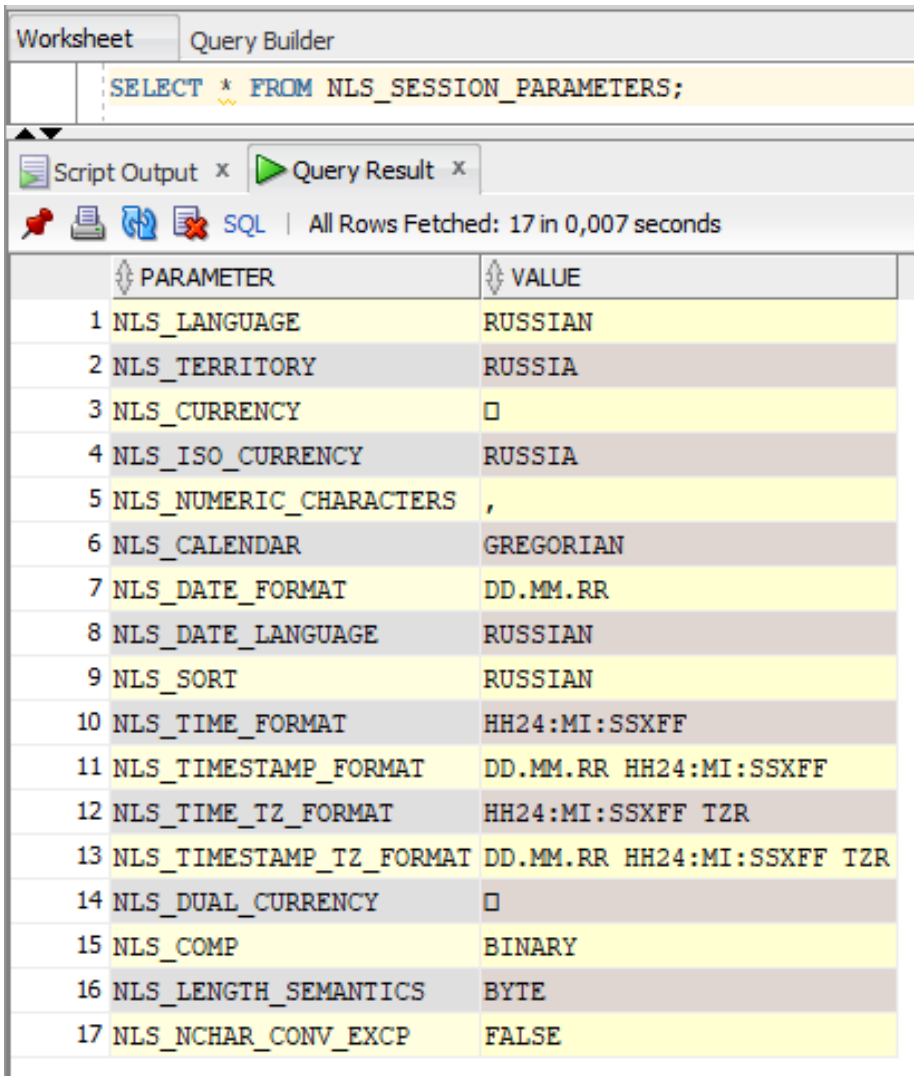


Рис. 22

Имена представлений словарей данных имеют префиксы, указывающие на существование четырех основных категорий:

- *ALL* – информация обо всех объектах, к которым можно получить доступ;
- *DBA* – вся информация в базе данных (только для администраторов);
- *CDB* – CDB-представление для каждого DBA-представления;
- *USER* – информация о собственных объектах пользователя;
- *[G]V$* – динамические представления производительности (только для администраторов баз данных).

Использование категорий подавляет информацию, не представляющую в данный момент интереса. Кроме того, в зависимости от привилегий, можно использовать лишь определенные категории представлений словарей данных.

Динамические представления производительности (с префиксом *[G]V$*) – это особая категория. Они основаны не на таблицах базы данных, а на информации из других источников, таких как структуры внутренней памяти. Они актуальны для администраторов баз данных и доступны им.

Имена представлений дают представление об их содержимом, но могут иметь большую длину. Некоторые из наиболее популярных представлений имеют альтернативные имена (синонимы) – *CAT*, *OBJ*, *IND*, *TABS* и *COLS*. Представление *CAT* особенно полезно, поскольку содержит список объектов в текущей схеме. На рис. 23 показан пример использования представления *CAT* с нашей схемой.

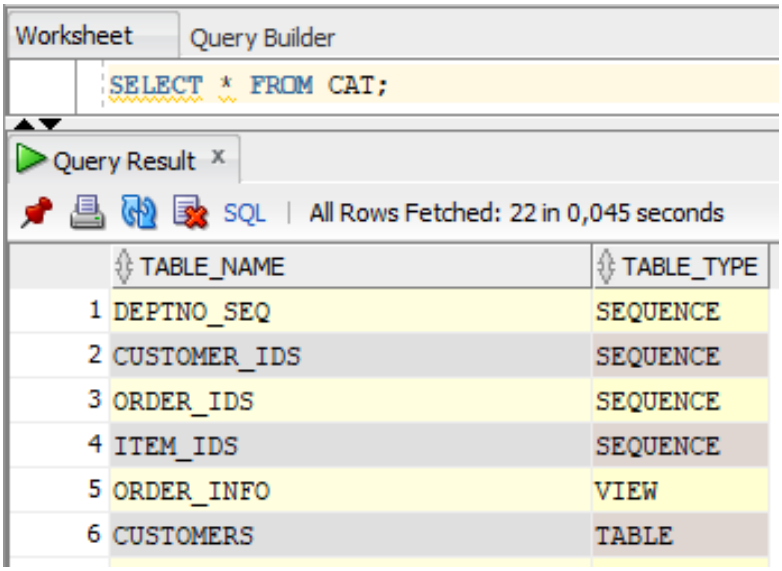


Рис. 23

Пусть необходимо запросить определенное представление словаря, но вам не известны фактические имена столбцов этого представления. Можно использовать команду SQL*Plus *DESCRIBE* как для обычных таблиц. Для этой же цели можно запросить представление словаря данных *DICT_COLUMNS* (рис. 24).

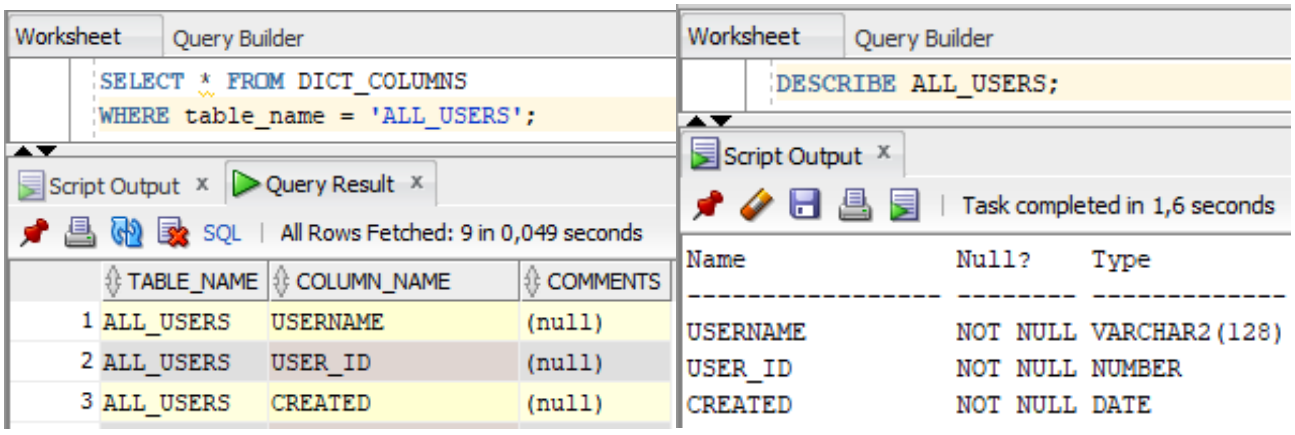


Рис. 24

Наиболее полезные представления словаря Oracle Database приведены в табл. 9.

Таблица 9

Представление	Псевдоним	Описание
DICTIONARY	*DICT*	Описание самого словаря данных
DICT_COLUMNS		Описание столбцов словаря данных
ALL_USERS		Информация обо всех пользователях базы данных
ALL_INDEXES *		Все индексы
ALL_SEQUENCES *		Все последовательности
ALL_OBJECTS *		Все объекты
ALL_SYNONYMS *		Все синонимы
ALL_TABLES *		Все таблицы
ALL_VIEWS *		Все представления
USER_INDEXES **	*IND*	Индексы
USER_SEQUENCES **		Последовательности
USER_OBJECTS **	*OBJ*	Объекты
USER_SYNONYMS **	*SYN*	Синонимы
USER_TABLES **	*TABS*	Таблицы
*USER_TAB_COLUMNS***	*COLS*	Столбцы
USER_VIEWS **		Представления
USER RECYCLEBIN		Удаленные объекты
USER_CATALOGS	*CAT*	Список объектов в текущей схеме
DUAL		Образец таблицы с одной строкой и одним столбцом

* – доступные пользователю

** – принадлежащие пользователю

4. ПРАКТИЧЕСКАЯ РАБОТА С ORACLE SQL

Этот раздел учебного пособия можно рассматривать как лабораторный практикум, состоящий из следующих частей:

- проектирование реляционной базы данных;
- создание пользователя;
- построение базовой реляционной схемы;
- доступ к базе данных с использованием запросов SQL.

Практическое занятие № 1
Проектирование базы данных

Теоретические сведения

Проектирование реляционной базы данных предполагает определение состава реляционных таблиц и логических связей между ними. Для каждого атрибута должны быть определены тип и размер данных, а также ограничения целостности. Для таблиц следует задать первичные ключи и внешние ключи.

Практическая работа

Для заданной предметной области необходимо:

- выделить сущности, атрибуты и связи предметной области;
- построить модель данных в соответствии со стандартом IE.

Пример выполнения задания

В качестве сквозного примера используется база данных интернет-магазина, торгующего литературой компьютерной тематики. Этот пример является развитием предметной области, описываемой ранее в учебных пособиях автора, в которых рассматривалась работа с разными СУБД, например см. [5]. База данных должна поддерживать следующую информацию:

- тематические каталоги, по которым сгруппированы книги;
- предлагаемые книги (название, автор, год, цена, количество на складе, дополнительно заказанное в издательстве количество);
- зарегистрированные клиенты (имя, фамилия, адрес, телефон, электронная почта, статус);
- заказы клиентов (дата заказа, дата оплаты, дата отгрузки, статус заказа);
- состав клиентского заказа (заказанные книги, количество книг).

В отличие от примера, приведенного в [5], учитываем, что в рамках одного заказа клиент может приобрести книги нескольких наименований в нескольких экземплярах.

Модель данных предметной области в стандарте IE для целевой СУБД Oracle Database XE представлена на рис. 25. Для построения диаграммы использовался бесплатный онлайн-сервис *GenMyModel*, доступный по адресу *https://www.genmymodel.com*.

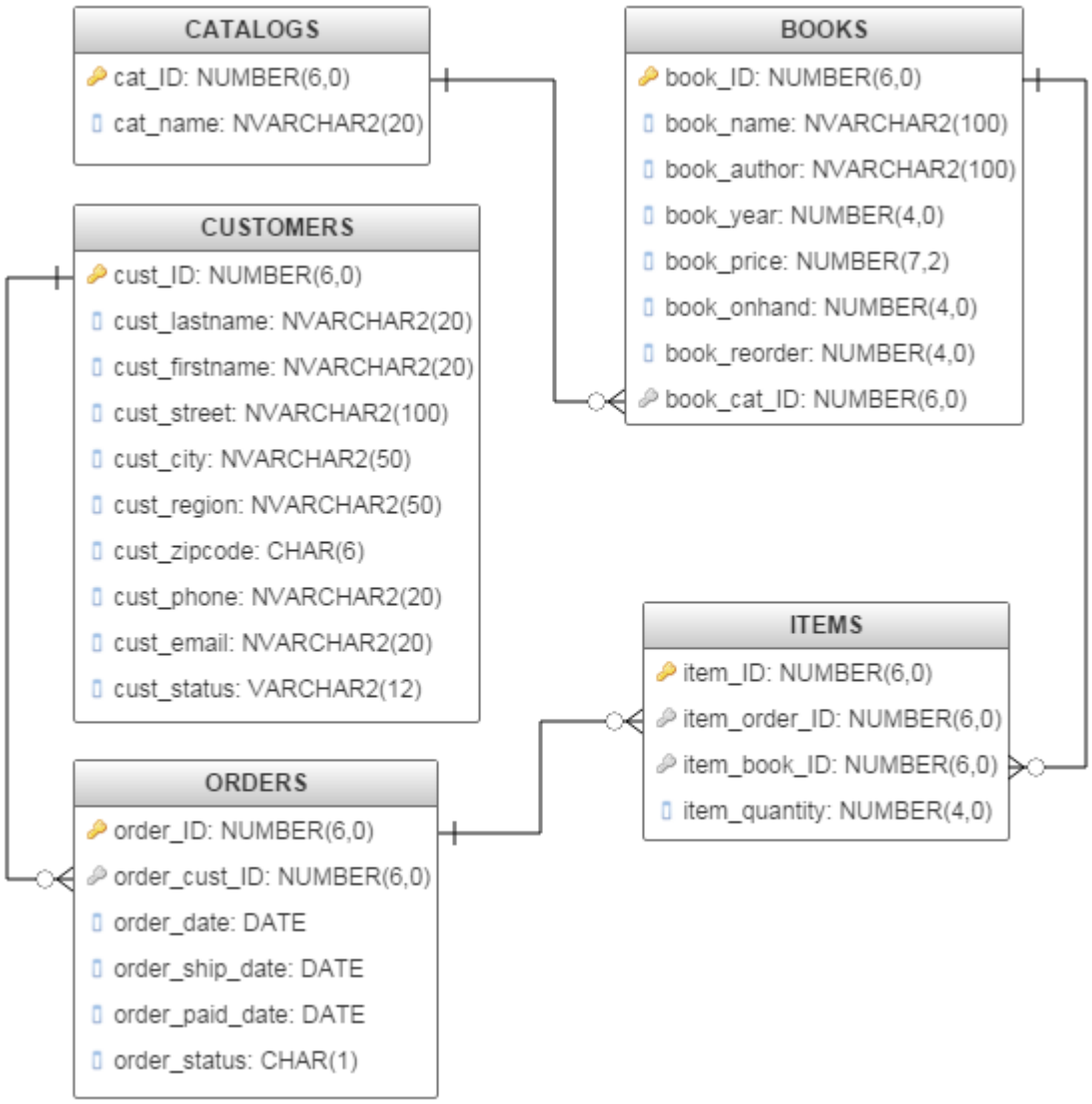


Рис. 25

База данных содержит пять таблиц:

- *Catalogs* – список тематических каталогов;
- *Books* – список предлагаемых книг;
- *Customers* – список зарегистрированных покупателей магазина;
- *Orders* – список заказов (выполненных и отмененных);
- *Items* – список элементов (строк или позиций) заказов.

Между таблицами *Catalogs* и *Books*, *Customers* и *Orders*, *Orders* и *Items*, *Books* и *Items* установлены связи «один ко многим». Один каталог может содержать много книг, но каждая книга может принадлежать одному каталогу. Один покупа-

тель может оформить много заказов, но каждый заказ принадлежит одному покупателю. Один заказ может содержать много элементов, но каждый элемент принадлежит одному заказу. Одна книга может входить во многие элементы заказа, но каждый элемент связан только с одной книгой.

Таблица *Catalogs* включает два поля:

- *cat_ID* – уникальный код каталога;
- *cat_name* – имя каталога.

Поле *cat_ID* должно иметь ограничение первичного ключа *PRIMARY KEY*. Оба поля должны иметь ограничения *NOT NULL*, поскольку неопределенные значения для них недопустимы.

Таблица *Books* включает восемь полей:

- *book_ID* – уникальный код книги;
- *book_name* – название книги;
- *book_author* – автор книги;
- *book_year* – год издания;
- *book_price* – цена книги;
- *book_onhand* – количество книг на складе;
- *book_reorder* – количество книг, заказанных дополнительно;
- *book _cat_ID* – код каталога из таблицы *Catalogs*.

Поле *book_ID* должно иметь ограничение первичного ключа *PRIMARY KEY*. Для всех полей, кроме *book_price*, *book_onhand* и *book_reorder*, должны быть установлены ограничения *NOT NULL*. Указанные поля могут принимать значение *NULL*, поскольку в момент доставки количество книг и их цена могут быть неизвестны.

По полю *book_cat_ID* устанавливается связь между таблицами *Catalogs* и *Books*, поэтому оно должно быть объявлено как внешний ключ *(FK)* с правилом каскадного удаления на уровне базы данных (каскадное обновление на уровне базы данных в Oracle Database отсутствует). Удаление каталога в таблице *Catalogs* приведет к автоматическому удалению всех записей в таблице *Books*, соответствующих этому каталогу.

Таблица *Customers* включает десять полей:

- *cust_ID* – уникальный код покупателя;
- *cust_lastname* – фамилия покупателя;
- *cust_firstname* – имя покупателя;
- *cust_street* – улица и дом покупателя;
- *cust_city* – город покупателя;

- *cust_region* – регион покупателя;
- *cust_zipcode* – почтовый код покупателя;
- *cust_phone* – телефон покупателя (если имеется);
- *cust_email* – электронная почта покупателя (если имеется);
- *cust_status* – статус покупателя.

Поле *cust_ID* должно иметь ограничение первичного ключа *PRIMARY KEY*. Для всех полей, кроме *cust_phone* и *cust_email*, должны быть установлены ограничения *NOT NULL*. Поля *cust_phone* и *cust_email* могут принимать значение *NULL*. Для этих же полей установим ограничение уникальности *UNIQUE*.

Статус покупателя может принимать одно из четырех значений:

- *active* – авторизованный покупатель, который может делать покупки через Интернет;
- *passive* – неавторизованный покупатель, который осуществил процедуру регистрации, но не подтвердил ее и пока не может делать покупки;
- *lock* – заблокированный покупатель, который не может делать покупки;
- *gold* – активный покупатель с хорошей кредитной историей, которому предоставляется скидка при покупках в интернет-магазине.

В соответствии с этими бизнес-правилами для поля *cust_status* установим ограничение проверки *CHECK*.

Таблица *Orders* включает шесть полей:

- *order_ID* – уникальный номер заказа;
- *order_cust_ID* – код покупателя из таблицы *Customers*;
- *order_date* – дата размещения заказа;
- *order_ship_date* – дата отгрузки заказа;
- *order_paid_date* – дата оплаты заказа;
- *order_status* – статус заказа.

Поле *order_ID* должно иметь ограничение первичного ключа *PRIMARY KEY*. Для полей *order_ship_date* и *order_paid_date* допустимы неопределенные значения *NULL*, поскольку на момент размещения заказа эти сведения еще не известны.

Статус заказа может принимать одно из трех значений:

- *filled (F)* – выполненный (отгруженный и оплаченный) заказ;
- *backordered (B)* – отмененный покупателем заказ;
- *other (O)* – иной статус (например, размещенный, но неоплаченный).

В соответствии с этими бизнес-правилами для поля *order_status* установим ограничение проверки *CHECK*.

Таблица *Orders* связана с таблицей *Customers* (по полю *order_cust_ID*). Это поле следует объявить как внешний ключ (*FK*) с правилом каскадного удаления. Удаление покупателя в таблице *Customers* приведет к автоматическому удалению всех записей в таблице *Orders*, связанных с этим покупателем.

Таблица *Items* включает четыре поля:

- *item_ID* – уникальный номер элемента (позиции) заказа;
- *item_order_ID* – уникальный номер заказа;
- *item_book_ID* – уникальный код книги в позиции заказа;
- *item_quantity* – количество книг в позиции заказа.

Поле *item_ID* должно иметь ограничение первичного ключа *PRIMARY KEY*.

Таблица *Items* связана с таблицей *Orders* по полю *item_order_ID*. Это поле следует объявить как внешний ключ (*FK*) с правилом каскадного удаления. Удаление заказа в таблице *Orders* приведет к автоматическому удалению всех записей в таблице *Items*, связанных с этим заказом.

Таблица *Items* связана с таблицей *Books* по полю *item_book_ID*. Это поле также следует объявить как внешний ключ (*FK*) с правилом каскадного удаления. Удаление книги в таблице *Books* приведет к автоматическому удалению всех записей в таблице *Items*, связанных с этой книгой.

Практическое занятие № 2
Создание схемы (пользователя) базы данных

Теоретические сведения

Любое приложение базы данных Oracle строится на множестве связанных между собой объектов. Такими объектами базы данных являются:

- схемы;
- таблицы;
- ограничения целостности данных;
- представления;
- последовательности;
- синонимы;
- индексы.

Oracle логически организует связанные объекты в *схему базы данных*. Схема включает все таблицы, представления, последовательности и другие объекты, необходимые для работы приложения. Схемы не касаются физического хранения объектов базы данных. Для физической организации хранения объектов Oracle использует физические структуры хранения.

Использование схем обеспечивает ряд преимуществ. Пусть в базе данных имеются две схемы *S1* и *S2* и в каждой схеме есть таблица *T1*. Обе таблицы можно уникально идентифицировать полными именами: *S1.T1* и *S2.T1*.

В Oracle концепция схемы базы данных связана с понятием *пользователя базы данных*. Схема имеет отношение «один-к-одному» с пользовательской учетной записью, при этом пользователь и соответствующая схема имеют одно и то же имя. В результате различие между пользователями и схемами размыто. Основные различия между пользователем и схемой базы данных:

- пользователь базы данных имеет пароль и определенные привилегии, позволяющие просматривать или манипулировать данными;
- схема базы данных – это логическая коллекция объектов базы данных, которые обычно принадлежат пользователю с таким же именем.

При удалении пользователя сначала следует удалить все объекты его схемы или использовать опцию операции удаления *CASCADE*, которая обеспечивает одновременное удаление пользователя и всех объектов его схемы.

Изменения в структуре пользователей. Как было отмечено выше, в облачных СУБД Oracle, имеющих контейнерную базу данных CDB и подключаемые базы данных PDB, могут быть созданы общие пользователи в CDB и локальные пользователи в PDB. В связи с этими нововведениями при управлении пользователями можно столкнуться со следующими ошибками:

- *ORA-65049: creation of local user or role is not allowed in CDB$ROOT* (создание локальных пользователей и ролей в *CDB$ROOT* запрещено). При подключении к контейнерной базе данных мы попадаем в *CDB$ROOT* – специальный объект, находящийся в CDB. Здесь возможно создание только общих пользователей, поэтому попытка создания обычного локального пользователя в CDB приведет к ошибке;
- *ORA-65096: invalid common user or role name* (недопустимое имя пользователя или имя роли). При создании общего пользователя в CDB может появиться ошибка, связанная с именем создаваемого пользователя, поскольку все имена общих пользователей должны начинаться с приставки C## или c##;
- *ORA-65066: the specified changes must apply to all containers* (указанные изменения необходимо применить ко всем контейнерам). Нельзя изменить параметры общего пользователя для определенной PDB. Кроме того, изменения можно применять, только находясь в CDB.

Находясь в PDB, можно получить список всех общих, а также локальных пользователей, относящихся к данной PDB. Из CDB можно получить только список общих пользователей. Это важно при работе с текущей версией приложения SQL Developer.

Атрибуты пользователя. Учетная запись пользователя Oracle содержит имя и атрибуты пользователя, которые включают:

- метод аутентификации;
- статус (состояние) – закрыт или открыт;
- привилегии и роли;
- установки использования пространства базы данных, управляющие ресурсами, к которым обращается пользователь (табличное пространство по умолчанию для объектов базы данных и временное табличное пространство по умолчанию для обработки запросов).

Имя пользователя базы данных должно отличаться от других имен пользователей и названий ролей, оно может иметь длину от 1 до 30 символов и не быть чувствительно к регистру. Должно начинаться с буквы, может включать цифры, символы подчеркивания, доллара ($), фунта (#). Не может содержать апострофы и кавычки. Не может быть зарезервированным словом Oracle.

Для пользователя базы данных должен быть определен *тип аутентификации*. При попытке пользователя подключиться к базе данных аутентификация гарантирует, что он тот, за кого себя выдает. Oracle поддерживает несколько механизмов аутентификации:

- аутентификация Oracle (аутентификация по паролю);
- аутентификация операционной системы;
- сетевая аутентификация.

Административный интерфейс домашней страницы Oracle XE поддерживает только аутентификацию Oracle. При запуске приложения (SQL*Plus, SQL Developer, домашней страницы базы данных) у пользователя запрашивается имя и пароль. После этого Oracle аутентифицирует запрос на основе данных учетной записи, хранящейся в базе данных.

Аутентификация по паролю предполагает определенную политику задания паролей. Сильный пароль обладает следующими характеристиками:

- содержит прописные и строчные буквы;
- имеет длину не менее восьми символов;
- содержит цифры, знаки препинания и другие специальные символы;
- не содержит имя пользователя;
- не является обычным словом или именем собственным.

Правильная политика безопасности должна требовать от пользователей замену паролей не реже, чем раз в полгода.

Администратор может в любое время закрывать и открывать учетную запись пользователя, управляя доступом этого пользователя к базе данных:

- можно закрыть учетную запись пользователя, ушедшего в отпуск или временно отсутствующего на работе;

- для уволившегося сотрудника закрытие учетной записи предпочтительнее ее удаления (схема содержит объекты, которые нужно сохранить);
- обычно закрывается учетная запись пользователя, которая используется только для логической организации всех объектов базы данных приложения.

Стандартное табличное пространство пользователя. Табличное пространство – логический раздел базы данных, который организует физическое хранение данных. Для каждого пользователя устанавливается *стандартное табличное пространство* (пространство по умолчанию). Если при создании нового объекта базы данных пользователь не указывает явно табличное пространство для его хранения, то Oracle сохраняет новый объект базы в табличном пространстве пользователя по умолчанию. Если не задано иное, то табличное пространство по умолчанию соответствует установкам по умолчанию для табличных пространств в конкретной базе данных.

Временное табличное пространство пользователя. Для выполнения команд SQL может потребоваться *временное рабочее пространство*. Например, временное рабочее место для создания результирующей таблицы. В этом случае Oracle выделяет такое пространство во *временном табличном пространстве пользователя*. Если не задано иное, то временным табличным пространством пользователя является временное табличное пространство базы по умолчанию.

Заданные по умолчанию для всех пользователей временное и постоянное табличные пространства можно определить при инсталляции базы данных. Тогда при создании пользователя эти табличные пространства можно не указывать.

Однако даже если пользователю явно или по умолчанию назначено постоянное табличное пространство, он не сможет создавать объекты, пока ему не будет предоставлена *квота* в этом табличном пространстве (по умолчанию квоты пользователям не предоставляются). Конкретные квоты табличных пространств явно присваивают во время создания пользователя.

Для временного использования пространства в соответствующем временном табличном пространстве квота пользователю не требуется.

Квоты могут быть:

- определенным значением в мегабайтах или килобайтах;
- неограниченными.

Администратор может предоставить квоту пользователю тремя способами:

- разрешить пользователю использовать все доступное место в определенном табличном пространстве (опция *UNLIMITED*);
- задать конкретное число килобайт или мегабайт, которые будут доступны пользователю (без гарантий, поскольку указанное значение может оказаться больше текущего доступного места в табличном пространстве);

- дать системную привилегию *UNLIMITED TABLESPACE*, которая переопределит все отдельные квоты табличных пространств и даст пользователю неограниченную квоту для всех табличных пространств, включая *SYSTEM* и *SYSAUX* (эту привилегию нужно предоставлять с осторожностью).

Внимание! Не следует назначать квоту пользователям для табличных пространств *SYSTEM* и *SYSAUX*. Только пользователи *SYS* и *SYSTEM* должны иметь право создавать объекты в табличных областях *SYSTEM* и *SYSAUX*.

Управление привилегиями. Созданный пользователь сможет подключиться к базе данных и выполнить действия только после того, как ему будут предоставлены определенные привилегии – системные и объектные.

Системные привилегии дают пользователю широкие полномочия, позволяющие выполнять операции на уровне системы. Эти операции могут затрагивать безопасность всей системы, поэтому предоставлять системные привилегии нужно очень осторожно.

Объектные привилегии позволяют пользователю выполнять определенные операции с конкретным объектом базы данных, таким как таблица, представление или хранимая процедура.

Системные и объектные привилегии могут быть предоставлены пользователю или отозваны у него. Операции предоставления и отзыва привилегий разрешается выполнять лицу, имеющему соответствующие полномочия. Управление привилегиями подчиняется следующим правилам:

- вы можете предоставить пользователю системную привилегию, если сами имеете системную привилегию на предоставление привилегий другим пользователям (*WITH ADMIN OPTION*);
- вы можете предоставлять любые привилегии другим пользователям, если имеете системную привилегию *GRANT ANY PRIVILEGE*;
- вы можете предоставить пользователю объектную привилегию, если являетесь собственником соответствующего объекта базы данных или имеете объектную привилегию с правом предоставления привилегий другим пользователям (*WITH GRANT OPTION*);
- вы можете предоставлять любые объектные привилегии на другие объекты базы данных, если имеете системную привилегию *GRANT ANY OBJECT PRIVILEGE*.

Для предоставления пользователю системных привилегий используется инструкция *GRANT*. Эта инструкция с опцией *WITH ADMIN OPTION* предоставит

пользователю системные привилегии, которые он сможет предоставить другим пользователям уже по своему усмотрению. Обычным разработчикам приложений не следует предоставлять такую возможность.

Для отзыва системных привилегий используется инструкция *REVOKE*. Если пользователю были предоставлены системные привилегии с опцией *WITH ADMIN OPTION* и администратор хочет отозвать привилегию передавать системные привилегии другим пользователям, то действия выполняют в два этапа. Инструкцией *REVOKE* отзывают у пользователя право использовать назначенные системные привилегии. Далее вновь предоставляют этому пользователю эти же системные привилегии, но уже без опции *WITH ADMIN OPTION*.

Вопросы предоставления и отзыва объектных привилегий в данном учебном пособии не рассматриваются.

Для управления привилегиями могут использоваться *роли* – именованные группы привилегий. Администратор создает роли, назначает ролям системные и объектные привилегии, а потом присваивает роли пользователям. Oracle Database XE поставляется с несколькими предопределенными ролями:

- роль *DBA*, позволяющая выполнять административные функции;
- роль *CONNECT*, позволяющая подсоединяться к базе данных;
- роль *RESOURCE*, предназначенная для разработчика приложений.

Внимание! Использовать эти роли специалисты Oracle не рекомендуют, поскольку их поддержка в будущих версиях не планируется. Следует предоставлять только те привилегии, которые понадобятся конкретному пользователю.

Практическая работа

При выполнении работы необходимо в базе данных Oracle XE:

- получить информацию о пользователях базы данных;
- создать новую учетную запись пользователя – разработчика программного обеспечения, не обладающего административными полномочиями;
- выделить этому пользователю квоты на использование дискового пространства и предоставить пользователю необходимые системные привилегии;
- при необходимости изменить или удалить учетную запись.

Внимание! На практических занятиях предполагается выполнение действий пользователем базы данных *BOOKSHOP* с паролем *BOOKSHOP* и схемой *BOOKSHOP*, содержащей представленные выше пять таблиц (рис. 25).

Пример выполнения задания

Создание нового пользователя базы данных. При инсталляции Oracle Database 18c Express Edition создается одна контейнерная база данных (*localhost:1521*) и одна подключаемая база данных (*localhost:1521/XEPDB1*).

Выполните следующие действия:

1. Запустите SQL Developer и создайте подключение *system-XE*, используя стандартную учетную запись *SYSTEM*, имеющую административные привилегии (с паролем, заданным при инсталляции Oracle XE) (см. п. 1.8).

При подключении к контейнерной базе данных вы попадаете в *CDB$ROOT*, где возможно создание только общих пользователей (рис. 26).

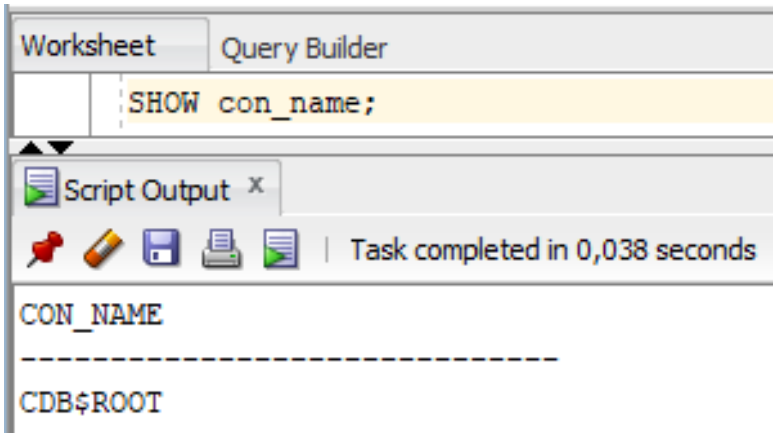


Рис. 26

2. Для создания нового локального пользователя необходимо перейти в PDB, чтобы там выполнить команду создания пользователя (рис. 27).

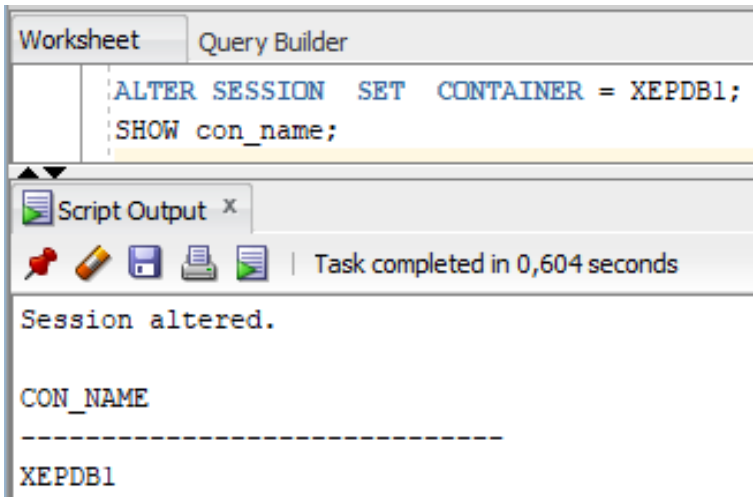


Рис. 27

3. Создайте пользователя *BOOKSHOP* с паролем *BOOKSHOP*, назначив ему стандартное табличное пространство *USERS* и предоставив неограниченную квоту в этом табличном пространстве (рис. 28).

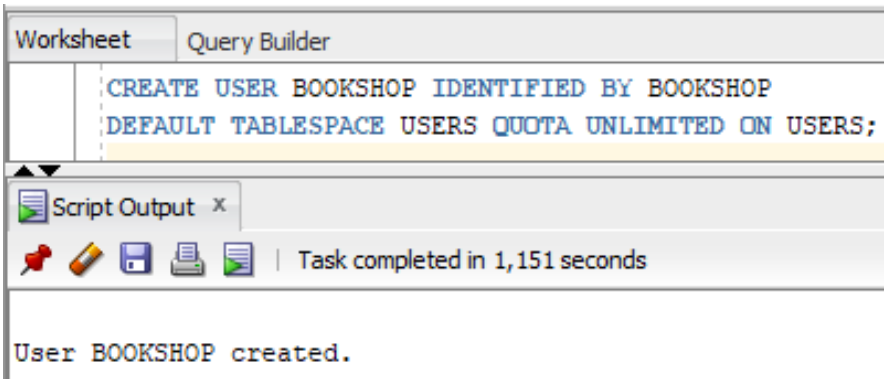


Рис. 28

4. Проверьте назначение стандартного табличного пространства и временного табличного пространства для пользователя *BOOKSHOP* (рис. 29).

