МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ДОНСКОЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра «Кибербезопасность информационных систем»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНОЙ РАБОТЫ № 1

«ЗНАКОМСТВО С ЯЗЫКОМ PL/SQL. БАЗОВЫЕ ОПЕРАЦИИ»

ПО ДИСЦИПЛИНЕ «ORACLE»

Ростов-на-Дону

ДГТУ

2019

УДК 004.65

Составитель: С.П. Новиков

Методические указания по выполнению лабораторной работы № 1 «Знакомство с языком PL/SQL. Базовые операции». – Ростов-на-Дону : Донской гос. техн. ун-т, 2019. – 11 c.

В методической разработке рассматриваются основные операторы для работы с языком PL/SQL. Поясняются преобразования типов данных. Даны задания к лабораторной работе, помогающие закрепить на практике полученные знания.

УДК 004.65

Печатается по решению редакционно-издательского совета   
Донского государственного технического университета

Научный редактор и.о. зав. кафедрой «Кибербезопасность информационных систем» канд. техн. наук, доцент Д.А. Короченцев

Ответственный за выпуск старший преподаватель кафедры «Кибербезопасность информационных систем» С.П. Новиков

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

В печать \_\_\_.\_\_\_.2019 г.

Формат 60x84/16. Объем \_\_\_\_ усл. п. л.

Тираж\_\_\_ экз. Заказ № \_\_\_.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Издательский центр ДГТУ

Адрес университета и полиграфического предприятия:

344000, г. Ростов-на-Дону, пл. Гагарина, 1

© Донской государственный   
 технический университет, 2019

**Цель работы:** познакомиться с базовыми командами языка SQL (CREATE TABLE, INSERT INTO, UPDATE, DELETE, SELECT) на примере построения модели университета.

1. **Теоретические сведения**
   1. **Язык PL/SQL**

PL/SQL – это сокращение от «Procedural Language extensions to the Structured Query Language», что в переводе с английского означает «процедурные языковые расширения для SQL». SQL — повсеместно распространенный язык для выборки и обновления информации (вопреки названию) в реляционных базах данных. Компания Oracle Corporation разработала PL/SQL для преодоления некоторых ограничений SQL, а также для того, чтобы предоставить более полное совершенное решение для разработчиков ответственных приложений баз данных Oracle [1, с. 36].

Язык PL/SQL обладает следующими определяющими характеристиками:

* высокая структурированность, удобочитаемость и доступность. Этот язык прост в изучении, а его ключевые слова и структура четко выражают смысл кода;
* стандартный переносимый язык разработки приложений для баз данных Oracle. Если вы написали на PL/SQL процедуру или функцию для базы данных Oracle, находящейся на портативном компьютере, то эту же процедуру можно будет перенести в базу данных на компьютере корпоративной сети и выполнить ее без каких-либо изменений (конечно, при условии совместимости версий Oracle). «Написать один раз и использовать везде» — этот основной принцип PL/SQL был известен задолго до появления языка Java. Впрочем, «везде» в данном случае означает «при работе с любой базой данных Oracle»;
* встроенный язык. PL/SQL не используется как самостоятельный язык программирования. Это встроенный язык, работающий только в конкретной управляющей среде. Таким образом, программы PL/SQL можно запускать из базы данных (скажем, через интерфейс SQL\*Plus). Также возможно определение и выполнение программ PL/ SQL из формы или отчета Oracle Developer (клиентский PL/SQL). Однако вы не сможете создать исполняемый файл программы на PL/SQL и запускать его автономно;
* высокопроизводительный, высокоинтегрированный язык баз данных. В настоящее время существует много способов написания программ, работающих с базами данных Oracle. Например, можно использовать Java и JDBC или Visual Basic и ODBC, а можно воспользоваться, скажем, Delphi, C++ и т. д. Однако эффективный код для работы с базой данных Oracle проще написать на PL/SQL, чем на любом другом языке программирования. В частности, Oracle имеет несколько расширений, предназначенных специально для PL/SQL, таких как инструкция FORALL, позволяющая повысить производительность обработки запросов на порядок и более [1, с. 37].

