

ВОЛГОГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

А.Г. Кравец

РАЗРАБОТКА БАЗ ДАННЫХ В СУБД ORACLE.

Учебное пособие



Волгоград 2013

УДК [658.52.011.56:658.512.011.56]:001.51

Рецензенты:

Кравец А.Г. Разработка баз данных в СУБД ORACLE: Учебное пособие. /
ВолгГТУ.

Волгоград, 2013. – 150 с.

ISBN 5-230-03738-5

Предназначено для приобретения навыков работы с СУБД ORACLE, для самостоятельного изучения СУБД ORACLE, а также проведения занятий по дисциплине «Базы Данных». Учебное пособие может быть полезно для специалистов, занимающихся разработками в сопутствующих направлениях.

Ил. 4 . Табл. 6 . Библиогр.: 8 назв.

Печатается по решению научно-методического совета факультета электроники и вычислительной техники Волгоградского государственного технического университета.

ISBN 5-230-03738-5

© Кравец А.Г., 2013

© Волгоградский государственный
технический университет, 2013

Содержание.

СУБД ORACLE. НАЧАЛЬНЫЕ СВЕДЕНИЯ.....	5
Язык SQL.....	7
Базовые компоненты SQL.....	8
Типы данных SQL.....	9
Преобразование типов.....	12
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 1.....	13
Пример выполнения работы.....	13
Содержание отчета.....	14
Контрольные вопросы.....	14
СОЗДАНИЕ ПРОСТОЙ БАЗЫ ДАННЫХ В СРЕДЕ ORACLE.....	15
ОБРАБОТКА СТРОК ТАБЛИЦЫ.....	16
ИНДЕКСАЦИЯ.....	17
Типы столбцов.....	17
Порядок создания индексов.....	19
Составные индексы.....	20
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 2.....	21
Пример выполнения работы.....	21
Содержание отчета.....	22
Контрольные вопросы.....	22
СОЗДАНИЕ РАСПРЕДЕЛЕННЫХ БАЗ ДАННЫХ.....	23
ЗАВЕРШЕНИЕ ТРАНЗАКЦИИ.....	25
РАЗДЕЛЕНИЕ ТРАНЗАКЦИИ.....	26
ОТМЕНА ТРАНЗАКЦИИ.....	26
МЕХАНИЗМ БЛОКИРОВКИ.....	27
Типы блокировок.....	28
ПАРАЛЛЕЛЬНАЯ РАБОТА В МНОГОПОЛЬЗОВАТЕЛЬСКОМ РЕЖИМЕ.....	30
Согласованное чтение.....	30
ОГРАНИЧЕНИЯ ЦЕЛОСТНОСТИ.....	31
Использование ограничений целостности NOT NULL.....	31
Использование ограничений целостности UNIQUE.....	31
Использование ссылочных ограничений целостности.....	32
Связи между родительскими и порожденными таблицами.....	32
Ограничение NOT NULL по внешнему ключу.....	32
Ограничение UNIQUE по внешнему ключу.....	33
Ограничения UNIQUE и NOT NULL по внешнему ключу.....	33
Множественные ограничения FOREIGN KEY.....	33
Использование ограничений целостности CHECK.....	33
Множественные ограничения CHECK.....	34
Ограничения целостности CHECK и NOT NULL.....	34
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 3.....	34
Пример выполнения работы.....	35
Содержание отчета.....	36
Контрольные вопросы.....	36
КОМПЛЕКСНЫЕ ОБЪЕКТЫ ORACLE.....	37
ПРЕДСТАВЛЕНИЯ, ПОСЛЕДОВАТЕЛЬНОСТИ И СИНОНИМЫ.....	37
Представления.....	37
Последовательности.....	41
Синонимы.....	45
ФОРМИРОВАНИЕ ЗАПРОСОВ.....	47
ПОЛЬЗОВАТЕЛИ, СВЯЗИ И СНИМКИ.....	50
Пользователи.....	50
Связь с удаленной базой данных.....	52
Снимки.....	55
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ № 4.....	57
Содержание отчета.....	57
Пример выполнения работы.....	57

Контрольные вопросы.	61
ORACLE CLOUD COMPUTING	62
Поддержка целостности данных	62
Разграничение уровней доступа к данным	63
Управление объектами схемы	64
Управление таблицами.....	65
Привилегии, требуемые для удаления таблиц.....	66
Функции для работы со строками.	67
Функции для работы с числами.	70
Функции для работы с датами.....	72
Функции преобразования данных.	75
Практическое занятие № 5	78
Практическое занятие № 6	81
Практическое занятие № 7	85
Контрольные вопросы.	88
JAVA И ORACLE JDEVELOPER	93
Место Java в архитектуре Oracle	94
Соотношение и взаимосвязь PL/SQL и Java в Oracle.....	94
Особенности Java и среда работы программ на Java	95
Программные компоненты в среде разработки на Java	95
Установка среды разработки на Java	96
Среда окружения ОС.....	96
Создание самостоятельных программ на Java.....	97
Создание хранимых программ на Java в Oracle	98
Обращение к загруженной в Oracle процедуре Java	102
Работа со словарем-справочником	102
Oracle JDeveloper.....	104
Практическое занятие № 8.	111
Практическое занятие № 9.	111
Практическое занятие № 10.	123
ПРАКТИЧЕСКОЕ ЗАНЯТИЕ 3 – СОЗДАНИЕ ТАБЛИЦ БАЗЫ ДАННЫХ	126
Практическое занятие № 11.	128
Практическое занятие № 12.	131
ORACLE XML	136
ORACLE DATA MINING.....	142
Функции Data Mining	142
Данные и поля.....	143
Контролируемый Data Mining.....	146
Неконтролируемый Data Mining	148
Процесс Data Mining.....	149

СУБД ORACLE. Начальные сведения.

Рассмотрим некоторые возможности ORACLE, наиболее полезные для разработчиков приложений.

Ограничения целостности позволяют определить некоторые требования к данным, которые могут включаться в таблицу, и гарантировать, что эти требования будут соблюдаться независимо от того, как вводятся данные. Эти ограничения включаются как часть определения таблицы, и не требуют никакого программирования.

Процедуры общего применения можно написать на языке SQL или его процедурном расширении — PL/SQL один раз, и сохранить их в базе данных для неоднократного использования различными приложениями. Это гарантирует согласованность поведения между приложениями, и может сократить ваше время на разработку и отладку.

Взаимосвязанные процедуры можно группировать в пакеты, и хранить спецификацию пакета отдельно от тела пакета. Тело пакета можно изменять и перекомпилировать, не затрагивая спецификацию пакета. Это позволяет вам вносить в тело пакета изменения, которые остаются невидимыми конечным пользователям и не требуют перекомпиляции объектов, обращающихся к спецификации данного пакета.

Сложные организационные правила, которые не могут быть реализованы посредством декларативных ограничений целостности, можно ввести в действие через триггеры. Триггеры, которые аналогичны хранимым процедурам, автоматически исполняются, когда выдается предложение триггера, независимо от пользователя или приложения.

Метод стоимостной оптимизации использует статистики по таблицам вместе с информацией о существующих индексах при выборе плана исполнения для предложений SQL. Это позволяет даже неопытным пользователям выдавать сложные запросы, не беспокоясь об их производительности.

Разделяемый SQL позволяет множественным пользователям совместно использовать (разделять) единственную исполняемую копию процедуры или

предложения SQL, что существенно сокращает требования к памяти. Если выдаются идентичные предложения SQL, разделяемая область SQL, использовавшаяся для обработки первого экземпляра предложения, повторно используется для обработки последующих экземпляров того же самого предложения.

ORACLE поддерживает как однобайтовые, так и мультибайтовые схемы кодирования символов. Поскольку зависящие от языка данные хранятся отдельно от обрабатывающего их кода, вы можете легко добавлять новые языки и зависящие от языка средства (такие как форматы дат), не изменяя кода вашего приложения.

По умолчанию ORACLE предусматривает блокировку на уровне строк, что позволяет множественным пользователям обращаться к разным строкам одной и той же таблицы, не сталкиваясь с соперничеством за блокировки. Хотя это в высокой степени сокращает вероятность возникновения захватов, вы при проектировании приложения все-таки должны заботиться о том, чтобы избегать этих захватов.

Больше всего выигрывают от блокировки на уровне строк приложения интерактивной обработки транзакций (т.е. приложения, в которых множественные пользователи одновременно модифицируют разные строки одной и той же таблицы). Необходимо проектировать приложения с учетом этих средств. Профили могут использоваться для установления ограничений на использование системных ресурсов, как на индивидуальный запрос, так и на сессию. При проектировании приложений можно учесть возможность того, что пользователь может быть лишен возможности доступа к системе из-за ограничений на ресурсы. С помощью профилей можно предоставлять таким редко встречающимся пользователям ограниченный доступ к базе данных. В этом случае может потребоваться учет их требований при формулировке проекта. Профили обычно контролируются администраторами базы данных.

Можно применять последовательности для автоматической генерации уникальных ключей для ваших данных и координации таких ключей между

множественными строками или таблицами. Генератор последовательных номеров устраняет очередь, которая возникала бы при генерации номеров программными средствами, за счет того, что блокирует последнее использовавшееся значение, а затем наращивает его. Последовательные номера могут также считываться из кэша последовательных номеров, вместо диска, что еще более увеличивает скорость их получения.

ORACLE разработан в соответствии с промышленными стандартами. Если ваши приложения должны удовлетворять таким стандартам, вам следует обратиться к документу ORACLE Server SQL Language Reference Manual за детальным объяснением вопросов согласованности ORACLE со стандартами SQL.

ORACLE поддерживает реляционную модель данных, поэтому к числу основных объектов ORACLE относятся: таблица, представления, пользователь. Для упрощения решения задач идентификации и именования в БД ORACLE поддерживает объекты: последовательность и синоним. Для управления эффективностью доступа к данным ORACLE поддерживает объекты: индекс, табличная область и кластер. Специфичными для распределенных систем являются объекты снимок и связь БД. Для эффективного управления разграничением доступа к данным ORACLE поддерживает объект роль. Для программирования алгоритмов обработки данных, реализации механизмов поддержки целостности БД используются объекты: процедура, функция, пакет и триггер.

Язык SQL.

Все операции, выполняемые с информацией в базе данных ORACLE, осуществляются с помощью предложений SQL. После завершения проектирования приложения, нужно будет начать написание предложений SQL, реализующих этот проект.

Язык структурированных запросов (SQL) — непроцедурный язык, используемый большинством реляционных БД. Операторы языка SQL

описывают действия, которые нужно выполнить относительно объектов и наборов данных в базе данных.

Базовые компоненты SQL.

Язык SQL имеет следующие базовые компоненты:

Значения – часть информации с ассоциированным типом данного. Другие компоненты SQL специфицируют и манипулируют значениями.

Объект – именованная структура базы данных, которая хранит или организует информацию.

Литерал (Константа) – фиксированное значение данных.

Операция выполняет некоторое действие с одним или более значениями и возвращает результат.

Функция выполняет вычисления и возвращает результат. Обычно оперирует одним или более параметрами, которые специфицируются при вызове функции.

Выражение представляет или вычисляет значение. Выражения могут состоять из литералов, объектных значений и функций – отдельных или объединенных операциями.

Условие (Критерий поиска) – комбинация одного или более выражений и логических операций, которая оценивается как TRUE или FALSE.

Команда выполняет вычисления или другие операции. Часто возвращает информацию, но не значение результата.

Расширения базового SQL обеспечивают процедурные возможности, такие как переменные и операторы управления потоком обработки.

PL/SQL – это процедурное расширение ORACLE стандартного языка SQL и является составной частью во многих продуктах ORACLE. Сервер ORACLE включает поддержку языка PL/SQL, предоставляя пользователю возможность создавать и использовать на сервере процедуры и триггеры БД, выполняющие задачи конкретного приложения. С помощью PL/SQL можно улучшить производительность разрабатываемого приложения и системы в целом. Вместо интерпретируемых операторов SQL система ORACLE позволяет

использовать предварительно скомпилированные и, следовательно, быстро выполняющиеся процедуры, созданные на языке PL/SQL.

Всякая программа на PL/SQL состоит из трех блоков: блока описаний, исполнительного блока и блока обработки исключительных операций. Исполнительный блок может быть структурирован с использованием операторных скобок Begin и End. Синтаксический программа на PL/SQL оформляется следующим образом:

DECLARE

Описание переменных, констант,
типов данных, определенных пользователем

BEGIN

Операторы основной программы

EXCEPTION

Операторы – фрагменты кода программы для обработки
исключительных ситуаций в программе

END

Типы данных SQL.

Любая переменная или константа должна иметь один из допустимых в ORACLE типов. Информация о типах данных ORACLE содержится в Таблице 1:

Таблица 1.

Тип	Описание	Максимальный размер: Oracle 9i/10g/11g	Максимальный размер: PL/SQL	PL/SQL подтип/ синоним
VARCHAR2(size)	Длина строки определяется значением size	4000 байт минимум, 1 байт	32767 байт минимум, 1	STRING VARCHAR
NVARCHAR2(size)	Длина строки с национальными символами определяется значением size	4000 байт минимум, 1	32767 байт минимум, 1	STRING VARCHAR
VARCHAR	В этой версии запрещено, поддерживается для обратной совместимости. VARCHAR это синоним VARCHAR, но он использует некоторые улучшения.	-		
CHAR(size)	Fixed length character data of length size bytes. This should be	2000 bytes Default and	32767 bytes Default and minimum	CHARACTER

	Использует фиксированную длину данных, например код A100, B102...	minimum размер 1 байт.	размер 1 байт.	
NCHAR(size)	Использует фиксированную длину данных национальных соилов, например код A100, B102...	2000 байт. По умолчанию выставляется минимум в 1 байт.	32767 байт. По умолчанию выставляется минимум в 1 байт	
NUMBER(p,s)	Число, имеющее точность p и приближение s.	Точность может быть от 1 до 38. Приближение может быть от -84 до 127.	Величина 1E-130 .. 10E125 Максимальная точность 126 двоичных знаков, что эквивалентно 38 десятичным символам. Приближение может быть от -84 до 127. Для чисел с плавающей запятой не определяют p,s REAL имеет максимальную точность 63 бинарных знаков, что эквивалентно 18 десятичным знакам	Числа с фиксированной запятой: DEC DECIMAL NUMERIC с плавающей запятой: DOUBLE PRECISION FLOAT binary_float (32 bit) binary_double (64 bit) целые: INTEGER INT SMALLINT simple_integer(10g) BOOLEAN REAL
PLS_INTEGER	Знаковое (signed) целое PLS_INTEGER минимальное значение для хранения NUMBER значений.	PL/SQL только	Диапазон значений - 2147483647 .. 2147483647	
BINARY_INTEGER	знаковое (signed) целое (старая версия PLS_INTEGER)		Диапазон значений - 2147483647 .. 2147483647	NATURAL NATURALN POSITIVE POSITIVEN SIGNTYPE
LONG	Имеет большую длину, чем VARCHAR2	Максимальная длина 2 Гигабайта, в текущей версии запрещено, поддерживается только для совместимости	32760 байт Меньше чем максимальная длина LONG	
DATE	Значение даты	От 1 января 4712 да нашей эры до 31 декабря 9999 нашей.	От 1 января 4712 да нашей эры до 31 декабря 9999 нашей (для Oracle7 = 4712 нашей веры)	
TIMESTAMP (fractional_seconds_precision)	the number of digits in the fractional part of the SECOND datetime field.	Accepted values of fractional_seconds_precision are 0 to 9. (default = 6)		
TIMESTAMP (fractional_seconds_precision) WITH {LOCAL} TIMEZONE	As above with time значение временного сдвига	precision are 0 to 9. По- умолчанию 6		

INTERVAL YEAR (year_precision) TO MONTH	Время в годах и месяцах, когда year_precision это количество знаков YEAR=false.	Допустимые значения от 0 до 9. (По-умолчанию = 2)		
INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)	Время в днях, годах, месяцах, часах и минутах. <i>day_precision</i> это максимальное число в 'DAY' <i>fractional_seconds_precision</i> это максимальное число в поле SECOND.	<i>day_precision</i> может быть от 0 до 9. (по-умолчанию = 2) <i>fractional_seconds_precision</i> может быть от 0 до 9. (по-умолчанию = 6)		
RAW(size)	Raw двоичные данные заданного размера.	Максимальный размер 2000 байт	Максимальный размер 32767 байт	
LONG RAW	Raw двоичные данные заданного размера. (не поддерживается в PL/SQL)	2 два гигабайта, в настоящих версиях не поддерживается	32760 байт Не меньше чем максимальная длина столбца LONG RAW	
ROWID	Шестнадцатеричное число, указывает на уникальный адрес этого столбца	10 байт	Шестнадцатеричное число, указывает на уникальный адрес этого столбца	
UROWID	Шестнадцатеричное число, указывает на уникальный адрес этого столбца в индексо-ориентированных таблицах	Максимальный размер, он установлен по-умолчанию, 4000 байт	Шестнадцатеричное число, указывает на уникальный адрес этого столбца в индексо-ориентированных таблицах	См. CHARTOROWID и пакет: DBMS_ROWID
MLSLABEL	Двоичный формат метки операционной системы.			
CLOB	Содержит большие объекты	4 гигабайта В Oracle 11g максимальный размер = (4 гигабайта - 1) * (размер блока данных в БД)	4 Гигабайта	
NCLOB	Большой объект с национальными символами	4 гигабайта В Oracle 11g максимальный размер = (4 гигабайта - 1) * (размер блока данных в БД)	4 Гигабайта	
BLOB	Большой двоичный объект	4 гигабайта В Oracle 11g максимальный размер = (4 гигабайта - 1) * (размер блока данных в БД)	4 Гигабайта	
BFILE	Указатель на двоичный файл на диске	4 гигабайта В Oracle 11g = (4 гигабайта - 1) * (размер блока данных в БД)	Размер BFILE зависит от ОС, но не превышает 4 гигабайт (2**32 - 1 байт).	
XMLType	XML данные	4 Гигабайта	Populate Содержит XML из	

			CLOB или VARCHAR2. или запрос к другим XMLстолбцам.	
--	--	--	---	--

Преобразование типов.

В некоторых случаях ORACLE позволяет применять данные одного типа там, где ожидаются данные другого типа. Это разрешается, когда ORACLE может автоматически преобразовать данные в ожидаемый тип данных, неявно используя одну из следующих функций:

- ASCIISTR()
- BFILENAME()
- BIN_TO_NUM()
- CAST()
- CHARTOROWID()
- COMPOSE()
- CONVERT()
- DECOMPOSE()
- HEXTORAW()
- NUMTODSINTERVAL()
- NUMTOYMINTERVAL()
- RAWTOHEX()
- RAWTONHEX()
- REFTOHEX()
- ROWIDTOCHAR()
- ROWIDTONCHAR()
- SCN_TO_TIMESTAMP()
- TIMESTAMP_TO_SCN()
- TO_BINARY_DOUBLE()
- TO_BINARY_FLOAT()
- TO_CHAR()
- TO_CLOB()

- TO_DATE()
- TO_DSINTERVAL()
- TO_LOB()
- TO_MULTI_BYTE()
- TO_NCHAR()
- TO_NCLOB()
- TO_NUMBER()
- TO_SINGLE_BYTE()
- TO_TIMESTAMP()
- TO_TIMESTAMP_TZ()
- TO_YMINTERVAL()
- TRANSLATE USING()
- UNISTR()

Операторы большинства языков программирования, в том числе и языка PL/SQL, выполняются последовательно. Такая схема называется потоком команд. В PL/SQL также предусмотрены операторы, с помощью которых можно управлять потоком команд, выполнять условные переходы, циклически выполнять группу операторов и осуществлять выход из группы

Практическое занятие № 1.

Цель работы: Ознакомиться с основными возможностями и средствами распределенной СУБД ORACLE.

Пример выполнения работы.

Рассмотрим пример простейшей программы, в которой определяются переменные и выполняются действия по вычислению натуральных логарифмов 2 и 3. Команды установки переменных окружения `serveroutput` и `echo` определяют режим вывода на терминал пользователя. Системная процедура `DBMS_OUTPUT.PUT_LINE` обеспечивает вывод данных на терминал пользователя, а функция `Ln` вычисляет натуральный логарифм аргумента. Символ `"/"` указывает на завершение текста процедуры и является командой к интерпретации и выполнению процедуры. Для избежания сбоя кодировки,

перед запуском SQLPlus необходимо выполнить команду set
NLS_LANG=russian_cis.ru8pc866

```
SQL> set serveroutput on;
```

```
SQL> set echo on;
```

```
SQL> DECLARE
```

```
2 Header1 CONSTANT VARCHAR2(20) := 'Логарифм двух равен ';
```

```
3 Header2 CONSTANT VARCHAR2(20) := 'Логарифм трех равен ';
```

```
4 Arg NUMBER := 2; -- здесь задается начальное значение аргумента
```

```
5 -- Исполнительный блок
```

```
6 BEGIN
```

```
7 DBMS_OUTPUT.PUT_LINE(Header1||Ln(Arg));
```

```
8 Arg := Arg+1;
```

```
9 DBMS_OUTPUT.PUT_LINE(Header2||Ln(Arg));
```

```
10 END;
```

```
/
```

Логарифм двух равен

.6931471805599453094172321214581765680814

Логарифм трех равен

1.09861228866810969139524523692252570466

PL/SQL procedure successfully completed.

Содержание отчета.

1. Титульный лист.
2. Распечатка листинга программы на языке SQL, выполняющей 3 – 5 арифметических функций.

Контрольные вопросы.

1. Перечислите основные возможности ORACLE.
2. Какие объекты ORACLE вы знаете?
3. Перечислите базовые компоненты языка SQL.
4. Какие виды литералов вы знаете?
5. Назовите категории группировки команд SQL. Приведите примеры.

6. Использование команды EXEC SQL.
7. Кратко охарактеризуйте язык PL/SQL.
8. Из каких блоков состоит программа на PL/SQL?
9. Назовите особенность процедур и функций ORACLE в отличие от других языков программирования?
10. Чем функции и процедуры отличаются друг от друга?
11. Почему типовые операции выгодно оформлять в виде процедур и функций?
12. Сравните типы данных ORACLE CHAR и VARCHAR2.
13. Как задаются числовые данные в ORACLE?
14. Как изменить установленный по умолчанию формат даты? Что такое юлианские даты?
15. Перечислите ограничения на данные типа LONG.
16. Как осуществляется преобразование данных?

Создание простой базы данных в среде ORACLE.

Таблица (Table) является базовой структурой реляционной модели. Вся информация в БД хранится в таблицах. Таблицы состоят из множества поименованных столбцов или атрибутов. Множество значений столбца определено с помощью ограничений целостности, т.е. поддерживается ограниченная концепция домена. Полное имя таблицы в БД состоит из имени схемы, в данной реализации совпадающего с именем пользователя, создавшего таблицу, и собственно имени таблицы. Таблица может быть пустой или состоять из одной или более строк значений атрибутов. Строки значений атрибутов таблицы называются кортежами.

Создавайте таблицы с помощью команды SQL CREATE TABLE. Например, выдавая следующее предложение, пользователь SCOTT создает некластеризованную таблицу с именем EMP в своей схеме и сохраняет ее в табличном пространстве USERS. Заметьте, что на нескольких столбцах этой таблицы определены ограничения целостности.

```
CREATE TABLE emp (  
empno NUMBER(5) PRIMARY KEY,
```

```
ename VARCHAR2(15) NOT NULL,  
job VARCHAR2(10),  
mgr NUMBER(5),  
hiredate DATE DEFAULT (sysdate),  
sal NUMBER(7,2),  
comm NUMBER(7,2),  
deptno NUMBER(3) NOT NULL  
)  
PCTFREE 10  
PCTUSED 40  
TABLESPACE users  
STORAGE ( INITIAL 50K  
NEXT 50K  
MAXEXTENTS 10  
PCTINCREASE 25 );.
```

Для удаления таблицы используется команда DROP TABLE.

Для изменения таблицы используйте команду ALTER TABLE.

Для вставки строк атрибутов в таблицу используйте команду INSERT INTO.

Обработка строк таблицы.

Операция DELETE применяется для удаления строк из таблицы. Все удаляемые строки и соответствующие индексы освобождают занимаемую ими память.

Модификация строк из таблицы осуществляется операцией UPDATE.

Набор встроенных в язык SQL функций позволяет выполнить многие типовые операции преобразования данных вызовом соответствующей функции.

Функция SUBSTR возвращает подстроку, начиная с заданной позиции, с указанной длиной.

Функция INSTR возвращает позицию вхождения строки поиска в заданную строку.

Функция LENGTH возвращает длину заданной строки.

Функция CHR возвращает символ, который имеет соответствующее значение номера.

Функция ASCII возвращает числовое значение (номер) заданного символа.

Индексация.

Индексы – объекты БД, которые обеспечивают быстрый доступ к отдельным строкам в таблице. Индекс создается с целью повышения производительности операции запросов и сортировки данных таблицы. Индексы также используются для поддержания в таблицах некоторых типов ограничений; эти индексы часто создаются автоматически при определении ограничения. Индекс – независимый объект, логически отделенный от индексированной таблицы – создание или удаление индекса никак не воздействует на определение или данные индексированной таблицы.

Индекс хранит высоко оптимизированные версии всех значений одного или больше столбцов таблицы. Когда значение запрашивается из индексированного столбца, процессор БД использует индекс для быстрого нахождения требуемого значения. Индексы обеспечивают наибольшие выгоды для относительно статичных таблиц, по которым, в то же время, часто выполняются запросы.

Типы столбцов.

Индексы обычно используются со следующими типами столбцов:

1. Столбцы Primary Key. Столбцы первичного ключа часто используются для поиска и сортировки, поэтому в данном случае индекс может улучшать производительность многих типов операций. Некоторые типы БД автоматически используют индекс для поддержания ограничения Primary Key.
2. Столбцы Foreign Key. Индексы могут повышать быстродействие соединения двух таблиц в отношении один-ко-многим.

3. Часто сортируемые столбцы. Индексы хранят информацию в предварительно отсортированном формате, который увеличивает быстродействие сортировки данных таблицы по индексированному столбцу.

Индексы используются в ORACLE для того, чтобы обеспечивать быстрый доступ к строкам таблицы. Индексы ускоряют доступ к данным для операций, затрагивающих небольшую часть строк таблицы. ORACLE не ограничивает количество индексов, которые вы можете создавать по таблице. Однако вы должны рассматривать возможные выгоды в производительности и потребности ваших приложений базы данных, когда определяете, какие столбцы индексировать.

Индекс может быть создан для таблицы, чтобы улучшить производительность запросов, выдаваемых по соответствующей таблице. Индекс может также быть создан для кластера. Вы можете создать СОСТАВНОЙ индекс по нескольким (до 16) столбцам.

ORACLE вводит в действие ограничение целостности UNIQUE или PRIMARY KEY, автоматически создавая уникальный индекс по уникальному или первичному ключу. В общем случае, для обеспечения уникальности предпочтительнее создавать ограничения, чем использовать устаревший синтаксис CREATE UNIQUE INDEX.

Индексы создаются с помощью команды SQL CREATE INDEX. Например, следующее предложение создает индекс с именем EMP_ENAME для столбца ENAME таблицы EMP:

```
CREATE INDEX emp_ename ON emp(ename)
TABLESPACE users
STORAGE (INITIAL 20K
NEXT 20K
PCTINCREASE 75)
PCTFREE 0;
```

Для удаления индекса используйте команду SQL DROP INDEX. Например, следующее предложение удаляет индекс EMP_ENAME:

`DROP INDEX emp_ename;`

Когда вы удаляете таблицу, все ассоциированные индексы удаляются автоматически.

Порядок создания индексов.

Необходимо создавать индекс по таблице после того, как данные были вставлены или загружены в таблицу. Гораздо эффективнее вставить строки данных в таблицу, не имеющую индексов, а затем создать индексы для последующего доступа к этим данным. Если вы создадите индексы перед тем, как загружать данные в таблицу, то при вставке каждой строки данных требуется обновление всех индексов. Исключение из этого правила касается кластера, для которого вы ДОЛЖНЫ создать индекс перед тем, как вставлять в кластер любые данные.

Когда индекс создается по таблице, уже имеющей данные, ORACLE должен использовать область сортировки, чтобы создать индекс. ORACLE использует область сортировки в памяти, распределяемой для создателя индекса (размер этой области на пользователя определяется параметром инициализации `SORT_AREA_SIZE`), но вынужден также обмениваться информацией сортировки с временными сегментами, распределяемыми от имени процесса создания индекса.

Если индекс исключительно велик, может оказаться полезным выполнить следующие шаги:

1. Создать новое табличное пространство для временных сегментов, используя команду `CREATE TABLE`.
2. С помощью опции `TEMPORARY TABLESPACE` команды `ALTER USER` изменить табличное пространство для временных сегментов для создателя индекса, указав вновь созданное табличное пространство.
3. Создать индекс, используя команду `CREATE INDEX`.
4. С помощью команды `DROP TABLESPACE` удалить табличное пространство для временных сегментов. Затем командой `ALTER USER` снова переопределить

табличное пространство для временных сегментов для создателя индекса, указав старое табличное пространство.

Порядок, в котором столбцы индекса перечисляются в команде CREATE INDEX, не обязан соответствовать порядку, в котором эти столбцы определены в таблице. Однако порядок столбцов в команде CREATE INDEX существенен, так как он может повлиять на производительность запросов. В общем случае, вы должны первым в индексе указывать тот столбец, который будет использоваться наиболее часто.

Составные индексы.

Составной индекс - это индекс, состоящий из более чем одного столбца. Составные индексы могут предоставлять дополнительные преимущества по сравнению с одностолбцовыми индексами:

1. Лучшая селективность. Иногда можно скомбинировать два или более столбцов, каждый из которых обладает низкой селективностью, в составной индекс, имеющий хорошую селективность.
2. Дополнительный источник данных. Если все столбцы, выбираемые запросом, входят в составной индекс, то ORACLE может вернуть эти значения прямо из индекса, не обращаясь к таблице.

Предложение SQL может использовать путь доступа, включающий составной индекс, если это предложение содержит конструкторы, которые используют ведущую порцию индекса. Ведущая порция индекса - это один или несколько столбцов, которые были специфицированы первыми и подряд в списке столбцов предложения CREATE INDEX, с помощью которого был создан индекс. Рассмотрим следующее предложение CREATE INDEX:

```
CREATE INDEX comp_ind  
ON tab1(x, y, z)
```

Следующие комбинации столбцов являются ведущими порциями этого индекса: X, XY и XYZ. Другие комбинации столбцов, например, XZ, YZ или Z, не являются ведущими порциями этого индекса.

Практическое занятие № 2.

Цель работы: Получение навыков создания базы данных и работы с ней в диалоговом режиме.

Пример выполнения работы.

Рассмотрим пример создания таблицы Tab1 с тремя атрибутами At1, At2, At3 в схеме пользователя U1. Ограничение pk_Tab1_At1 указывает, что атрибут At1 является первичным ключом: ограничение nn_Tab1_At2 указывает, что атрибут At2 не допускает ввода неопределенных значений; значение атрибута At3 по умолчанию есть текущая дата.

```
SQL> CREATE TABLE U1.Tab1  
2 ( At1 NUMBER CONSTRAINT pk_Tab1_At1 PRIMARY KEY,  
3 At2 NUMBER CONSTRAINT nn_Tab1_At2 NOT NULL,  
4 At3 DATE DEFAULT SYSDATE);
```

Table created.

Создадим индекс с именем Ind_name для столбца At1 таблицы Tab1:

```
SQL> CREATE INDEX Ind_name ON Tab1(At1)  
TABLESPACE users  
STORAGE (INITIAL 20K  
NEXT 20K  
PCTINCREASE 75)  
PCTFREE 0;
```

Index created.

Следующий пример демонстрирует создание таблицы Tab2 с двумя атрибутами At1 и At2, размещенной в табличном пространстве app_data (которое должно быть создано заранее) и проиндексированной по значению первичного ключа At1 с индексом, размещенном в табличном пространстве index_data. Под таблицу резервируется начальный экстенст в 100 килобайт и определяется экстенст приращения в 50 килобайт.

```
SQL> CREATE TABLE Tab2  
2 ( At1 NUMBER CONSTRAINT pk_Tab1_At1 PRIMARY KEY
```

3 USING INDEX TABLESPACE index_data,

4 At2 NUMBER)

5 TABLESPACE app_data

Table created.

Содержание отчета.

1. Титульный лист.
2. Оператор CREATE TABLE с описанием структуры проиндексированной таблицы.
3. Распечатка структуры таблицы.
4. Распечатка содержимого таблицы (не менее 30 записей).
5. Распечатка оператора CREATE INDEX.

Контрольные вопросы.

1. Как создать таблицу в ORACLE?
2. Какая команда SQL используется для изменения таблиц? Ее формат.
3. Как изменить таблицу или полностью удалить?
4. Операция вставки строк.
5. Операция удаления строк.
6. Операция модификации строк.
7. Какие однострочные символьные функции вы знаете?
8. Перечислите функции, устанавливающие соответствие числовых кодов и символов.
9. Перечислите функции, связанные с поиском вхождений подстрок.
10. В чем отличие функций CHR() и ASCII()?
11. Что такое индекс?
12. Когда лучше создавать индексы и почему?
13. Создание и удаление индекса с помощью команд SQL. Формат этих команд.
14. Перечислите действия, необходимые для создания очень большого индекса.
15. На что влияет порядок столбцов в команде CREATE INDEX?
16. Что такое составной индекс?
17. Объясните, что такое ведущая позиция индекса. Приведите пример.

Создание распределенных баз данных.

Транзакция — это последовательность операторов обработки данных, которые рассматриваются как логически неделимая единица работы с базой данных. В дальнейшем будем считать, что в роли операторов обработки данных выступают SQL-предложения. Система гарантирует невозможность фиксации некоторой части действий из транзакции в базе данных. Например, если модификация строк некоторой таблицы оформлена в виде транзакции, то система гарантирует, что пользователь, выполняющий выборку из таблицы, будет получать либо только "старые" данные, либо только "новые", но не часть "старых" и часть "новых" данных.

До тех пор пока транзакция не зафиксирована, ее можно "откатить", то есть отменить все сделанные операторами из транзакции изменения в базе данных. Обратите внимание, что смысл фразы "SQL-операторы транзакции успешно завершены" отличается от смысла фразы "транзакция зафиксирована" (committed). Успешное выполнение SQL-операторов означает, что операторы проанализированы, интерпретированы как правильные, а затем безошибочно исполнены. Зафиксировать транзакцию означает сделать изменения, выполненные данной транзакцией в базе данных, постоянными. Пока транзакция не зафиксирована, результат ни одного из ее действий не виден другим пользователям.

Система Oracle гарантирует согласованность данных, основанную не на единственном SQL-операторе, а на транзакции (хотя транзакция может состоять и из одного оператора). Именно для транзакции данные либо сохраняются в базе данных, как ее видят все пользователи, либо откатываются назад. Если в процессе выполнения транзакции произошел сбой операционной системы или прикладной программы, данные в базе автоматически восстанавливаются в состояние, предшествующее началу выполнения этой транзакции. Например, если регистрируется переход работника из одного подразделения в другое, изменения в базе данных организации либо будут

проведены вместе, либо отменены, если в процессе их проведения будет зафиксирована ошибка.

Транзакция начинается при появлении первого выполнимого SQL-оператора, то есть оператора описания или манипулирования данными. Транзакция завершается при появлении одного из следующих событий:

- выдана команда языка SQL COMMIT или ROLLBACK;
- выдана одна из таких команд языка описания данных (DDL), как CREATE, DROP или ALTER (при этом фиксируется предыдущая транзакция);
- завершился оператор DDL (транзакция, содержащая в себе оператор языка описания данных, фиксируется автоматически);
- пользователь завершил сеанс с системой ORACLE (последняя транзакция фиксируется автоматически);
- процесс пользователя аварийно завершен (транзакция откатывается автоматически).