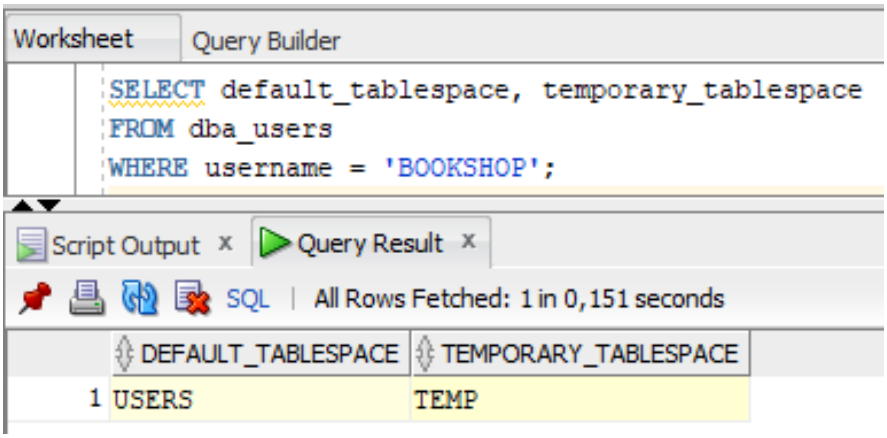


Рис. 29

5. Проверьте квоту, предоставленную пользователю *BOOKSHOP* в стандартном табличном пространстве *USERS* (рис. 30).

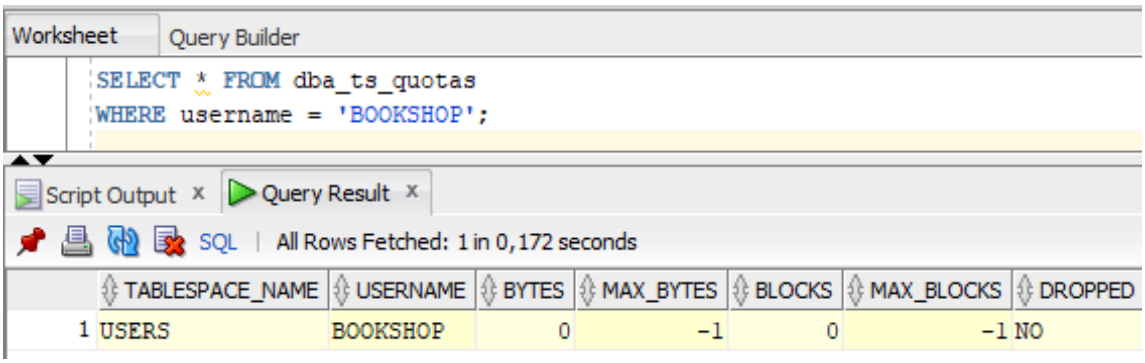


Рис. 30

Принятые обозначения: *TABLESPACE_NAME* – имя табличного пространства; *USERNAME* – имя пользователя; *BYTES* – количество занятых пользователем байтов; *MAX_BYTES* – квота пользователя в байтах или –1, если квота не ограничена; *BLOCKS* – количество занятых пользователем блоков Oracle; *MAX_BLOCKS* – квота пользователя в блоках или –1, если квота не ограничена.

6. Задайте пользователю *BOOKSHOP* необходимые системные привилегии (рис. 31).

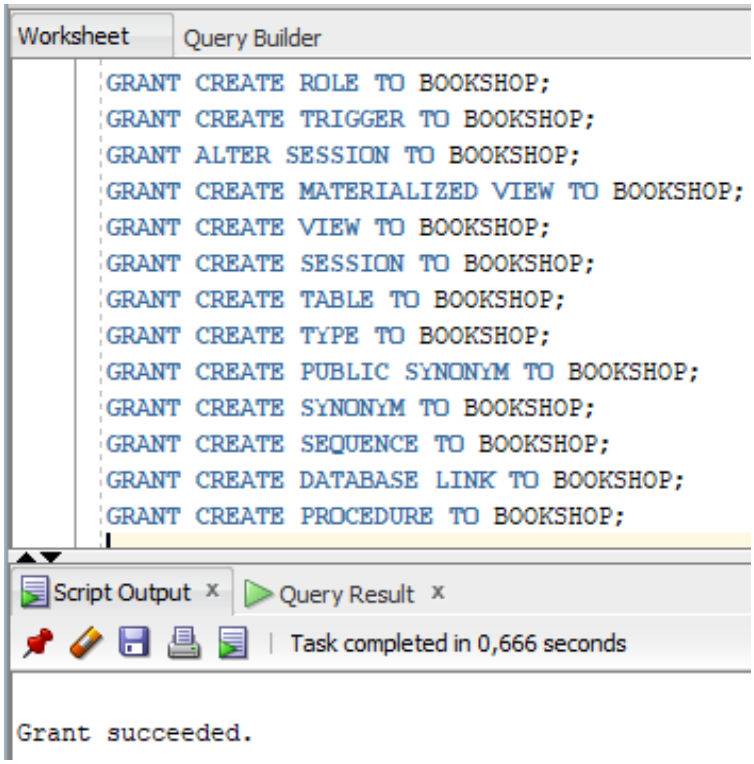
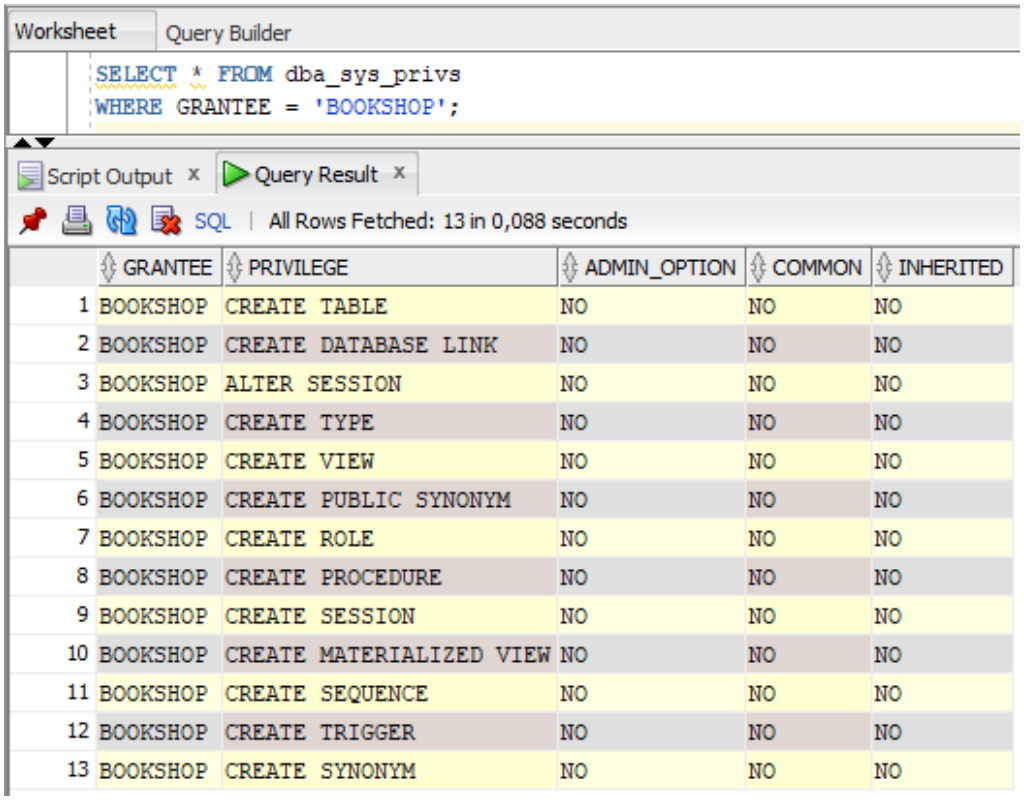


Рис. 31

7. Проверьте назначение системных привилегий пользователю *BOOKSHOP* (рис. 32).



	GRANTEE	PRIVILEGE	ADMIN_OPTION	COMMON	INHERITED
1	BOOKSHOP	CREATE TABLE	NO	NO	NO
2	BOOKSHOP	CREATE DATABASE LINK	NO	NO	NO
3	BOOKSHOP	ALTER SESSION	NO	NO	NO
4	BOOKSHOP	CREATE TYPE	NO	NO	NO
5	BOOKSHOP	CREATE VIEW	NO	NO	NO
6	BOOKSHOP	CREATE PUBLIC SYNONYM	NO	NO	NO
7	BOOKSHOP	CREATE ROLE	NO	NO	NO
8	BOOKSHOP	CREATE PROCEDURE	NO	NO	NO
9	BOOKSHOP	CREATE SESSION	NO	NO	NO
10	BOOKSHOP	CREATE MATERIALIZED VIEW	NO	NO	NO
11	BOOKSHOP	CREATE SEQUENCE	NO	NO	NO
12	BOOKSHOP	CREATE TRIGGER	NO	NO	NO
13	BOOKSHOP	CREATE SYNONYM	NO	NO	NO

Рис. 32

8. Находясь в PDB, можно получить список всех общих, а также локальных пользователей, относящихся к данной PDB (из CDB можно получить только список общих пользователей) (рис. 33).

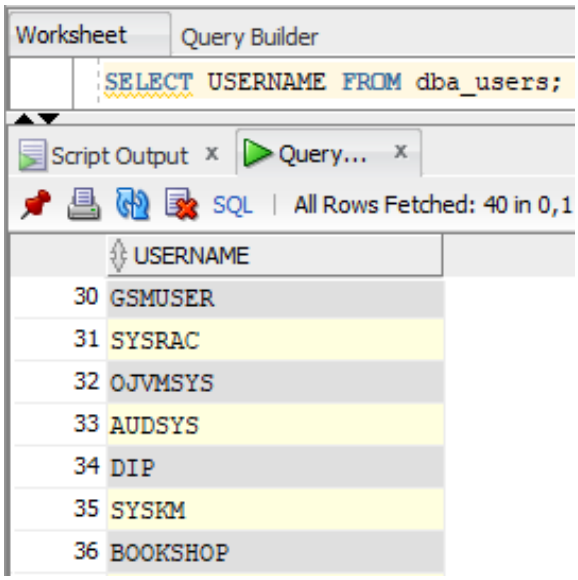


Рис. 33

9. Создайте подключение для пользователя *BOOKSHOP* (см. п. 1.8 и рис. 34). Разорвите соединение с базой данных пользователя *SYSTEM*, подключитесь к базе как пользователь *BOOKSHOP* и все дальнейшие действия выполняйте в качестве этого пользователя.

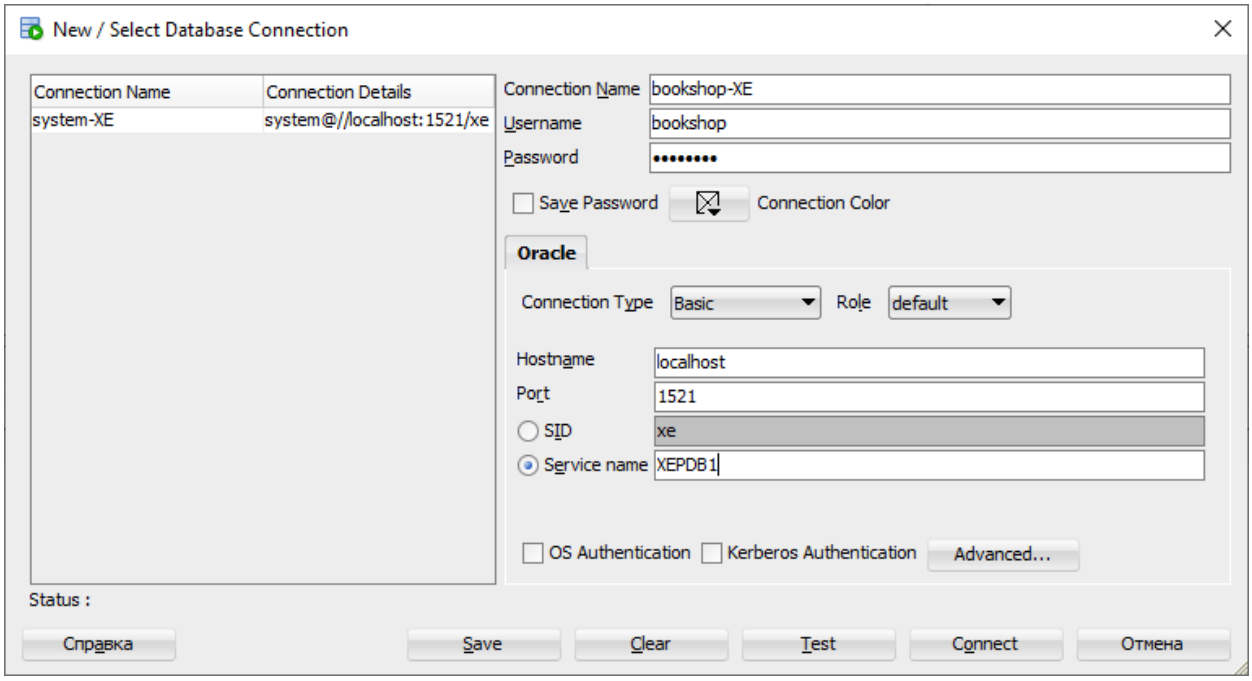


Рис. 34

Отображение информации о пользователях базы данных. В навигаторе подключений войдите как пользователь *SYSTEM* и раскройте узел *Other Users*.

Изменение учетной записи пользователя. Администратор может изменить некоторые атрибуты пользователя, кроме его имени. Чтобы изменить имя пользователя, необходимо удалить пользователя и создать заново. Для изменения учетной записи пользователя:

1. В навигаторе подключений SQL Developer раскройте подключение *SYSTEM* и узел *Other Users*.

2. Щелкните правой кнопкой мыши пользователя, учетную запись которого хотите изменить, и выберите в контекстном меню команду *Edit User*… .

3. Диалоговое окно изменения пользователя выглядит так же, как диалоговое окно создания новой учетной записи. С помощью этого окна можно:

- изменить пароль пользователя;
- установить статус пароля как пароля с истекшим сроком;
- закрыть или открыть учетную запись и т. д.

4. Подтвердите сделанные установки, щелкнув кнопку *Apply*, и закройте диалоговое окно, щелкнув кнопку *Close*.

Удаление учетной записи пользователя. Администратор может удалить учетную запись пользователя базы данных. Перед удалением пользователя следует удалить все объекты его схемы. Можно использовать каскадное удаление, при котором одновременно удаляется пользователь и объекты схемы. Для удаления пользователя и всех объектов его схемы:

1. В навигаторе подключений SQL Developer раскройте подключение *SYSTEM* и узел *Other Users*.

2. Щелкните правой кнопкой мыши пользователя, учетную запись которого хотите удалить, и выберите в контекстном меню команду *Drop User* … .

3. В диалоговом окне *Drop User* будет предложено подтвердить запрос на удаление. Если вы удаляете пользователя, которому принадлежат объекты схемы, то следует выбрать опцию каскадного удаления (*Cascade*), иначе Oracle выведет сообщение об ошибке, препятствуя удалению объектов.

4. Подтвердите удаление, щелкнув кнопку *Apply*.

Практическое занятие № 3
Создание таблиц и ограничений целостности

Теоретические сведения

Рассмотрим основные понятия, связанные с таблицами, полями, типами данных и ограничениями целостности.

Таблицы базы данных. Являются базовой структурой реляционной базы данных. *Таблица* – это организованное семейство записей (строк), которые имеют

одинаковые атрибуты (столбцы). При создании таблицы необходимо учитывать главные ее составляющие:

- поля, описывающие структуру таблицы;
- ограничения целостности, задающие приемлемые для таблицы данные.

Для создания таблицы необходимо сформировать ее структуру, указав поля (столбцы). Каждый столбец таблицы имеет определенный *тип данных*, допустимых для этого столбца. Базовые типы данных Oracle описаны выше в п. 3.3.

Значения полей по умолчанию. При объявлении поля можно задать для него значение по умолчанию. При создании заказа таким значением для поля *order_date* таблицы *Orders* может быть текущее системное время.

Внимание! Если не указано иное, то значением по умолчанию для любого поля является *NULL* (отсутствие значения).

Ограничения целостности. *Целостность данных* – фундаментальный принцип реляционной модели, гарантирующий точность и непротиворечивость информации в базе данных. Базовый уровень целостности данных устанавливает простые типы данных, которые могут храниться в столбцах (например, столбец типа *DATE* может содержать корректные значения даты и времени, но не может содержать числа). Кроме того, сама реляционная модель определяет несколько правил целостности, которые должны поддерживаться СУБД.

Целостность домена. Определяет область допустимых значений поля с позиций бизнес-правил предметной области. Кроме того, Oracle поддерживает два типа ограничений целостности, позволяющих еще больше ограничить область возможных значений поля (домен):

- ограничение на значения *NULL* в столбце;
- ограничение проверки *CHECK*, содержащее явный список допустимых значений столбца.

Целостность по существованию. Гарантирует, что каждая строка таблицы идентифицируется значением уникального ключа. Устраняется возможность дублирования записей в таблице. *Первичный ключ таблицы (Primary Key)* – столбец, который уникально идентифицирует каждую строку в таблице и не содержит значений *NULL*. Обычно таблицы создают с первичным ключом в виде произвольного числового идентификатора. Первичный ключ может быть *составным*. *Альтернативные ключи (Alternate Keys)*, также называемые уникальными ключами, представляют собой столбцы (группы столбцов), не содержащие повторяющихся значений (например, поле *cust_email* в таблице *Customers*).

Ссылочная целостность. Устанавливает связи между столбцами и таблицами в базе данных. Гарантирует, что каждое значение внешнего ключа дочерней таблицы соответствует значению первичного или альтернативного ключа родительской таблицы. Например, строка дочерней таблицы *Orders* не корректна, если ее поле *order_cust_ID* не ссылается на корректный идентификатор *cust_ID* клиента в родительской таблице *Customers*. Если родительская и дочерняя таблицы одна и та же, то связь является рекурсивной.

Чтобы гарантировать ссылочную целостность, СУБД должна контролировать операции с родительскими таблицами. Действия по ссылке описывают, что будет сделано, если приложение обновит или удалит строку родительской таблицы, имеющую зависимые строки в дочерней таблице. Реляционная модель предусматривает следующие действия по ссылке:

- *ограничение обновления/удаления*. Не позволяет обновлять родительский ключ или удалять родительскую строку, имеющую зависимые дочерние записи. Например, нельзя удалить из таблицы *Orders* заказ, имеющий связанные элементы в таблице *Items*;
- *каскадное удаление*. При удалении строки из родительской таблицы СУБД последовательно удаляет все зависимые записи дочерней таблицы. Например, если из таблицы *Orders* удаляется заказ, то автоматически будут удалены все связанные элементы в таблице *Items*;
- *каскадное обновление*. При обновлении родительского ключа СУБД выполняет каскадное обновление зависимых внешних ключей. Например, если меняется идентификатор заказа *order_ID* в таблице *Orders*, то СУБД автоматически обновляет это значение идентификатора заказа у всех связанных записей в таблице *Items* (Oracle не поддерживает каскадное обновление);
- *обновление/удаление с установкой NULL-значения*. Когда приложение обновляет или удаляет родительский ключ, всем зависимым ключам присваивается значение *NULL*;
- *обновление/удаление с установкой значения по умолчанию*. Когда приложение обновляет или удаляет родительский ключ, все зависимые ключи получают некоторое значение по умолчанию.

Внимание! По умолчанию Oracle осуществляет действия по ссылке с ограничением обновления/удаления. Oracle может также осуществлять каскадное удаление или удаление с установкой *NULL*-значений.

Внешние и внутренние ограничения. Определения ограничений можно задать в командах *CREATE TABLE* и *ALTER TABLE*. При этом существует два способа задания ограничений:

- *внешние ограничения* – ограничения рассматривают как независимые компоненты таблицы (например, в конце команды *CREATE TABLE* после всех определений столбцов);
- *встроенные ограничения* – ограничения рассматривают как часть определения столбца.

Для каждого ограничения можно указать его имя. Рекомендуется назначать информативные имена. Если этого не сделать, то Oracle создаст для ограничения свое неинформативное имя. Имена позволяют управлять ограничениями (активировать, деактивировать, снять).

Отложенные ограничения. По умолчанию Oracle проверяет все ограничения целостности сразу после того, как приложение дает SQL-команду вставки, обновления или удаления строк в таблице. Если эта команда вызывает нарушение целостности данных, то Oracle автоматически отменяет ее результат.

Oracle может отложить выполнение допускающего задержку ограничения целостности до фиксации транзакции. Во время фиксации транзакции, которая изменила данные так, что они не соответствуют ограничениям целостности, Oracle автоматически выполнит откат всей транзакции.

Определения ограничений в словаре данных. Определения ограничений хранятся в словаре данных. Наиболее важными представлениями являются *USER_CONSTRAINTS* и *USER_CONS_COLUMNS*. Ниже в листинге приведен обзор ограничений ссылочной целостности для текущего пользователя (рис. 35).

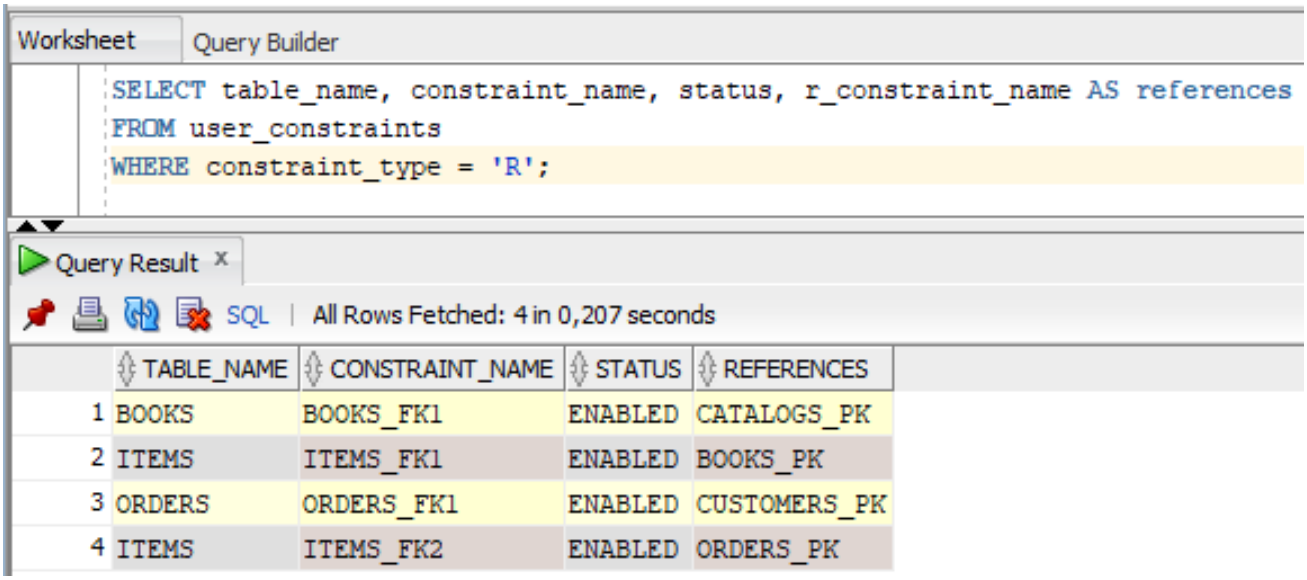


Рис. 35

Создание таблиц. Таблицы создает инструкция *CREATE TABLE*. При создании таблицы указывается ее имя, за которым следует спецификация всех столбцов таблицы. Столбцы указывают в скобках через запятую.

Дополнительное предложение *STORAGE* в инструкции *CREATE TABLE* позволяет задать спецификации физического хранения таблицы и тем самым оптимизировать физическое размещение данных на диске.

Можно создать новую таблицу по образу существующей таблицы, используя команду *CREATE TABLE ... AS SELECT (CTAS).* Новая таблица создается и заполняется одной инструкцией DDL. При использовании *CTAS* можно опустить спецификации столбцов. Но если они есть, указывать типы данных нельзя: новая таблица наследует типы данных из результатов вложенного запроса.

Ограничения могут быть заданы двумя способами:

- в качестве независимых компонентов оператора *CREATE TABLE*;
- как ограничения, встроенные в спецификацию столбца.

Создание схемы. В соответствии со стандартом ANSI/ISO Oracle поддерживает инструкцию *CREATE SCHEMA*:

CREATE SCHEMA AUTHORIZATION <имя_схемы>
[CREATE TABLE <имя_таблицы> (опции)] ...
[CREATE VIEW <имя_представления> (опции)] ...
[GRANT опции];

На самом деле инструкция *CREATE SCHEMA* схему не создает (это делает только инструкция *CREATE USER*). Оператор выполняется успешно, только если имя схемы совпадает с именем пользователя базы данных. *CREATE SCHEMA* позволяет создать схему из таблиц, представлений и разрешений с помощью одной инструкции DDL, которую Oracle рассматривает как одну транзакцию.

Oracle разрешает использовать в *CREATE SCHEMA* только стандартные инструкции *CREATE TABLE*, *CREATE VIEW* и *GRANT*, которые могут следовать в любом порядке. В определении компонента можно ссылаться на более ранние или более поздние компоненты схемы. Oracle выполняет инструкцию *CREATE SCHEMA* только в том случае, если все входящие в нее инструкции *CREATE* и *GRANT* были выполнены успешно.

Модификация таблиц. Осуществляется инструкциями *ALTER TABLE* и *RENAME*. Первая позволяет изменить структуру существующей таблицы (увеличить ширину столбца, добавить столбец, изменить ограничения). С помощью опции *ADD* можно добавить в таблицу столбцы или определения ограничений. Опция *MODIFY* позволяет изменить определения существующих столбцов (расширить столбец, разрешить или запретить неопределенные значения).

Можно физически удалить столбцы из таблиц с помощью параметра *DROP COLUMN*. Можно также сделать столбцы неиспользуемыми с помощью команды *ALTER TABLE ... SET UNUSED* и позже физически удалить их из базы данных командой *ALTER TABLE ... DROP UNUSED COLUMNS*. Параметр *RENAME COLUMN ... TO ...* позволяет изменить имя столбца.

Удаление столбцов в очень больших таблицах может потребовать много времени и ресурсов. Избежать исчерпания пространства отмены, используемого для отката, позволяет опция *CHECKPOINT* в конструкции *ALTER TABLE ... DROP UNUSED COLUMNS CHECKPOINT 250*. В этом случае изменения будут сохраняться через каждые 250 строк.

Секция управления ограничениями инструкции *ALTER TABLE* дает возможность удалять, включать и отключать ограничения. Как и *CREATE TABLE*, инструкция *ALTER TABLE* позволяет влиять на физические атрибуты хранения таблиц. Как правило, можно изменять структуру существующих таблиц, даже если эти таблицы содержат данные. Но есть исключения, например нельзя добавить столбец *NOT NULL* в непустую таблицу, если в той же команде *ALTER TABLE* сразу не указать значение по умолчанию. Инструкция *RENAME* позволяет изменить имя таблицы или представления.

Удаление таблиц. Осуществляется инструкцией *DROP TABLE*. Нельзя удалить таблицы, принадлежащие другим пользователям базы данных, не имея соответствующих системных привилегий. Инструкцию *DROP TABLE* нельзя откатить, поскольку она относится к DDL.

Начиная с версии Oracle Database 10g, используется концепция *корзины базы данных*. Удаленные таблицы и зависимые от них объекты по умолчанию попадают в корзину. Сведения о содержимом корзины можно получить с помощью представления *USER_RECYCLEBIN*. В экспериментальной схеме *PROBA*, включающей только таблицы *Catalogs* и *Books*, выполним очистку корзины (рис. 36).

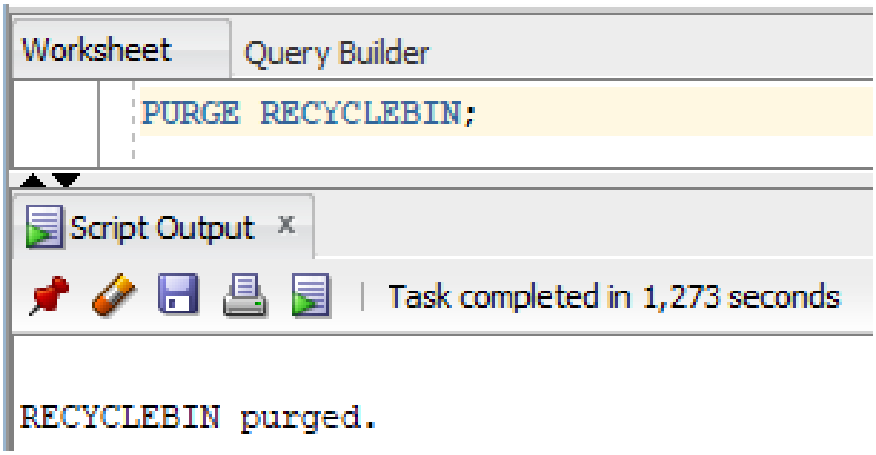


Рис. 36

Удалим таблицу *Books* и проанализируем содержимое корзины (рис. 37).

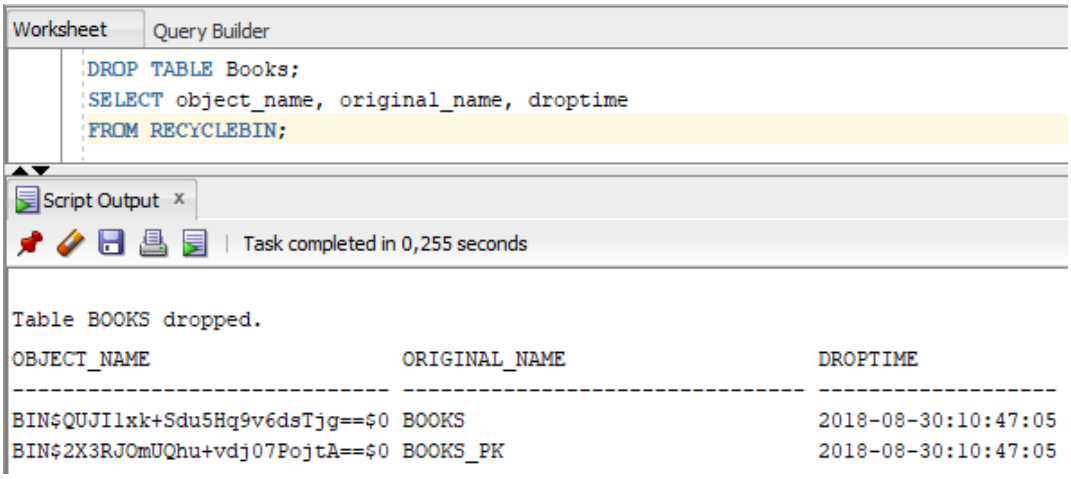


Рис. 37

В корзине объекты переименовываются, но при этом исходные имена также сохраняются. Теперь в корзине одна запись для таблицы *Books* и одна запись для индекса первичного ключа этой таблицы. Можно восстановить таблицы из корзины с помощью команды *FLASHBACK TABLE* (рис. 38).

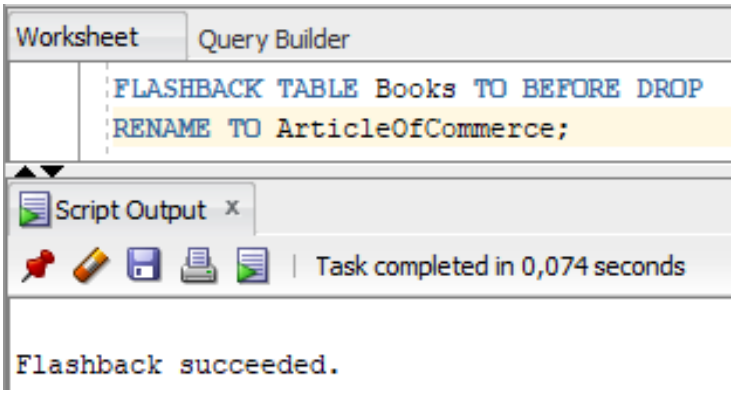


Рис. 38

Удаленная таблица *Books* восстановлена под новым именем *ArticleOfCommerce*, но имевшаяся связь по внешнему ключу с таблицей *Catalogs* утрачена.

> Внимание! Успешное выполнение инструкции *FLASHBACK TABLE* не гарантируется. Корзина может быть очищена явно администратором базы данных или неявно СУБД Oracle.

Если необходимо удалить таблицу, не помещая ее в корзину, можно использовать опцию *PURGE* команды *DROP TABLE*. Например, дать команду *DROP TABLE Books PURGE* вместо *DROP TABLE Books*. Таблица *Books* не будет перемещена в корзину, и восстановить ее уже не удастся.

При удалении таблицы удаляются и некоторые зависимые объекты базы данных (индексы, триггеры, права доступа к таблице для других пользователей). Аннулируются также представления и пакеты. Для повторного создания таблицы необходимо полностью восстановить окружение удаленной таблицы.

Если другие таблицы содержат ограничения внешнего ключа, ссылающиеся на удаляемую таблицу, то при выполнении команды *DROP TABLE* может появиться следующее сообщение об ошибке:

```
ORA-02449: unique/primary keys in table referenced by foreign keys
```

Это произойдет, например, при попытке удалить таблицу *Catalogs* в той же схеме *PROBA*. Проблему можно решить с помощью опции *CASCADE CONSTRAINTS*. При этом удаляются и ограничения внешнего ключа (рис. 39).

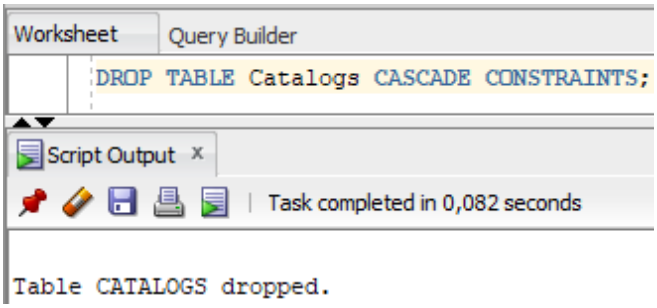


Рис. 39

Инструкция *TRUNCATE TABLE* удаляет все строки таблицы. Она аналогична инструкции *DELETE* без предложения *WHERE*, но работает с большими таблицами быстрее и потребляет меньше системных ресурсов. При использовании *TRUNCATE* не вызываются триггеры. Не применима к таблице, на которую ссылается внешний ключ другой таблицы. Выполнить *TRUNCATE TABLE Catalogs* нельзя, так как на таблицу *Catalogs* ссылается внешний ключ таблицы *Books* (рис. 40).

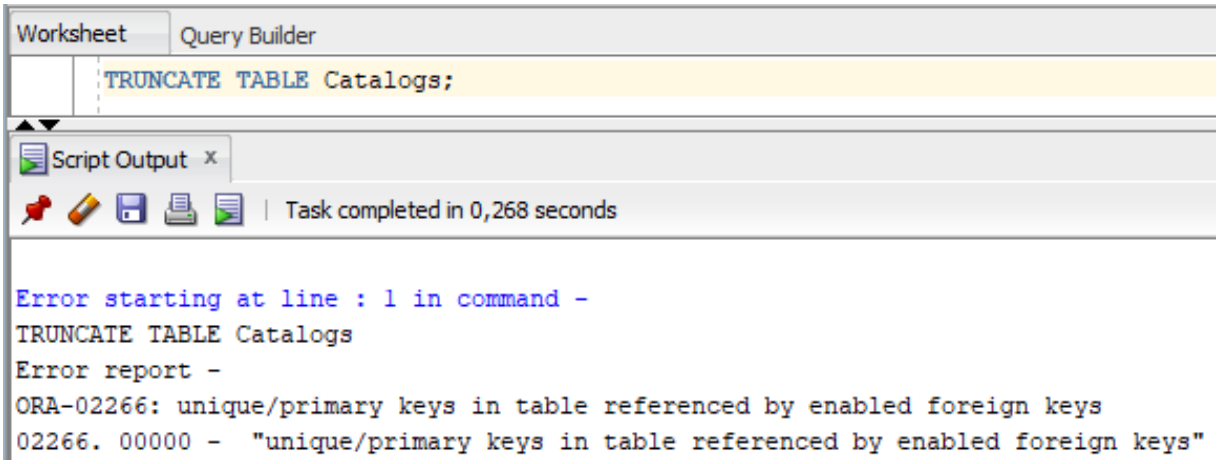


Рис. 40

Иногда отключение ограничений ссылочной целостности, выполнение команды *TRUNCATE* и включение отключенных ограничений будет эффективнее использования команды *DELETE*, тем более что в Oracle есть возможность включения и отключения ограничений (рис. 41).

