БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

А. В. Кузьмина

ТЕХНОЛОГИЯ БИЗНЕС-АНАЛИТИКИ В СРЕДЕ ORACLE

Рекомендовано
Учебно-методическим объединением
по естественно-научному образованию в качестве
учебно-методического пособия для студентов
учреждений высшего образования, обучающихся
по специальности 1-31 03 04 «Информатика»

МИНСК
БГУ
2019

УДК 005.5:004.9(075)
ББК 65.291.216с51я7
К89

Рецензенты:
кафедра программного обеспечения информационных технологий
Белорусского государственного университета информатики
и радиоэлектроники (заведующий кафедрой
кандидат технических наук, доцент *Н. В. Лапицкая*);
доктор технических наук, профессор *А. А. Дудкин*

Кузьмина, А. В.

К89 Технология бизнес-аналитики в среде Oracle : учеб.-метод. пособие / А. В. Кузьмина. – Минск : БГУ, 2019. – 96 с.
ISBN 978-985-566-727-9.

Изложены вопросы применения инструментов языка SQL для проведения бизнес-аналитики, эволюции продуктов бизнес-аналитики Oracle. На примере продукта Oracle Business Intelligence Suite Enterprise Edition рассмотрено создание репозитория Oracle Business Intelligence, аналитических отчетов, информационных панелей и других составляющих бизнес-аналитики.

УДК 005.5:004.9(075)
ББК 65.291.216с51я7

Учебное издание

Кузьмина Анна Викентьевна

ТЕХНОЛОГИЯ БИЗНЕС-АНАЛИТИКИ В СРЕДЕ ORACLE

Учебно-методическое пособие

Ответственный за выпуск *О. С. Гладкова*. Художник обложки *Т. Ю. Таран*.
Технический редактор *Л. В. Жаборовская*. Компьютерная верстка *В. Н. Васиной*.
Корректор *О. С. Гладкова*

Подписано в печать 29.05.2019. Формат 60×84/16. Бумага офсетная.
Ризография. Усл. печ. л. 5,58. Уч.-изд. л. 6,3. Тираж 50 экз. Заказ 391.

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/270 от 03.04.2014.
Пр. Независимости, 4, 220030, Минск.

Республиканское унитарное предприятие
«Издательский центр Белорусского государственного университета».
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 2/63 от 19.03.2014.
Ул. Красноармейская, 6, 220030, Минск.

ISBN 978-985-566-727-9

ВВЕДЕНИЕ

Термин «бизнес-аналитика» (Business Intelligence, BI) был введен аналитиками Gartner[1] в конце 1980-х гг. и означал направленный на бизнес-пользователя процесс, содержащий доступ к информации, ее исследование, анализ, развитие интуиции и понимания, которые ведут к улучшенному и нестандартному принятию решений.

В настоящее время бизнес-аналитика в широком смысле слова трактуется как:

- процесс превращения данных в информацию и знания о бизнесе для поддержки принятия улучшенных и нестандартных решений;
- информационные технологии (методы и средства) сбора данных, консолидации информации и обеспечения доступа бизнес-пользователей к знаниям;
- знания о бизнесе, добытые в результате углубленного анализа детальных данных и консолидированной информации.

BI-технологии позволяют анализировать большие объемы информации, акцентируя внимание пользователей на ключевых факторах эффективности, моделируя исход различных вариантов действий, отслеживая результаты принятия тех или иных решений. BI поддерживается данными из хранилищ, методами разработки данных, технологиями поддержки принятия решений.

В соответствии с подходом аналитиков Gartner выделяют три основных типа инструментальных средств бизнес-аналитики.

1. Средства предоставления информации.
   1.1. Средства создания отчетов.
   1.2. Информационные панели показателей.
   1.3. Генератор нерегламентированных запросов.
   1.4. Интеграция с Microsoft Office.
2. Средства интеграции.
   2.1. Общая BI-инфраструктура.
   2.2. Управление метаданными.
   2.3. Средства разработки.
   2.4. Совместная работа и управление рабочими процессами.

[1] Gartner является исследовательской и консалтинговой компанией, специализирующейся на рынках информационных технологий. Основана в 1979 г.

3. Средства анализа.

3.1. Оперативная аналитическая обработка данных (OLAP).

3.2. Продвинутая визуализация.

3.3. Предиктивное моделирование.

3.4. Интеллектуальный анализ данных.

3.5. Карты показателей.

Рассмотрим средства представления информации.

Средства создания отчетов. Предоставляют возможность создавать форматированные интерактивные отчеты. В дополнение к этому поставщики BI-платформ должны предоставлять широкий набор типов отчетов (финансовых, операционных и т. п.) в виде информационных панелей показателей (dashboards).

Информационные панели показателей. Являются одной из составных частей отчетов; представляют информацию в виде интуитивно понятного графического изображения, включая диаграммы, круговые шкалы, светофоры и т. п.

Генератор нерегламентированных запросов. Выступает в качестве функции создания отчетов в режиме самообслуживания, дает пользователям возможность получать ответы на возникающие вопросы. Система предоставляет средства навигации по доступным ресурсам данных.

Интеграция с Microsoft Office. В ряде случаев BI-платформы используются как промежуточное звено в цепочке анализа информации, а Microsoft Office (в частности, Excel) выступает как BI-клиент. В этих случаях очень важно, чтобы BI-вендор обеспечивал интеграцию с Microsoft Office, включая поддержку форматов документов, формул и сводных таблиц.

Охарактеризуем средства интеграции.

Общая BI-инфраструктура. Все инструменты платформы должны использовать одни и те же средства обеспечения безопасности, общие метаданные, общие средства администрирования, общие средства генерации запросов, а также иметь однотипные интерфейсы.

Управление метаданными. Все инструменты приложения должны основываться на одних и тех же метаданных, обеспечивать быстрый поиск, хранение, использование и публикацию таких объектов метаданных, как размерности, иерархии, параметры оценки производительности и оформления отчетов. Метаданные – это «данные о данных», предоставляющие в распоряжение пользователей объяснение характера данных, источника их происхождения и способов доступа к ним.

Средства разработки. Помимо средств создания отдельных BI-приложений, BI-платформа должна предоставлять средства программной разработки для интеграции приложений в общий бизнес-процесс или обеспечивать их встраивание в другое приложение. BI-платформа должна давать разработчикам возможность создания BI-приложений без кодирования на основе применения мастеров для визуального редактирования.

Совместная работа и управление рабочими процессами. Позволяет BI-пользователям разделять информацию и обсуждать ее с помощью общих папок. В дополнение BI-приложения могут назначать и отслеживать события или задачи, возложенные на отдельных пользователей, на основе заранее определенных бизнес-правил.

Рассмотрим средства анализа.

OLAP (Online Analytical Processing). Представляет собой класс приложений и технологий, предназначенных для сбора, хранения и анализа многомерных данных в целях поддержки принятия управленческих решений. Технология OLAP позволяет аналитикам, менеджерам и управляющим сформировать собственное видение данных, используя быстрый, единообразный, оперативный доступ к различным формам предоставления информации.

Одновременный анализ по нескольким измерениям определяется как многомерный анализ. Каждое измерение включает направления консолидации данных, состоящие из последовательных уровней обобщения, где каждый вышестоящий уровень соответствует большей степени агрегации данных по соответствующему измерению.

OLAP – инструмент для анализа больших объемов данных в режиме реального времени, обеспечивающий следующие возможности работы с многомерными данными:

- гибкий просмотр информации;
- произвольные срезы данных;
- детализация;
- свертка или консолидация;
- вращение;
- сравнение во времени.

В ячейках многомерного куба помещаются числовые параметры, предназначенные для анализа, например, объемов продаж. Измерениями OLAP-куба могут служить такие параметры, как время, продукция, регионы, продавцы. Продажи по времени в консолидированном виде могут отображаться по годам, при детализации – по кварталам, месяцам и дням.

Продвинутая визуализация. Инструменты продвинутой визуализации позволяют представлять данные для более эффективного их восприятия посредством использования интерактивных картинок и диаграмм вместо таблиц. Обычно пользователи в динамическом режиме могут менять графическое представление, использовать масштабирование, объединять данные, изменять цвета.

Предиктивное моделирование. Является процессом создания (или выбора) модели для прогнозирования вероятности наступления некоторого события.

Интеллектуальный анализ данных. Представляет собой компьютерную технику извлечения знаний, которая использует искусственный интеллект для распознавания образов и выделения значимых закономерностей из данных, находящихся в хранилищах или входных/выходных потоках. Эти методы основываются на статистическом моделировании, нейронных сетях, генетических алгоритмах и др. Информация, найденная в процессе использования методов Data Mining, должна описывать новые связи между свойствами, предсказывать значения одних признаков на основе других и т. д. Задачи, решаемые методами интеллектуального анализа данных, включают:

- классификацию – отнесение объектов (наблюдений, событий) к одному из заранее известных классов;
- регрессию, в том числе задачи прогнозирования; установление зависимости непрерывных выходных от входных переменных;
- кластеризацию – группировку объектов (наблюдений, событий) на основе данных (свойств), описывающих сущность этих объектов. Объекты внутри кластера должны быть похожими друг на друга и отличаться от объектов, входящих в другие кластеры. Чем больше похожи объекты внутри кластера и чем больше различий между кластерами, тем точнее кластеризация;
- ассоциацию – выявление закономерностей между связанными событиями. Примером такой закономерности X следует событие Y. Такие правила называются ассоциативными;
- последовательные шаблоны – установление закономерностей между связанными во времени событиями, т. е. обнаружение зависимости, согласно которой если произойдет событие X, то спустя заданное время произойдет событие Y;
- анализ отклонений – выявление наиболее нехарактерных шаблонов.

Карты показателей. Используют контрольные показатели, отображаемые на информационной панели, для более глубокого анали-

за путем наложения их на некоторую стратегическую карту, которая увязывает ключевые параметры производительности со стратегическими задачами.

Источниками информации для систем BI являются хранилища данных. *Хранилище данных* (*Data Warehouse*) – система, работающая по принципу центрального склада. Хранилища обычно содержат многолетние версии обычной базы данных (БД), физически размещаемые в той же самой базе. Когда все данные содержатся в едином хранилище, изучение и анализ связей между отдельными элементами может быть более плодотворным.

Хранилище данных – это очень большая предметно-ориентированная информационная корпоративная база данных, специально разработанная и предназначенная для подготовки отчетов, анализа бизнес-процессов в целях поддержки принятия решений в организации. Строится на базе клиент-серверной архитектуры, реляционной системы управления базами данных (СУБД) и утилит поддержки принятия решений. Данные, поступающие в хранилище данных, доступны только для чтения.

Данные из промышленной OLTP[2]-системы копируются в хранилище данных таким образом, чтобы при построении отчетов и OLAP-анализе не использовались ресурсы промышленной системы и не нарушалась ее стабильность. Данные загружаются в хранилище с определенной периодичностью, поэтому актуальность данных несколько отстает от OLTP-системы.

Данные собираются (консолидируются) из внутренних и внешних источников (учетных систем, баз данных, файлов и т. п.) и интегрируются в хранилище данных. При консолидации осуществляется трансформация данных (реструктуризация, согласование, очистка и/или агрегирование) в процессе их передачи от первичных систем в хранилище данных. В среде хранилищ данных используется технология поддержки консолидации ETL (извлечение, преобразование и загрузка).

Хранилища содержат не первоначальные данные, а определенным образом обработанную информацию в соответствии с заданными форматами и структурой, прошедшую процедуру фильтрации и другие необходимые процедуры. Структура данных в хранилище должна быть

[2] OLTP (Online Transaction Processing – обработка транзакций в реальном времени) является способом организации БД, при котором система работает с небольшими по размерам транзакциями, но идущими большим потоком, и при этом клиенту требуется от системы максимально быстрый ответ.

понятна пользователям, при этом необходимо обеспечить поддержку сложных запросов. Поскольку хранилища данных предназначены для обеспечения информацией бизнес-пользователей, этим пользователям необходимо предоставить соответствующий контекст в виде метаданных.

Создание хранилища данных – сложный процесс, требующий значительных затрат ресурсов и времени, поэтому некоторые организации строят витрины данных[3], обслуживающие одно или несколько направлений своей деятельности.

[3] Витрина данных – это срез хранилища данных, представляющий собой массив тематической, узконаправленной информации, ориентированный, например, на пользователей одной рабочей группы или департамента.

1. ЯЗЫК SQL ДЛЯ БИЗНЕС-АНАЛИЗА

1.1. ВВЕДЕНИЕ В ЯЗЫК SQL. ОБЩИЙ ФОРМАТ КОМАНДЫ SELECT. ФИЛЬТРОВАНИЕ ДАННЫХ С ПОМОЩЬЮ ПРЕДЛОЖЕНИЯ WHERE. УПОРЯДОЧЕНИЕ НАБОРА ДАННЫХ С ПОМОЩЬЮ ORDER BY. ВЫРАЖЕНИЯ, SQL-ФУНКЦИИ. ГРУППИРОВКА СТРОК

Язык SQL (Structured Query Language – язык структурированных запросов) является стандартным языком для работы с реляционными базами данных. Язык SQL с первоначальным названием SEQUEL (Structured English Query Language) появился в середине 1970-х гг. и был разработан корпорацией IBM в рамках проекта экспериментальной реляционной СУБД System R. В 1979 г. компания Relational Software (в настоящее время Oracle) выпустила первую коммерческую систему управления реляционными базами данных Oracle v2, поддерживающую язык SQL. Сегодня язык SQL широко распространен и фактически является стандартным языком реляционных баз данных. Стандарт на язык SQL был выпущен Американским национальным институтом стандартов (ANSI) в 1986 г., а в 1987 г. Международная организация стандартов (ISO) приняла его в качестве международного[4]. Дальнейшее развитие языка поставщиками СУБД потребовало принятия в 1992 г. нового расширенного стандарта (ANSI SQL-92, или просто SQL2). Впоследствии были разработаны стандарты SQL:1999 (SQL3), SQL:2003 и др. В настоящее время стандарт ISO состоит из девяти разделов: SQL/Framework:2011, SQL/Foundation:2011, SQL/CLI:2008, SQL/PSM:2011, SQL/MED:2008, SQL/OLB:2008, SQL/Schemata:2011, SQL/JRT:2008, SQL/XML:2011. Стандарт ANSI SQL, помимо перечисленных стандартов ISO, содержит еще одну часть: SQL/RPR:2012.

Различают интерактивную и встроенную формы языка SQL. Интерактивная используется для формулирования пользователем запросов

[4] ANSI (American National Standards Institute – Американский национальный институт стандартов) основан в 1918 г. Является частной некоммерческой организацией, которая управляет и координирует систему добровольной стандартизации и оценки соответствия стандартам США.

к базе данных при работе в интерактивном режиме (например, с использованием пакета SQL*Plus). Встроенный SQL состоит из команд SQL, помещенных в программы, написанные на каком-либо процедурном языке программирования. Механизм встраивания может быть самым различным – от прямой записи команд в исходных текстах до вызова специальных функций, одним из параметров которых является команда SQL.

Язык SQL – непроцедурный язык, применяемый для создания, модификации и управления данными в реляционных базах данных. Реляционная модель данных, разработанная американским математиком Е. Ф. Коддом в 1970 г., основана на математическом понятии отношения, физическим представлением которого является двухмерная таблица, содержащая строки одинаковой структуры и именованные столбцы. Таблица содержит первичный ключ – столбец (или совокупность столбцов), значения которого однозначно идентифицируют каждую строку этой таблицы. Связи между таблицами реализуются через внешние ключи. Внешний ключ – столбец (или совокупность столбцов), значения которого соответствуют значениям столбца – первичного ключа другой таблицы. Таблица должна обладать следующими свойствами: имя отлично от имен других таблиц; каждая ячейка содержит атомарное значение; каждый столбец имеет уникальное имя; важен порядок следования столбцов; каждый столбец таблицы содержит данные одного типа; каждая строка уникальна, т. е. в таблице отсутствуют строки-дубликаты; порядок следования строк не имеет значения. Логическая структура данных представлена совокупностью связанных таблиц. Модель поддерживает связи «один к одному», «один ко многим», имеется реализация связи «многие ко многим» через перекрестную таблицу и две связи «один ко многим».

К наиболее часто используемым типам данных языка SQL относятся следующие: символьные, числовые, тип DATE, двоичные и большие объекты. Рассмотрим поддерживаемые Oracle типы данных и их описание (табл. 1). Символьные типы данных представлены типами CHAR, VARCHAR2, LONG; числовые – типом NUMBER; двоичные – RAW, LONGRAW; большие объекты – CLOB, BLOB, BFILE.

Команды языка SQL предназначены для создания объектов базы данных и манипулирования ими, начальной загрузки данных в таблицы, обновления и удаления имеющихся в таблицах данных, выполнения запросов к БД, управления доступом к БД и общего администри-

рования БД. В зависимости от выполняемых функций команды языка SQL принадлежат к определенной категории:

- Data Definition Language, DDL – язык определения данных;
- Data Manipulation Language, DML – язык манипулирования данными;
- Transaction Control Language, TCL – язык управления транзакциями;
- команды управления сеансом;
- команда управления системой;
- встроенные SQL-операторы[5].

Таблица 1

Типы данных Oracle

Тип данных	Описание
CHAR (длина)	Символьные данные фиксированной длины (минимальная длина – 1 байт, максимальная – 2000 байт)
VARCHAR2 (длина)	Символьные данные переменной длины (минимальная длина – 1 байт, максимальная – 4000 байт)
LONG (длина)	Символьные данные переменной длины (минимальная длина – 1 байт, максимальная – 2 Гб)
NUMBER [(p [, s])]	Действительные числа (точность p – количество цифр в числе (от 1 до 38), масштаб s – количество цифр после запятой (от –84 до127))
DATE	Даты в диапазоне от 1 января 4712 г. до н. э. до 31 декабря 9999 г.; этот тип имеет поля YEAR, MONTH, DAY, HOUR, MINUTE, SECOND
RAW (длина)	Двоичные данные (минимальная длина – 1 байт, максимальная – 2000 байт)
LONGRAW (длина)	Двоичные данные (минимальная длина – 1 байт, максимальная – 2 Гб)
CLOB	Символьные данные (максимальная длина – (4 Гб – 1)*размер блока данных)
BLOB	Двоичные данные (максимальная длина – (4 Гб – 1)*размер блока данных)
BFILE	Указатель на двоичный файл, хранящийся вне БД (максимальная длина – 4 Гб)

[5] Здесь и далее рассматриваются операторы Oracle SQL.

Язык определения данных DDL позволяет создавать, изменять, удалять объекты базы данных, предоставлять и отменять пользователям или ролям различного вида привилегии и роли, анализировать информацию об объектах БД, задавать параметры аудита, добавлять комментарии в словарь данных. Командами языка DDL являются: CREATE, ALTER, RENAME, DROP, TRUNCATE, FLASHBACK, GRANT, REVOKE, ANALYZE, AUDIT, COMMENT и др.

Команды INSERT, UPDATE, DELETE, MERGE языка манипулирования данными DML используются для модификации данных: вставки, изменения, удаления строк таблицы; команда SELECT предназначена для построения запросов, результатом выполнения которых являются выбранные из одной или нескольких таблиц либо представлений[6] данные.

Команды COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION языка управления транзакциями TCL позволяют управлять изменениями, произведенными DML-командами. Так, например, команда COMMIT выполняет фиксацию текущей транзакции, а ROLLBACK – откат текущей транзакции. Транзакция – это действие или серия действий, которые осуществляют доступ или изменение содержимого базы данных и рассматриваются СУБД как единое целое. Транзакция является логической единицей работы, выполняемой в БД. В случае успешного выполнения всех составляющих транзакцию действий ее результаты фиксируются (COMMIT). В противном случае можно отменить транзакцию (ROLLBACK) и вернуть БД в состояние, в котором она находилась до выполнения транзакции.

Команды управления сеансом ALTER SESSION, SET ROLE динамически управляют настройками сеанса пользователя.

Команда управления системой ALTER SYSTEM динамически управляет настройками экземпляра СУБД Oracle.

Встроенные SQL-операторы позволяют встраивать команды языков DDL, DML и TCL в процедурный язык программирования, например C, C++.

Общий формат команды SELECT. Выбор информации из одной или нескольких таблиц либо представлений БД осуществляется путем формирования запроса с использованием команды SELECT. В простейшем виде запрос на выборку содержимого одной таблицы должен состоять из предложения SELECT, после которого указывается столбец(-цы)

[6] Представление – это виртуальная таблица, которая формируется на основе выбранной в результате выполнения запроса информации из одной или нескольких таблиц.

этой таблицы, и предложения FROM, после которого следует таблица, содержащая указанные после SELECT столбец(-цы):

```
SELECT имя_столбца1 [, имя_столбца2,...] | * | выражение
FROM имя_таблицы;
```

В качестве разделителя имен столбцов, указываемых после SELECT, используется символ «,». Признаком окончания запроса является символ «;».

Для того чтобы выбрать все столбцы таблицы, следует указать после SELECT символ «*».

Определим некоторые понятия, используемые при дальнейшем изложении материала. Понятие «ключевое слово» соответствует отдельной команде языка SQL. Например, SELECT, FROM – ключевые слова. Под предложением будем понимать часть оператора SQL. Так, SELECT <имя_столбца1>, как и FROM <имя_таблицы>, является предложением. Комбинация двух и более предложений соответствует понятию «оператор». Например, SELECT last_name FROM employees; является оператором.

В команде SELECT реализованы две операции реляционной алгебры: проекция и выборка. Суть операции проекции состоит в том, что из таблицы выбираются и (или) в ней переставляются некоторые столбцы. Операция выборки заключается в отборе строк таблицы, удовлетворяющих некоторому условию.

В списке выбора оператора SELECT, состоящего из имен столбцов и выражений, могут использоваться имена столбцов таблиц и представлений, литералы, функции, соединяемые арифметическими операциями +, –, *, /.

