

# Содержание

## ТОМ I

<b>Часть I</b>	<b>Критические концепции базы данных</b> .....	1
<b>Глава 1</b>	<b>Опции архитектуры Oracle Database 11g</b> .....	2
	Базы данных и экземпляры .....	4
	Внутри базы данных .....	4
	Хранение данных .....	7
	Защита данных .....	8
	Программные структуры .....	9
	Выбор архитектур и опций .....	10
<b>Глава 2</b>	<b>Инсталлирование Oracle Database 11g и создание базы данных</b> .....	12
	Обзор опций лицензирования и инсталляции .....	14
	Использование OUI для инсталляции программного обеспечения Oracle .....	14
<b>Глава 3</b>	<b>Обновление до Oracle Database 11g</b> .....	21
	Выбор метода обновления .....	22
	Перед обновлением .....	23
	Запуск информационного инструмента предворяющего обновление .....	24
	Использование Database Upgrade Assistant .....	25
	Выполнение прямого обновления вручную .....	26
	Использование утилит Export и Import .....	27
	Какие версии Export и Import нужно использовать .....	27
	Выполнение обновления .....	28
	Использование методов копирования данных .....	29
	После обновления .....	30
<b>Глава 4</b>	<b>Планирование приложений Oracle: подходы, риски, стандарты</b> .....	31
	Кооперативный подход .....	33
	«Данные» имеются у каждого .....	34
	Хорошо известный язык Oracle .....	35

Таблицы с информацией.....	36
Язык структурированных запросов (SQL).....	36
Простой запрос Oracle.....	37
Почему все это называется «реляционным».....	38
Несколько обычных каждодневных примеров.....	40
В чем заключаются риски.....	42
Важность нового видения.....	43
Изменяющиеся среды.....	44
Коды, сокращения и стандарты именования.....	44
Как уменьшить путаницу.....	46
Нормализация.....	47
Английские слова для данных.....	54
Использование заглавных букв в именах и данных.....	54
Нормализация имен.....	55
На хорошем проекте остается отпечаток человеческого прикосновения.....	56
Понимание задач приложений.....	56
Конспект задач.....	58
Понимание данных.....	61
Атомарные модели данных.....	63
Атомарная модель бизнеса.....	63
Бизнес-модель.....	63
Ввод данных.....	64
Составление запросов и отчетов.....	64
По направлению к нормализации имен объектов.....	65
Целостность на уровне имен.....	66
Внешние ключи.....	67
Имена.....	67
Краткость.....	68
Тезаурус имен объектов.....	68
Интеллектуальные ключи и значения столбцов.....	69
Заповеди.....	70
<b>Часть II SQL и SQL *Plus.....</b>	<b>71</b>
<b>Глава 5 Основные части речи языка SQL.....</b>	<b>72</b>
Стиль.....	73
Создание таблицы NEWSPAPER.....	74
Использование SQL для выборки данных из таблиц.....	75
Фразы select, from, where и order by.....	79
Логика и значения.....	81
Проверка одиночных значений.....	81
LIKE.....	84

	Простые проверки по списку значений .....	87
	Объединение логических операторов .....	89
	Еще одно использование выражения where: Подзапросы .....	90
	Одиночные значения из подзапроса.....	91
	Списки значений из подзапроса .....	92
	Объединение таблиц .....	94
	Создание представления .....	96
	Расширение представления .....	98
<b>Глава 6</b>	<b>Основные команды и отчеты SQL*Plus .....</b>	<b>100</b>
	Построение простого отчета.....	103
	1 remark .....	104
	2 set headsep .....	104
	3 ttitle и btitle .....	105
	column.....	106
	8 break on.....	107
	9 compute avg .....	108
	10 set linesize .....	109
	set pagesize .....	109
	set newpage.....	109
	11 spool.....	110
	12 /* */ .....	111
	Некоторые разъяснения относительно заголовков столбцов .....	112
	Прочие возможности .....	113
	Редактор командной строки .....	113
	set pause .....	116
	save .....	116
	store .....	117
	Редактирование .....	117
	host .....	118
	Добавление команд SQL *Plus.....	118
	start .....	118
	Проверка среды SQL *Plus .....	120
	Строительные блоки .....	121
<b>Глава 7</b>	<b>Получение и изменение текстовой информации .....</b>	<b>123</b>
	Типы данных .....	123
	Что такое строка.....	124
	Нотация .....	125
	Конкатенация (    ) .....	127
	Как вырезать и вставлять строки.....	128

RPAD и LPAD.....	128
LTRIM, RTRIM и TRIM.....	129
Комбинирование двух функций.....	130
Использование функции TRIM.....	133
Добавление еще одной функции.....	134
LOWER, UPPER и INITCAP.....	134
LENGTH.....	136
SUBSTR.....	137
INSTR.....	140
ASCII и CHR.....	145
Использование фраз order by и where со строковыми функциями.....	145
SOUNDEX.....	147
Поддержка национальных языков.....	148
Регулярные выражения.....	149
Повторение.....	149
<b>Глава 8 Поиск регулярных выражений.....</b>	<b>150</b>
Поисковые цепочки.....	150
REGEXP_SUBSTR.....	155
REGEXP_INSTR.....	157
REGEXP_LIKE.....	158
REPLACE и REGEXP_REPLACE.....	158
REGEXP_COUNT.....	163
<b>Глава 9 Игры с цифрами.....</b>	<b>165</b>
Три класса числовых функций.....	165
Система обозначений.....	166
Функции, работающие с одиночными значениями.....	166
Сложение (+), вычитание (-), умножение (*) и деление (/).....	167
NULL.....	167
NVL: Замена значений NULL.....	168
ABS: Абсолютное значение.....	170
CEIL.....	170
FLOOR.....	170
MOD.....	170
POWER.....	171
SQRT: Квадратный корень.....	171
EXP, LN и LOG.....	171
ROUND и TRUNC.....	172
SIGN.....	174
SIN, SINH, COS, COSH, TAN, TANH, ACOS, ATAN ATAN2 и ASIN.....	174

Агрегатные функции .....	174
Значения NULL в групповых функциях .....	175
Примеры функций, работающих с одним значением, и групповых функций .....	176
AVG, COUNT, MAX, MIN и SUM .....	177
Комбинирование групповых функций и функций, работающих с одним значением .....	177
STDDEV и VARIANCE .....	180
Опция DISTINCT в групповых функциях .....	180
Функции, работающие со списками .....	181
Нахождение строк с помощью MAX и MIN .....	183
Старшинство операций и скобки .....	184
Повторение .....	186
<b>Глава 10 Даты: Затем, сейчас и различие между ними .....</b>	<b>188</b>
Арифметика дат .....	188
SYSDATE, CURRENT_DATE и SYSTIMESTAMP .....	189
Разница между двумя датами .....	190
Сложение месяцев .....	190
Вычитание месяцев .....	191
GREATEST и LEAST .....	192
NEXT_DAY .....	192
LAST_DAY .....	194
Функция MONTH_BETWEEN .....	195
Комбинирование функций, работающих с датами .....	195
Использование функций ROUND и TRUNC в вычислениях с датами .....	196
Форматирование в функциях TO_DATE и TO_CHAR .....	197
Наиболее часто встречающиеся ошибки при применении функции TO_CHAR .....	202
NEW_TIME: Переключение часовых поясов .....	203
Вычисления в TO_DATE .....	203
Даты во фразах where .....	206
Как быть с несколькими столетиями .....	207
Использование функции EXTRACT .....	208
Использование типов данных TIMESTAMP .....	209
<b>Глава 11 Функции конвертирования и преобразования .....</b>	<b>211</b>
Элементарные функции конвертирования .....	214
Автоматическое конвертирование типов данных .....	216
Предупреждение относительно автоматического конвертирования .....	219
Специализированные функции конвертирования .....	219