Как только транзакция завершена, следующий выполнимый SQL-оператор начинает новую транзакцию. Если в конце транзакции не появились операторы COMMIT или ROLLBACK, то нормальное завершение программы приведет к фиксации транзакции, а аварийное завершение (вызванное, например, разрывом связи в локальной сети) вызовет откат транзакции. Если произошел сбой операционной системы (например, из-за нехватки ресурсов), то откат транзакции будет автоматически выполнен при запуске экземпляра ORACLE после восстановления операционной системы.

Для фиксации или отката транзакции используются предложения COMMIT WORK, SAVEPOINT, ROLLBACK WORK.

Предложение COMMIT фиксирует транзакцию. Предложение ROLLBACK выполняет откат транзакции, то есть отменяет все изменения, выполненные данной транзакцией в базе данных. Предложение SAVEPOINT выполняет промежуточную "текущую копию" состояния базы данных для того, чтобы впоследствии, при необходимости, можно было вернуться к состоянию базы данных в точке сохранения.

Завершение транзакции.

Предложение COMMIT WORK имеет следующий синтаксис:

COMMIT [WORK]

Ключевое слово WORK может быть опущено. Предложение COMMIT WORK обеспечивает выполнение следующих действий:

- фиксируются, то есть делаются видимыми всем пользователям системы все изменения в базе данных, сделанные в текущей транзакции;
- уничтожаются все точки сохранения для данной транзакции;
- завершается транзакция;
- освобождаются объекты, заблокированные в процессе выполнения транзакции.

Хорошим стилем разработки кода прикладных программ является явное завершение транзакций использованием предложения COMMIT WORK. Рекомендуется выдавать предложение COMMIT WORK и перед завершением сеанса. Использование такой практики обеспечит автоматический откат неуспешной транзакции с прогнозируемым результатом при аварийном завершении прикладного процесса, вызванного программным или аппаратным сбоем.

В процессе выполнения транзакции, в которой содержатся операторы изменения данных, происходит выполнение следующих действий:

- сервер создает в собственной памяти специальные записи в сегментах отката;
- сервер выполняет формирование соответствующих записей в журнальный файл;
- производятся изменения в буферах базы данных. Когда транзакция явно или неявно завершается, выполняется следующий набор операций:
- транзакция помечается как зафиксированная;

- если записи из журнального файла еще не записаны в файлы базы данных, то выполняется запись в долговременную память (память на магнитных дисках);
- заблокированные строки и таблицы освобождаются.

После выполнения предложения COMMIT (возможно, с некоторой временной задержкой), информация из кэша, размещенного в оперативной памяти сервера, записывается в долговременную память. После фиксации транзакции освобождается пространство из сегментов отката.

Разделение транзакции.

Точки сохранения (savepoint) используются для разделения длительной транзакции на более мелкие элементы. В большой транзакции, выполняющей значительные изменения в базе данных, целесообразно сохранять частичные изменения в определенных точках, чтобы при возникновении сбоя выполнять откат не к точке старта транзакции, а к последней успешно выполненной точке сохранения.

Предложение SAVEPOINT имеет следующий синтаксис:

SAVEPOINT имя_контрольной_точки

Параметр имя_контрольной_точки задает идентификатор, который соответствует правилам именования объектов базы данных. Если значение параметра имя_контрольной_точки совпадает с ранее использованным значением, информация о предыдущей точке сохранения теряется. Число различных точек сохранения на транзакцию зависит от версии сервера ORACLE и может быть изменено заданием соответствующего параметра файла INIT<sid>.ORA.

Отмена транзакции.

Предложение ROLLBACK WORK служит для отмены транзакции и имеет следующий синтаксис:

ROLLBACK[WORK] [TO[SAVEPOINT] имя_контрольной_точки]

Использование ключевого слова WORK необязательно. Использование предложения ROLLBACK без фразы TO SAVEPOINT приводит к выполнению следующих действий:

- завершению выполнения транзакции;
- отмене всех изменений в текущей транзакции;
- очистке точек сохранения для текущей транзакции;
- отмене всех блокировок транзакции.

Использование предложения ROLLBACK с фразой TO SAVEPOINT приводит к выполнению следующих действий:

- откату только части транзакции, то есть отмене части изменений в базе данных, произведенных операторами транзакции;
- фиксации данных, измененных до точки сохранения;
- освобождению всех блокировок строк, выполненных после создания последней точки сохранения, но не освобождению блокировок уровня транзакции.

Механизм блокировки.

Блокировка — это механизм, предназначенный для предотвращения некорректного изменения данных при параллельной и асинхронной работе пользователей распределенной системы. Блокировка используется:

- для обеспечения гарантированной неизменности данных другими пользователями в рамках транзакции;
- для обеспечения естественного временного порядка изменений, проводимых в базе данных.

В большинстве стандартных ситуаций сервер ORACLE обеспечивает необходимые блокировки автоматически и не требует дополнительных действий пользовательского процесса. Однако иногда может потребоваться явное управление блокировками для настройки производительности или выполнения специальных требований к прикладной системе.

Транзакция явно получает указанные блокировки таблицы, когда она выдает предложение LOCK TABLE. Предложение LOCK TABLE вручную

перекрывает умалчиваемое блокирование. Когда предложение LOCK TABLE выдается для обзора, блокируются базовые таблицы этого обзора. Следующее предложение запрашивает монопольные блокировки для таблиц EMP и DEPT от имени содержащей транзакции:

```
LOCK TABLE emp, dept  
IN EXCLUSIVE MODE NOWAIT;
```

Можно указать несколько таблиц или обзоров, которые требуется заблокировать в одном и том же режиме; однако на одно предложение LOCK TABLE может быть специфицирован лишь один режим блокировки. Можно также указать, хотите ли вы ждать получения блокировки. Если вы специфицируете опцию NOWAIT, то вы получите блокировку лишь в том случае, если она доступна немедленно. В противном случае возвращается ошибка, указывающая, что блокировка в данный момент недоступна. В этом случае вы можете повторить попытку заблокировать ресурс позже. Если опция NOWAIT опущена, то транзакция не будет продолжена, пока не получит запрошенную блокировку. Если ожидание блокировки таблицы становится слишком долгим, вы можете захотеть снять операцию блокирования и повторить ее позже; вы можете предпочесть закодировать эту логику в ваших приложениях.

Типы блокировок.

Типы блокировок ROW SHARE и ROW EXCLUSIVE:

```
LOCK TABLE таблица IN ROW SHARE MODE;
```

```
LOCK TABLE таблица IN ROW EXCLUSIVE MODE;
```

Блокировки ROW SHARE (разделяемая для строк) и ROW EXCLUSIVE (монопольная для строк) обеспечивают наибольшую степень одновременного доступа.

Тип блокировки SHARE:

```
LOCK TABLE таблица IN SHARE MODE;
```

Блокировка SHARE (разделяемая) является более ограничительной блокировкой таблицы.

Блокировка типа SHARE ROW EXCLUSIVE

LOCK TABLE таблица IN SHARE ROW EXCLUSIVE MODE;

Блокировка типа EXCLUSIVE

LOCK TABLE таблица IN EXCLUSIVE MODE;

Вы можете перекрывать умалчиваемое блокирование, выдав предложение SELECT с фразой FOR UPDATE. Предложение SELECT ... FOR UPDATE используется для получения монопольных блокировок строк для выбираемых строк в предвидении действительного обновления выбранных строк.

Вы можете использовать предложение SELECT ... FOR UPDATE для блокирования строк без фактического изменения этих строк. Предложения SELECT ... FOR UPDATE часто используются интерактивными программами, позволяющими пользователю модифицировать поля в одной или нескольких выбранных строках (что может потребовать некоторого времени); по этим строкам запрашиваются блокировки строк, чтобы только один пользователь в любой данный момент времени мог обновлять эти строки.

Если предложение SELECT ... FOR UPDATE используется для определения курсора, то строки в возвращаемом множестве блокируются перед первым извлечением, когда курсор открывается; строки не блокируются по одной по мере их извлечения из курсора. Все блокировки освобождаются не при закрытии курсора, а при подтверждении или откате транзакции, открывшей этот курсор. Каждая строка в возвращаемом множестве предложения SELECT ... FOR UPDATE блокирована индивидуально; если какая-либо строка захвачена конфликтующей блокировкой другой транзакции, то предложение SELECT ... FOR UPDATE будет ожидать освобождения этой блокировки. Поэтому, если предложение SELECT ... FOR

UPDATE блокирует много строк в таблице, для которой высока активность одновременных обновлений, вы, скорее всего, улучшили бы производительность, запросив вместо этого монопольную блокировку таблицы.

Все типы блокировок отменяются при выполнении фиксации транзакции выполнением предложения COMMIT. При откате транзакции к точке

сохранения блокировка таблиц снимается, а блокировка строк, как правило, — нет.

Параллельная работа в многопользовательском режиме.

Отличительной особенностью СУБД промышленного уровня является поддержка параллельной работы многих пользователей. Пользователи формируют запросы на доступ и изменение информации в базе данных асинхронно, то есть в произвольные моменты времени. Одно из базовых требований к СУБД — поддержка целостности данных, то есть такого состояния, когда в произвольный момент времени данные адекватно отображают состояние моделируемых объектов реального мира. Суть проблемы состоит в том, что в процессе управления параллельной работой СУБД данные могут быть изменены или модифицированы не в надлежащей последовательности, что может привести к потере их целостности.

Очевидным решением проблемы согласованного изменения базы данных является формирование очереди пользователей к каждому ресурсу системы (таблице, представлению, индексу и т. п.). Недостатком этого решения является резкое снижение производительности системы. Если количество пользователей исчисляется десятками или сотнями, то блокировка дефицитного ресурса приведет к очень большому времени ожидания.

Согласованное чтение.

Сервер ORACLE использует иной метод решения сформулированной проблемы, известный как согласованное чтение. Механизм согласованного чтения означает автоматическое обеспечение такой ситуации, когда данные, используемые оператором выборки, не меняются в течение работы этого оператора, и процессы выборки из базы не ожидают результатов записи строк в те же таблицы базы данных.

Метод обеспечения согласованного чтения состоит в поддержке для каждого запроса "мгновенной копии" базы данных, с которой он работает. Модель согласованного чтения ORACLE иногда называют также многоверсионной моделью, так как в системе может одновременно

существовать несколько различающихся версий одной таблицы. Различные версии таблиц поддерживаются с помощью специальных объектов ORACLE, называемых сегментами отката.

Когда возникает необходимость в получении конкретных данных для согласованного чтения, сервер ORACLE использует информацию из сегментов отката. Поэтому для чтения данных нет проблемы, связанной с доступом к данным, заблокированным для модификации.

Ограничения целостности.

Использование ограничений целостности NOT NULL

По умолчанию, все столбцы в таблице допускают пустые значения (т.е. отсутствие значения). Определяйте ограничение NOT NULL только для тех столбцов таблицы, которые действительно требуют, чтобы значение было всегда. Ограничения целостности NOT NULL часто комбинируются с другими типами ограничений целостности, чтобы еще более ограничить значения, которые могут существовать в специфических столбцах таблицы. Используйте комбинацию ограничений целостности NOT NULL и UNIQUE, чтобы осуществлять ввод значений в уникальный ключ; эта комбинация ограничений целостности гарантирует, что данные новых строк никогда не совпадут с данными существующих строк.

Использование ограничений целостности UNIQUE

Тщательно выбирайте уникальные ключи. Во многих ситуациях уникальные ключи некорректно составляются из столбцов, которые должны входить в первичный ключ таблицы. Решая, использовать ли ограничение UNIQUE, руководствуйтесь простым правилом: ограничение целостности UNIQUE требуется только для того, чтобы предотвращать повторение значений уникального ключа в строках таблицы. Природа данных уникального ключа такова, что эти данные не должны дублироваться в таблице. Не путайте понятие уникального ключа с понятием первичного ключа. Первичные ключи используются для того, чтобы уникально идентифицировать каждую строку

таблицы. Поэтому уникальные ключи не должны иметь цели уникальной идентификации строк в таблице.

Использование ссылочных ограничений целостности

Всегда, когда две таблицы связаны друг с другом через общий столбец (или группу столбцов), определяйте ограничение PRIMARY KEY или UNIQUE по столбцу в родительской таблице, и ограничение FOREIGN KEY по столбцу в порожденной таблице, чтобы поддержать эту связь между таблицами. В зависимости от характера этой связи, вы можете захотеть определить дополнительные ограничения целостности, включающие внешний ключ. Внешние ключи могут состоять из нескольких столбцов. Однако составной внешний ключ должен ссылаться на составной первичный или уникальный ключ, состоящий из того же количества столбцов тех же типов данных. Так как составные первичные и уникальные ключи не могут состоять из более чем 16 столбцов, составной внешний ключ также ограничивается до 16 столбцов.

Связи между родительскими и порожденными таблицами

Некоторые связи между родительской и порожденной таблицами могут быть выражены через другие типы ограничений целостности, определяемые по внешнему ключу в порожденной таблице. Если по внешнему ключу не определены никакие дополнительные ограничения, то любое количество строк в порожденной таблице могут ссылаться на одно и то же значение родительского ключа. Эта модель позволяет иметь пустые значения во внешнем ключе. Эта модель устанавливает такое отношение "один ко многим" между родительским и внешним ключами, которое позволяет иметь неопределенные (пустые) значения во внешнем ключе.

Ограничение NOT NULL по внешнему ключу.

Когда пустые значения во внешнем ключе не допускаются, каждая строка в порожденной таблице должна явно ссылаться на некоторое значение родительского ключа. Однако по-прежнему любое количество строк в порожденной таблице могут ссылаться на одно и то же значение родительского ключа. Эта модель устанавливает связь "один ко многим" между родительским

и внешним ключами. Однако каждая строка в порожденной таблице должна ссылаться на значение родительского ключа; отсутствие значения (пустота) внешнего ключа не допускается.

Ограничение UNIQUE по внешнему ключу.

Когда по внешнему ключу определено ограничение UNIQUE, лишь одна строка в порожденной таблице может ссылаться на любое данное значение родительского ключа. Эта модель позволяет иметь пустые значения во внешнем ключе. Эта модель устанавливает связь "один к одному" между родительским и внешним ключами, позволяющую иметь неопределенные (пустые) значения во внешнем ключе.

Ограничения UNIQUE и NOT NULL по внешнему ключу.

Когда по внешнему ключу определены оба ограничения UNIQUE и NOT NULL, лишь одна строка в порожденной таблице может ссылаться на любое данное значение родительского ключа, и каждая строка в порожденной таблице обязана ссылаться на некоторое значение родительского ключа. Эта модель устанавливает связь "один к одному" между родительским и внешним ключами, не позволяющую иметь неопределенные (пустые) значения во внешнем ключе.

Множественные ограничения FOREIGN KEY.

ORACLE позволяет, чтобы на данный (родительский) столбец имелись ссылки через множественные ограничения FOREIGN KEY; в действительности не существует ограничения на количество зависимых ключей. Такая ситуация может иметь место, в частности, если один и тот же столбец является частью двух различных составных внешних ключей.

Использование ограничений целостности CHECK.

Используйте ограничения CHECK, когда вам необходимо задействовать правила целостности, которые должны вычисляться на базе логических выражений. Никогда не применяйте ограничений CHECK там, где необходимая проверка может быть осуществлена через другие типы ограничений целостности. Ограничение целостности CHECK требует, чтобы определенное

условие было истинным или неизвестным (пустым) для каждой строки таблицы. Если предложение SQL приводит к ложности этого условия, то это предложение откатывается.

Множественные ограничения CHECK.

Для любого столбца можно иметь несколько ограничений CHECK, в определениях которых участвует этот столбец. Не существует лимита на число ограничений CHECK, которые можно определить с участием столбца.

Ограничения целостности CHECK и NOT NULL.

Согласно стандарту ANSI/ISO, ограничение целостности NOT NULL является частным случаем ограничения CHECK, в котором условие имеет следующий вид:

CHECK (имя_столбца IS NOT NULL)

Поэтому ограничения NOT NULL для одиночных столбцов можно, на практике, записывать двумя способами: через ограничение NOT NULL или через ограничение CHECK. Для простоты всегда выбирайте ограничение NOT NULL вместо ограничения CHECK с условием "IS NOT NULL". В случае, когда составной ключ должен допускать только пустоту или непустоту одновременно всех составляющих значений, вы можете использовать только ограничение CHECK. Например, следующее выражение позволяет составному ключу по столбцам C1 и C2 состоять либо из обоих пустых, либо из обоих непустых значений:

CHECK ((c1 IS NULL AND c2 IS NULL)

OR (c1 IS NOT NULL AND c2 IS NOT NULL))

Определяйте ограничение целостности с помощью фразы CONSTRAINT команды SQL CREATE TABLE или ALTER TABLE.

Практическое занятие № 3.

Цель работы: ознакомиться с ключевыми понятиями распределенной СУБД ORACLE – транзакциями, блокировками и целостностью данных.

Пример выполнения работы.

Предположим, что две таблицы, EMP и BUDGET, требуют согласованности множества данных в третьей таблице, DEPT. Иными словами, для данного номера отдела вы хотите обновить информацию в обеих таблицах EMP и BUDGET, и хотите гарантировать, чтобы информация об отделе не обновлялась между этими двумя транзакциями. Хотя этот сценарий довольно нетипичен, его можно осуществить, заблокировав таблицу DEPT в режиме SHARE, как показано в следующем примере. Поскольку таблица DEPT не слишком изменчива, маловероятно, чтобы кому-либо из пользователей потребовалось обновить ее, пока она заблокирована для обновлений таблиц EMP и BUDGET.

```
LOCK TABLE dept IN SHARE MODE
```

```
UPDATE emp
```

```
SET sal = sal * 1.1
```

```
WHERE deptno IN (SELECT deptno FROM dept WHERE loc = 'DALLAS')
```

```
UPDATE dept
```

```
SET totalsal = totalsal * 1.1
```

```
WHERE deptno IN (SELECT deptno FROM dept WHERE loc = 'DALLAS')
```

```
COMMIT
```

Следующие примеры предложений CREATE TABLE показывают определения нескольких ограничений целостности:

```
CREATE TABLE dept (  
    deptno NUMBER(3) PRIMARY KEY,  
    dname VARCHAR2(15),  
    loc VARCHAR2(15)  
    CONSTRAINT dname_ukey UNIQUE (dname, loc),  
    CONSTRAINT loc_check1  
    CHECK (loc IN ('NEW YORK', 'BOSTON', 'CHICAGO')));  
CREATE TABLE emp (  
    empno NUMBER(5) PRIMARY KEY,
```

```

ename VARCHAR2(15) NOT NULL,
job VARCHAR2(10),
mgr NUMBER(5) CONSTRAINT mgr_fkey
REFERENCES emp,
hiredate DATE,
sal NUMBER(7,2),
comm NUMBER(5,2),
deptno NUMBER(3) NOT NULL
CONSTRAINT dept_fkey
REFERENCES dept ON DELETE CASCADE);

```

Определение ограничений целостности в команде ALTER TABLE

Вы можете также определять ограничения целостности с помощью фразы CONSTRAINT команды ALTER TABLE. Следующие примеры предложений ALTER TABLE показывают определения нескольких ограничений целостности:

```

ALTER TABLE dept
    ADD PRIMARY KEY (deptno);
ALTER TABLE emp
    ADD CONSTRAINT dept_fkey FOREIGN KEY (deptno) REFERENCES dept
    MODIFY (ename VARCHAR2(15) NOT NULL);

```

Содержание отчета.

1. Титульный лист.
2. Распечатка кода программы, реализованной с помощью транзакций.
3. Распечатка предложений SQL, осуществляющих блокировку.
4. Распечатка команды создания таблицы с наложенными ограничениями целостности.

Контрольные вопросы.

1. Что такое транзакция?
2. В каком случае можно отменить все сделанные операторами из транзакции изменения в БД?

3. В каких случаях транзакция фиксируется автоматически?
4. В каких случаях происходит откат в транзакции?
5. Какие действия происходят в процессе выполнения транзакции с операторами изменения данных?
6. Что такое точки сохранения? Как они используются?
7. Объясните назначение опции TO SAVEPOINT в предложении ROLLBACK.
8. Для чего предназначен механизм блокировки данных?
9. Перечислите основные типы блокировок ресурсов ORACLE.
10. На каких уровнях могут быть перекрыты механизмы автоматических блокировок ORACLE?
11. Для чего используется опция NOWAIT предложения LOCK TABLE?
12. Для чего предназначены ограничения целостности?
13. Какие вы знаете типы ограничения целостности?
14. Для каких типов ограничения целостности можно использовать множественные ограничения?
15. В чем различие первичного и уникального ключа?
16. Как через ограничения целостности можно выразить связи между родительскими и порожденными таблицами?

Комплексные объекты ORACLE.

Представления, Последовательности и синонимы.

Представления.

Представление - это поименованная, динамически поддерживаемая сервером выборка из одной или нескольких таблиц. Оператор SELECT, определяющий выборку, ограничивает видимые пользователем данные. Кроме того, представление позволяет эффективно ограничить данные, которые пользователь может модифицировать. Сервер гарантирует актуальность представления, то есть формирование представления (материализация соответствующего запроса) производится каждый раз при использовании представления. Используя представления, администратор базы данных ограничивает доступную пользователям часть логического пространства базы

данных только теми данными, которые реально необходимы для выполнения его работы.

Оператор определения представлений Oracle использует следующий синтаксис:

```
CREATE [OR REPLACE] [{FORCE | NO FORCE}]  
VIEW [имя_схемы.] имя_представления  
[(альтернативное_имя [альтернативное_имя...])]  
AS запрос WITH { READ ONLY | CHECK OPTION  
[CONSTRAINT ограничение_целостности ]}
```

Ключевое слово OR REPLACE указывает на принудительное замещение старого представления новым, образуемым командой CREATE.

Ключевое слово FORCE указывает на принудительное создание представления вне зависимости от того, существуют ли базовые таблицы представления и есть ли у пользователя, создающего представление, привилегии на выборку из базовых таблиц.

Ключевое слово NO FORCE (используемое по умолчанию) указывает на обязательность существования базовых таблиц и наличия у пользователя, создающего представление, привилегий на доступ к базовым таблицам. При нарушении какого-либо из этих условий представление не создается.

Параметр *запрос* используется для обозначения любого синтаксически правильного запроса, не содержащего ключевого слова ORDER BY или ключевого слова FOR UPDATE.

Ключевое слово WITH READ ONLY указывает на запрещение для данного представления операций модификации данных, то есть невозможно с помощью стандартных методов предоставления привилегий разрешить какому-либо пользователю выполнять над строками данного представления операции INSERT, UPDATE или DELETE.

Ключевое слово WITH CHECK OPTION указывает на то, что строки, вводимые в представление в результате операций INSERT и UPDATE, должны соответствовать критерию отбора в запросе, определяющем представление.

Отметим, что при наличии сложного запроса с подзапросами корректность данной проверки не гарантируется.

Ключевое слово CONSTRAINT определяет имя ограничения, используемое для проверки. Если имя опущено, сервер генерирует имя SYS_Cn, где вместо n сервер автоматически подставляет число, гарантирующее уникальность имени ограничения.

Рассмотрим пример создания представления VU1 с ограничением *только для чтения*. Попытка записи данных в представление VU1 отвергается системой, несмотря на явное предоставление привилегии на вставку данных пользователю U2. Вставка данных в базовую таблицу Tab2 выполняется успешно. Ниже приведен протокол создания и манипулирования представлением VU1.

```
SQL> CONNECT U1/U1PSW@SUNORA;
```

Connected.

```
SQL> CREATE TABLE Tab2(At1 NUMERIC, At2 NUMERIC);
```

Table created.

```
SQL> CREATE VIEW VU1 AS SELECT * FROM Tab2 WITH READ ONLY;
```

View created.

```
SQL> GRANT INSERT ON Tab2 TO U2;
```

Grant succeeded.

```
SQL> GRANT INSERT ON VU1 TO U2;
```

Grant succeeded.

```
SQL> CONNECT U2/U2PSW@SUNORA;
```

Connected.

```
SQL> INSERT INTO U1.VU1 VALUES (1,2);
```

```
INSERT INTO U1.VU1 VALUES (1,2)
```

*

ERROR at line 1:

ORA-01732: data manipulation operation not legal on this view

```
SQL> INSERT INTO U1.Tab2 VALUES (1,2);
```

1 row created.

Для того, чтобы конструктивно работать с представлением, пользователь должен, как минимум, иметь привилегию SELECT во всех таблицах, которые участвуют в запросе, формирующем данные представления. Поэтому привилегии, которыми обладает пользователь на базовые таблицы, наследуются представлением для пользователя, который его создает. Если пользователь обладает любой комбинацией привилегий INSERT, UPDATE, DELETE для базовых таблиц, то эти привилегии будут автоматически наследоваться представлением. В то же время пользователь, не имеющий привилегий на модификацию строк базовых таблиц, не может получить соответствующие привилегии в представлении.

Еще одно полезное свойство представлений состоит в том, что они позволяют реализовать доступ пользователей к данным, которые являются производными от данных базовых таблиц. Например, пусть пользователь U1 создает представление, в котором для таблицы Tab2 вычисляются поэлементная сумма и среднее значение значений в столбцах At1 и At 2, соответственно. Пользователю U2 предоставляется право выборки из представления, но не базовой таблицы. Ниже приведен протокол примера, иллюстрирующего данное свойство.

```
SQL> CONNECT U1/U1PSW@SUNORA;
```

Connected.

```
SQL> CREATE VIEW VU3 (Sat1, AVGA2) AS SELECT SUM (At1), AVG  
(At2) FROM Tab2;
```

View created.

```
SQL> GRANT SELECT ON VU3 TO U2;
```

Grant succeeded.

```
SQL> CONNECT U2/U2PSW@SUNORA;
```

Connected.

```
SQL> SELECT * FROM U1.VU3;
```


SAT1	AVGAT2
5	2.67

5

2.67

```
SQL> SELECT * FROM U1.Tab2;
```

```
SELECT * FROM U1.Tab2
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01031: insufficient privileges
```

Удаление представления из базы данных выполняется командой DROP VIEW. Для удаления представления необходимо быть его владельцем или иметь привилегию DROP ANY VIEW.

```
DROP VIEW [имя_схемы.}имя_представления
```

При уничтожении представления синонимы и представления, ссылающиеся на уничтожаемое представление, не уничтожаются, а становятся неверными. Их можно уничтожить или переопределить, а также можно переопределить другие представления и таблицы, с ним связанные, чтобы представления и синонимы стали вновь актуальными. Привилегии на удаляемое представление отменяются.

Представление можно "изменить" путем его отмены и повторного создания.

Для установления связей представления используют представление словаря данных USER_CROSS_REFS для определения представлений и таблиц, связанных с данным представлением.

Последовательности.

Последовательностью называется объект БД Oracle, генерирующий неповторяющиеся целые числа. Полученные из последовательности числа очень часто используются в качестве значений для первичных ключей.

Чтобы пользоваться последовательностью, пользователь должен иметь на нее право выборки (SELECT). Доступные последовательности перечислены в представлениях словаря данных USER_SEQUENCES и ALL_SEQUENCES.

Числа, создаваемые последовательностью, могут либо возрастать постоянно, либо только до определенного предела, либо, по достижении предела, начинать возрастание заново, с начального значения. Последовательность может создавать цепочки как увеличивающихся чисел, так и уменьшающихся. Можно задавать также и приращение значений.

Псевдостолбец NEXTVAL используется для генерирования очередного номера из указанной последовательности. Ссылка на NEXTVAL приводит к генерированию очередного номера. Обращение имеет следующий синтаксис:

имя_последовательности.NEXTVAL

где параметр *имя_последовательности* - это имя последовательности.

Псевдостолбец CURRVAL используется для ссылки на текущее значение последовательного номера. В текущем сеансе NEXTVAL должен быть использован хотя бы один раз до ссылки на CURRVAL. Обращение к CURRVAL имеет следующий синтаксис:

имя_последовательности.CURRVAL

где параметр *имя_последовательности* задает имя последовательности, на которую ссылался CURRVAL.

Для создания последовательности требуется привилегия CREATE SEQUENCE. Для создания последовательности, размещенной в схеме другого пользователя, необходима привилегия CREATE ANY SEQUENCE.

Оператор определения последовательности Oracle использует следующий синтаксис:

```
CREATE SEQUENCE [имя_схемы.]имя_последовательности
[INCREMENT BY приращение]
[START WITH начальное_значение]
[MAXVALUE наибольшее_значение | NOMAXVALUE]
[MINVALUE наименьшее_значение | NOMINVALUE]
{CYCLE | NOCYCLE}
[CACHE число_элементов | NOCACHE]
[ORDER | NOORDER]
```

Параметр *имя_последовательности* задает имя последовательности. Имя последовательности должно соответствовать соглашениям Oracle по именованию объектов базы данных. Параметр *имя_схемы* указывает на схему, в которой определяется последовательность. Если владелец последовательности не указан явно, то подразумевается пользователь, выдавший команду CREATE SEQUENCE.

Ключевое слово INCREMENT BY определяет интервал между последовательными номерами. Если параметр *приращение* имеет отрицательное значение, то последовательность убывающая, если положительное - последовательность возрастающая. Допустимо любое целое число не равное нулю. Значение по умолчанию 1 (возрастающая последовательность значений).

Ключевое слово START WITH через параметр *начальное_значение* задает первый генерируемый последовательный номер. Если ключевое слово не указано, то по умолчанию для возрастающих последовательностей начальный генерируемый последовательный номер равен значению параметра MINVALUE, а для убывающих последовательностей - MAXVALUE.

Ключевое слово MAXVALUE через параметр *наибольшее_значение* задает максимальное значение последовательного номера, которое будет генерироваться. Параметр *наибольшее_значение* определяет верхнюю границу последовательности, которая может быть любым целым числом, с количеством знаков, не превышающим 28 цифр, и большим чем параметры начальное значение и *наименьшее_значение* (если они заданы). Отсутствие верхней границы указывается ключевым словом NOMAXVALUE, которое определяет для убывающих последовательностей значение -1, а для возрастающих последовательностей - 1027.

Ключевое слово MINVALUE через параметр *наименьшее_значение* задает минимальное значение последовательного номера, которое будет генерироваться. Параметр *наименьшее_значение* определяет нижнюю границу последовательности, которая может быть любым целым числом, с количеством

знаков, не превышающим 28 цифр и меньшим, чем параметры *наибольшее_значение* и *начальное_значение* (если значение параметров задано). Отсутствие нижней границы указывается ключевым словом NOMINVALUE, которое определяет для убывающих последовательностей значение -1026, а для возрастающих последовательностей значение -1.

Ключевое слово NOCYCLE является значением, используемым по умолчанию, и предполагает завершение генерирования последовательных номеров по достижении конца последовательности. Если при определении последовательности указан параметр CYCLE, то после достижения очередным членом последовательности значения параметра *наибольшее_значение* (для возрастающих последовательностей) выдается значение параметра *наименьшее_значение*. Если параметры *наибольшее_значение* и *наименьшее_значение* не указаны, то используются их значения по умолчанию.

Ключевое слово CACHE указывает на использование техники предварительной подготовки элементов последовательности, что обеспечивает их быстрое получение при запросе. Число последовательных номеров, хранящихся в области кеша оперативной памяти, определяется параметром *число_элементов* ключевого слова CACHE. Кеширование последовательности обеспечивает более быструю генерацию элементов последовательности. Заполнение кеша для каждой данной последовательности происходит после запроса первого элемента этой последовательности.

Ключевое слово ORDER обеспечивает генерацию последовательных элементов точно в порядке поступления запросов. В большинстве случаев независимо от того, указано ли ключевое слово ORDER, элементы последовательности генерируются в порядке поступления запросов.

Одна последовательность может использоваться для генерации первичных ключей для нескольких таблиц. Если два пользователя одновременно обращаются к одной последовательности, номера для каждого пользователя могут иметь промежутки, так как из непрерывной последовательности попеременно получают номера оба пользователя. Каждый

из двух пользователей не будет видеть последовательные номера, сгенерированные для другого пользователя. При генерации элемента последовательности счетчик элементов увеличивается независимо от того, успешно или неуспешно завершена транзакция.

Последовательности могут экспортироваться и импортироваться, как и другие объекты Oracle.

Для удаления последовательностей используется команда DROP SEQUENCE. Для выполнения данной операции необходимо быть владельцем последовательности либо иметь привилегию DROP ANY SEQUENCE.

Оператор удаления последовательностей Oracle использует следующий синтаксис:

DROP SEQUENCE [имя_схемы.]имя_последовательности

Одной из ситуаций, когда необходимо уничтожение последовательности, является повторный старт последовательности.

Синонимы.

Синоним - это объект БД Oracle, используемый для альтернативного именования. Обычно синонимы создаются для таблиц, представлений, последовательностей для общего использования (без указания префикса схемы) или для скрытой ссылки на удаленную базу данных. Наличие синонимов позволяет приложениям обеспечивать независимость от того, в какой схеме размещена таблица или представление, а также - в какой конкретно локальной базе данных распределенной системы хранятся требуемые приложению данные.

Для создания синонима необходимо быть владельцем или иметь привилегию SELECT объекта, для которого создается синоним. Чтобы создать синоним типа PUBLIC или синоним для объекта в схеме другого пользователя, необходимо иметь привилегию CREATE PUBLIC SYNONYM или CREATE ANY SYNONYM, соответственно.

Оператор определения синонима Oracle использует следующий синтаксис:

```
CREATE [PUBLIC] SYNONYM [имя_схемы.]имя_синонима  
FOR [имя_схемы.]имя_объекта[@ имя_связиБД]
```

Ключевое слово PUBLIC определяет, что синоним будет доступен всем пользователям. По умолчанию синоним доступен только создавшему его пользователю.

Параметр *имя_синонима* - имя синонима, следующее соглашениям по именованию объектов Oracle. Параметр *имя_схемы* задает существующее в базе данных имя схемы. Для создания синонима в произвольной схеме нужны соответствующие привилегии. Если имя схемы для синонима опущено, то предполагается, что синоним создается в схеме пользователя, выдавшего команду.

Параметр *имя_связиБД* указывает на существующую связь к удаленной базе данных. Если параметр *имя_схемы* опущен, синоним ссылается к объекту, принадлежащему пользователю, определенному связью с удаленной базой данных.

Синоним должен иметь имя, отличное от остальных объектов данного пользователя.

Для удаления из базы данных синонима используется команда DROP SYNONYM.

Для удаления личного синонима необходимо быть его владельцем или иметь привилегию DROP ANY SYNONYM. Для удаления общего синонима необходимо также иметь привилегию DROP ANY SYNONYM.

Оператор удаления синонима Oracle использует следующий синтаксис:

```
DROP [PUBLIC] SYNONYM [имя_схемы.]имя_синонима
```

Необязательное ключевое слово PUBLIC определяет факт удаления общего синонима.

Параметр *имя_синонима* определяет имя удаляемого синонима.

Изменить синоним можно, отменив его и назначив заново.

Формирование запросов.

Рассмотрим полный синтаксис средства формирования запросов к базе данных. Запрос к базе данных выполняет оператор выборки, определяемый ключевым словом SELECT. Операция SELECT наилучшим образом демонстрирует изящество и мощь средств выборки данных в реляционных системах управления базами данных.

Операция SELECT используется для выборки атрибутов одного или нескольких отношений в соответствии с указанным критерием отбора. Фрагменты предложения SELECT должны быть записаны в указанном порядке.

Предложение SELECT имеет в Oracle следующий синтаксис:

```
SELECT [DISTINCT | ALL] { * | { [имя_схемы. ] {  
имя_таблицы | имя_представления | имя_снимка }.* |  
выражение [ [AS] альтернативное_имя_столбца ] }  
[, { [имя_схемы.]{ имя_таблицы | имя_представления |  
имя_снимка }.* | выражение  
[ [AS] альтернативное_имя_столбца ]}]...}  
FROM {[имя_схемы. ]{{ имя_таблицы | имя_представления |  
имя_снимка } [ @ имя_связиБД ] } | (имя_подзапроса) }  
[ локальное_альтернативное_имя]  
[, [[имя_схемы. ]{{ имя_таблицы | имя_представления | имя_снимка }  
[@имя_связиБД ]} | (имя_подзапроса) }  
[локальное_альтернативное_имя] ] ...  
[WHERE условие ]  
{ {[GROUP BY выражение [, выражение ] ...  
[HAVING условие ]}  
| {[START WITH условие] CONNECT BY условие} }...  
[{ UNION | UNION ALL | INTERSECT | MINUS) предложение_SELECT ]  
[ORDER BY { выражение | положение | альтернативное_имя_столбца }  
[ASC | DESC]  
[, { выражение | положение | альтернативное_имя_столбца ]
```

[ASC | DESC]] ...]