В случае необходимости сложному выражению можно присвоить псевдоним, указав ключевое слово AS и псевдоним после имени столбца.

Приведем запрос, при выполнении которого будут выбраны все строки таблицы employees и три столбца first_name, last_name, (SYSDATE-date_of_birth)/365. При формировании последнего столбца, значения которого будут соответствовать текущему возрасту сотрудников, используется функция SYSDATE и математические операции –, /. Кроме того, этому столбцу назначается псевдоним employee's_age:

```
SELECT first_name,
    last_name,
    (SYSDATE-date_of_birth)/365 AS employee's_age
FROM employees;
```

Следует отметить, что наличие ключевого слова AS необязательно, а сам псевдоним необходимо заключать в двойные кавычки, если

он содержит пробелы, специальные символы или символы как в верхнем, так и нижнем регистрах.

Для обозначения ситуации, когда в некотором столбце таблицы данные отсутствуют, т. е. значения являются неопределенными, в языке SQL имеется ключевое слово NULL. Оно не эквивалентно нулевому значению для числовых типов или понятию «пустая строка» для символьных типов. Если значение столбца, участвующего в арифметическом выражении, является неопределенным, т. е. NULL, то и результатом этого арифметического выражения будет неопределенное значение NULL.

Фильтрование данных с помощью предложения WHERE. Условие фильтрации строк в операторе SELECT задается в предложении WHERE. При формировании условия, которому должны удовлетворять выбираемые строки, можно использовать операции сравнения и логические операции языка SQL. Над значениями двух выражений X и Y могут быть выполнены следующие операции сравнения, представленные в табл. 2.

Таблица 2

Операции сравнения

Операция сравнения	Описание
X = Y	Проверка значений выражений X и Y на равенство; результат равен TRUE, если соотношение выполняется
X != Y, X <> Y, X ^= Y	Проверка значений выражений X и Y на неравенство; результат равен TRUE, если соотношение выполняется
X < Y, X > Y, X >= Y, X <= Y	Проверка значений выражений X и Y на соотношение «меньше, чем», «больше, чем», «больше или равно», «меньше или равно»; результат равен TRUE, если указанное соотношение выполняется
X [NOT] BETWEEN A AND B	Проверка, (не) находится ли значение выражения X в указанном диапазоне, определяемом значениями выражений A и B; результат равен TRUE, если указанное соотношение выполняется
X IN (список выражений | подзапрос)	Проверка значения выражения X на равенство некоторому элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом

Окончание табл. 2

Операция сравнения	Описание
X NOT IN (список выражений | подзапрос)	Проверка значения выражения X на неравенство ни одному элементу из списка значений выражений или множества значений, возвращенных подзапросом; результат равен TRUE, если указанное соотношение выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом
X LIKE Z	Проверка значения выражения X на подобие; результат проверки равен TRUE, если X совпадает с шаблоном Z. Шаблон представляет собой символьную строку, внутри которой символ «%» используется для сопоставления с любой строкой из нуля или более символов, кроме NULL – строки, а символ подчеркивания (_) сопоставляется с любым одиночным символом
X IS [NOT] NULL	Проверка значения выражения X на (не)пустое значение NULL; результат равен TRUE, если указанное соотношение выполняется
Операция сравнения с квантором ANY	Сравнение проверяемого значения со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >, <, >=, <=) выполняется хотя бы для одного элемента списка выражений или множества значений, возвращенных подзапросом
Операция сравнения с квантором ALL	Сравнение проверяемого значения со всеми элементами из заданного списка значений выражений или множества значений, возвращенных подзапросом; результат проверки равен TRUE, если указанная операция сравнения (=, !=, >, <, >=, <=) выполняется для всех элементов списка выражений или множества значений, возвращенных подзапросом
Операция сравнения EXISTS	Проверка результата выполнения подзапроса; результат проверки равен TRUE, если подзапрос возвращает не пустое множество значений

При построении сложных условий в предложении WHERE используются логические операции AND, OR, NOT. Логические операции объединяют простые условия в более сложное и выполняются в трехзначной логике.

AND	True	False	NULL
True	True	False	NULL
False	False	False	False
NULL	NULL	False	NULL

OR	True	False	NULL
True	True	True	True
False	True	False	NULL
NULL	True	NULL	NULL

NOT	True	False	NULL
	False	True	NULL

Упорядочение набора данных с помощью ORDER BY. В запросе SELECT возможно упорядочивание возвращаемых строк по возрастанию или убыванию значений определенных выражений. В качестве выражения может быть использовано имя столбца. Для реализации упорядочивания строк используется предложение ORDER BY, в котором указывается перечень, состоящий из одного или нескольких выражений, разделенных запятыми, по значениям которых и выполняется упорядочивание. По умолчанию выбираемые строки упорядочиваются по возрастанию значений указанных в перечне выражений. Для изменения порядка упорядочивания строк после выражения в перечне указывается ASC (по возрастанию) либо DESC (по убыванию). В следующем запросе формируется условие фильтрации, в соответствии с которым в результате выполнения запроса будут выводиться строки со сведениями о сотрудниках пенсионного возраста. Кроме того, полученные в результате фильтрации строки будут отсортированы по убыванию возраста:

```
SELECT first_name, last_name, (SYSDATE-date_of_birth)/360 AS employee's_age
FROM employees
WHERE (SYSDATE-date_of_birth)/360 >= 60
ORDER BY employee's_age DESC;
```

Выражения, SQL-функции. Язык SQL содержит функции, которые классифицируются по типу и количеству обрабатываемых функциями данных. В зависимости от типа данных различают символьные функции, числовые функции, функции работы с датой и временем, функции преобразования типа, функции работы с неопределенным значением NULL.

Рассмотрим символьные функции языка SQL, предназначенные для работы со строками. Эти функции могут возвращать строку или целое значение и используются в предложениях SELECT, WHERE и ORDER BY.

В случае, когда точно неизвестно, в каком регистре хранятся строковые данные (верхнем, нижнем, смешанном), целесообразно восполь-

зоваться символьными функциями преобразования регистра UPPER, LOWER, INITCAP.

Функция UPPER(str) возвращает строку str, все символы которой преобразованы в верхний регистр.

Функция LOWER(str) возвращает строку str, все символы которой преобразованы в верхний регистр.

Функция INITCAP(str) возвращает строку str, первый символ каждого слова которой преобразован в верхний регистр, а остальные – в нижний регистр.

Описание символьных функций преобразования строки приводится ниже.

Функция CONCAT(str1, str2) соединяет строки str1 и str2. Альтернативой использования CONCAT является оператор ||: str1 || str2.

Функция INSTR(str, substr [, start_pos [, nth_app]]) возвращает n-е вхождение подстроки substr в строку str. Необязательным аргументом start_pos является положение символа в строке, с которого начнется поиск. Если аргумент не указан, то по умолчанию его значение равно 1. Если параметр start_pos отрицательный, то функция INSTR рассчитывает позицию start_pos в обратном направлении от конца строки, а затем идет к началу строки. Необязательный аргумент nth_app является n-м вхождением подстроки. Если nth_app опущен, то по умолчанию его значение равно 1.

Функция SUBSTR(str, n, m) выделяет из строки str подстроку длины n, начиная с символа в позиции m.

Функция LPAD(str, n, chr) возвращает строку str, дополненную слева указанным символом chr до указанной длины n.

Функция RPAD(str, n, chr) возвращает строку str, дополненную справа указанным символом chr до указанной длины n.

Функция TRIM([[LEADING | TRAILING | BOTH] trim_char FROM] str1) удаляет все указанные символы с начала или окончания строки. Необязательные параметры LEADING | TRAILING | BOTH указываются для удаления символов trim_char с начала | окончания | как с начала, так и с окончания строки str1 соответственно. В случае отсутствия trim_char из строки str1 удаляются все пробелы.

Функция REPLACE(str1, str2, [str3]) заменяет последовательность символов в строке другим набором символов. Параметр str1 – это строка для замены последовательности символов другим набором символов. Параметр str2 – строка, которая будет искаться в str1. Параметр str3 не является обязательным. Все вхождения str2 будут заменены str3 в строке str1. Если параметр str3 опущен, то функция REPLACE просто удалит все вхождения str2 и вернет получившуюся строку.

Символьной функцией, которая возвращает целое значение, является вычисляющая длину строки str функция LENGTH(str).

Рассмотрим наиболее часто используемые числовые функции языка SQL.

Функция ROUND(n, [r]) осуществляет округление значения аргумента n, имеющего числовой тип, с точностью до количества указанных знаков r. При этом если значение r положительно, то округление производится до указанного количества знаков после запятой, если значение r отрицательно, то округление производится до указанного количества знаков до запятой. При r = 0 функция возвращает округленную целую часть аргумента n.

Функция TRUNC(n, [r]) выполняет усечение аргумента n, имеющего числовой тип с заданной точностью r.

Функция CEIL(n) возвращает наименьшее целое число, которое больше либо равно n.

Функция FLOOR(n) возвращает наибольшее целое число, которое меньше либо равно n.

Функции для работы с датами предназначены для работы с данными типа DATE. Подробно функции для работы с датами рассматриваются в теме 2.4 данного пособия.

Функции преобразования типа используются для преобразования данных символьного типа в числовой тип, символьного типа в тип DATE и, наоборот, для преобразования данных числового типа или типа DATE в символьный тип. Преобразование осуществляется в соответствии с задаваемым форматом. Формат преобразования имеет вид символьной строки, где каждый символ или группа символов имеет определенное назначение.

Функция TO_CHAR(x, [fmt, nls_lng]) преобразует значение x типа DATE или типа NUMBER в значение типа VARCHAR2 по формату fmt. Это формат, который будет использоваться для преобразования значения в строку. Параметры fmt и nls_ lng не являются обязательными. В nls_ lng задается язык, используемый для преобразования значения в строку.

Функция TO_DATE(str, [fmt, nls_ lng]) преобразует значение str типа CHAR или VARCHAR2 в значение типа DATE по формату fmt.

Функция TO_NUMBER(str, [fmt]) преобразует значение str типа CHAR или VARCHAR2 в значение типа NUMBER по формату fmt.

В форматах для даты используются группы символов, представленные в табл. 3.

Таблица 3

Группы символов в форматах для даты

Группа символов	Пояснение
DD	Номер дня месяца в диапазоне от 1 до 31
DAY	Полное название дня недели
MON	Задает краткое название месяца
MONTH	Полное название месяца
YY	Две последние цифры номера календарного года
YYYY	Полный номер календарного года

В форматах для чисел используются символы, представленные в табл. 4.

Таблица 4

Символы в форматах для чисел

Символ	Пояснение
9	Цифра (от 0 до 9)
.	Десятичная точка
0	Обязательный 0
S	Обязательное наличие знака + или –
$	Знак доллара, указанный в начале числа

Функции, которые выполняют операции над группами строк, классифицируют как групповые или агрегатные функции. Эти функции анализируют несколько или все строки таблицы, и результатом является обобщенное (агрегированное) значение.

Функция COUNT(*) возвращает количество строк в таблице. COUNT([DISTINCT] выражение) возвращает количество строк в группе, игнорируя значение NULL.

Функция SUM([DISTINCT] выражение) возвращает сумму значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция AVG([DISTINCT] выражение) возвращает среднее значение указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция MIN([DISTINCT] выражение) возвращает минимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция MAX([DISTINCT] выражение) возвращает максимальное из значений указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция VARIANCE([DISTINCT] выражение) возвращает дисперсию указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Функция STDDEV([DISTINCT] выражение) возвращает среднеквадратичное отклонение указанного выражения для группы строк или списка значений, игнорируя значение NULL.

Групповые (агрегатные) функции игнорируют неопределенное значение NULL. Групповые функции не используются в предложении WHERE. Функции MIN, MAX и COUNT применяются к данным любого типа. Функции AVG, SUM, STDDEV, VARIANCE и STDDEV применяются только к данным числового типа.

Прочие функции языка SQL относят к категории дополнительных функций. В этой категории имеются несколько функций для работы с неопределенным значением NULL.

Функция NVL(expr1, expr2) сравнивает значения выражения expr1 с неопределенным значением NULL. Если значение выражения expr1 равно NULL, то функция возвращает значение выражения expr2; если же значение выражения expr1 не равно NULL, то функция возвращает значение expr1.

Функция NVL2(expr1, expr2, expr3) расширяет функциональность функции NVL. Если значение выражения expr1 не равно NULL, то функция возвращает значение выражения expr2; если же значение выражения expr1 равно NULL, то функция возвращает значение expr3.

Функция NULLIF (expr1, expr2) сравнивает значения выражений expr1 и expr2. Если expr1 и expr2 равны, функция NULLIF возвращает NULL. В противном случае она возвращает expr1.

К дополнительным функциям также относятся аналитические функции языка SQL, среди которых имеются оконные, ранжирующие, статистические функции.

Ранжирование результатов выполняется функциями RANK OVER, DENSE_RANK OVER, ROW_NUMBER OVER. Функции RANK OVER и DENSE_RANK OVER возвращают ранг в группе значений и имеют синтаксис следующего вида:

```
RANK () OVER ([partition_clause] order_by_clause)
DENSE_RANK () OVER ([partition_clause] order_by_clause)
```

В скобках после ключевого слова OVER может присутствовать предложение PARTITION BY, которое задает сегментирование строк, и ука-

зывается предложение ORDER BY, определяющее порядок расположения строк в сегменте.

В отличие от RANK OVER функция DENSE_RANK OVER возвращает последовательные значения ранга.

Функция ROW_NUMBER OVER присваивает строкам, отсортированным по какому-либо столбцу, уникальные значения ранга начиная с 1. Синтаксис ROW_NUMBER OVER аналогичен синтаксису функций RANK OVER, DENSE_RANK OVER:

ROW_NUMBER () OVER ([partition_clause] order_by_clause)

Оконные функции, подобно групповым функциям, выполняют агрегацию заданной группы строк, но возвращают не по одному значению на группу, а несколько значений для каждой группы. Группа строк, которая подвергается агрегации, – это окно. С помощью оконных функций можно вычислить кумулятивные версии функций SUM, AVG, COUNT, MAX, MIN и многих других. Например, оконная функция SUM OVER имеет следующий синтаксис:

SUM (expr1) OVER ([PARTITION BY expr2] ORDER BY expr3 [,...] [{ASC/DESC}] [{NULLS FIRST/NULLS LAST}])

Предложение PARTITION BY определяет сегмент или группу строк, которая подвергается агрегации. При отсутствии этого предложения сегментом, агрегат которого будет вычислять оконная функция, является все результирующее множество. Упорядочение строк в группе выполняется с помощью предложения ORDER BY.

Для оконных функций может быть задано логическое смещение, т. е. можно дополнительно указать объем строк, участвующих в вычислении, выполняемом для каждой строки в группе. Логическое смещение задается константой, например RANGE 10 PRECEDING, или выражением, которое оценивается константой, или с помощью спецификации интервалов, например RANGE INTERVAL N DAY / MONTH / YEAR PRECEDING.

Функция LAG возвращает значение выражения, вычисленного для предыдущей строки результирующего множества.

Функция LEAD возвращает значение выражения, вычисленного для следующей строки результирующего множества.

В языке SQL присутствуют объекты, в которых реализована условная логика: выражение CASE и функция DECODE. Выражение CASE является стандартом ANSI/ISO, а функция DECODE разработана Oracle. Различают простое и поисковое выражение CASE. Простое выражение CASE для определения результата сравнивает выражение

с набором простых выражений. Синтаксис простого выражения CASE имеет следующий вид:

```
CASE имя_столбца | выражение
  WHEN выражение_сравнения1 THEN результирующее_выражение1
       [WHEN  выражение_сравнения2 THEN
              результирующее_выражение2...]
  [ELSE результирующее_выражение]
      END
```

Поисковое выражение CASE для определения результата вычисляет набор логических выражений. Синтаксис поискового выражения CASE представлен ниже:

```
CASE
  WHEN условие1 THEN результирующее_выражение1
  [WHEN условие2 THEN результирующее_выражение2...]
  [ELSE результирующее_выражение]
END
```

Рассматриваемая ранее функция NULLIF эквивалентна выражению CASE, в котором выполняется проверка на равенство каких-либо двух выражений.

Функция DECODE сравнивает значение столбца или выражение с каждым искомым значением. Если обнаруживается соответствие, то функция возвращает результат, в противном случае – значение по умолчанию. Если соответствий не обнаружено, а значение по умолчанию не задано, то функция возвращает неопределенное значение NULL. Синтаксис функции DECODE следующий:

```
DECODE (имя_столбца | выражение,
                поисковое_значение1, результирующее_выражение1
                [, поисковое_значение2, результирующее_выражение2,
            ...
                , default])
```

Группировка строк. Предложение GROUP BY. В SQL-операторе можно сгруппировать результирующие данные так, чтобы на основе нескольких записей таблицы формировалась одна выходная запись. Критерием для объединения в группу является совпадение значений в так называемых столбцах группировки. Кроме того, можно сформулировать дополнительное условие для создания новой группы.

Группировка задается указанием в операторе SELECT предложения GROUP BY:

```
GROUP BY <имя_столбца_группировки1> [, <имя_столбца_ группировки 2>, ...]
[HAVING <условие_отбора_в_группу>]
```

Здесь предложение HAVING работает подобно предложению WHERE, но определяет условие для формирования новой группы. Групповые (агрегирующие) функции, которые указываются при формировании выходных столбцов, работают не для всего оператора, а для одной группы.

Необходимость бизнес-аналитики в многогранной статистической информации из БД требует средств более высокого уровня агрегирования этих данных. Такими средствами расширения возможностей GROUP BY являются предложения CUBE и ROLLUP, функции GROUPING, выражение GROUPING SETS. Предложение ROLLUP строит агрегаты – промежуточные итоги на каждом запрашиваемом уровне и общее итоговые значение. ROLLUP предоставляет полезную, высокого уровня обобщения информацию, которая учитывается управленцами при принятии бизнес-решений. Предложение CUBE расширяет возможности ROLLUP для всех потенциальных комбинаций промежуточных итогов для некоторого заданного GROUP BY. Используя CUBE, легко получить данные для кросс-табличных отчетов. Для того чтобы улучшить обработку NULL-значений в строках, созданных ROLLUP и CUBE, используется функция GROUPING, которая возвращает значение 1, если строка – это промежуточный итог, созданный ROLLUP или CUBE, и 0 в противном случае. Для задания множественной группировки данных используется предложение GROUPING SET.

1.2. СОЕДИНЕНИЯ ТАБЛИЦ. ВЛОЖЕННЫЕ ПОДЗАПРОСЫ. ИЕРАРХИЧЕСКИЕ ЗАПРОСЫ

Соединения таблиц. Смысл операции соединения таблиц состоит в выборе строк из двух или более таблиц одним запросом SELECT. Условиями соединения таблиц называют условия запроса SELECT, в которых сравниваются значения столбцов соединяемых таблиц. Для установления соединений таблиц в базе данных Oracle допустимо использовать два набора команд: стандартные соединения ANSI/ISO SQL 99 и соединения Oracle.

Рассмотрим команды стандартных соединений ANSI/ISO SQL 99: NATURAL JOIN, CROSS JOIN, JOIN USING, JOIN ON, LEFT OUTER JOIN, RIGHT OUTER JOIN, FULL OUTER JOIN. Применение команды NATURAL JOIN для соединения таблиц возможно в том случае, если названия и типы данных столбцов с совпадающими значениями являются одинаковыми. В этом случае условие соединения в запросе не указывается. При выполнении соединения таблиц с использовани-

ем команды CROSS JOIN получаем декартово произведение таблиц, в котором каждой строке одной таблицы соответствует каждая строка другой таблицы. Результирующее множество строк содержит все возможные комбинации строк из соединяемых таблиц. При таком соединении таблиц условие соединения в запросе отсутствует.

Когда требуется выполнить соединение таблиц, названия столбцов с совпадающими значениями которых являются одинаковыми, а типы данных – различными, следует использовать команду JOIN USING.

Команда JOIN ON применяется для соединения таблиц, если названия столбцов с совпадающими значениями неодинаковы либо когда условие соединения задается неравенством.

В результате выполнения внешнего соединения получают все строки, которые удовлетворяют условию соединения, а также строки таблицы, указанной в запросе слева/справа от соответствующей команды LEFT OUTER JOIN, RIGHT OUTER JOIN или строки обеих таблиц (FULL OUTER JOIN), которые не удовлетворяют условию соединения.

Самосоединение используется для соединения таблицы самой с собой подобно соединению двух таблиц. В предложении FROM название таблицы указывается дважды с различными псевдонимами.

Операции соединения Oracle представлены операциями эквисоединения таблиц, декартовым произведением таблиц, соединением по неравенству, внешним соединением таблиц, самосоединением.

Результатом эквисоединения являются строки с одинаковыми значениями столбцов условия соединения, которое указывается в предложении WHERE. Эквисоединение Oracle эквивалентно выполнению соединения таблиц с использованием NATURAL JOIN, JOIN USING, JOIN ON (когда в условии соединения используется знак «=») ANSI/ISO SQL 99.

Формируя условие соединения в предложении WHERE с помощью операторов сравнения <=, >=, BETWEEN... END, получаем соединение по неравенству.

Декартово произведение таблиц получаем, если в запросе на соединение нескольких таблиц отсутствует условие соединения предложения WHERE.

Указав (+) в условии соединения предложения WHERE, получим правое либо левое внешнее соединение. Полное внешнее соединение с помощью синтаксиса соединений Oracle выполнить невозможно. Сравним синтаксис соединений Oracle с ANSI/ISO SQL 99 (табл. 5).