Функции преобразования .....	220
TRANSLATE .....	220
DECODE .....	222
Повторение .....	223
<b>Глава 12 Создание групп .....</b>	<b>224</b>
Использование фраз group by и having .....	224
Добавление фразы order by .....	226
Порядок выполнения .....	227
Представления групп .....	229
Переименование столбцов с помощью псевдонимов .....	230
Возможности представлений групп .....	231
Фраза order by в представлениях .....	233
Логика во фразе having .....	233
Фраза order by со столбцами и с групповыми функциями .....	235
Соединение столбцов .....	236
Дополнительные возможности создания групп .....	236
<b>Глава 13 Если один запрос зависит от другого .....</b>	<b>237</b>
Расширенные подзапросы .....	237
Коррелированные подзапросы .....	238
Координирование логических проверок .....	239
Использование EXISTS и связанных с ним коррелированных подзапросов .....	241
Внешние соединения .....	243
Синтаксис внешних соединений до Oracle9i .....	243
Используемый сейчас синтаксис внешних соединений .....	245
Замена NOT IN внешним соединением .....	246
Замена NOT IN фразой NOT EXISTS .....	247
Соединения типов NATURAL и INNER .....	249
UNION, INTERSECT и MINUS .....	249
Подзапросы IN .....	253
Ограничения на операторы UNION, INTERSECT и MINUS .....	253
<b>Глава 14 Некоторые сложные возможности .....</b>	<b>254</b>
Сложные группировки .....	254
Использование временных таблиц .....	256
Использование функций ROLLUP, GROUPING и CUBE .....	257
Генеалогические деревья и connect by .....	261
Исключение индивидуумов и ветвей .....	265
Путешествие в поисках корней .....	266
Основные правила .....	268
<b>Глава 15 Изменение данных: операторы insert, update, merge и delete .....</b>	<b>269</b>

insert.....	269
NUMBER .....	270
Вставка времени .....	270
insert вместе с select.....	271
Использование подсказки APPEND для повышения производительности операций вставки .....	272
rollback, commit и autocommit .....	273
Использование контрольных точек.....	274
Неявный commit.....	275
Автоматический откат .....	275
Многократная вставка данных в таблицы .....	275
delete .....	279
update.....	280
update со встроенным select .....	282
update и NULL.....	283
Использование команды merge .....	283
Обработка ошибок .....	286
<b>Глава 16 DECODE и CASE: if, then и else в SQL.....</b>	<b>289</b>
if, then, else.....	289
Замена значений с помощью DECODE.....	292
DECODE внутри DECODE .....	294
Применение в DECODE сравнений GREATER THAN и LESS THEN .....	297
Использование CASE.....	299
Использование PIVOT .....	302
<b>Глава 17 Создание таблиц, представлений, индексов, кластеров и последовательностей и управление ими .....</b>	<b>305</b>
Создание таблицы.....	305
Число символов и разрядность данных типа NUMBER.....	306
Округление во время вставки значений .....	309
Ограничения в операторе create table.....	311
Назначение индексных табличных пространств .....	312
Именованые ограничения.....	314
Удаление таблиц .....	314
Изменение таблиц .....	315
Правила для добавления или модификации столбцов.....	318
Создание таблиц только для чтения.....	319
Активное изменение используемых таблиц .....	319
Создание виртуальных столбцов.....	320
Удаление столбца.....	321
Создание таблицы на базе другой таблицы .....	322

Создание индексно-организованной таблицы .....	324
Создание представлений .....	325
Стабильность представления .....	325
Фраза <code>order by</code> в представлениях .....	327
Создание представления «только для чтения» .....	327
Индексы .....	328
Создание индекса .....	329
Принудительное обеспечение уникальности .....	330
Создание уникального индекса .....	330
Создание битовых индексов .....	331
Когда создавать индекс .....	332
Создание невидимых индексов .....	332
Разновидности индексируемых столбцов .....	333
Сколько индексов можно использовать для таблицы .....	334
Размещение индексов в базе данных .....	334
Перестройка индекса .....	335
Индексы на базе функций .....	336
Кластеры .....	336
Последовательности .....	338
<b>Глава 18 Разбиение таблиц .....</b>	<b>340</b>
Создание секционированных таблиц .....	340
Список разделов .....	343
Создание подразделов .....	343
Создание диапазона и интервала разделов .....	344
Индексация разделов .....	346
Управление секционированными таблицами .....	346
<b>Глава 19 Основы системы безопасности Oracle .....</b>	<b>348</b>
Пользователи, роли и привилегии .....	348
Создание пользователей .....	349
Управление паролями .....	350
Стандартные роли .....	354
Формат команды <code>grant</code> .....	355
Отзыв привилегий .....	356
Что пользователи могут предоставить другим пользователям .....	356
Переход к другому пользователю с помощью <code>connect</code> .....	358
<code>create synonym</code> .....	361
Использование непредоставленных привилегий .....	361
Передача привилегий .....	362
Создание роли .....	363



Предоставление привилегий ролям .....	364
Назначение роли другой роли.....	364
Назначение роли пользователям.....	365
Добавление к роли пароля.....	365
Удаление пароля из роли .....	366
Активация и блокировка ролей.....	366
Отзыв привилегий из ролей.....	367
Удаление роли.....	368
Предоставление привилегии UPDATE для конкретных столбцов....	368
Отзыв объектных привилегий.....	368
Защита по пользователям.....	369
Предоставление доступа пользователю public.....	370
Предоставление ограниченных ресурсов .....	371
<b>Часть III Строим дальше.....</b>	<b>373</b>
<b>Глава 20 Расширенная система безопасности – виртуальная частная база данных .....</b>	<b>374</b>
Первоначальная конфигурация.....	375
Создание контекста приложения .....	376
Создание триггера входа в систему.....	377
Создание политики безопасности.....	378
Применение политики безопасности к таблицам .....	380
Тест VPD .....	380
Как реализовать VPD на уровне столбца.....	382
Как отключить VPD .....	382
Как использовать группы политик.....	384
<b>Глава 21 Расширенные средства защиты: шифрование данных .....</b>	<b>386</b>
Шифрование данных в столбцах .....	386
Установка.....	387
Дополнительные установки для RAC .....	388
Открытие и закрытие кошелька.....	388
Шифрование и дешифрование столбцов .....	389
Шифрование табличного пространства .....	390
Установка.....	391
Создание зашифрованного табличного пространства .....	391
<b>Глава 22 Работа с табличными пространствами .....</b>	<b>393</b>
Табличные пространства и структура базы данных.....	393
Содержимое табличных пространств.....	394
Пространство мусорной корзины в табличных пространствах .....	397
Табличные пространства «только для чтения».....	397

Табличные пространства nologging .....	398
Временные табличные пространства .....	398
Табличные пространства для системно-управляемых отмен действий .....	399
Табличные пространства Bigfile .....	399
Зашифрованные табличные пространства .....	400
Поддержка ретроспекции базы данных .....	400
Переносимые табличные пространства .....	401
Планирование использования табличных пространств .....	401
Разделите активные и статические таблицы .....	402
Разделите индексы и таблицы .....	402
Разделите большие и малые объекты .....	402
Отделите таблицы приложения от основных объектов базы данных .....	402
<b>Глава 23 Использование SQL*Loader для загрузки данных .....</b>	<b>404</b>
Управляющий файл .....	405
Загрузка данных переменной длины .....	406
Запуск загрузки .....	407
Логические и физические записи .....	410
Замечания о синтаксисе управляющего файла .....	412
Управление загрузкой данных .....	414
Повторение загрузки данных .....	415
Настройка загрузки данных .....	416
Загрузка методом Direct Path .....	418
Дополнительные возможности .....	420
<b>Глава 24 Использование утилит Data Pump Export и Data Pump Import .....</b>	<b>421</b>
Создание каталога .....	421
Опции Data Pump Export .....	422
Запуск задания Data Pump Export .....	425
Приостановка и перезапуск выполняющихся заданий .....	426
Экспорт из другой базы данных .....	427
Использование EXCLUDE, INCLUDE и QUERY .....	428
Опции Data Pump Import .....	429
Запуск задания Data Pump Import .....	432
Остановка и возобновление выполняющихся заданий .....	434
EXCLUDE, INCLUDE и QUERY .....	434
Трансформирование импортированных объектов .....	435
Генерация SQL .....	435
<b>Глава 25 Доступ к удаленным данным .....</b>	<b>438</b>