[FOR UPDATE [OF [[имя_схемы .]{ имя_таблицы |
имя_представления }] имя_столбца [, [имя_схемы .]
{ имя_таблицы | имя_представления }] имя_столбца ...]
[NOWAIT]]

Указание ключевого слова DISTINCT приводит к устранению из отобранных данных повторяющихся кортежей. Указание ключевого слова ALL приводит к предъявлению всех отобранных данных, включая повторяющиеся кортежи. По умолчанию используется значение ALL.

Наличие параметра "*" (звездочка) означает выбор всех столбцов из всех таблиц, представлений и снимков, указанных в перечне значений ключевого слова FROM.

Конкретный выбор значения параметра {имя_таблицы | имя_представления | имя_снимка}. * означает выбор всех столбцов из таблицы, представления или снимка. Необязательный параметр имя_схемы используется для уточнения имени схемы, в которой находится соответствующий объект Oracle. По умолчанию используется схема пользователя, выполняющего запрос.

Параметр *выражение* заменяется на вычисляемое выражение, которое обычно базируется на данных столбцов из таблиц, представлений и снимков, указанных в перечне значений ключевого слова FROM.

Параметр *альтернативное_имя_столбца* задает альтернативное имя столбца или выражения для формирования заголовка при выводе ответа на запрос. Заданное значение параметра может также использоваться в выражении ORDER BY.

Ключевое слово FROM определяет таблицы, представления или снимки, из которых будут отбираться данные.

Параметр *имя_связиБД* устанавливает имя связи с удаленной базой данных. Если имя связи с удаленной базой данных не указано, предполагается, что соответствующий объект Oracle расположен в основной базе данных.

Параметр *имя_подзапроса* задает подзапрос, который в данном контексте рассматривается также, как представление.

Параметр *локальное_альтернативное_имя* задает альтернативное, обычно короткое, имя, которое в контексте данного запроса является обязательным для ссылок именем соответствующей таблицы, представления или снимка.

Ключевое слово WHERE определяет логическое условие отбора данных. Если ключевое слово WHERE опущено, то осуществляется выбор всех данных из таблиц, представлений и снимков, указанных в перечне значений ключевого слова FROM.

Ключевые слова GROUP BY и HAVING используются для формирования некоторой обобщающей информации о группах строк, имеющих определенные значения в одном или нескольких полях, описываемые параметрами выражение и условие. Для каждой группы строк, формируемой предложением GROUP BY выражение, создается одна строка производных данных. Формируемая группа строк может быть уточнена логическим условием отбора, определяемым предложением HAVING.

Ключевые слова START WITH и CONNECT BY задают иерархический порядок отбора данных запроса. Конкретная иерархическая упорядоченность задается параметрами условие.

Ключевые слова UNION, UNION ALL, INTERSECT, MINUS задают теоретико-множественные операции объединения результатов нескольких запросов, сформированных в соответствии со значением параметра *предложение_SELECT*. Для объединенных результатов не допускается использование ключевого слова FOR UPDATE.

Ключевое слово ORDER BY определяет порядок, в котором будут выдаваться строки результирующего отношения. Параметр выражение определяет значение, по которому выполняется сортировка. Базис сортировки может также быть указан параметром положение, то есть порядковым номером

в списке вывода (задаваемом после ключевого слова SELECT). По умолчанию используется сортировка по возрастанию (ASC).

Ключевое слово FOR UPDATE определяет необходимость блокировки отобранных строк. Необязательное ключевое слово OF уточняет перечень таблиц или представлений, данные из которых должны быть заблокированы.

Необязательное ключевое слово NOWAIT указывает на то, что управление будет передано следующему после SELECT оператору, даже если требуемые для блокировки строки недоступны (то есть заблокированы другим процессом). Если ключевое слово NOWAIT не указано, то выполнение запроса будет приостановлено до тех пор, пока не будут освобождены все требуемые для блокировки строки.

Обратите внимание, что выполнение запроса, не содержащего ключевого слова ORDER BY, не предполагает какого-либо упорядочивания вывода. Одно и то же предложение SELECT, выполненное с теми же самыми данными в разное время, может приводить к выводу различно упорядоченных строк. Также порядок вывода необязательно тот, в котором вводились данные.

Пользователи, связи и снимки.

Пользователи.

Пользователь - это базовый объект Oracle, который может быть владельцем других объектов и выполнять предусмотренные действия с объектами Oracle в соответствии с поддерживаемой технологией.

Оператор определения пользователей Oracle использует следующий синтаксис:

```
CREATE USER имя_пользователя IDENTIFIED
{ BY пароль | EXTERNALLY }
[DEFAULT TABLESPACE имя_табличной_области1]
[TEMPORARY TABLESPACE имя_табличной_области2]
[QUOTA [число _единиц [{ K | M }]] I UNLIMITED ]
ON имя_табличной_области]
[PROFILE имя_профиля]
```

Параметр *имя_пользователя* задает имя, под которым пользователь регистрируется в системе. Имя пользователя должно быть указано в кодировке, поддерживаемой сервером.

Ключевое слово **BY** указывает, что подтверждающий подлинность пользователя пароль будет указан явно в параметре *пароль*. Если указан параметр **EXTERNALLY**, то сервер проверяет соответствие зарегистрированного операционной системой пользователя и пользователя Oracle. Если имена совпадают, то Oracle не проводит собственную аутентификацию.

Ключевое слово **DEFAULT TABLESPACE** указывает имя табличной области, задаваемое параметром *имя_таблич-ной_области1*, которая используется для объектов создаваемого пользователя по умолчанию. Если ключевое слово **DEFAULT TABLESPACE** не указано, то для объектов создаваемого пользователя будет использоваться табличная область **SYSTEM**. Учитывая, что в табличной области **SYSTEM** размещен словарь данных, использовать ее для пользовательских объектов нецелесообразно.

Ключевое слово **TEMPORARY TABLESPACE** указывает имя табличной области, задаваемое параметром *имя_табличной_области2*, которая используется для временных сегментов создаваемого пользователя по умолчанию. Временные сегменты используются для хранения промежуточных данных сложных запросов. Если ключевое слово **TEMPORARY TABLESPACE** не указано, то для временных сегментов создаваемого пользователя будет использоваться табличная область **SYSTEM**.

Ключевое слово **QUOTA** задает ограничения на используемое пользователем пространство в конкретной табличной области. Максимально допустимое пространство задается параметром *число_единиц* в мегабайтах, если указано ключевое слово **M**; если указано ключевое слово **K** – в килобайтах; а если не указано ни **M**, ни **K** – в байтах. Указание ключевого слова **UNLIMITED** разрешает пользователю использовать табличное пространство без ограничений.

Ключевое слово PROFILE назначает регистрируемому пользователю профиль, задаваемый параметром *имя_профиля*. Профиль пользователя – это поименованный набор ограничений на ресурсы системы. Если ключевое слово PROFILE не указано, пользователю приписывается профиль DEFAULT.

Для успешной регистрации пользователя необходимо иметь привилегию CREATE USER.

Для исключения из базы данных пользователя используется оператор DROP USER. При исключении пользователя должны быть удалены все объекты, принадлежащие этому пользователю. Для выполнения операции исключения пользователя необходимо иметь привилегию DROP USER.

Оператор исключения пользователя Oracle использует следующий синтаксис:

DROP USER *имя_пользователя* [CASCADE]

Параметр *имя_пользователя* задает имя пользователя в системе. Если указано ключевое слово CASCADE, то автоматически удаляются все объекты исключаемого пользователя. Если ключевое слово CASCADE не указано, а в схеме пользователя содержатся объекты, возвращается сообщение об ошибке. Связь с удаленной базой данных.

Для создания связи из локальной базы данных с объектом в удаленной базе данных используется команда CREATE DATABASE LINK. Oracle поддерживает как связь с удаленной базой данных Oracle, так и с базами данных некоторых других производителей, например, DB2 фирмы IBM. Естественно, что и на локальной, и на удаленной базе данных должно быть установлено программное обеспечение SQL*Net.

Для создания связи с удаленной базой данных необходимо иметь некоторую точку входа в удаленной базе данных. Данная точка входа может совпадать с именем пользователя, который создает связь, а может быть иной точкой входа. Для создания общей (PUBLIC) связи необходимо обладать привилегией CREATE PUBLIC DATABASE LINK.

Оператор создания связи с удаленной базой данных Oracle использует следующий синтаксис:

```
CREATE [PUBLIC] DATABASE LINK имя_связиБД [CONNECT TO  
имяпользователя IDENTIFIED BY пароль_пользователя] USING  
'строка_связи'
```

Если параметр PUBLIC опущен, создается связь, доступная только создавшему ее пользователю. Если же параметр определен, связь становится доступной всем пользователям.

Параметр *имя_связиБД* задает имя, которое дается создаваемой связи. Параметры *имя_пользователя*, *пароль_пользователя* задают имя пользователя и пароль в удаленной базе данных. Параметр '*строка_связи*' определяет имя раздела параметров SQL*Net, задающего спецификации связи с удаленной базой данных.

Создав связь с удаленной базой данных, можно обращаться к таблицам удаленной базы в запросах, ссылаясь на них во фразе FROM с тем же эффектом, что и при прямом подключении определенного пользователя в удаленной базе.

Если фраза, содержащая имя пользователя и пароль, отсутствует, будет использоваться имя и пароль текущего пользователя. Например, если U1 выдаст команду создания связи и не задаст пользователя и пароль, Oracle будет использовать для доступа к удаленной базе данных имя пользователя U1 и его пароль.

Удаленные таблицы определяются добавлением к таблице имени связи (@*имя_связи*) во фразе FROM оператора SELECT.

Запросы, выполняемые с использованием связи с удаленной базой данных, подчиняются следующим ограничениям:

- максимальное количество связей с удаленными базами, которые можно использовать в одном запросе, определяется значением параметра OPEN_LINKS файла инициализации;
- через связь не могут быть выбраны столбцы типа LONG;

- использование команд языка манипулирования данными INSERT, DELETE, UPDATE требует наличия установки для сервера Oracle возможностей, реализуемых компонентой Distributed Options.

Определим связь с удаленной базой данных ora_sun_link, которая связывает пользователя U1 с паролем U1PSW в базе данных, определяемой строкой связи orasun. После успешного создания связи с удаленной базой данных можно выполнить запрос, как показано в приведенном ниже примере.

```
SQL> create database link sun_ora_link
```

```
2 connect to u1 identified by u1psw
```

```
3 using 'sunora';
```

Database link created.

```
SQL> select * from tab1@sun_ora_link;
```

ATI

1

2

Для того чтобы скрыть от пользователя факт, что таблица Tab1 пользователя U1 находится на удаленной базе, можно использовать синоним.

```
SQL> create synonym suntab1 for u1.tab1@sun_ora_link;
```

Synonym created.

```
SQL> select * from suntab1;
```

ATI

1

2

Для удаления определенной связи с удаленной базой данных используется команда DROP DATABASE LINK. Для выполнения этой команды необходимо быть либо владельцем связи с удаленной базой данных, либо иметь привилегию DROP ANY DATABASE LINK. Для отмены общей связи необходимо иметь привилегию DROP PUBLIC DATABASE LINK.

Оператор уничтожения связи с удаленной базой данных Oracle использует следующий синтаксис:

DROP [PUBLIC] DATABASE LINK *имя_связиБД*

Параметр PUBLIC должен быть определен для отмены общей связи с удаленной базой данных.

Параметр *имя_связиБД* задает имя отменяемой связи.

Пользователь, обладающий привилегией DROP ANY DATABASE LINK, может отменить связь, владельцем которой является другой пользователь (независимо от привилегий данного пользователя).

Рассмотрим пример отмены (удаления) связи с удаленной базой данных по имени sun_oracle_link.

```
SQL> drop database link sun_oracle_link;
```

Database link dropped.

Снимки.

Снимок - это поименованная, динамически поддерживаемая сервером выборка из одной или нескольких таблиц или представлений, обычно размещенных на удаленной базе данных. Сервер гарантирует актуальность снимка в рамках принятой технологии, а именно: формирование снимка (материализация соответствующего запроса) производится в соответствии с некоторым расписанием. Используя снимки, администратор безопасности обеспечивает доступ пользователям к тем частям удаленных баз данных, которые реально необходимы для выполнения их работы.

Для того чтобы механизм снимков работал, на серверах локальной и удаленной баз данных должен быть установлен пакет DBMS_SNAPSHOT, в котором размещены Процедуры, выполняющие обновление снимков. Для серверов с Procedural Option такой пакет включается автоматически. Для остальных серверов для создания пакета необходимо, чтобы пользователь SYS выполнил следующие сценарии: dbmsnap.sql и prvtsnap.plb. Обычно эти файлы находятся в каталоге \$oracle_home/rdbms/admin, хотя их имена и размещение зависят от платформы (например, Personal Oracle для Windows 95 размещает эти файлы в каталоге \$oracle_home/rdbms72/admin). Для создания снимков, использующих таблицы и представления удаленной базы данных, необходимо,

чтобы Oracle был установлен с дополнительными возможностями, реализуемыми компонентой Distributed Option.

Рассмотрим пример создания снимка с таблицы, размещенной на удаленном сервере. Пусть таблица Tab1 размещена на удаленном сервере, и доступ к ней осуществляется через связь с именем sun_ora_link. Приводимый пример показывает, как создается снимок, и иллюстрирует тот факт, что изменения в мастер-таблице не распространяются на данные снимка автоматически.

```
SQL> create snapshot snap_suntab1 as select
* from tab1 @sun_ora_link where at1 > 0;
```

Snapshot created.

```
SQL> select * from snap_suntab1;
```

AT1

1

2

4

```
SQL> connect u1/u1psw@sunora
```

Connected.

```
SQL> insert into tab1 values(10);
```

1 row created.

```
SQL> delete from tab1 where at1 = 1;
```

1 row deleted.

```
SQL> connect system/managers 2:
```

Connected.

```
SQL> select * from snap_suntab1;
```

AT1

1

2

4

Для модификации снимка с целью установки частоты автоматического изменения в 1 час можно воспользоваться командой ALTER SNAPSHOT. После того как введением ключевого слова REFRESH снимок сделан обновляемым автоматически, изменения, введенные в мастер-таблицу в предыдущем примере, актуализируются в снимке автоматически.

```
SQL> alter snapshot snap_suntab1 refresh complete
```

```
2 start with sysdate next sysdate + 1/24;
```

```
Snapshot altered.
```

```
SQL> select * from snap_suntab1
```

```
AT1
```

```
2
```

```
4
```

```
10
```

Практическое занятие № 4.

Цель работы: Освоение средств разработки запросов, пользователей, последовательностей и синонимов, знакомство со специфичными для распределенных систем объектами ORACLE – снимками и связями с удаленными БД.

Содержание отчета.

1. Титульный лист.
2. Распечатка листинга создания собственного представления.
3. Распечатка листинга протокола регистрации пользователя и исключения (удаления) пользователя.
4. Распечатка листинга создания последовательности.
5. Распечатка листинга создания синонима для ранее созданной таблицы.

Пример выполнения работы.

Создадим представление VU2 с проверкой принадлежности вводимых данных области допустимых значений. Для представления VU2 попытка вставки данных, не удовлетворяющих критерию запроса, отвергается системой.

```
SQL> CONNECT U1/U1PSW@SUNORA;
```

Connected.

```
SQL> CREATE VIEW VU2 AS SELECT * FROM Tab2
WHERE At1 > 1 WITH CHECK OPTION;
```

View created.

```
SQL> GRANT INSERT, SELECT ON VU2 TO U2;
```

Grant succeeded.

```
SQL> CONNECT U2/U2PSW@SUNORA;
```

Connected.

```
SQL> INSERT INTO U1.VU2 VALUES (1,2);
```

```
INSERT INTO U1.VU2 VALUES (1,2)
```

*

ERROR at line 1:

ORA-01402: view WITH CHECK OPTION where-clause violation

```
SQL> INSERT INTO U1.VU2 VALUES (2,3);
```

1 row created.

```
SQL> SELECT * FROM U1.VU2;
```

```
AT1  AT2
```

```
-----
```

```
2    3
```

Создадим пользователя U1, который использует пароль U1PSW для подтверждения подлинности и которому назначена табличная область по умолчанию app_data с ограничением на используемое пространство в 1 мегабайт. Пользователю U1 также разрешено использовать табличную область tools с ограничением 500 килобайт.

```
SQL> CREATE USER U1 IDENTIFIED BY U1PSW
```

```
2  DEFAULT TABLESPACE APP_DATA
```

```
3  QUOTA 1M ON APP_DATA
```

```
4  QUOTA 500K ON TOOLS;
```

User created.

Выполним операцию исключения пользователя.

```
SQL> DROP USER U1;
```

```
DROP USER U1
```

```
*
```

```
ERROR at line 1:
```

```
ORA-01922: CASCADE must be specified to drop 'U1'
```

```
SQL> drop user u1 cascade;
```

```
User dropped.
```

Создадим последовательность с именем Seq1. Начальный элемент последовательности определен равным 2, параметры наибольшее_значение и наименьшее_значение определены равными 3 и 1 соответственно.

```
SQL> CREATE SEQUENCE Seq1
```

```
2 MAXVALUE 3 MINVALUE 1 START WITH 2;
```

```
Sequence created.
```

```
SQL> SELECT Seq1.NEXTVAL FROM DUAL;
```

```
NEXTVAL
```

```
2
```

```
SQL> SELECT Seq1.NEXTVAL FROM DUAL
```

```
NEXTVAL
```

```
3
```

```
SQL> SELECT Seq1.NEXTVAL FROM DUAL
```

```
ERROR:
```

```
ORA-08004: sequence SEQ1.NEXTVAL exceeds MAXVALUE and cannot  
be instantiated no rows selected
```

Уничтожим последовательность Seq1:

```
SQL> DROP SEQUENCE Seq1;
```

```
Sequence dropped.
```

Создание синонима. Пусть в таблице RefCodes пользователя Administrator находится информация, необходимая для замены некоторых кодов на действительные наименования некоторых объектов. Для пользователя U1 заводится одноименный синоним, упрощающий обращение к требуемой

таблице (конечно, одноименность не является обязательным требованием). Наличие необходимых привилегий у пользователя, создающего синоним, предполагается. Обратите внимание на необходимость явного разрешения пользователю U1 выполнять операцию выборки из таблицы RefCodes (даже несмотря на то, что она скрыта за синонимом). Пусть таблица RefCodes сформирована и загружена пользователем Administrator с использованием следующих предложений:

```
CREATE TABLE RefCodes(At1 NUMBER, At2 VARCHAR2(5));  
INSERT INTO RefCodes VALUES(1,'Text1');  
INSERT INTO RefCodes VALUES (2,'Text2');
```

Пользователь SYSTEM создает синоним для пользователя U1, ссылаясь на таблицу RefCodes из схемы пользователя Administrator. При этом, несмотря на правильность синонима, объект, на который ссылается синоним, пользователю U1 недоступен.

```
SQL> CONNECT SYSTEM/MANAGER@SUNORA;
```

Connected.

```
SQL> CREATE SYNONYM U1.REFCODES FOR  
Administrator.RefCodes;
```

Synonym created.

```
SQL> CONNECT U1/U1PSW@SUNORA;
```

Connected. SQL> SELECT * FROM REFCODES;

```
SELECT * FROM REFCODES
```

```
*
```

ERROR at line 1:

ORA-00942: table or view does not exist

После предоставления прав на выполнение операции выборки пользователь U1 может оперировать с объектом, скрытым за синонимом.

```
SQL> CONNECT ADMINISTRATOR/ADMINPSW@SUNORA;
```

Connected.

```
SQL> GRANT SELECT ON REFCODES TO U1;
```

Grant succeeded.

```
SQL> CONNECT U1/U1PSW@SUNORA;
```

Connected.

```
SQL> SELECT * FROM REFCODES;
```

AT1	AT2
-----	-----

----	-----
------	-------

1	Text1
---	-------

2	Text2
---	-------

Контрольные вопросы.

1. Что такое представления? Для чего они предназначены?
2. Назначение и формат оператора CREATE VIEW.
3. Объясните преимущества использования ключевого слова OR REPLACE оператора CREATE VIEW.
4. Использование привилегий при работе с представлениями.
5. Перечислите условия, выполнение которых необходимо при удалении представлений.
6. Что такое пользователь – объект ORACLE?
7. Формат команды CREATE USER и DROP USER.
8. Создание запросов в ORACLE. Формат команды SELECTE.
9. Создание связей с удаленной БД в ORACLE.
10. Перечислите преимущества, предоставляемые связями с удаленными БД.
11. Назовите условия, необходимые для удаления связи с удаленной БД.
12. Что такое снимки и для чего они используются?
13. Создание последовательностей в ORACLE.
14. Какие виды последовательностей можно создавать в ORACLE?
15. Что такое синонимы, для чего они используются?
16. Создание синонимов.

Oracle Cloud Computing

Облачные вычисления (Cloud computing) – вычислительная модель, обеспечивающая быстрый, простой и удобный сетевой доступ к пулу вычислительных ресурсов (сеть, сервера, диски, приложения и сервисы) по требованию, причем такой доступ требует минимального привлечения администраторов или сервис провайдеров.

Поддержка целостности данных

Этот раздел объясняет, как ввести в действие организационные правила, ассоциированные с базой данных, и как предотвратить ввод в таблицы некорректной информации, используя ограничения целостности.

Можно определять ограничения целостности для того, чтобы задействовать организационные правила по данным в ваших таблицах. Когда ограничение целостности включено, все данные в таблице должны подчиняться правилу, которое им специфицировано. Если вы после этого выдадите предложение SQL, которое модифицирует данные в этой таблице, то ORACLE проверяет, чтобы результирующие данные удовлетворяли ограничению целостности. Без ограничений целостности, такие правила должны были бы реализовываться программно в вашем приложении.

Реализация правил через ограничения целостности обходится дешевле, чем реализация эквивалентных правил путем выдачи предложений SQL в вашем приложении. Семантика ограничений целостности очень четко определена, так что внутренние операции, которые ORACLE выполняет для их реализации, оптимизированы на более низком уровне, чем уровень предложений SQL. Так как ваши приложения используют SQL, вы не можете достичь такой степени оптимизации.

Реализация организационных правил в самом приложении может обойтись еще дороже в сетевом окружении, потому что предложения SQL должны передаваться через сеть. В таких случаях использование ограничений целостности устраняет накладные расходы, вызываемые этими передачами.

Реализация правил через ограничения целостности обходится дешевле, чем реализация эквивалентных правил путем выдачи предложений SQL в вашем приложении. Семантика ограничений целостности очень четко определена, так что внутренние операции, которые ORACLE выполняет для их реализации, оптимизированы на более низком уровне, чем уровень предложений SQL. Так как ваши приложения используют SQL, вы не можете достичь такой степени оптимизации.

Реализация организационных правил в самом приложении может обойтись еще дороже в сетевом окружении, потому что предложения SQL должны передаваться через сеть. В таких случаях использование ограничений целостности устраняет накладные расходы, вызываемые этими передачами.

Разграничение уровней доступа к данным

Каждый пользователь может использовать много приложений и ассоциированных ролей. Однако вы должны разрешать пользователю иметь в каждый момент лишь те привилегии, которые ассоциированы с ролью текущего приложения, с которым работает этот пользователь. Например, рассмотрим следующий сценарий:

- * Роль ORDER (для приложения ORDER) содержит привилегию UPDATE для таблицы INVENTORY.
- * Роль INVENTORY (для приложения INVENTORY) содержит привилегию SELECT для таблицы INVENTORY.
- * Нескольким клеркам в отделе приема заказов были назначены обе роли, ORDER и INVENTORY.

Следовательно, клерк, которому были назначены обе эти роли, при работе с приложением INVENTORY сможет обновлять таблицу INVENTORY, воспользовавшись привилегией UPDATE, которую он имеет

благодаря роли ORDER. Однако обновление таблицы INVENTORY при работе с приложением INVENTORY не является санкционированной операцией.

Чтобы избежать таких проблем, в начале каждого приложения выдавайте команду SET ROLE, чтобы автоматически включить ассоциированную с этим приложением роль, и, как следствие, выключить все остальные роли. С помощью команды SET ROLE каждое приложение динамически включает конкретные привилегии для пользователя лишь тогда, когда они требуются. Пользователь сможет воспользоваться привилегиями приложения только при работе с этим приложением, и не сможет (преднамеренно или нет) воспользоваться этими привилегиями вне контекста данного приложения.

Предложение SET ROLE упрощает управление привилегиями тем, что, помимо установления контроля над информацией, доступной пользователю, оно позволяет вам контролировать и то, КОГДА пользователь может осуществлять этот доступ. Кроме того, предложение SET ROLE удерживает пользователя в хорошо определенном домене привилегий: если пользователь получает все привилегии через роли, то он не сможет комбинировать эти привилегии, чтобы осуществить несанкционированные операции.

Управление объектами схемы

Этот раздел обсуждает процедуры, необходимые для создания и сопровождения различных типов объектов, содержащихся в схеме пользователя. Темы этой главы включают обсуждение следующих вопросов:

- процедуры для создания, управления и удаления объектов схемы
- переименование объектов схемы
- процедуры для управления использованием памяти в блоках данных
- разрешение имен объектов схемы

Управление таблицами

Таблица – это структура, которая хранит данные в реляционной базе данных. Таблица состоит из строк и столбцов. Таблица может представлять единственную сущность, которую вы хотите отобразить в вашей системе. Такая таблица может представлять, например, список сотрудников вашей организации, или заказы, размещенные на продукты вашей компании. Таблица может также представлять отношение между двумя сущностями. Такая таблица может, например, отображать ассоциацию между сотрудниками и их профессиями, или связи между продуктами и заказами. Внутри таблиц, такие отношения и связи представляются внешними ключами.

Хотя некоторые хорошо определенные таблицы могут одновременно как представлять сущность, так и описывать связь между этой сущностью и другой сущностью, большинство таблиц должны представлять либо только сущность, либо только отношение. Например, таблица EMP описывает сотрудников фирмы, но эта таблица также включает внешний ключ, DEPTNO, который представляет связь от сотрудников к отделам.

Следующие секции объясняют, как создавать, изменять и удалять таблицы. Представлены некоторые простые рекомендации, которым необходимо следовать при управлении таблицами в вашей базе данных; для дополнительных сведений обратитесь к документу ORACLE7 Server Administrator's Guide. Вы должны также прибегать к помощи учебников по проектированию реляционных баз данных и таблиц.

Чтобы удалить ненужную таблицу, используйте команду SQL DROP TABLE. Например, следующее предложение удаляет таблицу EMP:

```
DROP TABLE emp;
```

Если удаляемая таблица содержит первичный или уникальный ключ, на который ссылаются внешние ключи других таблиц, то вы можете одновременно с этой таблицей удалить ограничения FOREIGN KEY для

порожденных таблиц, включив в команду DROP TABLE опцию CASCADE, например:

DROP TABLE emp CASCADE CONSTRAINTS;

Прежде чем удалять таблицу, примите во внимание следующие эффекты этого действия:

- Удаление таблицы приводит к удалению ее определения из словаря данных. Все строки таблицы необратимо теряются.
- Все индексы и триггеры, ассоциированные с таблицей, также удаляются.
- Все обзоры и программные единицы PL/SQL, зависящие от удаляемой таблицы, остаются, но становятся недействительными (непригодными для использования).
- Все синонимы удаленной таблицы остаются, но возвращают ошибку при обращении к ним.
- Все экстенты, распределенные удаляемой некластеризованной таблице, возвращаются в свободную память табличного пространства и могут использоваться любым другим объектом, требующим новых экстенгов.
- Все строки, соответствующие удаляемой кластеризованной таблице, удаляются из блоков кластера.

Если вы хотите удалить все строки таблицы, но сохранить определение этой таблицы, вы должны использовать команду TRUNCATE TABLE. Эта команда описана в документе ORACLE7 ServerAdministrator's Guide.

Привилегии, требуемые для удаления таблиц

Чтобы удалить таблицу, либо она должна содержаться в вашей схеме, либо вы должны иметь системную привилегию DROP ANY TABLE.

Функции для работы со строками.

Для упрощения работы со строками имеется ряд встроенных функций, что значительно облегчает такие операции как преобразование строк к данным других типов, поиск подстроки в строке, определение длины строки и т. д. В данной статье мы рассмотрим самые распространенные функции для работы со строками.

1) Функция определения длины строки `LENGTH(строка)`, возвращает количество символов в строке, включая концевые пробелы.

`SELECT LENGTH('string ') FROM DUAL` вернет значение 7.

2) Функции преобразования регистров символов `UPPER(строка)`, `LOWER(строка)`, `INITCAP(строка)`. Для преобразования символов к верхнему регистру используется функция `UPPER()`.

`SELECT UPPER('string') FROM DUAL` вернет `STRING`.

Если необходимо преобразовать символы строки к нижнему регистру используется функция `LOWER()`.

`SELECT LOWER('STriNG') FROM DUAL`
вернет `string`.

Функция `INITCAP` преобразовывает каждый первый символ слова к верхнему регистру, а все остальные символы к нижнему при условии, что символ-разделитель между словами пробел.

`SELECT INITCAP('string1 string2') FROM DUAL`
вернет строку `String1 String2`.

3) Функции для обрезания начальных и концевых пробелов `LTRIM(строка)`, `RTRIM(строка)`, `TRIM(строка)`. Соответственно первая функция обрезает все начальные пробелы строки, вторая – все концевые, а третья все начальные и концевые.

SELECT LTRIM(' str1 ') FROM DUAL вернет строку str1,
SELECT RTRIM('str2 ') FROM DUAL вернет строку str2,
SELECT TRIM(' str3 ') FROM DUAL вернет строку str3.

4) Функция замены части строки другой строкой REPLACE(исходная_строка, заменяемая_подстрока, заменяющая_подстрока). Для большей ясности рассмотрим пример, в некотором текстовом поле таблицы хранится число. Причем символ-разделитель между целой и дробной частью в некоторых полях «.», а нам для дальнейшей обработки данных нужно, чтобы он во всех полях должен быть «,». Для этого воспользуемся функцией REPLACE следующим образом. REPLACE(field1, '.', ',') и все символы «.» в поле field будут заменены на символ «,».

SELECT REPLACE('My_string','_', '@') FROM DUAL вернет строку My@string.

5) Функции преобразования данных к другим типам данных. TO_CHAR(число) преобразует число в текст. TO_NUMBER(строка) преобразует текст в число. TO_DATE(строка, формат_даты) преобразует строку в дату определенного формата.

SELECT TO_CHAR(123) FROM DUAL вернет строку 123,
SELECT TO_NUMBER('12345') FROM DUAL вернет число 12345,
SELECT TO_DATE('01.01.2010','dd.mon.yyyy') FROM DUAL вернет дату 01.JAN.2010.

6) Функция определения вхождения подстроки в строку INSTR(исходная_строка, подстрока, номер_символа). Даная функция позволяет определять номер символа в исходной строке с которого начинается искомая подстрока (если такая есть). Иначе возвращается 0. Например нам нужно определить все должности в таблице Table1, в наименовании которых встречается подстрока «менеджер». Для этого вполне подойдет следующий оператор

`SELECT * FROM TABLE1 WHERE INSTR(POST, 'менеджер', 1) > 0.`

То есть оператор `SELECT` выведет только те записи из таблицы `TABLE1` где искомая подстрока «менеджер» будет найдена. Причем поиск будет осуществляться с первого символа. Если поиск нужно осуществлять с другой позиции, то номер символа для начала поиска указывается в третьем параметре.

`SELECT INSTR('Small string', 'string', 1) FROM DUAL` вернет значение 7,
`SELECT INSTR('Small string', 'String', 1) FROM DUAL` вернет значение 0.

7) Функция выделения в исходной строке подстроки `SUBSTR(исходная_строка, номер_начального_символа, количество_символов)`. Рассмотрим такой пример, в пользовательской таблице хранится адрес в виде наименование населенного пункта, название улицы, номер дома. Причем мы точно знаем, что для наименования населенного пункта отводится строго 20 символов (если наименовании населенного пункта меньше чем 20 символов, то остальная часть заполняется пробелами), для наименования улицы 30 символов, для номера дома 3 символа. Далее нам необходимо перенести все адреса из нашей таблицы в другую и при этом все 3 компонента адреса должны быть в разных полях. Для выделения компонент адреса применим функцию `SUBSTR()`.

`SELECT SUBSTR(TABLE_1.ADDRESS, 1, 20) CITY, SUBSTR(TABLE_1.ADDRESS, 21, 30) STREET, SUBSTR(TABLE_1.ADDRESS, 52, 3) TOWN FROM TABLE_1`

Конечно для переноса данных необходимо воспользоваться оператором `INSERT`, но для понимания работы функции `SUBSTR` вполне подойдет рассмотренный пример.

`SELECT SUBSTR('My_string', 4, 3) FROM DUAL` вернет строку `str`.

Рассмотренные выше функции можно использовать во входных параметрах. Так если нам нужно выделить все символы, после какого-то определенного, то в функцию `SUBSTR` можно передать номер искомого символа из функции

INSTR. Например если нужно перенести все символы из поля таблицы, которые расположены после «,» то можно использовать такую конструкцию

```
SELECT SUBSTR(My_string, INSTR(My_string, ',', 1), LENGTH(My_string)-  
INSTR(My_string, ',', 1)+1) FROM DUAL.
```

Для определения начального символа мы вызываем функцию INSTR(), которая вернет номер символа первого вхождения подстроки «,». Далее мы определяем количество символов до конца строки как разницу длины строки и номера первого вхождения подстроки.

8) Для определения кода символа используется функция ASCII(строка), которая возвращает код 1 символа строки. Например

```
SELECT ASCII(W) FROM DUAL
```

 вернет значение 87.

9) Обратная функция преобразования кода символа в символ CHR(число).

```
SELECT CHR(87) FROM DUAL
```

 вернет символ W.

Функции для работы с числами.

В СУБД Oracle имеется ряд функций для работы с числами. К ним относятся функции возведение числа в степень POWER(), округление ROUND() и т. д.

1) Функция ABS(число) возвращает абсолютное значение аргумента.

```
SELECT ABS(-3) FROM DUAL
```

 вернет значение 3.

2) Функция CEIL(число) возвращает наименьшее целое, большее или равное переданному параметру.

```
SELECT CEIL(4.5) FROM DUAL
```

 вернет значение 5.

3) Функция FLOOR(число) возвращает наибольшее целое, меньшее или равное переданному параметру.

```
SELECT FLOOR(3.8) FROM DUAL
```

 вернет значение 3.

4) Функция MOD(число_1, число_2) возвращает остаток от деления первого параметра на второй.

SELECT MOD(5, 3) FROM DUAL вернет значение 2. Примечание. Если второй параметр равен 0, то функция возвращает первый параметр.

5) Функция округления ROUND(число_1, число_2). Округляет первый переданный параметр до количества разрядов, переданного во втором параметре. Если второй параметр не указан, то он принимается равным 0, то есть округление производится до целого значения.

Примеры

SELECT ROUND(101.34) FROM DUAL вернет значение 101,

SELECT ROUND(100.1268, 2) FROM DUAL вернет значение 100.13

SELECT ROUND(1234000.3254, -2) FROM DUAL вернет значение 1234000,

SELECT ROUND(-100.122, 2) FROM DUAL вернет значение -100.12.