Подобно другим стандартным языкам, PL/SQL имеет языковые элементы для объявления переменных, присвоения значений, проверки условий и ветвления, а также для построения итеративных конструкций. Как С и Pascal язык PL/SQL в значительной степени блочно-ориентирован. Он имеет строгие правила, касающиеся областей видимости переменных, поддерживает параметризованные вызовы. PL/SQL предусматривает строгий контроль типов; ошибки несовместимости типов контролируются на этапах компиляции и выполнения. Выполняются также явное и неявное преобразования типов. В языке поддерживаются определенные пользователем сложные структуры данных. В нем также допустима перегрузка подпрограмм для создания гибкой среды прикладного программирования.

Поскольку язык PL/SQL является процедурной оболочкой языка SQL, он хорошо интегрирован с SQL. Различные возможности языка позволяют ему взаимодействовать с СУБД Oracle, выполняя операции над наборами и отдельными строками таблиц.

Язык PL/SQL имеет элемент Exception Handler (обработчик исключительных ситуаций) для синхронной обработки ошибок и исключительных ситуаций, которые могут возникнуть во время выполнения.

Язык PL/SQL не является объектно-ориентированным и имеет некоторые общие черты с языками Pascal и Ada, его изучение не будет проблемой для того, кто знает синтаксис языка Pascal. В отличие от С и Pascal, в языке PL/SQL не поддерживаются указатели. Язык PL/SQL является, в первую очередь, средством разработки программ, имеющих дело непосредственно с таблицами и другими объектами баз данных. Взаимодействие с операционной системой и внешними программными компонентами осуществляется через поставляемые пакеты управления базами данных.

Язык PL/SQL обеспечивает строгий контроль за выполнением логических транзакций. Одним из средств такого контроля является система полномочий на выполнение. Вместо того чтобы выдавать разрешения на обновление таблиц ролям или пользователям, вы выдаете разрешение только на выполнение процедуры, которая управляет и предоставляет доступ к необходимым структурам данных. Владельцем процедуры является другая схема базы данных Oracle («определитель»), которой, в свою очередь, предоставляются разрешения на обновление таблиц, участвующих в транзакции. Таким образом, процедура становится «привратником» транзакции. Программный код (будь то приложение Oracle Forms или исполняемый файл Pro\*C) может выполняться только посредством вызова процедуры, и это гарантирует целостность транзакций приложения. Начиная с Oracle8i, в Oracle появилось ключевое слово *AUTHID*, значительно расширяющее гибкость модели защиты PL/SQL. Оно позволяет выполнять программы в соответствии с описанной ранее моделью прав определителя или же выбрать режим *AUTHID*.

* 1. **Основные операторы языка**

Синтаксис основных операторов языка PL/SQL:

1. создание таблицы – оператор *CREATE**TABLE:*

CREATE [GLOBAL TEMPORARY] TABLE <имя\_таблицы>

({<имя\_колонки> <тип\_колонки> [DEFAULT <выражение>]})

[TABLESPACE <имя\_табличного\_пространства>]

[PCTFREE <процент при\_заполнении ниже\_которого\_блок\_считается свободным>]

[PCTUSED <процент заполнения блока>]

STORAGE (INITIAL <размер\_начального\_экстента>

NEXT <размер\_следующего\_экстента>

PCTINCREASE <процент\_увеличения\_таблиц\_передающих\_экстентов>

MINEXTENTS <число\_экстентов\_при\_создании>

MAXEXTENTS <максимальное\_число\_экстентов>

FREELISTS <число\_списков\_свободного\_пространства>)

PCTFREE – процент свободного пространства. Если свободное пространство занимает меньший процент, чем PCTFREE, то считается, что блок полностью свободен. Оставшееся свободное пространство резервируется для поддержки расширения имеющихся в блоке записей, без изменения их размера. По умолчанию 10 %.

PCTUSED – процент заполнения блока. Если после полного заполнения (нет свободных строк (записей)), блок будет заполнен менее, чем на PCTUSED, то считается, что блок содержит свободные субблоки, и он (блок) возвращается в список FREELIST. По умолчанию 40 % [2, 3].