Связи баз данных .....	438
Как работает связь баз данных.....	438
Использование связи баз данных для дистанционных запросов ...	439
Использование связей баз данных для синонимов и представлений .....	440
Использование связей базы данных для дистанционных обновлений.....	442
Синтаксис связей баз данных .....	443
Использование синонимов для обеспечения прозрачности дислокации	446
Использование в представлениях псевдосто́лбца User.....	448
<b>Глава 26 Использование материализованных представлений.....</b>	<b>451</b>
Функциональное назначение.....	451
Требуемые системные привилегии.....	452
Требуемые привилегии для таблиц .....	453
Представления «только для чтения» и обновляемые представления.....	453
Синтаксис команды create materialized view .....	454
Типы материализованных представлений .....	458
Сравнение материализованных представлений на базе первичных ключей и Rowid .....	459
Использование предварительно созданных таблиц .....	460
Индексирование таблиц материализованных представлений.....	460
Использование материализованных представлений для изменения пути выполнения запроса .....	461
Применение DBMS_ADVISOR .....	463
Обновление материализованных представлений .....	466
Какие типы обновлений могут выполняться? .....	466
Быстрое обновление с CONSIDER FRESH .....	469
Автоматические обновления.....	470
Ручные обновления .....	471
Синтаксис команды create materialized view log .....	472
Изменение материализованных представлений и их журналов.....	474
Удаление материализованных представлений и журналов.....	474
<b>Глава 27 Использование Oracle Text для поиска в текстах.....</b>	<b>476</b>
Добавление текста в базу данных.....	476
Текстовые запросы и текстовые индексы.....	477
Текстовые запросы .....	478
Допустимые выражения текстовых запросов.....	479
Поиск точного совпадения слов .....	480
Поиск точного совпадения нескольких слов .....	481
Поиск точно совпадающих фраз .....	484

Поиск слов, расположенных недалеко друг от друга .....	485
Использование групповых символов при поиске .....	486
Поиск по словам с общей основой.....	487
Поиск нечетких совпадений .....	488
Поиск слов, звучащих похоже на другие слова.....	489
Использование оператора ABOUT .....	491
Синхронизация индексов .....	491
Индексные наборы.....	492
<b>Глава 28 Использование внешних таблиц .....</b>	<b>494</b>
Доступ к внешним данным.....	494
Создание внешней таблицы .....	495
Опции создания внешних таблиц .....	498
Загрузка внешних таблиц при создании .....	504
Изменение внешних таблиц .....	505
Параметры доступа.....	505
Добавить столбец .....	506
Каталог по умолчанию .....	506
Удалить столбец.....	506
Расположение .....	506
Модифицировать столбец .....	506
Параллельность .....	506
Проецировать столбец.....	506
Reject Limit.....	507
Ограничения, преимущества и потенциальное использование внешних таблиц.....	507
<b>Глава 29 Использование ретроспективных запросов.....</b>	<b>510</b>
Ретроспективные запросы по временному критерию .....	511
Сохранение данных .....	512
Пример ретроспективы на базе SCN.....	513
Если ретроспективный запрос заканчивается аварийно.....	514
Какой SCN ассоциирован с каждой строкой .....	515
Запросы ретроспективных версий.....	516
Планирование ретроспекций.....	518
<b>Глава 30 Возврат назад – Таблицы и базы данных.....</b>	<b>520</b>
Команда flashback table.....	520
Необходимые привилегии.....	520
Обработка удаленных таблиц.....	521
Включение и отключение корзины.....	522
Возврат назад до определенного SCN или метки времени .....	523

Индексы и статистики .....	524
Команда flashback database .....	524
<b>Глава 31 SQL Replay .....</b>	<b>528</b>
Высокоуровневая конфигурация .....	528
Локализация и ссылки .....	528
Создание каталога для журнала выполненных команд .....	529
Сбор данных .....	530
Работа с фильтрами .....	530
Запуск записи журнала .....	531
Остановка записи .....	532
Экспорт данных Автоматического хранилища данных рабочей нагрузки (AWR) ПРОТОКОЛА .....	532
Обработка журнала .....	532
Воспроизведение журнала выполненных команд .....	533
Запуск клиентов воспроизведения и управление ими .....	533
Инициализация и запуск воспроизведения .....	534
Экспорт данных AWR .....	536
<b>Часть IV PL/SQL .....</b>	<b>537</b>
<b>Глава 32 Введение в PL/SQL .....</b>	<b>538</b>
Обзор PL/SQL .....	538
Секция объявления переменных .....	539
Секция выполняемых команд .....	542
Условная логика .....	544
Циклы .....	545
Операторы CASE .....	554
Секция обработки исключительных ситуаций .....	555
<b>Глава 33 Обновления приложений Онлайн .....</b>	<b>559</b>
Базы данных высокой готовности .....	559
Архитектура Oracle Data Guard .....	560
Создание конфигурации резервной базы данных .....	562
Управление ролями — через переключения и обработку отказов .	564
Создание мало воздействующего изменения DDL .....	567
Создание виртуальных столбцов .....	568
Изменения активно используемых таблиц .....	569
Добавление столбцов NOT NULL .....	569
Онлайн реорганизация объектов .....	570
Удаление столбца .....	573
<b>Глава 34 Триггеры .....</b>	<b>574</b>

Необходимые системные привилегии .....	574
Привилегии, необходимые для работы с таблицами .....	575
Типы триггеров .....	575
Триггеры уровня строки .....	575
Триггеры уровня оператора .....	575
Триггеры BEFORE и AFTER .....	576
Триггеры INSTEAD OF .....	576
Триггеры схемы .....	577
Триггеры уровня базы данных .....	577
Комбинированные триггеры .....	577
Синтаксис триггеров .....	577
Объединение типов триггеров DML .....	580
Установка вставляемых значений .....	581
Сопровождение дублированных данных .....	582
Настройка сбойных ситуаций .....	583
Вызов процедур из триггеров .....	586
Именованые триггеров .....	586
Создание триггеров для событий DDL .....	586
Создание триггеров событий базы данных .....	591
Создание комбинированных триггеров .....	592
Активизация и отключение триггеров .....	593
Замена триггеров .....	594
Удаление триггеров .....	595
<b>Глава 35 Процедуры, функции и пакеты .....</b>	<b>596</b>
Необходимые системные привилегии .....	597
Необходимые привилегии на таблицы .....	598
Процедуры как функции .....	599
Процедуры как пакеты .....	599
Синтаксис команды create procedure .....	599
Синтаксис команды create function .....	601
Обращение в процедурах к удаленным таблицам .....	603
Отладка процедур .....	604
Создание собственных функций .....	606
Настройка сбойных ситуаций .....	607
Именованые процедур и функций .....	609
Синтаксис команды create package .....	609
Просмотр исходного кода объектов типа процедуры .....	612
Компиляция процедур, функций и пакетов .....	613
Замена процедур, функций и пакетов .....	614
Удаление процедур, функций и пакетов .....	614

<b>Глава 36</b>	<b>Использование динамического SQL и DBMS_SQL</b> .....	616
	Использование EXECUTE IMMEDIATE .....	616
	Использование переменных связи .....	618
	Использование DBMS_SQL .....	619
	OPEN_CURSOR .....	620
	PARSE .....	620
	BIND_VARIABLE и BIND_ARRAY .....	621
	EXECUTE .....	621
	DEFINE_COLUMN .....	621
	FETCH_ROWS, EXECUTE_AND_FETCH и COLUMN_VALUE .....	622
	CLOSE_CURSOR .....	623
<b>Глава 37</b>	<b>Настройка PL/SQL</b> .....	624
	Настройка SQL .....	624
	Шаги по настройке PL/SQL .....	625
	Использование DBMS_PROFILER для выявления проблем .....	626
	Использование функций PL/SQL для массовых операций .....	631
	forall .....	631
	Оператор bulk collect .....	633
<b>Часть V</b>	<b>Объектно-реляционные базы данных</b> .....	637
<b>Глава 38</b>	<b>Реализация типов, объектных представлений и методов</b> .....	638
	Работа с типами объектов .....	638
	Защита для объектных типов данных .....	639
	Индексирование атрибутов объектного типа .....	642
	Реализация объектных представлений .....	644
	Манипулирование данными посредством объектных представлений .....	647
	Использование триггеров INSTEAD OF .....	647
	Методы .....	650
	Синтаксис создания методов .....	651
	Управление методами .....	652
<b>Глава 39</b>	<b>Коллекторы</b> .....	654
	Массивы переменной длины .....	654
	Создание массива переменной длины .....	654
	Описание массива переменной длины .....	655
	Вставка записей в массив переменной длины .....	656
	Выборка данных из массивов переменной длины .....	658
	Вложенные таблицы .....	661
	Определение табличных пространств для вложенных таблиц .....	662