6) Функция усечения значения TRUNC(число_1, число_2). Возвращает усеченное значение первого параметра до количества десятичных разрядов, указанного во втором параметре. Примеры

SELECT TRUNC(150.58) FROM DUAL вернет значение 150

SELECT TRUNC(235.4587, 2) FROM DUAL вернет значение 235.45

SELECT TRUNC(101.23, -1) FROM DUAL вернет значение 100

7) В СУБД Oracle имеется ряд тригонометрических функций SIN(число), COS(число), TAN(число) и обратные им ACOS(число), ASIN(число), ATAN(число). Они возвращают значение соответствующей названию тригонометрической функции. Для прямых функции параметром является значение угла в радианах, а для обратных – значение функции. Примеры

SELECT COS(0.5) FROM DUAL вернет значение 0.877582561890373

SELECT SIN(0.5) FROM DUAL вернет значение 0.479425538604203

SELECT TAN(0.5) FROM DUAL вернет значение 0.546302489843791

SELECT ACOS(0.5) FROM DUAL вернет значение 1.0471975511966

SELECT ASIN(0.5) FROM DUAL вернет значение 0.523598775598299
SELECT ATAN(0.5) FROM DUAL вернет значение 0.463647609000806

8) Гиперболические функции. SINH(число), COSH(число), TANH(число). SINH() возвращает гиперболический синус переданного параметра, COSH() возвращает гиперболический косинус переданного параметра, TANH() возвращает гиперболический тангенс переданного параметра. Примеры

SELECT COSH(0.5) FROM DUAL вернет значение 1.12762596520638
SELECT SINH(0.5) FROM DUAL вернет значение 0.521095305493747
SELECT TANH(0.5) FROM DUAL вернет значение 0.46211715726001

9) Функция возведения в степень POWER(число_1, число_2). Примеры

SELECT POWER(10, 2) FROM DUAL вернет значение 100
SELECT POWER(100, -2) FROM DUAL вернет значение 0.0001

10) Логарифмические функции. LN(число) возвращает натуральный логарифм переданного параметра, LOG(число_1, число_2) возвращает логарифм второго переданного параметра по основанию, переданному первому параметру. Причем первый параметр должен быть больше нуля и не равен 1. Примеры

SELECT LN(5) FROM DUAL вернет значение 1.6094379124341
SELECT LOG(10, 3) FROM DUAL вернет значение 0.477121254719662

11) Функция извлечения квадратного корня SQRT(число). Пример

SELECT SQRT(4) FROM DUAL вернет значение 2.

12) Функция возведение числа e в степень EXP(число). Пример

SELECT EXP(2) FROM DUAL вернет значение 7.38905609893065.

Функции для работы с датами.

На практике очень часто необходимо анализировать данные в виде дат, производить некоторые операции над ними, изменять формат. Все эти

операции уже реализованы в виде встроенных функций. Рассмотрим самые основные из них.

1) `ADD_MONTHS(дата, количество_месяцев)` возвращает дату, отстоящую от даты, переданной в первом параметре на количество месяцев, указанном во втором параметре. Примеры

`SELECT ADD_MONTHS('01-JAN-2010', 2) FROM DUAL` вернет дату '01.03.2010'

`SELECT ADD_MONTHS('01-JAN-2010', -3) FROM DUAL` вернет дату '01.10.2009'

`SELECT ADD_MONTHS('30-JAN-2010', 1) FROM DUAL` вернет дату '28.02.2010'

2) Для определения текущей даты и времени применяется функция `SYSDATE`. Область применения данной функции намного шире чем может показаться на первый взгляд. В первую очередь это контроль за вводом данных в БД. Во многих таблицах выделяется отдельное поле для сохранения даты последнего внесения изменений. Также очень удобно контролировать некие входные параметры для отчетов, особенно если они не должны быть больше чем текущая дата. Помимо даты данная функция возвращает еще и время с точностью до секунд. Пример

`SELECT SYSDATE FROM DUAL` вернет дату '22.05.2010 14:51:20'

3) Если необходимо определить последний день месяца, то для этого вполне подойдет функции `LAST_DAY(дата)`. Её можно использовать для определения количества дней, оставшихся в месяце.

`SELECT LAST_DAY(SYSDATE) – SYSDATE FROM DUAL.`

В результате выполнения данного оператора будет выведено количество дней от текущей даты до конца месяца. Пример

`SELECT LAST_DAY('15-FEB-2010') FROM DUAL` вернет дату '28.02.2010'.

4) Функция для определения количества месяцев между датами MONTHS_BETWEEN(дата_1, дата_2).

Примеры

```
SELECT MONTHS_BETWEEN('01-JUL-2009', '01-JAN-2010') FROM DUAL
```

вернет значение -6

```
SELECT MONTHS_BETWEEN('01-JUL-2009', '10-JAN-2010') FROM DUAL
```

вернет значение -6.29032258064516.

Примечание. Если дни месяцев совпадают, то функция возвращает целое число, в противном случае результат будет дробным, причем количество дней в месяце будет принято 31.

5) Функция NEXT_DAY(дата, день_недели) позволяет определить следующую дату от даты, переданной в первом параметре, которая соответствует дню недели, переданном во втором параметре. Пример

```
SELECT NEXT_DAY('01-JUL-2009', 'mon') FROM DUAL
```

вернет дату '06.07.2009', то есть следующий понедельник после 1 июля 2009 наступил 6 числа.

Значение параметра День недели

mon	Понедельник
tue	Вторник
wed	Среда
thu	Четверг
fri	Пятница
sat	Суббота
sun	воскресенье

6) Округление даты ROUND(дата, формат). Второй параметр не обязателен, если его не указывать, то он принимается за 'DD', то есть округление будет произведено до ближайшего дня.

Примеры

SELECT ROUND(SYSDATE) FROM DUAL вернет дату '23.05.2010'

SELECT ROUND(SYSDATE, MONTH) FROM DUAL вернет дату '01.06.2010', округляется до ближайшего первого дня месяца.

Формат	Единица округления
CC, SCC	Век
YYYYY, YYYY, YEAR	Год
Q	Квартал
MM, MONTH	Месяц
WW	Тот же день недели, что и первый день года
W	Тот же день недели, что и первый день месяца
DD, J	День
Day, DY	Первый день недели
HH, HH12, HH24	Час
MI	Минута

7) Усечение даты. Функция TRUNC(дата, формат). Также как и рассмотренная выше может не иметь второго параметра. В таком случае усечение будет производиться до ближайшего дня. Примеры

SELECT TRUNC(SYSDATE) FROM DUAL вернет дату '22.05.2010'

SELECT TRUNC(SYSDATE, 'WW') FROM DUAL вернет дату '01.05.2010'

SELECT TRUNC(SYSDATE, 'Day') FROM DUAL вернет дату '16.05.2010'.

Функции преобразования данных.

Данный раздел посвящен рассмотрению преобразования данных в различные форматы. На практике довольно распространены ситуации, когда необходимо строковые величины рассматривать как числа и наоборот. Несмотря на небольшое количество функции их возможностей вполне хватает для решения очень сложных прикладных задач.

1) TO_CHAR(данные, формат). На первый взгляд синтаксис довольно прост, но за счет второго параметра можно очень точно описать в какой формат преобразовать данные. Итак в строку можно преобразовать как дату, так и числовое значение. Рассмотрим вариант преобразования даты к строке. Значения самых распространенных форматов приведены в таблице, более полная информация содержится в технической документации.

Формат	Описание формата
D	День недели
DD	День месяца
DDD	День года
MM	Номер месяца
MON	Сокращенное название месяца
MONTH	Полное название месяца
Q	Квартал
YY, YYY, YYYY	Год
HH, HH12, HH24	Час
MI	Минут
SS	Секунда

Таблица значений форматов для преобразования числа в строку.

Формат	Описание формата
99D9	Указание позиции разделителя десятичной точки. Число девяток соответствует максимальному количеству цифр
999G99	Указание позиции группового разделителя
99,999	Возвращает запятую в указанной позиции
99.999	Возвращает точку в указанной позиции
99V9999	Возвращает значение умноженное на 10 в степени n, где n число девяток после V.

0999	Возвращает ведущие нули, а не пробелы
9990	Возвращает конечные нули, а не пробелы
9.99EEEE	Возвращает число в экспоненциальной форме
RM	Возвращает число в римской системе исчисления

Примеры

SELECT TO_CHAR(SYSDATE, 'D-MONTH-YY') FROM DUAL вернет строку '7-MAY -10'

SELECT TO_CHAR(SYSDATE, 'DDD-MM-YYYY') FROM DUAL вернет строку '142-05-2010'

SELECT TO_CHAR(SYSDATE, 'Q-D-MM-YYY') FROM DUAL вернет строку '2-7-05-010'

SELECT TO_CHAR(1050, '9.99EEEE') FROM DUAL вернет строку ' 1.050E+03'

SELECT TO_CHAR(1400, '9999V999') FROM DUAL вернет строку '1400000'

SELECT TO_CHAR(48, 'RM') FROM DUAL вернет строку ' XLVIII'

2) Функция преобразования строки в дату TO_DATE(строка, формат). Возможные значения форматов уже рассмотрены выше, поэтому приведу несколько примеров использования данной функции. Примеры

SELECT TO_DATE('01.01.2010', 'DD.MM.YYYY') FROM DUAL вернет дату '01.01.2010'

SELECT TO_DATE('01.JAN.2010', 'DD.MON.YYYY') FROM DUAL вернет дату '01.01.2009'

SELECT TO_DATE('15-01-10', 'DD-MM-YY') FROM DUAL вернет дату '15.01.2010'.

3) Функция преобразования строки в числовое значение TO_NUMBER(строка, формат). Самые распространенные значения форматов перечислены в таблице, поэтому рассмотрим применение данной функции на примерах. Примеры

SELECT TO_NUMBER('100') FROM DUAL вернет число 100

SELECT TO_NUMBER('0010.01', '9999D99') FROM DUAL вернет число 10.01
SELECT TO_NUMBER('500,000','999G999') FROM DUAL вернет число 500000.

Практическое занятие № 5

Цель работы: Создание формы ввода данных

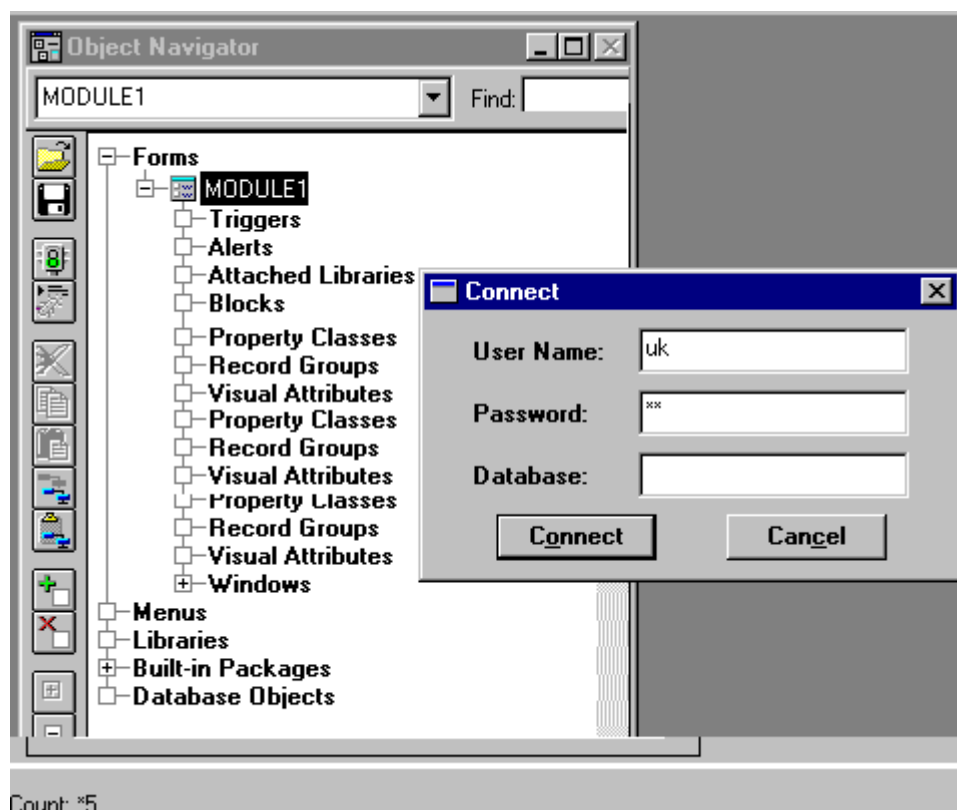
Пример выполнения работы.

Создадим форму, в которой можно было бы вводить названия должности и соответствующий ей оклад.

Подсоединитесь к БД, используя меню **File / Connect**.

В результате соединения с БД становятся доступными все объекты подсоединенной БД.

Создайте новый модуль формы, используя меню **File / New / Form**.



Дерево подчиненных объектов новому модулю формы **MODULE1** содержит только один объект - Окно **WINDOW0** - так как форма не может существовать без окна.

Используя меню **Tools \ New Block**, создайте новый базовый блок связанный с таблицей **DOL**, пусть этот блок также называется **DOL** и принадлежит канве **CANVAS1**:

MODULE1: New Block Options

General | Items | Layout | Master/Detail

Base Table:

Block Name:

Canvas:

Sequence ID:

На закладке **Items** нажмите кнопку **Select Columns** для отображения полей таблицы.

Выделите курсором поле **Name_D** и определите заголовок этого поля **Label** как *Должность*; Для поля **OKLAD** как *Оклад* :

MODULE1: New Block Options

General | Items | Layout | Master/Detail

+ DOL_ID
+ **NAME_D**
+ OKLAD

Item Options:

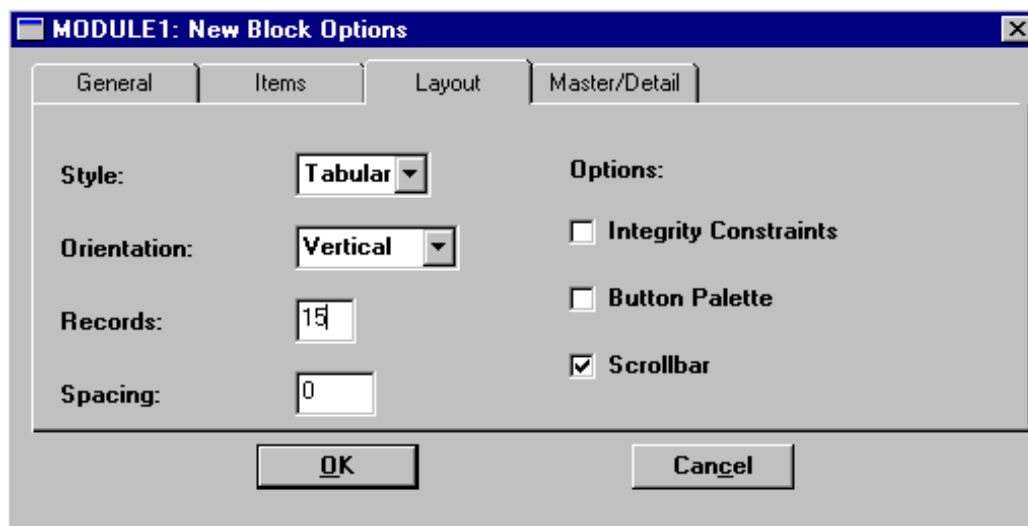
Label:

Width:

Type: ▼

☒ Include

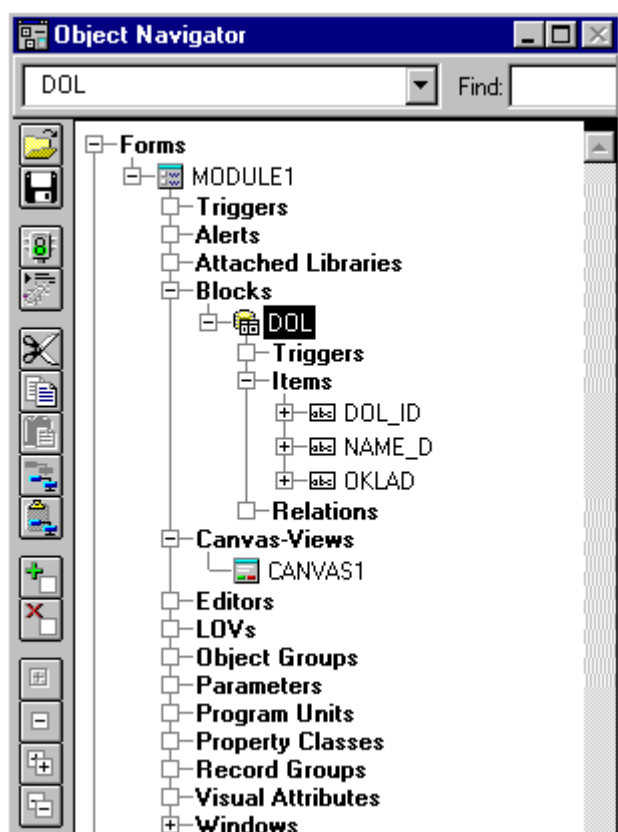
На закладке **Layout** определим внешний вид блока : Наличие полосы прокрутки - **ScrollBar** , Количество строк в блоке Records установим **15** , Ориентация блока - вертикальная, Стил - в виде таблицы: **Tabular**. После чего нажмите кнопку **OK**



В результате в дереве объектов модуля формы появились новые объекты:

В узле **Blocks** - блок с именем **DOL**

В узле **Canvas-Views** - канва с именем **CANVAS1**



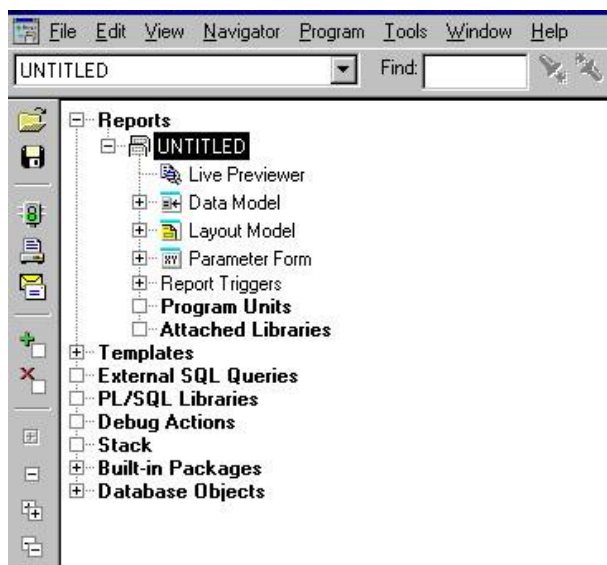
Посмотрим внешний вид Формы , расположение на ней полей блока **DOL**

Для этого вызовем редактор канвы (двойным кликом мыши на иконке **CANVAS1** , или через меню **Tools / Layout Editor**)

1. Вызовите диалоговое окно, в котором предлагается сделать выбор – использовать для построения отчета мастер, либо строить новый отчет вручную. Выбираем последнее:



При этом в окне Дизайнера открывается Объектный Навигатор. Объектный Навигатор показывает существующие объекты и типы объектов, используется для вызова таблиц атрибутов объектов, а также для создания и переименования объектов.



2. Соединение с базой данных

Для соединения с базой данных выберите в меню File -> Connect.

Появится диалоговое окно Connect.

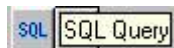
3. Определение данных

Определение данных делается в редакторе модели данных, который можно вызвать из Объектного Навигатора.



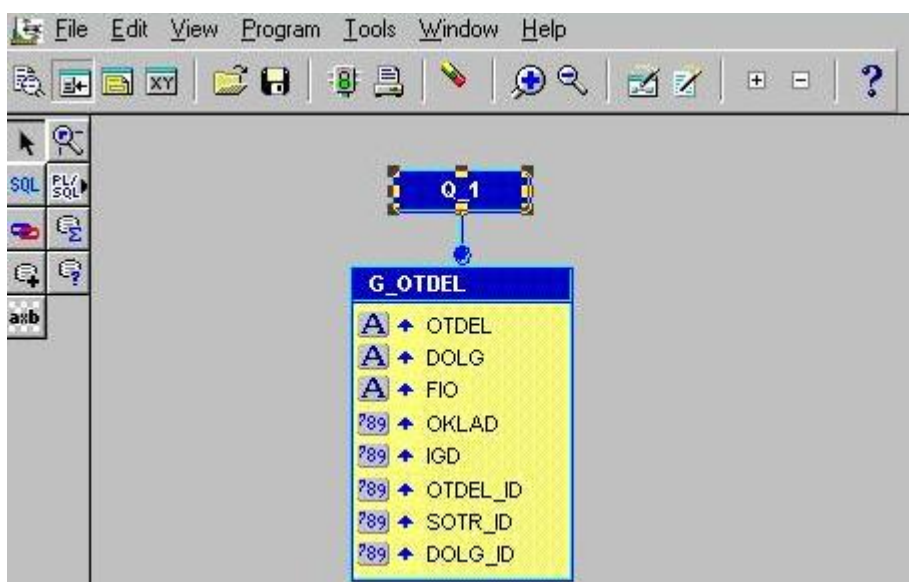
Дважды щелкните на пиктограмме узла модели данных. Появится редактор модели данных.

Выберите инструмент SQL Query, щелкнув на нем в палитре инструментов.



Перейдите в рабочую область редактора модели данных и щелкните один раз.

Появится запрос Q_1.



Двойной щелчок на запросе вызывает окно для ввода и редактирования запроса.

Запрос можно ввести в окне SQL Query Statement,

импортировать из файла – закладка Import SQL Query и

построить с помощью специального построителя запросов – закладка

Query Builder.

Оформление отчета

В качестве оформления отчета можно добавить красочный заголовок, нумерацию страниц, текущую дату, колонтитул, создать рисунки с помощью инструментов рисования и т.п.

Добавление выделенного заголовка



Для этого перейдем в редактор Layout и выберем режим редактирования окружения отчета - кнопка Margin на панели инструментов.



В верхней части отчета создайте текстовое поле(кнопка Text на левой панели) с содержанием «Отчет о личном составе по отделам», используя палитру красок раскрасьте его по своему вкусу.

Создайте статический объект - прямоугольник с скругленными краями.

Разместите его за статическим текстом, используя меню

Arrange \ Send to Back.

Добавление текущей даты и нумерация страниц

Для этого на инструментальной панели редактора макета существуют специальные инструменты:



При этом вы выбираете местоположение и формат из списка доступных, формат можно также задать собственный.

Для добавления даты и нумерации страниц можно также воспользоваться инструментом Field:



Источником данных для нового поля в макете могут быть все данные из модели данных — колонки, вычисляемые столбцы, системные и пользовательские параметры, а также так называемые системные переменные:

- Current Date - текущая дата
- Page Number - номер логической страницы
- Panel Number - номер панели на странице

Physical Pages - номер физической страницы
 Total Pages - общее число логических страниц
 Total Panels - общее число панелей
 Total Physical Pages - общее число физических страниц

Вид отчета после оформления:

Отчет о личном составе				
10/08/2000 13:24:06			Лист 1 из 1	
Отдел	ФИО	Должность	Оклад	В долларах
Отдел 1	Федоров	Программист	5000	200
	Максимов	Начальник отдела	6000	240
	Кириллов	Оператор	3000	120
	Итого по отделу		14000	
Отдел 2	Иванов	Начальник отдела	2000	80
	Петров	Программист	1000	40
	Сидоров	Менеджер	1000	40
	Носов	Программист	1500	60
	Косов	Программист	1800	72
	Пупкин	Программист	1500	60
	Чубайс	Менеджер	1000	40
	Итого по отделу		9800	
Всего			23800	

Практическое занятие № 7

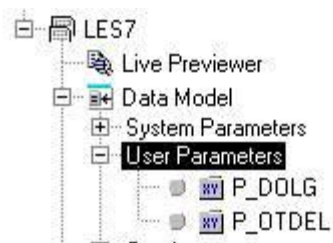
Цель работы: Отчет с параметрами

Пример выполнения работы.

Создайте простой отчет с одним запросом

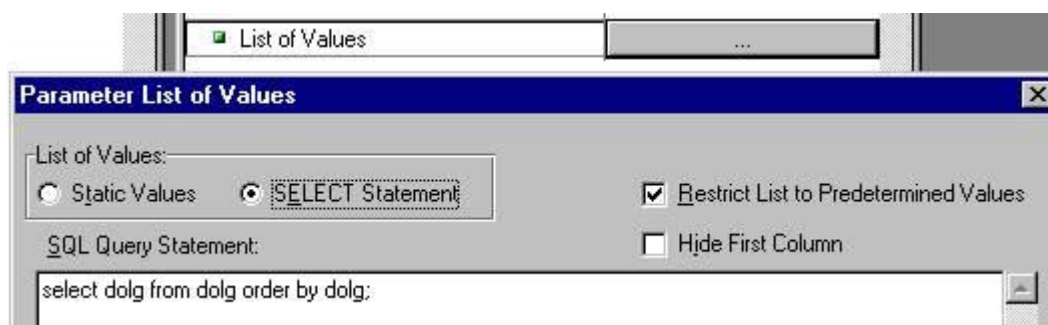
```
SQL Query Statement:
select fio from spisok
where otdel = :P_otdel and dolg = :P_dolg;
```

Созданные параметры в окне Объектного навигатора:

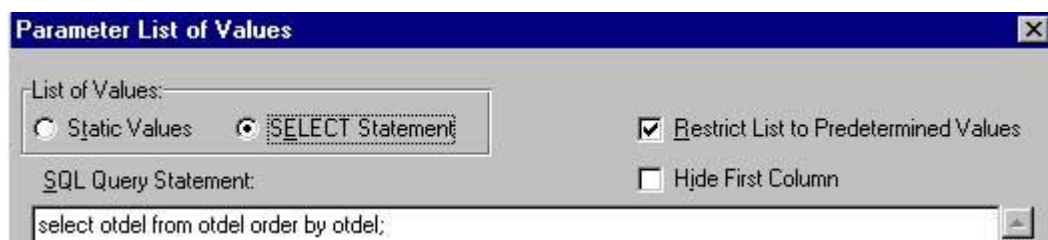


В таблице атрибутов параметра можно указать, что для параметра должен выводиться список значений в форме параметров.

Определение списка значений для параметра P_Dolg:



Определение списка значений для параметра P_Otdel:



Для создания формы параметров используйте Tools->Parameter Form Builder. Измените заголовок, исключите системные параметры, по своему вкусу выберите шрифт и цвет для названия формы и заголовков параметров.

Parameter Form Builder

Title: Форма параметров

Hint Line: Введите значения следующих

Status Line:

Parameter:	Label:
ORIENTATION	Orientation
BACKGROUND	Run in Background
MODE	Output Mode
PRINTJOB	Show Print Job Dialog
P_OTDEL	Отдел
P_DOLG	Должность

OK Cancel Help

les7: Report Editor - Parameter Form

MS Sans Serif 12 B I U

0 1 2 3 4

0 1

Форма параметров

Введите значения следующих параметров:

Отдел PF_P_OTDEL

Должность PF_P_DOLG

Контрольные вопросы.

- Что такое Oracle Cloud Computing?
- В чем разница между правилами через ограничения целостности и эквивалентными правилами?
- Какова организация структуры данных Oracle Cloud Computing?
- Перечислите типы данных, поддерживаемых в Oracle Cloud Computing.
- Какие существуют элементы управления?
- Как создать соединение в Oracle?
- Как импортировать базу из текстового файла?
- Как создать графическое отображение базы данных?
- Как организовать отображение статистики данных?
- Как создать таблицу?
- Как организовать создание параметрического отчета?

Oracle jDeveloper. Начальные сведения

В конце 2001 года корпорация Oracle выпустила версию Oracle Java Developer 9i. Данный продукт написан полностью на Java. Это дает возможность запускать среду разработки на любом компьютере, где может быть установлен Java SDK. Построен jDeveloper на основе технологий JBuilder некогда купленных у фирмы Borland. За несколько лет на основе этих технологий было выпущено несколько версий JDeveloper, который перерос в самостоятельный продукт, по многим параметрам превосходящий родителя. Сейчас на рынке доступна Oracle jDeveloper 11g, о которой и пойдет речь в этой книге.

В JDeveloper'e используется аналогичная JBuilder'у структура проектов и принципы работы среды визуального проектирования, что позволяет с минимальными затратами адаптировать проекты, созданные в Borland Java Builder. Правда сам файл проекта изменился, но создание нового занимает не более минуты, при использовании интерактивного мастера создания проектов. Oracle Java Developer 10g является полноценной средой для разработки приложений, с использованием новейших Internet стандартов. Бытующее мнение, что Oracle JDeveloper предназначен для работы с СУБД Oracle в корне неверно. Хотя в данной IDE и введена оптимизация на использование СУБД Oracle (Oracle 9i и Oracle 9i Lite), есть возможность работы с любой СУБД, поддерживающей стандарт SQL92. А использование стандартных методов работы с JDBC снимает все ограничения. К тому же достаточно просто добавить свои бины в палитру компонентов, и расширять IDE по своему усмотрению. В новой версии 11g существенно переработан интерфейс пользователя. Многооконный MDI интерфейс был заменен на Tabbed control, в закладки панелей которого добавлены динамически формируемые кнопки закрытия закладки при наведении указателя мыши, за счет чего освободилась часть рабочего пространства. Ранее не связанные визуальный редактор, редактор класса и редактор кода теперь объединены. Переключение между ними осуществляется выбором соответствующей закладки. Ориентирование на

J2EE технологии положительно сказалось на функциональности встроенного HTML редактора. Доступен так же визуальный дизайнер HTML. Так, что данную среду можно использовать, как достаточно мощный редактор Web контекста. Поддерживается большое количество типов файлов. Особенно хочется отметить поддержку JSP и XML, который используется здесь повсеместно. Поддерживаемые типы документов парсятся в дерево структуры документов, которое позволяет осуществлять быструю навигацию по документу, а так же контролировать ошибки.

jDeveloper- это мощнейшая интегрированная среда разработки. Конечно, изучение этой программы требует времени. Однако, оно того стоит. Чтобы доказать вам это, ниже приводится краткий свод «за» использование jDeveloper.

- Ускорение процесса разработки за счет использования средств моделирования, интегрированной среды визуальной разработки, мощного отладчика и встроенных средств оптимизации приложения;
- Уменьшение затрат на средства разработки за счет интеграции в одной среде разработки модулей, позволяющих вести полный цикл разработки от проектирования и макетирования до реализации и отладки разрабатываемых приложений. Данная интеграция упрощает процесс создания приложений, так как исключается процесс синхронизации результатов работы в продуктах разных разработчиков;
- Наличие большого количества мастеров, упрощает выполнение рутинных операций и позволяет создавать готовые модули на основе шаблонов;
- Увеличение производительности и качества приложений за счет использования встроенного профайлера, анализатора кода и контроля ошибок без компилирования модулей. JDeveloper включает в себя профайлеры выполнения кода, событий и использования памяти;
- Встроенный отладчик позволяет вести отладку нескольких процессов, удаленную отладку, просмотр загруженных классов, стека вызовов и значений

экземпляров объектов. При этом отслеживается область видимости данных объектов. Хочется отметить, что процесс отладки организован на очень высоком уровне и позволяет полностью контролировать практически все аспекты при выполнении программ;

- Построитель плана выполнения SQL запросов помогает оптимизировать SQL запросы, за счет чего можно иногда ускорить выполнение критических запросов в несколько раз;
- Для упрощения процесса анализа качества кода используется утилита CodeCoach, которая сканирует код приложения во время выполнения, и формирует набор рекомендаций по увеличению производительности и снижению затрат системных ресурсов;
- Упрощение процесса формирования пакетов развертывания (deploy) проектов, за счет большого количества шаблонов развертывания и мастеров, упрощающих создание сценария пакета развертывания;
- Ориентация на разработку приложений по приобретающей в последнее время широкое распространение и признание J2EE технологии;
- Встроенный Oracle9i Application Server позволяет оперативно тестировать, отлаживать и настраивать J2EE приложения и Web сервисы, прямо из среды разработки. Одним кликом мышки, отлаженное приложение можно развернуть в WAR архив, или на внешний J2EE – сертифицированный сервер. В JDeveloper включены шаблоны развертывания в Oracle9i Application Server, BEA WebLogic, JBoss, IBM WebSphere и другие J2EE – сертифицированные серверы;
- В JDeveloper включены средства для групповой работы над проектом. Есть возможность использования единого репозитория проектов;
- Снижены риски разработки за счет использования промышленных стандартов. Oracle JDeveloper 11g направлен на использование стандартов Java, XML и Web сервисов. Он занимает лидирующие позиции за

счет поддержки последних стандартов J2EE, J2SE и J2ME. Поддержка XML стандарта включает DOM, SAX, XML схемы, JAXP и XSL. Поддержка стандарта Web сервисов включает SOAP, WSDL и UDDI. Поддерживаются также другие стандарты, такие как UML, XMI, WebDAV и SQL;

- JDeveloper поддерживает средства контроля версий такие как Oracle9i Software Configuration Manager (SCM), Rational ClearCase, и Concurrent Versions System (CVS). Для этого просто конфигурируются параметры среды JDeveloper для использования выбранной системы контроля версий. Есть возможность создания собственного плагина для подключения любой другой системы контроля версий;
- Встроенная среда моделирования с использованием унифицированного языка моделирования Unified Modeling Language UML упрощает процесс проектирования приложений. В новой версии JDeveloper'a значительно увеличен состав UML диаграмм. Наряду с базовыми диаграммами введены шаблоны для расширения UML, позволяющие моделировать структуру базы данных, EJB, Business компоненты и Web сервисы;
- Большинство UML моделей позволяют автоматически генерировать Java код и другие объекты приложения. Возможен обратный процесс реинженеринга Java кода в UML модель. Связанные с кодом проекта UML модели, автоматически синхронизируются при любых изменениях кода;
- Есть возможность расширять функциональность IDE за счет интеграции модулей сторонних разработчиков (плагины).
- На сайте <http://otn.oracle.com/products/jdev/htdocs/partners/addins/exchange/index.html> можно найти большое количество модулей расширения;
- Для снижения трудоемкости создания J2EE приложений был разработан Oracle Application Development Framework (ADF). Oracle ADF делает разработку J2EE приложений доступной более широкому сообществу

программистов. Базирующийся на модели разработки Model-View-Controller (MVC), Oracle ADF позволяет разработчикам сконцентрироваться на реализации проекта. Используя технику визуального декларативного программирования, ADF позволяет разработчикам быстро создавать готовые решения, без необходимости детального изучения J2EE технологии.

Java и Oracle JDeveloper

Начиная с версии 8.1 в состав СУБД Oracle можно дополнительно включать так называемый JServer, позволяющий использовать для хранимых процедур помимо PL/SQL еще и язык Java. В состав JServer входят следующие элементы: виртуальная Java-машина JVM под названием Auroга, поддерживающая среду для выполнения Java-программ и библиотеки классов Java средства увязки с PL/SQL ряд других

JVM Auroга способна исполнять методы Java ("хранимые Java-процедуры") и классы, хранимые в Oracle.