Примеры:

* CREATE TABLE group\_name (id int not null, name char, kurs char);
* CREATE TABLE new\_tbl AS SELECT \* FROM old\_tbl;
* CREATE TABLE lookup

(id INT, INDEX USING BTREE (id))

ENGINE = MEMORY;

* CREATE TABLE t1 (col1 INT, col2 CHAR(5))
* PARTITION BY HASH(col1).
* CREATE TABLE table\_name(id INT NOT NULL) tablespace tbs\_perm\_01 pctfree 10 pctused 90 storage (INITIAL 100M NEXT 10M minextents 1 maxextents 10 pctincrease 0);

1. получение выборки информации – оператор *SELECT*:

SELECT [ALL | DISTINCT [ON (выражение [. ...])]] цель [AS имя] [. ...] [FROM источник [. ... ]] [[NATURAL] тип\_обьединения источник [ON условие \ USING (список\_полей)]] [. ...]

[WHERE условие] [GROUP BY критерий [. ...]] [HAVING условие [. ...]] [{UNION | INTERSECT | EXCEPT} [ALL] подзапрос] [ORDER BY выражение [ASC | DESC | USING оператор] [. ...]]

[FOR UPDATE [OF таблица [. ...]]] [LIMIT {число | ALL} [{OFFSET | . } начало ]]

*ALL*. Необязательное ключевое слово ALL указывает на то, что в выборку включаются все найденные записи.

*DISTINCT [ON (выражение [, ...] ) ]*. Секция DISTINCT определяет поле или выражение, значения которого должны входить в итоговый набор не более одного раза.

*цель [AS имя ] [, ...].* В качестве цели обычно указывается имя поля, хотя цель также может быть константой, идентификатором, функцией или общим выражением. Перечисляемые цели разделяются запятыми, существует возможность присвоения псевдонимов именам целей в секции AS. Звездочка (\*) является сокращенным обозначением всех несистемных полей, вместе с ней в списке могут присутствовать и другие цели.

*FROM источник [. ... ].* В секции FROM указывается источник, в котором SQL ищет заданные цели. В данном случае источник является именем таблицы или подзапроса. Допускается перечисление нескольких источников, разделенных запятыми (примерный аналог перекрестного запроса). В этом описании источник представляет собой имя таблицы или подзапрос. Эти общие формы имеют следующий синтаксис:

FROM {[ONLY] таблица [[AS] псевдоним [(псевдоним\_поля [....])]]] (запрос) [AS] псевдоним [(псевдоним\_поля [. ...])]

* *[ONLY] таблица.* Имя таблицы, используемой в качестве источника для команды SELECT. Ключевое слово ONLY исключает из запроса записи всех таблиц-потомков;
* *[AS] псевдоним*. Источникам FROM могут назначаться необязательные псевдонимы, упрощающие запрос (например, на таблицу books можно ссылаться по псевдониму). Ключевое слово AS является необязательным;
* *(запрос) [AS] псевдоним*. В круглых скобках находится любая синтаксически правильная команда SELECT. Итоговый набор, созданный запросом, используется в качестве источника FROM так, словно выборка производится из статической таблицы. При выборке из подзапроса обязательно должен назначаться псевдоним;
* *(псевдоним\_поля [. ...])*. Псевдонимы могут назначаться не только всему источнику, но и его отдельным полям. Перечисляемые псевдонимы полей разделяются запятыми и группируются в круглых скобках за псевдонимом источника FROM. Псевдонимы перечисляются в порядке следования полей в таблице, к которой они относятся. При выборке информации имя объекта и свойства объекта разделяются точкой.

*[NATURAL] тип\_объединения источник [ON условие \ USING (список\_полей)].* Источники FROM могут группироваться в секции JOIN с указанием типа объединения (INNER, FULL, OUTER, CROSS). В зависимости от типа объединения также может потребоваться уточняющее условие или список полей.

*WHERE* условие. Секция WHERE ограничивает итоговый набор заданными критериями. Условие должно возвращать простое логическое значение (true или false), но оно может состоять из нескольких внутренних условий, объединенных логическими операторами (например, AND или OR).