Вставка записей во вложенные таблицы .....	662
Выполнение запросов к вложенным таблицам .....	664
Дополнительные функции для вложенных таблиц и массивов переменной длины .....	666
Вопросы управления для вложенных таблиц и массивов переменной длины .....	666
Изменчивость в коллекторах .....	667
Размещение данных .....	668
<b>Глава 40 Использование больших объектов .....</b>	<b>669</b>
Имеющиеся типы данных .....	669
Назначение области хранения для данных LOB .....	671
Манипулирование и выбор значений LOB .....	673
Инициализация значений .....	675
Команда insert с подзапросами .....	677
Обновление значений LOB .....	677
Использование строковых функций для манипулирования значениями LOB .....	677
Использование DBMS_LOB для манипуляции значениями LOB .....	678
Удаление LOB .....	696
<b>Глава 41 Расширенные объектно-ориентированные концепции .....</b>	<b>697</b>
Строковые объекты в сравнении с объектами-столбцами .....	697
Объектные таблицы и OID .....	698
Вставка строк в объектные таблицы .....	699
Выборка значений из объектных таблиц .....	700
Операции update и delete для объектных таблиц .....	700
Функция REF .....	701
Использование функции Deref .....	702
Функция VALUE .....	704
Недействительные ссылки .....	705
Объектные представления с REF .....	706
Краткий обзор объектных представлений .....	706
Объектные представления, включающие ссылки .....	707
Объектный PL/SQL .....	711
Объекты в базе данных .....	712
<b>ТОМ II</b>	
<b>Часть VI Использование Java в Oracle .....</b>	<b>1</b>
<b>Глава 42 Введение в Java .....</b>	<b>2</b>
Сравнение Java и PL/SQL: Общее представление .....	2



Первое знакомство.....	3
Объявления.....	4
Исполняемые команды.....	4
Условные операторы.....	5
Циклы.....	9
Обработка исключительных ситуаций.....	11
Зарезервированные слова.....	12
Классы.....	13
<b>Глава 43 Программирование на JDBC.....</b>	<b>18</b>
Использование классов JDBC.....	19
Использование JDBC для манипуляции данными.....	22
<b>Глава 44 Хранимые процедуры Java.....</b>	<b>26</b>
Загрузка класса в базу данных.....	28
Как получить доступ к классу.....	30
Непосредственный вызов хранимых процедур Java.....	32
Где выполнять команды.....	32
<b>Часть VIII Справочное руководство программиста.....</b>	<b>35</b>
<b>Глава 45 Справочное руководство программиста по словарю данных Oracle.....</b>	<b>36</b>
Замечание о номенклатуре.....	37
Новые представления, появившиеся в Oracle Database 11g.....	37
Дорожные карты: DICTIONARY (DICT) и DICT_COLUMNS.....	43
Объекты, из которых можно делать выборку: таблицы (и столбцы), представления, синонимы и последовательности.....	45
Каталог: USER_CATALOG (CAT).....	45
Объекты: USER_OBJECTS (OBJ).....	46
Таблицы: USER_TABLES (TABS).....	47
Столбцы: USER_TAB_COLUMNS (COLS).....	49
Представления: USER_VIEWS.....	50
Синонимы: USER_SYNONYMS (SYN).....	53
Последовательности: USER_SEQUENCES (SEQ).....	54
Мусорная корзина – Представления USER_RECYCLEBIN и DBA_RECYCLEBIN.....	54
Ограничения и комментарии.....	55
Ограничения: USER_CONSTRAINTS.....	55
Столбцы ограничений: USER_CONS_COLUMNS.....	57
Исключительные ситуации ограничений: EXCEPTIONS.....	58
Комментарии к столбцам: USER_COL_COMMENTS.....	60
Индексы и кластеры.....	60

Индексы: USER_INDEXES (IND) .....	60
Индексированные столбцы: USER_IND_COLUMNS .....	63
Кластеры: USER_CLUSTERS (CLU).....	64
Столбцы кластера: USER_CLU_COLUMNS .....	64
Абстрактные типы данных и LOB .....	65
Абстрактные типы данных: USER_TYPES.....	65
LOBs: USER_LOBS .....	67
Связи базы данных и материализованные представления .....	68
Связи базы данных: USER_DB_LINKS.....	68
Материализованные представления .....	69
Журналы материализованных представлений: USER_MVIEW_LOGS.....	71
Триггеры, процедуры, функции и пакеты.....	71
Триггеры: USER_TRIGGERS .....	71
Процедуры, функции и пакеты: USER_SOURCE .....	72
Измерения .....	74
Выделение и использование дисковой памяти, включая разделы и подразделы.....	75
Табличные пространства: USER_TABLESPACES.....	75
Квоты дискового пространства: USER_TS_QUOTAS.....	76
Сегменты и экстененты: USER_SEGMENTS и USER_EXTENTS.....	76
Разделы и подразделы.....	77
Свободное пространство: USER_FREE_SPACE .....	80
Пользователи и привилегии .....	80
Пользователи: USER_USERS.....	80
Лимиты ресурсов: USER_RESOURCE_LIMITS .....	81
Привилегии на таблицы: USER_TAB_PRIVS .....	81
Привилегии на столбцы: USER_COL_PRIVS.....	82
Системные привилегии: USER_SYS_PRIVS.....	82
Роли .....	82
Аудит .....	84
Разное .....	86
Мониторинг: Динамические таблицы производительности V\$ .....	87
CHAINED_ROWS .....	87
PLAN_TABLE .....	87
Взаимозависимости: USER_DEPENDENCIES и IDEPTREE .....	88
Представления только для DBA .....	88
Oracle Label Security.....	88
Представления прямой загрузки SQL*Loader.....	89
Представления поддержки глобализации.....	89

Библиотеки.....	89
Гетерогенные сервисы.....	89
Операторы и типы индексов.....	90
Планы.....	90
Консультанты.....	90
Планировщики.....	91
<b>Глава 46 Руководство по настройке приложений и SQL.....</b>	<b>92</b>
Новые особенности настройки в Oracle Database 11g.....	92
Новые особенности настройки в Oracle 11g.....	93
Настройка – лучший способ применения.....	94
Делайте как можно меньше.....	95
Делайте как можно проще.....	99
Дайте БД необходимую информацию.....	100
Увеличьте по максимуму производительность среды.....	101
Разделите данные.....	103
Тестируйте правильно.....	104
Создание и чтение пояснений к запросам.....	107
Использование set autotrace on.....	107
Использование команды Explain Plan.....	111
Основные операции в рамках разьяснения плана выполнения запросов.....	113
Операция TABLE ACCESS FULL.....	113
Операция TABLE ACCESS BY INDEX ROWID.....	113
Соответствующие инструкции.....	113
Операции, использующие индексы.....	114
Когда используются индексы.....	116
Операции с наборами данных.....	122
Операции, выполняющие объединения.....	130
Как Oracle обрабатывает объединения более чем двух таблиц.....	131
Вопросы параллельности и кэширования.....	139
Применение сохраненных планов.....	140
Обзор.....	142
<b>Глава 47 Кэш Результатов SQL и клиентский кэш.....</b>	<b>143</b>
Настройки параметров базы данных для кэша результатов SQL.....	151
Пакет DBMS_RESULT_CACHE.....	152
Представления кэша результатов SQL.....	153
Дополнительные детали кэша результатов SQL.....	153
Интерфейс обращений Oracle (OCI), клиентский кэш.....	154
Ограничения для кэша запросов OCI.....	154

<b>Глава 48 Примеры настройки .....</b>	<b>156</b>
Учебный пример 1: Ожидания, ожидания и еще раз ожидания.....	156
Учебный пример 2: Запросы, «убивающие» приложения.....	160
Использование события трассировки 10053.....	163
Учебный пример 3: Долго выполняющиеся пакетные задания .....	165
<b>Глава 49 Дополнительные возможности архитектуры – программы DB Vault, Content DB и Records DB .....</b>	<b>169</b>
Oracle Database Vault .....	169
Новые концепции Oracle database Vault.....	170
Отключение Oracle Database Vault.....	171
Включение Oracle Database Vault .....	172
Замечания по установке Oracle Database Vault .....	173
Oracle Content Database Suite.....	177
Система хранения.....	177
Управление документами.....	178
Безопасность пользователей.....	178
Oracle Records Database .....	180
<b>Часть VII Кластерный Oracle – Вычислительная сеть .....</b>	<b>183</b>
<b>Глава 50 Oracle Real Application Clusters .....</b>	<b>184</b>
Прединсталляционные шаги .....	184
Инсталляция RAC .....	185
Дисковая память.....	186
Параметры инициализации.....	187
Запуск и остановка экземпляров RAC.....	190
Прозрачное преодоление последствий сбоя приложения .....	192
Сервисы.....	193
Добавление в кластер узлов и экземпляров.....	194
<b>Глава 51 Справочное руководство по администрированию базы данных .....</b>	<b>195</b>
Создание базы данных.....	196
Использование Oracle Enterprise Manager.....	196
Запуск и остановка базы данных.....	197
Задание размеров и управление областями памяти .....	198
Файл параметров инициализации .....	201
Выделение пространства для объектов и управление им .....	201
Результаты применения фразы storage .....	202
Табличные сегменты.....	204
Сегменты индексов.....	205
Отмена действий, управляемая системой.....	205