Таблица 5

Сравнение синтаксиса соединений Oracle с ANSI/ISO SQL 99

Тип соединения	Синтаксис Oracle	Синтаксис ANSI/ ISO SQL 99
Эквисоединение	Условие соединения предложения WHERE (когда используется оператор =)	NATURAL JOIN JOIN USING JOIN ON (когда в условии соединения используется оператор =)
Декартово произведение	Условие соединения предложения WHERE отсутствует	CROSS JOIN
Соединение по неравенству	Условие соединения в предложении WHERE (когда используются операторы сравнения <=, >=, BETWEEN... END)	JOIN ON (когда в условии соединения используются операторы сравнения <=, >=, BETWEEN... END)
Левое (правое) внешнее соединение	Условие соединения в предложении WHERE с (+) слева (справа) от оператора =	LEFT OUTER JOIN (RIGHT OUTER JOIN)
Полное внешнее соединение	-	FULL OUTER JOIN

Иерархические запросы. Иерархические запросы тесно связаны с операцией самосоединения. Предыдущий пример использует самосоединение, чтобы вывести непосредственного начальника для каждого сотрудника. Аналогичный результат можно получить, используя иерархические запросы. Иерархические запросы содержат такие ключевые слова, как START WITH, CONNECT BY PRIOR, LEVEL. Предложение START WITH задает строку/строки, лежащие в корне иерархии. С помощью предложения CONNECT BY PRIOR указывается, как выполнять соединения строк. Псевдостолбец LEVEL возвращает значение 1 для корневого узла иерархии, 2 – для узлов, которые являются непосредственными потомками корневого узла, и т. д. В результате выполнения следующего запроса получаем представление отношений «потомок – родитель – прародитель» таблицы employees, в которой столбец empno является первичным ключом, а столбец manager_id – внешним ключом, значения которого соответствуют значениям столбца empno:

```
SELECT ltrim(sys_connect_by_path(empname,'-->'),'-->') leaf___branch___
root
FROM employees
WHERE LEVEL = 3
START WITH
empname = 'CLARK'
CONNECT BY PRIOR manager_id = empno;
```

С помощью функции SYS_CONNECT_BY_PATH получаем сотрудника CLARK и его непосредственного руководителя, затем руководителя этого непосредственного руководителя.

Вложенные подзапросы. Вложенные запросы, или подзапросы, используются для возврата строк или множества значений, которые будут использованы внешним запросом. Различают коррелированные и некоррелированные подзапросы. Подзапрос является некоррелированным, если он выполняется один раз для внешнего запроса. Если подзапрос выполняется один раз для каждой строки, извлеченной внешним запросом, то такой запрос называют коррелированным. Отличительным признаком коррелированного подзапроса является присутствие в предложении WHERE ссылок на столбцы внешнего запроса.

Однострочные подзапросы используют операторы >, =, >=, <, <>, <= и возвращают одну строку. Многострочные подзапросы используют операторы IN, ANY, ALL и возвращают группу строк. В приводимом далее коррелированном многострочном запросе определяются коды работников, зарплата которых в ноябре 2017 г. снизилась по сравнению с каким-либо предыдущим месяцем этого же года:

```
SELECT empno FROM salary s1
WHERE month = 11
AND year = 2017
AND salvalue <
                ANY(
                    SELECT salvalue
                    FROM salary s2
                    WHERE s1. empno = s2.empno
                    AND s2.month < 11
                    AND s2.year = 2017);
```

В многокомпонентном операторе SELECT можно выполнить определенные действия по объединению в единое целое группы строк, извлекаемых отдельно выполняемыми подзапросами. Такие строки должны быть согласованными по структуре, т. е. структуры таких строк должны совпадать по количеству, порядку следования и типу данных,

входящих в них столбцов. Порядок объединения задается операциями, представленными в табл. 6.

Таблица 6

Операции объединения таблиц

Операция объединения	Пояснение
UNION ALL	Объединение всех строк, извлеченных одним или несколькими запросами, включая повторяющиеся
UNION	Объединение всех строк, извлеченных одним или несколькими запросами, с устранением дублирующих строк
INTERSECT	Объединение только тех строк, которые присутствуют в результатах выполнения каждого из запросов, с устранением дублирующих строк
MINUS	Объединение всех неповторяющихся строк, извлеченных первым запросом, но не извлеченных вторым

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Перечислите типы данных и категории команд языка SQL.
2. Какие команды языка SQL применяются для выборки, фильтрации, сортировки данных? Приведите примеры использования этих команд.
3. Перечислите функции языка SQL для работы с символьными данными. Приведите пример использования символьных функций SUBSTR, TRIM, REPLACE.
4. Перечислите функции языка SQL для работы с числовыми данными. Поясните отличие функций ROUND и TRUNC.
5. Охарактеризуйте функции преобразования типа языка SQL.
6. Перечислите агрегирующие функции языка SQL.
7. Перечислите аналитические функции языка SQL.
8. Каким образом реализована условная логика в языке SQL?
9. Какие возможности группировки строк предусмотрены в языке SQL?
10. Перечислите типы и команды соединения таблиц.
11. Что такое вложенные подзапросы? Приведите примеры коррелированного и некоррелированного вложенных подзапросов.
12. Перечислите команды иерархических запросов.

2. СОСТАВЛЕНИЕ ОТЧЕТОВ НА ЯЗЫКЕ SQL

2.1. ТРАНСПОНИРОВАНИЕ РЕЗУЛЬТИРУЮЩИХ МНОЖЕСТВ. РАЗВОРАЧИВАНИЕ РЕЗУЛЬТИРУЮЩЕГО МНОЖЕСТВА В ОДНУ/НЕСКОЛЬКО СТРОК. ОБРАТНОЕ РАЗВОРАЧИВАНИЕ РЕЗУЛЬТИРУЮЩЕГО МНОЖЕСТВА

При составлении отчетов, как правило, применяются специфическое форматирование, различные уровни агрегации, транспонирование (или разворачивание) результирующих множеств. Рассмотрим задачи разворачивания результирующего множества в одну/несколько строк, обратного разворачивания результирующего множества в БД Oracle.

В задаче *разворачивания результирующего множества в одну строку* требуется транспонировать (развернуть) группу строк, превращая их значения в столбцы. Принцип решения заключается в применении поискового выражения CASE к каждой строке, возвращенной запросом, чтобы разложить строки в столбцы, и использовании групповой функции. Рассмотрим следующий пример. Пусть имеется результирующее множество dept_emp с двумя столбцами, информирующее о количестве сотрудников в каждом отделе: столбец deptno содержит номера отделов (10, 20, 30), столбец count(*) – количество сотрудников в каждом отделе.

Требуется переформатировать результат так, чтобы он содержал одну строку со значениями количества сотрудников, а количество столбцов соответствовало количеству отделов (в данном примере три столбца).

Выполним транспонирование результирующего множества с помощью предложения CASE и агрегирующей функции SUM:

```
SELECT SUM(CASE WHEN deptno = 10 THEN 1 ELSE 0 END) AS
deptno_10,
       SUM(CASE WHEN deptno = 20 THEN 1 ELSE 0 END) AS
deptno_20,
       SUM(CASE WHEN deptno = 30 THEN 1 ELSE 0 END) AS
deptno_30
FROM dept_emp;
```

Для того чтобы преобразовать строки в столбцы в данном запросе к каждой строке, возвращаемой запросом, применяется предложение

CASE. Далее с помощью агрегирующей функции SUM подсчитывается количество экземпляров каждого значения deptno.

Другой подход к решению задачи разворачивания результирующего множества заключается в использовании вложенного запроса для подсчета количества сотрудников в отделе. Предложение CASE основного запроса преобразует строки в столбцы по аналогии с предыдущим решением:

```
SELECT  MAX(CASE WHEN deptno = 10 THEN emp_count ELSE NULL
END) AS deptno_10,
        MAX (CASE WHEN deptno = 20 THEN emp_count ELSE NULL
END) AS deptno_20,
        MAX (CASE WHEN deptno = 30 THEN emp_count ELSE NULL
END) AS deptno_30
FROM (
        SELECT deptno, COUNT(*) AS emp_count
        FROM dept_emp
        GROUP BY deptno
        );
```

Для разворачивания результирующего множества в БД Oracle содержится функция PIVOT, синтаксис которой имеет вид

```
SELECT...
FROM <table-expr>
   PIVOT
     (
     aggregate-function(<column>) AS <alias>
     FOR <pivot-column> IN (<value1>, <value2>,..., <valuen>)
     ) AS <alias>
WHERE...
```

В задаче *разворачивания результирующего множества в несколько строк* требуется преобразовать строки в столбцы, создавая для каждого значения заданного столбца отдельный столбец.

Пусть имеется результирующее множество emp с двумя столбцами, информирующее о должностях, которые когда-либо занимали сотрудники: столбец job содержит названия должностей, столбец emp_name – фамилии сотрудников.

Требуется переформатировать результат так, чтобы названия должностей соответствовали названиям столбцов нового множества, а в строках были указаны фамилии сотрудников, занимающих соответствующую названию столбца должность.

Для решения поставленной задачи используется ранжирующая функция ROW_NUMBER OVER, которая позволяет сделать комбина-

цию job/ emp_name уникальной. Разворачивается результирующее множество с помощью предложения CASE и агрегирующей функции MAX:

```
SELECT  MAX(CASE WHEN job = 'PRESIDENT' THEN emp_name ELSE
        NULL END) AS president,
        MAX(CASE WHEN job = 'MANAGER' THEN emp_name ELSE
        NULL END) AS managers,
        MAX(CASE WHEN job = 'ANALYST' THEN emp_name ELSE
        NULL END) AS analysts,
        MAX(CASE WHEN job = 'SALESMAN' THEN emp_name ELSE
        NULL END) AS salesmen
FROM (
        SELECT job,
                emp_name,
                ROW_NUMBER () OVER (PARTITION BY job ORDER
                BY emp_name) rn
        FROM emp
      )
GROUP BY rn;
```

В задаче *обратного разворачивания результирующего множества* требуется преобразовать столбцы в строки. Для того чтобы преобразовать столбцы в строки, используется декартово произведение. Количество столбцов, которые требуется преобразовать в строки, должно быть известно, потому что количество строк табличного выражения, используемого для создания декартового произведения, должно быть больше либо равно количеству транспонируемых столбцов.

Пусть имеется результирующее множество dept_emp с тремя столбцами (deptno_10, deptno_20, deptno_30) и одной строкой, информирующее о количестве сотрудников в каждом отделе. Требуется преобразовать это множество таким образом, чтобы получилось два столбца и три строки. Первый столбец deptno должен содержать номера отделов (10, 20, 30), второй столбец counts_by_dept – количество сотрудников в каждом отделе:

```
SELECT d.deptno,
        CASE d.deptno
                WHEN 10 THEN emp_cnts.deptno_10
                WHEN 20 THEN emp_cnts.deptno_20
                WHEN 30 THEN emp_cnts.deptno_30
        END AS counts_by_dept
FROM (
SELECT SUM (CASE WHEN deptno = 10 THEN 1 ELSE 0 END) AS
deptno_10,
```

```
        SUM (CASE WHEN deptno = 20 THEN 1 ELSE 0 END) AS deptno_20,
        SUM (CASE WHEN deptno = 30 THEN 1 ELSE 0 END) AS deptno_30
FROM dept_emp
        ) emp_cnts,
        (SELECT deptno FROM dept) d;
```

Вложенный запрос emp_cnts является исходным результирующим множеством, которое требуется преобразовать в строки. Поскольку в исходном результирующем множестве три столбца, то будет создано три строки.

Для обратного разворачивания результирующего множества в БД Oracle используется функция UNPIVOT.

Задача *обратного разворачивания результирующего множества в один столбец* состоит в том, чтобы вывести все возвращаемые запросом столбцы в одном столбце. Например, стоит задача получить имя emp_name, должность job, среднюю зарплату avg_salary всех сотрудников 30-го отдела. Все три значения должны быть выведены в одном столбце в трех строках для каждого сотрудника, и значения для разных сотрудников должны быть разделены пустой строкой:

```
SELECT CASE rn
                WHEN 1 THEN emp_name
                WHEN 2 THEN job
                WHEN 3 THEN CAST (avg_salary as CHAR(4))
         END emps
FROM (
        SELECT e.emp_name, e.job, e,avg_salary,
                ROW_NUBER () OVER (PARTITION BY e.emp_no ORDER
                BY e.emp_no) rn
        FROM emps_info e,
              (SELECT *
               FROM emps_info) r
        WHERE deptno = 30
        );
```

Используя ранжирующую функцию ROW_NUMBER OVER, присваиваем ранг каждому сотруднику 30-го отдела. Сегментирование проводится по empno, поэтому всем сотрудникам 30-го отдела присвоен ранг 1. Выполняется декартово произведение между полученным результирующим множеством и некоторой таблицей, имеющей три строки (в данном примере в качестве такой таблицы выступает представление emps_info со столбцами empno, emp_name, job, deptno, avg_salary). С помощью выражения CASE значения emp_name, job, avg_salary всех сотрудников размещаются в один столбец (чтобы зна-

чения avg_salary могли использоваться в CASE, их необходимо привести к строковому типу).

При формировании отчета часто ставится задача *исключения повторяющихся значений из результирующего множества*, т. е. повторяющиеся значения в столбце должны отображаться только один раз. Например, требуется извлечь значения deptno и emp_name, при этом необходимо сгруппировать вместе все строки для каждого значения deptno и выводить каждое значение deptno только один раз:

```
SELECT TO_NUMBER(
            DECODE (LAG(deptno) OVER (ORDER BY deptno),
                    deptno, NULL, deptno)
                    ) deptno, emp_name
FROM emp_t;
```

Оконная функция LAG OVER возвращает в формируемый ею столбец lag_deptno предыдущее значение deptno для каждой строки. Для строк, в которых значения deptno и lag_deptno совпадают, должно быть присвоено значение NULL. Для этого используется функция DECODE (функция TO_NUMBER включена, чтобы привести значение deptno к числовому типу).

2.2. СОЗДАНИЕ БЛОКОВ ДАННЫХ И ГИСТОГРАММ

При *создании блоков данных фиксированного размера* требуется организовать данные в одинаковые по размеру блоки с определенным количеством элементов в каждом блоке. Общее количество блоков неизвестно, но каждый из них должен содержать одинаковое число элементов. Например, необходимо организовать сотрудников таблицы employees в группы по три на основании значения столбца empno:

```
SELECT  ROW_NUMBER()OVER(ORDER BY empno) rn,
        ROUND(ROW_NUMBER()OVER(ORDER BY empno)/3, 2) div,
        CEIL (ROW_NUMBER () OVER (ORDER BY empno)/3) grp,
        empno,
        empname
FROM employees;
```

В приведенном запросе используется функция ROW_NUMBER OVER, чтобы ранжировать сотрудников по EMPNO. Эта функция ROW_NUMBER OVER присваивает «порядковые номера» строкам, сортированным по столбцу EMPNO. Применяем функцию CEIL после деления результата на 3. Деление на 3 логически организует строки

в группы по 3, т. е. 3 значения, которые меньше или равны 1; 3 значения, которые больше 1, но меньше либо равны 2. Функция CEIL возвращает наименьшее целое число, которое больше, чем переданное в нее значение; это обеспечит создание групп целых чисел.

Рассмотрим задачу *создания заданного количества блоков*, в которой требуется организовать данные в определенное число блоков. Особенность этой задачи заключается в том, что заранее неизвестно количество элементов в блоке, но известно количество блоков. Например, записи таблицы employees должны быть разделены на три группы с указанием соответствующего номера группы для каждой строки этой группы:

```
SELECT NTILE(3) OVER(ORDER BY empno) grp,
                empno,
                empname
                FROM employees
                ORDER BY 1;
```

Для создания заданного числа блоков используем оконную функцию NTILE. NTILE разбивает упорядоченное множество на требуемое число сегментов. Если количество записей не делится на это число без остатка, записи «остатка» распределяются в доступные блоки, начиная с первого.

С помощью языка SQL можно создать гистограммы простейшего вида. Например, требуется создать *горизонтальную гистограмму*: отобразить количество служащих в каждом отделе в виде горизонтальной гистограммы, в которой каждый служащий представлен экземпляром символа «*»:

```
SELECT deptno, LPAD ('*', COUNT(*), '*') cnt
FROM career
GROUP BY deptno
ORDER BY cnt;
```

С помощью агрегирующей функции COUNT подсчитывается количество служащих в каждом отделе. Затем для каждого отдела возвращается соответствующее число символов «*», исходя из значения, возвращаемого COUNT(*). Для этого передаем COUNT(*) как аргумент в строковую функцию LPAD.

При создании *вертикальной гистограммы* требуется получить гистограмму, в которой значения увеличиваются вдоль оси снизу вверх. Путь необходимо отобразить количество служащих в каждом отделе в виде вертикальной гистограммы, в которой каждый служащий представлен экземпляром символа «*»:

```
SELECT MAX(deptno_10) d10,
       MAX(deptno_20) d20,
```

```
        MAX(deptno_30) d30
FROM (
        SELECT ROW_NUMBER()OVER(PARTITION BY deptno ORDER
        BY empno) rn,
        CASE WHEN deptno=10 THEN '*' ELSE NULL END deptno_10,
        CASE WHEN deptno=20 THEN '*' ELSE NULL END deptno_20,
        CASE WHEN deptno=30 THEN '*' ELSE NULL END deptno_30
        FROM career
      )
GROUP BY rn
ORDER BY 1 DESC, 2 DESC, 3 DESC, 4 DESC;
```

Используется ранжирующая функция ROW_NUMBER OVER, чтобы уникально идентифицировать каждый экземпляр символа «*» для каждого deptno. С помощью агрегирующей функции MAX разворачивается результирующее множество и группируется по значениям, возвращенным ROW_NUMBER OVER.

2.3. Вычисления в отчетах. Вычисление простых подсумм. Вычисление подсумм для сочетаний. Одновременная агрегация разных групп. Агрегация скользящего множества значений

В задаче *вычисления простых подсумм* под «простой суммой» подразумевается результирующее множество строк, содержащее значения, полученные в результате агрегации одного столбца, и общий итог таблицы. Пусть требуется получить результирующее множество, содержащее суммы заработных плат таблицы employees по должностям, а также сумму всех заработных плат таблицы employees.

Для решения этой задачи используют расширение ROLLUP оператора GROUP BY:

```
SELECT CASE GROUPING(jobname)
        WHEN 0 THEN jobname
        ELSE 'Total'
        END jobname,
        SUM(salary) salary
FROM employees
GROUP BY ROLLUP(jobname);
```

Для суммирования заработных плат используется агрегирующая функция SUM, а с помощью расширения ROLLUP оператора GROUP

BY результаты организовываются в подсуммы по jobname и находится общая сумма.

Расширение ROLLUP применяется также для вычисления промежуточных сумм. Например, пусть требуется получить результирующее множество, содержащее суммы заработных плат таблицы employees по отделам и должностям, а также сумму всех заработных плат таблицы employees:

```
SELECT deptname, jobname,
        SUM(salary) salary
FROM employees
GROUP BY ROLLUP(deptname, jobname);
```

Рассмотрим *вычисление подсумм для всех возможных сочетаний*. Требуется найти суммы всех заработных плат по отделам (deptname), должностям (jobname) и для каждого сочетания отдел/должность (deptname/jobname). Должна быть также вычислена общая сумма всех заработных плат. В данном случае следует использовать расширение CUBE оператора GROUP BY:

```
SELECT deptname, jobname,
        CASE GROUPING(deptname) || GROUPING(jobname)
            WHEN '00' THEN 'TOTAL BY DEPT AND JOB'
            WHEN '10' THEN 'TOTAL BY JOB'
            WHEN '01' THEN 'TOTAL BY DEPT'
            WHEN '11' THEN 'GRAND TOTAL for TABLE'
        END category,
        SUM(salary) salary
        FROM employees
        GROUP BY CUBE(deptname, jobname)
        ORDER BY GROUPING(deptname), GROUPING(jobname);
```

Применяя агрегатную функцию SUM и группируя значения по deptname и jobname, находим суммарные заработные платы для каждого сочетания deptname, jobname. Следующий шаг – вычислить подсуммы по deptname и jobname и общую сумму для всей таблицы. С помощью расширения CUBE оператора GROUP BY выполняем агрегацию значений salary по deptname, jobname и затем для всей таблицы. Далее используем функцию GROUPING в сочетании с выражением CASE, чтобы представить результаты в более выразительном формате. GROUPING(jobname) возвращает значения 1 или 0 в зависимости от того, получены ли значения SALARY оператором GROUP BY или его расширением CUBE. Если значение возвращено CUBE, получаем 1, в противном случае – 0. Аналогично для GROUPING(deptname).

Группировка выполняется по deptname и jobname. Таким образом, в результате вызова GROUPING для строки, представляющей сочетание deptname и jobname, должен быть возвращен 0.

При *агрегации разных групп одновременно* требуется выполнить агрегацию в разных измерениях одновременно. Например, необходимо получить результирующее множество, в котором для каждого сотрудника указаны имя (empname), отдел (deptname), количество сотрудников в отделе (включая его самого), количество сотрудников, занимающих ту же должность, что и этот сотрудник (включая его самого), и общее число сотрудников в таблице. Выполнение такой агрегации осуществляется с помощью оконной функции COUNT OVER:

```
SELECT empname, deptname,
    COUNT(*)OVER(PARTITION BY deptname) deptname_emp_cnt,
    jobname,
    COUNT(*)OVER(PARTITION BY jobname) jobname_emp_cnt,
    COUNT(*)OVER() total
FROM employees;
```

Предложение PARTITION BY определяет сегмент, т. е. логически разбивает результирующее множество по критериям; если не указать сегмент, то все результирующее множество рассматривается как одно целое. В данном запросе строки таблицы employees группируются по deptname, операция COUNT выполняется над всеми строками каждой группы. В предложении COUNT(*)OVER(PARTITION BY deptname) оператор PARTITION BY выделяет сегменты или группы по значениям названий отделов. Аналогичные действия выполняются для второй функции COUNT с сегментированием по jobname. Пустые круглые скобки означают «все результирующее множество». Таким образом, две первые функции COUNT обрабатывают заданные группы или сегменты данных, а последняя COUNT подсчитывает все строки таблицы employees.