*GROUP* *BY* критерий [, ... ]. Секция GROUP BY обеспечивает группировку записей по заданному критерию. Причем критерий может быть простым именем поля или произвольным выражением, примененным к значениям итогового набора.

*HAVING* условие [. ... ]. Секция HAVING похожа на секцию WHERE, но условие проверяется на уровне целых групп, а не отдельных записей.

*{UNION | INTERSECT | EXCEPT} [ ALL ]* подзапрос. Выполнение одной из трех операций, в которых участвуют два запроса (исходный и дополнительный). Итоговые данные возвращаются в виде набора с обобщенной структурой, из которого удаляются дубликаты записей (если не было задано ключевое слово ALL):

* *UNION* – объединение (записи, присутствующие в любом из двух наборов);
* *INTERSECT –* пересечение (записи, присутствующие одновременно в двух наборах);
* *EXCEPT* – исключение (записи, присутствующие в основном наборе SELECT, но не входящие в дополнительный).

*ORDER* *BY* выражение. Сортировка результатов команды SELECT по заданному выражению.

*[ASC | DESC | USING оператор]*. Порядок сортировки, определяемой секцией ORDER BY выражение: по возрастанию (ASC) или по убыванию (DESC). С ключевым словом USING может задаваться оператор, определяющий порядок сортировки (например, < или >).

*FOR UPDATE* *[OF таблица [. ... ]].* Возможность монопольной блокировки возвращаемых записей. В транзакционных блоках FOR UPDATE блокирует записи указанной таблицы до завершения транзакции. Заблокированные записи не могут обновляться другими транзакциями.

*LIMIT {число | ALL}*. Ограничение максимального количества возвращаемых записей или возвращение всей выборки (ALL).

*{ OFFSET | } начало*. Точка отсчета записей для секции LIMIT. Например, если в секции LIMIT установлено ограничение в 100 записей, а в секции OFFSET – 50, запрос вернет записи с номерами 50-150 (если в итоговом наборе найдется столько записей) [2, 3].

Пример: SELECT name, p.home FROM people, р WHERE id=0

1. удаление записей из таблицы – оператор *DELETE****:***

DELETE [Таблица.\*] FROM Таблица WHERE ...;

*Таблица* – имя таблицы, из которой удаляются записи.

Использование предложения *WHERE* аналогично его использованию в команде SELECT.

Аргумент команды DELETE можно не указывать, поскольку он фактически дублируется в предложении FROM.

Пример**:** DELETE FROM Buyers WHERE ID=8;

Этот запрос удаляет из таблицы Buyers запись, в которой ID равно 8. Для удаления не всей записи, а только ее поля, следует воспользоваться запросом на изменение записи (команда UPDATE) и поменять значения нужных полей на Null [2, 3].

1. вставка новых записей в таблицу – оператор *INSERT**INTO***:**

Команда INSERT INTO предназначена для добавления одной или нескольких записей в таблицу. Возможны 2 варианта использования этой команды. Первый вариант добавляет одну запись в таблицу, а второй вариант добавляет в таблицу результат выполнения SELECT выражения.

Синтаксис первого варианта:

INSERT INTO ТаблицаНазначения [(Поля)] VALUES (Значения);

Синтаксис второго варианта:

INSERT INTO ТаблицаНазначения [(Поля)] [IN БазаДанных] SELECT…;

*ТаблицаНазначения* – таблица, в которую добавляются записи.

*Поля* – названия полей.

*Таблица* – имя таблицы, источника данных.

*База* *данных* – путь и имя внешней базы данных, в которой содержатся таблицы. Если таблицы находятся в текущей базе данных, то этот аргумент необязателен.

*Значения* – значения полей добавляемой записи.

Все поля записи и соответствующие им значения должны быть определены, иначе им будут присвоены значения Null.

Важно! Если таблица, в которую добавляются записи, имеет ключевое поле, то в него должны добавляться уникальные, непустые значения. Иначе запись не будет добавлена.