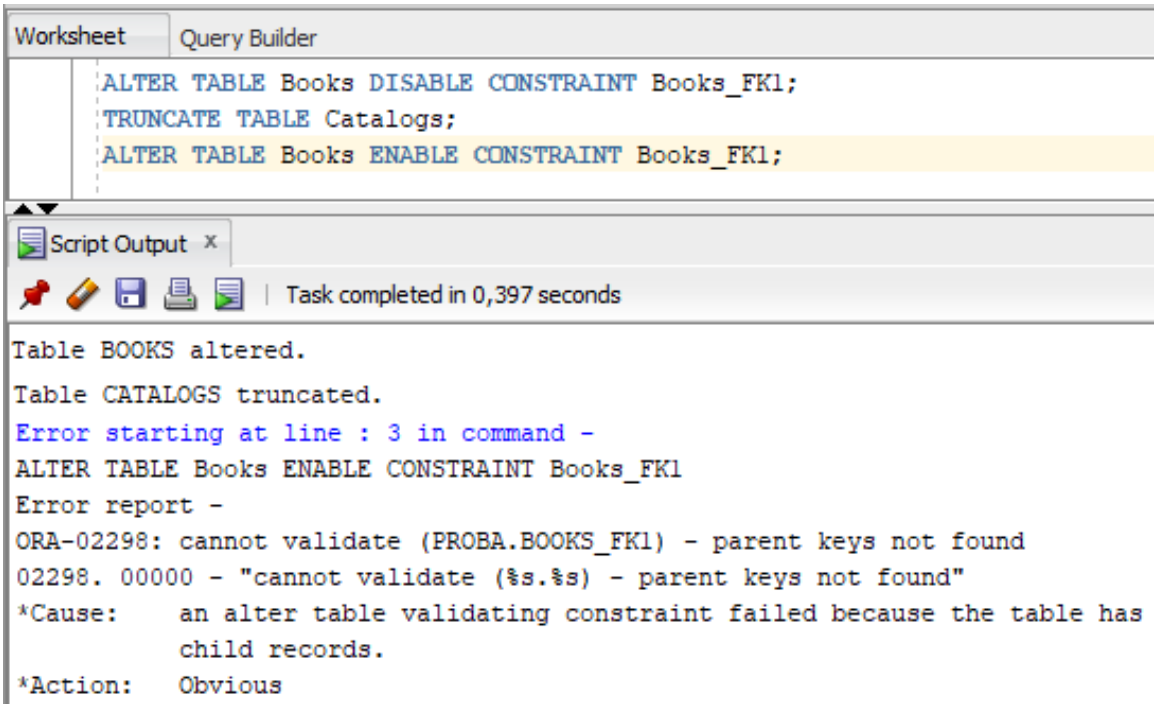


Рис. 41

Мы отключили ограничение, вновь применили команду *TRUNCATE* и снова включили ограничение внешнего ключа. При выполнении последней команды появилось сообщение об ошибке ORA-02298 – не удалось проверить ограничение, поскольку родительские ключи не найдены, а дочерние записи в связанной таблице *Books* существуют. Тем не менее таблица *Catalogs* очищена.

По сравнению с очисткой таблицы путем использования команд *DROP TABLE* и *CREATE TABLE* преимущество *TRUNCATE* в том, что все связанные ограничения, индексы и привилегии сохраняются и после операции *TRUNCATE*.

Однако в этой ситуации невозможно выполнить откат для команды *TRUNCATE* с помощью инструкции *ROLLBACK*, поскольку *TRUNCATE* считается командой DDL. Oracle рассматривает команды DDL как транзакции с одним оператором и немедленно их фиксирует неявной командой *COMMIT*. Если в нескольких таблицах выполнить три команды вставки (*INSERT*), шесть обновлений (*UPDATE*) и два удаления (*DELETE*), то выполнение команды *TRUNCATE* в другой таблице, но в той же транзакции, зафиксирует команды и завершит транзакцию.

Комментарии к таблицам и столбцам таблиц. Добавить в словарь данных пояснения к таблицам и столбцам таблиц можно командой *COMMENT*. Добавим

с помощью этой команды в словарь данных комментарии для таблицы *Customers* и столбца *book_onhand* таблицы *Books* (рис. 42).

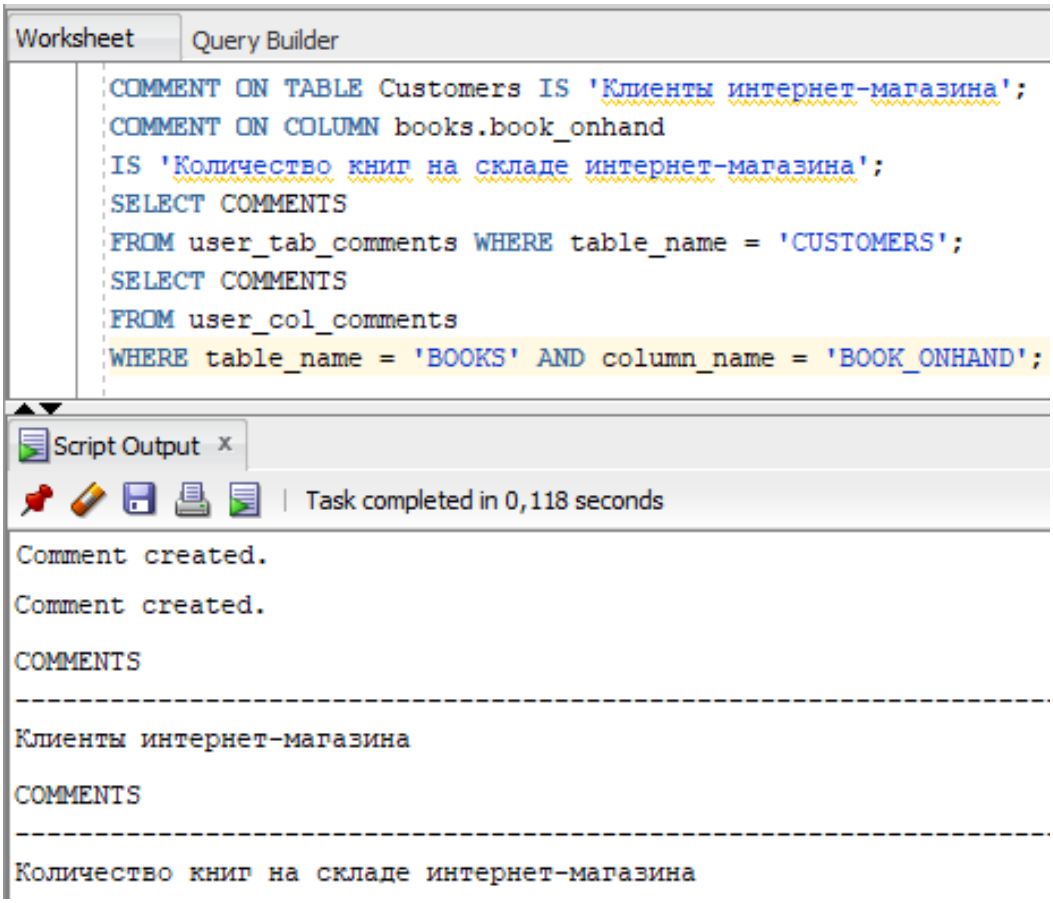


Рис. 42

Практическая работа

При выполнении работы необходимо в Oracle Database XE:

- создать в схеме *Bookshop* реляционные таблицы в соответствии с моделью данных предметной области (рис. 25);
- создать ограничения целостности.

Пример выполнения задания

Создание таблицы *Catalogs*

1. Щелкните правой кнопкой мыши по узлу *Tables* в иерархии схемы в навигаторе и выберите *New Table*. В диалоговом окне *Create Table* установите флажок *Advanced* и заполните поля, как показано на рис. 43.

2. После создания очередного столбца нажимайте кнопку *Add Column*. Если случайно нажмете *OK*, щелкните правой кнопкой мыши по таблице *Catalogs* в окне навигатора соединений, выберите *Edit* и продолжайте добавлять столбцы. Чтобы удалить столбец, выделите его мышью и щелкните *Remove Column*.

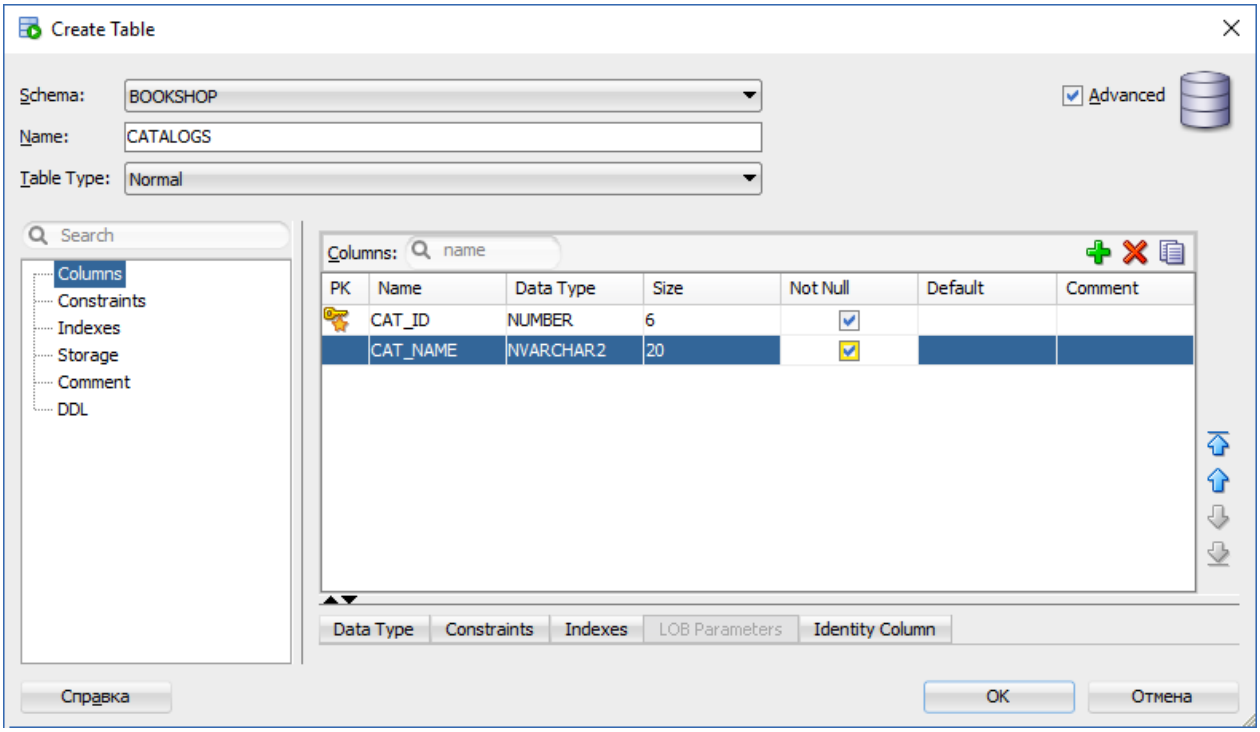


Рис. 43

При заполнении полей обратите внимание на следующее:

- для поля *cat_ID* используется тип данных *NUMBER* с точностью 6 и масштабом 0 (поле может хранить целые числа с точностью до шести значащих цифр); поскольку масштаб равен 0 по умолчанию, ввод масштаба не обязателен;
- для поля *cat_name* используется тип данных *NVARCHAR2* (поле может содержать строки переменной длины до 20 символов кириллицы).

3. Щелкните мышью в поле *PK* строки, соответствующей столбцу *cat_ID*. Будет создан первичный ключ по этому полю. Ограничение первичного ключа автоматически получит имя *Catalogs_PK*.

4. Страница создания таблицы *Create Table* использует одноименную SQL-команду. Эту команду можно увидеть, щелкнув закладку *DDL*:

```
CREATE TABLE Catalogs (
   cat_ID NUMBER (6, 0) NOT NULL,
   cat_name NVARCHAR2 (20) NOT NULL,
   CONSTRAINT Catalogs_PK  PRIMARY KEY (cat_ID)
   ENABLE  );
```

Создание таблицы *Books*

1. Щелкните правой кнопкой мыши по узлу *Tables* в иерархии схемы в навигаторе и выберите *New Table*. В диалоговом окне *Create Table* установите флажок *Advanced* и заполните поля так, как показано на рис. 44.

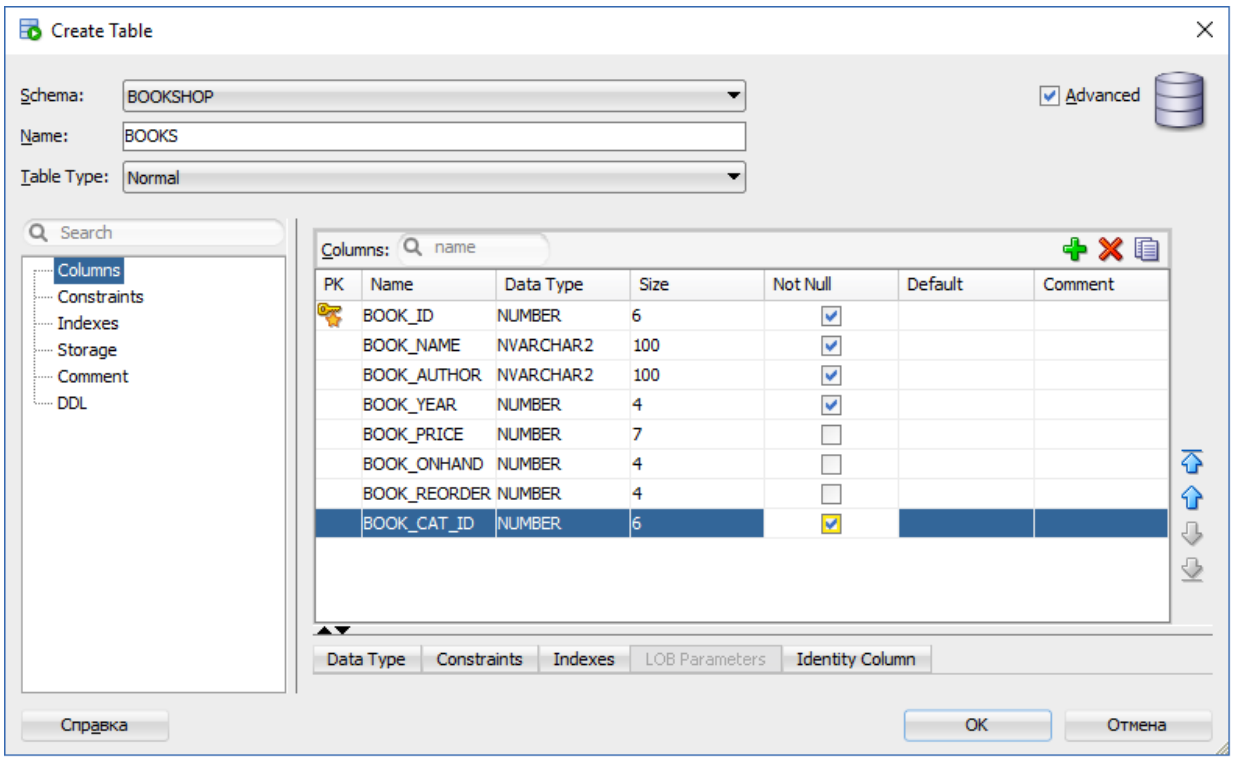


Рис. 44

2. При заполнении полей обратите внимание на следующее:

- для полей первичного и внешнего ключей *book_ID* и *book_cat_ID* используется тип данных *NUMBER* с точностью 6 и масштабом 0;
- для полей *book_name* и *book_author* используется тип *NVARCHAR2* (поля могут содержать строки переменной длины до 100 символов);
- для полей *book_year*, *book_onhand* и *book_reorder* используется тип данных *NUMBER* с точностью 4 и масштабом 0 (поле может хранить целые числа с точностью до четырех значащих цифр);
- для поля *book_price* используется тип данных *NUMBER* с точностью 7 и масштабом 2 (в этом поле могут храниться числа с точностью до десяти цифр, округленные до двух цифр справа от десятичной точки).

Внимание! Для числовых типов в поле *Size* по умолчанию задается значение *Precision*. Если в столбце предполагается хранить вещественные данные с целой и дробной частью (цена в рублях и копейках), то необходимо щелкнуть на закладке *Data Type* и в поле *Scale* задать число позиций под дробную часть.

3. Щелкните мышью в поле *PK* строки, соответствующей столбцу *book_ID*. Будет создан первичный ключ по этому полю. Ограничение первичного ключа автоматически получит имя *Books_PK*.

4. Создайте ограничение внешнего ключа по полю *book_cat_ID*. Для этого щелкните мышью пункт *Constraint*, раскройте меню создания нового ограничения (около зеленого крестика), выберите *New Foreign Key Constraint*. После этого в диалоговом окне выберите значения в списках так, как показано на рис. 45. Ограничение внешнего ключа автоматически получит имя *Books_FK1*.

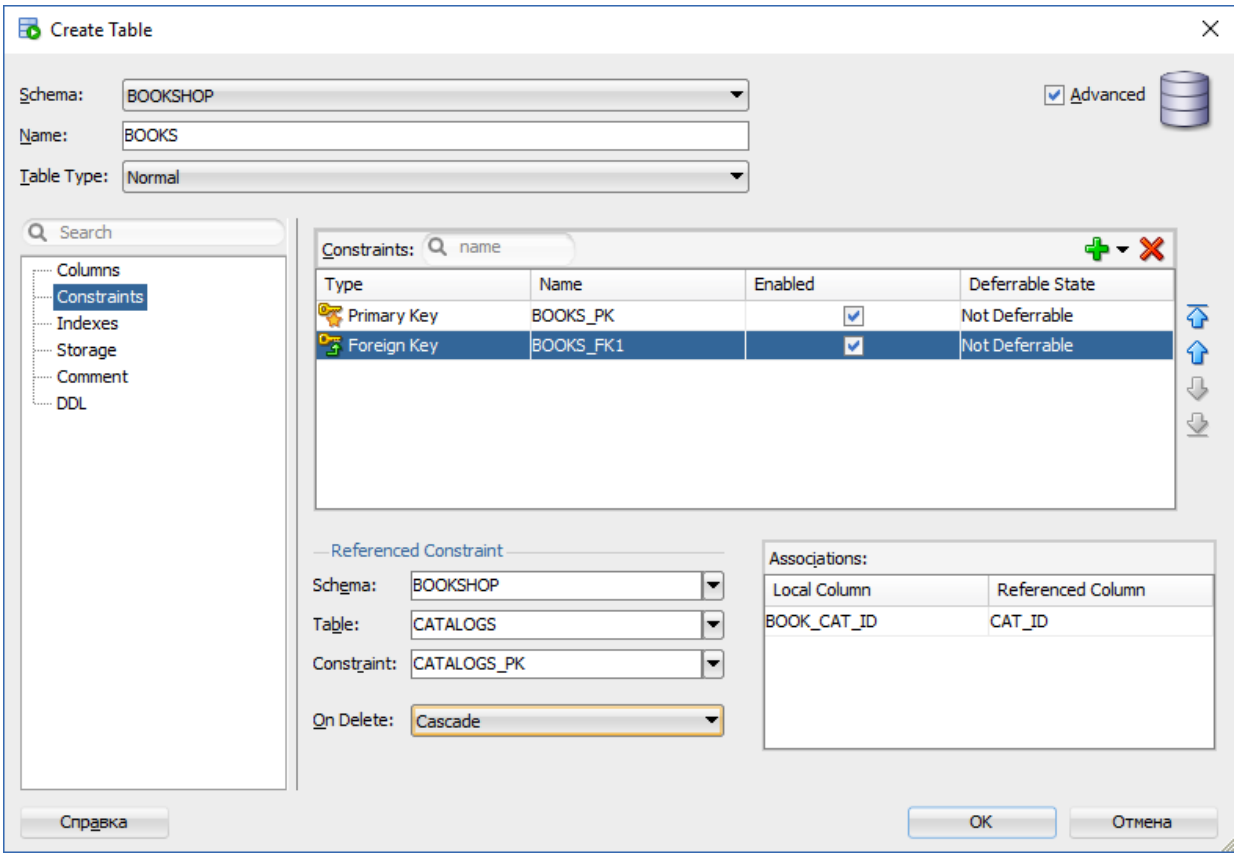


Рис. 45

5. Щелкнув закладку *DDL*, можно увидеть SQL-инструкции создания и изменения таблицы *Books*:

```
CREATE TABLE Books (
    book_ID NUMBER (6, 0) NOT NULL,
    book_name NVARCHAR2 (100) NOT NULL,
    book_author NVARCHAR2 (100) NOT NULL,
    book_year NUMBER (4, 0) NOT NULL,
    book_price NUMBER (7, 2),
    book_onhand NUMBER (4, 0),
    book_reorder NUMBER (4, 0),
    book_cat_ID NUMBER (6, 0) NOT NULL,
    CONSTRAINT Books_PK  PRIMARY KEY (book_ID)
    ENABLE   );
ALTER TABLE Books
```

ADD CONSTRAINT Books_FK1 FOREIGN KEY (book_cat_ID)
REFERENCES Catalogs (cat_ID)
ON DELETE CASCADE
ENABLE;

Внимание! Многие администраторы базы данных не определяют ограничения в запросе *CREATE TABLE*. Вместо этого они добавляют запросы *ALTER TABLE* для добавления в таблицу всех ограничений. При этом приходится вводить больше текста, однако запрос *CREATE TABLE* проще понять без ограничений. Независимое определение ограничений облегчает использование запросов, если нужно удалить и воссоздать ограничения заново.

Создание таблицы *Customers*

1. Щелкните правой кнопкой мыши по узлу *Tables* в иерархии схемы в навигаторе и выберите *New Table*. В диалоговом окне *Create Table* установите флажок *Advanced* и заполните поля так, как показано на рис. 46.

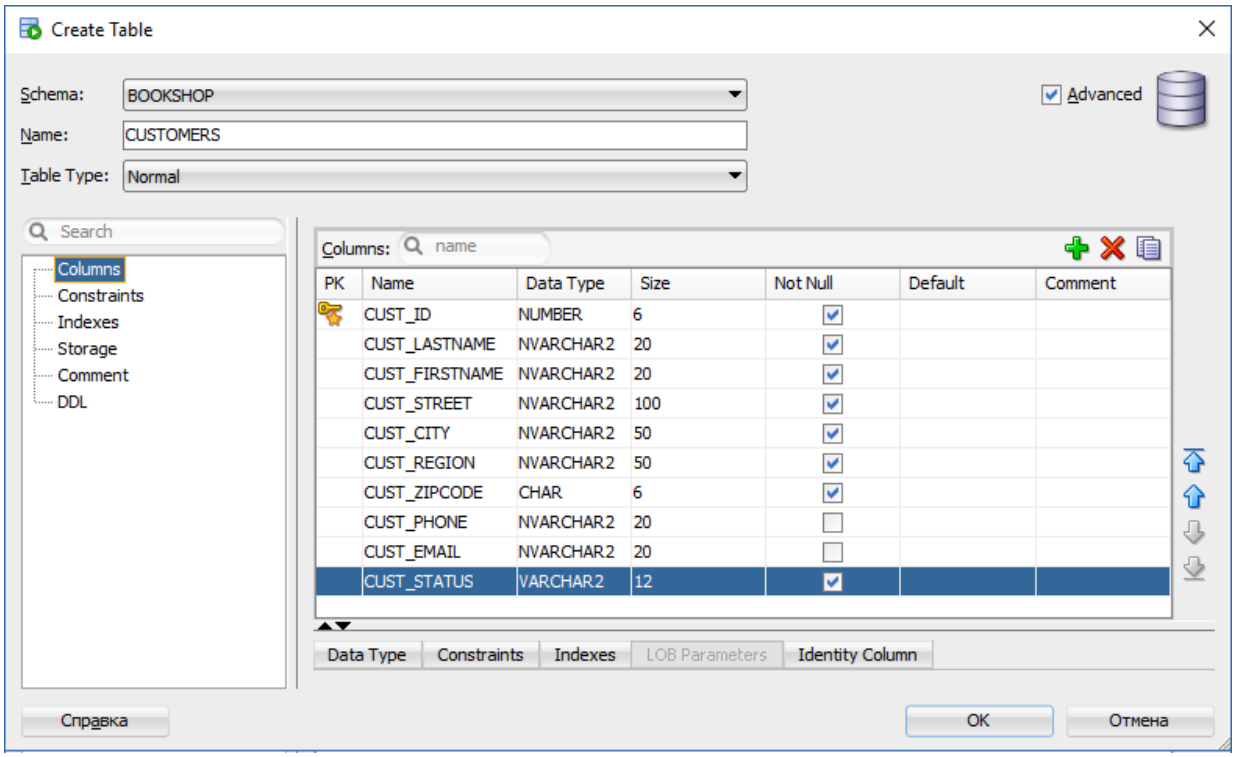


Рис. 46

2. При заполнении полей обратите внимание на следующее:

- для поля внешнего ключа *cust_ID* используется тип данных *NUMBER* с точностью 6 и масштабом 0;
- для полей *cust_lastname*, *cust_firstname*, *cust_phone* и *cust_email* используется тип *NVARCHAR2* (строки переменной длины до 20 символов);

- для полей *cust_street*, *cust_city* и *cust_region* используется тип *NVARCHAR2* (строки переменной длины до 100 и 50 символов соответственно);
- для поля *cust_zipcode* используется тип *CHAR* (поле может содержать строку длиной 6 символов);
- для поля *cust_status* используется тип *VARCHAR2* (поле может содержать строку переменной длины до 12 символов).

Внимание! Для столбцов символьных типов *CHAR* и *VARCHAR2* в поле *Size* по умолчанию устанавливается длина в байтах (*BYTE*). Чтобы задать длину поля в символах, необходимо на закладке *Data Type* в списке *Units* выбрать *CHAR*.

3. Щелкните мышью в поле *PK* строки, соответствующей столбцу *cust_ID*. Будет создан первичный ключ по этому полю. Ограничение первичного ключа автоматически получит имя *Customers_PK*.

4. Чтобы создать ограничение уникальности по полю *cust_phone*, щелкните мышью пункт *Constraint*, раскройте меню около зеленого крестика, выберите *New Unique Constraint*. В диалоговом окне переместите поле *cust_phone* из списка доступных полей в список выбранных. Аналогично создается ограничение уникальности по полю *cust_email*. Эти ограничения получат имена *Customers_UK1* и *Customers_UK2*.

5. Наконец, чтобы установить для поля *cust_status* ограничение проверки *CHECK*, в том же пункте *Constraint* раскройте меню создания нового ограничения (около зеленого крестика) и выберите *New Check Constraint*. В окне *Check Condition* введите *cust_status IN ('gold', 'lock', 'passive', 'active')*.

6. Щелкнув закладку *DDL*, можно увидеть SQL-инструкции создания и изменения таблицы *Customers*:

```
CREATE TABLE Customers (
   cust_ID  NUMBER (6,0)  NOT NULL,
   cust_lastname  NVARCHAR2 (20) NOT NULL,
   cust_firstname  NVARCHAR2 (20) NOT NULL,
   cust_street  NVARCHAR2 (100) NOT NULL,
   cust_city  NVARCHAR2 (50) NOT NULL,
   cust_region  NVARCHAR2 (50) NOT NULL,
   cust_zipcode  CHAR (6 CHAR) NOT NULL,
   cust_phone  NVARCHAR2 (20),
   cust_email  NVARCHAR2 (20),
   cust_status  VARCHAR2 (12 CHAR) NOT NULL,
   CONSTRAINT  Customers_PK  PRIMARY KEY (cust_ID)
   ENABLE  );
```

```
ALTER TABLE Customers
   ADD CONSTRAINT Customers_UK1 UNIQUE (cust_phone)
ENABLE;
ALTER TABLE Customers
   ADD CONSTRAINT Customers_UK2 UNIQUE (cust_email)
ENABLE;
ALTER TABLE Customers
   ADD CONSTRAINT Customers_CHK1 CHECK
      (cust_status IN ('gold', 'lock', 'active', 'passive'))
ENABLE;
```

Создание таблицы *Orders*

1. Щелкните правой кнопкой мыши по узлу *Tables* в иерархии схемы на левой стороне, выберите *New Table*. В диалоговом окне *Create Table* установите флажок *Advanced* и заполните поля так, как показано на рис. 47.

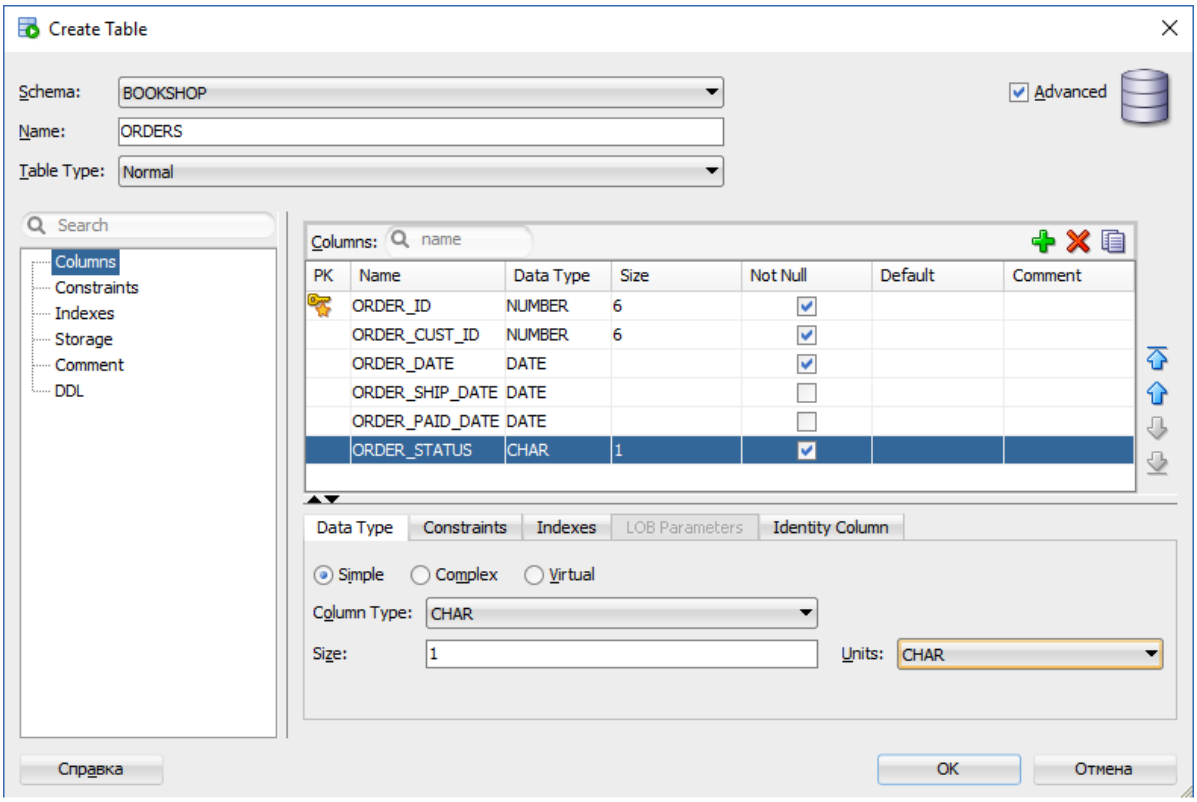


Рис. 47

2. При заполнении полей обратите внимание на следующее:

- для полей первичного и внешнего ключей *order_ID* и *order_cust_ID* используется тип данных *NUMBER* с точностью 6 и масштабом 0;
- для полей *order_date*, *order_ship_date*, *order_paid_date* используется тип *DATE* (поля могут содержать корректные значения даты и времени);

• для поля *order_status* используется тип *CHAR* (поле может содержать только один символ).

3. Для создания ограничения внешнего ключа щелкните мышью пункт *Constraint*, раскройте меню около зеленого крестика и выберите *New Foreign Key Constraint*. В диалоговом окне выберите значения, как показано на рис. 48.

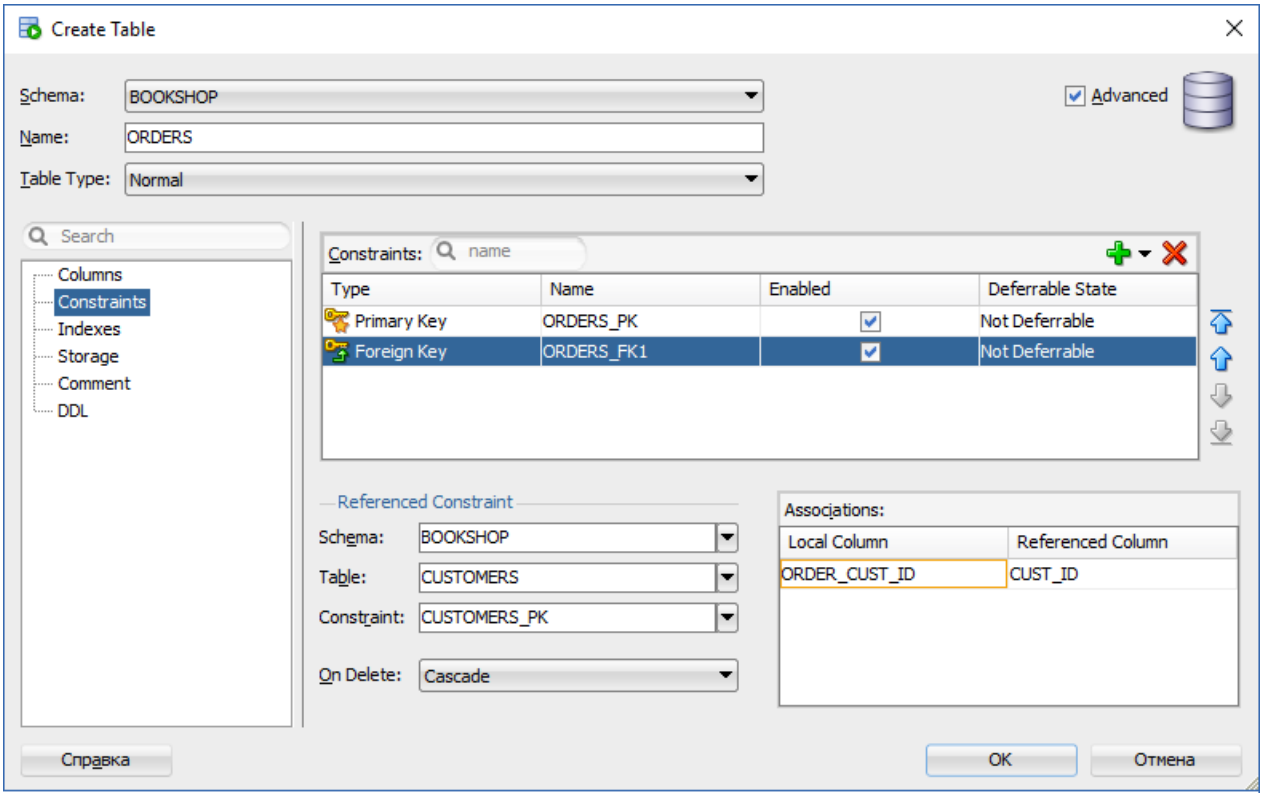


Рис. 48

4. Чтобы установить для поля *order_status* ограничение проверки *CHECK*, в пункте *Constraint* раскройте меню около зеленого крестика и выберите *New Check Constraint*. В окне *Check Condition* введите *order_status IN ('F', 'B', 'O')*.