Временные сегменты.....	206
Свободное пространство.....	208
Оценка размеров объектов базы данных.....	209
Мониторинг табличного пространства отмены.....	212
Автоматизация управления дисковой памятью.....	212
Конфигурирование ASM.....	213
Управление дисковым пространством в сегментах.....	214
Переносимые табличные пространства.....	215
Генерация набора переносимых табличных пространств.....	215
Включение набора переносимых табличных пространств.....	216
Выполнение резервного копирования.....	217
Data Pump Export и Import.....	218
Автономное резервное копирование.....	218
Оперативное резервное копирование.....	219
Утилита Recovery Manager.....	223
Куда двигаться дальше.....	225
<b>Глава 52 Справочное руководство программиста по использованию XML в Oracle.....</b>	<b>226</b>
Определения типа, элементы и атрибуты документа.....	226
Схема XML.....	230
Использование XSU для выбора, вставки, обновления и удаления значений XML.....	233
Обработка операций insert, update и delete с помощью XSU.....	234
XSU и Java.....	235
Настройка процесса запроса.....	236
Использование XMLType.....	237
Прочие свойства.....	240
<b>Часть VIII Краткий путеводитель.....</b>	<b>241</b>

# **Часть VI**

## **ИСПОЛЬЗОВАНИЕ JAVA В ORACLE**

# ГЛАВА 42

## ВВЕДЕНИЕ В JAVA

В этой главе представлен обзор языка Java в той степени, в какой этот язык имеет отношение к приложениям базы данных Oracle. Java применяется во множестве случаев, помимо приложений Oracle, но многие свойства этого языка не используются большинством разработчиков Oracle. Цель данной главы состоит в том, чтобы предоставить разработчикам, которые знакомы с SQL и PL/SQL, базовые знания о структурах языка Java. Эта глава не дает исчерпывающего обзора Java, а является скорее кратким обзором компонентов языка Java, наиболее часто используемых разработчиками Oracle.

Хотя во многих случаях PL/SQL и Java хорошо соотносятся друг с другом, существуют значительные различия в терминологии и использовании, поскольку эти два языка программирования разрабатывались независимо друг от друга. Более того, объектно-ориентированные возможности PL/SQL не входили в его исходную реализацию, в то время как Java с самого начала был объектно-ориентированным языком программирования. Чтобы эффективно использовать Java, необходимо применить подход, отличный от того, который использовался в случае PL/SQL.

Глава 32 представляет собой введение в структуры языка PL/SQL и управление потоком выполнения. Данная глава повторяет этот подход: здесь показаны структуры, которые использует Java, и существующие базовые методы управления потоком выполнения. Где возможно, показаны также соответствующие структуры PL/SQL. Прежде чем читать эту главу, необходимо познакомиться с PL/SQL (глава 32), пакетами, функциями и процедурами (глава 38), пользовательскими типами и методами (глава 38).

### **Сравнение Java и PL/SQL: Общее представление**

При сравнении PL/SQL и Java значительные различия в терминологии между двумя языками проявляются уже на уровне базовой структуры PL/SQL, т.е. блока. В PL/SQL блок является структурированной частью кода, который имеет раздел объявлений, раздел исполняемых команд и раздел обработки исключительных ситуаций. Структуру блока PL/SQL можно представить следующим образом:

```
□ declare
  <раздел объявлений>

begin
  <исполняемые команды>
exception
  <обработка исключительных ситуаций>
end;
```

В Java термин блок относится к значительно меньшему подмножеству кода. Блок является совокупностью операторов, заключенных в фигурные скобки. Например, следующий псевдокод содержит два блока Java:

```
□ if (некоторое условие) {
  блок кода для выполнения, если условие удовлетворяется
}
else {
  блок кода для выполнения, если условие не удовлетворяется
}
```

В PL/SQL весь раздел кода будет просто частью большего блока, в Java он содержит два отдельных блока. В этой главе все ссылки на «блоки» имеют в виду блоки Java, если явно не определено иначе.

Процедура PL/SQL ограничена операторными скобками `begin` и `end`, а метод Java ограничивается открывающей и закрывающей фигурными скобками. Эти же скобки используются для отделения блоков рассматриваемого кода или как индикатор открытия и закрытия условных сегментов.

Вторым существенным различием между PL/SQL и Java является объявление переменных. В блоке PL/SQL переменные определяются до начала раздела исполняемых команд. В программе Java можно определять переменные, когда они понадобятся в программе. В программе Java нет раздела объявлений, аналогичного тому, который используется в блоках PL/SQL.

В данной главе будут показаны и другие различия между PL/SQL и Java. Важно помнить, что Java не заменяет PL/SQL в приложениях - вы можете продолжать использовать PL/SQL для приложений Oracle. Необходимо протестировать производительность PL/SQL и Java, прежде чем принять решение, какого направления придерживаться для операций извлечения данных.

## Первое знакомство

Чтобы воспользоваться примерами из этого раздела, необходимо иметь копию пакета разработки Java (Java Development Kit, JDK), который доступен для бесплатной загрузки на сайте <http://java.sun.com>. Необходимо установить JDK на сервер, на котором будут выполняться программы Java.

---

**Примечание** JDK устанавливается как часть стандартной инсталляции Oracle Database. Убедитесь, что у вас установлено как программное обеспечение ядра базы данных, так и связанное с ним клиентское программное обеспечение.

---



## Объявления

В Java переменные можно объявлять по мере необходимости. Переменная имеет имя и тип данных и используется для хранения значения данных во время выполнения программы. Переменная имеет также область действия, позволяющую ей быть общедоступной или частной – аналогично тому, как процедуры в пакете могут иметь частные переменные.

Вот примеры объявления переменных в Java:

```
❑ char aCharVariable = 'J';
  boolean aBooleanVariable = false;
```

По соглашению, имена переменных Java всегда начинаются с буквы нижнего регистра, как показывает этот пример. Здесь используются типы данных CHAR и BOOLEAN и каждой переменной присваивается начальное значение. Знаком присваивания в Java является =, в противоположность := или => в PL/SQL. Каждое объявление заканчивается точкой с запятой.

Имеющиеся примитивные типы данных Java перечислены в таблице 42.1.

**Таблица 42.1.** Примитивные типы данных Java

Тип данных	Описание
Byte	Целое длиной в один байт
Short	Короткое целое
Int	Целое
Long	Длинное целое
Float	Действительное число с плавающей точкой одинарной точности
Double	Действительное число с плавающей точкой двойной точности
Char	Одиночный символ Юникода
Boolean	Булево значение – true или false

Кроме примитивных типов данных, можно использовать ссылочные типы данных, которые основываются на содержимом переменных. Для строк символов переменной длины не существует примитивного типа данных - Java предоставляет для этого класс String.

## Исполняемые команды

Переменным можно присваивать значения с помощью выражений и операторов. Java поддерживает следующие арифметические операторы:

Оператор	Описание
*	Умножение
/	Деление
+	Сложение
-	Вычитание
%	Сравнение по модулю

Кроме этих операторов, для упрощения кодирования можно использовать унарные операторы Java. Например, можно увеличивать переменные с помощью оператора ++:

```
❑ aLoopCounter = aLoopCounter++;
```

или можно просто написать:

```
❑ aLoopCounter++
```

---

**Примечание** Оператор ++ называется унарным, потому что он имеет только один операнд. Если оператору требуется два операнда (как в случае \*), он называется бинарным.