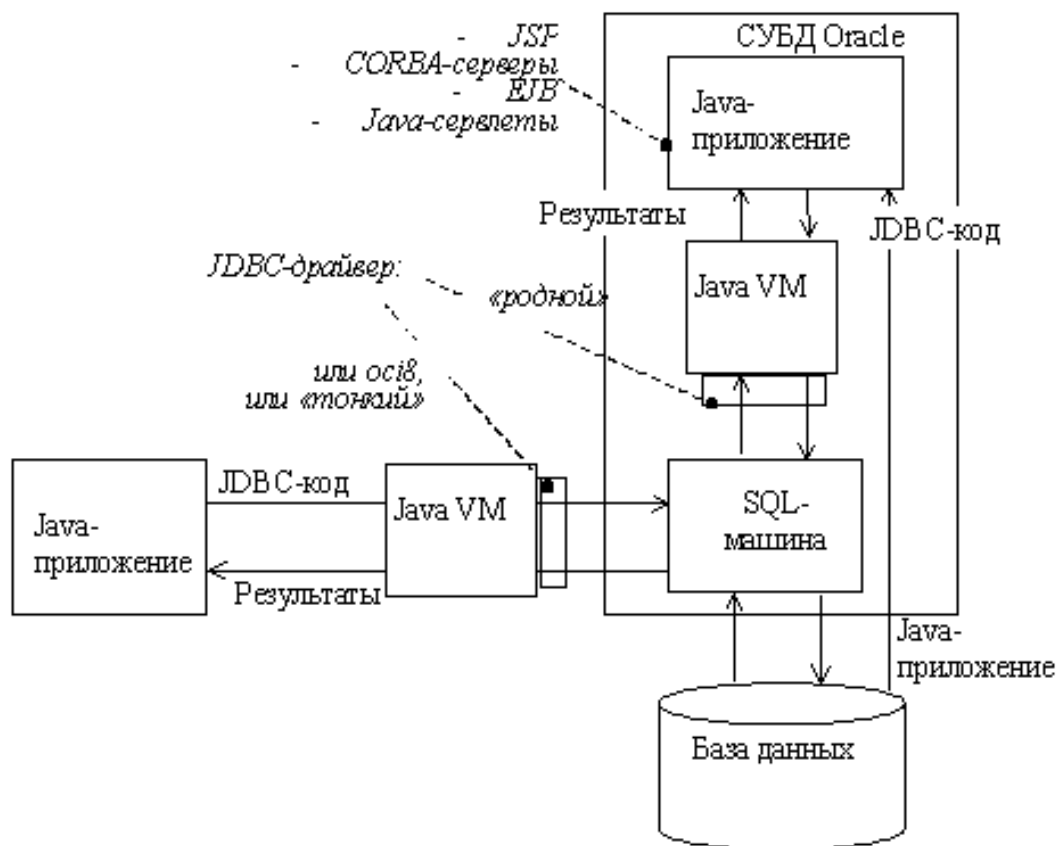
Место Java в архитектуре Oracle

Наличие встроенной виртуальной машины Java в Oracle -дополнительная возможность по отношению к базовой поставке.

Начиная с версии 9.2 встроенная в СУБД виртуальная машина Java не поддерживает магазины протоколов J2EE и CORBA. В частности, с этого времени нет возможности разместить в БД под управлением Oracle контейнеры для EJB, JSP и сервлетов (OSE), как это было раньше. Их предлагается размещать в среде OC4J из состава Oracle 11G Application Server. Встроенная машина Java продолжает поддерживать только работу хранимых процедур, JDBC и SQLJ.

Соотношение и взаимосвязь PL/SQL и Java в Oracle

Java в Oracle представляет собой полнофункциональную замкнутую



систему, однако классы Java средствами Oracle можно "публиковать" для PL/SQL-машины и вызывать из программ на PL/SQL.

Вплоть до версии 11G включительно PL/SQL в Oracle несравнимо эффективнее обрабатывает SQL-запросы. С другой стороны Java обладает более богатой и универсальной языковой средой для описания приложений.

Особенности Java и среда работы программ на Java

Архитектура и принципы работы Java резко отличаются от архитектуры и принципов работы PL/SQL. Ниже излагаются некоторые особенности Java, существенные для использования этого языка при работе с Oracle.

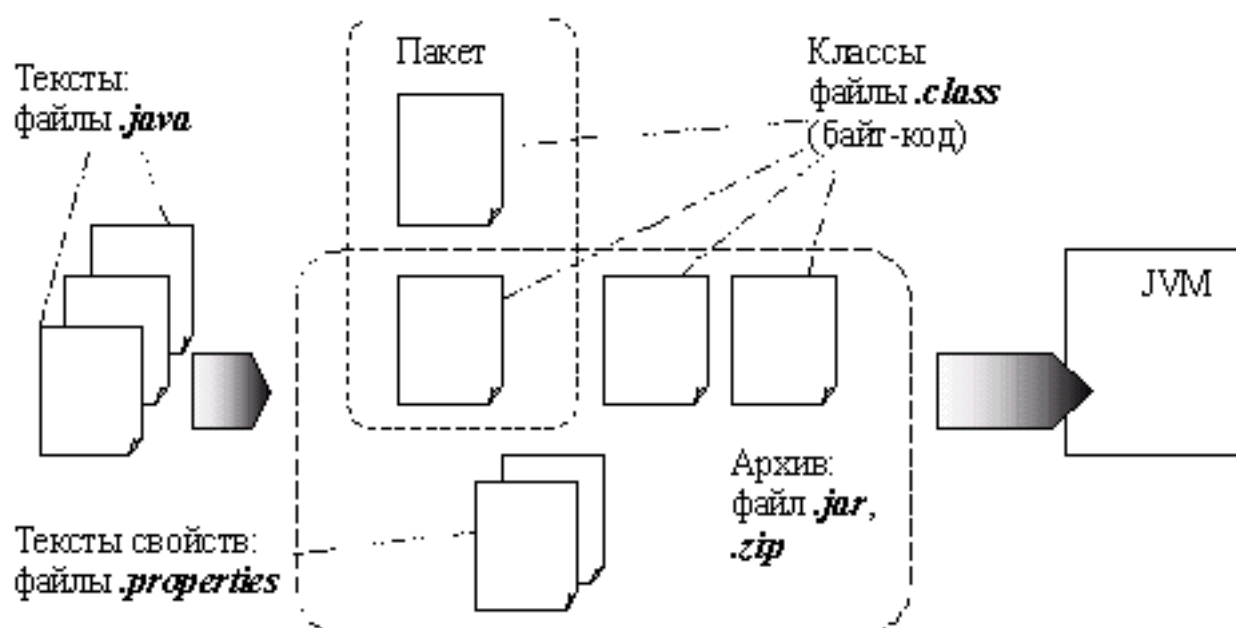
Программные компоненты в среде разработки на Java

Основными программными компонентами в среде разработки на Java являются исходный код, класс, пакет, интерфейс, файл ресурсов.

Взаимоотношение показано на рисунке.

Пакет используется для логической группировки программных единиц Java.

Архив используется для физической группировки программных единиц Java, необходимых для работы конкретной Java-программы, могущих быть вызванных прямо или по цепочке. Технологически часто единственная альтернатива неимоверному числу .class-файлов.



Установка среды разработки на Java

Для ведения разработок с использованием Java необходимо установить на компьютере JDK (Java Development Kit, прежнее название - SDK, Software Development Kit for Java).

Начиная с версии Oracle 8.1 JDK присутствует на CD с основной поставкой и может устанавливаться штатной программой Oracle Installer путем специального указания. В типовых вариантах установки программной среды Oracle (например, в вариантах Typical или Minimal в версии 8.1) JDK появляется на компьютере автоматически.

JDK можно установить и независимо от Oracle, переписав этот программный комплект с <http://www.javasoft.com/products/>.

Среда окружения ОС

Для работы программ среды разработки Java должны быть выставлены следующие минимально необходимые переменные среды окружения ОС:

CLASSPATH. Переменная, которая указывает на местонахождение файлов с классами, необходимыми для трансляции или выполнения java-программы. Местонахождением может быть (а) каталог файловой системы, в котором расположены файлы с классами и (б) zip- или jar-файл с теми же файлами, упакованными внутрь. Путь к файлу с классом должен быть согласован с полным именем класса, включающим имя пакета. Если имя пакета не используется, в CLASSPATH следует включить "." (указание на текущий каталог). (Строго говоря, для работы программ java и javac переменную CLASSPATH можно и не выставить, но тогда эти программы обязаны использовать ключ -classpath, иначе не обязательный.)

PATH. Сюда нужно включить доступ к программам среды разработки.

Исполняемые модули из состава JDK в версии 8.1 расположены в %ORACLE_HOME%\apache\jdk\bin, а в версии 9 и выше - в %ORACLE_HOME%\jdk\bin.

Основные библиотеки классов classes111.zip и classes12.zip (разница между ними - в версиях Java) в обеих версиях Oracle находятся в %ORACLE_HOME%\jdbc\lib

Для проведения экспериментов удобно создать командный файл со следующим текстом для версии Oracle 8.1:

```
set nls_lang=american_america.ru8pc866
set oracle_home=c:\oracle\ora81
set classpath=%oracle_home%\jdbc\lib\classes111.zip;.
path=%path%;%oracle_home%\apache\jdk\bin;%oracle_home%\lib
```

или со следующим текстом для версии Oracle 9.2 и выше:

```
set nls_lang=american_america.ru8pc866
set oracle_home=c:\oracle\ora92
set classpath=%oracle_home%\jdbc\lib\classes111.zip;.
path=%path%;%oracle_home%\jdk\bin
```

Теперь можно открыть консольное окошко и прогнать нужный командный файл.

Создание самостоятельных программ на Java

Пример транслирования и выполнения Java-программы

Файл с программой под названием MyJavaAgent.java может иметь следующее содержание:

```
public class MyJavaAgent {
    public static String sayHello (String toWhom) {
        return "Hello, " + toWhom + "!";
    }
    public static void main (String[] args) {
        System.out.println(sayHello ("World"));
    }
}
```

Транслирование программы (класса):

```
javac MyJavaAgent.java
```

Запуск программы (класса):

```
java MyJavaAgent
```

Создание хранимых программ на Java в Oracle

Oracle позволяет хранить Java-программы и вызывать их на исполнение с помощью встроенной JVM, полностью наподобие хранимым PL/SQL-процедурам, исполняемым встроенной PL/SQL-машиной.

Дополнительные компоненты СУБД Oracle для работы с хранимыми Java-программами

Для работы с хранимыми Java-программами посредством Jserver/OJVM в Oracle добавлены следующие компоненты разного характера:

Компонента	Описание
JVM Aurora/Oracle JVM	Java Virtual Machine, выполняющая хранимый Java-код
loadjava	Программа, вызываемая из операционной системы для загрузки в БД Java-элементов из файлов .java, .class, .properties, .jar, .zip, .sqlj
dropjava	Программа, вызываемая из операционной системы для удаления из БД ранее загруженных Java-элементов

Компонента	Описание
CREATE JAVA SYSTEM	Создает в БД структуры для работы Java; аналогична SQL-предложению заведения БД CREATE DATABASE ...
{CREATE ALTER DROP}	JAVA ... SQL-предложения категории DDL, во многом дублирующие функции программ loadjava и dropjava
Модификации в CREATE PROCEDURE/FUNCTION ...	Позволяют предъявлять хранимые Java-программы в зону видимости PL/SQL-программ
JAVA_POOL_SIZE JAVA_MAX_SESSIONSPACE_SIZE JAVA_SOFT_SESSIONSPACE_LIMIT	INIT-параметры, регулирующие использование памяти Java-программами в Oracle
JAVASYSPRIV JAVAUSERPRIV	Роли, которые дают возможность хранимым программам взаимодействовать с операционной системой (например, читать из файла)
DBMS_JAVA	Системный пакет с процедурами и функциями для работы с Oracle JVM (большой частью - внутреннего пользования)

Компонента	Описание
DBMS_JAVA_TEST	Системный пакет для отладки хранимых процедур
Jpublisher	Средство построения классов Java на основе объектных типов и типов REF в Oracle

В зависимости от характера перечисленных компонент они заводятся либо при установке программной среды работы Oracle, либо при создании в БД среды JServer/OJVM.

Схема вызова хранимых Java-программ

Хранимым Java-программам в Oracle соответствуют методы Java, подверженные следующим ограничениям (версия 8.1):

методы, публикуемые для использования в SQL или PL/SQL, должны быть объявлены как статические

классы не могут делать во время исполнения обращения к GUI-классам (например, к awt)

Установка JServer/OJVM в версиях 8.1 и 9

Проще и короче всего установить JServer/OJVM в виде побочного следствия установки одной из стандартных конфигураций программной среды Oracle (например, Typical или Minimal в версии 8.1).

Тем не менее, JServer/OJVM можно доустановить к имеющейся программной среде, если он отсутствовал ранее, путем запуска сценария initjvm.sql из каталога %ORACLE_HOME%\javavm\install (система обозначений Windows).

Пример создания хранимой Java-программы

Хранимые Java-программы могут создаваться в БД под Oracle двумя способами:

загрузкой извне с помощью программы loadjava и SQL-предложением CREATE/ALTER JAVA ...

Ниже показаны оба способа на примере класса, создаваемого в рамках пакета training.demos.

Создание с помощью loadjava

Пусть в каталоге training/demos имеется файл MyJavaAgentInOracle.java (имеет отличия от файла MyHi.java, приведенного выше):

```
package training.demos; public class MyJavaAgentInOracle {  
    public static String sayHello (String toWhom) {  
        return "Hello, " + toWhom + "!";  
    }  
}
```

Загрузка в схему SCOTT БД текста кода для класса в этом файле (система обозначений Windows; в Unix-оболочках аналогично):

```
set CLASSPATH=%CLASSPATH%;%ORACLE_HOME%\javavm\aurora.zip  
(в версии 9 %CLASSPATH%;%ORACLE_HOME%\javavm\lib\aurora.zip)  
loadjava -user scott/tiger -o  
training/demos/MyJavaAgentInOracle.java
```

Если в том же каталоге у нас будет транслированный программой java с класс MyHiFromOracle, можно будет загрузить в БД сразу его:

```
loadjava -user scott/tiger -o  
training/demos/MyJavaAgentInOracle.class
```

Создание SQL-предложением

Загрузить код того же класса можно по-другому:

```
CREATE          JAVA          SOURCE          NAMED  
"training/demos/MyJavaAgentInOracle" AS  
public class MyJavaAgentInOracle { public static String  
sayHello (String toWhom) {          return "Hello, " +
```

```

toWhom + "!" ;
    }
};
/

```

Обращение к загруженной в Oracle процедуре Java

Обращение к Java-программе из Java-кода делается как обычно.

Для обращения к сохраненной в БД Java-программе из PL/SQL, ее следует опубликовать для этого языка:

```

CREATE FUNCTION say_hello_from_java_to (to_whom IN
VARCHAR2)
RETURN VARCHAR2

```

AS LANGUAGE JAVA

NAME 'training.demos.MyJavaAgentInOracle.sayHello (java.lang.String)

return java.lang.String';

После этого можно выполнить в SQL*Plus:

```

SET SERVEROUTPUT ON
EXEC
DBMS_OUTPUT.PUT_LINE(say_hello_from_java_to('World'))

```

Работа со словарем-справочником

Организация справочной информации

Справочная информация о программных элементах Java распределена между словарем-справочником СУБД (таблица DBA_OBJECTS) и специальными структурами, создаваемыми в каждой схеме, владеющей этими элементами.

При первой загрузке программных элементов Java в любую схему loadjava или команда CREATE JAVA создадут там:

CREATE\$JAVA\$LOB\$TABLE - таблицу для хранения кода Java-программ

JAVA\$CLASS\$MD5\$TABLE - хеш-таблицу для хранения цифровых подписей (digest) для каждого загружаемого объекта (с целью учета необходимости перетранслировать предъявляемый объект)

Несколько вспомогательных объектов, играющих вместе с этими двумя таблицами роль своеобразного "словаря-справочника программных элементов Java" в конкретной схеме.

Фактически загрузка программой loadjava вызывает неявную выдачу команды CREATE JAVA Описание программных элементов Java заносится в таблицу CREATE\$JAVA\$LOB\$TABLE. Повторная загрузка одного и того же Java-элемента реально выполняться не будет, если только (а) он не изменил свое описание или (б) не указан ключ -force при вызове программы loadjava.

Просмотр Java-элементов

Java-объекты, заведенные в схеме, можно просмотреть из таблицы USER_OBJECTS словаря-справочника обычным способом:

```
COLUMN object_name FORMAT A30
SELECT object_name, object_type, status, timestamp
FROM user_objects
WHERE object_name NOT LIKE 'SYS_%' AND
      object_name NOT LIKE 'CREATE$%' AND
      object_name NOT LIKE 'JAVA$%' AND
      object_name NOT LIKE 'LOADLOB%' AND
      object_type LIKE 'JAVA %'
ORDER BY 2, 1;
```

Просмотр исходных текстов

Выгрузить из БД исходные тексты из "словаря-справочника объектов Java" конкретной схемы можно с помощью процедур пакета DBMS_JAVA:

```
DECLARE
  PROCEDURE put_java_source(jclass IN VARCHAR2) IS
    b CLOB;
    v VARCHAR2(4000);
    i INTEGER := 4000;
  BEGIN
    DBMS_LOB.CREATETEMPORARY(b, FALSE);
```

```

        DBMS_JAVA.EXPORT_SOURCE(jclass, b);
        DBMS_LOB.READ(b, i, 1, v);
        DBMS_OUTPUT.PUT_LINE(v);
    END;
BEGIN
put_java_source('training/demos/MyJavaAgentInOracle');
END;
/

```

Исходные тексты программ на Java в БД можно посмотреть также в консоли Oracle Enterprise Manager или в аналогичных системах третьих фирм.

Преобразование имен

Стандарт именования классов в Java допускает более длинные имена, чем предел в 30 знаков в SQL Oracle. Достаточно длинные Java-имена Oracle при помещении в словарь-справочник самостоятельно заменяет на придуманные более короткие. Получить первоначальное имя по присвоенному Oracle можно с помощью функции DBMS_JAVA.LONGNAME. Пример ее использования:

```

COLUMN shortname FORMAT A30
COLUMN longname FORMAT A60

SELECT                                     object_name
shortname,DBMS_JAVA.LONGNAME(object_name) longname
FROM user_objects
WHERE object_type = 'JAVA CLASS';

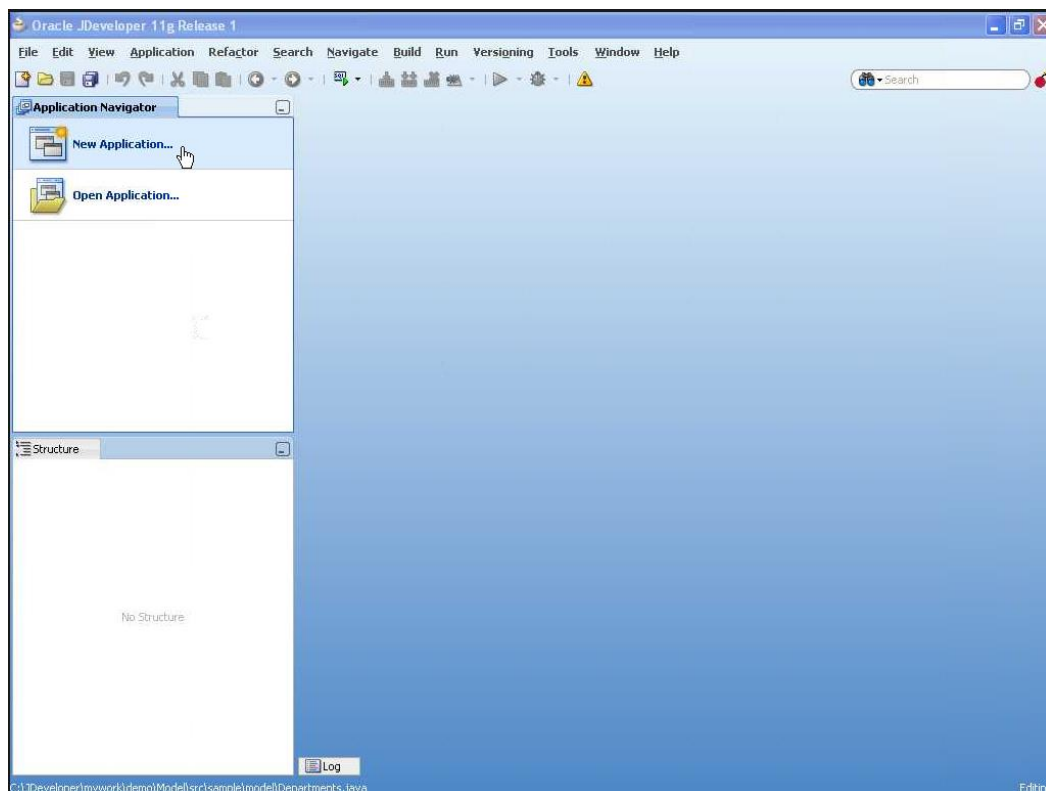
```

Oracle JDeveloper

Oracle JDeveloper представляет собой полную интегрированную среду для разработки сервисно-ориентированной архитектуры (Service-Oriented Architecture, SOA) и Java-приложений.

JDeveloper является частью ПО Oracle Fusion Middleware. Он базируется на архитектуре "подключаемых модулей" и может встраиваться как в среды Oracle, так и в среды других производителей. JDeveloper поддерживает все основные сервера приложений и баз данных J2EE.

Общий вид Oracle JDeveloper представлен на следующем рисунке

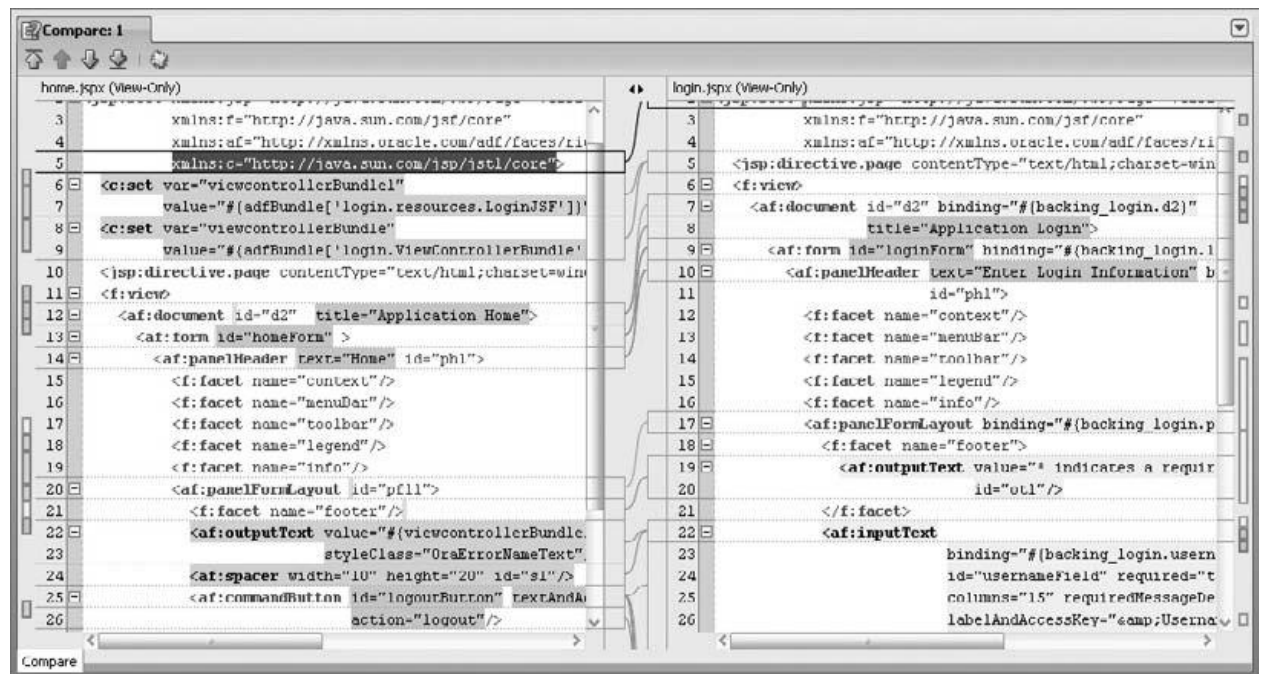


1. Описание меню

2.1. Меню Файл (File Menu)

Это меню содержит такие операции как Открыть файл (Open), закрыть файл (Close), сохранить файл (Save) и сохранить все файлы, которые были изменены (Save All). Это меню также содержит операцию Удалить файл (Delete). Операция Новый (New) создает новый файл или другой элемент, такой как связь. Опция Переименовать (Rename) меняет имя файла.

Подменю Сравнить файлы (Compare With) позволяет сравнить два файла: текущий и выбранный, для того чтобы найти разницу между ними, как показано ниже



Меню Правка (Edit Menu)

Меню предоставляет такие стандартные функции, как: вставить, вырезать, копировать, удалить, отменить, вернуть.

2.3. Меню Вид (View Menu)

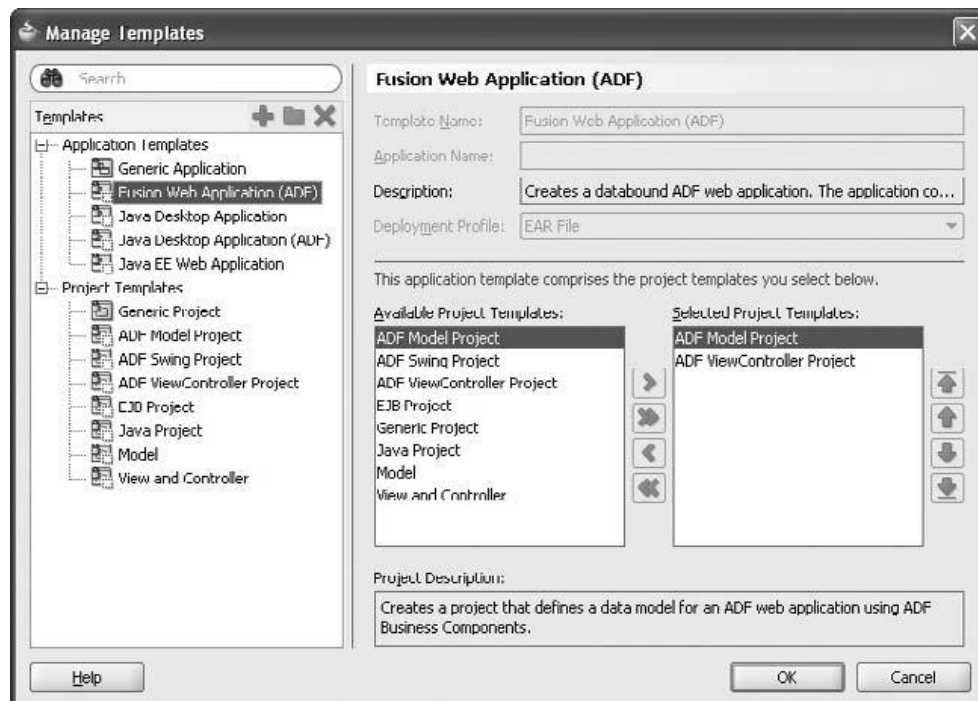
Обычно JDeveloper предоставляет набор окон и редакторов, соответствующих решаемой задаче. С помощью меню Вид, Вы можете отобразить дополнительные окна, редакторы, панели инструментов, необходимы Вам. Так же Вы можете скрыть не нужные Вам окна, редакторы, панели инструментов. С помощью меню Вид можно так же сделать окно активным или не активным.

2.4. Меню Приложений (Application Menu)

Это меню предоставляет средства для управления приложениями JDeveloper. Это меню позволяет открывать, закрывать, переименовывать и удалять приложения JDeveloper. Меню содержит следующие подменю:

- Найти файлы приложения (Find Application Files) Позволяет находить файлы, относящиеся к данному приложению

- Управление шаблонами (Manage Templates). Подменю открывает окно управления шаблонами, где можно добавить, удалить и просмотреть шаблоны для JDeveloper. Окно управления шаблонами представлено ниже:



- Редактировать пакет ресурсов (Edit Resource Bundle) Это подменю позволяет редактировать пакеты ресурсов для Вашего проекта JDeveloper. Здесь можно редактировать сообщения и значения, используемые в Вашем проекте

- Свойства проекта по умолчанию (Default Project Properties) Это подменю позволяет редактировать и создавать свойства Ваших последующих проектов

2.5 Меню Рефакторинг (Refactor Menu)

Меню содержит функции для рефакторинга кода, переименования или перемещения файла и каскадного изменений всех зависимых от них файлов. Также позволяет переименовывать классы, так что все дочерние классы будут соответствовать новому имени. Эти функции позволят Вам благополучно изменять код в любое время.

2.6. Меню Поиска (Search Menu) Это меню предоставляет стандартные функции поиска – поиска по всему проекту, по заданному файлу, искать файл по имени или части кода.

2.7 Меню Навигации (Navigate Menu) Это меню позволяет для определенных строк кода создавать закладки, для того чтобы проще ориентироваться в коде. Через это меню можно создавать, удалять закладки и перемещаться от одной к другой. С помощью этого меню также можно переходить к определенной строке кода, задавая в меню навигации ее номер.

2.8 Build Menu. Это меню позволяет скомпилировать проект.

2.9 Запуск (Run Menu) Позволяет запустить проект. Проект запустится, только если не содержит ошибок

2.10 Отладка (Debug) С помощью этого меню можно запустить отладку проекта. Отладка будет проходить до первой точки останова. Такие точки может задать пользователь. Отладку так же можно запустить с определенной точки.

2.11 Меню Версия (Versioning Menu) Это меню предоставляет возможности создавать резервные копии Вашего проекта, а так же сравнивать текущую версию проекта с предыдущими.

2.12 Меню Инструментов (Tools Menu) Содержит инструменты для экспорта и импорта диаграмм базы данных Oracle, меню для управления библиотеками. Позволяет связать базу данных с Вашим проектом.

2.13 Меню Окна (Windows Menu) Меню позволяет переходить от одного окна к другому, располагать окна каскадом.

2.14 Меню Помощь (Help Menu) Меню содержит справку, позволяет обратиться в центр помощи Oracle.

2. Элементы управления JDeveloper

Чтобы перейти к элементам управления, необходимо в окне Вашего файла проекта выбрать дизайн (Design). Справа появятся элементы управления. Для того чтобы добавить элемент управления необходимо перетащить его на форму. При двойном щелчке на добавленный элемент управления появится функция, которая будет происходить при взаимодействии с элементом

управления. Туда необходимо добавить код, чтобы по нажатию происходило необходимое действие. Внизу располагаются свойства элементов управления, где можно задать их имя, текст, внешний вид элемента управления. Далее представлены элементы управления, доступные в JDeveloper:

3.1 JButton (Кнопка). Щелчок по кнопке вызывает некоторое действие.

3.2 JCheckBox (Флажок)

Элемент управления для взаимно исключающего выбора.

3.3 JComboBox (Выпадающий список)

Дает возможность создать список, по нажатию кнопки стрелка вниз, будут выпадать другие значения этого списка. В свойствах можно задать количество и названия выпадающих значений, а так же то значение, которое будет выбрано по умолчанию.

3.4. JLabel (Метка)

В нее можно добавить текст, или задать вывод текста в нее. Для нее невозможно задать код логики.

3.5. JList (Список)

С помощью этого элемента управления можно создать список в JDeveloper.

3.6. JProgressBar

Позволяет добавить на форму индикатор выполнения или загрузки

3.7. JScrollBar

Позволяет добавить на форму вертикальную или горизонтальную полосу прокрутки

3.8 JRadioButton (Переключатель)

Позволят разместить на форме переключатели, для взаимоисключающего выбора.

3.9 JTextField (Текстовое поле)

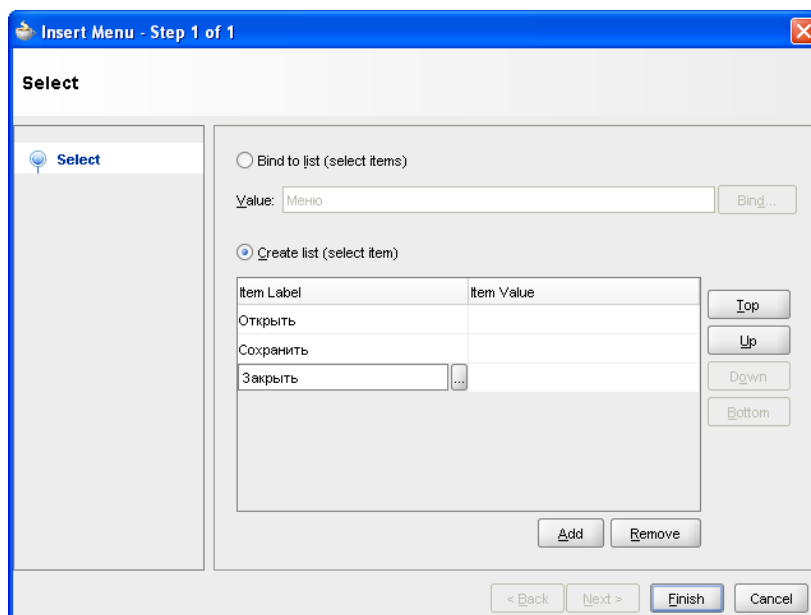
Позволяет добавить на форму поле для ввода текста

3.10 JTable (Таблица)

Позволяет добавить на форму таблицу. В свойствах можно задать количество колонок и строк таблицы.

3. Создание меню в JDeveloper

Для того чтобы создать меню в JDeveloper необходимо в окне Component Palette выбрать JMenuBar и перетащить на форму. В появившемся окне задать имя меню, количество и названия подменю:

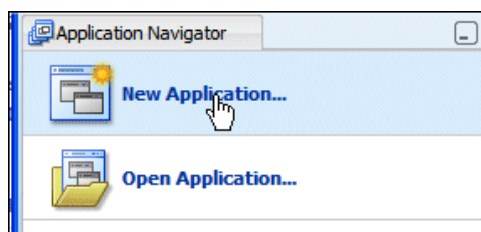


После чего нажать Готово.

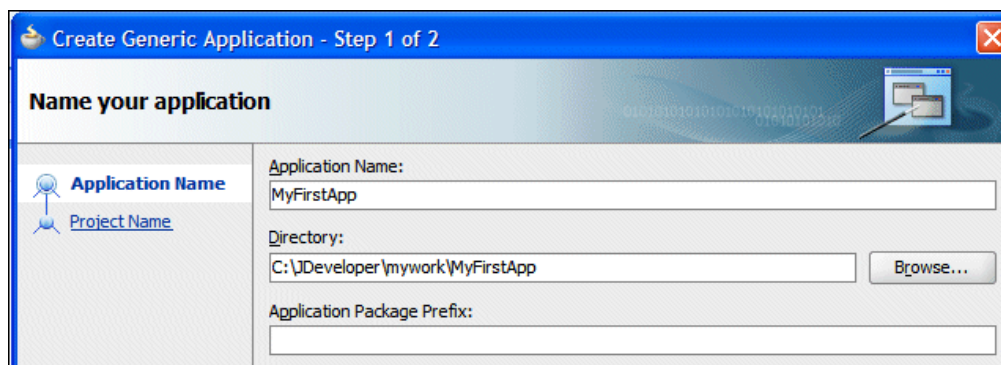
4. Создание проекта в JDeveloper

5.1 Запустите Oracle JDeveloper

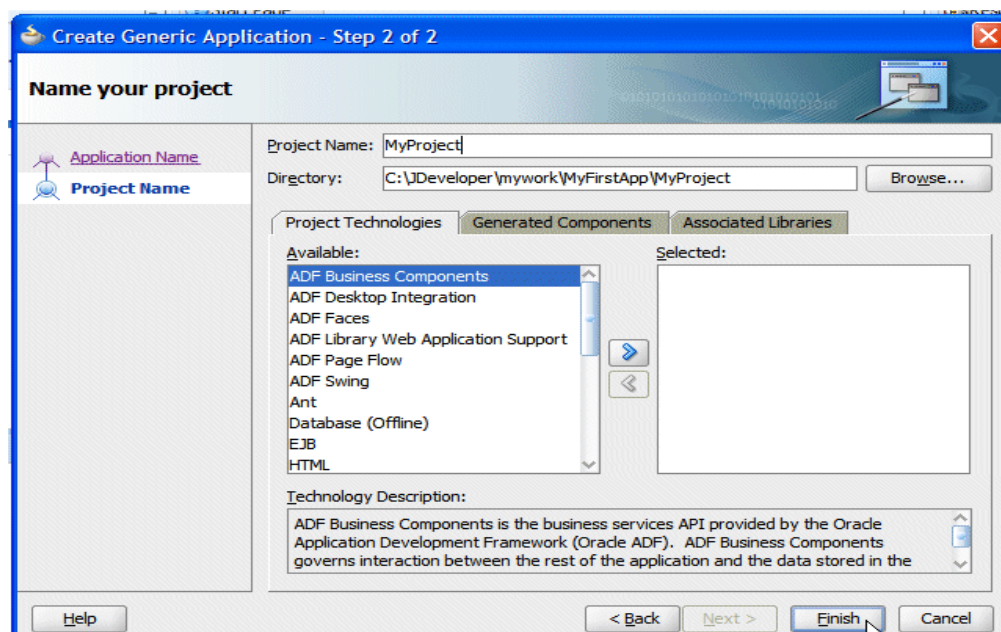
5.2 Нажмите New Application



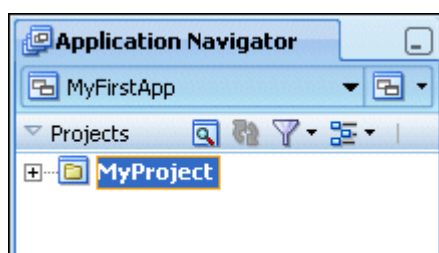
5.3 В появившемся окне измените имя приложения по умолчанию на MyFirstApp



5.4 В следующем окне измените имя проекта



5.5 В Application Navigator слева вы увидите следующее



Практическое занятие № 8.