5. Щелкнув закладку *DDL*, можно увидеть SQL-команды создания и изменения таблицы *Orders*:

```
CREATE TABLE Orders (
    order_ID NUMBER (6, 0) NOT NULL,
    order_cust_ID NUMBER (6, 0) NOT NULL,
    order_date DATE NOT NULL,
    order_ship_date DATE,
    order_paid_date DATE,
    order_status CHAR (1 CHAR) NOT NULL,
    CONSTRAINT Orders_PK PRIMARY KEY (order_ID)
    ENABLE );
```

```
ALTER TABLE Orders
   ADD CONSTRAINT Orders_FK1 FOREIGN KEY (order_cust_ID)
      REFERENCES Customers (cust_ID)
   ON DELETE CASCADE
ENABLE;
ALTER TABLE Orders
   ADD CONSTRAINT Orders_CHK1 CHECK
      (order_status IN ('F', 'B', 'O'))
ENABLE;
```

Создание таблицы *Items*

1. Щелкните правой кнопкой мыши по узлу *Tables* в иерархии схемы на левой стороне, выберите *New Table*. В диалоговом окне *Create Table* установите флажок *Advanced* и заполните поля так, как показано на рис. 49.

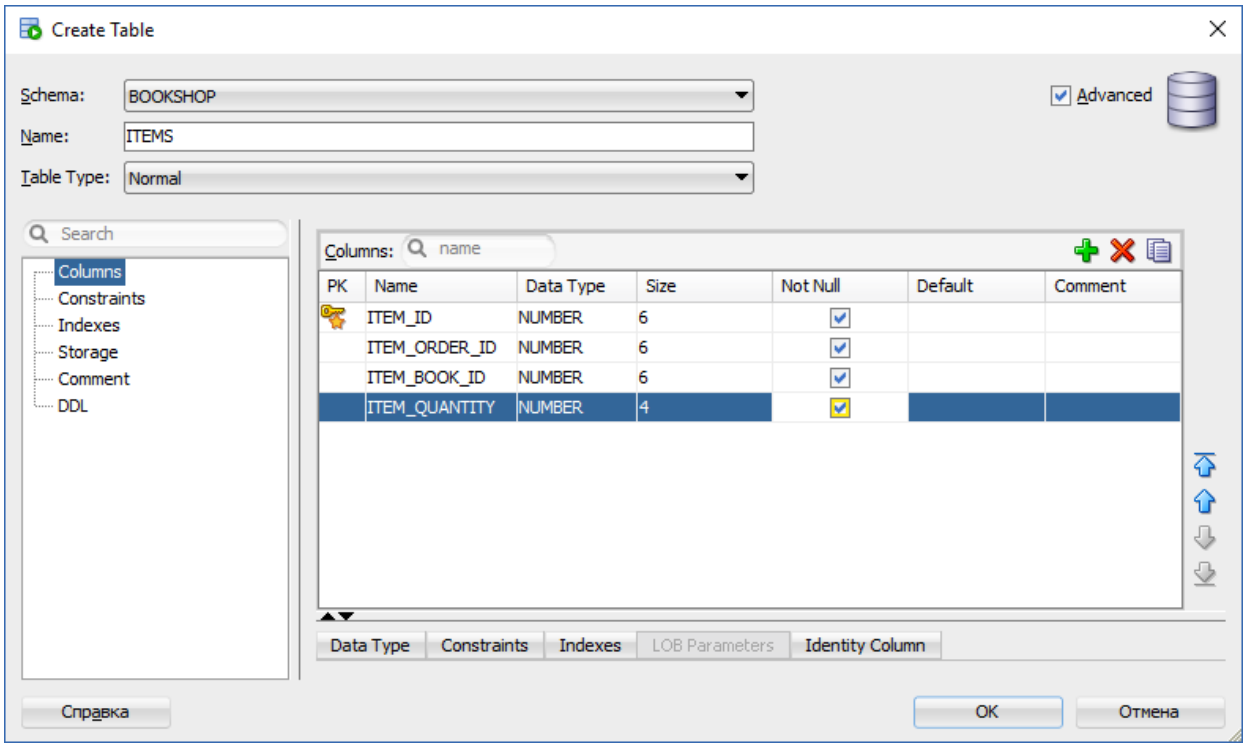


Рис. 49

2. При заполнении полей обратите внимание на следующее:

- для полей первичного и внешних ключей *item_ID*, *item_order_ID* и *item_book_ID* используется тип данных *NUMBER* с точностью 6 и масштабом 0;
- для поля *item_quantity* используется тип данных *NUMBER* с точностью 4 и масштабом 0 (целые числа с точностью до четырех значащих цифр).

3. Для создания первого ограничения внешнего ключа щелкните мышью пункт *Constraint*, раскройте меню около зеленого крестика, выберите *New Foreign Key Constraint*. В диалоговом окне выберите значения, как на рис. 50.

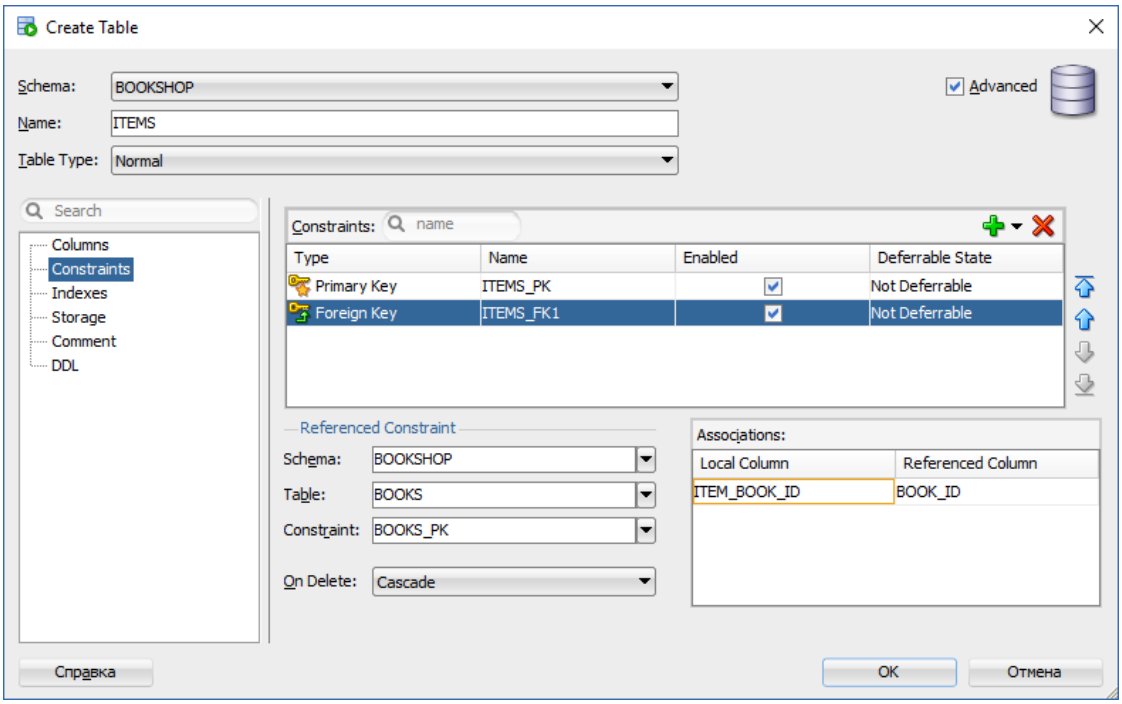


Рис. 50

4. Для создания второго ограничения внешнего ключа щелкните мышью пункт *Constraint*, раскройте меню около зеленого крестика и выберите *New Foreign Key Constraint*. В диалоговом окне выберите значения, как на рис. 51.

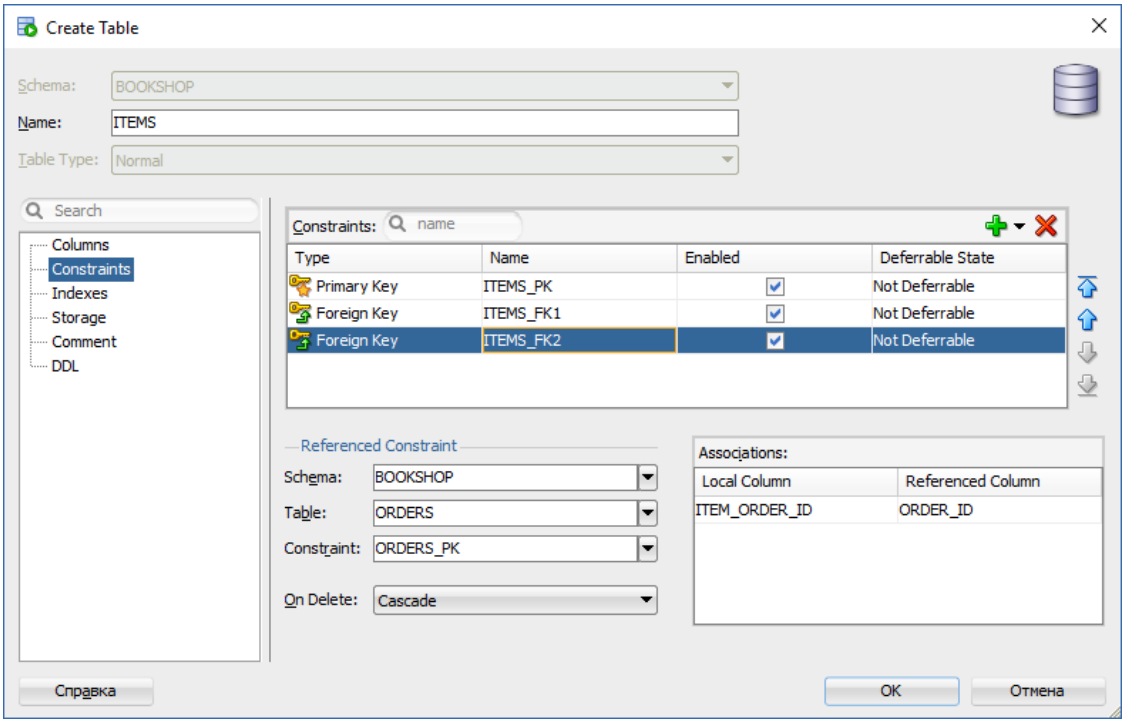


Рис. 51

5. Щелкнув закладку *DDL*, можно увидеть SQL-команды создания и изменения таблицы *Items*:

```
CREATE TABLE Items (
   item_ID NUMBER (6, 0) NOT NULL,
   item_order_ID NUMBER (6, 0) NOT NULL,
   item_book_ID NUMBER (6, 0) NOT NULL,
   item_quantity NUMBER (4, 0) NOT NULL,
   CONSTRAINT Items_PK  PRIMARY KEY (item_ID)
   ENABLE  );
ALTER TABLE Items
   ADD CONSTRAINT Items_FK1  FOREIGN KEY (item_book_ID)
      REFERENCES Books (book_ID)
      ON DELETE CASCADE
ENABLE;
ALTER TABLE Items
   ADD CONSTRAINT Items_FK2  FOREIGN KEY (item_order_ID)
      REFERENCES Orders (order_ID)
      ON DELETE CASCADE
ENABLE;
```

Внимание! В приведенных операторах *ENABLE* указывает на включение ограничений целостности для создаваемой (изменяемой) таблицы. Это ограничение должно быть определено в операторе создания (изменения) таблицы. По умолчанию включаются все ограничения целостности, определенные в операторе.

Построение диаграммы базы данных. Диаграмму для схемы *Bookshop* построим в Data Modeler. Выберите в меню *File → Data Modeller → Import → Data Dictionary*. Затем в мастере:

1) выберите подключение *bookshop-XE*;

2) выберите схему *Bookshop* и *All Selected* (два флажка);

3) выберите таблицы схемы и отмените выбор нежелательных объектов;

4) нажмите кнопку *Finish* и увидите результат (рис. 52).

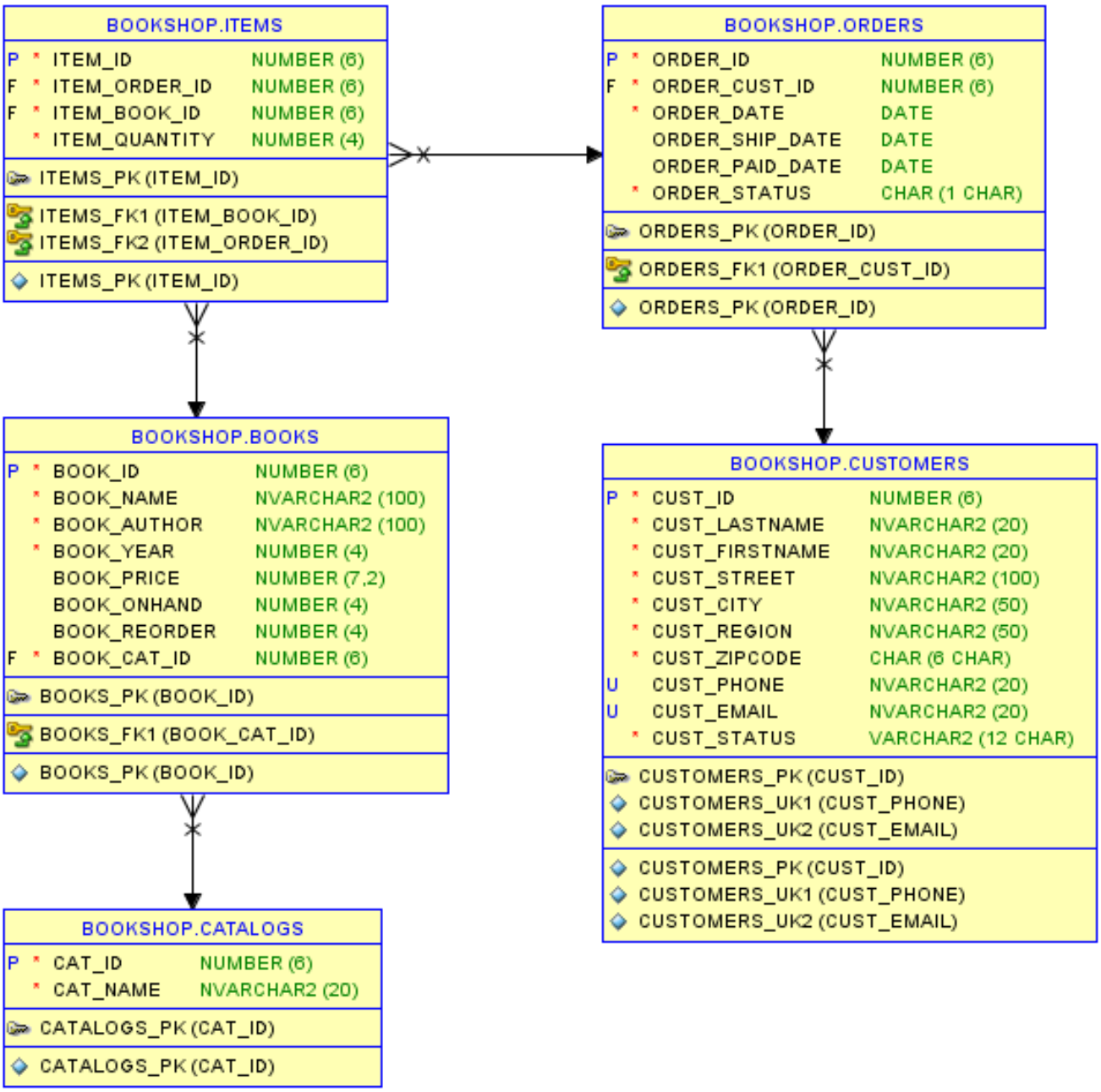


Рис. 52

Сценарии создания таблиц и ограничений. Объединив инструкции SQL, сгенерированные в процессе создания и связывания таблиц схемы *Bookshop*, получим сценарий (скрипт), приведенный в прил. А. В этом сценарии ограничения (кроме *NOT NULL*) добавляются в отдельных запросах *ALTER TABLE*. Рекомендовано использование именно такого подхода, поскольку, во-первых, запрос *CREATE TABLE* проще понять без ограничений. Во-вторых, независимое определение ограничений облегчает их удаление и модификацию.

Практическое занятие № 4
Загрузка, модификация и удаление данных

Теоретические сведения

Есть несколько вариантов загрузки данных в таблицы схемы *Bookshop*:

- ввод данных вручную с помощью приложения SQL Developer;
- ввод данных с помощью инструкции *INSERT*;
- загрузка данных из текстовых файлов;
- загрузка данных из файлов XML.

Ввод данных вручную. Предполагает открытие таблицы в приложении SQL Developer и последовательное заполнение ячеек этой таблицы.

Ввод данных с помощью инструкции *INSERT*. Инструкция обеспечивает добавление строк в таблицу несколькими способами. С ее помощью можно:

- добавить одну полную строку;
- добавить часть строки;
- добавить результаты запроса.

Добавление полных строк реализует базовый синтаксис *INSERT*. Если известна полная информация обо всех столбцах таблицы (которую предоставляет, например, команда SQL*Plus *DESCRIBE*), то в инструкции *INSERT* можно не указывать имена столбцов после имени таблицы (рис. 53).

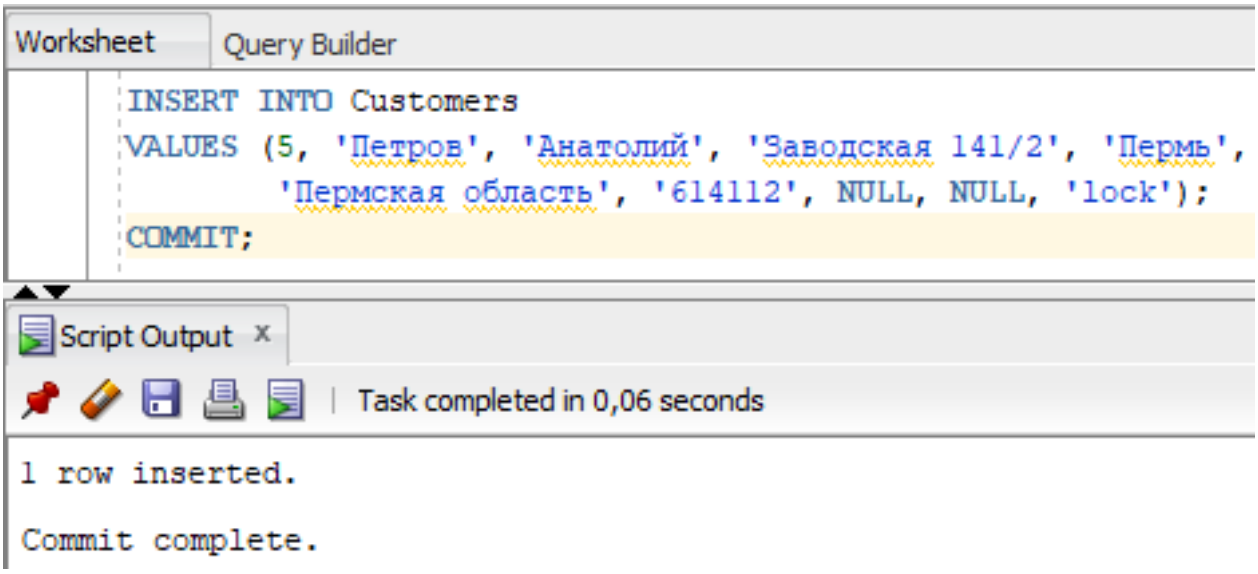


Рис. 53

Значения указываются в секции *VALUES* через запятую для каждого столбца как литералы или выражения. Если они не известны (*cust_phone* и *cust_email*), указывают *NULL* (значения *NULL* в этих столбцах допускаются). Эта форма *INSERT* небезопасна. Безопасный способ записи представлен на рис. 54.

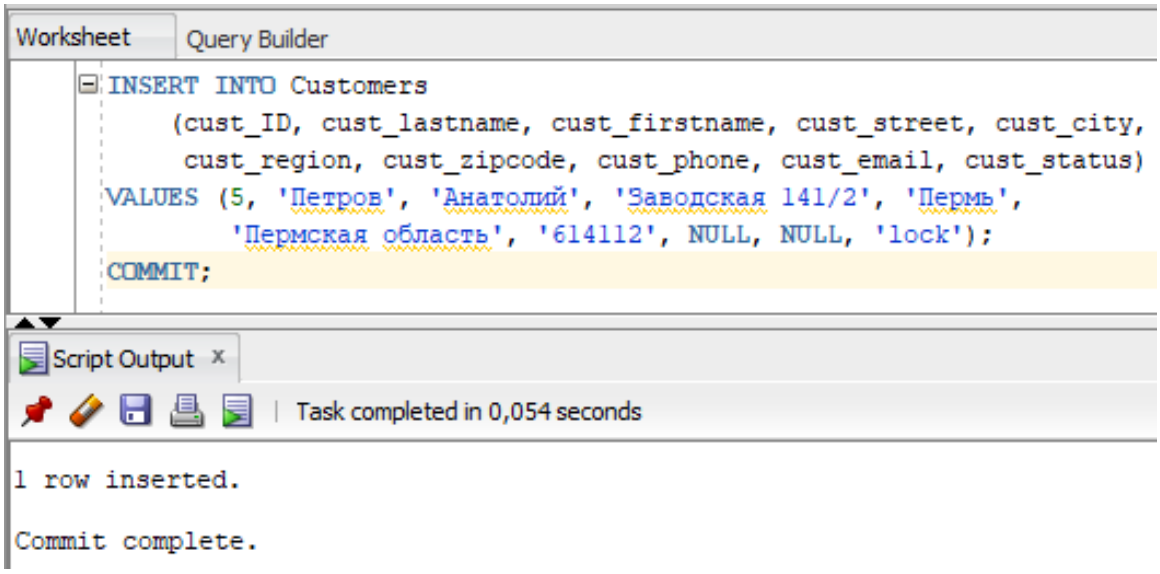


Рис. 54

Имена столбцов явно указаны в круглых скобках после имени таблицы. Значения в секции *VALUES* должны соответствовать столбцам (порядок следования столбцов не обязательно должен совпадать с порядком столбцов в таблице).

> Внимание! Если имена столбцов отсутствуют, необходимо привести значение для каждого столбца таблицы. Если имена столбцов указаны, то должно быть задано значение для каждого столбца в списке.

В рассмотренном выше примере для столбцов *cust_phone* и *cust_email* вводились значения *NULL*. Инструкция *INSERT* может не включать эти два столбца и два соответствующих им значения. Так реализуется добавление части строки (рис. 55).

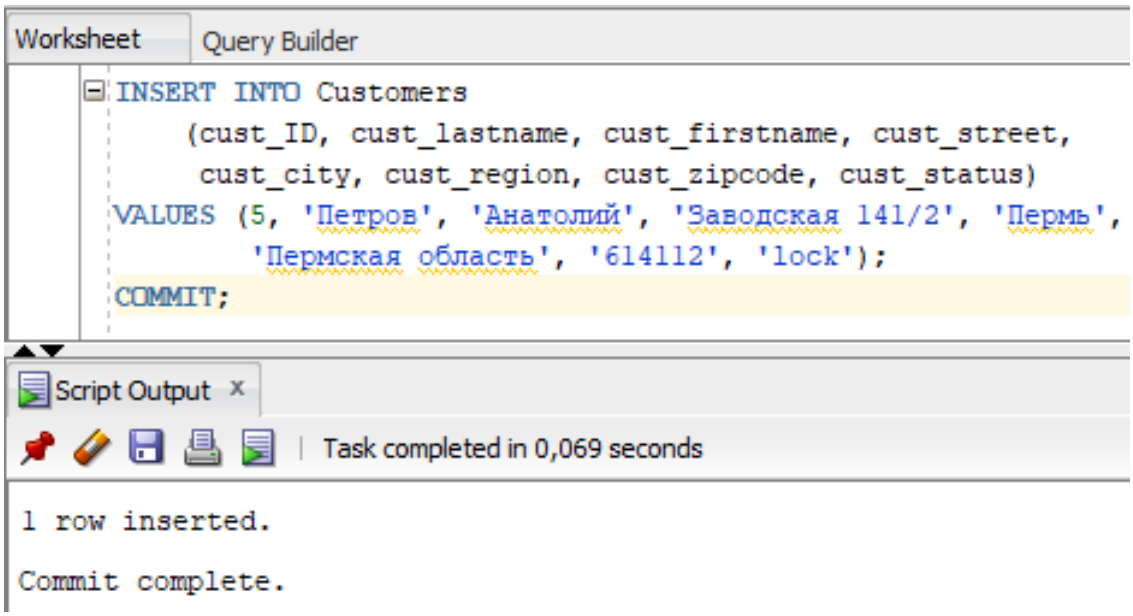


Рис. 55

Столбцы могут быть исключены из *INSERT*, если:

- столбец определен как допускающий значения *NULL*;
- в определении столбца задано значение по умолчанию.

Инструкция *INSERT* в Oracle добавляет только одну строку. Для добавления нескольких строк выполняют несколько *INSERT* (запускают сценарий). Исключением является инструкция *INSERT INTO … SELECT*, которая может добавлять множество строк с помощью одного запроса.

Пусть необходимо занести в таблицу *CustNew* список клиентов из основной таблицы *Customers*. Это можно сделать, как показано на рис. 56.

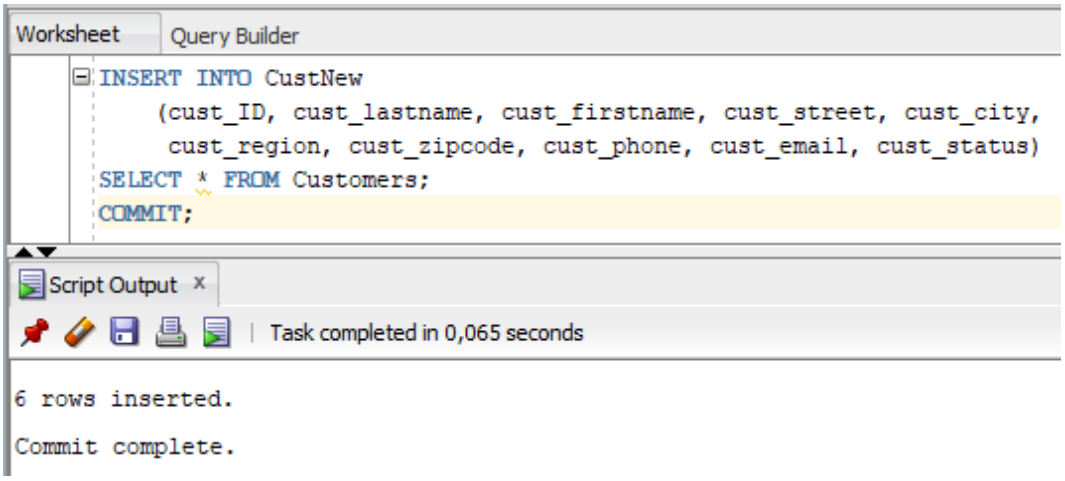


Рис. 56

В этом примере данные импортируются из таблицы *Customers* в таблицу *CustNew*. Таблица *CustNew* должна существовать и иметь такой же формат, как таблица *Customers*. В таблице *CustNew* уже могут быть данные, но не должны использоваться значения *cust_ID*, имеющиеся в таблице *Customers*. Для простоты использованы одинаковые имена столбцов в секциях *INSERT INTO* и *SELECT*, что не обязательно. Oracle не обращает внимания на имена столбцов, возвращаемых инструкцией *SELECT*, и учитывает лишь их положение.

Есть еще один способ добавления данных в таблицу, при котором команда *INSERT* вообще не применяется. Можно скопировать содержимое любой таблицы в новую таблицу, создаваемую сразу же (рис. 57).

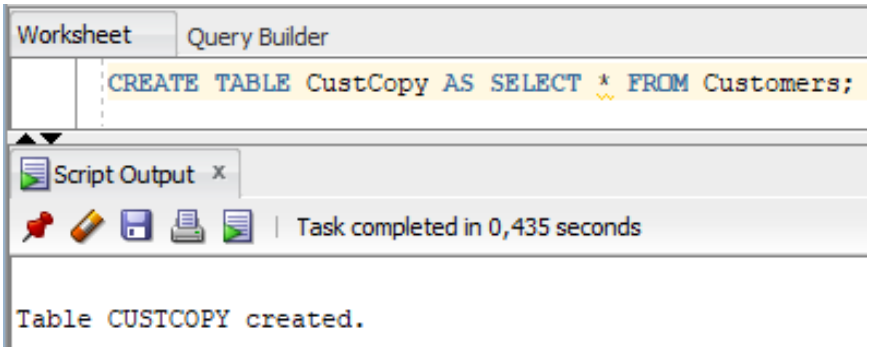


Рис. 57

Инструкция создает новую таблицу *CustCopy* (таблица создается без ограничения первичного ключа) и копирует в нее содержимое таблицы *Customers*. Поскольку применена конструкция *SELECT **, каждый столбец таблицы *Customers* будет воссоздан в таблице *CustCopy*. Чтобы скопировать часть столбцов, следует явно указать их имена. При использовании инструкции:

- допускается применение любых ключевых слов и секций инструкции *SELECT*, в том числе *WHERE* и *GROUP BY*;
- допускается добавление данных из нескольких таблиц посредством использования операции соединения;
- данные добавляются только в одну таблицу независимо от того, из скольких таблиц они извлечены.

Можно даже использовать вложенный запрос к той же самой таблице, в которую вставляются строки (рис. 58).

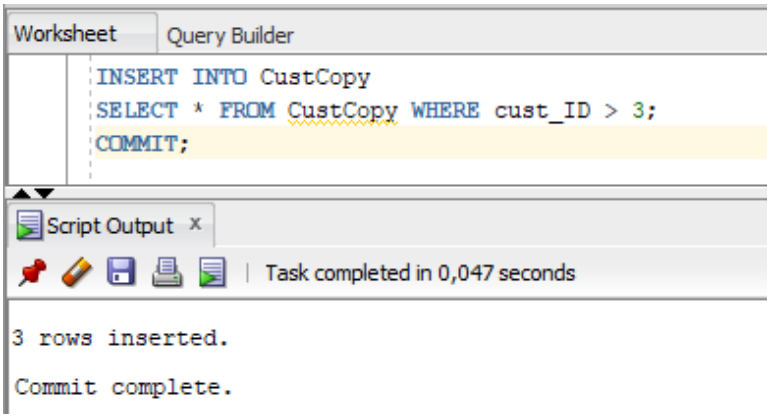
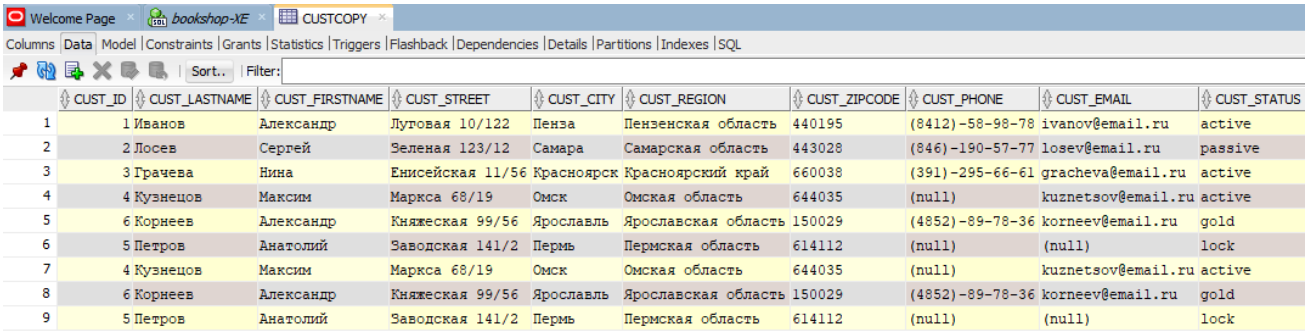


Рис. 58

Это один из наиболее быстрых методов заполнения таблицы, если в ней отсутствуют ограничения по уникальному или первичному ключу (рис. 59).


Welcome Page | bookshop-XE | CUSTCOPY

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter:

	CUST_ID	CUST_LASTNAME	CUST_FIRSTNAME	CUST_STREET	CUST_CITY	CUST_REGION	CUST_ZIPCODE	CUST_PHONE	CUST_EMAIL	CUST_STATUS
1	1	Иванов	Александр	Луговая 10/122	Пенза	Пензенская область	440195	(8412)-58-98-78	ivanov@email.ru	active
2	2	Лосев	Сергей	Зеленая 123/12	Самара	Самарская область	443028	(846)-190-57-77	losev@email.ru	passive
3	3	Грачева	Нина	Енисейская 11/56	Красноярск	Красноярский край	660038	(391)-295-66-61	gracheva@email.ru	active
4	4	Кузнецов	Максим	Маркса 68/19	Омск	Омская область	644035	(null)	kuznetsov@email.ru	active
5	6	Корнеев	Александр	Княжеская 99/56	Ярославль	Ярославская область	150029	(4852)-89-78-36	korneev@email.ru	gold
6	5	Петров	Анатолий	Заводская 141/2	Пермь	Пермская область	614112	(null)	(null)	lock
7	4	Кузнецов	Максим	Маркса 68/19	Омск	Омская область	644035	(null)	kuznetsov@email.ru	active
8	6	Корнеев	Александр	Княжеская 99/56	Ярославль	Ярославская область	150029	(4852)-89-78-36	korneev@email.ru	gold
9	5	Петров	Анатолий	Заводская 141/2	Пермь	Пермская область	614112	(null)	(null)	lock

Рис. 59

Обновление данных с помощью SQL-команды *UPDATE*. Обновить данные какой-либо таблицы позволяет инструкция *UPDATE*, которая обеспечивает:

- обновление определенных строк в таблице;
- обновление всех строк в таблице.

Инструкция *UPDATE* состоит из трех основных частей:

- имени обновляемой таблицы;
- имен столбцов и их новых значений;
- условий фильтрации, определяющих, какие строки следует обновить.

Рассмотрим пример. Пусть у клиента с кодом 4 появился номер телефона. Его запись можно обновить с помощью инструкции, показанной на рис. 60.

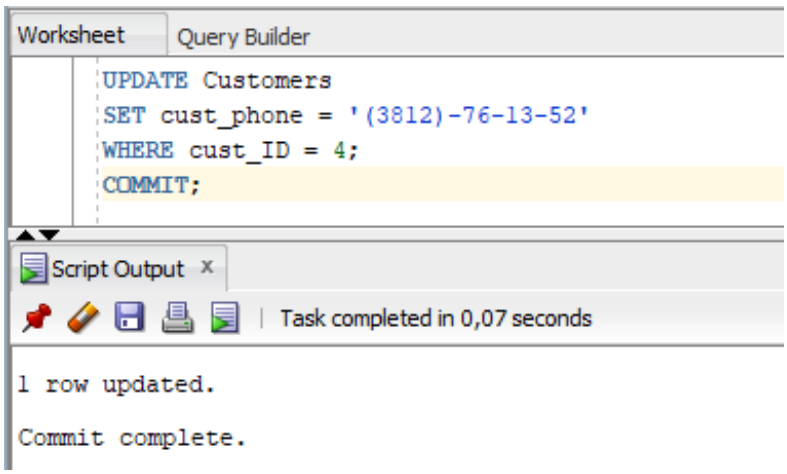


Рис. 60

> Внимание! Не забывайте указывать предложение *WHERE*, которое сообщает, какая именно строка подлежит обновлению. При отсутствии этого предложения СУБД обновит все строки таблицы.