---

В PL/SQL это можно было записать как:

```
❑ aLoopCounter := aLoopCounter + 1;
```

Если оператор ++ помещается перед именем переменной, то он является оператором прединкремента (предварительного увеличения), а не оператором постинкремента (увеличения после):

```
❑ int anotherCounter = ++LoopCounter;
```

В этом примере переменной anotherCounter присваивается увеличенное значение aLoopCounter. В противоположность этому

```
❑ int anotherCounter = aLoopCounter++;
```

присваивает anotherCounter значение aLoopCounter до увеличения. Значение переменной можно уменьшать с помощью унарного оператора -:

```
❑ aLoopCounter = aLoopCounter--;
```

Можно также комбинировать присваивание значений с выполняемой операцией. Например, вместо записи

```
❑ aNumberVariable = aNumberVariable * 2;
```

можно написать:

```
❑ aNumberVariable *= 2;
```

Можно выполнять аналогичные комбинации, используя /=, %=, +=, и -=.

При выполнении нескольких арифметических операций можно использовать скобки для четкого указания порядка, в котором операции должны выполняться:

```
❑ aNumberVariable = (aNumberVariable * 2) + 2;
```

Завершение этих операций точкой с запятой делает их *операторами* - законченными единицами выполнения.

## Условные операторы

Переменные можно оценивать с помощью выражений и операторов. Для этого используются один или несколько фундаментальных классов, которые являются частью Java. Перечисление всех этих методов выходит за рамки нашей книги. Их можно найти на сайте компании Sun с документацией Java или в одной из множества книг по Java, которые сейчас доступны.

---

**Примечание** Число предоставляемых классов существенно увеличивается с каждой новой версией Java.

---

В следующем примере метод `Character.isUpperCase` используется для вычисления объявленной ранее переменной `aCharVariable`:

```
❑ if(Character.isUpperCase(aCharVariable)) {
    команды для выполнения
}
```

Если значение `aCharVariable` представлено символом верхнего регистра, будут выполнены команды блока, иначе управление будет передано следующей части программы.

Далее следует общий синтаксис фраз `if`:

```
❑ if (выражение) {
    оператор
}
```

Отметим отсутствие фразы `then`. Если оцениваемое выражение является `true`, следующий блок выполняется автоматически. Также отсутствует фраза `end if`. Блоки Java просто завершаются закрывающей скобкой оператора.

Могут присутствовать фразы `else` для оценивания других условий:

```
❑ if (выражение) {
    оператор
}
else {
    оператор
}
```

Для проверки нескольких условий можно использовать фразу `else if` (для оценивания каждого из них по очереди) и закончить все фразой `else`. Следующий листинг иллюстрирует использование фразы `else if`:

```
❑ if (aCharVariable == 'V') {
    оператор
}
else if (aCharVariable == 'J') {
    оператор
}
else {
    оператор
}
```

---

**Примечание** PL/SQL поддерживает использование для выбора между несколькими альтернативами фраз `elsif`. В одной фразе `if` может быть несколько фраз `elsif`.

---

Кроме фраз `if` для условных операторов, Java использует фразу `switch`. Вместе с фразой `break` и метками операторов фраза `switch` может имитировать функциональность фразы `goto` (которые Java не поддерживает). Рассмотрим сначала простой пример `switch`. С помощью оператора `case` оценивается несколько значений переменной `aMonth`. В зависимости от значения `aMonth` выводится текстовое название месяца, и обработка покидает блок `switch` с помощью `break`.

```
❑ int aMonth=2;
  switch (aMonth) {
  case 1:
    System.out.println("January");
    break;
  case 2:
    System.out.println("February");
    break;
  case 3:
    System.out.println("March");
    break;
  default:
    System.out.println("Out of range");
    break;
  }
```

---

**Примечание** В Java аргументами оператора case должны быть целые числа.

---

Код такого вида значительно сложнее, чем простой TO\_CHAR, но он иллюстрирует использование switch. Оператор switch получает на входе переменную aMonth и оценивает ее значение. Если значение соответствует значению case, вызывается метод System.out.println для вывода названия месяца, и управление обработкой покидает блок switch с помощью break. Если значение не соответствует ни одной из фраз case, выполняется вариант default.

Куда передается управление? По умолчанию оно переходит к следующему разделу программы. Однако можно создать метки для разделов кода и передать имя такой метки во фразу break. Следующий листинг показывает пример метки и фразы break:

```
❑ somelabel:
  if (aCharVariable == 'V') {
  оператор
  }
  else {
  оператор
  }
  int aMonth=2;
  switch (aMonth) {
  case 1:
    System.out.println("January");
    break somelabel;
  case 2:
    System.out.println("February");
    break someotherlabel;
  case 3:
    System.out.println("March");
    break somethirdlabel;
  default:
    System.out.println("Out of range");
    break;
  }
```

В этом примере оценивается фраза `if`, за которой следует фраза `switch`. Переменная `aMonth` не объявляется до тех пор, пока она не потребуется. Здесь переменной `aMonth` присвоено значение 2, поэтому программа будет выводить слово «February», а затем перейдет в секцию кода, идентифицированную меткой `someotherlabel`. Если бы значение `aMonth` было равно 1, была бы повторно выполнена фраза `if` в начале кода.

Java поддерживает также тернарный оператор - оператор, который получает три операнда и функция которого аналогична функции `DECODE`. Действуя как встроенная комбинация `if-else`, тернарный оператор оценивает выражение. Если выражение оценивается как `true`, возвращается второй операнд, иначе возвращается третий. Синтаксис имеет следующий вид:

❑ выражение ? операнд1 : операнд2

Следующий листинг показывает пример использования тернарной операции:

❑ `aStatusVariable = (aCharVariable == 'V') ? «OK» : «Invalid»;`

В этом примере оценивается выражение

❑ `(aCharVariable == 'V')`

Если оно справедливо, возвращается «OK», иначе возвращается «Invalid».

Тернарный оператор можно использовать для упрощения применения функции `GREATEST`:

❑ `double greatest = (a > b) ? a : b;`

В этом примере оценивается выражение `(a > b)`, где `a` и `b` являются именами переменных. Если это выражение будет `true`, возвращается значение `a`, иначе возвращается значение `b`.

---

**Примечание** В Java имеется метод “`greatest`” (наибольший) для цифровых значений - метод `Max` в классе `java.lang.Math`. Пример с тернарным оператором можно переписать следующим образом: `double greatest = Math.max(a, b);`

---

Можно объединить несколько логических проверок с помощью операций `AND` и `OR`. В Java операция `AND` представляется с помощью оператора `&&`:

❑ `if (aCharVariable == 'V' && aMonth == 3) {`  
оператор  
`}`

Оператор `OR` представляется с помощью `||`:

❑ `if (aCharVariable == 'V' || aMonth == 3) {`  
оператор  
`}`

По соображениям производительности операции `&&` и `||` выполняются, только когда в этом возникает необходимость; второе выражение не оценивается, если результат определяется первым выражением. В случае оператора `AND`, если первое значение будет `false`, то оценивание заканчи-

вается в этом месте и возвращается булево значение `false`. В случае оператора `OR`, если первое выражение будет `false`, то второе выражение все равно будет оцениваться, так как только одно из проверяемых условий должно быть `true`, чтобы все выражение было оценено как `true`. Если первое значение оператора `OR` будет `true`, последующие выражения оценивать не потребуется, и будет возвращено `true`.

Операторы `&` и `|` являются побитовыми `AND` и `OR`, и они не сокращаются, как операторы `&&` и `||`. Все выражения будут всегда оцениваться. Эта функциональность может быть важной, если проверяются возвращаемые значения вызовов методов и необходимо убедиться, что методы выполнены.

## Циклы

Java поддерживает три основных типа циклов: `WHILE`, `DO-WHILE` и `FOR`. Так как Java не является расширением SQL, то в отличие от PL/SQL курсорные циклы `FOR` не поддерживаются.

### Циклы `WHILE` и `DO-WHILE`

Фраза `while` оценивает условие: если значением условия будет `true`, выполняется соответствующий блок операторов. Синтаксис цикла `WHILE` следующий:

```
❑ while (выражение) {
    оператор
    оператор
}
```

Цикл `WHILE` повторно выполняет блок операторов (секция между скобками `{` и `}`), пока выражение не перестанет быть `true`:

```
❑ int aNumberVariable = 3;
    while (aNumberVariable < 7) {
```

некоторые операторы для обработки

```
❑ aNumberVariable++;
}
```

В этом примере создается и инициализируется переменная счетчик (`aNumberVariable`). Затем значение переменной вычисляется во фразе `while`. Если это значение меньше 7, выполняется соответствующий блок операторов. Как часть этого блока переменная получает приращение. Когда блок завершается, переменная снова оценивается, и цикл продолжается.