Рассмотрим задачу *агрегации скользящего среднего*. Пусть требуется выполнить скользящую агрегацию, например, найти скользящую сумму заработных плат. Вычислим сумму для каждого интервала в 90-й день, начиная с даты приема на работу (столбец startdate таблицы career) первого сотрудника, чтобы увидеть динамику изменения расходов для каждого 90-дневного периода между датами приема на работу первого и последнего сотрудника:

```
SELECT startdate, salvalue,
    SUM(salvalue)OVER(ORDER BY startdate RANGE BETWEEN 90
PRECEDING AND CURRENT ROW) spending_pattern
FROM career;
```

При решении этой задачи применяется оконная функция SUM OVER и сортировка по столбцу startdate. Диапазон в 90 дней задается в операторе сегментирования для того, чтобы в сумму были включены заработные платы всех сотрудников, принятых на работу в течение предыдущих 90 дней.

2.4. РАБОТА С ДАТАМИ В ОТЧЕТАХ. СОЗДАНИЕ КАЛЕНДАРЯ

Запросы с использованием дат являются достаточно распространенной составляющей отчетов. В БД Oracle по умолчанию форматом ввода и отображения даты является формат DD-Mon-YYYY, например 01-Jan_2018. Но тип данных DATE хранит столетие, год, месяц, день, часы, минуты, секунды, т. е. этот тип имеет поля YEAR, MONTH, DAY, HOUR, MINUTE, SECOND соответственно. Валидные даты БД Oracle находятся в диапазоне от 1 января 4712 г. до н. э. до 31 декабря 9999 г.

К данным типа DATE допустимо прибавить или отнять числовую константу, которая соответствует количеству дней или части дня. Например, добавим месяц к дате приема сотрудника на работу (столбец startdate):

```
SELECT first_name, last_name, startdate + 30
FROM employees;
```

Над переменными типа DATE можно выполнить вычитание. Результат операции определяет количество дней между участвующими в операции датами. Например, определим количество проработанных сотрудником лет. Для этого от даты окончания работы (столбец enddate) вычтем дату начала работы (столбец startdate), полученное количество дней поделим на 365. Если сотрудник не уволен, т. е. значение столбца enddate не определено (NULL), то с помощью функций NVL и SYSDATE устанавливаем это значение равным текущей дате сервера БД:

```
SELECT first_name, last_name, (NVL(enddate, SYSDATE) – startdate)/365
FROM employees;
```

Для работы с данными типа DATE используются ниже перечисленные функции.

Функция SYSDATE возвращает текущие дату и время сервера БД.

Функция ADD_ MONTHS(date1, n) возвращает дату плюс n месяцев.

Функция MONTHS_BETWEEN(date1, date2) возвращает количество месяцев между датами date1, date2.

Функция NEXT_DAY(date1, day) возвращает первый день недели, который больше date1. Параметр day может принимать следующие значения: понедельник, вторник, среда, четверг, пятница, суббота, воскресенье.

Функция LAST_DAY(date1) возвращает дату последнего дня месяца на основе значения даты.

Функция ROUND (date1, [fmt]) округляет значение даты в соответствии с форматом fmt.

Функция TRUNC (date1, [fmt]) усекает значение даты в соответствии с форматом fmt.

В задаче *создания календаря* требуется создать календарь текущего месяца, который содержит семь столбцов, соответствующих дням недели (понедельник, вторник и т. д.), и пять строк. При решении этой задачи необходимо вернуть все дни текущего месяца, а затем разделить их на недели. Возврат всех дней текущего месяца реализовывается с помощью рекурсивного оператора CONNECT BY, благодаря которому получаем столько строк, сколько дней в текущем месяце. Для каждого дня месяца определяем порядковый номер, день недели, текущий месяц, ISO-номер недели. Далее разбиваем месяц на недели по выбранному дню, используя выражение CASE и функцию MAX:

```
WITH x
  AS (
SELECT *
FROM (
     SELECT TRUNC(SYSDATE, 'mm')+LEVEL-1 month_date,
     TO_CHAR(TRUNC (SYSDATE, 'mm')+ LEVEL -1, 'iw') week_number,
     TO_CHAR (TRUNC (SYSDATE, 'mm')+ LEVEL -1, 'dd') day_number,
     TO_NUMBER(TO_CHAR (TRUNC (SYSDATE, 'mm')+ LEVEL -1, 'd'))
week_day,
     TO_CHAR (TRUNC (SYSDATE, 'mm')+ LEVEL -1, 'mm') curr_month,
     TO_CHAR (SYSDATE, 'mm') mnth
     FROM DUAL
CONNECT BY LEVEL <=31
        )
WHERE curr_month = mnth
)
SELECT MAX(CASE week_day WHEN 2 THEN day_number end) "Пн",
      MAX (CASE week_day WHEN 3 THEN day_number END) "Вт",
      MAX (CASE week_day WHEN 4 THEN day_number END) "Ср",
      MAX (CASE week_day WHEN 5 THEN day_number END) "Чт",
      MAX (CASE week_day WHEN 6 THEN day_number END) "Пт",
      MAX (CASE week_day WHEN 7 THEN day_number END) "Сб",
```

```
    MAX (CASE week_day WHEN 1 THEN day_number END) "Вс"
  FROM x
GROUP BY week_number
ORDER BY week_number;
```

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Поясните суть задачи разворачивания результирующего множества. Какие команды языка SQL используются для решения этой задачи?

2. Поясните суть задачи исключения повторяющихся значений из результирующего множества. Какие функции языка SQL используются для решения этой задачи?

3. Поясните суть задачи создания блоков данных фиксированного размера. Какие функции языка SQL используются для решения этой задачи?

4. Какие расширения команды GROUP BY используются для вычисления подсумм для всех возможных сочетаний?

5. Поясните использование функции GROUPING при вычислении подсумм для всех возможных сочетаний.

6. Поясните использование оконной функции COUNT OVER (), предложения PARTITION BY при выполнении агрегации разных групп одновременно.

7. Перечислите функции языка SQL для работы с данными типа DATE.

3. ВВЕДЕНИЕ В ORACLE BUSINESS INTELLIGENCE

3.1. ХАРАКТЕРИСТИКА, ЗАДАЧИ И ПРЕИМУЩЕСТВА ORACLE BUSINESS INTELLIGENCE SUITE ENTERPRISE EDITION

Oracle Business Intelligence – это комплекс технологий и приложений для бизнес-анализа и создания хранилищ данных, предназначенный для обеспечения лучшего видения и понимания бизнеса и предоставления быстрого веб-доступа к актуальной информации. В Oracle BI входят инструменты для формирования нерегламентированных запросов, доставки и представления результатов, а также поддержки соединения с разнородными источниками данных.

Первоначально корпорацией Oracle был выпущен продукт Oracle Business Intelligence, который базировался на платформе Oracle Application Server Enterprise Edition и включал в себя такие инструменты, как Oracle Discovery, Oracle Reports.

В 2005 г. корпорация Oracle после приобретения Sieble CRM Systems анонсировала Oracle Business Intelligence Enterprise Edition. Этот продукт был получен в результате обновления платформы бизнес-аналитики Oracle посредством продукта Sieble Analytics. В настоящее время существуют три продукта бизнес-аналитики Oracle BI.

Oracle Business Intelligence Enterprise Edition – продукт, основанный на Sieble Analytics и содержащий полный набор инструментов и серверов бизнес-аналитики Oracle BI. Это интегрированный пакет продуктов на основе единой аналитической и вычислительной инфраструктуры, единой модели управления метаданными, единой модели безопасности и управления привилегиями пользователей, а также общих инструментов администрирования. Преимуществами пакета являются высокая производительность и масштабируемость, наличие средств формирования специальных оптимизированных запросов в зависимости от типа источника данных, наличие развитых средств вычислений, службы кэширования и кластеризации.

Oracle Business Intelligence Standard Edition One – бюджетная версия бизнес-аналитики Oracle BI, основанная на Sieble Analytics, поставляемая с определенными ограничениями, например с ограниченным количеством допустимых пользователей.

Oracle Business Intelligence Standard Edition – продукт бизнес-аналитики Oracle BI, содержащий Oracle Discovery, Oracle Reports (первоначальные инструменты Oracle BI).

Основой Oracle BI Enterprise Edition и Oracle BI Standard Edition One является Oracle BI Server и Oracle BI Presentation Server. Oracle BI Server представляет собой хорошо масштабируемый, высокопроизводительный сервер запросов и анализа, эффективно интегрирующий данные из множества реляционных, неструктурированных, OLAP и готовых приложений-источников, разработанных как Oracle, так и другими производителями. Oracle BI Presentation Server выполняет соединение с Oracle BI Server и предоставляет пользователям каталог аналитик, отчетов и информационных панелей, которые могут быть использованы для анализа данных.

В состав Oracle BI также входят следующие инструменты. Охарактеризуем их.

Инструмент Oracle BI Interactive Dashboard представляет собой интерактивные информационные панели с широкими функциональными возможностями, построенные в веб-архитектуре и отображающие персонализированную информацию, которая помогает пользователям принимать точные и эффективные решения. В Oracle BI Interactive Dashboards реализованы механизмы, которые обеспечивают неограниченную детализацию отчетных показателей.

Oracle BI Answers – это мощный инструмент для выполнения произвольных запросов и анализа в веб-интерфейсе.

Oracle BI Publisher – хорошо масштабируемый сервер формирования отчетов, позволяющий генерировать отчеты в разных форматах на основе данных из множества источников и рассылать их по различным каналам.

Ключевым преимуществом Oracle BI является функционирование в гетерогенной среде, т. е. возможность выборки и анализа информации из разнородных источников данных (СУБД Oracle, MS SQL Server, IBM DB2 UDB, Excel, XML, SAP и т. д.). Также Oracle Business Intelligence:

- обеспечивает высокую масштабируемость и производительность;
- является интегрированным набором дополняющих друг друга компонент;
- использует единый набор метаданных для всех компонент;
- предоставляет единый веб-интерфейс;
- позволяет работать в режиме оффлайн.

3.2. КЛЮЧЕВЫЕ ПРИНЦИПЫ И АРХИТЕКТУРА ORACLE BUSINESS INTELLIGENCE SUITE ENTERPRISE EDITION

Архитектура Oracle Business Intelligence высокого уровня обеспечивает доступ к данным через два основных сервера и семантическую модель и является четырехуровневой. Рассмотрим данную архитектуру с точки зрения конечного пользователя, который запрашивает информационную панель с бизнес-информацией.

1. Веб-браузер запрашивает информационную панель с данными, состоящую из аналитик, опубликованных отчетов и другого содержимого BI.

2. Oracle BI Presentation Server получает этот запрос и преобразовывает запросы для аналитик и отчетов в логические SQL-запросы. Далее эти логические запросы передаются серверу Oracle BI Server.

3. Oracle BI Server принимает эти логические SQL-запросы, написанные по содержащейся в репозитории Oracle BI Repository семантической модели, и преобразовывает их в SQL- или MDX-запросы[7], которые отправляются к базовым источникам данных.

4. Базовые источники данных обрабатывают SQL- или MDX-запросы и возвращают результаты серверу Oracle BI Server.

5. Oracle BI Server возвращает набор данных серверу Oracle BI Presentation Server. В тех случаях, когда для выполнения запроса требуется более одного источника данных, Oracle BI Server способен объединить нескольких наборов данных в один результирующий набор.

6. Oracle BI Presentation Server возвращает результаты конечному пользователю в виде аналитик, опубликованных отчетов, информационных панелей и других компонент бизнес-аналитики.

В отличие от многих других инструментов бизнес-аналитики, которые объединяют представление данных с генерацией запроса в одном сервере, Oracle Business Intelligence разделяет эти функции на два различных сервера: Oracle BI Server и Oracle BI Presentation Server.

Oracle BI Server обеспечивает одновременное подключение к гетерогенным источникам данных, выполнение вычислений и объединение данных, доступ к семантической модели и уровень защищенности.

[7] MDX (Multi-Dimensional eXpressions – язык многомерных выражений) является языком запросов, используемым для извлечения данных из многомерных баз данных.

Oracle BI Presentation Server соединяется с Oracle BI Server и обеспечивает пользователя каталогом аналитик, отчетами и информационными панелями, которые они могут использовать для анализа данных.

Oracle Business Intelligence 11g наряду с основными серверами Oracle BI Server и Oracle BI Presentation Server использует и другие серверы: Oracle BI Cluster Controller, Oracle BI Java Host, Oracle BI Scheduler.

Oracle BI Cluster Controller обеспечивает для Oracle BI Presentation Server центральную точку доступа, балансировку нагрузки, отказоустойчивость и другие сервисы кластера, когда в кластере работают два или более сервера BI.

Oracle BI Java Host работает вместе с Oracle BI Presentation Server для обеспечения возможности подключения к Java-задачам и серверу формирования отчетов Oracle BI Publisher, основанному на Java, а также для поддержки создания диаграмм.

Oracle BI Scheduler используется для планирования и автоматизации производства и распространения аналитик, а также для автоматизации задач рабочего процесса бизнес-аналитики.

Эти три сервера в совокупности с серверами Oracle BI Server и Oracle BI Presentation Server называют в терминологии Oracle Business Intelligence 11g Release 1 системными компонентами. Они запускаются непосредственно на платформе хоста и являются исполняемыми файлами операционной системы, написанными на языках производных от языка C.

Связь между браузером конечного пользователя и информационными панелями, аналитиками и отчетами, предоставляемыми Oracle BI Presentation Server, создается при запуске Java-приложения Oracle BI Analytics Plug-In на сервере Java-приложений, которое направляет входящие запросы на Oracle BI Presentation Server.

Системные компоненты Oracle Business Intelligence 11g Release 1 дополнены технологиями Oracle Fusion Middleware на основе Java, базирующимися на сервере приложений Oracle WebLogic Server. Oracle Business Intelligence 11g Release 1 использует технологии Fusion Middleware и WebLogic при решении вопросов безопасности и аутентификации, системного администрирования, подключения к внешним приложениям и процессам, управления процессами компонентов системы, высокой доступности.

Логическая архитектура Oracle Business Intelligence 11g Release 1 называется доменом Oracle BI и состоит из следующих элементов:

- домен Oracle BI (Oracle BI domain) – это полный набор Java-компонент и не только (преимущественно C++ компонент), которые

в совокупности с репозиториями, каталогами и файлами конфигурации, использующими эти компоненты, образуют единую среду Oracle BI;

• домен WebLogic (WebLogic Domain) отвечает за размещение Java-компонент в архитектуре. В соответствии с архитектурой Oracle Business Intelligence 11g Release 1 каждый домен Oracle BI имеет один домен WebLogic. В свою очередь, домен WebLogic имеет один Admin Server и один или более Managed Server, который содержит Java-3 компоненты Oracle Business Intelligence.

• сервер WebLogic Server Admin (WebLogic Server Admin Server) – это сервер JEE[8], который содержит виртуальную машину Java Virtual Machine, предназначенную для мониторинга и управления системой;

• сервер WebLogic Managed Server (WebLogic Server Managed Server) – это сервер JEE, с помощью которого выполняется размещение Java-приложений, используемых Oracle Business Intelligence;

• сервер WebLogic Server Node Managed используется для запуска, остановки, перезапуска и мониторинга сервера Managed Server или нескольких серверов Managed Server, если WebLogic Server работает с кластеризацией;

• сервер Oracle Process Manager and Notification Server (OPMN) используется для запуска, остановки, перезапуска и составления отчетов о состоянии отслеживаемых приложений системных компонент;

• компоненты Java (Java Components) представляют собой Java-приложения, такие как Oracle BI Analytics Plug-In, Action Service и Oracle BI Office, которые работают отдельно от рассмотренных ранее системных компонент;

• системные компоненты (System Components) – это совокупность серверов Oracle BI Server, Oracle BI Presentation Server, Oracle BI Cluster Controller, Oracle BI Java Host, Oracle BI Scheduler;

• консоль WebLogic Server Administration (WebLogic Server Administration Console) – это приложение, которое запускается на Admin Server и используется для управления сервером WebLogic Server;

• компонент Oracle Enterprise Manager Fusion Middleware Control используется для управления системой Oracle Business Intelligence через один или несколько узлов кластера;

• поддержка схем базы данных (Supporting Database Schemas) – это элемент, созданный с применением Oracle Fusion Middleware Repository

[8] JEE (Java Enterprise Edition) – это продвинутый (максимальный) набор инструментов, используемый большими компаниями для построения сложных Java-приложений с высокой производительностью, красивым интерфейсом и возможностью работы по сети.

Creation Utility, содержит реляционные таблицы, используемые для хранения дополнительных метаданных Oracle Business Intelligence и метаданных (BIPLATFORM), используемых Oracle Fusion Middleware's Metadata Services (MDS).

3.3. Источники данных Oracle Business Intelligence Server

Oracle BI не хранит данные, а использует слой метаданных для создания «виртуальной многомерной модели» на основе одного или более источников данных, а затем генерирует SQL- или MDX-запросы на выборку из источников данных и предоставляет их пользователю. В то время как Oracle BI может быть настроен для хранения кэша данных, чтобы обеспечить более быстрое отображение результатов, запросы, которые он генерирует, отправляются непосредственно в источник данных. Из этого следует, что исходные источники данных должны быть оптимизированы для запросов настолько, насколько это возможно. Кроме того, Oracle BI Server может также воспользоваться любой аналитической функциональностью, доступной в конкретном источнике данных, чтобы обеспечить более эффективную обработку путем передачи вычислений в источник данных. Именно поэтому с целью обеспечения быстрого доступа к данным рекомендуемым оптимальным источником для Oracle BI будет хранилище корпоративных данных, работающее на платформе базы данных, такой как Oracle Database Enterprise Edition, дополненное или расширенное с помощью OLAP – Oracle Essbase или OLAP Option для Oracle Database Enterprise Edition. Oracle BI также имеет возможность подключаться к более чем одному источнику данных и объединять результаты каждого из них в единый набор данных, предоставляя возможность создавать «виртуальные» хранилища данных, состоящие из данных, полученных в реальном времени из отдельных баз данных, которые могут быть хранилищами данных, витринами данных, OLTP-базами данных, источниками нереляционных баз данных, таких как OLAP-серверы, файлы, электронные таблицы Microsoft Excel. Работа с такими «объединенными» данными возможна благодаря наличию двух ключевых особенностей, предоставляемых сервером Oracle BI Server:

- семантической модели, которая может создавать метаданные на основе нескольких источников данных от разных вендоров, предоставляя единое унифицированное представление данных, не зависящее от источников данных;

• способности Oracle BI Server генерировать собственные оптимизированные запросы для каждого источника данных и комбинировать возвращаемый результат в один результирующий набор.

Семантическая модель Oracle BI предназначена:

• для предоставления данных предприятия в виде логической модели;

• отображения этой логической модели в источник данных, используемый на предприятии;

• предоставления персональных представлений для подмножеств пользователей с доступом только к тем данным, которые им необходимы.

Семантическая модель Oracle BI имеет три различных уровня. Охарактеризуем их.

Физический уровень. Этот уровень содержит метаданные о физических базах данных или других источниках данных, который предоставляют данные семантический модели.

Уровень бизнес-модели и отображения. Этот уровень содержит логическую модель, определенную для бизнеса.

Презентационный уровень. Этот уровень предоставляет персонализированные подмножества логической модели, предназначенные для различных групп пользователей.

Данные, начиная с физического уровня, передаются через семантическую модель посредством сопоставлений с бизнес-моделью (уровнем отображения) и через уровень представления поступают к конечным пользователям. Семантическая модель определяется и поддерживается средством Oracle BI Administration, а хранится в репозитории Oracle BI Repository.

Основным средством разработки решений бизнес-аналитики Oracle является инструмент Oracle BI Intelligence Administration, основанный на Microsoft Windows и используемый для определения и поддержания слоя бизнес-метаданных, т. е. репозитория Oracle BI (Oracle BI Repository); создания отчетов и аналитик.

Кроме того, инструмент Oracle Business Intelligence Administration используется для управления соединениями с источниками данных; определения политики безопасности, которая контролирует доступ пользователей к данным; выполнения задач администрирования: определение переменных, управление кэшированием, проверка статуса кластера.

Начиная с версии Oracle Business Intelligence 11g, некоторые задачи системного администрирования, ранее выполняемые инструментом Oracle Business Intelligence Administration, решаются с использованием

Oracle Enterprise Manager Fusion Middleware Control. Это задачи включения/отключения кэширования, установки размера кэша, управления статусом системных компонент в кластере.

Другими инструментами, которые Oracle Business Intelligence 11g предоставляет разработчикам, являются Catalog Manager, Oracle Enterprise Manager Fusion Middleware Control, Oracle WebLogic Server Administration Console. Catalog Manager – это Java-приложение, используемое для управления каталогом отчетов, информационными панелями и другими объектами бизнес-анализа. Oracle Enterprise Manager Fusion Middleware Control предназначен для администрирования платформы бизнес анализа. Консоль Oracle WebLogic Server Administration Console контролирует функциональность сервера приложения WebLogic Server.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Что представляет собой Oracle Business Intelligence? Каковы преимущества Oracle Business Intelligence?
2. Дайте характеристику инструментам Oracle Business Intelligence.
3. Из каких элементов состоит архитектура Oracle Business Intelligence?
4. Что является оптимальным источником данных для Oracle Business Intelligence?
5. Что представляет собой семантическая модель Oracle Business Intelligence?
6. Дайте характеристику уровням семантической модели Oracle Business Intelligence.

4. СОЗДАНИЕ РЕПОЗИТОРИЯ ORACLE BUSINESS INTELLIGENCE

4.1. ПОНЯТИЕ РЕПОЗИТОРИЯ ORACLE BUSINESS INTELLIGENCE. АРХИТЕКТУРА РЕПОЗИТОРИЯ ORACLE BUSINESS INTELLIGENCE: ФИЗИЧЕСКИЙ УРОВЕНЬ, ЛОГИЧЕСКИЙ УРОВЕНЬ, УРОВЕНЬ ПРЕДСТАВЛЕНИЯ

Репозиторий Oracle BI (Oracle BI Repository) – это хранилище метаданных, содержащее логические модели, с которыми работают пользователи при создании аналитик и отчетов (analyses and reports). Репозиторий представляет собой одну или несколько предметных областей, состоящих из таблиц, столбцов, иерархий, которые сопоставляются в репозитории с исходными источниками, предоставляющими данные для отчетов. В основном репозиторий используется сервером Oracle BI Server, чтобы перевести входящие логические SQL-запросы в физические SQL-, MDX- или другие запросы к исходным источникам данных.