Обновить несколько столбцов можно одним предложением *SET*. Каждая пара «столбец-значение» отделяется запятой. Например, пусть клиент с кодом 4 сменил адрес места жительства, оставаясь в том же городе (рис. 61).

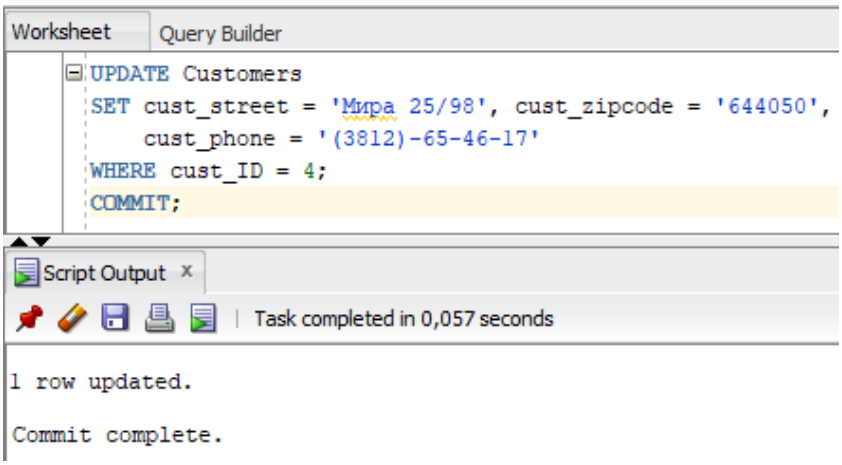


Рис. 61

В инструкциях *UPDATE* можно использовать подзапросы, что позволяет обновлять столбцы данными, извлеченными с помощью инструкции *SELECT*.

Вложенный запрос можно использовать для фильтрации обновляемых записей. Так, вместо литерального значения *cat_ID = 2* можно использовать вложенный запрос, позволяющий управлять обновлением с помощью данных таблицы (рис. 62).

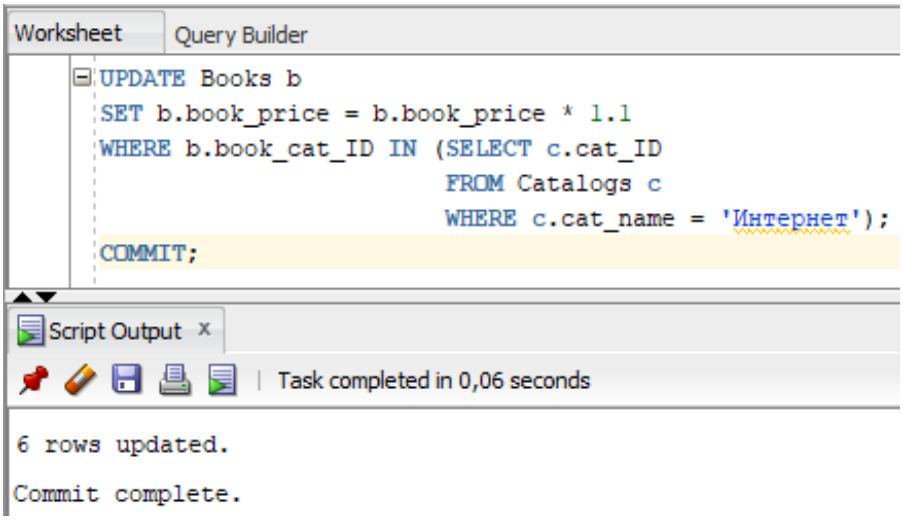


Рис. 62

Удаление данных. Удалить данные можно с помощью инструкции *DELETE*, обеспечивающей удаление определенных строк или всех строк таблицы.

> Внимание! Не забывайте указывать предложение *WHERE*, которое сообщает, какие именно строки подлежит удалению. При отсутствии этого предложения СУБД удалит все строки таблицы.

Следующая инструкция удаляет из таблицы *Customers* одну строку, соответствующую клиенту с кодом 5 (рис. 63).

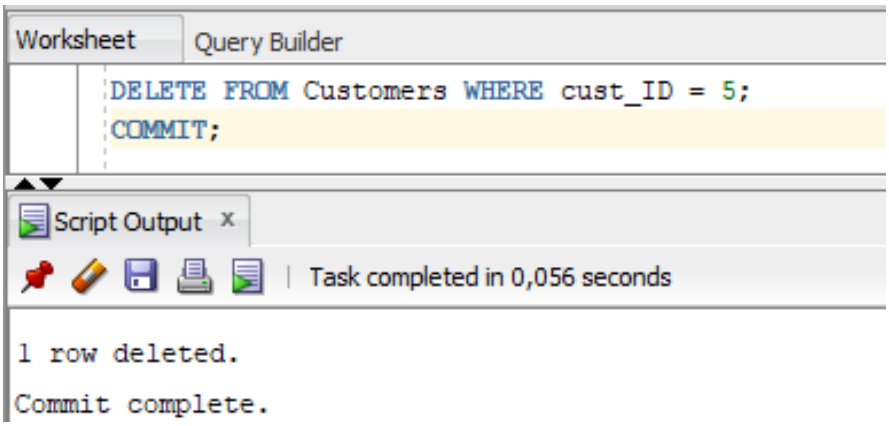


Рис. 63

Есть разница между инструкциями *DROP TABLE* и *DELETE*. Первая удаляет и содержимое таблицы, и саму таблицу, включая все зависимые объекты и струк-

туры (индексы, привилегии и пр.). *DROP TABLE* – это инструкция языка определения данных DDL; ее последствия нельзя отменить с помощью *ROLLBACK*. Инструкция *DELETE* удаляет только содержимое. Это инструкция языка обработки данных DML, и ее последствия отменяются *ROLLBACK*.

В предложениях *FROM* и *WHERE* инструкции *DELETE* можно использовать вложенные запросы. Например, удаление всех книг из каталога с кодом 2 (рис. 64).

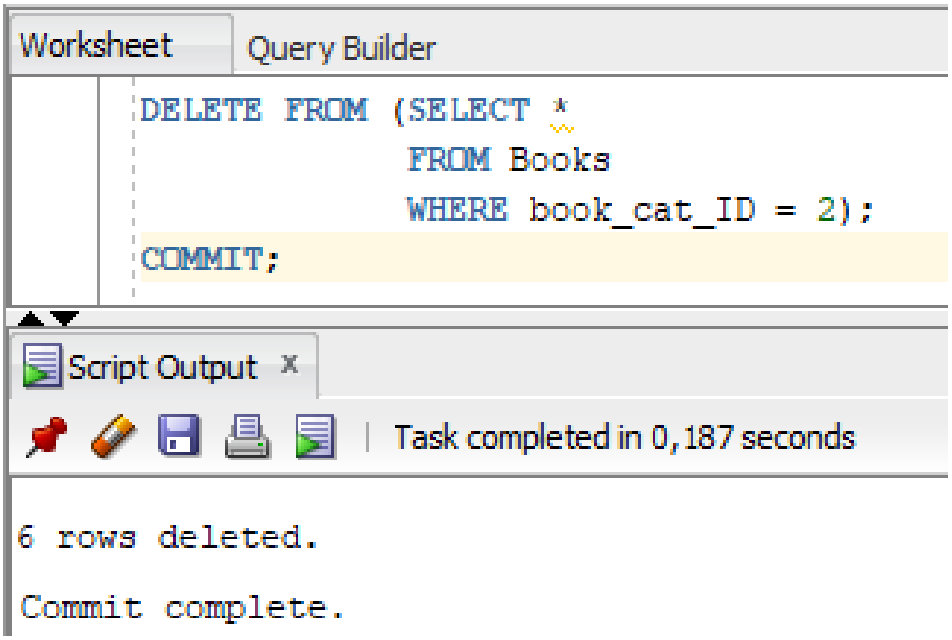


Рис. 64

Альтернативный вариант – удаление всех книг из каталога «Базы данных» (рис. 65).

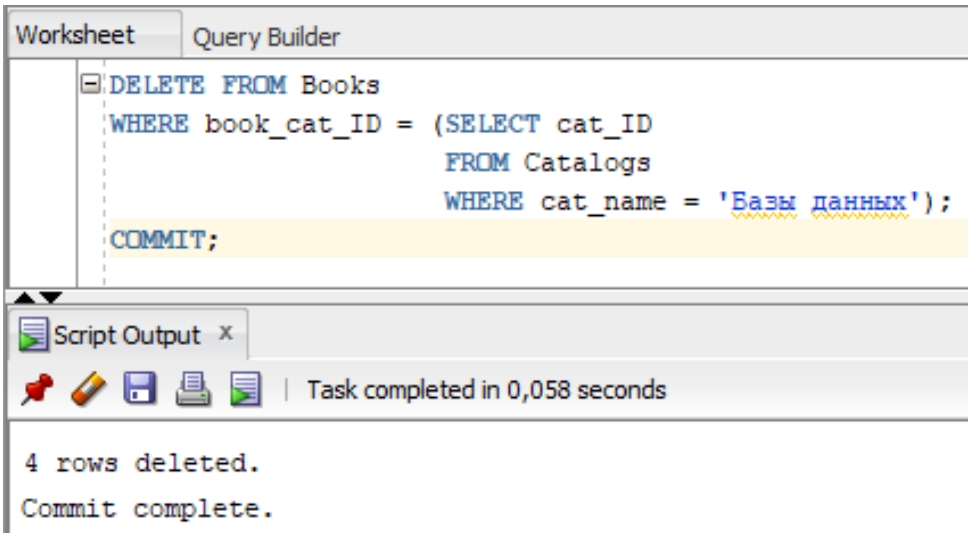


Рис. 65

В этих случаях использование вложенных запросов не имеет преимуществ перед использованием обычной инструкции *DELETE*.

При удалении строк могут возникнуть сложности из-за нарушения ограничений ссылочной целостности. В общем случае, когда строки существуют в дочерней таблице, удаление завершается ошибкой. Поэтому при установке ограничения ссылочной целостности стоит предусмотреть каскадное удаление.

Есть еще инструкция *TRUNCATE*, которая более эффективно удаляет все строки таблицы, чем *DELETE*. Инструкция *TRUNCATE* относится к языку определения данных DDL и была рассмотрена выше.

Наконец, существует инструкция *MERGE*, способная выполнять вставку, обновление и удаление в одном операторе. Эта команда эффективна в хранилищах данных, где таблицы массово заполняются и обновляются из внешних источников. Команда *MERGE* может реагировать на существование или отсутствие определенных строк в обновляемых таблицах.

Практическая работа

При выполнении задания необходимо в базе данных Oracle XE:

- ввести данные в таблицы схемы *Bookshop*, используя рассмотренные выше способы.

Пример выполнения задания

Загрузка данных вручную в таблицу *Catalogs*. Для добавления строк вручную в таблицу Catalogs посредством приложения SQL Developer выполните следующие действия:

1. В окне *Connections* разверните подключение *bookshop-XE* и раскройте узел *Tables*. Выберите в списке таблицу *Catalogs*, при этом в правой части окна отобразится структура столбцов таблицы.

2. В правой части окна выберите закладку *Data*, при этом появится панель данных с указанием столбцов таблицы и без каких-либо строк.

3. На панели инструментов щелкните кнопку *Insert Row*. Появится новая строка с пустыми столбцами. Зеленая рамка вокруг номера строки указывает на то, что вставка не была совершена.

4. Щелкните ячейку под заголовком столбца *cat_ID* и введите значение *1*.

5. Нажмите клавишу *Tab* или щелкните ячейку под заголовком столбца *cat_name* и введите значение *Программирование*. Нажмите клавишу *Enter*.

6. Заполните оставшиеся строки таблицы, повторяя шаги 3–5.

7. Нажмите на значок *Commit Changes*. Зеленые границы вокруг номеров строк исчезнут. Под панелью данных появится панель *Messages – Log*, на которой будет сообщение *Commit Successful*. В области данных проверьте новые строки (рис. 66).

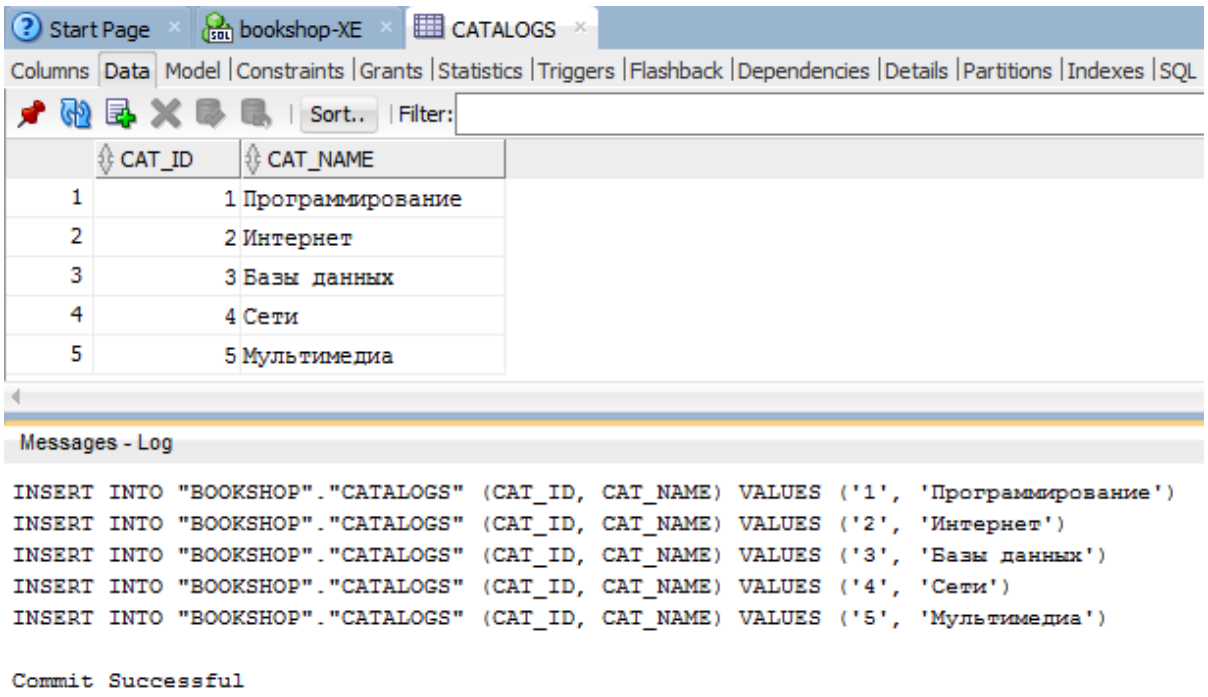


Рис. 66

Изменение данных вручную в таблицах области данных. Чтобы изменить данные в таблице *Catalogs* с помощью панели данных приложения SQL Developer, выполните следующие действия:

1. В окне *Connections* разверните подключение *bookshop-XE* и раскройте узел *Tables*. Выберите в списке таблицу *Catalogs*, при этом в правой части окна отобразится структура столбцов таблицы.

2. В правой части окна выберите закладку *Data (Данные)*, при этом появится панель данных с заполненной таблицей *Catalogs*.

3. В строке, где *cat_ID* имеет значение *3*, щелкните значение *Базы данных*, введите значение *СУБД* и нажмите клавишу *Enter*. Слева от номера строки появится звездочка, указывающая на то, что изменение не было совершено.

4. Нажмите на значок *Commit Changes*. Звездочка слева от номера строки исчезнет. Под панелью данных проверьте наличие сообщения *Commit Successful (Фиксация успешна)*. В области данных проверьте измененные данные.

Удаление строк вручную из таблиц в области данных. Чтобы удалить строку из таблицы *Catalogs* с помощью инструмента *Delete Selected Row(s) (Удалить выбранную строку)*, выполните следующие действия:

1. В окне *Connections* разверните подключение *Bookshop-XE* и раскройте узел *Tables*. Выберите в списке таблицу *Catalogs*, при этом в правой части окна отобразится структура столбцов таблицы.

2. В правой части окна выберите закладку *Data (Данные)*, при этом появится панель данных с заполненной таблицей *Catalogs*.

3. В области данных щелкните строку, в которой в поле *cat_name* находится *СУБД*, и нажмите на иконку *Delete Selected Row(s)*. Вокруг номера строки появляется красная граница, показывающая, что удаление не было совершено.

4. Нажмите на значок *Commit Changes.* Строка удаляется. Под панелью данных проверьте наличие сообщения *Commit Successful (Фиксация успешна).* В области данных проверьте измененные данные.

Ввод, изменение и удаление данных с помощью инструкций *INSERT, UPDATE* и *DELETE.* Ввод данных в таблицы схемы *Bookshop* осуществляется так, как показано выше. Сценарий (скрипт) заполнения данными таблиц демонстрационной схемы *Bookshop* приведен в прил. Б.

Практическое занятие № 5
Поддержка транзакций в Oracle

Теоретические сведения

Транзакция (Transaction) – последовательность действий, выполняемая одной или несколькими связанными инструкциями SQL. Фиксация транзакции (Committing) обеспечивает запись на диск изменений, сделанных в процессе выполнения транзакции. Откат транзакции (Rolling back) обеспечивает аннулирование всех изменений, сделанных операторами в теле транзакции.

Понятие транзакции в реляционной теории отсутствует и составляет самостоятельный по отношению к ней предмет изучения. Транзакции позволяют:

- восстанавливать данные при аварийном прекращении сеанса связи приложения с СУБД;
- гарантировать целостное представление данных приложению при одновременной работе с данными нескольких приложений;
- гарантировать целостное хранение данных в базе при одновременной работе с ними нескольких приложений.

Oracle обеспечивает два режима выполнения транзакций.

- *Режим автофиксации*. В этом режиме каждая инструкция SQL автоматически фиксируется по завершении. В Oracle режим автофиксации по умолчанию отключен. Он включается и выключается инструкциями:

SET AUTOCOMMIT ON;

SET AUTOCOMMIT OFF.

- *Неявный режим*. Транзакция неявно начинается, когда пользователь соединяется с базой данных (начинается новая сессия). В Oracle этот режим задан по умолчанию. Когда текущая транзакция заканчивается фиксацией или откатом, автоматически начинается новая. Вложенные транзакции не разрешены.

Oracle поддерживает следующие инструкции работы с транзакциями [6]:

- *COMMIT [WORK];*
- *ROLLBACK [WORK] [TO SAVEPOINT <имя_точки_сохранения>];*
- *SAVEPOINT <имя_точки_сохранения>;*
- *SET TRANSACTION <тип_транзакции>.*

Слово *WORK* в инструкциях *COMMIT* и *ROLLBACK* имеет необязательный характер и употребляется по желанию.

Инструкции *COMMIT* и *ROLLBACK.* В Oracle нет команды создания новой транзакции (отличие от стандарта SQL!), но есть две команды завершения: с фиксацией результатов (*COMMIT*) и отказом от них (*ROLLBACK*). Соединение с СУБД автоматически начинает новую транзакцию. То же самое происходит по завершении команды *COMMIT* или *ROLLBACK*. Таким образом, все операции с данными всегда выполняются в рамках какой-нибудь транзакции, а сеанс связи с СУБД – это последовательность сменяющих друг друга транзакций. Команды завершения транзакции касаются только операций DML, но иногда СУБД порождает такие команды самостоятельно:

- всякая инструкция языка DDL (*CREATE, DROP, ALTER*) завершается неявной выдачей *COMMIT*;
- обычное завершение сессии завершается неявной выдачей *COMMIT*;
- аварийное завершение сессии (отменяется соединение клиента, база данных повреждается или выключается с использованием установок, прекращающих клиентские соединения) сопровождается неявной выдачей *ROLLBACK*.

Внимание! Все изменения, вызываемые в сеансе командами DML *(INSERT, UPDATE, DELETE* и *MERGE*), изначально получают статус «ожидание». Сеанс видит измененные строки, но другие пользователи при запросе этих строк будут видеть исходные данные. Другие пользователи не смогут изменить эти строки, пока вы не подтвердите ожидающие изменения или не откажетесь от них.

Инструкции *SAVEPOINT* и *ROLLBACK TO SAVEPOINT.* Инструкция *SAVEPOINT* позволяет вставить внутри транзакции именованную *точку сохранения*, к которой в рамках текущей транзакции можно вернуться и продолжить выполнение с этого места *(ROLLBACK TO SAVEPOINT).* Точек сохранения может быть сколь угодно много, и одно и то же имя можно использовать повторно. Однако если в пределах транзакции имена точек сохранения совпали, то вернуться можно будет только к последней.

Инструкция *SET TRANSACTION.* Позволяет в начале транзакции (до первой изменяющей данные инструкции DML) назначить тип транзакции:

- транзакция «чтение-запись»;
- транзакция «только для чтения».

По умолчанию каждая новая транзакция в текущем сеансе помечается как *транзакция «чтение-запись»*. Она может включать любую инструкцию SQL, в том числе инструкции DML, выполняющие запросы, вставляющие, изменяющие и удаляющие строки в таблицах. При этом видны результаты, как будто непосредственно получаемые в базе данных. Можно явно объявить тип «чтение-запись»:

SET TRANSACTION READ WRITE;

Транзакции «только для чтения» включают в себя только инструкции запросов. Такие транзакции не изменяют содержимое базы данных. Явное объявление транзакции как предназначенной только для чтения:

SET TRANSACTION READ ONLY;

Если транзакция объявлена «только для чтения», то Oracle гарантирует для нее *согласование при чтении на уровне транзакции*. Результирующие таблицы всех запросов этой транзакции будут отражать состояние базы данных на момент выдачи *SET TRANSACTION READ ONLY*, даже если другие транзакции уже изменили и зафиксировали изменения в базе данных.

Приложения, составляющие отчеты, используют объявленные «только для чтения» транзакции для выполнения группы запросов и получения отчета с согласованными данными из нескольких таблиц. Однако попытка выполнить в них инструкции *INSERT*, *UPDATE* или *DELETE* приведет к ошибке.

Понятие типа транзакции в Oracle связано с понятием *уровня изоляции транзакции* (*Isolation Level)*, определяемого стандартом SQL. Под уровнем изоляции понимается способность транзакции видеть данные вне своей области определения, т. е. изменяемые другими транзакциями. Стандарт SQL определяет четыре уровня изоляции транзакций (высокие уровни включают в себя более низкие):

- чтение неподтвержденного (Read Uncommitted);
- чтение подтвержденного (Read Committed);
- повторяемое чтение (Repeatable Read);
- упорядочение (Serializable).

Соответственно, в Oracle определены три уровня изоляции транзакций (не вполне соответствующие стандарту SQL):

- *READ ONLY;*
- *(ISOLATION LEVEL) READ COMMITED;*
- *(ISOLATION LEVEL) SERIALIZABLE.*

Если уровень изоляции *READ ONLY*, то внутри транзакции нельзя изменять данные инструкциями *UPDATE, INSERT* и *DELETE*. Тип транзакции *ISOLATION LEVEL SERIALIZABLE* аналогичен *READ ONLY*, но не запрещает выполнять собственные операции *INSERT, UPDATE* и *DELETE*. Это дает программисту большую свободу действий, но возникает риск блокирования транзакции.

Безопаснее всего использовать принятый в Oracle по умолчанию уровень изоляции транзакций *READ COMMITTED*, обеспечивающий максимальную про-

пускную способность при небольшом риске аномалий невоспроизводимых и фантомных чтений.

Два последних уровня изоляции устанавливаются стандартной инструкцией:

SET TRANSACTION ISOLATION LEVEL { SERIALIZABLE | READ COMMITTED };

Транзакция любого типа в Oracle не может увидеть изменения, внесенные другими незавершенными транзакциями (отсутствует «грязное чтение»).

Практическая работа

При выполнении работы необходимо в базе данных Oracle:

- создать в транзакции точку сохранения, осуществить фиксацию и откат транзакции в неявном режиме;
- выполнить действия с данными в режиме автофиксации;
- объявить в неявном режиме транзакцию «только для чтения».

Пример выполнения задания

Фиксация и откат транзакции. Объедините в неявную транзакцию несколько операций по добавлению в таблицу *Catalogs* новых каталогов, создайте точку сохранения, добавьте еще один каталог, а затем выполните откат транзакции к точке сохранения (рис. 67).

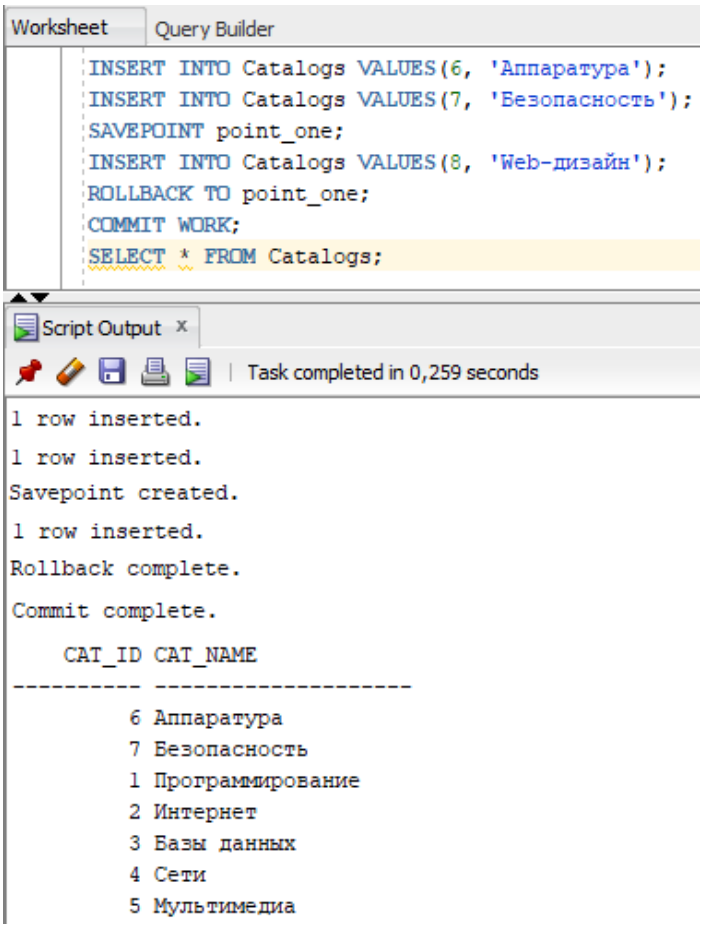


Рис. 67

Режим автофиксации. Включите режим автофиксации и выполните те же действия с таблицей *Catalogs* (удалив предварительно добавленные ранее строки). Обратите внимание на то, как в этом режиме фиксируются результаты (рис. 68).

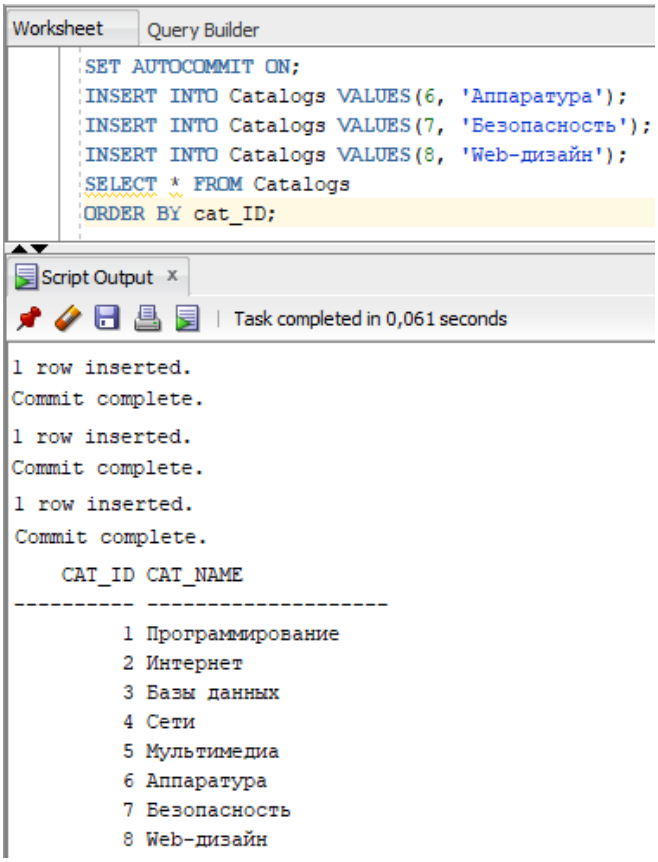


Рис. 68

Транзакция «только для чтения». Отключите режим автофиксации, объявите транзакцию «только для чтения» и попытайтесь выполнить какую-либо операцию DML (например, попробуйте удалить добавленные строки) (рис. 69).

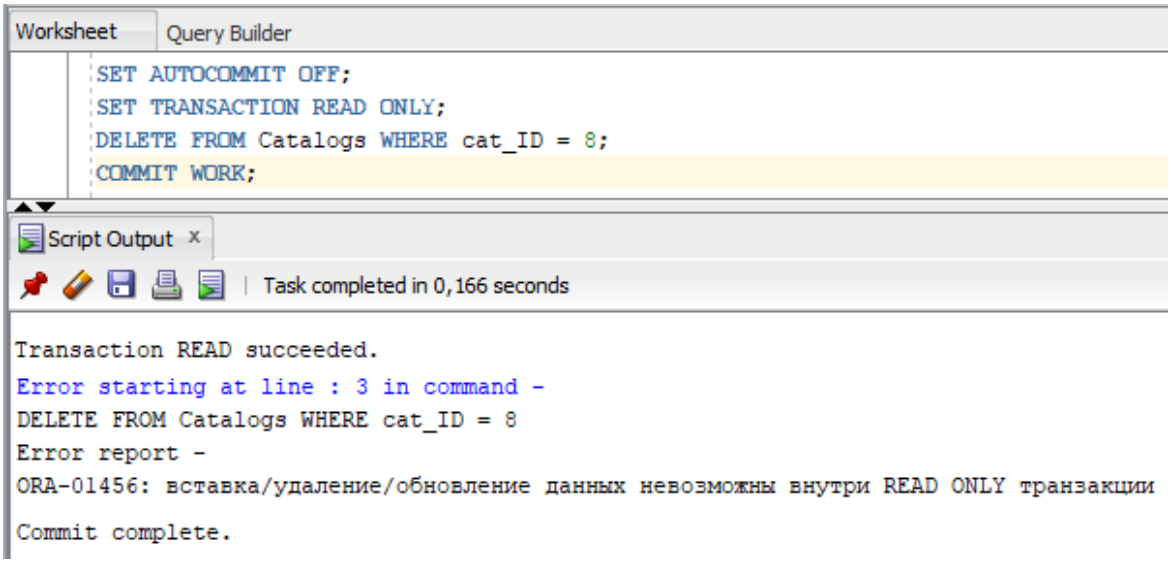


Рис. 69

Практическое занятие № 6
Последовательности и синонимы

Теоретические сведения

Рассмотрим объекты схемы, применение которых позволяет сделать использование схемы более эффективным [2, 7].

Последовательности. OLTP-приложения одновременно работают с большим количеством пользователей. Конкурирующие транзакции вставляют строки в таблицы, и генерирование уникальных первичных ключей может стать сложной задачей. Для первичных ключей таблиц используют монотонно увеличивающиеся порядковые номера заказов, счетов-фактур и пр. Можно сохранять последнее значение каждого первичного ключа во вспомогательной таблице, однако такой подход гарантированно создаст проблемы в многопользовательской среде. В таких случаях лучше использовать последовательности.

Последовательность (Sequence) – объект схемы, генерирующий ряд уникальных целых чисел. Используется для таблиц с ключами в виде простых числовых полей. Приложение, добавляющее строку в таблицу, запрашивает для первичного ключа очередное значение в последовательности. Приложение может использовать сгенерированный номер для координации значений внешних ключей. Генерация последовательностей незначительно увеличивает нагрузку, поэтому она эффективна даже для требовательных к ресурсам OLTP-приложений.

> Внимание! Некоторые значения в последовательности могут быть пропущены (исчезают и больше не появляются). Oracle не может гарантировать отсутствие пропусков в последовательности. Однако это не является проблемой.

Последовательности могут:

- начинаться с любого числа: 0, 25, –123 и т. д.;
- создаваться в виде ряда возрастающих или убывающих чисел с любым постоянным шагом;
- генерировать циклический набор чисел (например, от 1 до 100 с шагом 1, от 999 до –999 с шагом –1 и т. д.).

Определение последовательности может состоять из начального значения, значения приращения, минимального и максимального значений. Можно указать, должен ли генератор останавливаться при достижении граничного значения или циклически повторять последовательность номера в пределах заданного диапазона. Все атрибуты необязательны и имеют значения по умолчанию.

Ниже показано, как можно с помощью таблицы *DUAL* создать и использовать последовательность *deptno_seq* для генерации номеров отделов (рис. 70).

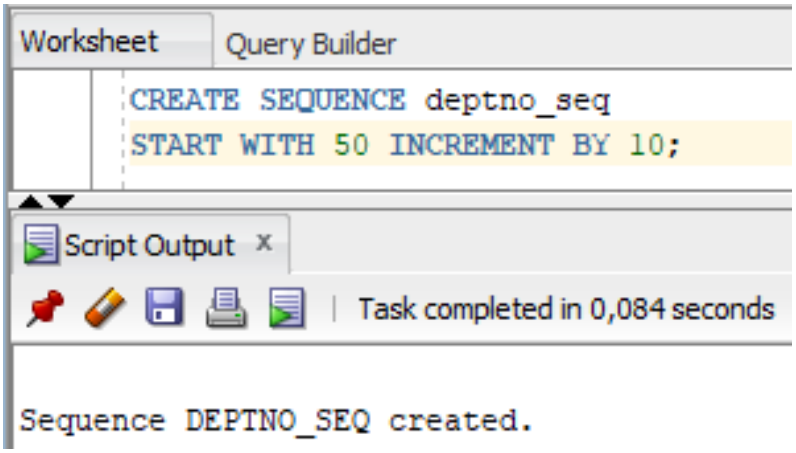


Рис. 70

Каждая последовательность имеет псевдостолбцы *NEXTVAL* и *CURRVAL*, причем *CURRVAL* для одного *NEXTVAL* можно использовать многократно (рис. 71).

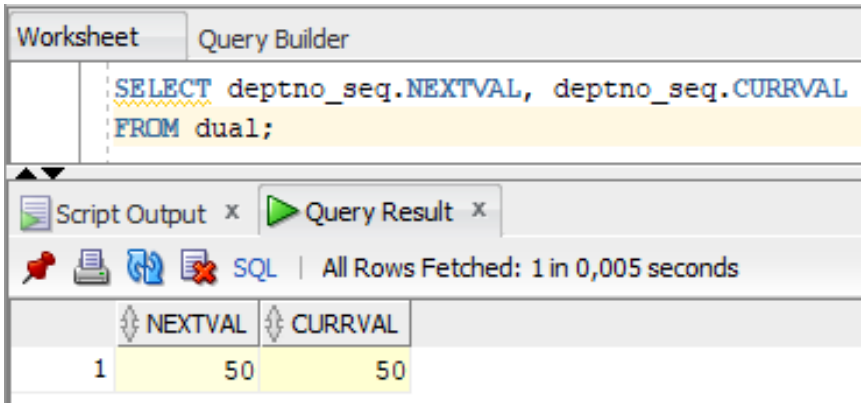


Рис. 71

В примере с интернет-магазином для вставки нового заказа можно выбрать значение последовательности с помощью *NEXTVAL*, а затем несколько раз использовать одно и то же значение *CURRVAL* при вставке элементов этого заказа (рис. 72).