---

**Примечание** Примеры обработки цикла `WHILE` приведены далее в разделе “Классы”.

---

Можно также записать это как цикл `DO-WHILE`:

```
❑ int aNumberVariable = 3;
    do {
```

некоторые операторы для обработки

```
❑ aNumberVariable++;
    } while (aNumberVariable < 7);
```

В цикле DO-WHILE значение выражения не оценивается, пока блок не будет обработан хотя бы один раз.

### Циклы FOR

Цикл FOR можно использовать для повторного выполнения блока кода. В цикле FOR определяется начальное значение, конечное значение и шаг счетчика цикла. Счетчик цикла будет увеличиваться на заданное значение шага, пока не будет достигнуто конечное значение. Ниже представлен синтаксис цикла FOR:

- `for (начальное значение; условие окончания; шаг цикла) {  
оператор;  
}`

Пример с циклом WHILE из предыдущего раздела можно переписать с помощью фразы `for`:

- `for (int aNumberVariable=3; aNumberVariable < 7; aNumberVariable++) {  
оператор для обработки  
}`

Версия с фразой `for` для этого примера значительно короче, чем версия с `while`. В цикле FOR объявляется и инициализируется переменная `aNumberVariable`. Пока значение переменной не превышает 7, блок будет выполняться. При каждом проходе цикла значение переменной увеличивается на 1 с помощью унарного оператора `++`.

Внутри цикла можно использовать фразу `continue` для перехода к другому оператору цикла. Если использовать просто фразу `continue`, процесс выполнения перейдет в конец тела цикла и выполнит проверку окончания цикла. Если `continue` используется с меткой (как было показано в разделе о фразе `switch`), обработка продолжится со следующей итерации помеченного цикла.

### Курсоры и циклы

В PL/SQL можно использовать курсорные циклы FOR для итераций по результатам запроса. Циклы WHILE и FOR из Java можно использовать для моделирования функциональности курсорного цикла FOR из PL/SQL.

Oracle предоставляет образцы файлов, которые сопровождают установку программного обеспечения Oracle, с большим количеством примеров, иллюстрирующих стандарты подходов к кодированию. Установка клиента использует Oracle Universal Installer для загрузки драйверов и демонстрации файлов, в том числе те, которые поддерживают развитие JDBC. После установки клиентского программного обеспечения, примеры JDBC становятся Java-связанными подкаталогами домашнего каталога программного обеспечения Oracle. Точное количество предоставляемых файлов может меняться от версии к версии.

---

**Примечание** Файл README в подкаталоге /JDBC домашнего каталога программного обеспечения Oracle предоставляет обширную информацию о поддерживаемой версии, настройке среды переменной платформы и поддерживаемых функций. Очень важно читать и следовать этим инструкциям, чтобы эффективно использовать Java от Oracle на вашей платформе.

---

В следующем примере показан фрагмент кода из примера, приведенного из установки клиента Oracle:

```

❑ Statement stmt = conn.createStatement();
  ResultSet rset = stmt.executeQuery ("select FIRST_NAME, "
  + "LAST_NAME from EMPLOYEES");
  // Итерации по множеству результатов и печать имен сотрудников
  while (rset.next())
    System.out.println (rset.getString (1) + " " + rset.getString(2));

```

В этом примере предполагается, что соединение с базой данных уже было установлено. В первой строке создается переменная `stmt` на основе переменной соединения с именем `conn` (не показанной в этом частичном примере). Во второй строке создается переменная `rset` на основе множества результатов, которое возвращает запрос `stmt`. Вслед за комментарием (который начинается с `//`) следует фраза `while`, извлекающая из множества результатов каждую строку. Если операция выборки (`fetch`) извлекает строку из множества результатов, печатается `ENAME` из этой строки `EMP`, иначе цикл заканчивается.

С точки зрения программирования следует быть предельно аккуратным с возвращаемыми данными и указывать имена столбцов, которые должны быть возвращены.

Примеры использования JDBC (Java Database Connectivity – средство организации доступа Java-приложений к базам данных в сети; см. главу 43), предоставленные Oracle, предлагают образцы запросов, включающие вызовы процедур, LOB и DML. В большинстве случаев код Java будет выполняться аналогично предыдущему примеру: напишите процесс управления потоком выполнения на Java и передайте ему оператор для выполнения. На базе результата этого оператора можно выполнять различные действия, используя фразы `while`, `for`, `if`, `break`, `continue` и `switch`.

## Обработка исключительных ситуаций

В PL/SQL имеется единый блок обработки исключительных ситуаций (`EXCEPTION`); в Java вы можете иметь столько блоков `try/catch`, сколько потребуется. Java предоставляет богатый набор средств обработки ошибок и позволяет создавать для этого сложные процедуры. В PL/SQL исключительные ситуации возбуждаются (`raise`), в Java исключительные ситуации порождаются (`throw`). Если порождена исключительная ситуация, для ее перехвата и соответствующей обработки необходимо использовать оператор Java `catch`.

Синтаксис обработки ошибок в Java основывается на трех блоках: `try`, `catch` и `finally`. Блок `try` включает операторы, которые могут породить исключительную ситуацию. Блок `catch`, следующий сразу за блоком `try`, связывает обработчики событий с исключительными ситуациями, которые могут породиться блоком `try`. Блок `finally` очищает все системные ресурсы, которые не были должным образом очищены во время обработки блока `catch`. Общая структура имеет следующий вид:

```

❑ try {
    операторы
}

```



```

catch ( ) {
    операторы
}
catch ( ) {
    операторы
}
finally {
    операторы
}

```

Рассмотрим, например, следующий код Java:

```

❑ try {
Statement stmt = conn.createStatement ();
stmt.execute ("drop table plsqttest");
}
catch (SQLException e) {
}

```

В этом примере создается переменная `stmt` для выполнения команды `drop table`. Если выполнение завершается аварийно (например, таблица не существует), как будет обрабатываться эта ошибка? Оператор `catch` приказывает Java обработать его с помощью объекта исключительной ситуации `SQLException` - стандартной части реализации SQL в Java. Фраза `catch` имеет два параметра (объект исключения и имя переменной), и за ней может при необходимости следовать блок операторов для выполнения.

Блок `try` может содержать несколько операторов, или можно создать отдельные блоки `try` для каждого оператора. Как правило, легче управлять обработкой ошибок, если объединить блоки `catch`, поэтому объединение блоков `try` поможет управлять кодом при его изменении и увеличении в размерах. Оператор `when others` в PL/SQL – это, по сути, то же самое, что и `catch(Throwable th)` в Java.

Блок `finally` очищает состояние текущего раздела кода, прежде чем передать управление любым последующим частям программы. Во время выполнения содержимое блока `finally` всегда выполняется независимо от результата блока `try`. Например, можно использовать блок `finally` для закрытия соединения с базой данных.

## Зарезервированные слова

Зарезервированные слова Java перечислены в таблице 42.2. Эти слова нельзя использовать в качестве имен классов, методов или переменных.

**Таблица 42.2.** Зарезервированные слова Java

<code>abstract</code>	<code>else</code>	<code>interface</code>	<code>super</code>
<code>boolean</code>	<code>extends</code>	<code>long</code>	<code>switch</code>
<code>break</code>	<code>false</code>	<code>native</code>	<code>synchronized</code>
<code>byte</code>	<code>final</code>	<code>new</code>	<code>this</code>
<code>case</code>	<code>finally</code>	<code>null</code>	<code>throw</code>
<code>catch</code>	<code>float</code>	<code>package</code>	<code>throws</code>

**Таблица 42.2.** *Зарезервированные слова Java (продолжение)*

char	for	private	transient
class	goto	protected	true
const	if	public	try
continue	implements	return	void
default	import	short	volatile
do	instanceof	static	while
double	int	strictfp	

## Классы

В предыдущих разделах были показаны базовые синтаксические структуры Java. В данном разделе показано, как этот синтаксис применяется для создания и использования объектов. В следующем примере представлена простая программа для печати слова «Oracle»:

```
❑ public class HelloOracle {
    public static void main (String[] args) {
        System.out.println("Oracle");
    }
}
```

Этот пример создает класс с именем HelloOracle, в котором создается открытый метод с именем main. Метод main печатает слово «Oracle». Это значительно сложнее, чем

```
❑ select 'Oracle' from dual;
```

Однако версия Java имеет свойства, которые отсутствуют в версии SQL. HelloOracle является классом, и поэтому его методы могут вызываться из других классов. HelloOracle может иметь несколько методов. В этом примере он имеет только метод main.