Репозиторий Oracle BI используется вместе с другим хранилищем метаданных – каталогом Oracle BI Presentation Server Catalog. Каталог содержит определение аналитик, информационных панелей, агентов, критериев и других объектов BI, которые пользователи создают для анализа данных.

Таким образом, Oracle Business Intelligence можно рассматривать как систему, содержащую три уровня данных и метаданных:

- каталог Oracle BI Presentation Server Catalog, который содержит определения отчетов;
- репозиторий Oracle BI Server Repository, содержащий логическую модель для заполнения отчета;
- базы данных, кубы OLAP и другие источники данных, которые фактически содержат данные.

Метаданные из репозитория Oracle BI в основном применяются иструментом Oracle BI Answer, который предназначен для выполнения произвольных запросов и анализа на веб-сайте Oracle Business Intelligence. Кроме того, указанные метаданные используются инстру-

ментом Oracle BI Publisher и сторонними инструментами отчетности, которые взаимодействуют с репозиторием посредством клиента Oracle BI ODBC[9].

Репозиторий обычно хранится в отдельном двоичном файле с расширением «.rpd» исходного каталога промежуточного программного обеспечения:

[middleware_home]\instances\instance1\bifoundation\OracleBIServerComponent\coreaaplication_obis1\repository.

Файл репозитория имеет расширение «.rpd», доступ к нему возможен только с аутентификацией с использованием пароля. Чтение файла возможно несколькими процессами одновременно, но только один может осуществлять запись в файл в любой момент времени.

Репозиторий Oracle BI содержит три уровня метаданных, т. е. имеет трехуровневую архитектуру.

Физический уровень (Physical Layer). На этом уровне описываются физические базы данных, схемы, таблицы, столбцы, соединения, ключи, которые составляют реляционное хранилище данных, и прочие метаданные для OLAP и других нереляционных источников данных.

Уровень бизнес-модели и отображения (Business model and mapping layer). Этот уровень содержит определения одной или более бизнес-моделей, состоящих из логических таблиц, столбцов, измерений (dimensions) вместе с источниками таблиц, которые содержат отображение логических объектов в бизнес-модель и физических объектов на физический уровень.

Презентационный уровень (Presentation Layer). Этот уровень содержит одну или более предметных областей, состоящих из презентационных таблиц, столбцов, иерархий и папок.

Каждый уровень метаданных строится на основе предыдущего уровня, и, таким образом, создается семантическая модель, которая описывает в бизнес-терминологии данные, поступающие из исходной базы данных. Все эти метаданные хранятся в репозитории, который обычно создается и управляется с помощью инструмента администрирования Oracle BI Administration.

Рассмотрим ***физический уровень*** метаданных репозитория Oracle BI. Физический уровень содержит информацию о физическом источнике данных, который предоставляет данные для отчетов. Как правило, ис-

[9] ODBC – соединение, через которое осуществляется основное подключение к серверу Oracle Business Intelligence Server внутри системы Oracle Business Intelligence.

пользуют функцию импорта метаданных Import Metadata в инструменте Oracle BI Administration для чтения метаданных таблицы и объекта из исходной базы данных на физический уровень. С целью оптимизации управления эти метаданные могут быть изменены путем их добавления, удаления или корректировки ключей и соединений, создания псевдонимов таблиц.

Источники данных, определенные на физическом уровне, бывают трех основных типов. Охарактеризуем их.

Реляционные источники данных – это источники данных, в которых метаданные представлены с помощью реляционных структур, таких как таблицы, столбцы, соединения и ключи.

Многомерные источники данных – это специальный тип данных, используемый для таких источников данных, как Oracle Essbase, Microsoft Analysis Services, SAP B/W и др.

Файлы, *XML*, *ADF или иные источники данных*.

Реляционные и подобные реляционным источники данных на физическом уровне описаны ниже:

1) база данных. Представляет собой общий контейнер для метаданных, связанный с физической базой данных. Он содержит определение типа базы данных (например, вендора, название, версию) вместе с поддерживаемым базой данных списком функций, которые сервер BI Server может использовать при создании запросов для этого типа базы данных;

2) пул соединений. Соединения с базой данных выполняются через пул соединений, который объединяет отдельные сеансы и способствует масштабируемости. Определение пула соединений включает имя пользователя, пароль и другие параметры соединения, а также количество одновременных подключений, используемый интерфейс вызова, сведения о любых сценариях, которые будут выполняться после подключения, а также о том, как BI Server физически подключается к базовому источнику данных;

3) физическая схема. В некоторых реляционных базах данных физические таблицы организованы в схемы для обеспечения физического разделения различных наборов данных приложений. Физические схемы могут быть настроены в базе данных для зеркалирования этой организации и предоставления префиксов имени схемы для физических таблиц;

4) физический каталог. Это необязательные объекты, которые используются для группировки наборов схем или в случае таких баз данных, как Microsoft SQL Server, – для группировки самих физических таблиц;

5) физическая таблица. Представляют собой фактические таблицы данных базы данных-источника. Эти таблицы могут быть сконфи-

гурированы либо как физические таблицы, полученные в результате выборки (аналогично представлениям базы данных, но определенных в репозитории, а не в словаре базы данных), либо как хранимые процедурные таблицы, которые обычно используют источниками данных Microsoft SQL Server;

6) псевдоним таблицы. Задают альтернативные имена для одного физического объекта базы данных. Создаются псевдонимы по следующим причинам:

– необходимость добавления к имени физического объекта репозитория дополнительной информации об использовании;

– необходимость участия физической таблицы в нескольких соединениях;

7) папка физического отображения. Является необязательным способом группировки коллекций физических объектов, который используется, чтобы помочь в организации и сортировке метаданных. Физические папки отображения особенно полезны при работе с псевдонимами таблиц, когда требуется поместить эти псевдонимы в папку, чтобы сохранить их отдельно от имен физических таблиц;

8) физический столбец. Физические столбцы хранятся в физических таблицах или таблицах с псевдонимами. Они представляют собой фактические столбцы, которые хранят данные в таблицах базы данных.

Объекты физического уровня многомерных источников данных отличаются от рассмотренных выше объектов. Охарактеризуем их:

1) база данных. Представляет собой общий контейнер для кубических таблиц, измерений физического куба и других объектов, образовывающих многомерные источники данных;

2) физический каталог. Выступает в качестве контейнера для одной или нескольких кубических таблиц;

3) кубическая таблица. Содержит столбцы физического куба, измерения физического куба и другие объекты;

4) столбец физического куба. Представляет собой отдельные элементы данных, которые будут использоваться при формировании логических столбцов уровня бизнес-модели и отображения;

5) измерение физического куба. Содержится в таблице кубов и представляет измерения, которые используются для организации показателей (measures) в таблице кубов;

6) физическая иерархия. Одна или несколько физических иерархий могут содержаться в каждом измерении куба и представлять отдельные свертки, такие как день-месяц-год или день-период-финансовый год.

На физическом уровне требуется определить такие объекты, как ключи и соединения таблиц, т. е. задать ограничения. Эти объекты могут быть импортированы из исходной базы данных или созданы на физическом уровне. На практике наиболее часто наблюдается сочетание первого и второго подходов определения ограничений. Определение ключа создается в физической таблице и задает столбец, однозначно идентифицирующий каждую строку в таблице, т. е. первичный ключ. Первичный ключ физического уровня может не совпадать с первичным ключом таблицы исходной базы данных, но он должен однозначно определять каждую строку и его значения должны быть уникальными. Правильная и эффективная настройка ключей и соединений на физическом уровне – одна из самых важных задач. Администратор Oracle Business Intelligence должен в первую очередь создать все соединения между физическими таблицами и только после этого предоставлять пользователям возможность выполнять выборки из этих таблиц.

При определении новой базы данных на физическом уровне предлагается выбор из нескольких серверов баз данных разных поставщиков (Oracle, SAP, Microsoft, Teradata и др.) в соответствии с тем, как хранятся данные. При выборе определенного типа базы данных в процессе определения или импортирования метаданных новой базы данных на физический уровень фактически требуется выбрать:

- интерфейс вызова, такой как Oracle OCI, ODBC, или другой собственный сетевой протокол базы данных, который будет использовать сервер Oracle BI Server, чтобы извлечь данные и метаданные базы данных;
- список опций базы данных по умолчанию, который сервер Oracle BI Server будет использовать при построении запросов к физической базе данных.

Как правило, создание метаданных исходной базы данных физического уровня осуществляется с Import Metadata инструмента администрирования Oracle BI Administration, а затем вручную вносятся изменения в эти метаданные с целью добавления, корректировки ключей или объединений, а также для добавления псевдонимов, отображаемых папок и т. п. Если после первоначального импорта метаданных исходная схема базы данных изменится, возможно повторно импортировать свои физические метаданные базы данных и обновить физическую модель. Кроме того, допустимо добавлять или удалять таблицы, столбцы или другие объекты самостоятельно с физического уровня, но при условии, что любые производимые изменения правильно отражают физические структуры базы данных.

Уровень бизнес-модели и отображения выполняет две основные функции:

- определение логической измеримой бизнес-модели для системы бизнес-аналитики;
- сопоставление логических столбцов этой бизнес-модели с физическими столбцами на физическом уровне семантической модели.

Рассмотрим объекты репозитория.

1. Бизнес-модель. Это контейнер для набора логических таблиц, логических столбцов, источников логических таблиц и логических измерений. Бизнес-модель может сопоставляться более чем с одним физическим источником данных, а данные, отображаемые в бизнес-модель, могут поступать из источника данных на разных уровнях детализации;

2. Папки логического отображения. По аналогии с физическим слоем семантической модели логические объекты бизнес-модели могут быть организованы в папки логического отображения. Папки логического отображения могут быть полезны, когда имеется большое количество объектов бизнес-модели и необходимо организовать эти объекты таким образом, чтобы облегчить перемещение по этой бизнес-модели.

3. Логические таблицы. Эти «виртуальные» таблицы предоставляют набор данных, полученный из одной физической таблицы или нескольких таблиц посредством прямых отображений или с помощью выражений SQL. Наборы данных будут использоваться для создания аналитик. Различают три типа логических таблиц: таблицы фактов (содержат меры, такие как прибыль, доход, стоимость, маржа), таблицы измерений (содержат справочные данные, по которым анализируются меры, такие как клиенты, продукты, время, магазины), таблицы поиска (этот тип логических таблиц является нововведением Oracle BI 11g).

Для простых бизнес-моделей с одним физическим источником данных логическая таблица может отображаться как один-к-одному с соответствующей физической таблицей в базе данных. Для более сложных моделей можно отображать данные из сводных таблиц и допустимо использовать некоторые логические столбцы из дополнительных источников, таких как файлы, кубы OLAP или другие базы данных. Логические таблицы представляют разработчикам семантической модели способ создания упрощенного, четкого представления данных организации, даже если эти данные хранятся в более сложных нормализованных схемах базы данных.

Для логических таблиц, как и для физических, характерны ключи и соединения. Каждая логическая таблица, содержащая измеримые данные, должна обладать логическим ключом, который, как правило, совпадает с первичным ключом ассоциированной физической таблицы. Логические соединения подобны физическим, но они не указывают логические столбцы для соединения. Вместо этого сервер Oracle BI Server определяет, какой конкретный физический столбец соединяется с базовыми физическими таблицами на основе конкретных источников логических таблиц, которые используются для доступа к базовому источнику данных или, в случае многомерных источников, использует другие методы для возврата данных из факта и измерения.

4. Логические измерения. Определяют иерархии в логической таблице.

5. Источники логической таблицы. Являются сопоставлениями, которые отображают логические столбцы непосредственно на физические столбцы либо косвенно через выражения SQL.

Разбиение бизнес-модели на тематические наборы данных с возможностью дальнейшего изменения имен и способа организации таблиц в соответствии с конкретной аудиторией пользователей является целью третьего слоя семантической модели – ***презентационного уровня***. На уровне представления семантической модели могут присутствовать представленные ниже объекты.

6. Предметные области. Предметная область является контейнером для презентационных таблиц и других объектов уровня представления. Предметные области – это аналоги баз данных или файлов данных, которые выбираются пользователями при создании аналитик для работы с определенным набором таблиц и столбцов. Предметная область может содержать отдельную таблицу фактов вместе со связанной таблицей измерений или множественные факты и измерения. В действительности предметная область может содержать любой выбор таблиц, полученных из одной бизнес-модели, при условии, что все таблицы взаимосвязаны. В противном случае пользователи могут выбирать комбинации таблиц для своих отчетов, которые нельзя объединить вместе.

7. Презентационные таблицы. Представляют собой контейнеры для презентационных столбцов и презентационных иерархий. В большинстве случаев таблицы производятся полностью из соответствующих логических таблиц уровня бизнес-модели и отображения, но на самом

деле вы можете перетаскивать столбцы из любой логической таблицы в одной и той же бизнес-модели в таблицу представлений.

8. Презентационные столбцы. Это столбцы презентационных таблиц, которые создаются при перетаскивании логических столбцов бизнес-модели в презентационные таблицы. Презентационные столбцы Oracle Business Intelligence 11g могут быть двух типов: столбцы атрибутов и столбцы измерений. Столбцы атрибутов содержат столбцы из измеримых логических таблиц, к которым сервер Oracle BI Server применяет предложение GROUP BY и предложение DISTINCT. Столбцы измерений обычно являются числовыми столбцами из фактических логических таблиц, к которым сервер Oracle BI Server применяет агрегирующие (групповые) функции.

9. Презентационные иерархии. Такие иерархии создаются при перетаскивании логических измерений из уровня бизнес-модели и отображения в презентационную таблицу. Отдельные презентационные иерархии создаются для каждой иерархии в пределах логического измерения. Включаясь в аналитику, они образуют столбец нового типа, называемый иерархическим столбцом, который может использоваться наряду со столбцами-атрибутами и столбцами-показателями (measure) при анализе данных.

4.2. СОЗДАНИЕ РЕПОЗИТОРИЯ НА ОСНОВЕ РАЗНОРОДНЫХ ИСТОЧНИКОВ ДАННЫХ. ВЫЧИСЛЕНИЯ И ФУНКЦИИ РЕПОЗИТОРИЯ. ПРИМЕР СОЗДАНИЯ РЕПОЗИТОРИЯ ORACLE BUSINESS INTELLIGENCE

Репозиторий определяет данные, с которыми работают пользователи, связь данных с различными источниками, а также расчеты и аналитики, которые может произвести пользователь. Хорошо спроектированный репозиторий отражает представление пользователей о данных и позволяет им быстро найти требуемую информацию. Рассмотрим этапы создания репозитория на основе хранилища данных Oracle.

Этап 1. Создание репозитория и импортирование исходных данных.

1. Откройте инструмент Oracle BI Administration: в меню кнопки Start выберите пункт Programs/Oracle Business Intelligence/BI Administration. В окне Oracle BI Administration выберите из главного меню пункт File/New Repository.

2. В диалоговом окне Create New Repository укажите следующие значения:

Create Repository: Binary

Name: GCBC_Repository.rpd (имя репозитория)

Location: d:\bi

Import Metadata: Yes

Repository Password: Repository1

Retype Password: Repository1

Если указанная в строке Location директория не существует, то переход к следующему диалоговому окну невозможен.

3. В диалоговом окне Create New Repository – Select Data Source выберите из раскрывающегося списка Connection Type тип соединения для исходной базы данных, укажите параметры соединения:

Connection Type: Oracle OCI10g/11g

Data Source Name: orcl (имя БД)

User Name: gcbc_sales

Password: password

4. В диалоговом окне Create New Repository – Select Metadata Types выберите типы метаданных, которые требуется импортировать (таблицы, первичные ключи, внешние ключи).

5. В открывшемся диалоговом окне Connection Pool щелкните кнопку Finish.

После выполнения первого этапа создания репозитория импортированные данные отображаются на физическом уровне семантической модели.

Этап 2. Создание псевдонимов для импортированных таблиц и папок физического отображения.

Псевдонимы являются альтернативными именами объектов физического уровня, которые используются для более удобного взаимодействия с объектами.

1. На физическом уровне выберите из контекстного меню таблицы команду New Object/Alias. В диалоговом окне Physical Table введите имя псевдонима.

2. Повторите эту последовательность действий для всех таблиц, которым требуется создать псевдонимы.

3. В контекстном меню имени базы данных (в рассматриваемом примере база данных имеет имя orcl) выберите команду New Object/Physical Display Folder, укажите имя папки и перетащите в нее созданные псевдонимы.

4. Перетаскивание псевдонимов в папку отображения копирует, а не перемещает их. Для того чтобы отобразить только эти псевдонимы в этой папке, выберите в меню Tools/Options, а затем выберите вкладку Repository. Установите флажок Show tables and dimentions only under display folders и нажмите OK. Затем вернитесь к физическому уровню и убедитесь, что псевдонимы отображаются только под папкой отображения.

В дальнейшем работа будет осуществляться только с псевдонимами, а не с первоначально импортированными физическими таблицами.

Этап 3. Создание первичных и внешних ключей физической модели.

В случае когда задание первичных и внешних ключей в исходной базе данных аналогично определению этих ключей в семантической модели, то допускается импортирование этих объектов на физический уровень в процессе импортирования метаданных. Если же ключи исходной базы данных некорректны с точки зрения их использования в семантической модели, то допускается создание ключей вручную.

1. Выберите из контекстного меню таблицы или псевдонима команду Properties, затем вкладку Keys. В столбце Key Name введите имя ключа, в столбце Column выберите столбец таблицы, который определяется как первичный ключ.

2. Внешние ключи можно создать с помощью диалогового окна Properties либо в графическом режиме Physical Diagram. В первом случае выберите из контекстного меню таблицы или псевдонима команду Properties, затем вкладку Foreign Keys и щелчком по кнопке Add откройте диалоговое окно Physical Foreign Keys. Чтобы задать внешние ключи в режиме Physical Diagram выберите все требуемые таблицы и псевдонимы на физическом уровне и вызовите их контекстное меню, в котором выберите команду Physical Diagram / Selected Object(s) Only.

Этап 4. Создание бизнес-модели, структуры логических таблиц и добавление логических столбцов.

После создания начальной физической модели можно приступать к созданию соответствующей бизнес-модели. Цель создания бизнес-модели заключается в создании измеримой модели, состоящей из фактических таблиц, содержащих показатели (measures) вместе с измеримыми таблицами, содержащими атрибуты. К этим таблицам впоследствии будут добавлены вычисления, иерархии и другая информация, с которой будет взаимодействовать пользователь при проведении аналитики.

Первоначально создаются пустые фактические и измеримые логические таблицы, которые затем объединяются, чтобы Oracle BI

Server мог распознать, является логическая таблица фактической или измеримой.

1. Используя инструмент Oracle BI Administration, выведите уровень бизнес-модели и отображения в семантической модели. Из контекстного меню рабочей области выберите команду New Business Model.

2. В открывшемся диалоговом окне Business Model введите имя новой модели.

3. После того как модель будет создана, выберите из контекстного меню этой модели команду New Object/Logical Table.

4. В диалоговом окне Logical Table введите имя логической таблицы. Выполните этапы 3 и 4, чтобы создать остальные логические таблицы вашей бизнес-модели, сначала для фактических таблиц, затем для измеримых.

5. Добавьте к созданным логическим таблицам столбцы. Начиная с фактической таблицы, найдите столбец на физическом уровне, который требуется отобразить, а затем перетащите его из физической модели в логическую фактическую таблицу. При добавлении столбцов в фактическую таблицу не перетаскивайте столбцы-идентификаторы (ID columns), так как Oracle BI Server выполняет соединение в фоновом режиме на основе ключей и соединений, настроенных в физическом слое семантической модели. Убедитесь, что вы перетаскиваете столбцы ключевых столбцов (dimension key columns), так как они понадобятся вам позже, чтобы создать логические ключи таблицы. В некоторых ситуациях вам могут потребоваться исходные столбцы для логической таблицы более чем из одной физической таблицы.

6. После добавления всех столбцов в логическую модель переименуйте их так, чтобы их имена стали более понятными для пользователя, а не совпадали с системными именами, которые использовались в исходной базе данных.

7. Далее требуется установить правило агрегации по умолчанию для столбцов показателей (measure columns) в логической фактической таблице. Для этого дважды щелкните каждый столбец измерения (measure column) в таблице фактов, чтобы отобразить диалоговое окно Logical Column, выберите вкладку Aggregation и требуемый тип агрегации в раскрывающемся списке Default Aggregation Rule.

Этап 5. Создание логических ключей и соединений.

На уровне бизнес-модели и отображения необходимо удостовериться, что все логические таблицы, использованные для измеримых данных (dimension data), имеют соответствующие определенные для них ключи. Если ключи уже созданы на физическом уровне, то они ото-

бразятся при перетаскивании соответствующих физических столбцов в логическую таблицу. Если на физическом уровне ключи не определены или необходимо создать логический ключ на основе другого столбца, то следует выполнить приведенные ниже действия.

1. Из контекстного меню таблицы, для которой создается логический ключ, выберите команду Properties. На вкладке Keys введите имя логического ключа и выберите соответствующий логический столбец.

2. Выделите все созданные логические таблицы. Выберите из контекстного меню Business Model Diagram/Selected Table(s) Only. Соедините таблицы фактов с таблицами измерений. Так как соединения являются логическими, нет необходимости в определении столбцов, по которым будет выполняться соединение, поскольку эти столбцы определяются сервером Oracle BI Server в момент запуска запроса.