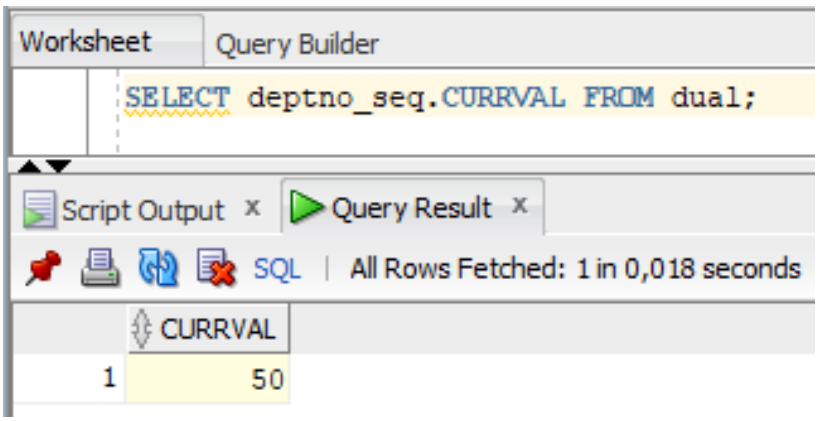


Рис. 72

Обратите внимание на то, что *CURRVAL* выбирается перед *NEXTVAL* (рис. 73).

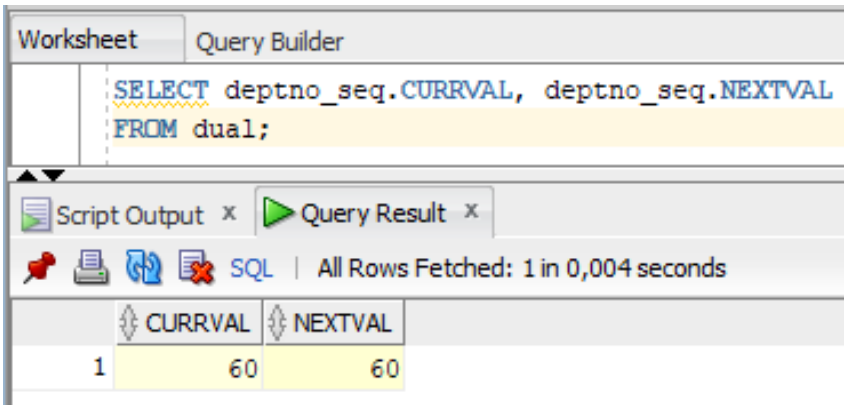


Рис. 73

Мы ожидаем увидеть текущее значение 50, а затем следующее значение 60, но это не так. Не имеет значения, в каком порядке указывают выражения в предложении *SELECT*, потому что выбираются они в одно и то же время.

> Внимание! Если во время сеанса генерируется новый номер последовательности, то повторно использовать его может только этот же сеанс. Другие сеансы с помощью той же последовательности получают свои номера.

Псевдостолбцы выполняют в таблице функции обычных столбцов. Инструкции SQL могут ссылаться на псевдостолбцы для извлечения данных, но не могут вводить в них данные, изменять или удалять данные.

> Внимание! Одной из новинок Oracle 12c стала возможность использовать значения *CURRVAL* и *NEXTVAL* в предложении *DEFAULT* для столбца в инструкциях *CREATE TABLE* и *ALTER TABLE*.

Синонимы. *Синонимы* – псевдонимы или альтернативные имена объектов схемы (таблиц, представлений, последовательностей и др.). Когда приложение использует синоним, Oracle передает запрос связанному с ним базовому объекту. Синонимы используют для того, чтобы упростить длинное или непонятное имя объекта. Они также позволяют скрыть истинную схему и имя объекта, усложняя несанкционированный доступ к нему.

Oracle позволяет создавать *частные (private) общедоступные (public)* синонимы. Частные синонимы действуют внутри конкретной схемы и используются разработчиком для сокрытия истинных имен объектов схемы (таблиц, представлений, хранимых процедур и т. д.). Общедоступные синонимы доступны любому пользователю и используются для маскировки имен системных структур баз данных, предопределенных пакетов PL/SQL и т. д. В многопользовательских базах данных разработчикам редко дают права создания общедоступных синонимов.

Внимание! Синонимы не позволяют обходить правила безопасности – пользователь может получить доступ к объекту через синоним, если имеет привилегии доступа к самому объекту. Это касается и частных, и общедоступных синонимов.

Синонимы обеспечивают удобство. Они не дают дополнительных привилегий и не угрожают безопасности. Они экономят ввод текста и позволяют сделать приложения независимыми от схемы. Но синонимы могут вызвать проблемы с производительностью (проблемы возникают из-за общедоступных синонимов).

Практическая работа

При выполнении работы необходимо в базе данных Oracle:

- создать в схеме *Bookshop* несколько последовательностей, в том числе для связанных таблиц, и продемонстрировать работу с ними;
- создать в схеме *Bookshop* несколько частных синонимов.

Пример выполнения задания

Создание последовательностей для таблиц схемы

1. Щелкните правой кнопкой мыши по узлу *Sequence* в иерархии схемы в навигаторе и выберите *New Sequence*... .

2. В диалоговом окне *Create Sequence* в поле *Name* задайте *order_ids*.

3. В таблице *Orders* уже есть записи с номерами 1001, 1002, 1003, 1004 и 1005, поэтому в поле *Start with* задайте значение 1006.

4. Для номеров заказов мы хотим получить возрастающий ряд значений с шагом +1. Оставьте поля *Min Value* и *Max Value* пустыми, затем задайте значение 1 для шага в поле *Increment*. Опцию *Cycle* выбирать не следует.

5. Для повышения производительности Oracle может предварительно генерировать и заносить в кэш упорядоченные значения номеров последовательности. Эту опцию мы не используем, поэтому выберите в поле *Cache* значение *NOCACHE* и не выбирайте опцию *Order*.

6. Откройте закладку *DDL* для просмотра инструкции *CREATE SEQUENCE*.

7. Щелкните мышью на кнопке *OK* (рис. 74).

Аналогично создаются последовательности для других таблиц схемы. Ускорить процесс можно, выполнив в окне *Worksheet* следующий сценарий:

```
CREATE SEQUENCE catalog_ids INCREMENT BY 1
   START WITH 6 NOCACHE;
CREATE SEQUENCE book_ids INCREMENT BY 1
   START WITH 31 NOCACHE;
CREATE SEQUENCE customer_ids INCREMENT BY 1
```

START WITH 7 NOCACHE;
CREATE SEQUENCE order_ids INCREMENT BY 1
START WITH 1006 NOCACHE;
CREATE SEQUENCE item_ids INCREMENT BY 1
START WITH 10 NOCACHE;

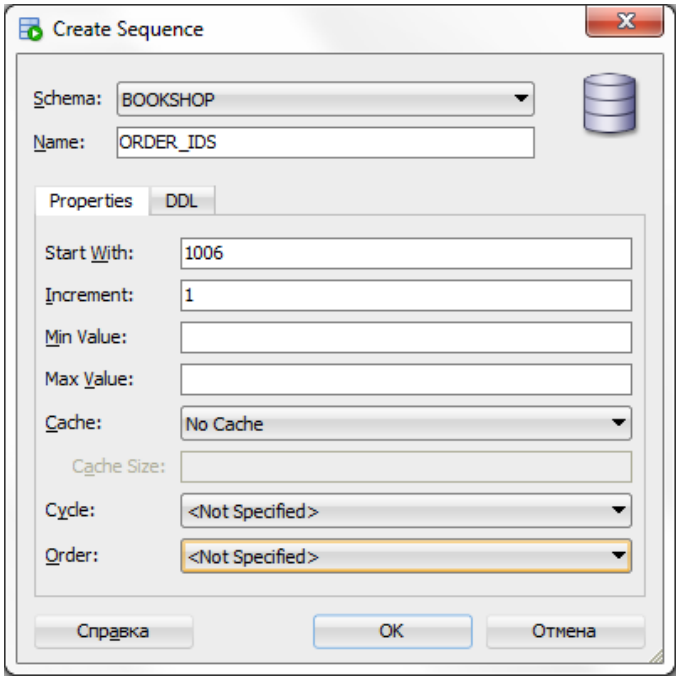


Рис. 74

Генерирование и повторное использование номера последовательности

Создайте в текущем сеансе нового покупателя и разместите новый заказ этого покупателя. Для этого в окне *Worksheet*:

1. Введите команду *INSERT*, вставляющую данные нового покупателя в таблицу *Customers*, и зафиксируйте транзакцию (уникальный код покупателя генерирует последовательность *customer_ids*) (рис. 75).

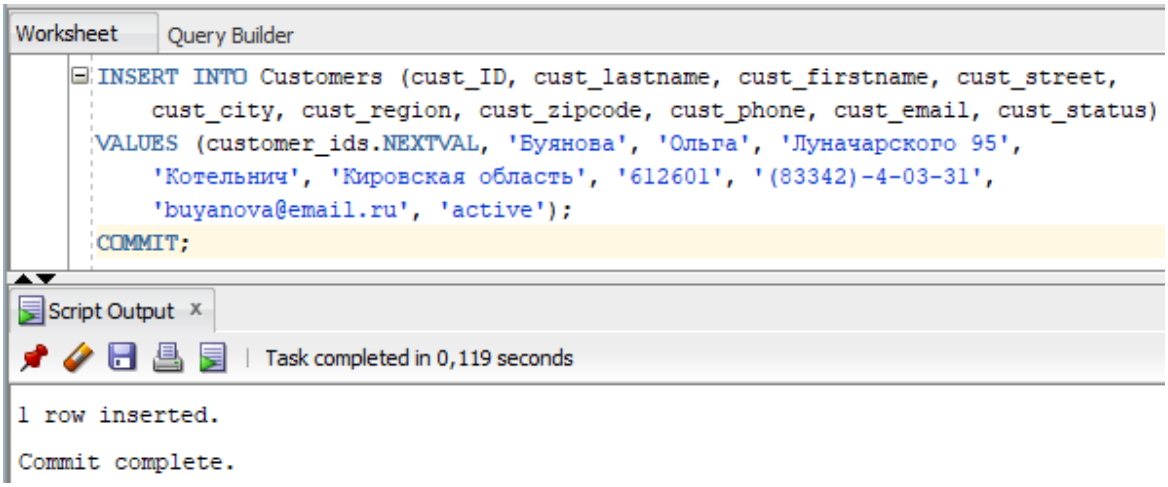


Рис. 75

2. Введите команду *INSERT*, вставляющую заказ этого покупателя в таблицу *Orders*, и зафиксируйте транзакцию (уникальный код заказа генерирует последовательность *order_ids*, функция *SYSDATE* возвращает системную дату) (рис. 76).

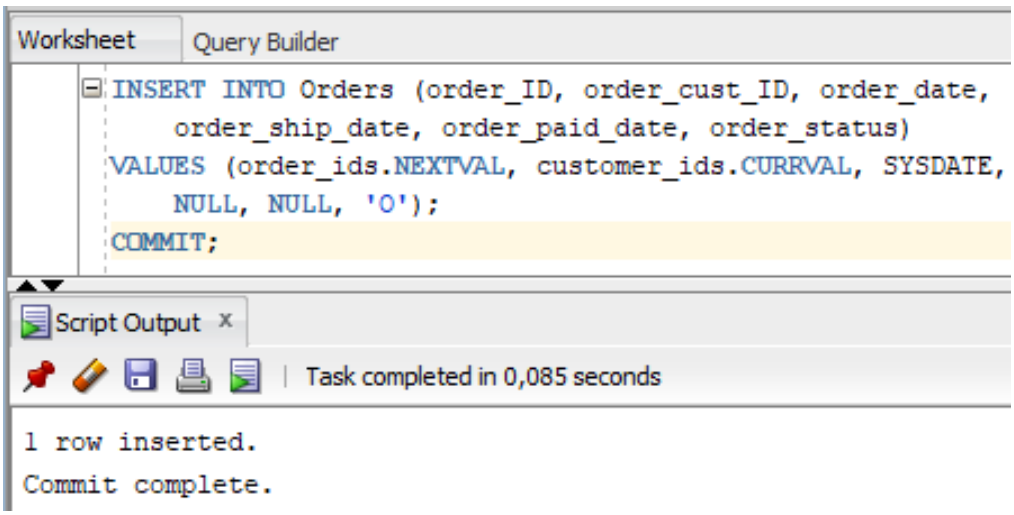


Рис. 76

3. Введите две команды *INSERT*, вставляющие записи о заказываемых товарах в таблицу *ITEMS* для текущего заказа, а затем зафиксируйте транзакцию (рис. 77).

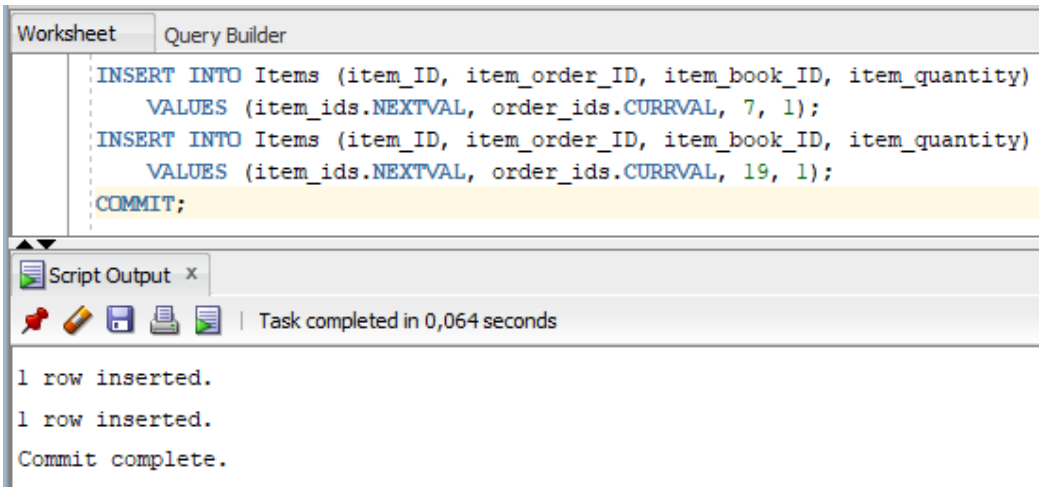


Рис. 77

Внимание! Для получения корректных результатов все действия должны выполняться в рамках одного сеанса.

Создание и использование частных синонимов

Для создания частного синонима *Cust* как псевдонима для таблицы *Customers* в схеме *Bookshop* выполните следующие действия:

1. Щелкните правой кнопкой мыши по узлу *Synonyms* в иерархии схемы в навигаторе и выберите *New Synonym*... .

2. В диалоговом окне *New Synonym* в поле *Synonym Name* задайте *Cust*, в раскрывающемся списке *Object Owner* выберите *Bookshop*, в раскрывающемся списке *Object Name* выберите *Customers* (рис. 78).

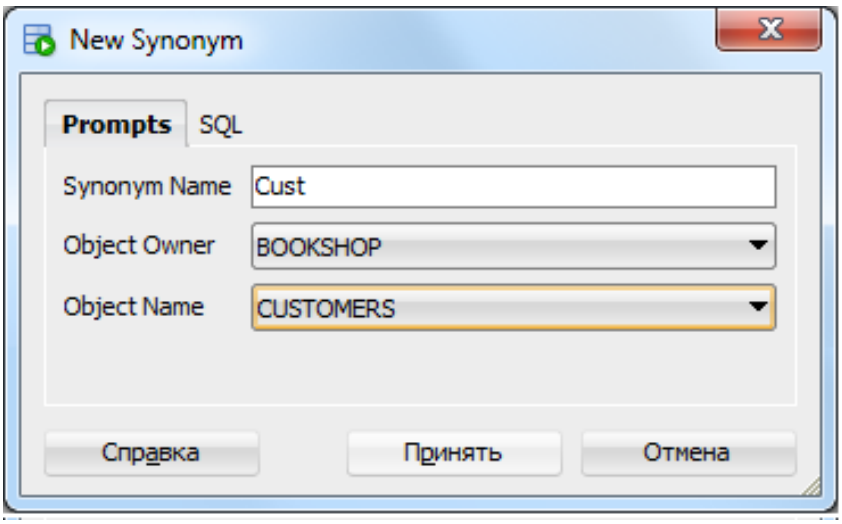


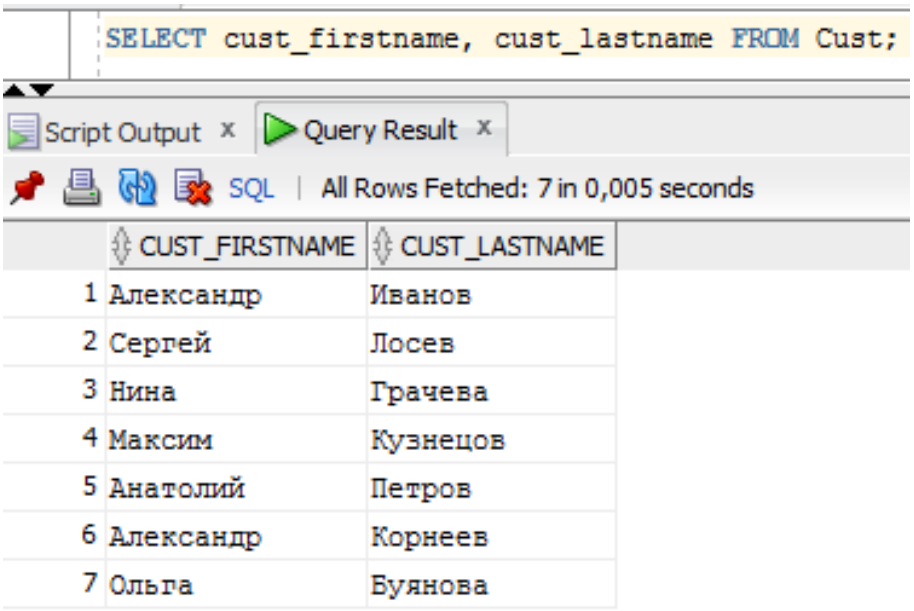
Рис. 78

3. Щелкните мышью на закладке *SQL* для просмотра автоматически сгенерированной SQL-команды, использованной для создания синонима:

CREATE SYNONYM "BOOKSHOP".Cust FOR "BOOKSHOP"."CUSTOMERS"

4. Щелкните мышью на кнопке *Принять*.

Теперь можно указывать синоним везде, где есть ссылка на соответствующий объект. В окне *Worksheet* введите запрос, использующий синоним *Cust* (рис. 79).



	CUST_FIRSTNAME	CUST_LASTNAME
1	Александр	Иванов
2	Сергей	Лосев
3	Нина	Грачева
4	Максим	Кузнецов
5	Анатолий	Петров
6	Александр	Корнеев
7	Ольга	Буянова

Рис. 79

Практическое занятие № 7
Работа с индексами

Теоретические сведения

Применение индексов также позволяет сделать работу со схемой более эффективной. Производительность приложений баз данных зависит от того, насколько быстро они получают доступ к данным. Определяющий фактор – количество операций дискового ввода-вывода. Сократить дисковый ввод-вывод позволяет индексирование таблиц. *Индекс* – физическая структура базы данных, занимающая пространство на диске и используемая СУБД для повышения производительности запросов. Пользователи, имеющие привилегии, могут создавать и удалять индексы, при этом обычные инструкции SQL никогда явно к индексам не обращаются. Oracle автоматически синхронизирует индекс с таблицей.

Стандарты SQL не упоминают об индексах, но все производители СУБД предлагают свои механизмы создания индексов. Наличие индекса в таблице не обязательно. Если индекс есть, то Oracle автоматически использует его для быстрого поиска нужной строки. Пусть для таблицы *Books* выполняется запрос:

```
SELECT * FROM Books
WHERE book_cat_ID = 3;
```

В первую очередь СУБД проверяет, существует ли индекс по столбцу *book_cat_ID*. Если индекс есть, то он используется для определения физического местоположения нужных строк (строки, где *book_cat_ID = 3*). Если индекса нет, то для поиска нужных строк выполняется сканирование всей таблицы.

Хотя индексы могут существенно повысить производительность запросов, индексировать каждый столбец не имеет смысла. Излишние индексы могут замедлить работу OLTP-систем и загрузку больших объемов данных в таблицы.

Регулярные индексы. Индексом по умолчанию в Oracle является *индекс на основе B-дерева (нормальный индекс)*. Он представляет собой упорядоченный набор индексных узлов, каждый из которых содержит одну или несколько индексных записей. Каждая индексная запись соответствует строке в таблице и содержит два элемента:

- значение индексированного столбца (столбцов) для данной строки;
- адрес *RowID* (место физического расположения) данной строки.

Если индексированный столбец в строке имеет значение *NULL*, то эта строка в индекс не включается. Oracle спускается по дереву индексных узлов в поиске значения индекса, соответствующего запросу. Обнаружив такое значение, Oracle использует адрес *RowID* для нахождения и считывания строки с диска.

Нормальные индексы лучше всего работают с ключевыми столбцами, содержащими много различных значений (первичный и альтернативные ключи табли-

цы). Oracle автоматически создает нормальные индексы для всего первичного ключа и уникальных ограничений целостности таблицы (либо автоматически использует имеющиеся для таких ограничений уникальные индексы).

Уникальные индексы неявно создаются для столбцов с ограничениями *UNIQUE* или *PRIMARY KEY*, в которых не допускается повторения значений. *Неуникальные индексы* создаются по столбцу (группе столбцов) без ограничений на дублирование значений.

Внимание! Хотя уникальные индексы можно создавать явно, Oracle не рекомендует это делать. Вместо этого следует использовать ограничения уникальности *PRIMARY KEY* и *UNIQUE*, оставив выбор физической реализации этих ограничений на усмотрение Oracle.

Индексы могут создаваться по одному столбцу или по комбинации столбцов. Последний подход полезен, когда в условии *WHERE* часто встречается комбинация столбцов. Например, если часто ищут заказы, размещенные конкретным клиентом в заданный день, можно создать неуникальный индекс по комбинации столбцов *order_cust_ID* и *order_date* таблицы *Orders*.

Специальные типы индексов. Oracle предоставляет несколько специальных типов индексов для специфических нужд.

Битовые индексы (Bitmap Indexes). Регулярные индексы хорошо работают для столбцов, содержащих много разных значений (наилучшую избирательность обеспечивают уникальные индексы, содержащие только разные значения). Однако существуют столбцы с низкой кардинальностью, содержащие только несколько значений (пол, статус, столбцы «Да/Нет»). Для них регулярный индекс – плохое решение (средняя избирательность поиска равенства будет плохой). Для столбцов с низкой кардинальностью Oracle поддерживает битовые индексы, которые создаются с указанием опции *BITMAP*. Битовые индексы могут превосходить обычные индексы, если предложение *WHERE* является сложным и использует много связок *AND*, *OR* и *NOT*.

Внимание! Индексы замедляют обработку данных, а битовые индексы являются наиболее дорогим типом индекса с точки зрения обслуживания. Не создавайте битовые индексы для таблиц с большим количеством операций DML.

Индексы на основе функций. В определении индекса вместо указания одного столбца или списка столбцов, разделенных запятыми, можно указать более сложное выражение. Индексы, содержащие такие выражения, называются *индексами*

на основе функций (Function-based Indexes). Индексы, основанные на функциях, можно использовать в сочетании с различными функциями NLS для лингвистической сортировки и поиска.

Внимание! При работе с индексами помните, что, хотя решение о существовании индекса принимает разработчик, решение об использовании индекса принимает оптимизатор Oracle. Оптимизатор выбирает план выполнения для каждой инструкции SQL.

Управление индексами. Поскольку индексы поддерживаются СУБД Oracle, все изменения таблиц немедленно распространяются на индексы. Индексы всегда актуальны. Тем не менее, если таблицы подвергаются непрерывному воздействию инструкций DML, следует подумать о перестройке индексов. Можно просто удалить, а затем воссоздать индексы. Однако более эффективно использование инструкций *ALTER INDEX ... REBUILD* или *ALTER INDEX ... COALESCE*. Удалить индексы можно с помощью команды *DROP INDEX*.

Узнать, использует ли оптимизатор ваши индексы, можно, просмотрев план выполнения инструкции SQL. В п. 1.2 упоминалось, что SQL Developer предлагает небольшой, но удобный набор диагностических инструментов: план выполнения *(Explain Plan)* и автотрассировку *(Autotrace)*. Они позволяют увидеть путь выполнения запроса и определить, использует ли запрос индексы.

Рекомендации по созданию индексов. Универсального правила не существует, но есть рекомендации, которые следует учитывать:

- не стоит создавать индексы в небольших таблицах, так как в этом случае они снижают производительность (в таблице с 50 строками лучше выполнить полное сканирование, чем использовать алгоритм B-дерева);
- в больших таблицах индексы следует создавать только там, где запросы, их использующие, отбирают небольшой процент строк (не более 15 %);
- рекомендуется создавать индексы по столбцам, участвующим в запросах с объединениями (первичные ключи и уникальные столбцы индексируются по умолчанию, но лучше создать индексы и по столбцам внешних ключей);
- для часто обновляемой таблицы рекомендуется создавать как можно меньше индексов. При обновлении индексированного столбца также обновляется индекс. При вставке строки будет заново выполнена балансировка B-дерева.

Таким образом, наилучшими для индексирования будут:

- столбцы внешнего ключа;
- столбцы, часто используемые в предложениях *WHERE*;
- столбцы, часто используемые в предложениях *ORDER BY* и *GROUP BY*.

Практическая работа

При выполнении работы необходимо в базе данных Oracle:

- создать в схеме *Bookshop* нормальный индекс;
- создать в схеме *Bookshop* битовый индекс;
- создать в схеме *Bookshop* индекс на основе функции.

Пример выполнения работы

Создание нормального индекса

Ускорить выполнение запросов, использующих соединение таблиц *Orders* и *Customers*, позволит нормальный индекс для столбца внешнего ключа *order_cust_ID* таблицы *Orders*:

1. Щелкните правой кнопкой мыши по узлу *Indexes* в иерархии схемы в навигаторе и выберите *New Index...* .

2. В диалоговом окне *Create Index* в поле *Name* задайте *cust_idx1*, в раскрывающемся списке *Table* выберите *Orders*. Внешний ключ в столбце *order_cust_ID* не гарантирует уникальное значение для каждой строки, поэтому в раскрывающемся списке *Index Type* выберите *Non-Unique*.

3. Задайте столбец для индекса. Щелкните зеленый крестик (*Add Index Expression*), в окне *Expressions* появится имя столбца *order_cust_ID*. Oracle XE по умолчанию сортирует значения столбца в индексе в порядке возрастания, однако в поле *Order* порядок сортировки можно изменить (рис. 80).

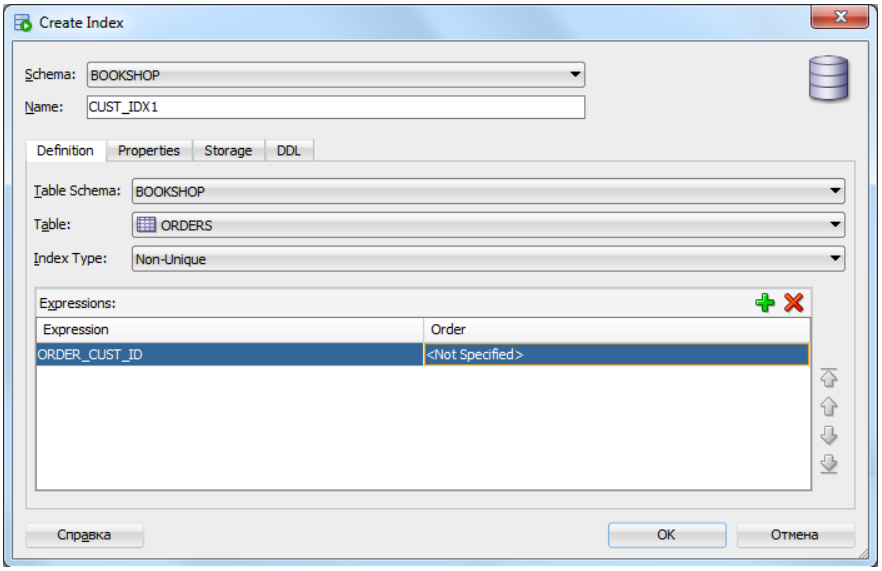


Рис. 80

4. Щелкните мышью на кнопке *DDL* для просмотра автоматически сгенерированной SQL-команды *CREATE INDEX*, использованной для создания индекса:

CREATE INDEX cust_idx1 ON Orders (order_cust_ID);

5. Щелкните на кнопке *OK*.

Панель *Сведения (Details)* отображает информацию о созданном индексе. Раскрыв узел *Indexes* в навигаторе, можно увидеть другие индексы, которые Oracle создала автоматически (в частности, для ограничений первичных ключей и ограничений уникальности, объявленных для разных таблиц схемы *Bookshop*).

Создание битового индекса

В соответствии со сказанным выше можно создать битовый индекс по столбцу *order_status* таблицы *Orders*. Это – столбец с низкой кардинальностью, содержащий несколько возможных значений *('F', 'B', 'O')* (рис. 81).

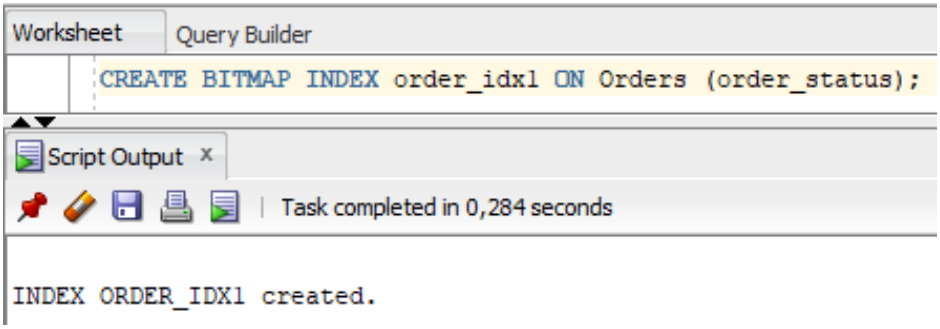


Рис. 81

Этот индекс используется Oracle при выполнении запроса, показанного на рис. 82.

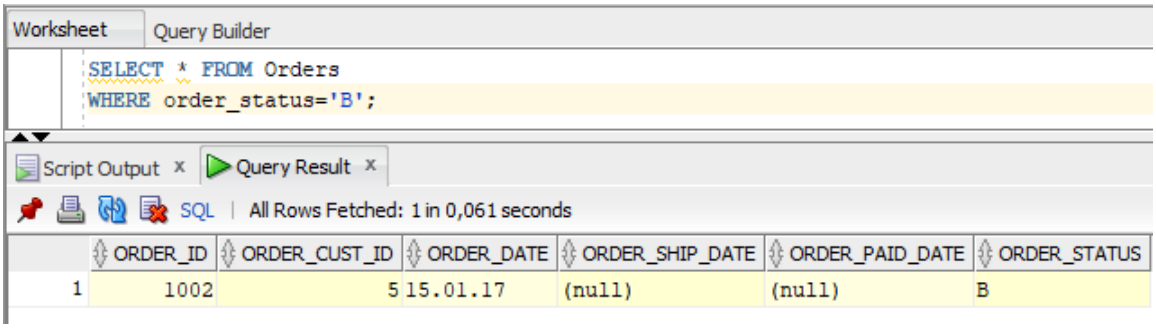


Рис. 82

Использование индекса подтверждает план выполнения запроса (рис. 83).

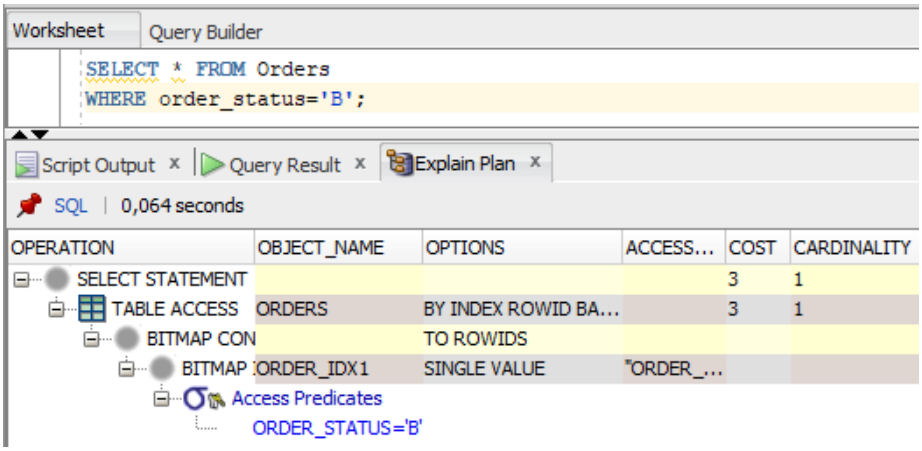


Рис. 83

Создание индекса на основе функции

Для таблицы *Books* можно создать индекс на основе выражения, вычисляющего суммарную стоимость книг, имеющихся на складе в интернет-магазине (рис. 84).

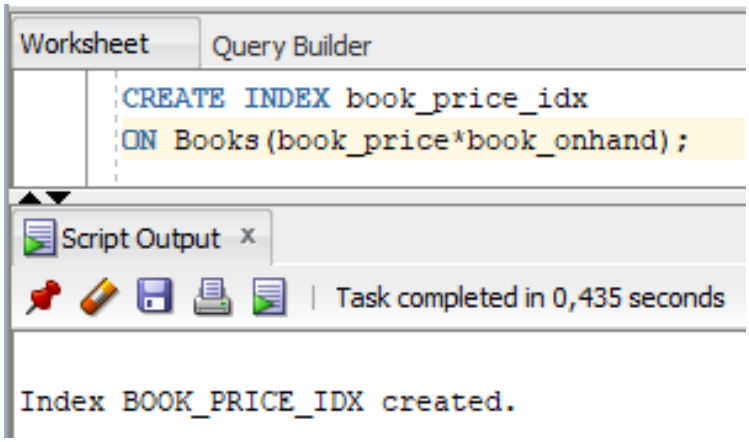


Рис. 84

Этот индекс обеспечит эффективное выполнение следующего запроса (рис. 85).

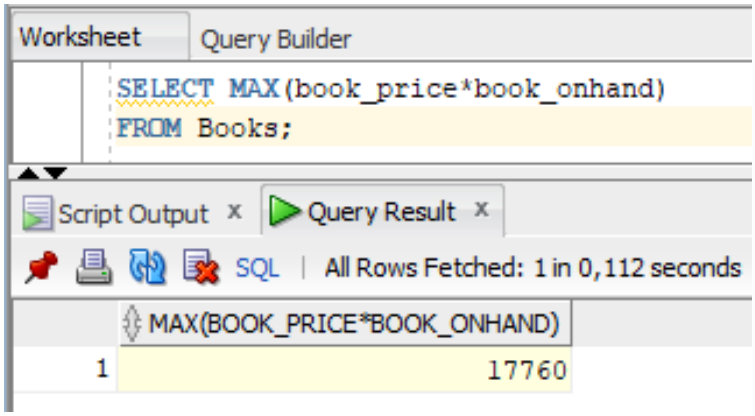


Рис. 85

План выполнения этого запроса подтверждает, что при его выполнении используется обращение к созданному выше индексу *book_price_idx* (рис. 86).