Пример: INSERT INTO Orders (ID, Name, Email, Order) VALUES (12, 'Вася Пупкин', 'vasya@pupkin.ru', 'Pentium II 450 MHz');

Добавляется новая запись, в которой полям ID, Name, Email, Order соответствуют значения 12, 'Вася Пупкин', 'vasya@pupkin.ru', 'Pentium II 450 MHz'.

INSERT INTO Orders2001 (ID, Name, Email, Order) SELECT ID, Name, Email, Order FROM Orders2000;

Этот запрос добавляет все записи из таблицы Orders2000 в таблицу Orders2001 [2, 3].

1. **Задание для выполнения**

Рекомендуемое программное обеспечение для выполнения задания – Oracle Database 19c.

Рассматриваемые объекты: факультеты, кафедры, группы, студенты.

Объект «Факультет» содержит следующую информацию:

* идентификатор;
* наименование факультета;
* декан факультета.

Объект «Кафедра» содержит следующую информацию:

* идентификатор;
* наименование кафедры;
* заведующий кафедрой.

Объект «Группа» содержит следующую информацию:

* идентификатор;
* наименование группы;
* специальность;
* курс.

Объект «Студент» содержит следующую информацию:

* ФИО студента;
* студенческая группа;
* номер студенческого билета;

Необходимо:

1. Провести анализ вышеперечисленных объектов, их свойств и связей согласно теории нормализации. Создать для них таблицы. Заполнить таблицы информацией (5-10 строк).
2. Продемонстрировать работу с данными (добавление, изменение, удаление данных).
3. Продемонстрировать примеры выбора информации из СУБД по заданию преподавателя (например, показать всех студентов такой-то группы, кафедры, факультета; показать все группы такой-то кафедры, факультета; показать всех студентов такого-то курса и т.п.).

**3. Критерии оценивания**

1. Для получения минимального количества баллов необходимо выполнить задание лабораторной работы. Провести обоснование выбора связей и определение необходимых типов данных. Выполнить нормализацию таблиц. Продемонстрировать работу основных операторов по заданию. Ответить на один теоретический вопрос.
2. Для получения баллов на оценку «хорошо», так же необходимо показать умение добавлять поля и данные в таблицы по заданию преподавателя. Составить с их использованием 2 и более сложноподчиненных запроса.
3. Для получения баллов на оценку «отлично», дополнительно необходимо показать умение устанавливать связи между таблицами к добавленным полям. Составить с их использованием 3 и более сложноподчиненных запроса. Ответить на дополнительный теоретический вопрос.

**4. Контрольные вопросы**

1. Что представляет собой язык PL/SQL?

2. Перечислите основные операторы PL/SQL.

3. С помощью какого оператора PL/SQL осуществляется выборка из БД? Приведите пример.

4. С помощью какого оператора PL/SQL можно удалить запись из БД? Приведите пример.

**5. Требования к отчету**

Отчёт выполняется каждым студентом индивидуально. Работа должна быть оформлена в электронном виде в формате .doc или .docx и распечатана на листах формата А4.

На титульном листе указываются: наименование учебного учреждения, наименование дисциплины, название и номер работы, вариант, выполнил: фамилия, имя, отчество, группа, проверил: преподаватель ФИО.

Отчет должен содержать:

* название и цель работы;
* краткие теоретические сведения, ответы на контрольные вопросы;
* протокол выполнения лабораторной работы, содержащий листинг программ, составленных при выполнении работы, и результаты их выполнения (в виде скриншотов);
* выводы по результатам работы.

**ЛИТЕРАТУРА**

1. Фейерштейн С., Прибыл Б. Oracle PL/SQL. Для профессионалов. 6-е изд. — СПб.: Питер, 2015. — 1024 с.
2. Oracle Database PL/SQL: Language Reference 12c Release 1 (12.1) [Электронный ресурс] / L. Morin, S. Moore, D. Alpern и др. – 2017. – Режим доступа: https://docs.oracle.com/database/121/LNPLS/E50727-06.pdf(дата обращения: 02.06.2019).
3. Алапати, Сэм Р. А45 Oracle Database 11g: руководство администратора баз данных.: Пер. с англ. — М.: ООО «И.Д. Вильямс», 2010. — 1440 с.