Таким образом, получена базовая бизнес-модель, состоящая из таблицы логических фактов и одного или нескольких логических измерений. Убедитесь, что бизнес-модель имеет символ хэша «#» у таблицы фактов; если же символ хэша «#» находится у таблиц измерений, то логические соединения созданы в неправильном порядке.

3. Для изменения порядка следования столбцов в логической таблице выберите в контекстном меню Properties, на вкладке General выберите логический столбец и переместите его, используя кнопки со стрелками вверх или вниз.

Этап 6. Определение логических измерений и иерархий.

1. Используя инструмент Oracle BI Administration уровня бизнес-модели и отображения модели из контекстного меню бизнес-модели, для которой требуется создать логические измерения, выберите New Object/Logical Dimension/Dimension With Level-Based Hierarchy.

2. В окне Logical Dimension введите имя логического измерения.

3. Создание уровней иерархии для логического измерения выполняется через контекстное меню этого логического измерения с помощью команды New Object/Logical Level. Первый уровень иерархии будет основным, поэтому следует дать ему имя типа All < Имя_измерения> (например, All Products) и установить флажок Grand Total Level.

4. Вызовите контекстное меню основного уровня иерархии и выберите New Object/Child Level. В диалоговом окне Logical Level введите имя уровня и закройте это окно. Затем вызовите контекстное меню созданного уровня и повторите описанные действия для создания всех уровней иерархии.

5. После создания уровней иерархии требуется связать логические столбцы с этими уровнями. Для этого перетащите столбцы из логи-

ческой таблицы, связанной с измерением, на соответствующий уровень, вызовите контекстное меню и выберите команду New Logical Level Key... .

Каждый логический столбец, который определяется в качестве ключа логического уровня, должен содержать уникальные значения. В противном случае Oracle BI Server может предоставить неверный результат для запроса.

Этап 7. Определение вычисляемых и других производных столбцов.

В дополнение к логическим столбцам, которые создают путем перетаскивания физических столбцов в логическую таблицу, можно создать дополнительные столбцы, выводя их значения из других логических столбцов в бизнес-модели.

1. В контекстном меню таблицы, в которой будет создан новый столбец, выберите команду New Object/Logical Column.

2. В диалоговом окне Logical Column на вкладке General введите имя столбца.

3. На вкладке Column Source выберите переключатель «Derived from existing columns using an expression». С помощью кнопки Edit Builder откройте диалоговое окно Expression Builder.

4. Выбирая соответствующие логические таблицы, столбцы, операторы, составьте выражение, с помощью которого будут вычисляться значения столбца.

5. После возвращения в диалоговое окно Logical Column составленное выражение будет указано на панели управления в диалоговом окне, а значения логического столбца будут вычисляться во время запроса, используя это выражение.

Этап 8. Представление бизнес-модели в качестве одной или более предметной области.

После того как бизнес-модель создана, необходимо опубликовать ее на уровне представления репозитория. Вы можете опубликовать бизнес-модель автоматически, создавая отдельные предметные области для каждой таблицы фактов и связанных с ними измерений, или создать пользовательскую предметную область, перетащив отдельные логические таблицы и столбцы на уровень представления.

Для того чтобы автоматически публиковать предметные области на основе таблиц фактов в бизнес-модели, щелкните правой кнопкой мыши бизнес-модель на уровне бизнес-модели и отображения семантической модели и выберите Create Subject Areas For Logical Starts And Snowflakes. Таким образом, будет создана отдельная предметная область на уровне представления для каждой таблицы фактов бизнес-модели.

С целью создания пользовательской предметной области на основе таблиц одной бизнес-модели выполните следующие действия:

1) с помощью инструмента Oracle BI Administration из контекстного меню уровня представления выберите команду New Subject Area;

2) в диалоговом окне Subject Area введите имя предметной области;

3) перетащите логические таблицы и логические столбцы, которые требуется добавить, из уровня бизнес-модели и отображения. Для переименования столбца или таблицы дважды щелкните мышью по имени соответствующего объекта.

Обратите внимание на то, что при перетаскивании логической таблицы из бизнес-модели на уровень представления также перемещаются логические измерения, связанные с этой логической таблицей.

Отделение слоя представления от бизнес-модели позволяет создать еще один уровень абстракции от исходных данных. Например, можно разбить существующую логическую таблицу на две или более таблицы представлений или создать таблицы представлений, содержащие столбцы, взятые из более чем одной логической таблицы.

Кроме того, можно вложить один набор презентационных таблиц в другой, чтобы создать набор подпапок для основной папки. Приведем последовательность действий, которую требуется выполнить для того, чтобы поместить каждый набор мер в свою отдельную папку для имеющейся таблицы фактов, содержащей как базовые (физически сопоставленные), так и производные (вычисленные) меры.

1. На уровне представления семантической модели перейдите к требуемой предметной области.

2. В контекстном меню этой предметной области выберите команду New Presentation Table.

3. В диалоговом окне Presentation Table введите имя для новой презентационной таблицы. В области Description введите «->». Повторите эти действия для других создаваемых подпапок.

4. Перетащите добавляемые столбцы из содержащей их презентационной таблицы или из соответствующей бизнес-модели.

5. Дважды щелкните мышью по предметной области уровня представления и выберите вкладку Presentation Tables, поместите две подпапки под презентационной таблицей, в которую требуется их вложить.

При возвращении в основной уровень представления в семантической модели вы увидите только эти две подпапки, перечисленные в основной папке. Однако при размещении репозитория онлайн и просмотре списка папок в Oracle BI Answers вы увидите папки с вложениями и столбцы представления, перечисленные в каждой подпапке.

Этап 9. Выполнение проверки согласованности на наличие ошибок и предупреждений.

Проверка репозитория на наличие ошибок выполняется с помощью Consistency Check Manager. Можно использовать Consistency Check Manager автоматически или вручную для одного объекта репозитория либо для всего репозитория. Для проверки глобальной согласованности в инструменте Oracle BI Administration выберите File/Save и ответьте Yes. В открывшемся окне Consistency Check Manager выберите категории проверяемых ошибок (Errors, Warnings, Best Practices). После устранения любых проблем вы можете сохранить репозиторий в файловой системе.

Для проверки согласованности вручную в инструменте Oracle BI Administration выберите один объект и воспользуйтесь командой Check Consistency из контекстного меню этого объекта либо выберите из главного меню File/Check Global Consistency, чтобы проверить все объекты в репозитории. В случае получения сообщения «Consistency check didn't find any errors, warnings or best practice violations» переходите к следующему этапу создания репозитория. В противном случае ознакомьтесь с обнаруженными ошибками, исправьте их и проверьте репозиторий повторно.

Этап 10. Предоставление репозитория в режиме онлайн пользователям.

Для того чтобы разместить репозиторий онлайн, воспользуйтесь средством Oracle Enterprise Manager Fusion Middleware Control, с помощью которого файл репозитория разворачивается в файловой системе сервера Oracle BI Server, активируются изменения, перезапускается компонент Oracle BI Server. Выполните следующие действия:

1) перейдите к URL-адресу, где расположен Fusion Middleware Control (например, http://obisrv1:7001/em);

2) введите имя пользователя и пароль учетной записи администратора;

3) на домашней странице Fusion Middleware Control перейдите в папку Business Intelligence и щелкните запись основного приложения;

4) перейдите на вкладку Deployment, затем на вкладку Repository. Заблокируйте конфигурацию системы для редактирования щелчком по кнопке Lock And Edit Configuration. Ни один другой администратор не сможет внести изменения в конфигурацию, пока вы не активируете свои изменения или не разблокируете конфигурацию;

5) перейдите в раздел Upload BI Server Repository и щелкните кнопку Browse, чтобы разместить файл репозитория на рабочей станции. В раз-

деле Repository Password введите пароль и подтвердите его в Confirm Password;

6) щелкните кнопку Apply, затем кнопку Activate Changes;

7) после получения сообщения «Activate Changes – Completed Successfully» перейдите на вкладку capacity Management, затем на вкладку Availability. В разделе System Components Availability выберите папку Oracle BI Servers и щелкните кнопку Restart Selected. Щелкните Yes после сообщения «Are you sure you want to restart the selected component?» и ожидайте завершения процесса перезапуска. По окончании процесса перезапуска появится сообщение «Restart Selected – Completed Successfully».

Рассмотрим подробно вопрос вычислений и функций репозитория. Существует несколько возможностей определения вычисления:

- если вычисление необходимо для специальной аналитики, то его можно определить в критерии аналитики;
- если несколько аналитик или разработчиков будут использовать вычисление, то его можно определить непосредственно в репозитории.

Вычисления в репозитории можно формировать из логических столбцов бизнес-модели или прямо из исходных столбцов семантической модели физического уровня. Сервер Oracle BI Server производит вычисления над данными следующим образом: по возможности Oracle BI Server преобразует ваши вычисления в SQL-функции, связанные с базами данных (*функция push-down*) либо сервер Oracle BI Server запрашивает исходные необработанные данные и сам выполняет вычисления (*функциональная компенсация*) в случае отсутствия у базы данных возможности выполнять вычисления напрямую или когда данные, используемые в вычислениях, поступают более чем из одной физической базы данных.

Рассмотрим реализацию функции push-down. Сервер Oracle BI Server использует форму языка запросов, называемую логический SQL. Отличие логического SQL от стандартного SQL состоит в отсутствии GROUP BY, ORDER BY, соединений, агрегирующих функций, так как они задаются в бизнес-модели репозитория.

Для того чтобы общий набор функций мог быть предоставлен для всех данных в семантической модели независимо от источника данных, сервер Oracle BI Server преобразует логические функции SQL, используемые в репозитории и аналитиках, в конкретные функции SQL и MDX, поддерживаемые источником данных. Локальный столбец, который запрашивает ранг продаж, будет использовать функцию

RANK (имя_столбца) логического SQL. Эта функция будет переведена в RANK () OVER (ORDER BY) при «спуске» (push down) в базу данных Oracle, Microsoft SQL Server или IBM. Возможность выполнения функции с «проталкиванием вниз» (function push-down) сервером Oracle BI Server определяется типом и версией базы данных, выбранной для физической базы данных, а также теми возможностями, для которых была настроена база данных. Эти сведения можно отобразить, щелкнув правой кнопкой мыши физическую базу данных в семантической модели и выбрав вкладку Features.

Сервер Oracle BI Server производит вычисления над данными способом функциональной компенсации, если исходная база данных имеет меньшую функциональность. В этом случае сервер Oracle BI Server запрашивает необработанную информацию из исходной базы данных, а затем сервер Oracle BI Server выполняет дополнительные вычисления в памяти. Такой подход позволяет Oracle Business Intelligence предоставлять одни и те же возможности вычислений, не зависящие от источника данных, поэтому пользователи могут создавать аналитики, охватывающие разные типы исходных баз данных, не беспокоясь о том, какие источники поддерживают различные типы вычислений. Все это происходит прозрачно для конечных пользователей, хотя разработчику нужно знать о дополнительной нагрузке.

В выражениях для вычислений, определяемых в репозитории, можно использовать либо логические столбцы, либо столбцы физического уровня семантической модели. Логические столбцы используются, когда вычисление определяется как часть свойств логического столбца. Физические столбцы используются, когда вычисления определяются в сопоставлении источника логической таблицы. Независимо от того, где создаются вычисления, они все равно будет перенаправлены на базовый источник данных (при условии, если источник данных поддерживает такие вычисления), но если в вычислениях используется более одного столбца, результаты могут быть агрегированы по-разному: логические вычисления выполняются после агрегации, а физические – до агрегации.

Oracle Business Intelligence предоставляет функции логического SQL, которые он преобразовывает в эквивалентные функции стандартного SQL или MDX, используемые каждым поддерживаемым источником данных. В случае когда источник данных не имеет эквивалентной функции, сервер Oracle BI Server запрашивает требуемые для выполнения вычислений данные и выполняет вычисления. Логический

SQL содержит математические, строковые функции, функции даты и времени, функции преобразования типа, агрегирующие (групповые) функции, функции временных рядов, функции поиска и др. Математические (ABS, MOD, LOG, TRUNCATE и др.) и строковые (CONCAT, LENGTH, LOCATE – аналог INSTR Oracle SQL и др.) функции логического SQL соответствуют функциям большинства SQL-баз данных, хотя для некоторых функций название или синтаксис могут отличаться. Синтаксис функций даты и времени логического SQL отличается от синтаксиса аналогичных функций большинства SQL-баз данных. Рассмотрим функции даты и времени логического SQL подробно.

CURRENT_DATE, CURRENT_TIME, CURRENT_TIMESTAMP (integer) возвращают текущую дату, текущее время, текущее время с дробными значениями секунд относительно системного времени соответственно.

DAYNAME (dateExpr), DAYOFWEEK (dateExpr), MONTHOFQUARTER (dateExpr), DAYOFQUARTER(dateExpr), MONTHNAME(dateExpr), SECOND(timeExpr), YEAR(dateExpr) извлекают элементы даты или времени с целью использования их в вычислениях.

TIMESTAMPADD(interval, intExpr, timestamp) и TIMESTAMPDIFF(interval, timestamp1, timestamp2) предназначены для добавления определенного количества дней, месяцев, секунд к данным или интервала и для вычисления количества интервалов между двумя датами соответственно.

Среди функций логического SQL присутствуют функции преобразования типа, функции работы с неопределенным значением NULL, функции, специфичные для Oracle BI, такие как использование переменных или переключение между столбцами.

IFNULL (expr, value) является аналогом функции NVL Oracle SQL.

CAST (expr| NULL as data_type) преобразовывает expr или NULL в тип данных, указанный в data_type.

VALUEOF (variable_name) – уникальная функция логического SQL, позволяет вычислять ссылку на значение, хранящееся в переменной репозитория. Например, VALUEOF (NQSESSION.home_store) возвращает значение, содержащееся в переменной home_store сеанса репозитория, которое обычно устанавливается с использованием блока инициализации в начале нового сеанса пользователя.

INDEXCOL (integer, expr_list) возвращает столбец или выражение в expr_list, начиная с 0 для первой записи в списке, на основе целочис-

ленного значения, переданного функции. Эта функция может быть полезна, например, когда необходимо динамически выбирать столбец из списка на основе значения в переменной, позволяя одной функции использовать набор столбцов с определенным столбцом, выбранным во время запроса (например, в ситуации, когда требуется отображать данные в долларах или фунтах в зависимости от кода валюты, установленного для пользователя). Для тех пользователей, которым требуются данные в долларах, переменная сеанса pref_currency устанавливается в 0, а для тех, которым требуются данные в фунтах, – в 1. Функцию INDEXCOL можно использовать для отображения корректного значения валюты на основе переменной сеанса pref_currency:

INDEXCOL (VALUEOF(NQ_SESSION.pref_currency), usd_amount, gbp_amount)

CHOOSE (expr1, expr2... exprN) выполняет аналогичную функцию с INDEXCOL, но выбирает столбец, который будет возвращен на основе первого в списке, который пользователь может просмотреть. Например, если имеются четыре столбца, содержащие итоговые данные продукта (product_total,), категорию (product_category), тип (product_type) и название продукта (product_name), а пользователь может просмотреть не все столбцы, а только те, на которые имеет привилегию, то функция следующего вида будет возвращать первый столбец в списке столбцов, на который пользователь имеет привилегию просмотра:

CHOOSE (product_total, product_category, product_type, product_name)

Агрегирующие функции логического SQL (RANK, TOPN, MAX, NTILE, MAVG и др.) преобразовываются в аналогичные функции SQL-баз данных (в случае, когда база данных поддерживает соответствующие функции).

В логическом SQL имеются три функции временных рядов:

- AGO (expr, time_dimention_level, periodoffset) используется для вычисления значения меры определенное количество периодов назад;
- TODATE (expr, time_dimention_level) рассчитывает значение периода до даты для меры;
- PERIODROLLING (measure, x, y, [, hierarchy]) вычисляет значение меры от смещения x до смещения y, причем x обычно является отрицательной цифрой для обозначения прошлого, а y – числом периодов в будущем.

Хотя логический SQL поставляется с большим количеством функций, включая те, которые обычно не встречаются в SQL-базе дан-

ных, в нем предусмотрена необходимость использования уникальной функции конкретной базы данных. Оценочные функции позволяют вызывать собственные функции SQL вместе с любыми требуемыми параметрами и использовать их для возврата скалярного значения, агрегируемого значения, аналитической функции базы данных или возврата логического значения.

EVALUATE ('db_function(%1...%N)' [AS data_type] [, column1,..., columnN]) используется со столбцами логической таблицы для применения функций, изменяющих значения столбца. Например, если требуется вернуть с третьего по пятый символы столбца Product Category, когда источником данных является база данных Oracle, то имеем EVALUATE ('SUBSTR(%1,%2,%3)', "Dim Products". "Product Category", 3, 5), где первый параметр функции EVALUATE содержит имя функции базы данных Oracle, а затем указаны заполнители для параметров функции: "Dim Products". "Product Category", 3, 5.

EVALUATE_AGGR ('db_agg_function (%1...%N)' [AS data_type] [, column1,..., columnN]) используется с показателями (measures) таблиц фактов и возвращает агрегированное значение, которое может быть использовано в предложении GROUP BY. Показатели (measures), определенные с помощью EVALUATE_AGGR, должны иметь правило агрегации по умолчанию, установленное в EVALUATE_AGGR, чтобы сервер Oracle BI Server знал, что внешняя функция обеспечивает агрегацию этой меры. Поэтому при моделировании репозитория эта функция может использоваться только с физическими столбцами и не может – с выражениями логическими столбцов. Например, функция, использующая встроенную функцию STDDEV базы данных Oracle для вычисления стандартного отклонения для измерения показателя (measure) дохода, имеет вид

EVALUATE_AGGR('STDDEV(%1)', "orcl"."GCBC_SALES"."Fact_SALES"."FCAST_REV_AMT")

EVALUATE_ANALYTICS ('db_function(%1...%N)' [AS data_type] [, column1,..., columnN]) используется для вызова аналитических функций базы данных. Например, следующая функция возвращает ранг для измерения дохода:

EVALUATE_ANALYTICS ('DENSE_RANK() OVER (ORDER BY %1)' AS INT, "Sales"."Fact Sales"."Revenue")

Oracle Business Intelligence 11g содержит два типа функций поиска:

1) LOOKUP (DENSE (lookupColumn, commaSepExpr)) используется для поиска, когда для каждого входного значения будет возвращено значение поиска (аналог INNER JOIN языка SQL);

2) LOOKUP (SPARSE (lookupColumn, alternateColumn, commaSepExpr) – применяется для поиска, когда не для каждого входного значения будет возвращено значение поиска (аналог LEFT OUTER JOIN языка SQL).

Функция LOOKUP используется как с физическими, так и с логическими столбцами в сочетании с логической таблицей, созданной в виде таблицы поиска. Таблицы поиска – это обычные логические таблицы. Однако обозначив их как поиск, вы устраняете необходимость в том, чтобы они играли роль таблиц фактов (с присоединением к ним других таблиц) или таблиц измерений (которые напрямую соединяются с таблицами фактов).

Системные функции логического SQL представлены функцией USER, которая возвращает имя пользователя, и функцией DATABASE, возвращающей имя предметной области по умолчанию.

Для автоматического определения вычислений Oracle BI предоставляет мастера Calculation Wizard. Рассмотрим пример использования Calculation Wizard. Пусть имеется бизнес-модель с таблицей фактов, содержащей четыре показателя (measures): Revenue (доход), Cost (стоимость), Sale Amount (количество продаж), Sale Amount Month Ago (количество продаж месяцем ранее). Показатель (measure) Sale Amount Month Ago определен с помощью функции временных рядов AGO. Используя мастер Calculation Wizard, определим стандартное отклонение, основанное на Sale Amount Month Ago и Sale Amount.

1. Для запуска Calculation Wizard выберите из контекстного меню логического столбца таблицы фактов (в рассматриваемом примере это столбец Sale Amount) команду Calculation Wizard.

2. В открывшемся диалоговом окне Calculation Wizard – Introduction щелкните кнопку Next.

3. Откроется диалоговое окно Calculation Wizard – Select Column. Для каждого выбранного столбца Calculation Wizard создаст набор производных мер, сравнивая их с показателями (measures) логического столбца таблицы фактов. Выберите показатели (measures), которые требуется сравнить с первым показателем (measure) (например, Sale Amount Month Ago), и щелкните кнопку Next.

4. Откроется диалоговое окно Calculation Wizard – New Calculation, в котором необходимо выбрать создаваемые вычисления из списка представленных:

• Change вычитает второй столбец из первого. Например, Sale Amount – Sale Amount Month Ago;

• Percent Change вычитает второй столбец из первого и возвращает значение в процентах. Например, 100*(Sale Amount – Sale Amount Month Ago)/Sale Amount Month Ago;

• Index выполняет деление первого столбца на второй. Например, Sale Amount/Sale Amount Month Ago;

• Percent выполняет деление первого столбца на второй и возвращает значение в процентах. Например,100*(Sale Amount / Sale Amount Month Ago).

5. В области слева задайте инструкции по обработке значений столбца, когда они могут быть NULL, 0 или отсутствовать (в рассматриваемом примере значения столбца Sale Amount Month Ago).

6. Щелкните кнопку Next, затем кнопку Finish. Затем в бизнес-модели будут созданы новые вычисления в соответствии со специальными инструкциями по обработке.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Дайте характеристику Oracle Business Intelligence.
2. Перечислите инструменты Oracle Business Intelligence.
3. Дайте характеристику основным компонентам архитектуры Oracle Business Intelligence.
4. Каковы назначение и уровни семантической модели Oracle Business Intelligence?
5. Дайте характеристику Oracle Business Intelligence.
6. Перечислите инструменты Oracle Business Intelligence.
7. Дайте характеристику основным компонентам архитектуры Oracle Business Intelligence.
8. Каковы назначение и уровни семантической модели Oracle Business Intelligence?
9. Дайте определение и характеристику репозитория Oracle Business Intelligence.
10. Перечислите этапы создания репозитория Oracle Business Intelligence на основе хранилища данных Oracle.