Создание проекта JDeveloper.

Создайте проект Oracle JDeveloper с использованием не менее 10 элементов обработки и не менее 5 методов обработки данных

Практическое занятие № 9.

Создание меню

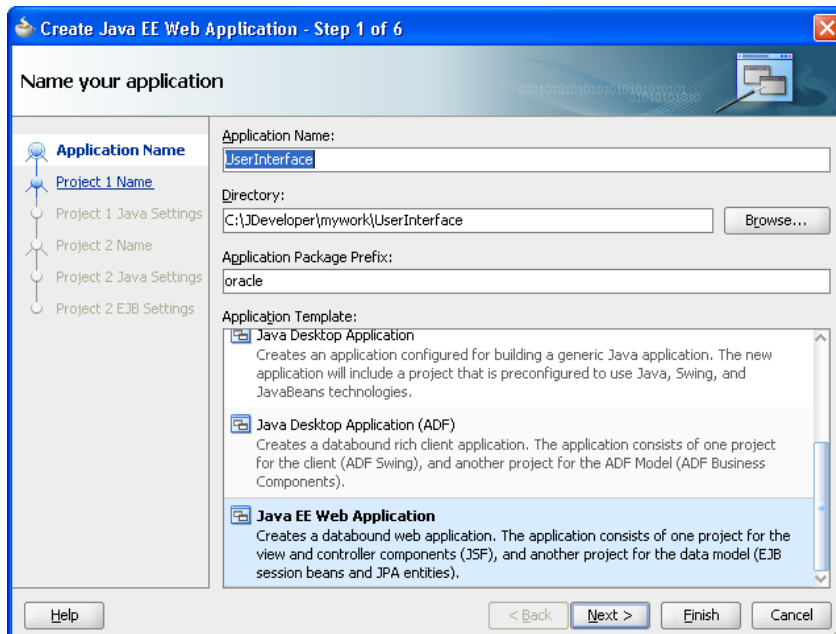
К предыдущему проекту добавьте 2 меню, и не менее 3х подменю в каждом

Создание пользовательских интерфейсов в jDeveloper

jDeveloper позволяет создавать различные пользовательские интерфейсы. Рассмотрим создание веб-интерфейса, работающего с базой данных.

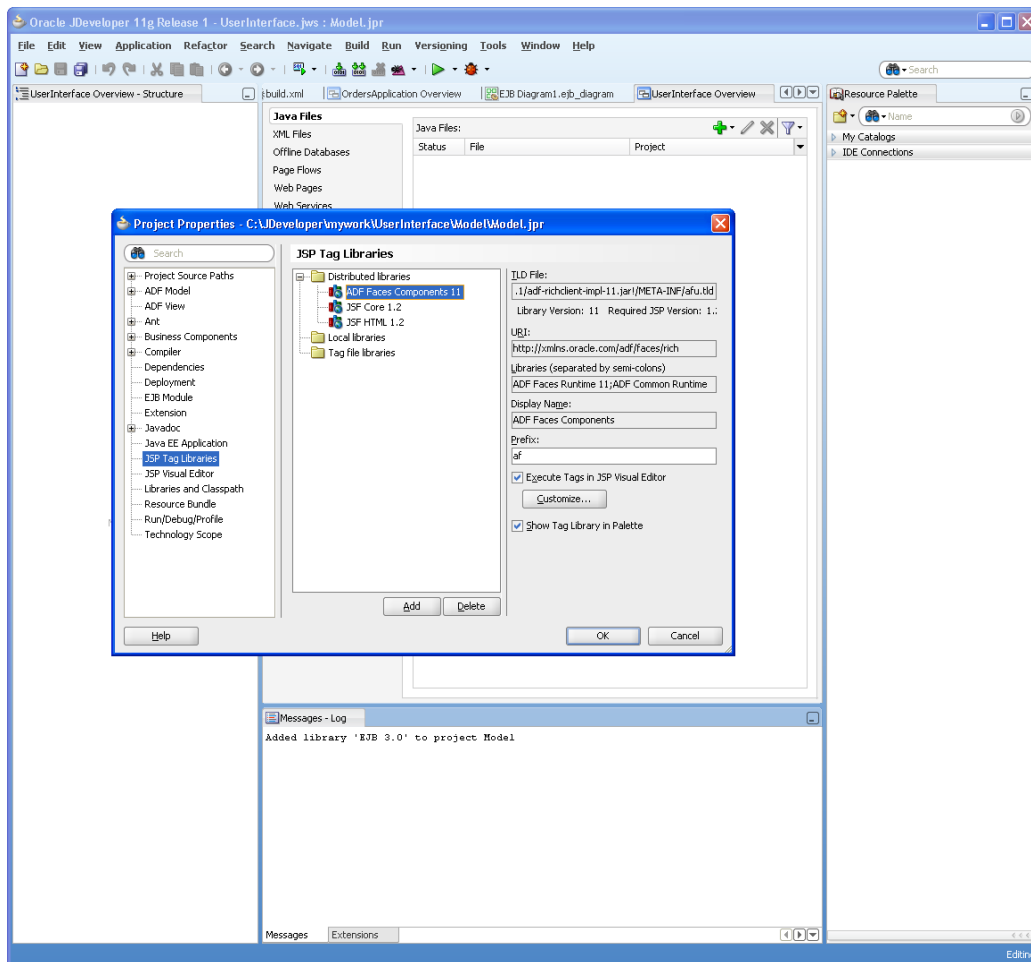
Создадим и сконфигурируем проект:

1. Создадим проект File->New и выберем Java EE Web Application:

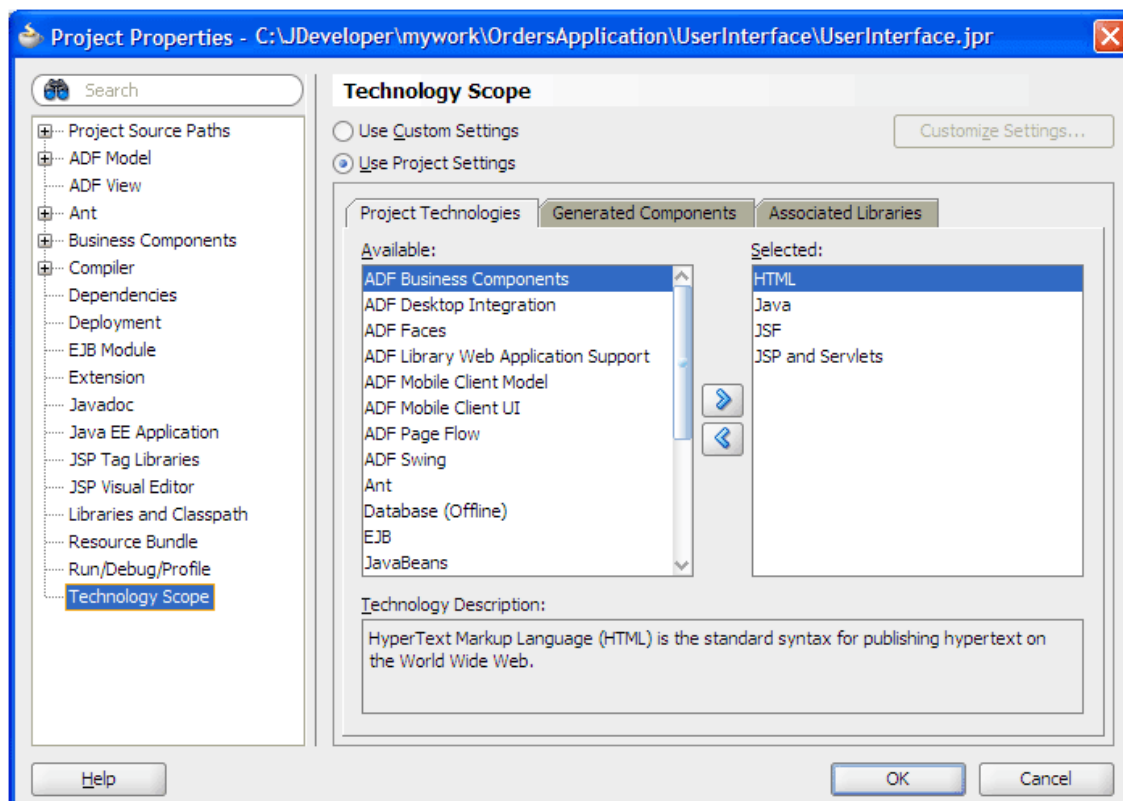


Оставим остальные параметры по-умолчанию, нажав кнопку Finish.

2. Добавим необходимые библиотеки расширения: Application->Project properties...->JSP Tag Libraries. Добавьте библиотеки, чтобы экран выглядел так:



3. Затем перейдите на вкладку Technology Scope и приведите экран в соответствие со следующим скриншотом:



В двух предыдущих шагах были добавлены необходимые для разработки библиотеки и Фреймворки.

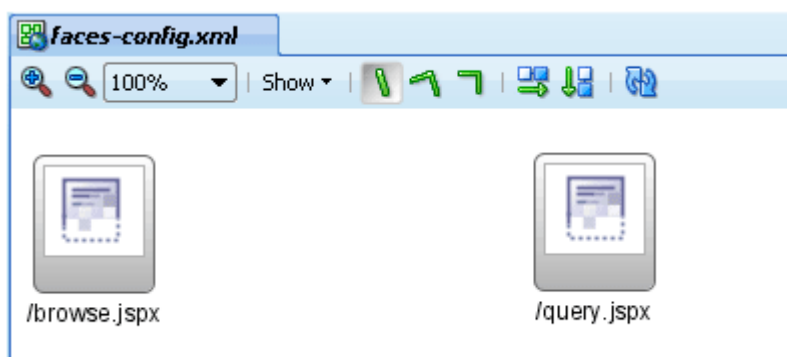
Далее необходимо создать обложку новой страницы.


1. В Application Navigator двойным кликом сделайте активным документ `faces-config.xml`

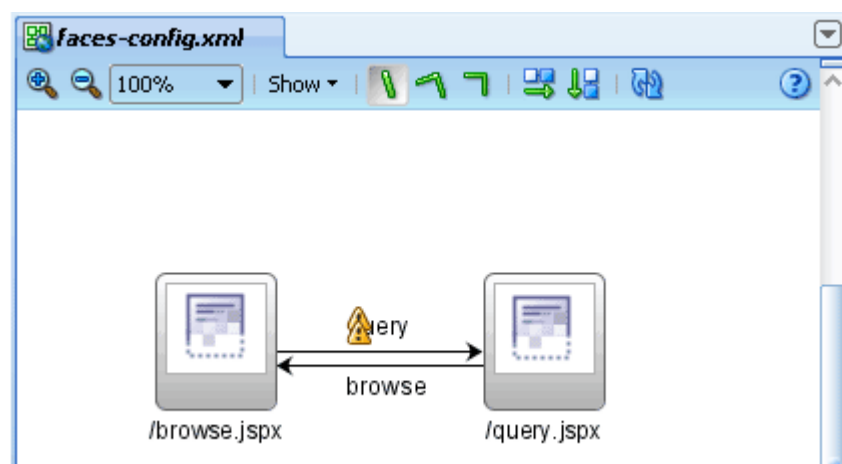
2. Откройте страницу JSF Diagram Objects, кликнув на  JSF Page

Все JSF приложения имеют JSF конфигурационный файл, называемый по умолчанию `faces-config.xml`. Этот файл содержит детальную информацию о приложении, такую как правила навигации по страницам и многое другое

3. Перетягивайте объекты из JSF Diagram Objects и создайте следующий вид окна:



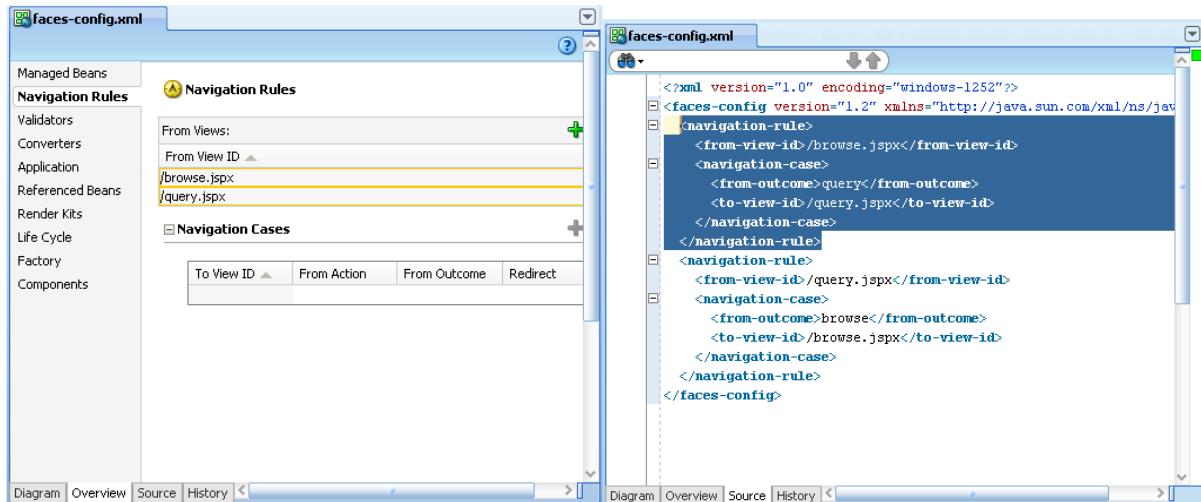
4. На навигационной панели кликните  JSF Navigation Case. Кликая по иконкам на диаграмме добейтесь следующего вида:



Это задаст правила перехода между страницами.

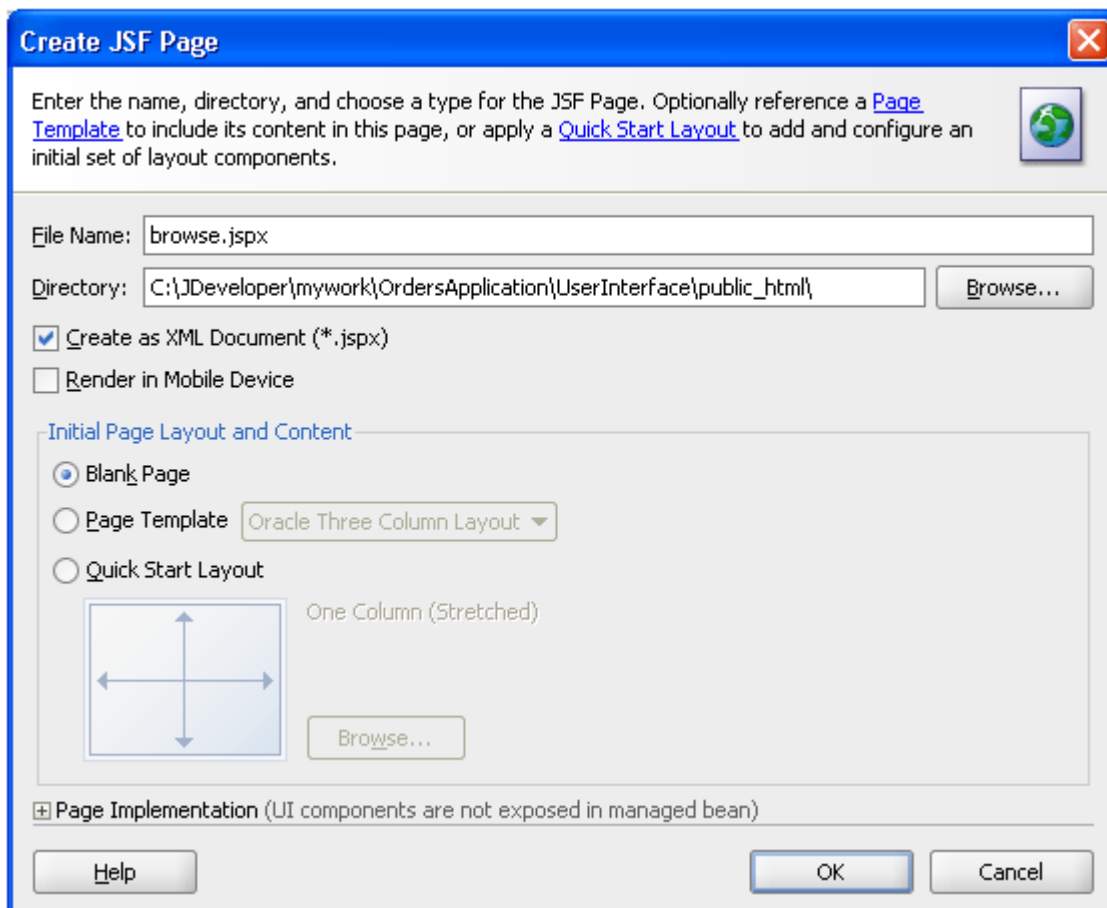
5. Убедитесь, что действия произведены успешно и сохраните изменения.




Вы можете увидеть результаты своей работы, заглянув в следующие вкладки:

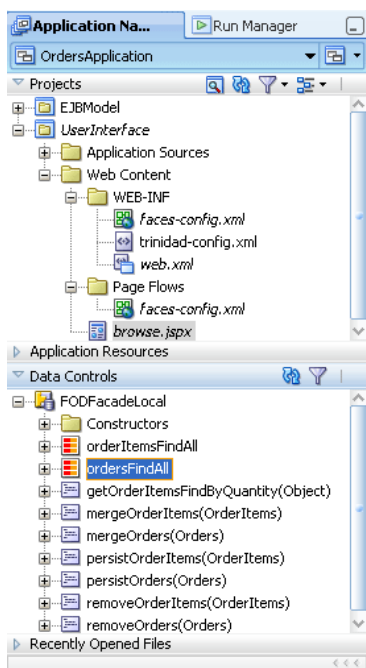


Создадим страницу Browse

1. Два раза кликните на browse.jspx в диаграмме для открытия диалога Create JSF Page dialog.
2. Приведите экран в соответствие со скриншотом

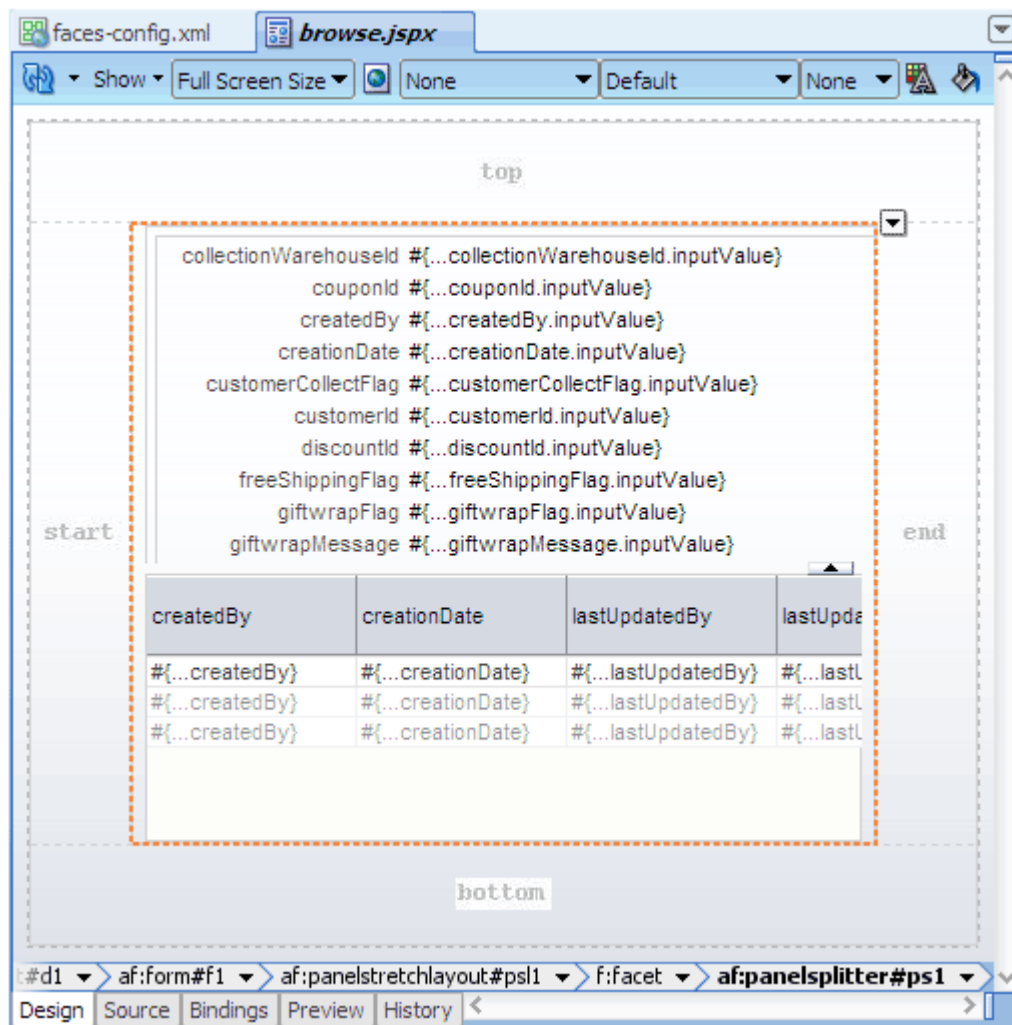


3. На Component Palette, странице ADF Faces, схватите  Panel Stretch Layout и перетащите его на визуальный редактор страницы.
4. Перетащите  Panel Splitter в центр страницы
5. Перетащите компонент  Panel Group Layout на левую сторону сплиттера. Укажите свойство scroll в выпадающем списке Layout.
6. В Application Navigator растяните панель Data Controls и узел FODFacadeLocal. Перетащите узел ordersFindAll на визуальный редактор. Должно получиться следующее:




7. В контекстном меню Create выберите Form->ADF Read-only Form
8. В диалоге Edit Form Fields выберите Include Navigation Controls и нажмите Ok.
9. На панели Data Controls растяните узел ordersFindAll. Перетащите orderItemsList на правую сторону страницы.
10. В контекстном меню Create выберите Table->ADF Read-only Table.
11. В диалоге Edit Table Columns выберите Row Selection и Sorting. Нажмите Ok.
12. В окне Structure страницы browse.jspx выберите компонент af:panelSplitter. В Property Inspector выберите vertical из выпадающего списка Orientation.

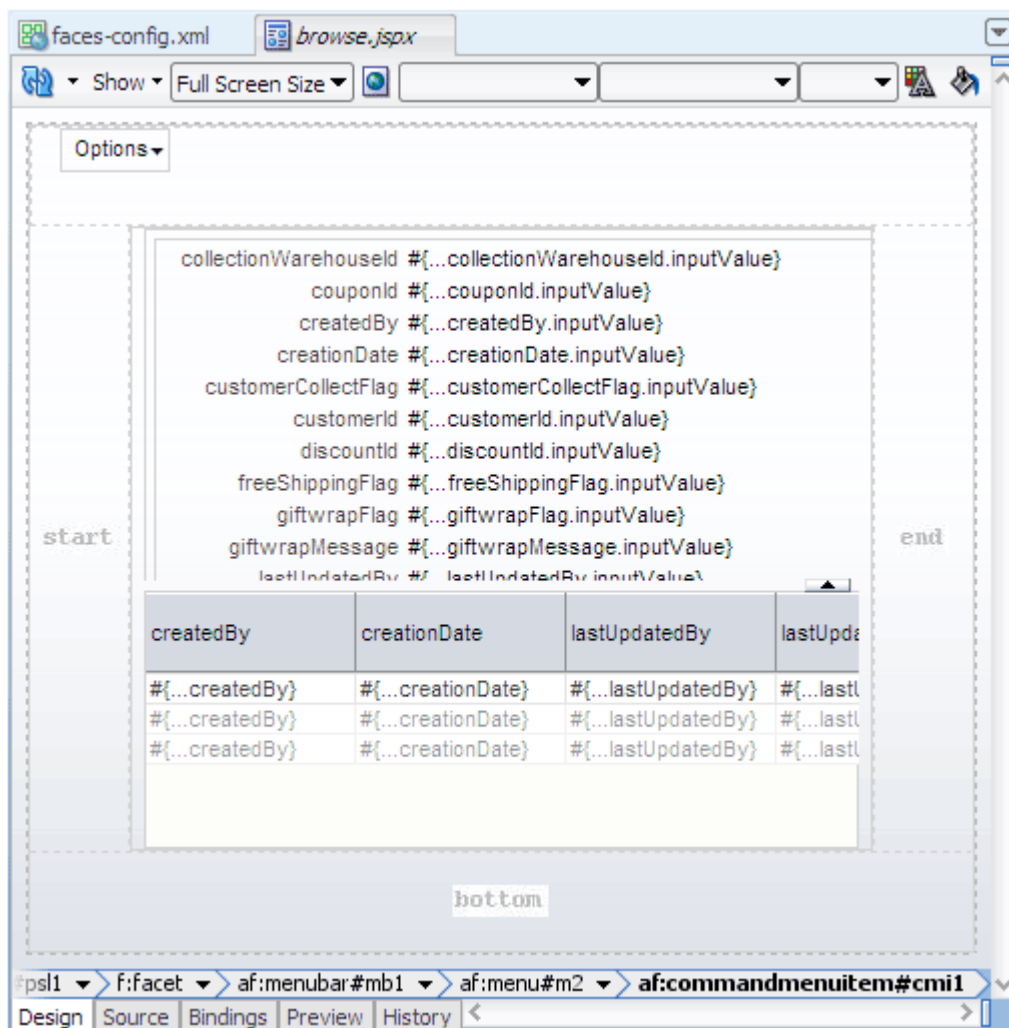
13. Приведите экран в соответствие со скриншотом



Добавим меню на главную страницу.

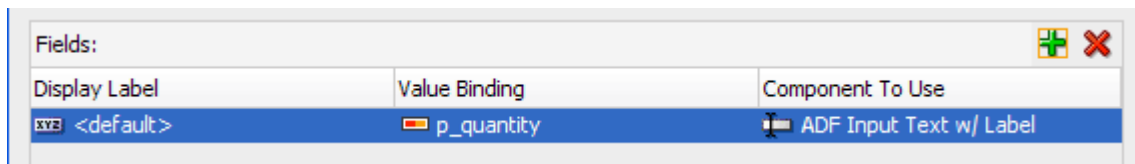
1. На Component Palette, странице ADF Faces, панели Common Components перетащите  Panel Menu Bar на верхнюю грань страницы в визуальном редакторе.
2. Перетащите Menu на панель области меню.
3. В Property Inspector, секции Common, задайте Options в текстовом поле для задания текста меню.
4. Растяните секцию Behavior и выберите true в выпадающем списке Detachable.
5. В окне Structure сделайте правый клик на компоненте af:menu и выберите insert Inside af:menu - Options | Menu Item.
6. В Property Inspector, секции Common, введите Query в текстовое поле.
7. Выберите query из выпадающего списка Action

8. Сохраните сделанную работу. Вы должны получить следующую картину:



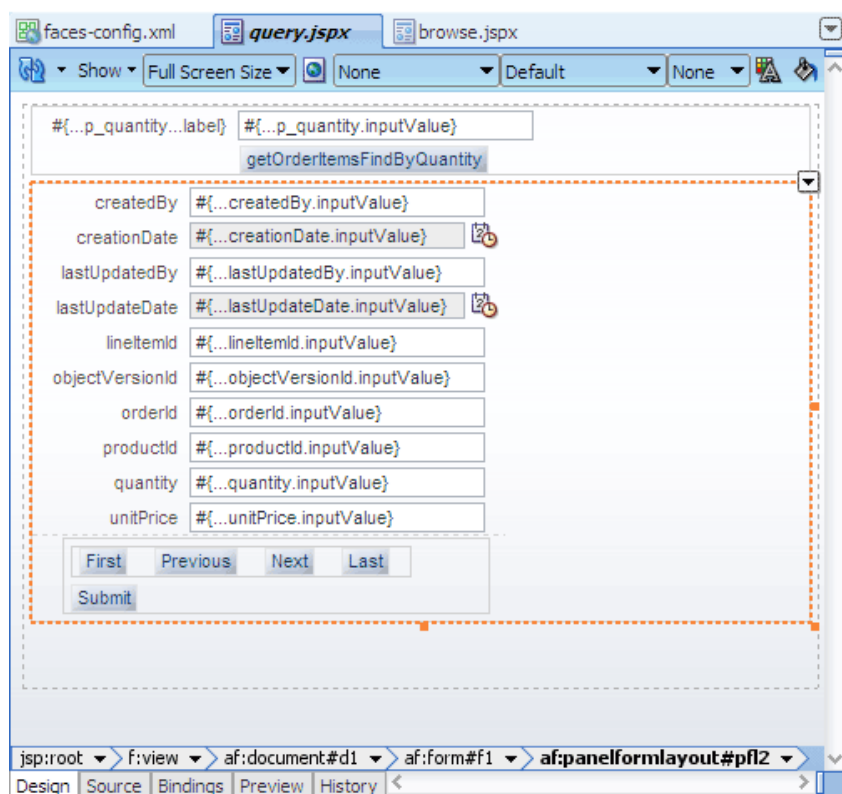
Создадим страницу Query

1. В окне editor кликните faces-config.xml для того, чтобы начать настройку.
2. На диаграмме дважды кликните query.jspx для того, чтобы открыть Create JSF Page.
3. Согласитесь с именем query.jspx и поставьте галочку Create as XML Document (*.jspx). Выберите Blank Page. Нажмите Ok.
4. На панели Data Controls, перетащите под узел FODFacadeLocal узел getOrderItemsFindByQuantity.
5. В контекстном меню Create выберите ADF Parameter Form. В диалоге Edit Form Fields примите все по-умолчанию и нажмите Ok. Все должно выглядеть как на скриншоте:

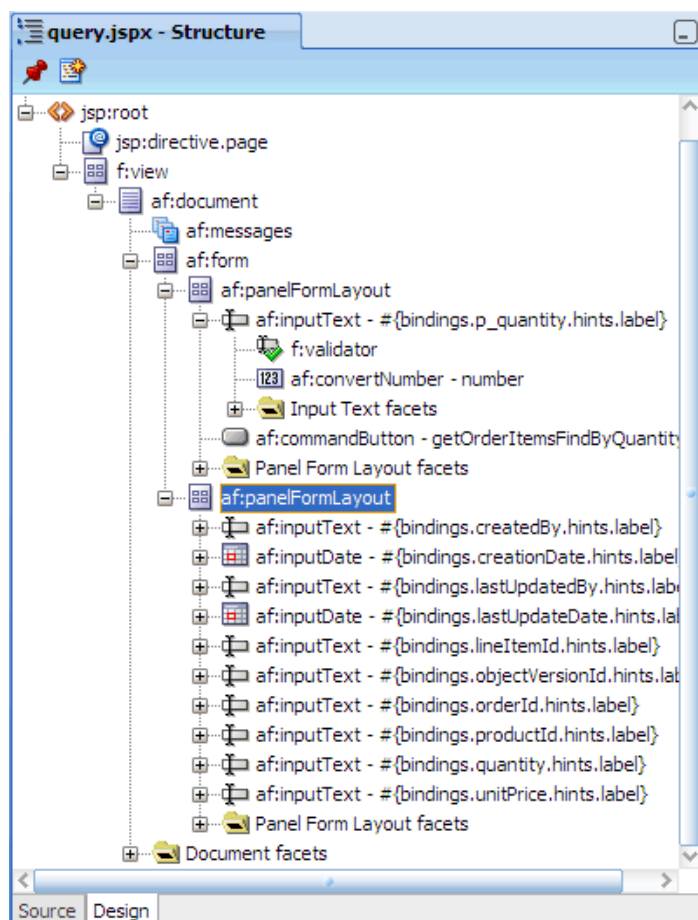


6. В компоненте Component Palette, странице ADF Faces, растяните панель Operations. Схватите узел Convert Number на странице и перетащите его на страницу. В Property Inspector выберите number из выпадающего списка Type.
7. На панели Data Controls растяните узел getOrderItemsFindByQuantity. Перетащите OrderItems на страницу и бросьте его ниже параметра, добавленного ранее.
8. В контекстном меню Create выберите Form->ADF Form.
9. В диалоге Edit Form Fields исключите следующее: createdBy, creationDate, lastUpdatedBy, lastUpdateDate, lineItemId, objectVersionId, orderId, productId, quantity и unitPrice кликая на кнопку Delete.
10. Выберите оба поля Include Navigation Controls и Include Submit Button. Нажмите Ok.
11. В окне Editor кликните Bindings.
12. Под узлом Executables выберите узел getOrderItemsFindByQuantityIterator. В Property Inspector, секции Advanced, выберите ifNeeded из выпадающего списка Refresh.
13. В окне editor кликните Design для возвращения обратно в визуальный редактор.