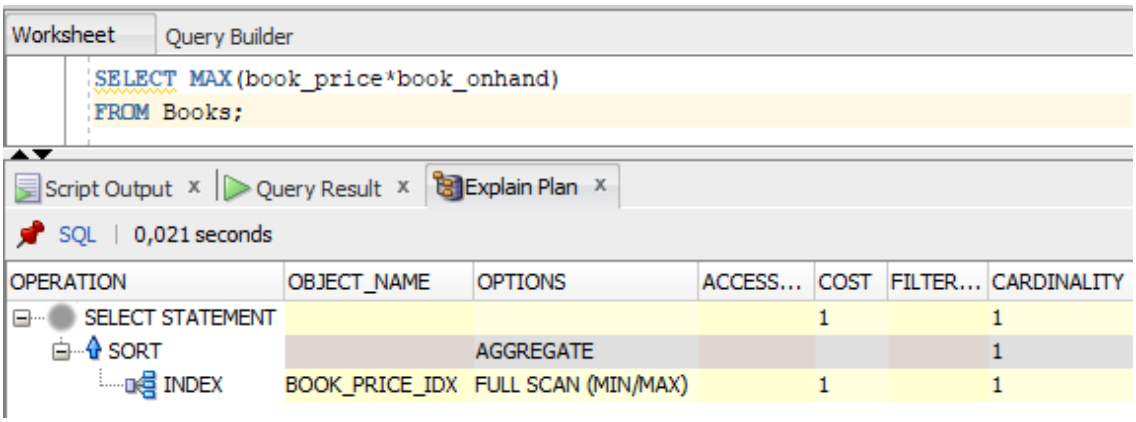


Рис. 86

Практическое занятие № 8
Работа с представлениями

Теоретические сведения

Оптимизация использования схемы начинается с определения представлений для таблиц схемы. *Представление (View)* – объект базы данных, представляющий данные таблиц в удобном виде, точнее, это виртуальная таблица с результатом сохраненного запроса, который выводится при доступе к представлению.

Представление можно рассматривать как *таблицу*. Представления имеют столбцы, поэтому к ним можно выполнять запросы. С другой стороны, представление – *виртуальная* таблица. Представления не имеют строк. Каждый раз при доступе к «содержимому» представления Oracle извлекает запрос представления из словаря и использует его для создания виртуальной таблицы.

СУБД преобразует инструкции SQL для представления в действия с базовыми таблицами, поэтому представления зависят от изменений в структуре базовых таблиц. Например, представление перестанет работать, если удалить или переименовать столбцы базовых таблиц, на которые ссылается его определение.

Создание представлений. Представления создаются инструкцией *CREATE VIEW*. Контент представления *Books_3* всегда будет зависеть от таблицы *Books* (рис. 87).

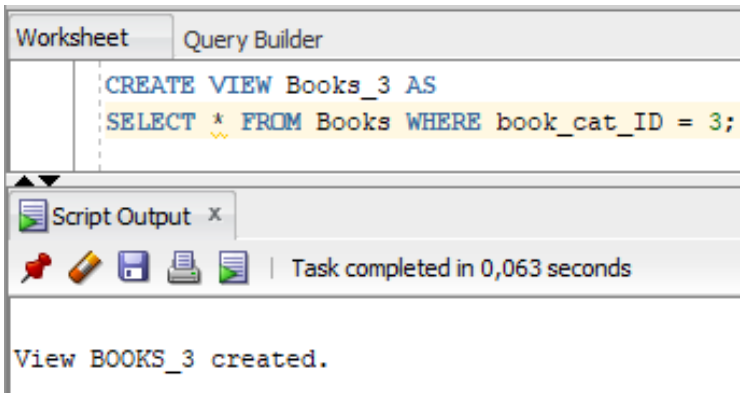


Рис. 87

Представления наследуют имена столбцов из определяющего запроса. При этом возможны осложнения (например, может отображаться результат запроса с одноименными столбцами). Решить эту проблему можно двумя способами:

- указать в предложении *SELECT* определяющего запроса псевдонимы столбцов;
- указать явные псевдонимы столбцов в команде *CREATE VIEW* между именем представления и словом *AS* (как показано на рис. 88).

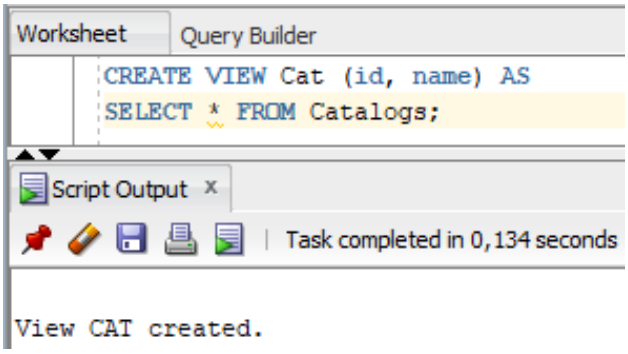


Рис. 88

Запрос к представлению осуществляется так же, как запрос к таблице (рис. 89).

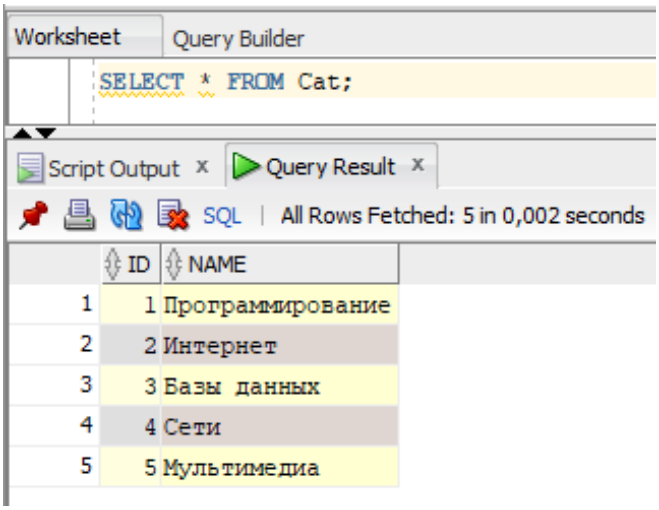


Рис. 89

Получить информацию о представлениях можно из словаря данных (рис. 90).

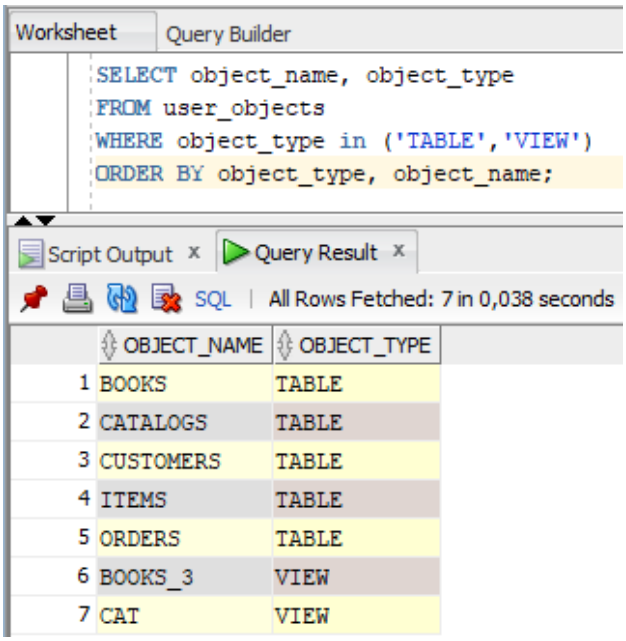


Рис. 90

Команду SQL*Plus *DESCRIBE* можно использовать для представлений, как и для обычных таблиц (рис. 91).

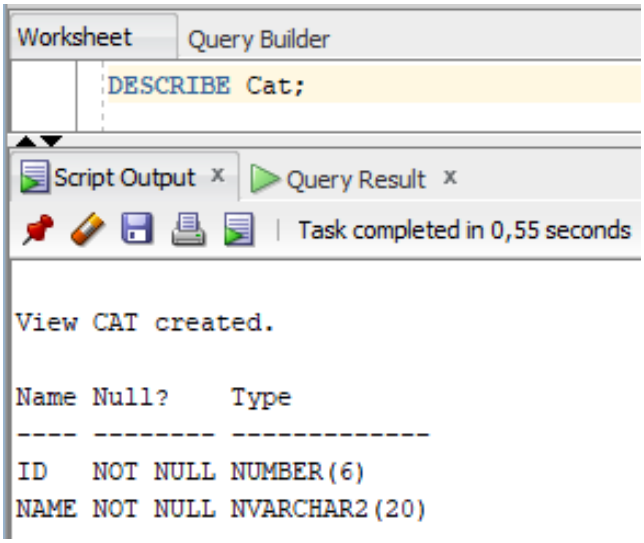


Рис. 91

Замена и удаление представлений. Изменить определение существующего представления нельзя. Oracle SQL предлагает команду *ALTER VIEW*, но ее можно использовать только для перекомпиляции представлений, ставших недействительными (например, после команды *ALTER* в базовой таблице). Можно только удалить определение представления с помощью команды *DROP VIEW*. Это подтверждает попытка применения команды контекстного меню *Edit…* к представлению *Cat*. Попробуем изменить запрос, лежащий в основе представления (рис. 92).

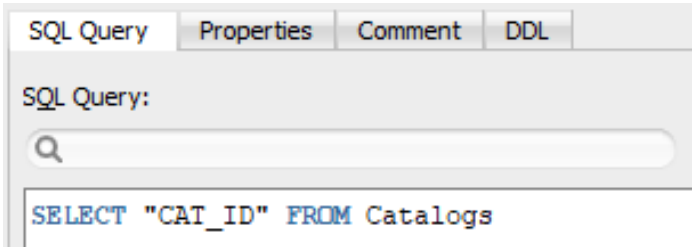


Рис. 92

Открыв закладку *DDL*, увидим, что создается новое представление *Cat* (рис. 93).

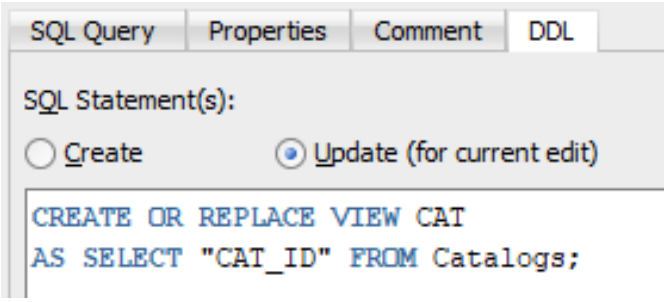


Рис. 93

Использование представлений. Возможности представления определяются запросом, лежащим в основе представления:

- простое представление может отображать все строки и столбцы таблицы, скрывая ее истинное имя (например, представление *cust*, показывающее все записи клиентов из таблицы *Customers*);
- представление может отображать часть строк и столбцов таблицы (например, представление *cust_omsk*, отображающее только имена, фамилии и телефоны клиентов из таблицы *Customers*, проживающих в Омске);
- сложное представление может отображать данные связанных таблиц (например, представление *catalogs_books* для таблиц *Catalogs* и *Books*);
- представление может отображать производные данные, которых нет в таблицах (например, представление для таблицы *Customers* со столбцом *total*, в котором вычисляется количество заказов пользователя).

Наиболее важные цели использования представлений:

- упрощение поиска в базе данных;
- сохранение логической независимости данных;
- обеспечение безопасности данных.

Упрощение поиска данных. Представления позволяют тестировать сложные запросы шаг за шагом. В представлениях можно хранить часто повторяющиеся запросы. Можно определять представления на основе часто соединяемых таблиц или сложных конструкций *GROUP BY*.

Пусть нас интересуют все заказы, имеющие стоимость, большую чем стоимость среднего заказа. Рассмотрим этот запрос в несколько этапов. В качестве первого шага зададим вопрос: «Какова стоимость каждого заказа?» (рис. 94).

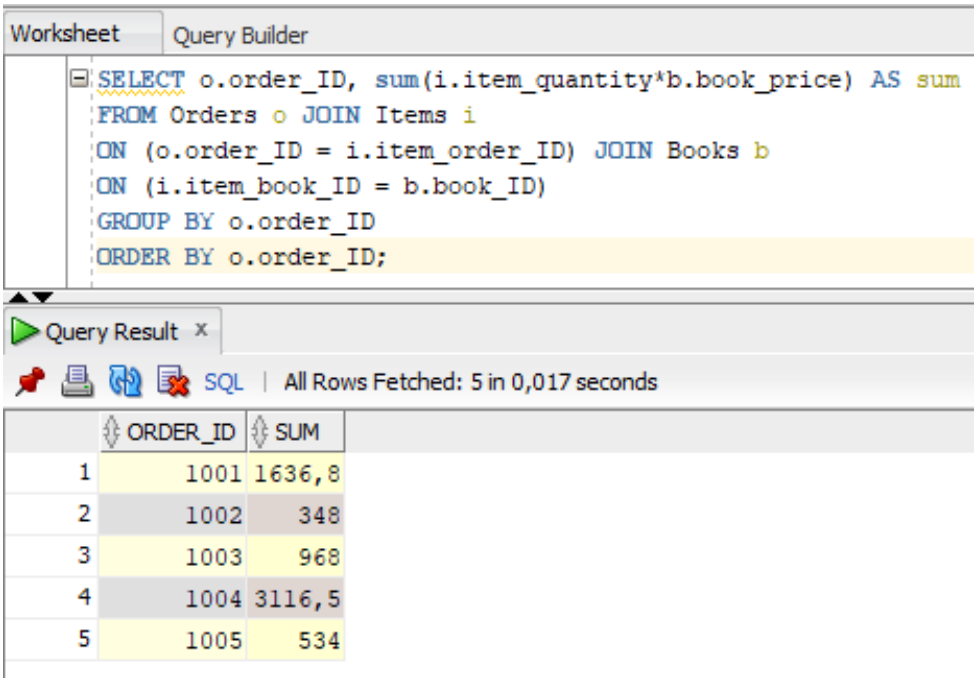


Рис. 94

Задачу было бы проще решить, если бы результат в листинге был настоящей таблицей. Мы можем моделировать ситуацию, определяя представление (рис. 95).

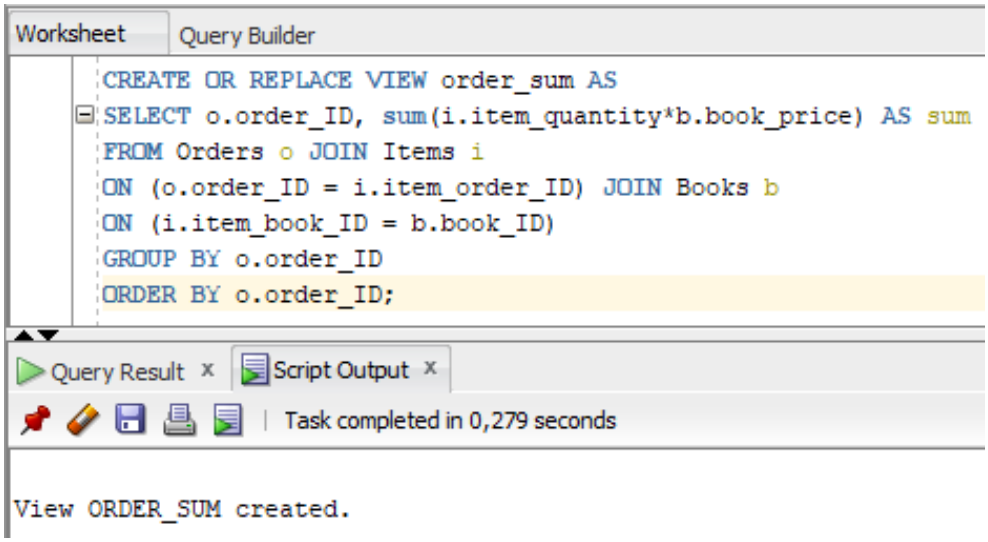


Рис. 95

С использованием представления *order_sum* задача решается просто (рис. 96).

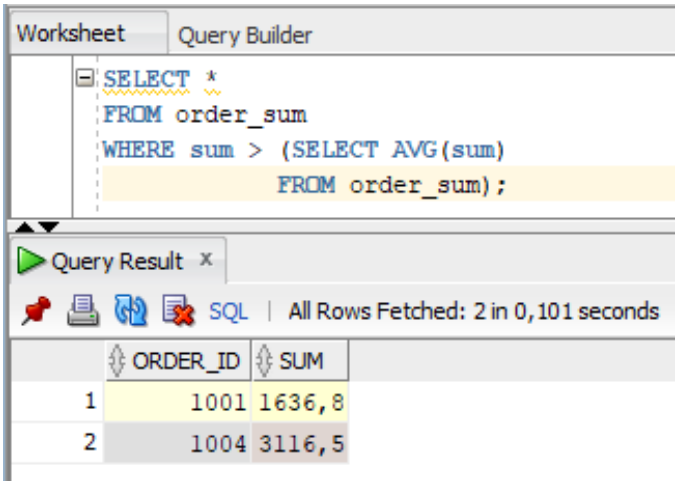


Рис. 96

Преимущество использования представлений в том, что определения представлений постоянны: одно и то же представление может оказаться полезным для нескольких проблем. Кроме того, представления занимают очень мало места (хранится только текст запроса) и избыточность отсутствует вообще.

Сохранение логической независимости данных. Представления можно использовать для изменения внешнего интерфейса базы данных, предоставляемого пользователям и приложениям. Разные пользователи могут иметь разные представления на одних и тех же базовых таблицах.

Распределенные базы данных часто используют представления для реализации логической независимости данных и сокрытия сложности. Можно определить

представление как «локальный» объект базы данных. При этом реально запрос представления будет обращаться к данным из других баз данных в сети.

Представления могут использоваться для отображения производной информации. Примером является рассмотренное выше представление *order_sum*, вычисляющее сумму каждого заказа.

Реализация защиты данных. Представления позволяют скрывать определенные данные от пользователей и приложений. Запрос представления точно определяет, какие предоставляются строки и столбцы. При этом пользователям вообще не предоставляется никаких прав доступа к базовым таблицам напрямую.

Типы представлений. Oracle поддерживает различные типы представлений:

- *Представления «только для чтения».* Используются только для извлечения данных, не позволяют вводить новые данные, изменять или удалять их.
- *Обновляемые представления.* Используются для ввода, изменения и удаления данных из таблиц. При этом Oracle должна корректно преобразовать операции представления в соответствующие операции с базовой таблицей. Если в списке запроса, определяющего представление, имеются виртуальные поля, то выполнить обновление строк базовой таблицы через представление можно, если инструкция *UPDATE* не ссылается на эти поля.

Можно создавать:

- *обновляемые представления с ограничениями* (с установкой опции *WITH CHECK OPTION)*, чтобы запретить некоторые операции ввода или изменения в представлении. Представление с ограничением не позволяет выполнить инструкции *INSERT* и *UPDATE*, создающие строки, которые определяющий представление запрос не смог бы внести в базовые таблицы;
- *обновляемые представления без ограничений* (без установки опции *WITH CHECK OPTION).*

Для экспериментов с представлениями создадим таблицу *Customers1*, как копию таблицы *Customers*. Рассмотрим простейшее представление *cust* (рис. 97).

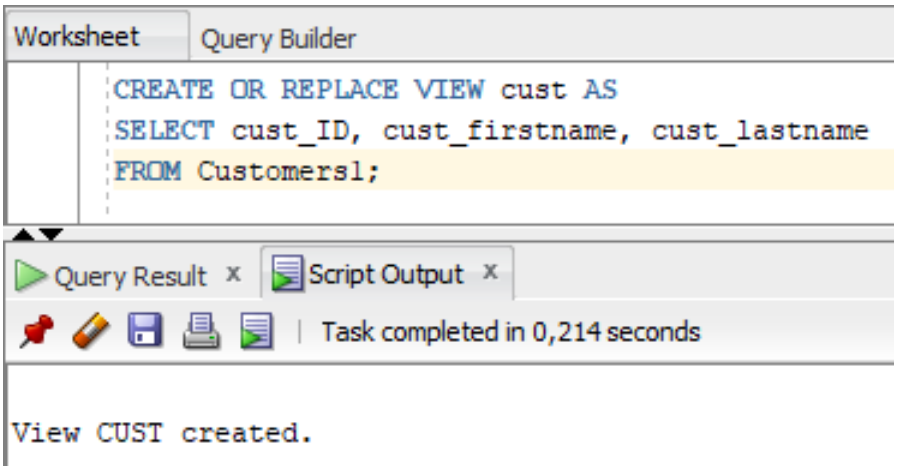


Рис. 97

Oracle должна иметь возможность удалять строки из таблицы *Customers1* с помощью этого представления или изменять любое из трех значений столбцов, предоставляемых представлением (здесь удаление строки, соответствующей покупателю с идентификатором 6) (рис. 98).

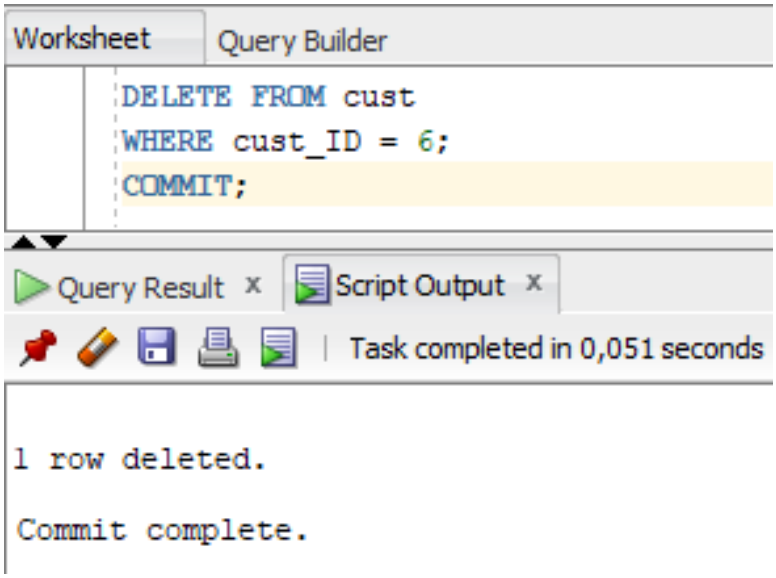


Рис. 98

Далее происходит изменение фамилии покупателя с идентификатором 5 (рис. 99).

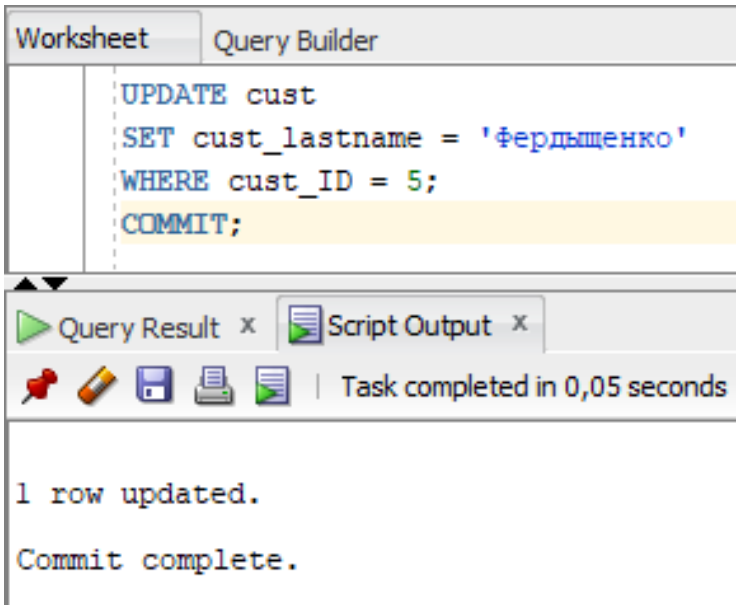


Рис. 99

Однако вставка новых строк через это представление невозможна, так как таблица *Customers1* не имеет столбцов *NULL* без значения по умолчанию вне области представления *cust* (рис. 100).

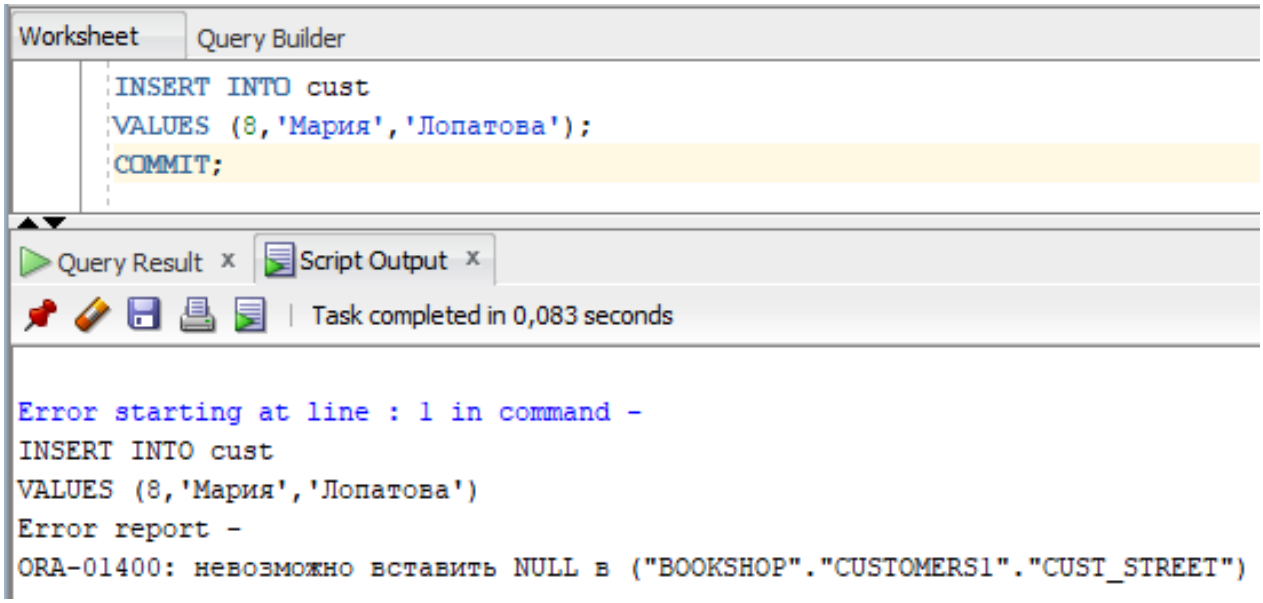


Рис. 100

Сообщение об ошибке ORA-01400 фактически раскрывает несколько фактов о базовой (и, возможно, скрытой) таблице *Customers1*: имя схемы (*BOOKSHOP*), имя таблицы (*Customers1*), наличие обязательного столбца *cust_street*.

Однако это сообщение об ошибке получает только пользователь, подключившийся как *BOOKSHOP*. Другой пользователь с привилегией *INSERT* только для представления *cust* получит сообщение *ORA-01400: cannot insert NULL into (???)*.

Обновляемые представления соединения. Можно манипулировать данными с помощью представления, основанного на соединении таблиц (если манипуляции могут быть преобразованы в действия с базовыми таблицами).

К обновляемым представлениям соединения применяются некоторые ограничения. Принципиальную роль играет схема *таблиц с одним сохраняемым ключом (single key-preserved table)*. Базовая таблица считается таблицей с сохраняемым ключом, если каждое значение первичного или уникального ключа в ней является уникальным в результирующей таблице представления с соединением – гарантируется, что одна запись в таблице соответствует ровно одной записи в представлении. Из нее выбирается *RowID*, и только ее столбцы можно изменять.

Представление с соединением, в котором объединяются данные двух и более таблиц, может быть обновляемым при следующих условиях [2]:

- для обновляемых представлений соединения разрешается выполнять инструкции DML только при изменении одной базовой таблицы;
- можно вставлять строки в представление с соединением без ограничений (без установки *WITH CHECK OPTION*), если инструкция *INSERT* обращается только к столбцам представления или таблицы с одним сохраняемым ключом;

- можно обновлять строки в представлении с соединением без ограничений, если инструкция *UPDATE* обращается только к столбцам представления или таблицы с одним сохраняемым ключом (в представлении с ограничениями инструкция *UPDATE* не может обновлять столбцы, используемые в условии соединения, и столбцы, на которые в представлении имеется более одной ссылки);
- для инструкций *DELETE*, если соединение приводит к нескольким таблицам с одним сохраняемым ключом, Oracle выполнит удаление из первой таблицы, названной в предложении *FROM*;
- если представление создано с параметром *WITH CHECK OPTION*, то применяются некоторые дополнительные ограничения DML.

Необновляемые представления. Если вы создаете представление с параметром *WITH READ ONLY*, то обработка данных с помощью этого представления уже невозможна по определению. Рассмотрим представление *avg_price*, позволяющее найти среднюю цену книг в каждом каталоге (рис. 101).

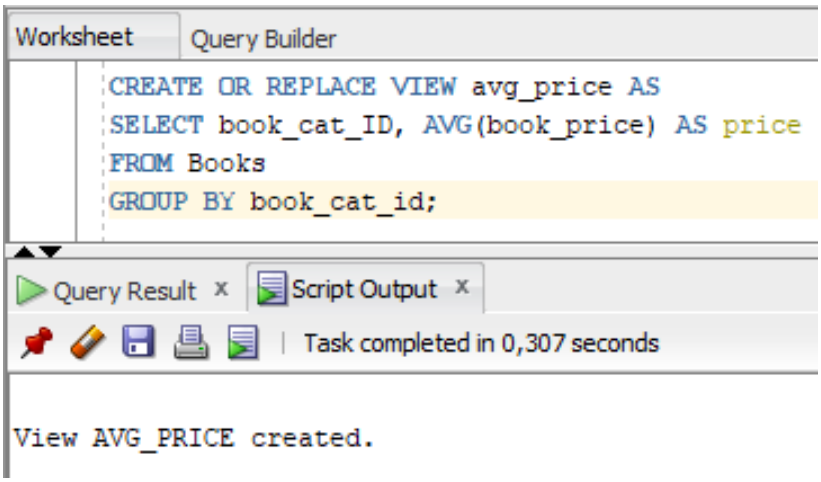


Рис. 101

Определение представления *avg_price* содержит предложение *GROUP BY*. Это означает, что между строками представления и строками базовой таблицы *Books* больше не существует связи «один-к-одному». Поэтому манипулирование данными через представление *avg_price* невозможно.

Использование *SELECT DISTINCT* в определении представления также делает представление необновляемым. Использование *SELECT DISTINCT* имеет и дополнительные недостатки: каждый доступ к представлению будет принудительно выполнять сортировку независимо от того, нужна она или нет.

Операторы *UNION*, *MINUS* и *INTERSECT* также приводят к необновляемым представлениям. Например, если вставлять строку через представление, основанное на объединении, то в какую из базовых таблиц СУБД должна ее вставить?

Словарь данных предлагает системное представление *USER_UPDATABLE_COLUMNS*, позволяющее узнать, какие из столбцов представления являются обновляемыми. Например, ничего нельзя сделать со столбцами необновляемого представления *avg_price* (рис. 102).

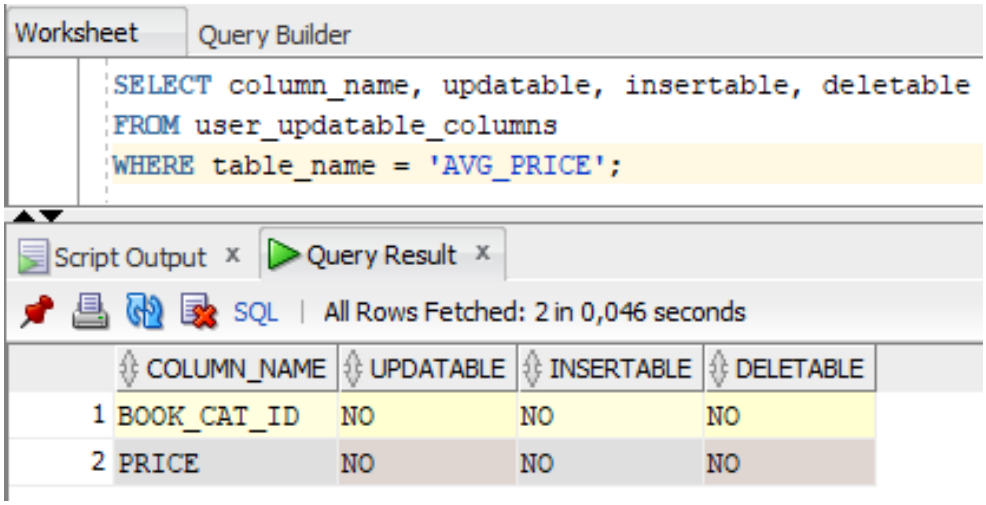


Рис. 102

С другой стороны, все столбцы представления *cust* можно вставлять, удалять и модифицировать (рис. 103).

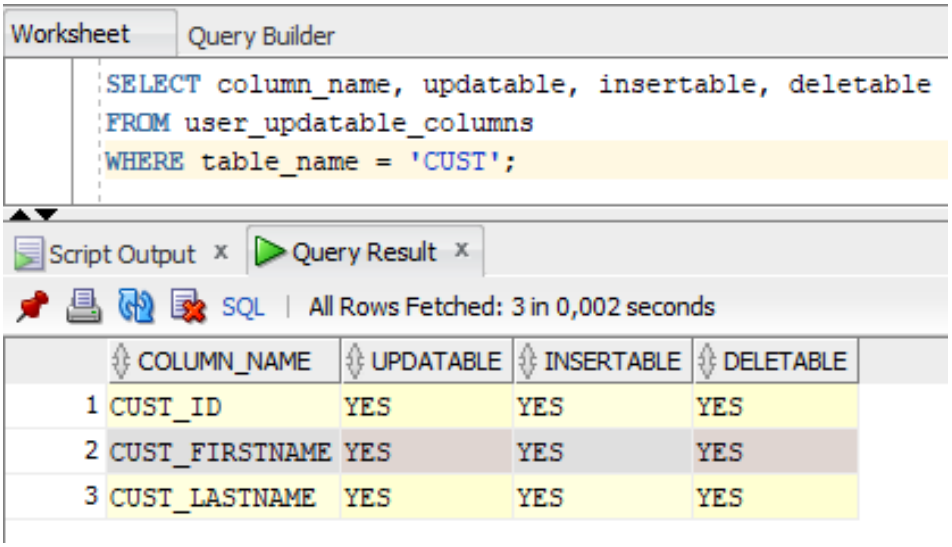


Рис. 103

Внимание! Процедурный язык PL/SQL позволяет сделать любое представление обновляемым. Он позволяет написать для представлений триггеры *INSTEAD-OF*, контролирующие все инструкции DML, касающиеся представления.

Материализованные представления. Используются в хранилищах с большими таблицами и редко изменяемыми данными, а также отдельным процессом извлечения, преобразования и загрузки. Кроме того, материализованные представления используются для репликации данных в распределенной базе данных.

Как следует из названия, материализованное представление – это представление, для которого хранится как его определение, так и результаты запроса. Материализованные представления имеют свои собственные строки, т. е. предполагают избыточное хранение данных.

Материализованное представление *Books_4* содержит все книги, относящиеся к каталогу 4 (рис. 104).

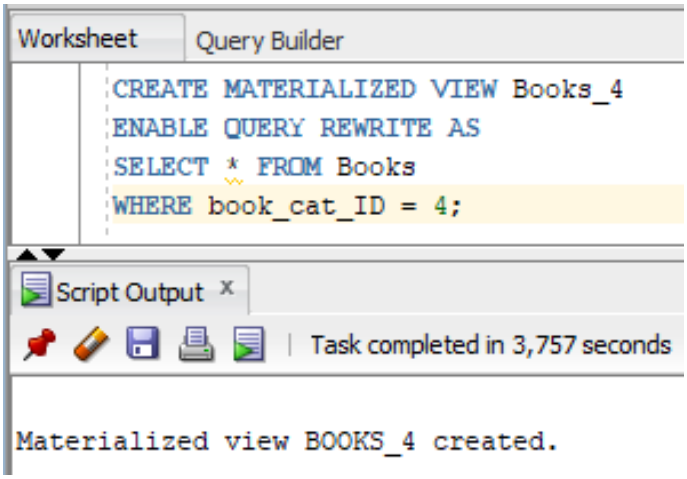


Рис. 104

Материализованные представления имеют два важных свойства:

- *обслуживание*: материализованные представления являются «моментальными снимками», т. е. они имеют в любой момент времени контент, взятый из обновленных базовых таблиц. При этом Oracle предлагает функции для полной и эффективной автоматизации обновления материализованных представлений;
- *использование*: оптимизатор Oracle, определяющий планы выполнения инструкций SQL, знает о существовании материализованных представлений. Он знает, актуальны они или устарели, и может использовать эти знания для замены запросов к базовым таблицам запросами к материализованным представлениям (функция перезаписи запроса).