5. АНАЛИТИЧЕСКИЕ ОТЧЕТЫ И ИНФОРМАЦИОННЫЕ ПАНЕЛИ

5.1. СОЗДАНИЕ АНАЛИТИЧЕСКОГО ОТЧЕТА. ОПРЕДЕЛЕНИЕ КРИТЕРИЕВ АНАЛИТИЧЕСКОГО ОТЧЕТА. СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ АНАЛИТИЧЕСКОГО ОТЧЕТА

Oracle Business Intelligence поставляется с набором мощных, интуитивно понятных инструментов для создания отчетов, аналитик, информационных панелей и других видов визуализации, к которым можно получить доступ через веб-браузер, мобильное устройство, такое как Apple iPhone или iPad, либо через инструменты Microsoft Office.

Создание аналитических отчетов (аналитик) и их последующее внедрение в информационные панели является основным способом, с помощью которого пользователи запрашивают источники данных, отображаемые в системе бизнес-аналитики, и создают отчеты и веб-страницы, предоставляющие данные в виде таблиц, сводных таблиц, графиков и других средств визуализации. Информационные панели предоставляют средства для отображения одной или нескольких аналитик в соответствии с тематикой (такой как продажи, закупки и расходы, клиенты) с целью облегчить пользователям понимание данных, с которыми они работают.

Аналитика Oracle Business Intelligence – это отчет, который содержит выбранные столбцы, вычисления, фильтры и отображает результаты в одной или более таблицах, сводных таблицах, диаграммах и других средствах визуализации. Аналитика Oracle Business Intelligence состоит из следующих компонент:

- ***аналитические критерии***, которые определяют элементы данных в отчете;
- ***аналитические результаты***, отображающиеся в виде одного или нескольких представлений;
- ***опциональные подсказки***, которые используются для предварительного анализа данных, предоставляемых аналитике.

Каждая аналитика имеет одно или несколько интерактивных представлений. Например, можно определить список продаж в виде табли-

цы для определенного бренда продукта, а также одно или несколько графических представлений, которые показывают продажи, разбитые по брендам и по периоду времени. Хотя аналитики можно просматривать и обрабатывать в редакторе аналитик Oracle BI, обычно их предоставляют пользователям, встраивая их в информационные панели. Информационные панели объединяют аналитики, а также подсказки и другие инструменты для выбора данных и отображаются в веб-браузере пользователя или на мобильном устройстве.

Рассмотрим процесс создания новых аналитик и выбор столбцов предметной области для критериев аналитик. Критерий определяет итоговый набор исходных столбцов для аналитики, который может быть выбран непосредственно из столбцов в предметной области или являться результатом вычислений, выполняемых с использованием столбцов предметной области и формул. Аналитический критерий может возвращать все строки для определенного набора столбцов или задать возвращаемые строки с помощью создания фильтра.

Процесс создания аналитики начинается с выбора одного или нескольких столбцов из предметных областей и добавления их к аналитическому критерию. Предметные области определены в репозитории Oracle BI и содержат атрибуты, измерения и иерархии.

Для создания критерия аналитики необходимо выполнить следующие действия:

1) на веб-сайте Oracle Business Intelligence выберите New/Analysis;

2) в диалоговом окне Select Subject Area будут представлены список доступных предметных областей и в зависимости от назначенной пользователю роли опция создания аналитики с использованием прямого запроса к базе данных или простого логического SQL-запроса;

3) в окне редактора аналитик, которое содержит список предметных областей с таблицами и столбцами, добавьте столбцы в критерий, дважды щелкнув по ним;

4) сохраните аналитику, щелкнув по кнопке Save Analyses. В открывшемся диалоговом окне Save As выберите папку, в которую следует сохранить аналитику. Если выбрана папка My Folders, то аналитика будет доступна только ее создателю. В случае выбора папки Shared Folders другие пользователи смогут просматривать и запускать аналитику.

В созданный критерий аналитики можно добавить дополнительную предметную область. Для этого в окне редактора аналитик на вкладке Criteria щелкните кнопку Add/Remove Subject Area, выберите добавляемую предметную область, установив соответствующий ей флажок, и щелкните кнопку OK.

После создания критерия аналитики на вкладке Results выводятся данные для всех столбцов критерия, которые могут быть ограничены лишь условиями соединения источника данных или заданными в репозитории фильтрами.

Фильтр критерия аналитики задается следующим образом:

1) в окне редактора аналитик перейдите на вкладку Criteria;

2) в области Selected Columns установите указатель мыши на кнопку меню столбца (справа от столбца), для которого требуется задать фильтр;

3) в отобразившемся меню столбца выберите команду Filter;

4) в окне New Filter сформируйте условие фильтрации и щелкните кнопку OK.

Диалоговое окно New Filter содержит опцию Protect Filter, установив которую, предоставляется возможность защитить фильтр от переопределения подсказками информационной панели или пользователем, который применяет действие Navigate To Bi Content для перехода к связанному отчету.

В некоторых ситуациях задать фильтр проще не в диалоговом окне New Filter, а с помощью языка SQL. Для этого в диалоговом окне New Filter имеется опция Convert This Filter To SQL. При выборе этой опции и последующем щелчке по кнопке OK открывается диалоговое окно Advanced SQL Filter, которое позволяет ввести условие фильтрации на языке SQL.

Сортировка данных аналитики может быть задана в критерии аналитики. Для того чтобы выполнить сортировку по одному или нескольким столбцам, требуются следующие действия:

1) в окне редактора аналитик перейдите на вкладку Criteria;

2) в области Selected Columns установите указатель мыши на кнопку меню столбца (справа от столбца), по которому требуется выполнить сортировку;

3) в меню столбца выберите команду Sort/Sort Ascending для сортировки по возрастанию или Sort/Sort Descending для сортировки по убыванию. Для выполнения сортировки по нескольким столбцам выберите команду Sort/Add Ascending Sort или Sort/Add Descending Sort.

Для отмены сортировки по одному или нескольким столбцам из меню столбца выбирают команду Sort/Clear Sort или Sort/Clear All Sorts in All Columns.

Вычисления в критерии аналитики выполняются по аналогии с вычислениями в репозитории. Как и в репозитории, доступны математические, агрегирующие (групповые) , строковые функции, функции даты

и времени, преобразования типа, временных рядов и др. Вычисление в критерии аналитики создается следующим образом:

1) используя область Subject Area в редакторе аналитики на вкладке Criteria, добавьте столбец в область Selected Columns;

2) в меню столбца выберите команду Edit formula;

3) открывшееся диалоговое окно Edit Column Formula предоставляет возможность изменить формулу столбца.

Представление аналитики может быть создано для каждой аналитики. Причем аналитике может соответствовать несколько представлений. Различают следующие типы представлений аналитики:

- табличные представления (Table Views);
- заголовочные представления (Title Views);
- сводные таблицы (Pivot Table Views);
- графические представления (Graph Views).

Табличное представление добавляется по умолчанию в аналитику при включении столбцов в критерий аналитики. В табличных представлениях, помимо столбцов, могут содержаться промежуточные итоги и итоги по всей таблице. Табличное представление доступно для просмотра и редактирования на вкладке Results редактора аналитики. Например, возможно изменить порядок следования столбцов, выполнить сортировку по столбцу. Для перемещения столбца необходимо навести указатель мыши на заголовок столбца так, чтобы появилась четырехнаправленная стрелка, с помощью которой следует перетащить столбец. Для сортировки необходимо использовать кнопки-стрелки, которые отображаются при наведении мыши на заголовок столбца. Кроме этих операций, контекстное меню ячейки таблицы позволяет создать группы, вычисления, подвести итоги, исключить столбец.

Редактирование макета табличного представления выполняется с помощью трех элементов управления (Format Container, Edit View, Remove View from Compound Layout), расположенных в верхнем правом углу представления.

Format Container. Этот элемент управления используется для редактирования выравнивания, границ, отступов и т. д.

Edit View. С помощью этого элемента выполняется редактирование макета табличного представления. При щелчке по элементу управления Edit View табличное представление отображается в режиме предварительного просмотра в верхней части окна и панели Layout в нижней части, используя которую можно выполнить следующие действия:

- перетащить один или несколько столбцов в область Table Prompts, чтобы добавить раскрывающиеся списки продуктов, клиентов или дру-

гие атрибуты, которые затем будут использоваться для фильтрации представления таблицы;

● перетащить один или несколько столбцов в область Sections, чтобы создать табличные представления, по одному на каждое значение в столбце (столбцах);

● изменить порядок следования столбцов;

● перетащить один или несколько столбцов в область Excluded, чтобы оставить столбец в критерии, но исключить его из этого конкретного представления.

Для каждого столбца панели Layout имеется меню, доступное по расположенной справа от названия столбца кнопке More Options. Это меню содержит команды по форматированию, дублированию столбца, настройке агрегации и др.

Главная панель инструментов окна содержит инструменты распечатки, экспорта, обновления, переименования, дублирования табличного представления и т. д.

Табличное представление может отображать общий итог и промежуточные итоги, которые могут быть сгенерированы для столбцов-атрибутов в основном табличном представлении, в области Table Prompts, в области Sections. Для указания позиции отображения итогов какого-либо столбца щелчком по кнопке Edit View перейдите к панели Layout и наведите указатель мыши на кнопку Sum (∑), расположенную справа от названия столбца. В появившемся меню присутствуют следующие команды: None – итоги отсутствуют; After – отображение итогов после всех значений категории, по которой они подводятся; Format Labels – форматирование подписей итогов; Format values – форматирование значений итогов.

Remove View from Compound Layout. Этот элемент управления позволяет удалить представление из компоновки (compound layout), которая содержит одно или несколько представлений для отображения на информационной панели. Каждая аналитика имеет исходную компоновку, содержащую заголовочное представление, табличное представление или сводную таблицу, которую можно изменить, дополнить или оставить в исходном состоянии.

Заголовочные представления, которые по умолчанию являются пустыми, позволяют определить заголовок и иконку для аналитики, чтобы помочь пользователям идентифицировать аналитику при просмотре в информационной панели. По аналогии с табличными представлениями заголовочные представления имеют три элемента управления, расположенных в верхнем правом углу на вкладке Results редактора ана-

литики. При щелчке по элементу управления Edit View открывается диалоговое окно, в котором определяются следующие характеристики заголовочного представления:

• текст заголовка, который по умолчанию является сохраненным именем аналитики и может быть изменен при необходимости; задается в строке Title;

• необязательный логотип, который обычно выбирается из набора предопределенных логотипов и адрес которого в формате fmap:images/[image_filename] указывается в строке Logo;

• необязательный подзаголовок, который задается в строке Subtitle;

• возможность отображения времени начала аналитики в Started Time;

• URL документов, содержащих справочную информацию для данной аналитики, указывается в строке Help URL.

Логотипы, которые могут быть использованы для заголовочного представления, хранятся в двух местах на сервере Oracle BI Server:

• [middleware_home]/Oracle_BI1/bifoundation/web/app/res/s_blafp/images;

• [middleware_home]/user_projects/domains/bifoundation_domain/servers/bi_srver1/tmp/_WL_user/analytics_11.1.1/[folder_name]/war/res/s_blapf/images.

Сводные таблицы позволяют отображать показатели (measures) в двух или более измерениях. Например, можно предоставить данные о продажах по каждому филиалу и продукту за каждый месяц текущего года.

Сводные таблицы поддерживают все возможности табличных представлений (добавление общего итога и промежуточных итогов, раскрывающихся списков и др.) и обладают уникальной возможностью отображать содержимое в виде сводной таблицы. Сводные таблицы от обычных таблиц отличает отображение атрибутов измерения по строкам и столбцам, что делает их подходящими для выполнения OLAP-аналитики.

Базовая сводная таблица, в которой два измерения представлены в строках и столбцах, создается следующим образом:

1) в меню сайта Oracle BI выберите New/Analysis для создания новой аналитики, а также предметную область, для которой создается аналитика;

2) добавьте столбцы в критерий аналитики, который будут присутствовать в сводной таблице;

3) откройте вкладку Results, чтобы просмотреть полученную аналитику, компоновка которой будет содержать заголовочное представление

и сводную таблицу. По умолчанию в сводной таблице будут находиться все атрибуты измерений и иерархии в области строк и показатели (measures) в области столбцов.

Графические представления являются одним из самых эффективных способов представления данных на информационных панелях, с помощью которых можно быстро, наглядно и доступно показать взаимосвязь между измерениями и атрибутами. Oracle BI предоставляет ряд типов диаграмм, которые можно добавить к аналитике, а также карты, воронки и калибровки. Все типы диаграмм имеют аналогичный набор элементов управления для распределения показателей (measures) по осям и для включения таких функций, как трехмерное отображение, размеры. Графическое представление можно использовать для замены табличного представления или сводной таблицы в компоновке аналитики. Возможно переключение между графическим и табличным представлением или сводной таблицей.

Для создания простейшего графического представления выполните нижеперечисленные операции.

1. Создайте новую аналитику и включите столбцы в критерий аналитики. После добавления столбцов в область Selected Columns перейдите на вкладку Results, которая будет содержать в компоновке заголовочное и табличное представления.

2. Для добавления нового графического представления в компоновку воспользуйтесь кнопкой New View панели инструментов Compound Layout и выберите тип создаваемого графического представления, например гистограмма (bar): New View/Graph/Bar/Default (Vertical). Oracle BI предоставляет следующие типы диаграмм: Bar, Line, Area, Pie, Line-bar, Time-series Line, Pareto, Scatter, Bubble, Radar. Тип диаграммы выбирается в соответствии с конкретной аналитикой. Линейчатые диаграммы рекомендуется использовать при отображении изменений значений во времени, например когда требуется отобразить изменение показателей продаж и доходности за последние шесть месяцев. Гистограммы применяют для представления значений измерений, члены которых не являются последовательностью (как, например, дни). Так, значения продаж по нескольким магазинам следует отобразить с помощью горизонтальной или вертикальной гистограммы. Круговые диаграммы (pie graphs) полезны в случае, когда требуется представить пропорциональное распределение показателя (например, распределение общих продаж по категориям товаров или типам клиентов). Диаграмма «горизонтальный стек» используется для того, чтобы продемонстрировать,

как показатель, например выручка от регионов, разбивается в каждом регионе по категории продуктов, предоставляя дополнительное измерение для вашего представления. Диаграммы с областями применяются, когда показатель, например доход, изменяется во времени. Эти диаграммы включают разбиение по показателю, чтобы отобразить, как значение последнего изменилось пропорционально в течение определенного периода времени. Используя диаграммы Парето, можно разбить показатель на компоненты и продемонстрировать, как они объединяются с течением времени.

Альтернативным вариантом создания графического представления является использование кнопки New View панели View. Диаграммы рассеяния показывают значения для двух независимых переменных в рассеянном кластере и полезны для наблюдения за связями и тенденциями в больших наборах данных. Графики линейных диаграмм отображают изменение двух или более показателей, например по времени, но с одним показателем, представленным в виде набора полос, а другим – в виде линии.

3. Созданное графическое представление отображается в компоновке ниже табличного представления. Удалите табличное представление с помощью кнопки Remove View From Compound Layout, расположенной в верхнем правом углу табличного представления.

В режиме редактирования графического представления, переход в который осуществляется с помощью кнопки Edit View, можно изменить тип графического представления, задать используемые в гистограмме показатели и измерения и выполнить другие изменения. Для выхода из режима редактирования используйте кнопку Done.

5.2. СОЗДАНИЕ ИНФОРМАЦИОННЫХ ПАНЕЛЕЙ

Информационные панели являются наилучшим способом представления результатов анализа. Они могут содержать разные аналитики на основе одного или нескольких источников данных, а также подсказки панели управления, списки папок, ссылки действий и другие интерактивные элементы для создания веб-приложения Oracle BI.

Создание информационной панели начинается с выбора команды главного меню New/Dashboard или ссылки Dashboard домашней страницы веб-сайта Oracle BI. Далее предлагается указать имя, описание и каталог расположения создаваемой информационной панели, а также

определиться, наполнять содержимым информационную панель прямо сейчас или отложить эти действия.

Обратите внимание, что для отображения создаваемой панели в списке информационных панелей, доступных из меню Dashboards, ее требуется сохранить во вложенной папке верхнего уровня в общей папке Shared Folders, например в Shared Folders/Sales/Dashboards. Если сохранить панель управления в другой папке (например, в папке /My Folders или Shared Folders/Sales/Products/Dashboards), панель управления не появится в меню Dashboards. Также обратите внимание, что если вы выберете вложенную папку верхнего уровня в /Shared Folders, у которой еще нет папки Dashboards, то в ней будет создана папка /Dashboards после сохранения информационной панели.

В большинстве случаев каждый пользователь будет иметь свою собственную уже созданную информационную панель, которая называется My Dashboard и расположена в папке My Folders. Объекты, сохраненные пользователем в этой папке, обычно недоступны для других пользователей. Таким образом, предпочтительнее для пользователей применять My Dashboard в качестве личной рабочей области, а не создавать объекты в общей папке.

Если после того как указаны данные панели (название, описание, каталог), выбрано добавление содержимого информационной панели (кнопка Add Content Now), панель будет открыта и готова к редактированию. Иначе (кнопка Add Content Later) панель будет только создана в папке.

Созданная информационная панель может быть открыта для редактирования со страницы Catalog веб-сайта Oracle BI или при выборе опции редактирования в открытой информационной панели: команда Edit Dashboard меню Page Options.

Рассмотрим пример создания информационной панели и наполнение ее содержимым: двумя аналитиками и отчетом BI Publisher, расположенными в двух столбцах и одном разделе. Для создания информационной панели выполните следующее:

1) выберите из главного меню New/Dashboard или воспользуйтесь ссылкой Dashboard домашней страницы веб-сайта Oracle BI;

2) в открывшемся диалоговом окне New Dashboard введите имя информационной панели в строку Name, описание в строку Description, папку, где будет сохранена панель, в строку Location и укажите переключатель Add Content Now, чтобы информационная панель осталась открытой для последующего редактирования;

3) перетащите поочередно два объекта Column из области Dashboard Objects и расположите их один рядом с другим;

4) перетащите объект Section области Dashboard Objects в первый столбец Column 1 так, чтобы объект Section оказался внутри столбца;

5) перетащите две аналитики из области Catalog в столбец Column 1;

6) для предварительного просмотра информационной панели щелкните кнопку Save на панели инструментов и кнопку Run. После чего отобразится информационная панель, которая будет содержать две аналитики (одна над другой), причем второй столбец не будет представлен, так как он не содержит никаких объектов;

7) далее добавьте отчет Oracle BI Publisher и организуйте информационную панель таким образом, чтобы столбцы отображались строками, верхняя строка содержала бы две аналитики, а нижняя – отчет Oracle BI Publisher. Для этого из меню Page Option выберите Edit Dashboard;

8) удерживайте указатель мыши в верхней части Section 1-го столбца Column 1 так, чтобы отобразилась панель инструментов с четырьмя кнопками. Щелкните кнопку Horizontal Layout. Теперь две аналитики расположены рядом, а не друг над другом;

9) перетащите отчет Oracle BI Publisher из области Catalog в столбец Column 2;

10) перетащите столбец Column 2 так, чтобы он оказался под столбцом Column 1;

11) щелкните кнопку Save на панели инструментов и кнопку Run. В результате получаем информационную панель с двумя аналитиками, находящимися в верхней части панели, и отчетом Oracle BI Publisher, расположенным под аналитиками;

Каждая создаваемая информационная панель содержит одну страницу. Для того чтобы добавить страницы, которые отобразятся вверху панели с помощью вкладок, выберите команду Edit Dashboard меню Page Options, если панель открыта, иначе ссылку Edit в области Catalog. Когда редактор информационной панели открыт, добавить/удалить панель можно с помощью кнопок Add Dashboard Page/Remove Dashboard Page панели инструментов. При добавлении страницы информационной панели требуется указать имя и описание этой страницы. По умолчанию новой странице присваивается имя Page 1, следующей странице – Page 2 и т. д. Для изменения имени страницы выберите Tools/ Dashboard Properties, для открытия диалогового окна – Dashboard Properties. С помощью этого окна можно изменить имя страницы, по-

рядок следования страниц, сбросить и активировать кнопку, задать доступ к страницам только для определенных ролей приложений.

Кроме того, имеется возможность настроить разделы на странице информационной панели так, чтобы они отображались только для определенных ролей приложений. Для этого откройте страницу информационной панели с помощью редактора панели инструментов, а затем наведите указатель мыши на раздел, для которого требуется ограничить доступ. Предоставляется выбрать один из следующих вариантов: Condition, который используется для установки условия, например, отображать раздел или нет, или Permissions, позволяющий ограничить доступ к разделу определенным ролям приложения.

Используя Condition и Permissions, можно условно отображать информацию на информационной панели на основе событий, которые происходят в исходной системе, или данных, находящихся в определенных диапазонах.

Также допускается создание информационных панелей общего назначения, которые содержат информацию, доступную для множества пользователей разных типов, а затем для определенных пользователей отображение только тех разделов, которые имеют отношение к назначенным им ролям приложений.

В теме 5.1 были рассмотрены фильтры, определяемые при создании критериев аналитики. Это встроенные фильтры, которые существуют только как часть аналитики. Однако допускается определение фильтра как именованного или сохраненного, когда его можно создать и сохранить для последующего использования. Существует два способа создания таких фильтров. Первый способ состоит в том, что на домашней странице Oracle BI в главном меню выбирается New/Filter или в левой части страницы – Create/More/Filter. Второй способ заключается в присвоении имен и сохранении встроенных фильтров, которые были созданы ранее в критерии аналитики. При первом сохранении фильтра предлагается сохранить его в специальной папке под названием Subject Area Content, которая находится либо в папке My Folders, либо в Shared Folders в зависимости от того, выбрали ли вы свою личную область каталога или общую область. Эта папка является местоположением по умолчанию для именованных объектов, таких как фильтры, и рекомендуется для их хранения. Для того чтобы использовать сохраненный фильтр в критерии аналитики, его требуется включить, копируя определение этого фильтра в определение аналитики или используя ссылку, когда только указатель на определение

фильтра включается в определение аналитики, что позволяет аналитике использовать обновленное определение фильтра при его изменении в дальнейшем. Копирование определения фильтра в определение аналитики оставляет фильтр неизменным, предотвращая любые последующие изменения и удаление фильтра из папки.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Перечислите этапы создания аналитического отчета.
2. Назовите типы и этапы создания представлений аналитического отчета.
3. Перечислите этапы создания простых информационных панелей.
4. Назовите этапы создания многостраничных информационных панелей.
5. Перечислите этапы создания информационных панелей с условным отображением, различными уровнями доступа, фильтрами.