Страница должна иметь такой вид

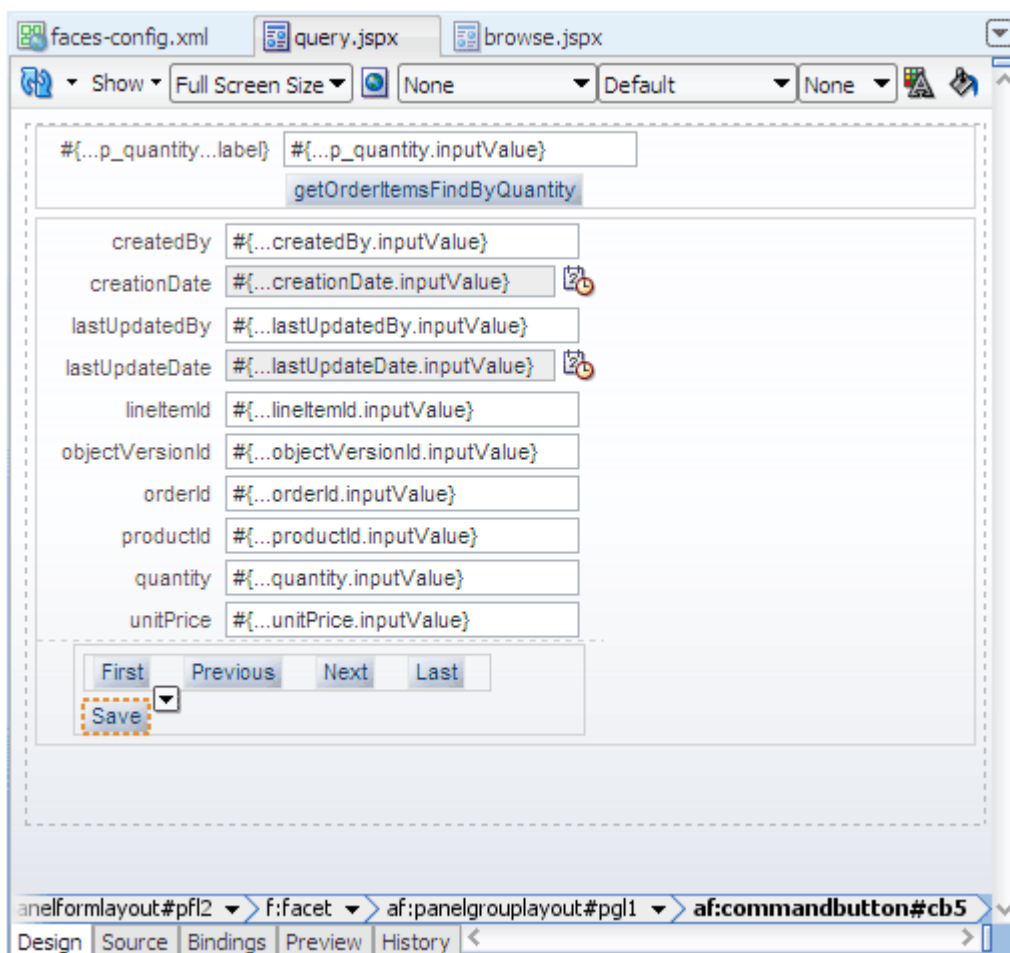


Окно Structure должно выглядеть так



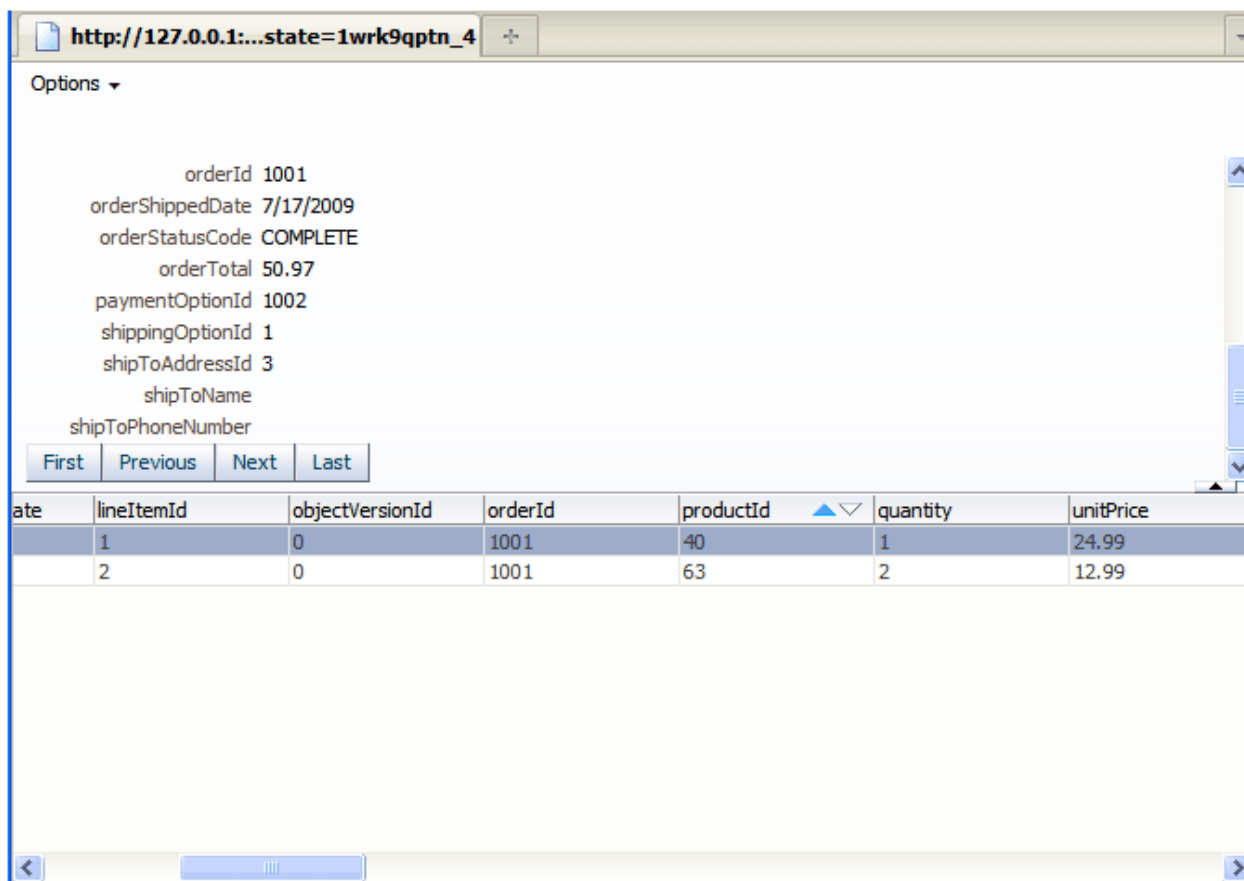
Делаем страницу Query обновляемой

1. С панели Data Controls перетяните метод mergeOrders(Orders) на кнопку Submit на странице query.jspx.
2. В диалоге Edit Action Binding кликните поле Value и выберите Show EL Expression Builder из выпадающего списка.
3. В диалоге Variables растяните ADF Bindings | bindings | getOrderItemsFindByQuantityIterator | currentRow и выберите dataProvider.
4. Кликните Ok для закрытия этого диалога.
5. Кликните Ok снова для закрытия диалога Action Binding.
6. В диалоге Confirm Component Rebinding кликните Ok.
7. В визуальном редакторе страницы query.jspx page выберите называющуюся теперь кнопку mergeOrders.
8. В Property Inspector, секции Common, введите Save в текстовое поле.
9. В Button Action выберите browse в выпадающем списке Action.
10. Сохраните изменения. Страница должна выглядеть следующим образом:



Запуск приложения.

1. В окне editor кликните таб faces-config.xml для возвращения в конфигурационный файл.
2. На диаграмме дважды кликните browse и нажмите кнопку Run.
3. В браузере запущенного приложения используйте скрол бар или двигайте сплиттер для того, чтобы скрыть или убрать кнопки.
4. Используйте навигационные кнопки для перемещения по строкам формы и изменения данных в таблицах. Это должно выглядеть так



5. Выберите заголовок и перетащите его в другое положение в таблице. Это изменит положение столбцов.
6. Сверху слева страницы кликните меню Options и переместите меню в любое другое место на странице.
7. В поле `getOrderItemsFindByQuantity_p_quantity` введите число, например 7 и кликните кнопку `getOrderItemsFindByQuantity`.
8. Поэкспериментируйте с навигацией.
9. Кликните на иконку календаря и выберите новую дату.

10. Кликните Save для сохранения изменений и возвращения в страницу browse.

Это должно выглядеть так

The screenshot shows a web browser window with the address bar displaying `http://127.0.0.1:...state=1wrk9qptn_4`. The page contains a form with the following fields and values:

- `getOrderItemsFindByQuantity_p_quantity`: 3
- `getOrderItemsFindByQuantity`: Button
- `createdBy`: 0
- `creationDate`: 7/27/2009
- `lastUpdatedBy`: 0
- `lastUpdateDate`: 7/27/2009
- `lineItemId`: 1
- `objectVersionId`: 0
- `orderId`: 1006
- `productId`: 63
- `quantity`: 4
- `unitPrice`: 12.99

At the bottom of the form, there are navigation buttons: `First`, `Previous`, `Next`, `Last`, and a `Save` button.

Практическое занятие № 10.

Создание графического изображения базы данных

Цель работы: создать графическое изображение базы данных и подключить к ней базу данных.

1. Для того чтобы создать графическое отображение базы данных выберите в меню Файл (File) Новый (New). Раскройте вкладку базы данных (Database Tier) и выберите категорию Оффлайновая база данных. Далее выберите графическое отображение базы данных (Data Base Diagram) и нажмите ОК.
2. Задайте графическому отображению имя DBOoffline1

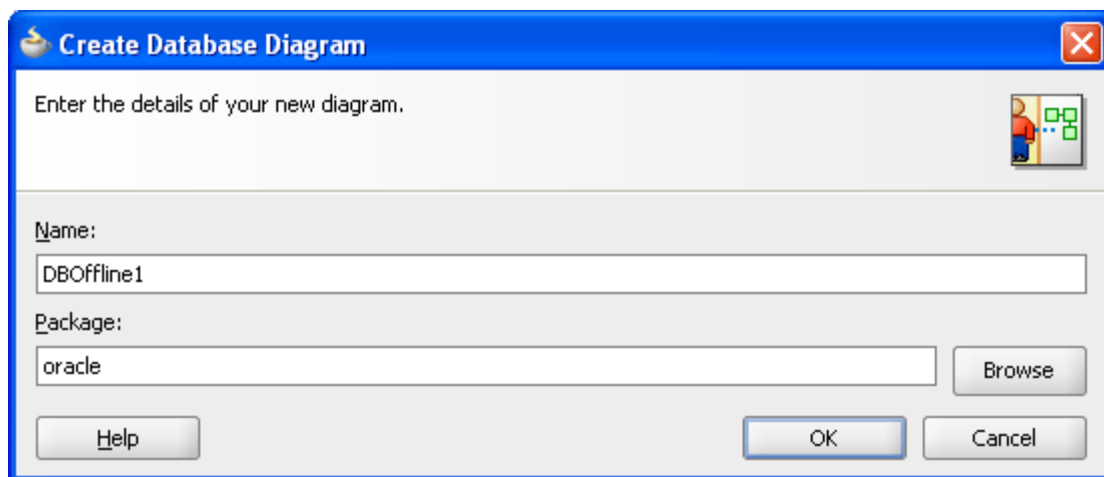


Рисунок 1 – создание графического изображения базы данных

3. Вы создали графическое отображение базы данных

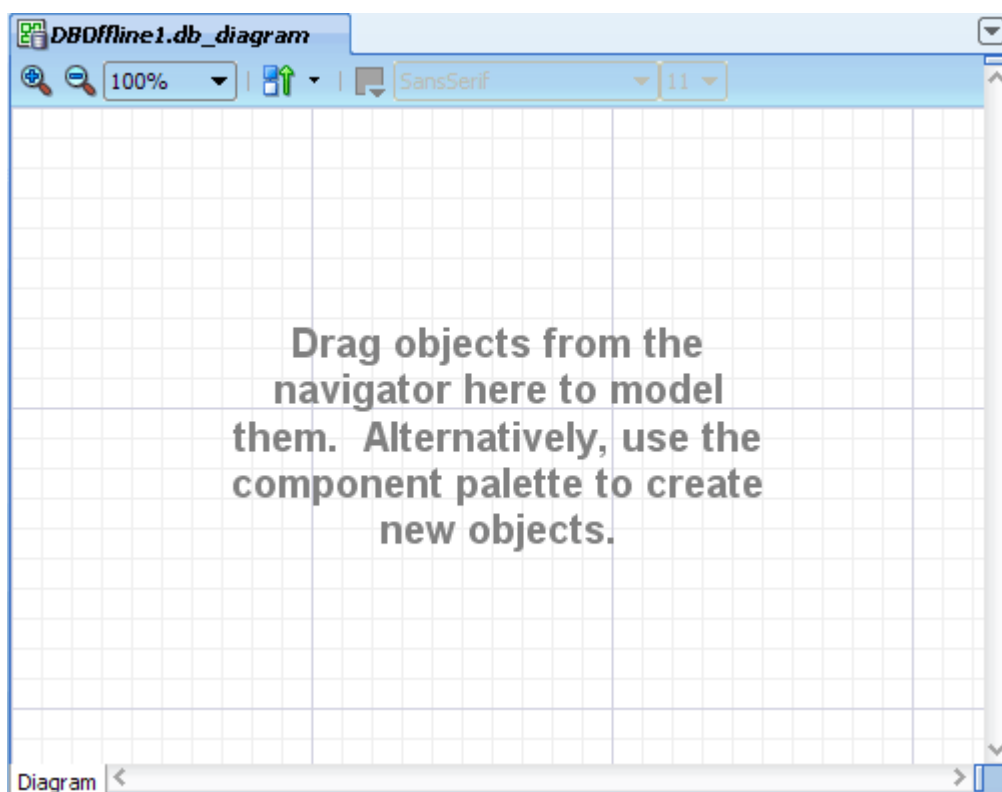

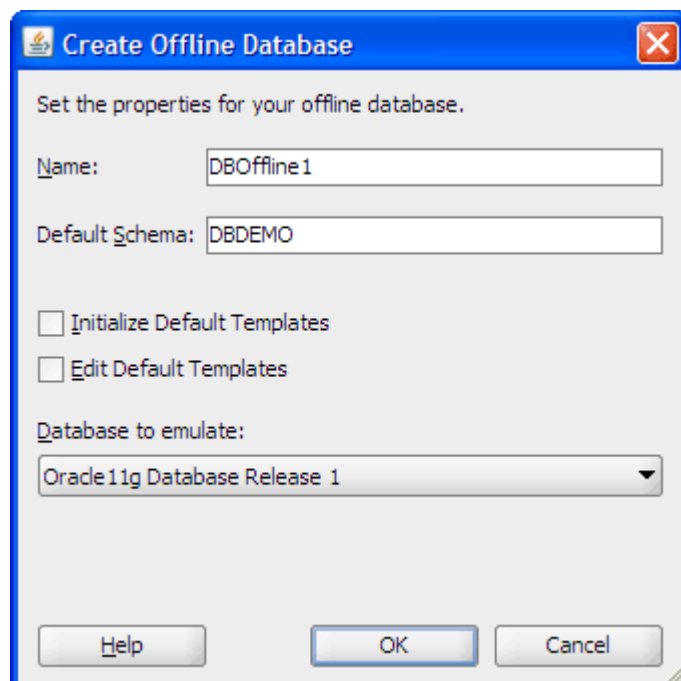
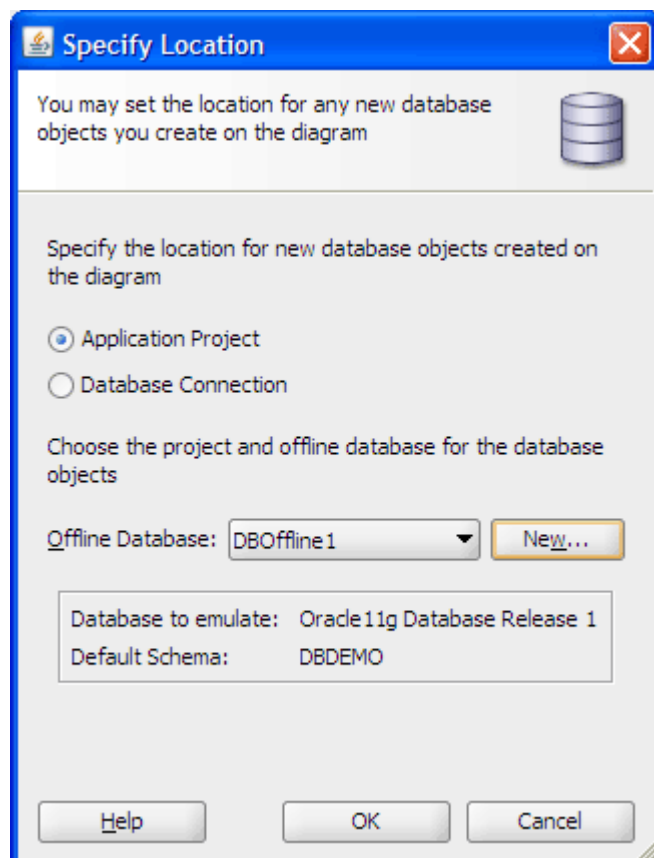



Рисунок 2 – графическое отображение базы данных

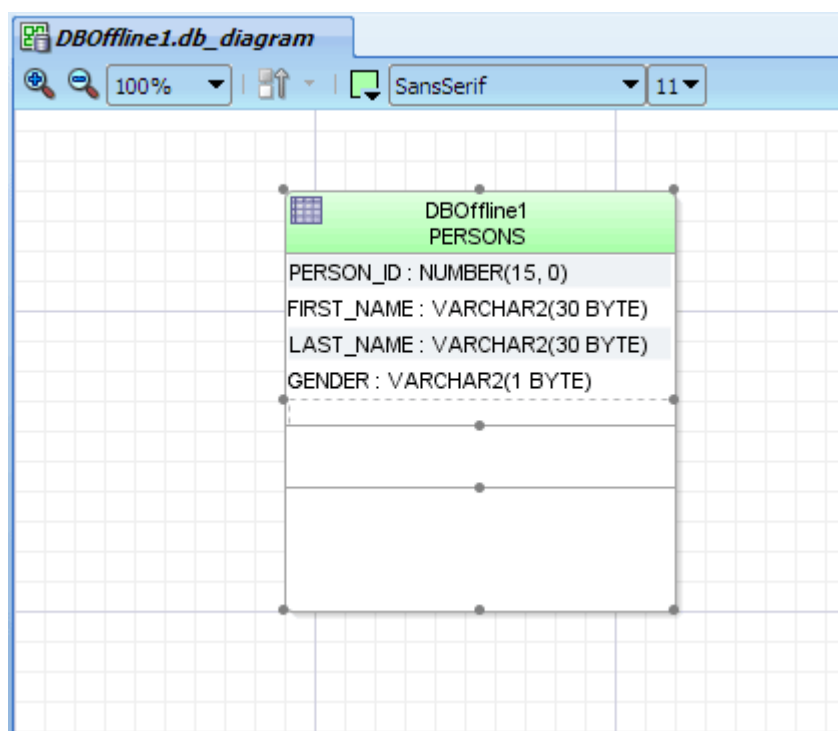
4. Из набора элементов управления для базы данных (Component Pallette) выберите таблицу  (Table) и перетащите на форму.
5. Далее нужно подключить базу данных. В меню Specify Location (определение местоположения) выберите Открыть (Open) и введите название базы данных DBOffline1. В качестве имени графического отображения введите DBDEMO.



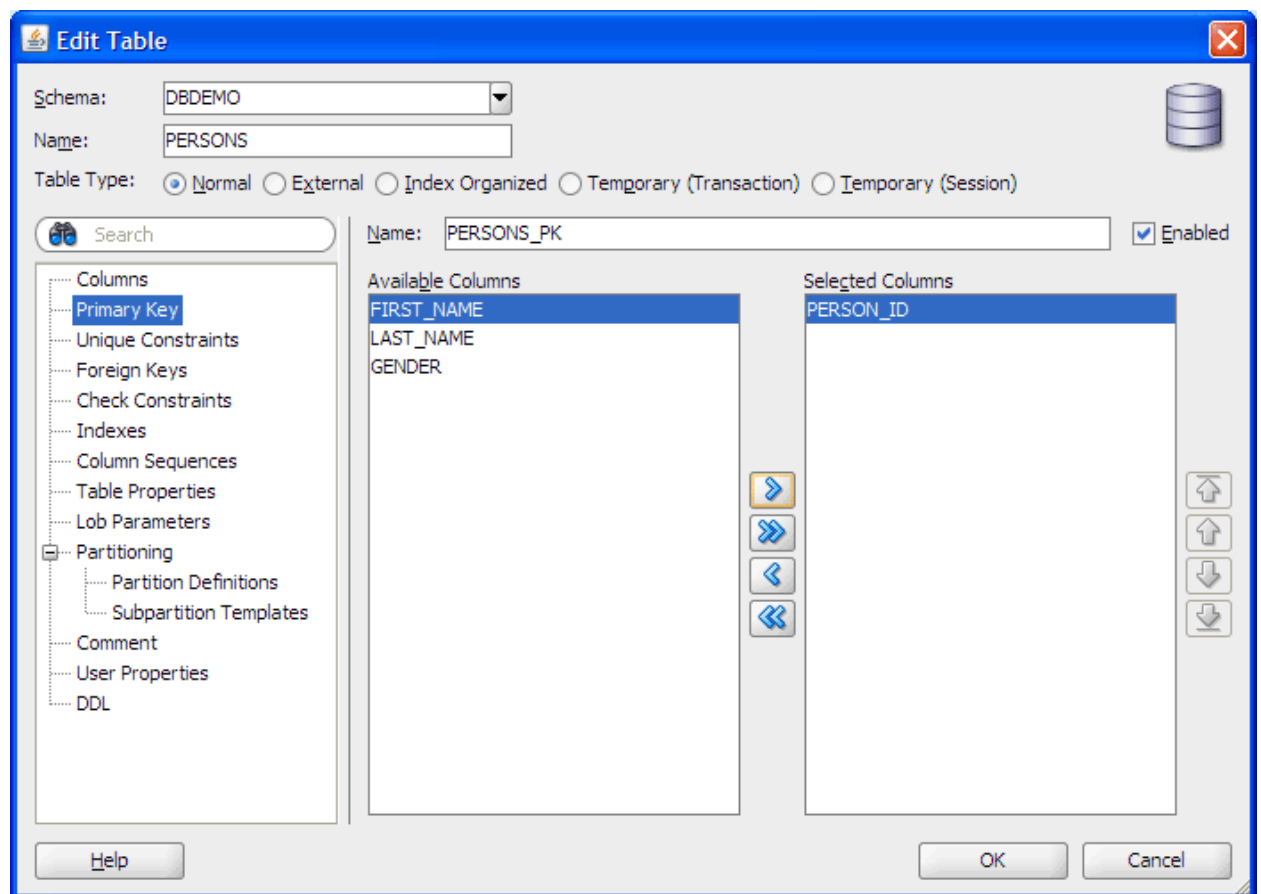
6. Нажмите  Save (сохранить) чтобы сохранить Ваше графическое отображение базы данных.

Практическое занятие 3 – Создание таблиц базы данных для графического отображения.

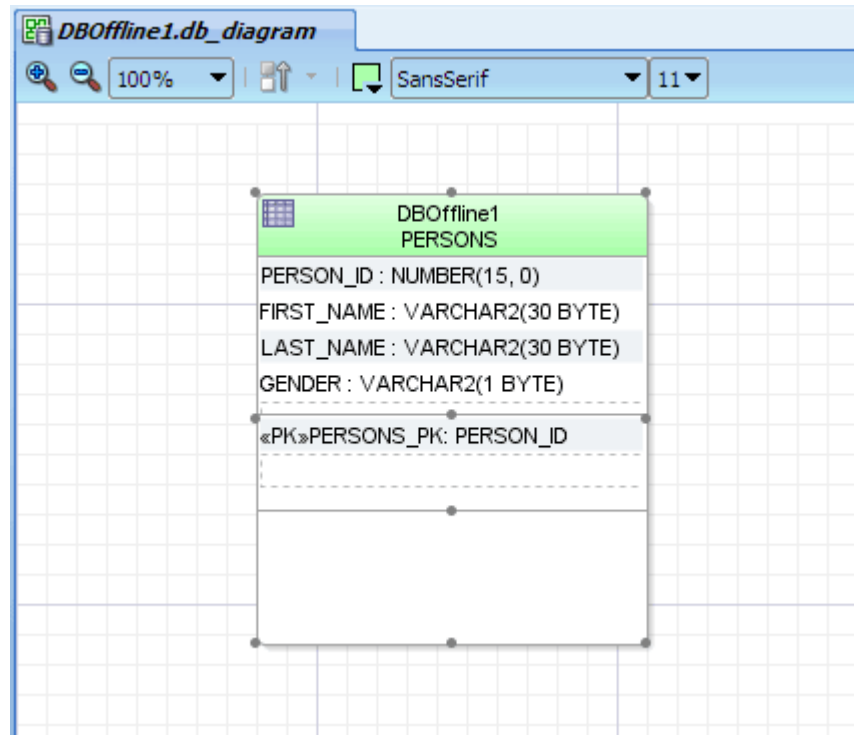
1. В графическом отображении базы данных выберите создать новую таблицу (create new table), по умолчанию ей будет присвоено имя TABLE1, измените его на PERSONS. Нажмите Enter.
2. Чтобы добавить колонку, кликните на таблице и введите PERSON_ID:NUMBER (15,0) и нажмите Tab, по нажатию кнопки Tab вы автоматически перейдете к созданию следующей колонки
3. Далее создайте колонки FIRST_NAME:VARCHAR2(30 BYTE), LAST_NAME:VARCHAR2(30 BYTE) и GENDER:VARCHAR2(1 BYTE).
Получится следующее:




4. Далее необходимо задать первичный ключ. Для этого дважды кликните на таблице и выберите меню редактирования (Edit Table dialog). Слева выберите Первичный ключ (Primary Key), справа будут отображены все доступные колонки. Выберите PERSON_ID и нажмите ➤



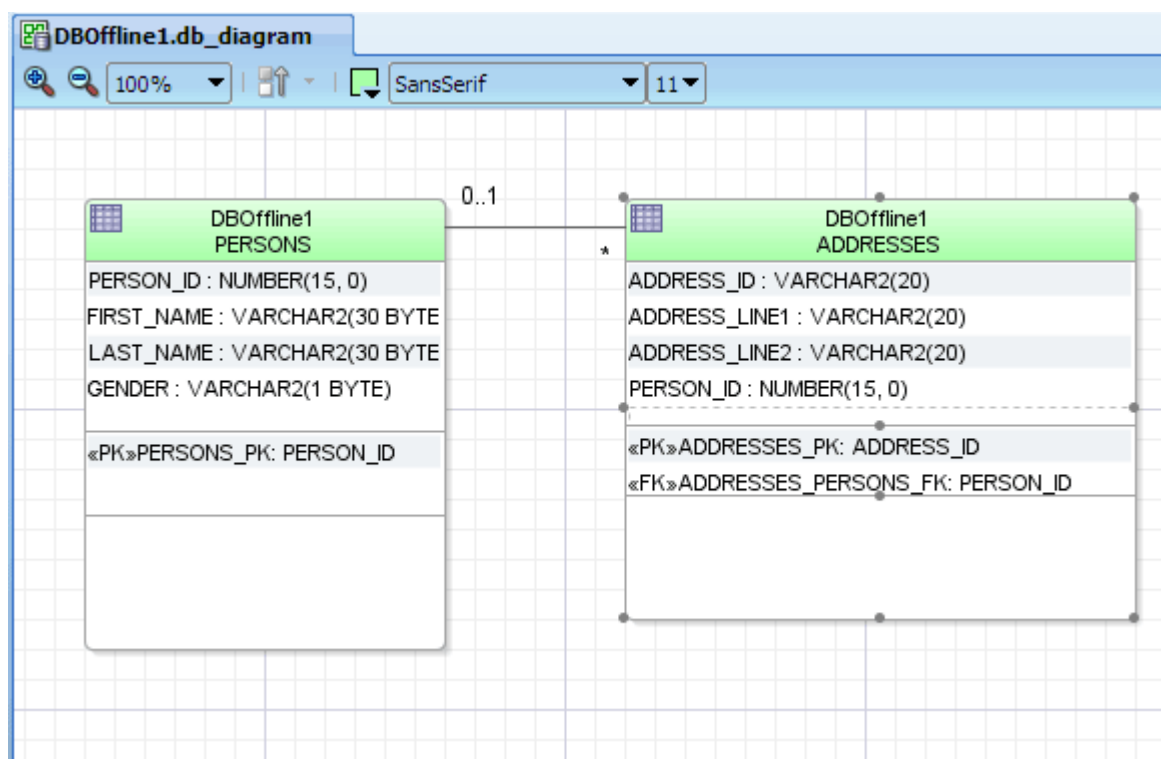
5. Первичный ключ будет добавлен к колонке PERSON_ID



6. Далее создадим еще одну таблицу, для связи с первой.

7. Из набора элементов управления для базы данных (Component Palette) выберите таблицу  (Table) и перетащите на форму.

8. Задайте таблице имя ADDRESSES
9. Создайте колонки ADDRESS_ID, ADDRESS_LINE1 and ADDRESS_LINE2 в таблице ADDRESSES.
10. Далее необходимо задать первичный ключ. Для этого дважды кликните на таблице и выберите меню редактирования (Edit Table dialog). Слева выберите Первичный ключ (Primary Key), справа будут отображены все доступные колонки. Выберите ADDRESS_ID и нажмите ➤
11. Далее из набора элементов управления для базы данных (Component Pallete) выберите 🗑 Foreign Key. После этого кликните сначала по таблице PERSONS, а потом по таблице ADDRESSES для того чтобы задать связь между таблицами.
12. Получится следующее



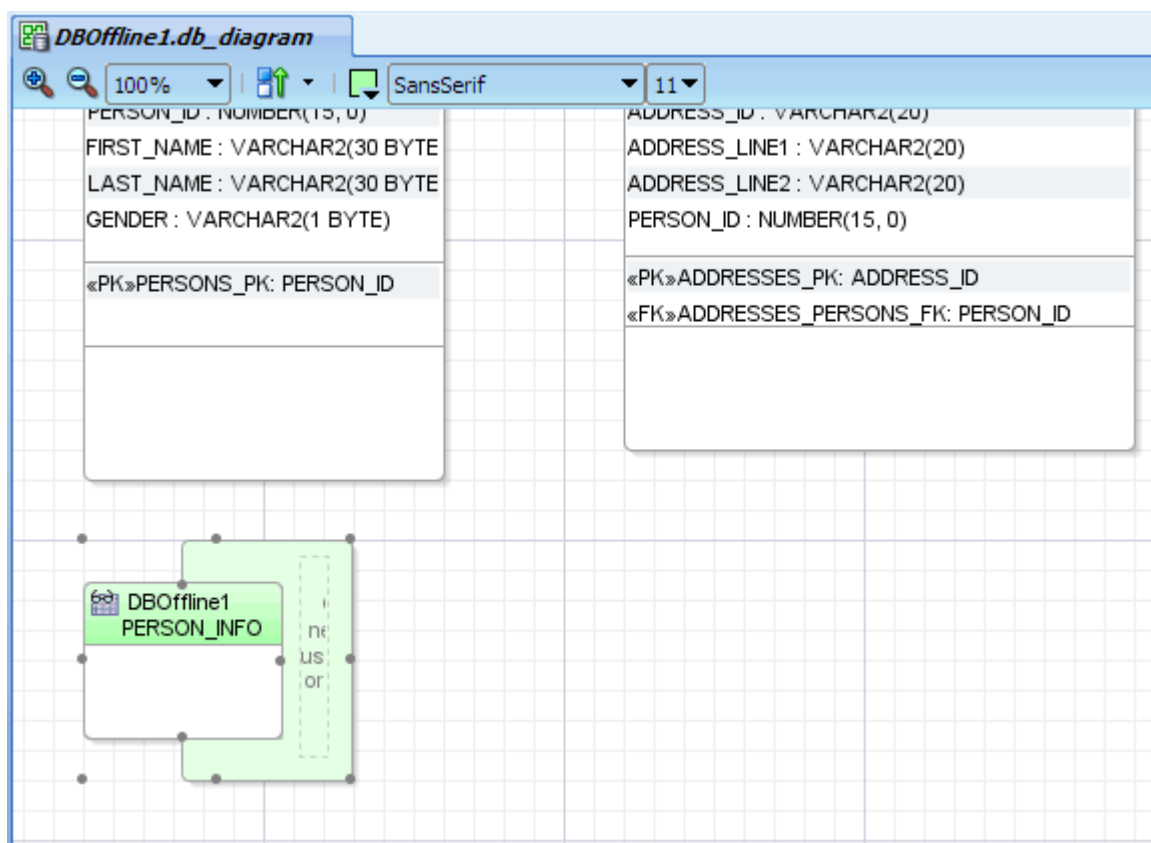
13. Нажмите 📁 Save (сохранить) чтобы сохранить.



Практическое занятие № 11.

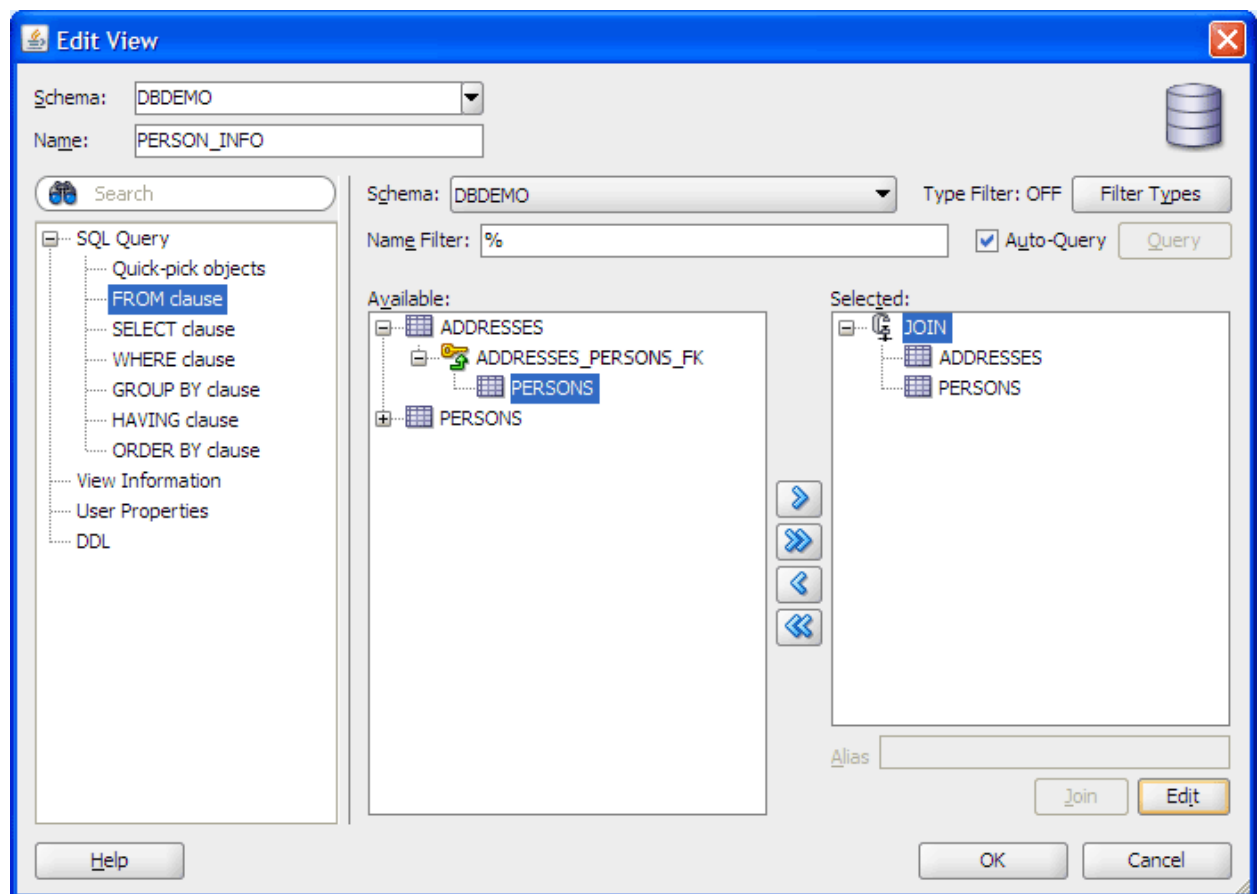
Создание виртуальной таблицы View (Обзор).

Обзор (View) – это виртуальная таблица, построенная на реальных таблицах. В них можно скомпоновать данные из реальных таблиц и представить данные в

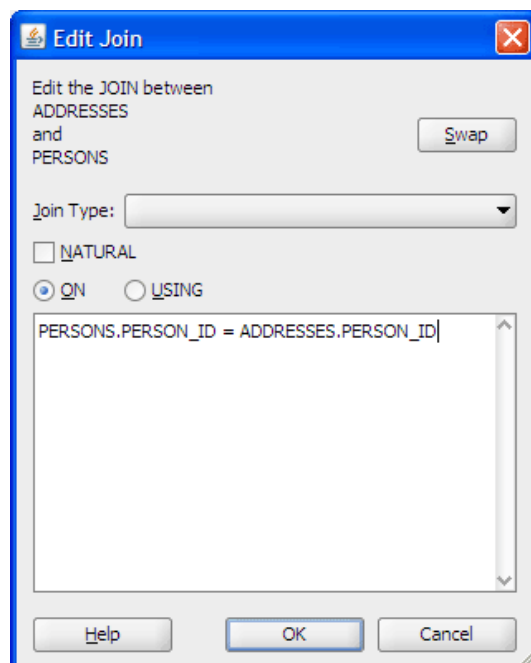
единой таблице. Благодаря, виртуальным таблицам разработчик выбирает из реальных таблиц то, что будет видеть конечный пользователь.



1. Из набора элементов управления для базы данных (Component Pallette) выберите Обзор (View)  и перетащите его на форму.
2. Задайте Обзор (View) имя PERSON_INFO.
3. Дважды кликните по View PERSON_INFO и вызовите меню редактирования (Edit View)
4. Раскройте ADDRESSES | ADDRESSES_PERSONS_FK в списке доступных таблиц и выберите PERSONS, нажмите , чтобы создать связь между ADDRESSES и PERSONS



5. Далее необходимо отредактировать первичный ключ. Для этого нажмите Редактировать (Edit). Выберите On и впишите в поле редактирования `PERSON.PERSON_ID = ADDRESSES.PERSON_ID`.