При создании материализованных представлений обычно указывают, следует ли включить перезапись запросов, и как Oracle должна обрабатывать обновление материализованного представления.

Продолжим пример с материализованным представлением. Допустим, нам нужно выбрать из каталога 4 все книги стоимостью свыше 400 рублей:

*SELECT * FROM Books WHERE book_cat_ID = 4 AND book_price > 400;*

Исходный запрос к таблице *Books* переадресуется материализованному представлению *Books_4*. Оно содержит меньше строк, чем таблица *Books* (сканируется меньше строк), поэтому оптимизатор считает, что этот вариант лучше (рис. 105).

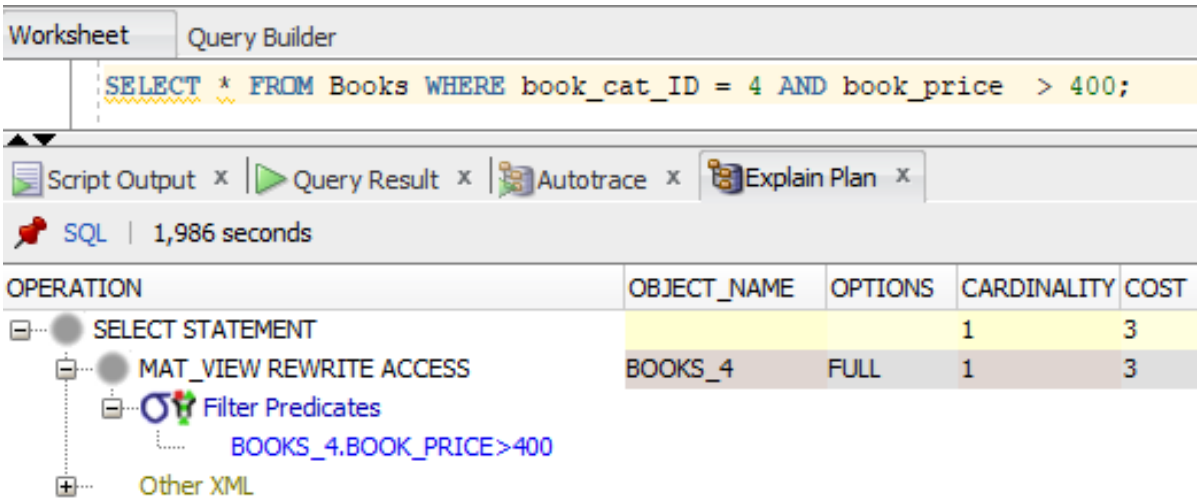


Рис. 105

Хотя запрос написан к таблице *Books*, план выполнения показывает, что вместо этого осуществляется доступ к материализованному представлению *Books_4*.

Материализованные представления обеспечивают лучшее время отклика, но результаты могут быть основаны на устаревших данных. Обычно материализованные представления создаются с трудоемкими операциями – агрегацией (*GROUP BY*), соединением (*JOIN*), операторами *UNION*, *MINUS* и *INTERSECT*.

Практическая работа

При выполнении работы необходимо в базе данных Oracle:

- создать в схеме *Bookshop* представление «только для чтения»;
- создать в схеме *Bookshop* обновляемое представление с ограничениями;
- создать в схеме *Bookshop* обновляемое представление без ограничений;
- создать в схеме *Bookshop* простое материализованное представление.

Пример выполнения работы

Создание представления «только для чтения»

Выполните следующие действия для создания представления *filled_orders*, отображающего выполненные и оплаченные заказы.

1. Щелкните правой кнопкой мыши по узлу *View* в иерархии схемы в навигаторе и выберите *New View...* .

2. В диалоговом окне *Create View* в поле *Name* задайте *filled_orders*.

3. В поле запроса *SQL Query* введите приведенный ниже запрос:

*SELECT * FROM Orders WHERE order_status = 'F' WITH READ ONLY*

(другой способ – открыть закладку *Properties* и в раскрывающемся списке *Query Restrictions* выбрать *Read Only*).

4. Откройте закладку *DDL* для просмотра инструкции *CREATE VIEW*, которую можно использовать для создания представления.

5. Щелкните мышью на кнопке *OK*.

> Внимание! К этому представлению можно обратиться с запросом (хотя в нем нет строк), но нельзя вставлять строки в базовую таблицу *Orders* через это представление, поскольку оно предназначено только для чтения.

Создание обновляемого представления с ограничениями

Выполните следующие действия для замены текущей версии представления *filled_orders* обновляемым представлением с ограничениями:

1. Найдите в иерархии схемы в навигаторе представление *filled_orders*, щелкните правой кнопкой мыши по нему и выберите *Edit...* .

2. В открывшемся диалоговом окне *Edit View* в поле запроса *SQL Query* введите запрос: *SELECT * FROM Orders WHERE order_status = 'F'*.

3. Откройте закладку *Properties* и в раскрывающемся списке *Query Restrictions* выберите *Check Option*.

4. Откройте закладку *DDL* для просмотра инструкции SQL:

*CREATE OR REPLACE VIEW filled_orders AS SELECT * FROM Orders WHERE order_status = 'F' WITH CHECK OPTION*

5. Щелкните мышью на кнопке *OK*.

Новую версию представления можно использовать для того, чтобы вставлять в таблицу *Orders* новые заказы, обновлять или удалять текущие заказы (это касается только заказов, у которых поле *order_status* имеет значение *'F'*) (рис. 106).

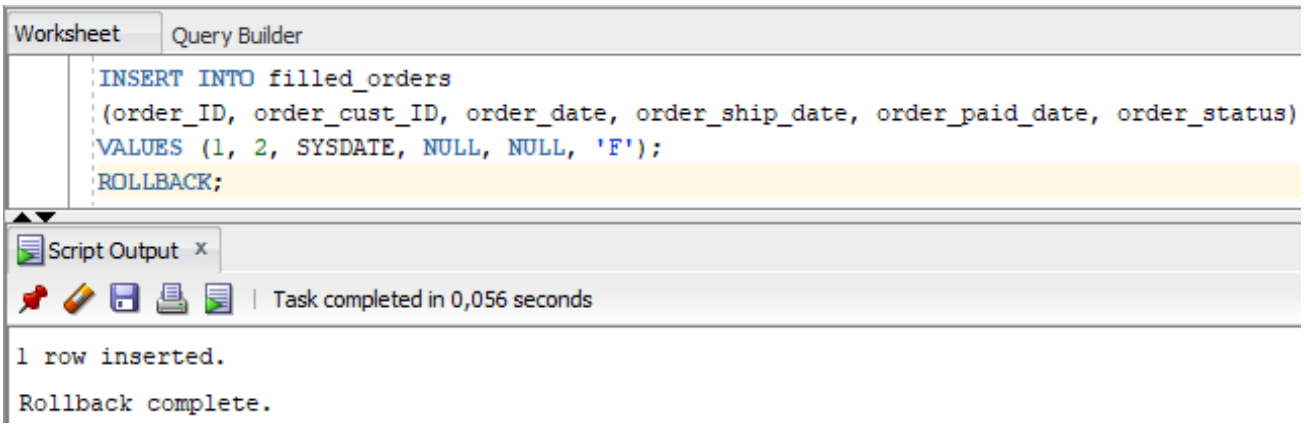


Рис. 106

Обновление заказа с номером 1001 и удаление всех заказов со статусом *'F'* представлено на рис. 107.

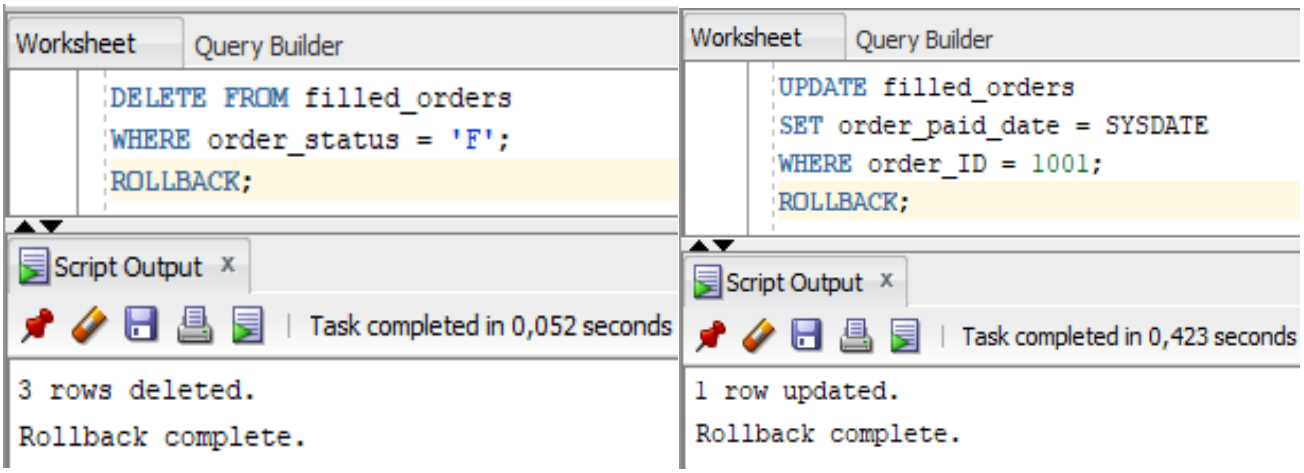


Рис. 107

Попытка выполнить следующий сценарий терпит неудачу – инструкция *INSERT* пытается вставить строку, которую нельзя реализовать в таблице *Orders* (рис. 108).

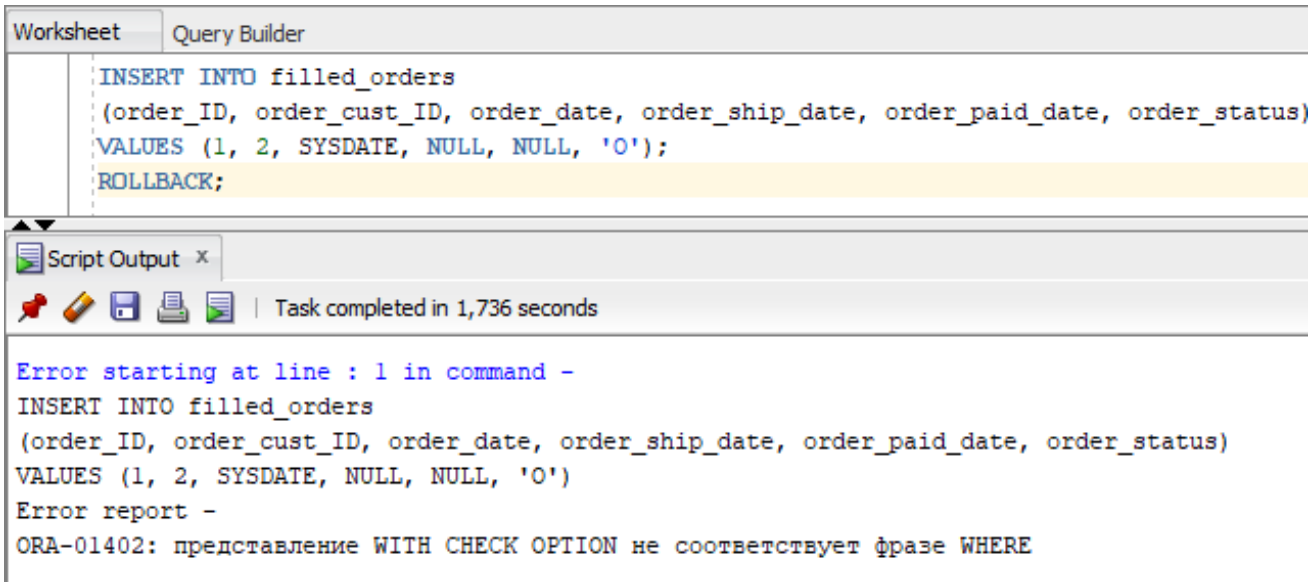


Рис. 108

Создание обновляемого представления с соединением без ограничений

Для создания обновляемого представления с соединением на основе таблиц *Orders* и *Customers* выполните следующие действия:

1. Щелкните правой кнопкой мыши по узлу *View* в иерархии схемы в навигаторе и выберите *New View...* .

2. В диалоговом окне *Create View* в поле *Name* введите *order_info*.

3. В поле запроса *SQL Query* введите приведенный ниже запрос (рис. 109).

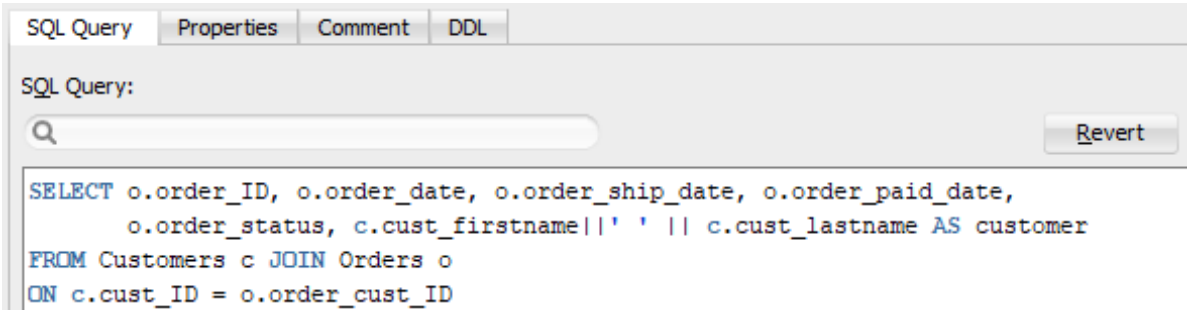


Рис. 109

4. Откройте закладку *Properties* и в раскрывающемся списке *Query Restrictions* выберите *No Restriction*.

5. Откройте закладку *DDL* для просмотра инструкции, которая будет использована для изменения представления.

6. Щелкните мышью на кнопке *OK*.

Используя окно *Worksheet*, вставьте два новых заказа в базовую таблицу *Orders*, а затем выполните запрос к представлению *order_info* (рис. 110).

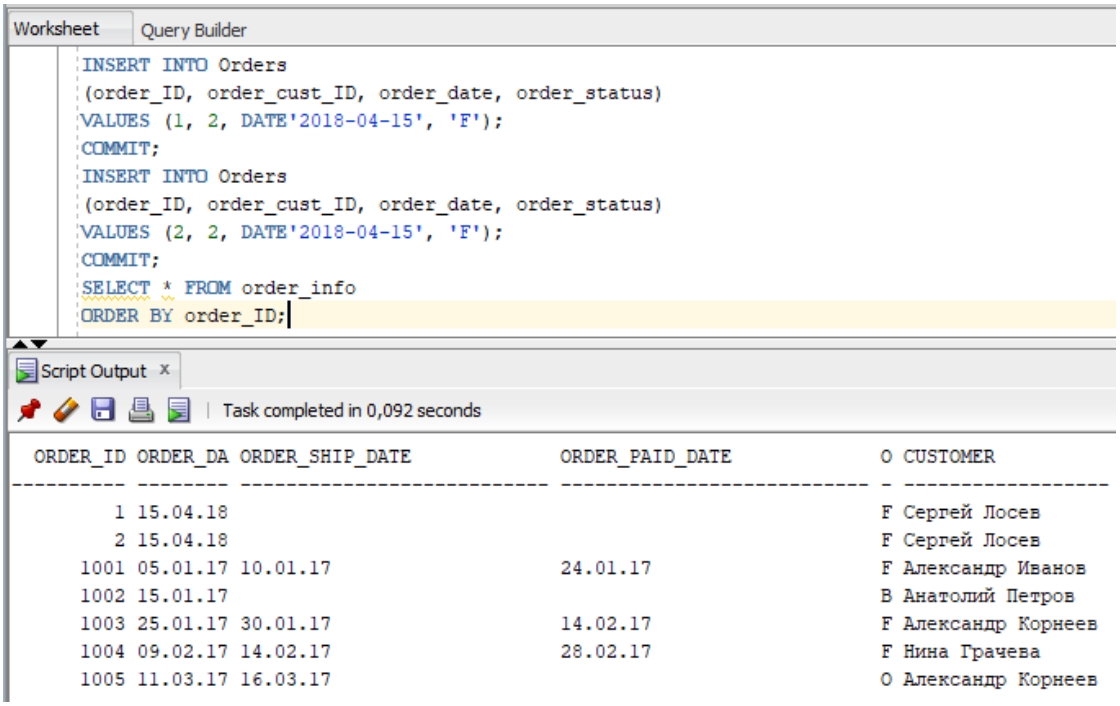


Рис. 110

Представление сохраняет целостность таблицы *Orders* (первичный ключ таблицы соответствует полю *order_ID* представления, таблица *Orders* является таблицей с сохраненным ключом). С другой стороны, представление не сохраняет целостность таблицы *Customers* (дублируемые клиенты в поле *customer*). Таблица *Customers* не является таблицей с сохраненным ключом.

ЗАКЛЮЧЕНИЕ

Oracle SQL, возможно, является наиболее важными и интересным диалектом языка SQL. Вместе с Oracle Database он широко используется во многих корпоративных средах и государственных структурах всего мира. Oracle SQL реализует множество функций, которых нет в конкурирующих продуктах.

Данное учебное пособие можно рассматривать как первый шаг в знакомстве и с самой СУБД Oracle Database, и с конкретным диалектом языка SQL, используемым этой СУБД. Для начального знакомства с Oracle Database выбрана свободно распространяемая Express-версия, которую можно скачать с сайта корпорации Oracle. Сведения, содержащиеся в пособии, позволяют установить эту версию на компьютере и выполнять минимально необходимые действия по ее администрированию. Студент получает навыки построения базовой реляционной схемы, содержащей необходимые таблицы и представления, а также создания объектов схемы (синонимов, последовательностей и индексов), применение которых позволяет сделать использование схемы более эффективным.

Учебное пособие использует подход, основанный на примерах, с подробными пояснениями. Большинство приведенных в пособии примеров реализованы в мощной графической среде разработки и отладки запросов Oracle SQL Developer. Практически без изменения эти примеры могут быть воспроизведены в утилите командной строки SQL*Plus, которая поставляется вместе с Oracle Database и является единственным инструментом, на который всегда может рассчитывать любой разработчик или администратор Oracle.

Выбор Express-версии ограничил рамки учебного пособия рассмотрением системы оперативной обработки транзакций OLTP (On-Line Transactional Processing), обеспечивающей ввод, обновление и удаление данных в реляционной базе. Но даже такое ограничение не дало возможности привести в рамках данного пособия многие важные моменты. В частности, за рамками пособия остались вопросы, связанные с использованием инструкции SELECT в Oracle SQL, а также с программированием доступа к базе данных на языке PL/SQL – процедурном расширении языка SQL, разработанном Oracle. Автор надеется, что программирование на PL/SQL, а также работа с курсорами, хранимыми процедурами и триггерами станет темой отдельного учебного пособия.

В конце учебного пособия приведен библиографический список, который безусловно будет полезен тем, кто желает более детально разобраться в рассматриваемых вопросах.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Алапати, Сэм. Р. Oracle Database 11g: руководство администратора баз данных : пер. с англ. / Сэм Р. Алапати. – М. : Вильямс, 2010. – 1440 с.

2. Бобровский С. Oracle Database XE для Windows. Эффективное использование : пер. с англ. / Стив Бобровский. – М. : Лори, 2009. – 486 с.

3. Гринвальд, Р. Oracle 11g. Основы : пер. с англ. / Р. Гринвальд, Р. Стаковьяк, Дж. Стерн. – 4-е изд. – СПб. : Символ-Плюс, 2009. – 464 с.

4. Кригель, Алекс. SQL. Библия пользователя : пер. с англ. / Алекс Кригель, Борис Трухнов. – 2-е изд. – М. : Вильямс, 2010. – 752 с.

5. Малков, О. Б. Работа с Transact-SQL [Электронный ресурс] / О. Б. Малков, М. В. Девятерикова. – Омск : Изд-во ОмГТУ, 2015. – 1 эл. опт. диск (DVD-ROM).

6. Пржиялковский, В. В. Введение в Oracle SQL / В. В. Пржиялковский. – 2-е изд. – М. : Национальный Открытый Университет «ИНТУИТ», 2016. – 358 с.

7. De Haan L., Gorman T., Jorgensen I., Caffrey M. Beginning Oracle SQL. For Oracle Database 12c. Third Edition. New York, Apress, 2014. 428 p.

8. Morton K., Osborne K., Sands R., Shamsudeen R., Still J. Pro Oracle SQL. New York, Apress, 2010. 575 p.

9. Oracle Database Express Edition. Getting Started Guide. 11g Release 2 (11.2). E18585-05. May 2014 [Электронный ресурс]. – Режим доступа: http://docs.oracle.com/cd/E17781_01/index.htm. – Загл. с экрана (дата обращения: 19.04.2019).

10. Oracle Database. 2 Day DBA. 18c. E83770-02. April 2018 [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/18/admqs/index.html – Загл. с экрана (дата обращения: 04.03.2019).

11. Oracle Database. Database 2 Day + Performance Tuning Guide. E83714-02. February 2019 [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/18/tdppt/index.html. – Загл. с экрана (дата обращения: 19.04.2019).

12. Oracle Database Express Edition Installation Guide. February 2019 [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/18/xeinw/installation-guide.html. – Загл. с экрана (дата обращения: 04.03.2019).

13. Oracle. Архитектура Oracle Multitenant. Июнь 2013 г. [Электронный ресурс]. – Режим доступа: https://www.oracle.com/assets/multitenant-wp-t-2995359-ru.pdf. – Загл. с экрана (дата обращения: 04.03.2019).

14. Oracle SQL Developer. User’s Guide. Release 18.2. E97013-01. July 2018 [Электронный ресурс]. – Режим доступа: https://docs.oracle.com/en/database/oracle/sqldeveloper/18.2/rptug/ index.html. – Загл. с экрана (дата обращения: 04.03.2019).

ПРИЛОЖЕНИЕ А

ПЕРВЫЙ СЦЕНАРИЙ СОЗДАНИЯ ТАБЛИЦ СХЕМЫ *BOOKSHOP*

```
-------------------------------------
-- Создание таблицы Catalogs
-------------------------------------
CREATE TABLE Catalogs (
      cat_ID NUMBER (6,0) NOT NULL,
      cat_name NVARCHAR2 (20) NOT NULL,
      CONSTRAINT Catalogs_PK  PRIMARY KEY (cat_ID)
      ENABLE
);
----------------------------------
-- Создание таблицы Books
----------------------------------
CREATE TABLE Books (
      book_ID NUMBER (6,0) NOT NULL,
      book_name NVARCHAR2 (100) NOT NULL,
      book_author NVARCHAR2 (100) NOT NULL,
      book_year NUMBER (4,0) NOT NULL,
      book_price NUMBER (7,2),
      book_onhand NUMBER (4,0),
      book_reorder NUMBER (4,0),
      book_cat_ID NUMBER (6,0) NOT NULL,
      CONSTRAINT Books_PK PRIMARY KEY (book_ID)
      ENABLE
);
---------------------------------------
-- Создание таблицы Customers
---------------------------------------
CREATE TABLE Customers (
      cust_ID  NUMBER (6,0)  NOT NULL,
      cust_lastname  NVARCHAR2 (20) NOT NULL,
      cust_firstname  NVARCHAR2 (20) NOT NULL,
      cust_street  NVARCHAR2 (100) NOT NULL,
      cust_city  NVARCHAR2 (50) NOT NULL,
      cust_region  NVARCHAR2 (50) NOT NULL,
      cust_zipcode  CHAR (6 CHAR) NOT NULL,
```

```
    cust_phone  NVARCHAR2 (20),
    cust_email  NVARCHAR2 (20),
    cust_status  VARCHAR2 (12 CHAR) NOT NULL,
    CONSTRAINT  Customers_PK  PRIMARY KEY (cust_ID)
    ENABLE
);
-----------------------------------
-- Создание таблицы Orders
-----------------------------------
CREATE TABLE Orders (
    order_ID NUMBER (6, 0) NOT NULL,
    order_cust_ID NUMBER (6, 0) NOT NULL,
    order_date DATE NOT NULL,
    order_ship_date DATE,
    order_paid_date DATE,
    order_status CHAR (1 CHAR) NOT NULL,
    CONSTRAINT Orders_PK PRIMARY KEY (order_ID)
    ENABLE
);
-----------------------------------
-- Создание таблицы Items
-----------------------------------
CREATE TABLE Items (
    item_ID NUMBER (6, 0) NOT NULL,
    item_order_ID NUMBER (6, 0) NOT NULL,
    item_book_ID NUMBER (6, 0) NOT NULL,
    item_quantity NUMBER (4, 0) NOT NULL,
    CONSTRAINT Items_PK  PRIMARY KEY (item_ID)
ENABLE
);
-----------------------------------------------------
-- Определение ограничений уникальности
-----------------------------------------------------
ALTER TABLE Customers
ADD CONSTRAINT Customers_UK1 UNIQUE (cust_phone)
ENABLE;
ALTER TABLE Customers
ADD CONSTRAINT Customers_UK2 UNIQUE (cust_email)
ENABLE;
```

```
-----------------------------------------
-- Определение ограничений проверки
-----------------------------------------
ALTER TABLE Customers
ADD CONSTRAINT Customers_CHK1 CHECK
(cust_status IN ('gold', 'lock', 'active', 'passive'))
ENABLE;
ALTER TABLE Orders
ADD CONSTRAINT Orders_CHK1 CHECK
(order_status IN ('F', 'B', 'O'))
ENABLE;
----------------------------------------
-- Определение внешних ключей
----------------------------------------
ALTER TABLE Books
ADD CONSTRAINT Books_FK1 FOREIGN KEY (book_cat_ID) REFERENCES
Catalogs (cat_ID)
ON DELETE CASCADE
ENABLE;
ALTER TABLE Orders
ADD CONSTRAINT Orders_FK1 FOREIGN KEY (order_cust_ID) REFERENCES
Customers (cust_ID)
ON DELETE CASCADE
ENABLE;
ALTER TABLE Items
ADD CONSTRAINT Items_FK1  FOREIGN KEY (item_book_ID)
REFERENCES Books (book_ID)
ON DELETE CASCADE
ENABLE;
ALTER TABLE Items
ADD CONSTRAINT Items_FK2  FOREIGN KEY (item_order_ID)
REFERENCES Orders (order_ID)
ON DELETE CASCADE
ENABLE;
```

ПРИЛОЖЕНИЕ Б

СЦЕНАРИЙ ЗАПОЛНЕНИЯ ДАННЫМИ ТАБЛИЦ СХЕМЫ *BOOKSHOP*

```
-- Заполнение таблицы Catalogs ----------------------------------------
INSERT INTO   Catalogs (cat_ID, cat_name) VALUES (1, 'Программирование');
INSERT INTO   Catalogs (cat_ID, cat_name) VALUES (2, 'Интернет');
INSERT INTO   Catalogs (cat_ID, cat_name) VALUES (3, 'Базы данных');
INSERT INTO   Catalogs (cat_ID, cat_name) VALUES (4, 'Сети');
INSERT INTO   Catalogs (cat_ID, cat_name) VALUES (5, 'Мультимедиа');
COMMIT;
-- Заполнение таблицы Books -------------------------------------
INSERT INTO Books VALUES
    (1, 'Java. Руководство для начинающих', 'Шилдт Г.', 2012, 636.00, 10, 5, 1);
INSERT INTO Books VALUES
    (2, 'Microsoft Visual Studio 2010', 'Голощапов А.Л.', 2011, 472.00, 2, 10, 1);
INSERT INTO Books VALUES
    (3, 'Язык программирования C++. Лекции и упражнения', 'Прата С.', 2012,
    1674.00, 4, NULL, 1);
INSERT INTO Books VALUES
    (4, 'Язык программирования C# 5.0 и платформа .NET 4.5', 'Троелсен Э.',
    2013, 1672.00, 1, 2, 1);
INSERT INTO Books VALUES
    (5, 'Язык C#. Базовый курс', 'Подбельский В.В.', 2011, 534.00, 6, 4, 1);
INSERT INTO Books VALUES
    (6, 'Delphi. Программирование для Windows, OS X, iOS и Android',
    'Осипов Д.', 2014, 440.00, 6, NULL, 1);
INSERT INTO Books VALUES
    (7, 'PHP, MySQL, XML. Программирование для интернета', 'Бенкен Е.С.',
    2018, 439.00, 5, 15, 1);
INSERT INTO Books VALUES
    (8,'Совершенный код', 'Макконнелл С.', 2014, 814.80, 1, 2, 1);
INSERT INTO Books VALUES
    (9, 'C#. Программирование на языке высокого уровня', 'Павловская Т.А.',
    2013, 442.00, 12, NULL, 1);
INSERT INTO Books VALUES
    (10, 'Новейшая энциклопедия. Компьютер и Internet 2014', 'Леонтьев В.П.',
    2014, 468.00, 4, 5, 2);
```

```
INSERT INTO Books VALUES
    (11, 'Internet: поиск информации и продвижение сайтов', 'Байков В.Д.', 2012,
    395.00, 2, 8, 2);
INSERT INTO Books VALUES
    (12, 'Internet у вас дома', 'Березин С.', 2012, 578.80, 6, NULL, 2);
INSERT INTO Books VALUES
    (13, 'Интернет. Быстрый старт', 'Шапошников И.В.', 2012, 395.00, 4, 6, 2);
INSERT INTO Books VALUES
    (14, 'Популярные интернет-браузеры', 'Маринин С.А.', 2017, 282.00, 6, 8, 2);
INSERT INTO Books VALUES
    (15, 'Анонимность и безопасность в Интернете', 'Колисниченко Д.Н.', 2012,
    376.00, 5, NULL, 2);
INSERT INTO Books VALUES
    (16, 'Основы использования и проектирования баз данных',
    'Илюшечкин В.М.',      2018, 555.00, 2, 4, 3);
INSERT INTO Books VALUES
    (17, 'SQL для чайников', 'Тейлор А.Г. ', 2012, 400.00, 6, 10, 3);
INSERT INTO Books VALUES
    (18, 'Раскрытие тайн SQL', 'Оппель Э.', 2017, 500.00, 3, NULL, 3);
INSERT INTO Books VALUES
    (19, 'Microsoft SQL Server 2012', 'Бондарь А.Г.', 2013, 529.00, 6, 5, 3);
INSERT INTO Books VALUES
    (20, 'Локальные сети. Модернизация и поиск неисправностей',
    'Поляк-Брагинский А.В.', 2012, 456.40, 6, NULL, 4);
INSERT INTO Books VALUES
    (21, 'Компьютерные сети и сетевые технологии', 'Кузьменко Н.Г.', 2013,
    421.70, 4, 4, 4);
INSERT INTO Books VALUES
    (22, 'Практическое руководство системного администратора',
    'Кении А.М.', 2013, 383.10, 5, 4, 4);
INSERT INTO Books VALUES
    (23, 'TCP/IP. Архитектура. Протоколы. Реализация', 'Фейт С.', 2014, 464.00,
    3, NULL, 4);
INSERT INTO Books VALUES
    (24, 'Технологии современных сетей Ethernet', 'Смирнова Е.В.', 2012, 245.50,
    8, 4, 4);
INSERT INTO Books VALUES
    (25, 'Цифровая обработка 2D- и 3D-изображений', 'Красильников Н.Н.',
    2016, 558.40, 20, NULL, 5);
```

```
INSERT INTO Books VALUES
    (26, 'Самоучитель Adobe Illustrator', 'Тучкевич Е.И.', 2014, 553.00, 15, 5, 5);
INSERT INTO Books VALUES
    (27, 'Photoshop для начинающих', 'Заика А.', 2013, 348.00, 10, 10, 5);
INSERT INTO Books VALUES
    (28, 'Звуковой дизайн в видеоиграх', 'Деникин А.А.', 2012, 888.00, 20, NULL, 5);
INSERT INTO Books VALUES
    (29, 'Слушаем музыку и смотрим фильмы на ПК', 'Дронов В.А.', 2017, 452.50,
    12, NULL, 5);
INSERT INTO Books VALUES
    (30, 'Звукозапись на компьютере', 'Петелин Р.Ю.', 2017, 592.00, 8, NULL, 5);
COMMIT;
-- Заполнение таблицы Customers ----------------------------------
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (1,'Иванов','Александр','Луговая 10/122','Пенза','Пензенская область',
    '440195','(8412)-58-98-78','ivanov@email.ru','active');
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (2, 'Лосев', 'Сергей', 'Зеленая 123/12', 'Самара', 'Самарская область',
    '443028','(846)-190-57-77','losev@email.ru','passive');
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (3,'Грачева','Нина','Енисейская 11/56','Красноярск','Красноярский край',
    '660038','(391)-295-66-61','gracheva@email.ru','active');
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (4, 'Кузнецов', 'Максим', 'Маркса 68/19','Омск','Омская область',
    '644035',NULL,'kuznetsov@email.ru','active');
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (5, 'Петров', 'Анатолий', 'Заводская 141/2','Пермь','Пермская область',
    '614112', NULL, NULL, 'lock');
INSERT INTO Customers(cust_ID, cust_lastname, cust_firstname, cust_street,
    cust_city, cust_region, cust_zipcode, cust_phone, cust_email, cust_status)
VALUES (6, 'Корнеев', 'Александр', 'Княжеская 99/56', 'Ярославль',
    'Ярославская область', '150029','(4852)-89-78-36','korneev@email.ru', 'gold');
COMMIT;
```

```
-- Заполнение таблицы Orders ---------------------------------------
INSERT INTO Orders(order_ID, order_cust_ID, order_date, order_ship_date,
    order_paid_date, order_status)
VALUES(1001, 1, DATE'2017-01-05', DATE'2017-01-10', DATE'2017-01-24', 'F');
INSERT INTO Orders(order_ID, order_cust_ID, order_date, order_ship_date,
    order_paid_date, order_status)
VALUES(1002, 5, DATE'2017-01-15', NULL, NULL, 'B');
INSERT INTO Orders(order_ID, order_cust_ID, order_date, order_ship_date,
    order_paid_date, order_status)
VALUES(1003, 6, DATE'2017-01-25', DATE'2017-01-30', DATE'2017-02-14','F');
INSERT INTO Orders(order_ID, order_cust_ID, order_date, order_ship_date,
    order_paid_date, order_status)
VALUES(1004, 3, DATE'2017-02-09', DATE'2017-02-14', DATE'2017-02-28', 'F');
INSERT INTO Orders(order_ID, order_cust_ID, order_date, order_ship_date,
    order_paid_date, order_status)
VALUES(1005, 6, DATE'2017-03-11', DATE'2017-03-16', NULL, 'O');
COMMIT;
-- Заполнение таблицы Items ----------------------------------
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(1, 1001, 12, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(2, 1001, 19, 2);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(3, 1002, 27, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(4, 1003, 7, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(5, 1003, 19, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(6, 1004, 28, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(7, 1004, 29, 1);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(8, 1004, 30, 3);
INSERT INTO Items(item_ID, item_order_ID, item_book_ID, item_quantity)
VALUES(9, 1005, 5, 1);
COMMIT;
```