В объявлении метода main (public static void main) имеется ряд ключевых слов, которые уже нам знакомы:

public	Определение класса как public (общедоступный) позволяет всем классам вызывать этот метод.
static	Объявляет, что это метод класса; используется для объявления методов класса. Дополнительной возможностью является ключевое слово final для методов и переменных, значения которых не могут изменяться (аналогично параметру constant в PL/SQL).
void	Определяет тип данных возвращаемого значения процедуры. Так как эта процедура не имеет возвращаемых значений, ее тип будет void.

В нашем примере методу дано имя `main`, так как это упрощает выполнение метода класса. Теперь, когда класс создан, его можно откомпилировать и загрузить.

---

**Примечание** Для того чтобы откомпилировать и выполнить класс Java, необходимо иметь на своем сервере JDK. В следующих примерах предполагается, что двоичные файлы, являющиеся частью этого пакета, расположены в каталоге, который автоматически доступен через переменную окружения `PATH`, а каталог, содержащий файлы с расширением `.class`, включен в переменную окружения `CLASSPATH` (смотрите файл `/jdc/README` для более подробной настройки переменной среды). Используемые здесь программы (`javac` и `java`) расположены в подкаталоге `/bin` каталога программного обеспечения JDK.

---

Сначала сохраним программу как обычный текстовый файл с именем `HelloOracle.java`. Затем откомпилируем его:

```
❑ javac HelloOracle.java
```

На сервере будет создан новый файл с именем `HelloOracle.class`. Затем можно выполнить этот класс:

```
❑ java HelloOracle
```

Будет выведен результат:

```
❑ Oracle
```

Как показано в примере `HelloOracle`, для класса сначала задается имя вместе с другими необязательными атрибутами, связанными с его наследованием атрибутов других классов. В теле класса определяются все переменные и методы класса. Как и у абстрактных типов данных в `Oracle`, у классов Java имеются связанные с ними методы манипулирования данными.

Рассмотрим более сложный пример. В следующем листинге приведена программа Java, которая вычисляет площадь круга, когда в качестве входного параметра задается значение радиуса (версию PL/SQL см. в главе 32).

```
❑ // AreaOfCircle.java (Площадь круга.java)
//
public class AreaOfCircle {
public static void main(String[] args) {
try {
int input = Integer.parseInt(args[0]);
double result=area(input);
System.out.println(result);
}
catch (ArrayIndexOutOfBoundsException oob) {
System.out.println("You did not provide the radius of the circle.");
System.out.println("Usage: java AreaOfCircle 10");
}
catch (NumberFormatException nfe) {
System.out.println("Enter a valid number for the radius.");
System.out.println("Usage: java AreaOfCircle 10");
}
} /// main
public static double area (int rad) {
```

```
double pi=3.1415927;
double areacircle=pi*rad*rad;
return areacircle;
} /// area
} /// AreaOfCircle
```

---

**Примечание** Не забудьте включить для каждого блока закрывающую скобку }. В показанных в этой главе примерах закрывающая скобка всегда находится на отдельной строке, чтобы упростить отладку и обработку ошибок управления потоком выполнения.

---

Прежде всего отметим, что имя класса и имя файла должны быть одинаковыми. В нашем случае класс имеет имя AreaOfCircle, поэтому этот текст сохраняется в имени файла: AreaOfCircle.java. Класс имеет два метода: main и area. При выполнении этого класса метод main выполняется автоматически. Метод main получает в качестве ввода строку и преобразует ее в целое число, которое помещается в переменную input:

```
int input=Integer.parseInt(args[0]);
```

Существуют по крайней мере две возможные исключительные ситуации, которые могут породиться при выполнении этого кода. Первая - `ArrayIndexOutOfBoundsException` - возникает, когда делается попытка обратиться к несуществующей позиции массива. Так как код явно ссылается на первую позицию массива с именем `args` (`args[0]`), то исключительная ситуация будет породиться в том случае, если программе не были переданы никакие данные. Другая возможная ошибка - `NumberFormatException` - возникает в том случае, когда в качестве аргумента передается нечисловое значение, например, если вместо "java AreaOfCircle 123" ввести "java AreaOfCircle XYZ" (XYZ не является допустимым числом). Обе эти исключительные ситуации обрабатываются двумя операторами `catch`, которые предоставляют пользователю полезную обратную связь в отношении причины ошибки:

```
catch (ArrayIndexOutOfBoundsException oob) {
    System.out.println("You did not provide the radius of the circle.");
    System.out.println("Usage: java AreaOfCircle 10");
}
catch (NumberFormatException nfe) {
    System.out.println("Enter a valid number for the radius.");
    System.out.println("Usage: java AreaOfCircle 10");
}
```

Затем объявляется переменная `result`, которая задается равной значению, возвращаемому вызовом `area(input)`:

```
double result=area(input);
```

Чтобы обработать эту директиву, выполняется метод `area`, использующий в качестве входного значения значения переменной `input`. Метод имеет следующий вид:

```
public static double area (int rad) {
    double pi=3.1415927;
    double areacircle=pi*rad*rad;
```

```
return areacircle;
}
```

В своем определении метод `area` указывает, что он получает одну переменную целочисленного типа с именем `rad`. Затем значение этой переменной (передаваемой из переменной `input` метода `main`) обрабатывается и вычисляется переменная с именем `areacircle`, которая возвращается в `main`. В `main` это значение печатается:

```
❑ double result=area(input);
  System.out.println(result);
```

Тестирование программы очевидно и легко выполнимо:

```
❑ javac AreaOfCircle.java
  java AreaOfCircle 10
```

Будет получен следующий результат:

```
❑ 314.15927
```

Давайте расширим этот пример, чтобы включить в него обработку цикла `WHILE`. В этом примере метод `area` будет вызываться повторно, пока получаемое входное значение не превысит заданное значение:

```
❑ // AreaOfCircleWhile.java
  //
  public class AreaOfCircleWhile {
  public static void main(String[] args) {
  try {
  int input = Integer.parseInt(args[0]);
  while (input < 7) {
  double result=area(input);
  System.out.println(result);
  input++;
  }
  }
  catch (ArrayIndexOutOfBoundsException oob) {
  System.out.println("You did not provide the radius of the circle.");
  System.out.println("Usage: java AreaOfCircle 10");
  }
  catch (NumberFormatException nfe) {
  System.out.println("Enter a valid number for the radius.");
  System.out.println("Usage: java AreaOfCircle 10");
  }
  } /// main
  public static double area (int rad) {
  double pi=3.1415927;
  double areacircle=pi*rad*rad;
  return areacircle;
  } /// area
  } /// AreaOfCircleWhile
```

В этом листинге метод `area` и блоки обработки исключительных ситуаций не изменились по сравнению с предыдущим примером. Изменение содержится в методе `main`:

```
❑ int input = Integer.parseInt(args[0]);
  while (input < 7) {
    double result=area(input);
    System.out.println(result);
    input++;
  }
```

Пока входное значение меньше 7, оно будет обрабатываться. После того как для этого значения радиуса вычисляется и печатается площадь, входное значение изменяется:

```
❑ input++;
```

и цикл оценивается заново. Пример вывода показан в следующем листинге:

```
❑ javac AreaOfCircleWhile.java
  java AreaOfCircleWhile 4
```

```
50.2654832
78.5398175
113.0973372
```

Вывод показывает вычисленную площадь для входных значений радиуса, равных 4, 5 и 6.

Этот пример не является полностью завершенной программой – может понадобиться дополнительная обработка исключений (например, для нецелочисленных значений), но его структура является важным уроком. Кроме методов, показанных в этом примере, можно создать конструктор для класса `AreaOfCircleWhile`. Все классы Java имеют конструкторы для инициализации новых объектов на основе класса. Имя конструктора совпадает с именем класса. Если создается класс без конструктора, во время выполнения Java создаст конструктор по умолчанию. Подобно методу, тело конструктора может содержать объявления локальных переменных, циклы и блоки операторов. Конструктор инициализирует эти переменные для класса.

В следующих главах вы увидите, как реализовать Java в Oracle – в хранимых процедурах и через JDBC. Эти главы предполагают наличие базовых знаний Java. Дополнительную информацию о языке Java можно найти на Web-сайте корпорации Sun Microsystems и во множестве книг.