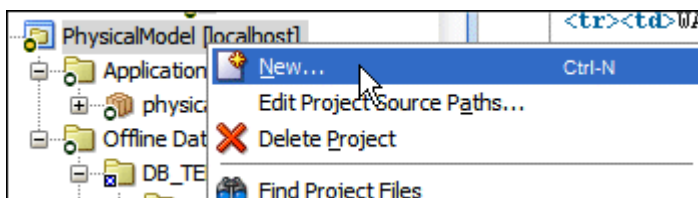
Нажмите  Save (сохранить) чтобы сохранить.

Практическое занятие № 12.

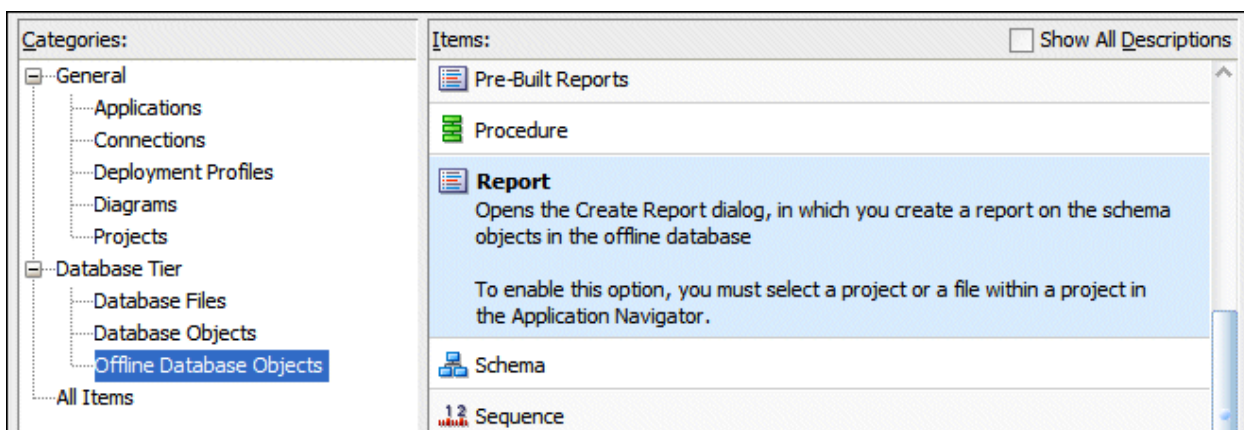
Создание отчета.

JDDeveloper предоставляет огромное количество стандартных отчетов. Но так же возможно создать собственный отчет.

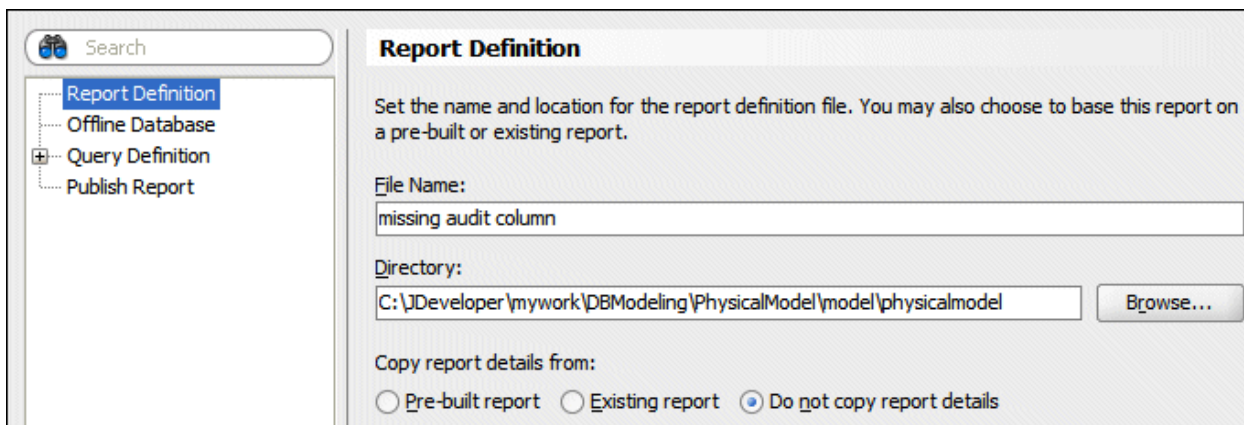
1. В Application Navigator (Обозреватель приложений), правой кнопкой мыши кликните на Физической модели (PhysicalModel) и выберите New (Новый) из контекстного меню



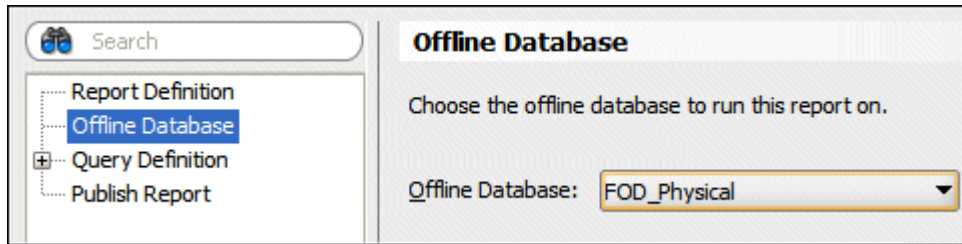
2. В меню New раскройте базы данных (Data base Tier) и выберите Оффлайновая база данных. В Items выберите Отчет (Report) и нажмите ОК



3. В окне создания отчета задайте отчету имя missing audit column (проверка отсутствующей колонки)



4. Выберите Оффлафновую базу данных



5. Кликните на Query Definition (формулирование запроса) и введите следующее:

```
SELECT  
T.NAME  
FROM  
DB_TABLES T  
WHERE  
NOT EXISTS (SELECT  
1  
FROM  
DB_COLUMNS C  
WHERE  
C.PARENT_ID = T.ID AND C.NAME = 'CREATION_DATE')
```

Этот запрос возвращает имя таблицы для каждой из таблиц, которая не содержит колонки CREATION_DATE. Кликните на Check Syntax (Проверка синтаксиса)

Query Definition

SQL Query:

```

SELECT
  T.NAME
FROM
  DB_TABLES T
WHERE
  NOT EXISTS (SELECT
    1
  FROM
    DB_COLUMNS C
  WHERE
    C.PARENT_ID = T.ID AND C.NAME = 'CREATION_DATE')

```

SQL Parse Results:
No errors found in SQL.

Revert

Check Syntax

Если все введено правильно, будет выдано сообщение No errors found in SQL (Ошибки не были найдены в SQL)

6. Раскройте Query Definition (формулирование запроса) чтобы увидеть все записи, которые могут быть добавлены в отчет.

Report Definition

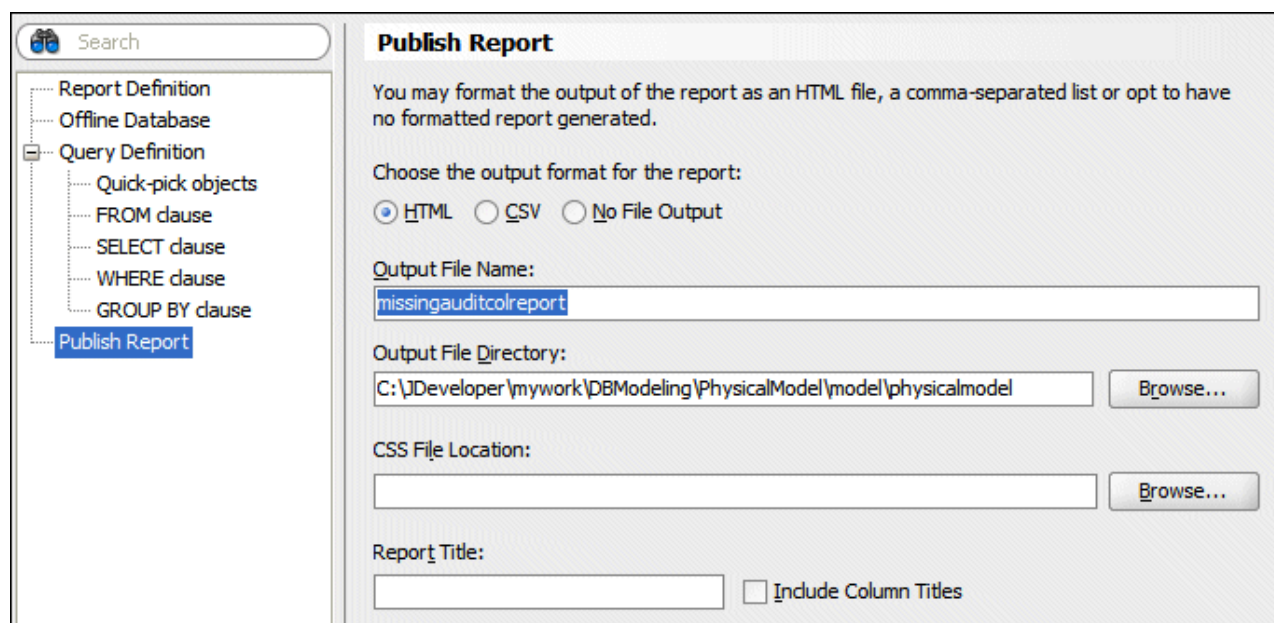
- Offline Database
- Query Definition
 - Quick-pick objects
 - FROM clause
 - SELECT clause
 - WHERE clause
 - GROUP BY clause
- Publish Report

Schema: OFFLINE_DB

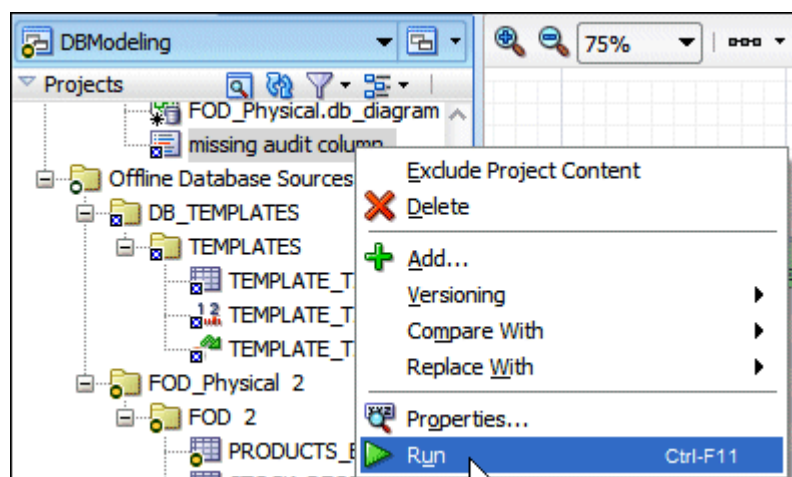
Name Filter: %

Available:	Selected:
DB_PKCONSTRAINTS	T.NAME
DB_PKCONSTRAINT_COLUMNS	
DB_PLSQLPARAMETERS	
DB_PROCEDURES	
DB_SCHEMAS	
DB_SEQUENCES	
DB_SYNONYMS	
DB_TABLES	
COMMENT	
ID	
NAME	
SCHEMA	
TABLE_TYPE	
DB_TABLESPACES	

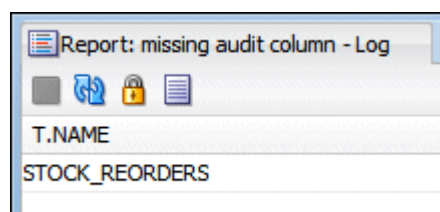
7. Выберите опубликовать отчет (Publish Report) по умолчанию он будет опубликован в HTML. Задайте сохранение в файл HTML и файлу задайте имя missingauditcolreport, все остальное оставьте по умолчанию и нажмите ОК



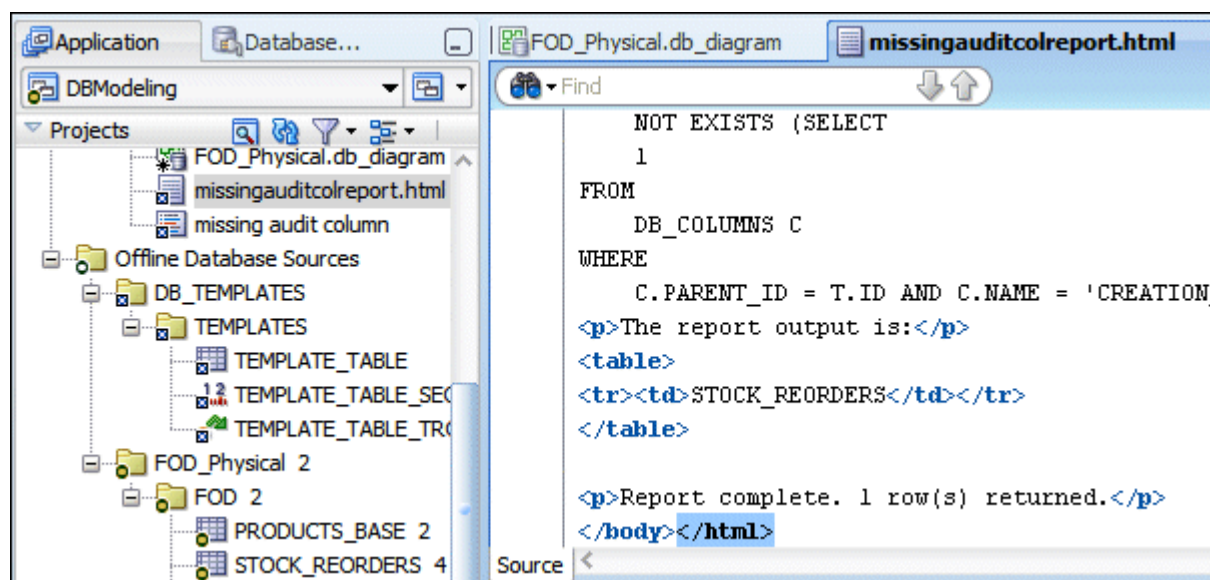
8. Новый отчет появится в Application Navigator (Обозреватель решений). Правой кнопкой щелкните на отчете и вызовите контекстное меню, в контекстном меню выберите Запустить (Run)



9. В следующем окне будут выведены результаты



10.Так как в качестве вывода отчета был выбран HTML, будет сгенерирована HTML версия отчета.



11.Нажмите  Save (сохранить) чтобы сохранить.

Oracle XML

Использовать БД Oracle для хранения XML данных можно было и ранее (чаще всего, для этого использовался тип CLOB, менее подходящий- VARCHAR2, имеющий ограничение по длине в 4000 символов). Однако, в 9 версии базы данных появился тип XMLTYPE, предназначенный специально для хранения данных XML. Рассмотрим, зачем же это было сделано. Сразу пример.

```
CREATE TABLE books
```

```
(id      NUMBER PRIMARY KEY  
, description XMLTYPE);
```

```
INSERT INTO books VALUES
```

```
(100
```

```
, XMLTYPE('<cover>  
    <title>Oracle SQL*Loader</title>  
    <author>Jonathan Gennick</author>  
    <author>Sanjay Mishra</author>  
    <pages>269</pages>  
</cover>'));
```

```
SET long 1000
```

```
SELECT id, description FROM books;
```

```
SELECT id, b.description.XMLDATA FROM books b;
```

XMLDATA – специально созданный для XMLTYPE «псевдостолбец».

XMLTYPE – тип XML

XMLTYPE дает возможность сообщить БД, что заносимый текст – это не просто строка, а строка документа XML. Следующая попытка приведет к ошибке:

```
INSERT INTO books VALUES (101, XMLTYPE('<cover><title></title>'));
```

С другой стороны, Oracle поймет правильно составленные директивы XML и встроенное в текст описание DTD:

INSERT INTO books VALUES

(101

```
, XMLTYPE('<?xml version="1.0"?>
  <!DOCTYPE cover [
    <!ELEMENT cover (title, author*, pages)>
    <!ELEMENT title (#PCDATA)>
    <!ELEMENT author (#PCDATA)>
    <!ELEMENT pages (#PCDATA)>
  ]>
  <cover>
    <title>SQL*Plus Pocket Reference</title>
    <author>Jonathan Gennick</author>
    <pages>94</pages>
  </cover>'));
```

Oracle соотносит описание DTD самому тексту документа.

Для выборки можно использовать специально придуманные для XMLTYPE функции. Так, функция EXTRACTVALUE извлекает значения элемента из документа XML:

```
SELECT id, EXTRACTVALUE(description, '/cover/title')
FROM books;
```

Функция EXISTSNODE дает возможность использовать в SQL условие отбора XPath (язык отбора, принятый в технологиях XML):

```
SELECT id, b.description.XMLDATA
FROM books b
WHERE b.description.EXISTSNODE('/cover[author="Sanjay Mishra"]')=1;
```

Доказательством утверждения в заголовке служит создание следующей таблицы объектов типа XMLTYPE, «таблицы документов XML»:

```
CREATE TABLE xbooks OF XMLTYPE;
```

Работать с ними можно, как и с XML-атрибутом в обычной таблице:

```
INSERT INTO xbooks VALUES
```

```
(XMLTYPE('<cover>
  <title>Oracle SQL*Loader</title>
  <author>Jonathan Gennick</author>
  <author>Sanjay Mishra</author>
  <pages>269</pages>
</cover>'));
```

```
INSERT INTO xbooks VALUES
```

```
(NEW XMLTYPE('<?xml version="1.0"?>
  <cover>
    <title>SQL*Plus Pocket Reference</title>
    <author>Jonathan Gennick</author>
    <pages>94</pages>
  </cover>'));
```

В первом случае объект XML создается с помощью конструктора, а во втором, к тому же, используется оператор NEW. Последний применяется в Oracle для работы с объектами, однако его использование носит лишь рекомендательный характер, так как в SQL он ничего содержательного не дает.

Далее:

```
SELECT * FROM xbooks;
SELECT VALUE(x) FROM xbooks x;
SELECT XMLDATA FROM xbooks;
```

Так же как для таблиц объектов прочих типов, элементы таблицы объектов XML имеют ссылки, то есть позволяют ссылаться на себя через REF в других типах и таблицах:

```
SELECT REF(x) FROM xbooks x;
SELECT Deref(REF(x)) FROM xbooks x;
```

У этого типа нет свойств, но есть методы. В этом можно убедиться, сделав запрос от имени SYS:

```

COLUMN text FORMAT A80
SELECT text
FROM user_source
WHERE name ='XMLTYPE' AND type='TYPE'
ORDER BY line;

```

Исследование каталога rdbms/admin позволяет обнаружить и исходное описание этого типа (но не его тела !) в файле dbmsxmlt.sql. К сожалению в документации описания этих методов разбросаны по разным местам, не всегда последовательны и ясны. Так например, EXTRACT и EXISTSNode (о последней речь шла выше), возведены в ранг функций SQL, то есть описаны в документации по SQL в разделе «Функции», в то время как из предыдущего запроса к словарию-справочнику следует, что это методы. О том же говорит синтаксис употребления. Для EXISTSNode пример уже приводился, а для EXTRACT он может выглядеть так:

```

SELECT b.description.EXTRACT('/cover/title') FROM books b;

```

(Сравните с примером использования функции EXTRACTVALUE выше).

Вот некоторые другие примеры методов XMLTYPE:

```

SELECT b.description.GETCLOBVAL() FROM books b;
SELECT b.description.GETSTRINGVAL() FROM books b;
SELECT b.description.GETROOTELEMENT() FROM books b;

```

Обратите внимание, что некоторые методы XMLTYPE, например TOOBJECT, могут использоваться только процедурно, так как сами исполнены в виде процедур, а не функций.

Объектность типа XMLTYPE реализована не в полной степени. Так, попытка создать в таблице столбец из коллекции документов XML (вложенной таблицы или массива VARRAY) терпит неудачу. Это относится только к БД; в PL/SQL этих проблем не возникает:

```

SQL> declare type xml_nt is table of xmltype index by varchar2(10);
2  begin null; end;
3 /

```

PL/SQL procedure successfully completed.

Взаимные преобразования табличного вида и XMLTYPE

Связь двух форм описания данных – табличной и XML – достигается не одной только возможностью создавать в таблицах столбец типа XMLTYPE. Возможно преобразование данных из одного вида в другой, благодаря чему исходный формат хранения данных может оказаться не столь существенен.

Преобразование из XMLTYPE в табличную форму

Для преобразования данных типа XMLTYPE в обычный табличный вид можно использовать функции SQL и методы XMLTYPE, в первую очередь упоминавшуюся метод-функцию EXTRACT:

```
COLUMN xdoc FORMAT A80
```

```
SELECT ROWNUM, id, b.description.EXTRACT('/cover/author') xdoc  
FROM books b;
```

Обратите внимание на возможность и способ обработки нескольких авторов в XML элементах <author>.

Использование функции SQL EXTRACTVALUE, в свою очередь, оставляет возможность отбора не более одного элемента XML для формирования каждой строки результата SELECT, но зато безболезненно убирает обрамляющие значение элемента XML метки:

```
SELECT id, EXTRACTVALUE(b.description.EXTRACT('/cover/title'), '/title')  
xdoc  
FROM books b;
```

То же самое можно записать проще, что уже демонстрировалось в начале главы.

Преобразование из табличной формы в XMLTYPE

Для обратного преобразования удобно воспользоваться функциями, объединенными в стандарте SQL:2003 названием SQL/XML (другое название – SQLX). В Oracle реализованы следующие (не все) функции из этого стандартного набора:

- XMLElement
- XMLAttributes
- XMLAgg
- XMLConcat
- XMLForest

Вот некоторые примеры использования:

```
SELECT XMLEMENT("Employee", ename) FROM emp;
```

```
SELECT XMLEMENT("Employee",  
    XMLATTRIBUTES(ename AS "Name", empno AS "Number"))  
FROM emp;
```

Обратите внимание, что в результатах выдаются поля типа XMLTYPE:

```
CREATE TABLE xtable (n) AS SELECT XMLEMENT("Name", ename) FROM  
emp;  
DESCRIBE xtable
```

Следующий пример – агрегирующей функции XMLAGG, допускающей использование в запросах с группировкой GROUP BY, подобно тому, как агрегирующие функции MIN, AVG и другие применяются для обычных данных, а не XMLTYPE:

```
SET LONG 2000  
SELECT XMLEMENT("department", XMLATTRIBUTES(deptno AS "no")),  
    XMLAGG(XMLEMENT("employee", ename))  
FROM emp  
GROUP BY deptno;
```

Интересно, что последний запрос допускает создания на своей основе выводимой таблицы, но не базовой:

```
CREATE VIEW xview (a, b) AS  
SELECT XMLEMENT("department", XMLATTRIBUTES(deptno AS "no")),  
    XMLAGG(XMLEMENT("employee", ename))  
FROM emp  
GROUP BY deptno;
```

(срабатывает)

CREATE TABLE xtable (a, b) AS

```
SELECT XMLELEMENT("department", XMLATTRIBUTES(deptno AS "no")),  
       XMLAGG(XMLELEMENT("employee", ename))  
FROM emp  
GROUP BY deptno;  
(ошибка !)
```

Это объясняется тем, что столбцы А и В в обоих случаях Oracle пытается создавать как XMLTYPE, а наши данные таковы, что в столбце В содержатся строго говоря некорректные строки XML.

Oracle Data Mining

Oracle Data Mining (ODM) включает поиск данных в пределах базы данных Oracle. Алгоритмы ODM воздействуют на отношения таблиц в базе данных Oracle, таким образом, избавляя от необходимости использовать для обработки базы данных отдельные программы. Интегрированная архитектура ODM приводит к более простой, более надежной, и более эффективной обработке данных. Задачи поиска данных могут быть запущены асинхронно и независимо от любого конкретного пользовательского интерфейса как часть стандартной базы данных, обрабатывающей запросы.

Функции Data Mining

Функции Data Mining могут быть разделены на две категории: контролируемые (направленные) и неконтролируемые (ненаправленные). Контролируемые функции используются для того чтобы прогнозировать значение; они требуют детализации цели (известный результат). Цели - это либо двоякие поля, указывающие на ответ да/нет (покупать/не покупать, делать/не делать и т.д.), либо мультикласс предназначенный для предпочтительной альтернативы (цвет свитера, вероятный диапазона зарплаты, и т.д.).

Неконтролируемые функции используются, чтобы найти собственную структуру, отношение, или сходства в данных.

Data mining также можно разделить на прогнозирующий или наглядный. Прогнозирующий поиск данных строит одну или более моделей; эти модели используются, для того чтобы предсказать результаты для новой совокупности данных. Прогнозирующие функции поиска данных это классификация и регрессия. Наглядный поиск данных описывает совокупность данных кратким способом и представляет интересные особенности данных. Наглядные функции Data Mining это кластеризация, ассоциативные модели и дифференциальная выборка.

Различные алгоритмы удовлетворяют различным целям; у каждого алгоритма есть преимущества и недостатки.

Oracle Data Mining поддерживает следующие функции data mining:

1. Контролируемый data mining:

1. 1 Классификация: Группировка элементов в дискретные классы и прогнозирование какому классу какой элемент принадлежит
1. 2 Регрессия: Приближение и прогноз непрерывных значений
1. 3 Важность Признака: Распознавание признаков, которые являются самыми важными в прогнозировании результатов
1. 4 Обнаружение Аномалии: Распознавание данных, которые не соответствуют особенностям "нормальных" данных (выбросы - резко выделяющиеся значения)

2. Неконтролируемый data mining:

2. 1 Кластеризация: Обнаружение естественных группировок в данных
- 2.2 Модели ассоциации: Анализ "корзины рынка"
- 2.3 Дифференциальная выборка: Создание новых свойств из комбинации начальных свойств.

Данные и поля

Данные, используемые Oracle Data Mining, состоят из таблиц хранящихся в базе данных Oracle. И обычные таблицы, и вложенные таблицы могут

использоваться как входные данные. Данные, используемые в операции data mining, часто называют набором данных.

У данных есть физическая организация и логическая интерпретация. Названия колонок это физическая организация;

Строки таблиц данных часто называют записями. Колонки таблиц данных называют атрибутами или полями; каждое поле в записи содержит ячейку информации. Названия полей являются постоянными от записи к записи для невложенных таблиц; значения полей могут меняться от записи к записи. Например, у каждой записи может быть маркированное поле "годовой доход." Значение в поле годового дохода может измениться от одной записи к другой.

Oracle Data Mining различает два типа полей: категориальный и числовой. Категориальные поля - те, которые определяют их значения как принадлежность к небольшому количеству дискретных классов; нет никакого неявного порядка, связанного с их значениями. Если есть только два возможных значения, например, да и нет, или мужчина или женщина, поле является двойным. Если есть больше чем два возможных значения, например, маленькие, средние, большие, поле является мультиклассом.

Числовые поля - поля, содержащие числа, которые имеют свой порядок. Для числовых полей также существуют различия между значениями. Например, годовой доход может теоретически быть любым значением от нуля до бесконечности, хотя на практике годовой доход имеет ограниченный диапазон и имеет конечное значение.

Вы можете преобразовывать числовые поля в категориальные. Например, годовой доход может быть разделен на три категории: низкий, средний, высокий. И наоборот, можно раскрыть категориальные значения, чтобы преобразовать их в числовые.

Классификация и алгоритмы регрессии требуют целевого поля. Контролируемая модель может предсказать единственное целевое поле. Целевое поле для всех алгоритмов классификации может быть числовым или

категориальным. Алгоритм регрессии Oracle Data Mining поддерживает только числовые целевые поля.

Определенные алгоритмы Oracle Data Mining поддерживают неструктурированные текстовые поля. Хотя неструктурированные данные могут включать так же изображения, аудио, видео данные, Oracle Data Mining поддерживает только текстовые данные. Входная таблица может содержать одну или более текстовых колонок.

Требования данных

Oracle Data Mining поддерживает несколько типов входных данных, в зависимости от формата таблицы данных, типа данных колонок, и типа данных полей.

Формат колонок данных поддерживаемый Oracle Data Mining

Данные Oracle Data Mining должны находиться в единственной таблице базы данных Oracle. Таблица должна быть стандартной и иметь стандартные отношения с другими таблицами, где каждый прецедент представлен одной строкой в таблице, соответствующей полю в колонке таблицы. Колонки должны иметь один из типов, поддерживаемых Oracle Data Mining.

Типы данных колонок, поддерживаемых Oracle Data Mining

Oracle Data Mining не поддерживает все типы данных, которые поддерживает Oracle. У каждой колонки в наборе данных, используемом Oracle Data Mining, должен быть один из следующих типов данных:

INTEGER

NUMBER

FLOAT

VARCHAR2

CHAR

DM_NESTED_NUMERICALS (вложенная колонка)

DM_NESTED_CATEGORICALS (вложенная колонка)

У поддерживаемых типов данных полей есть тип поля по умолчанию (категориальный или числовой).

Контролируемый Data Mining

Классификация

Классификация набора данных состоит из деления элементов, которые составляют набор данных на категории или классы. В контексте data mining классификация делается, с использованием модели, которая основана на исторических данных. Цель прогнозирующей классификации состоит в том, чтобы точно предсказать целевой класс для каждой записи в новых данных, то есть, для данных, которые не находятся в исторических данных.

Задача классификации начинается с построения данных (также известный как обучение данных), для которых значение цели известны. Различные алгоритмы классификации используют различные методы для того, чтобы найти отношения между предсказанными признаками значений и целевыми значениями признака в строящихся данных. Эти отношения суммируются в модели; модель может тогда быть применена к новым случаям с неизвестными целевыми значениями, чтобы предсказать их. Модель классификации может также быть применена к данным, которые держали в стороне от обучаемых данных, для того чтобы сравнить прогнозы с известными целевыми значениями; такие данные также известны как испытательные данные или данные оценки. Технику сравнения называют тестом модели, которая измеряет прогнозирующую точность модели.

Классификация, в качестве целей использует двойные значения или мультиклассовые.

Регрессия

Модели регрессии подобны моделям классификации. Различие между регрессией и классификацией в том, что регрессия имеет дело с числовыми или непрерывными целевыми полями, тогда как классификация имеет дело с дискретными или категориальными целевыми полями. Другими словами, если целевое поле содержит непрерывные значения (с плавающей точкой) или

целочисленные значения, у которых есть свой порядок, регрессия может использоваться. Если целевое поле содержит категориальное значение, то есть, последовательность значений или значение целого числа, где порядка нет, необходимо использовать классификацию. Однако одни данные могут быть преобразованы в другие, что значит, что вместо регрессии можно использовать классификацию.

Значимое поле

Значимое поле обеспечивает автоматизированное получение решения с большей скоростью и, возможно, более высокой точностью для моделей классификации основываясь на таблицах данных с большим количеством полей.

Время, требуемое для построения модели классификации Oracle Data Mining, увеличивается с числом полей. Значимое поле идентифицирует характерное подмножество полей, которые имеют наибольшее отношение к прогнозированию цели. Модель построения может продолжать свой процесс, используя только отобранные поля.

Использование меньшего количества полей не обязательно приводит к потере точности прогнозирования. Использование слишком большого количества полей (особенно тех, которые являются "шумом") может затронуть модель и ухудшить ее работу и точность. Использование наименьшего количества полей может существенно сэкономить время вычисления и позволить построить более точные модели.

Программный интерфейс для значимого поля позволяет пользователю определять число или процент используемых полей.

Анализ отклонений

Анализ отклонений состоит из обнаружения нестандартных или аномальных образцов. Модель обнаружения аномалии предсказывает, типичен ли пункт данных для данного распределения или нет. Анализ отклонений модели прогнозирует, типичны ли данные для модели или нет. Нетипичные данные могут быть либо выбросом, либо примером ранее невидимого класса.

Модель обнаружения аномалии обнаруживает данные, которые не соответствуют распределению данных.

Неконтролируемый Data Mining

Кластеризация

Кластеризация полезна для исследования данных, когда во многих случаях невозможно определить естественные группы данных, алгоритм кластеризации может помочь найти естественные группы данных.

Кластеризация находит группы, вложенные в данные. Группа - набор объектов данных, которые в некотором смысле подобны друг другу. Хорошая кластеризация производит высококачественные группы, гарантируя, что объединение будет верным, а подобие групп высоким; другими словами, члены одной группы больше схожи между собой и не похожи на членов других групп. Кластеризация может также служить полезным шагом при предварительной обработке данных, для нахождения однородных групп для построения контролируемой модели. В этом модели кластеризации отличаются от контролируемых моделей, результат процесса не находится под контролем известного результата, то есть, нет никакого целевого поля. Для того чтобы руководить построением модели, контролируемые модели прогнозируют значения для целевого поля и частоту ошибок между целью и прогнозируемым значением. С другой стороны, модели кластеризации, построены, с использованием критериев оптимизации, которые обеспечивают высокое соответствие элементов внутри группы и низкое между группами. Модель может использоваться, чтобы назначить идентификаторы группам данных.

Например, следующее правило в Oracle Data Mining заставит группировать значение с определенным ростом и возрастом в группу 10:

```
If AGE >= 25 and AGE <= 40 and HEIGHT >= 5.0ft and HEIGHT <= 5.5ft then  
CLUSTER = 10
```

Ассоциативная модель

Ассоциативная модель часто используется для анализа корзины рынка, которая пытается обнаружить отношения в ряде элементов. Анализ корзины рынка

широко используется в анализе данных для прямого маркетинга, проекта каталога, и других деловых процессов принятия решений.

Процесс Data Mining

Oracle Data Mining объединяет data mining с базой данных Oracle и экспонирует data mining через следующие интерфейсы:

Интерфейс Java: Java Data Mining (JSR-73) интерфейс, который позволяет пользователям включать data mining в приложения Java.

Интерфейс PL/SQL: пакеты DBMS_DATA_MINING и DBMS_DATA_MINING_TRANSFORM позволяют пользователям включать data mining в приложения PL/SQL.

Автоматизированный data mining: DBMS_PREDICTIVE_ANALYTICS PL/SQL пакет, автоматизирует весь процесс data mining от предварительной обработки данных до конца.

DBMS_PREDICTIVE_ANALYTICS обеспечивает следующие функциональные возможности:

EXPLAIN - Располагает признаки в порядке влияния на целевую колонку

PREDICT - Прогнозирует ценность признака

Data mining функции SQL: функции Data Mining SQL (CLUSTER_ID, CLUSTER_PROBABILITY, CLUSTER_SET, FEATURE_ID, FEATURE_SET, FEATURE_VALUE, ПРЕДСКАЗАНИЕ, PREDICTION_COST, PREDICTION_DETAILS, PREDICTION_PROBABILITY, и PREDICTION_SET)

Литература

1. ORACLE 9 (tm) Server. Administrator's Guide. Руководство Администратора: Oracle (R) / 2002.
2. ORACLE 9 (tm) Server. Application Developer's Guide. Руководство разработчика приложений: Oracle (R) / 2002.
3. ORACLE 9 (tm) Server. Server Concepts Manual. Руководство по концепциям сервера: Oracle (R) / 2002.
4. Смирнов С.Н. Работаем с ORACLE. М.: Гелиос, 1998.
5. Анзер Г. ORACLE POWER OBJECTS. Визуальное проектирование приложений клиент/сервер для реляционных баз данных. М.: АБФ, 1997.
6. PL/SQL (tm). User's Guide and Reference. Руководство пользователя и справочник: Oracle (R) / 2002.
7. Смирнов С.Н. Работаем с ORACLE. М.: Гелиос, 1998.
8. <http://www.oracle.ru>
9. http://www.citforum.ru/seminars/cbd2003/1_01_rivkin.shtml
10. <http://www.interface.ru/home.asp?artId=2388>
11. <http://www.citforum.ru/database/articles/oracleov.shtml>
12. http://www.omega.ru/notes/note_galatenko.html

Алла Григорьевна Кравец

Разработка баз данных в СУБД ORACLE

Учебное пособие.

Редактор Е.И. Кагальницкая

Темплан 2013 г. Поз. № 48.

Лицензия ЛР № 020251 от 16.04.1996 г.

Подписано в печать . Формат 64х80 1/16. Бумага газетная.

Печать офсетная. Усл. печ. л. . Уч. – изд. л. . Тираж экз. Заказ

Волгоградский государственный технический университет.

400131, Волгоград, пр. Ленина, 28.

РПК «Политехник» ВолгГТУ

400131, Волгоград, ул. Советская, 35.