6. РЕГЛАМЕНТИРОВАННАЯ ОТЧЕТНОСТЬ

6.1. ВВЕДЕНИЕ В ORACLE BUSINESS INTELLIGENCE PUBLISHER. АДМИНИСТРИРОВАНИЕ ORACLE BUSINESS INTELLIGENCE PUBLISHER

Oracle BI Publisher предназначен для создания отчетов из нескольких источников данных. Oracle BI Publisher является компонентом аналитической платформы Oracle BI, но может использоваться как самостоятельный продукт либо как продукт, интегрированный с такими продуктами, как Oracle Business Suite Enterprise Edition, Oracle PeopleSoft.

В рамках аналитической платформы Oracle BI продукт Oracle BI Publisher используется для создания отчетов различных форматов с использованием данных как из репозитория Oracle BI, так и из других источников данных. Сгенерированные отчеты распространяются по различным каналам доставки, например доставляются пользователям на электронную почту. BI Publisher дополняет отчеты, созданные с использованием Oracle BI Answers, предоставляя средства для добавления блоков текста, изображений и другого контента в отчеты, а также для создания отчетов в формах писем, брошюр и др.

Oracle BI Publisher предоставляет возможность создания онлайн-отчетов в стиле информационных панелей. Начиная с Oracle BI Publisher 11g в отчетах, созданных с помощью этого инструмента, используется один и тот же механизм построения графиков, применяемый в аналитиках BI, что позволяет легко интегрировать отчеты BI Publisher в информационные панели. BI Publisher предоставляет дополнительную возможность доступа к источникам данных за пределами репозитория Oracle BI. Oracle BI Publisher поставляется со своей собственной средой разработки и средствами доступа к источникам данных.

Ключевым отличием отчетов, созданных с помощью Oracle BI Publisher, от отчетов, созданных с помощью редактора аналитики Oracle BI, является использование отдельного редактора Data Model Editor для подготовки и форматирования данных. Редактор Data Model Editor имеет доступ к таким типам источников данных, как репозиторий, выво-

димые данные аналитик, SQL-запросы, MDX-запросы, XML, файлы Microsoft Excel, LDAP-запросы, ADF-объекты представлений. При использовании редактора Data Model Editor осуществляется выбор из диапазона типов источников данных, формируется возвращающий в отчет данные запрос. При этом данные из разных источников могут быть объединены.

Когда модель данных для отчета создана, редактор BI Publisher Online Layout Editor предоставляет для размещения элементов отчета среду разработки, подобную Microsoft Office, в которой можно выбрать элементы данных для добавления в шаблон, диапазон объектов отчета, таких как таблицы, сводные таблицы, диаграммы и описания, а также добавить текст и изображения.

Данные, извлеченные из источников данных отчета, BI Publisher объединяет во время выполнения с помощью шаблона отчета для вывода отчета в виде PDF-файлов, электронных таблиц Excel и динамического HTML. Шаблоны берут элементы данных, предоставленные моделью данных, и упорядочивают их в формах таблиц, сводных таблиц, графиков, текста в макет отчета. Основными возможностями, предоставляемыми Oracle BI Publisher, являются:

- создание модели данных на основе репозитория Oracle BI, Oracle BI Answers, SQL-запросов, файловых источников данных;
- создание шаблонов с помощью редактора Data Model Editor для формирования информационных панелей и отчетов для печати;
- анализ данных с использованием Interactive Viewer либо автономно, либо с отчетами, встроенными в информационные панели;
- формирование расписания для доставки отчетов пользователям.

Полная информация о возможностях Oracle BI Publisher представлена на веб-сайте Oracle Technology Network (http://otn.oracle.com).

Для того чтобы приступить к использованию функционала BI Publisher, следует открыть браузер и зайти на домашнюю страницу Oracle BI. В разделе Create домашней страницы присутствует заголовок Published Reporting, под которым указаны три ссылки – Report, Report Job и More. Щелчок по ссылке More отображает дополнительные ссылки Data Model, Style Template, Sub Template. Пункт New главного меню также содержит опции Report, Report Job, Data Model, Style Template, Sub Template. Под разделом Create расположен раздел Browse/Manage, содержащий ссылку My Reports. Отчеты BI Publisher хранятся в том же каталоге Oracle BI Presentation, в котором хранятся аналитики и другие объекты BI, когда BI Publisher интегрирован в Oracle BI.

Администрирование BI Publisher выполняется с помощью отдельной страницы веб-сайта Oracle BI. Для перехода к этой странице следует сначала перейти на страницу Administration (по ссылке Administration, расположенной вверху домашней страницы веб-сайта Oracle BI), затем воспользоваться ссылкой Manager BI Publisher.

Когда BI Publisher интегрирован в Oracle BI, ему могут быть доступны два источника данных: репозиторий Oracle BI (посредством соединения JDBC, автоматически устанавливаемого на сервер Oracle BI) и логический SQL (из аналитики Oracle BI, содержащейся в каталоге Presentation Services). Поскольку основной причиной использования BI Publisher является доступ к источникам данных в дополнение к двум предопределенным источникам, необходимо найти эти источники данных. Рассмотрим определение источника данных SQL для базы данных Oracle, которое затем позволит создавать наборы данных, использующие этот источник, с помощью SQL-запросов. С целью определить источник данных SQL для базы данных Oracle выполните следующие действия:

1) выполните вход на веб-сайт Oracle BI, используя учетную запись пользователя с правами администратора;

2) на домашней странице Oracle BI перейдите по ссылке Administration. В открывшемся окне Administration перейдите по ссылке Manage BI Publisher;

3) далее откроются страницы BI Publisher Administration. Ссылки для создания источников данных разных типов расположены под заголовком Data Sources. Чтобы создать источник данных SQL, воспользуйтесь ссылкой JDBC Connection;

4) отобразится страница, показывающая предварительно сконфигурированные источники данных. Выберите вкладку JDBC и нажмите кнопку Add Data Source, чтобы приступить к созданию нового источника данных;

5) на странице Add Data Source введите имя источника данных в поле Data Source Name (например, имя схемы базы данных Oracle), тип драйвера в поле Driver Type (например, Oracle 11g), класс драйвера в поле Database Driver Class (например, oracle.jdbc.OracleDriver), имя пользователя и пароль в поля Username и Password соответственно. Для проверки соединения воспользуйтесь кнопкой Test Connection, затем кнопкой Apply для сохранения определения источника данных;

6) для этого подключения можно определить резервный источник данных, который используется, когда основное соединение недоступно. Для этого в разделе Backup Data Source страницы Add Data Source

установите флажок Use Backup Data Source, чтобы включить эту функцию, а затем введите строку подключения, имя пользователя и пароль для того же типа базы данных и класса драйвера. Проверьте соединение и сохраните его, как было описано на предыдущем этапе.

Рассмотрим создание источников данных, которые подключаются к электронным таблицам Microsoft Excel. Необходимость в таком подключении может возникнуть в случае, когда не все требуемые для отчета данные содержатся в базе данных. Создание файлового источника данных осуществляется следующим образом:

1) выполните вход на веб-сайт Oracle BI, используя учетную запись пользователя с правами администратора;

2) на домашней странице Oracle BI перейдите по ссылке Administration. В открывшемся окне Administration перейдите по ссылке Manage BI Publisher, чтобы открыть страницу BI Publisher Administration;

3) перейдите по ссылке File, расположенной под заголовком Data Source. Перейдите на вкладку File, воспользуйтесь кнопкой Add Data Source;

4) на странице Add Data Source введите имя источника данных в поле Data Source Name, путь к файлу Microsoft Excel в поле Full Path of Top-Level Directory;

5) воспользуйтесь кнопкой Apply для сохранения определения источника данных.

По умолчанию любой созданный источник данных не может быть доступен кому-либо, пока вы не предоставите доступ ролям приложения. Предоставить доступ к источникам можно с помощью ссылки Roles and Permissions, предназначенной для управления ролями и разрешениями, страницы администрирования BI Publisher Administration следующим образом:

1) на странице BI Publisher Administration в разделе Security Center воспользуйтесь ссылкой Roles and Permissions;

2) на странице Security Center перейдите на вкладку Roles and Permissions. Найдите предоставляемую роль приложения в столбце Name Role. Щелкните по иконке в столбце Add Data Source, расположенной напротив имени этой роли;

3) отобразится окно, в котором группируются источники данных, уже определенные в системе по типу (например, Database Connections, File Directories и др.). С помощью кнопок Move and Move All выберите те источники данных, которым вы предоставляете доступ к роли, а затем нажмите Apply, чтобы сохранить выбранные параметры.

Созданные в BI Publisher отчеты можно отравить на принтеры, электронную почту пользователей, факсы или другие устройства. Настрой-

ки распространения отчетов по различным каналам доставки определяются с помощью страницы BI Publisher Administration.

Доступ к принтерам через BI Publisher осуществляется через Iternet Printing Prtocol (IPP), который определяет соединение с сервером печати, обеспечивающим доступ к одному или нескольким принтерам. Это означает, что для доступа BI Publisher к принтеру на сервере Unix необходимо установить и скомпилировать Common Unix Printing Services (CUPS), а находясь на сервере под управлением Microsoft Windows, нужно настроить Internet Information Services (IIS) и сервер Windows Print для IPP. Для того чтобы выполнить эти конфигурации, обратитесь к руководству Oracle Fusion Middleware Administrator's Guide для Oracle BI Publisher на веб-сайте Technology Network (http://otn.oracle.com).

С целью определения сервера печати для вывода на печать отчета BI Publisher после настройки IPP для своего сервера выполните следующие действия:

1) на странице BI Publisher Aministration под заголовком Delivery перейдите по ссылке Printer;

2) на странице Delivery перейдите на вкладку Printer и щелкните по кнопке Add Server;

3) в диалоговом окне Add Server введите имя сервера, URI (IPP адрес сервера принтера) и другие параметры соединения (такие как имя пользователя, пароль), которые необходимы для получения доступа к серверу принтера.

В дальнейшем, когда этот сервер печати будет использоваться при доставке, следует указать конкретный принтер, который будет использоваться, и другие параметры, которые предоставляются конкретным сервером печати.

Определение рассылки отчетов BI Publisher на электронную почту пользователей выполняется также с помощью страницы BI Publisher Aministration следующим образом:

1) на странице BI Publisher Aministration под заголовком Delivery перейдите по ссылке Email;

2) на странице Delivery перейдите на вкладку Email и щелкните по кнопке Add Server;

3) в диалоговом окне Add Server введите имя сервера, хост, порт, имя пользователя и пароль, укажите наличие или отсутствие безопасного соединения сервера электронной почты.

Фактический адрес электронной почты, на который доставляются отчеты, указывается при дальнейшем планировании доставки отчета.

6.2. ПОНЯТИЕ МОДЕЛИ ДАННЫХ. РЕДАКТОР МОДЕЛИ ДАННЫХ. СОЗДАНИЕ МОДЕЛИ ДАННЫХ

Отчеты Oracle BI Publisher состоят из двух основных компонентов: модели данных, которая определяет набор данных, используемый отчетом, и шаблона отчета, который определяет, как формируется отчет.

Каждому отчету BI Publisher требуется связанная с ним модель данных, которая выбирается при первом определении отчета. Модель данных задает источники данных, которые предоставляют данные отчету, и используется BI Publisher для создания отдельного XML-документа, содержащего данные отчета, которые впоследствии объединяются с шаблоном для создания отчета.

Модель данных может содержать один или несколько наборов данных. Набор данных – это запрос к типу источника данных, например SQL-запрос к базе данных или MDX-запрос к серверу OLAP. При запуске отчета каждый из этих наборов данных запрашивается для возврата данных, используемых в отчете. Можно выполнить слияние наборов данных или соединение, используя общее поле или общий элемент.

Приведем список всех элементов модели данных.

Наборы данных (Data Sets). Допускается наличие одного или нескольких соединенных либо конкатенированных наборов данных.

Триггеры событий (Event Triggers). Используются для запуска на выполнение PL/SQL-кода до или после запуска отчета для заполнения, например таблицы данных, требуемой для отчета.

Сегментные поля (Flexfields). Применяются при запрашивании данных приложением Oracle E-Business Suite.

Параметры (Parameters). Используются в сочетании с наборами данных для возврата, например только данных для определенного региона или продукта.

Список значений (List of Values). Применяется для формирования меню, например продуктов или регионов.

Пакетная передача (Bursting). Пакетная передача необходима для разбиения одного отчета на отдельные элементы, которые затем отправляются по разным адресам доставки, но основываются на одном выполнении отчета.

Модели данных определяются как подлежащие повторному использованию в разных отчетах. Они хранятся вместе с определениями шаблонов в каталоге Presentation Services при использовании BI Publisher, интегрированного с Oracle BI.

Рассмотрим создание модели данных, использующей один источник данных. Пусть требуется создать отчет на основе таблиц некоторой схемы базы данных. Формирование модели данных возможно тремя способами:

- создание аналитики, использование ее в качестве источника для набора данных модели данных BI Publisher;
- формирование набора данных с помощью логического SQL-запроса на основе модели данных репозитория Oracle BI;
- установка соединения со схемой напрямую.

Создание модели данных на основе репозитория Oracle BI подобно процессу создания SQL-запроса для извлечения данных, когда в качестве источника данных используется репозиторий Oracle BI, однако запросы к этому источнику данных могут быть значительно упрощены, так как не нужно включать функции агрегации, группировки или сортировки, поскольку сервер Oracle BI автоматически добавляет их на основе определения столбцов в семантической модели репозитория Oracle BI.

Воспользуемся последним из указанных выше способов создания модели данных, так как установление соединения со схемой напрямую наиболее подробно демонстрирует процесс создания модели данных. Модель данных, содержащая один набор данных, который основан на базе данных Oracle, создается следующим образом:

1) выполните вход на домашнюю страницу Oracle BI с учетными данными пользователя, который имеет доступ к источнику данных Oracle BI EE;

2) выберите New/Data Model. Откроется редактор Data Modeler Editor, предоставляющий обзор определений модели данных;

3) используя панель Properties, определите источник данных по умолчанию, который становится начальной настройкой для всех наборов данных в этой модели данных;

4) для создания набора данных, использующего данные из определенного источника, выберите элемент Data Sets в древовидном меню Data Model. На вкладке Diagram воспользуйтесь кнопкой New Data Set и выберите SQL Query из списка типов источников данных;

5) в диалоговом окне Create Data Set – SQL введите имя набора данных, убедитесь, что выбран соответствующий источник данных и воспользуйтесь кнопкой Query Builder;

6) для формирования набора данных для запроса убедитесь, что в диалоговом окне Query Builder выбрана требуемая схема в раскрыва-

ющемся списке Schema с каталогом, соответствующим предметной области репозитория;

7) в диалоговом окне Query Builder укажите таблицы, а затем столбцы, которые требуется добавить к набору данных;

8) так как набор данных создается на основе реляционной базы данных, определите соединение таблиц в наборе данных. Для этого укажите столбцы, по которым будет выполняться соединение;

9) перейдите по ссылке Results в диалоговом окне Query Builder, чтобы просмотреть результат;

10) воспользуйтесь кнопкой Save диалогового окна Query Builder, чтобы сохранить запрос, и перейдите в окно Create Data Set – SQL. Щелкните кнопку OK.

При создании модели данных можно определить параметры в модели данных, которые затем используются в запросах набора данных для ограничения данных, отправляемых обратно в BI Publisher при запуске отчета; добавить столбцы, полученные с использованием выражений.

Рассмотрим создание модели данных, которая содержит набор данных, основанный на аналитике Oracle BI. При запуске отчета, который использует аналитику в качестве источника данных, выполнятся аналитика, затем данные передаются обратно в BI Publisher, который преобразовывает их в XML и предоставляет шаблону. Например, на основе аналитики, которая отображает продукт, даты, требуется создать форматированный отчет и отправить его на электронную почту сотрудникам. Для этого выполните следующие действия:

1) осуществите вход на домашнюю страницу Oracle BI и выберите New/Data Model;

2) выберите Data Set/Oracle BI Analysis;

3) в диалоговом окне Create Data Set – Oracle BI Analysis введите имя набора данных, время ожидания извлечения данных. Щелкните кнопку поиска и выберите аналитику;

4) сохраните модель данных, щелкните кнопку Get XML Output, сгенерируйте выборочные данные, выберите пункт Save As Sample Data из раскрывающегося списка, расположенного справа от кнопки Return;

5) выполните сохранение модели данных. Модель данных готова для использования в шаблоне.

6.3. ВВЕДЕНИЕ В ONLINE LAYOUT EDITOR. СОЗДАНИЕ МАКЕТОВ ОТЧЕТОВ. ДОБАВЛЕНИЕ НОВЫХ ШАБЛОНОВ МАКЕТОВ

Шаблон отчета определяет, как элементы данных из модели данных организованы в печатный, рассылаемый электронный или экранный отчет. Для создания шаблона в Oracle BI Publisher можно использовать документы RTF (Rich Text Format), электронные таблицы Microsoft Excel, документы Adobe PDF. Редактор Online Layout Editor, представленный в выпуске 11g, используют при создании макетов, содержащих такие BI-объекты, как диаграммы, сводные таблицы, датчики.

Перечислим этапы создания макета отчета вида информационной панели, который будет содержать таблицу, сводную таблицу, диаграмму, датчик, список.

1. Создание модели данных с одним или несколькими наборами данных источника данных.

2. Создание определения отчета, выбор источника данных и шаблона для указания ориентации страницы, размеров страницы и т. д.

3. Создание шаблона, который отчет может использовать, содержащего объекты данных и компоненты шаблона для создания вида информационной панели для данных.

4. Сохранение шаблона, а затем и определения отчета, которое включает шаблон, в каталог, для последующего запуска.

5. Просмотр отчета либо автономно в BI Publisher Enterprise, либо его можно добавить в качестве объекта информационной панели в BI-информационную панель для отображения рядом с другим BI-содержимым.

Рассмотрим этапы создания определения отчета в BI Publish Release 11.1.1.6.

1. Выполните вход на домашнюю страницу Oracle BI и выберите New/Report.

2. Сначала будет предложено использовать существующую модель данных, загрузить электронную таблицу или создать новую модель данных для отчета. Полагая, что модель данных для этого отчета уже создана, выберите Use Existing Data Model.

3. Выберите в каталоге созданную ранее модель данных.

4. Определите, будет ли использован мастер создания отчетов Create Report при создании шаблона отчета или шаблон будет создан с помощью опции Report Editor.

5. Выбор опции использования функции Guide Me полезен для конечных пользователей, которые хотят получить простой отчет с минимальным участием с их стороны, а создаваемый отчет и макет могут впоследствии быть расширены и изменены с помощью Report Editor.

6. Далее выберите функцию Use Report Editor, нажмите кнопку Finish, а затем сохраните определение исходного отчета в каталоге, используя диалоговое окно Save As.

7. Выберите для отчета вида информационной панели опцию Blank (Landscape).

В результате выполнения указанной выше последовательности действий создан отчет, содержащий пустой макет, в который требуется добавить следующие элементы: гистограмму, сводную таблицу, таблицу, датчики, горизонтальный список. Для добавления перечисленных элементов выполните нижеперечисленные действия.

1. Выберите опцию Blank (Landscape). Отобразится пустой макет в редакторе Online Layout Editor.

2. Для добавления списка на вкладке Insert выберите List в меню панели инструментов и перетащите в макет. Перетащите в список требуемый элемент данных с панели Data Source. По умолчанию данные списка располагаются по вертикали. Для того чтобы изменить расположение данных списка на горизонтальное на панели Properties установите свойство Orientation в значение Horizontal. Расположите список в верхней части макета.

3. Расположите основные две строки с двумя столбцами панели инструментов. Для этого перейдите на вкладку Insert и перетащите элемент Layout Grid так, чтобы он оказался под списком, выберите количество строк – 2, столбцов – 2.

4. Аналогично добавьте элемент «диаграмма Chart» вкладки Insert в первый столбец первой строки. Укажите, какие данные будут значениями осей OX и OY, выбрав в Data Source столбцы данных и перетащив их в область Drop Label Here и область Drop Value Here.

5. Расположите во втором столбце первой строки сводную таблицу. Для этого на вкладке Insert выберите элемент Pivot Table и перетащите его в требуемую позицию макета. Определите, какие данные будет содержать сводная таблица, перетаскивая данные из Data Source в области Drop Rows Here, Drop Columns Here, Drop Data Here.

6. Добавьте таблицу в первый столбец второй строки. Для этого на вкладке Insert выберите элемент Table и перетащите его в требуемую позицию макета. Определите, какие данные будет содержать таблица.

Выделив таблицу, можно задать шрифт, формат и другие параметры с помощью соответствующей панели инструментов.

7. Добавьте датчик во второй столбец второй строки, выбрав на вкладке Insert элемент Gauge. Перетащите данные в области Drop Value Here, Drop Label Here, Drop Series Here и воспользуйтесь панелью Properties для выбора между Dial, Status Meter, Vertical Status Meter.

8. Сохраните макет, воспользовавшись кнопкой Save.

Таким образом, макет отчета создан. Для того чтобы вернуться к определению отчета, щелчком по кнопке Return редактора Layout Editor откройте окно редактирования отчета Report. С помощью этого окна можно просмотреть и изменить метки и значения по умолчанию для параметров, установить другие свойства отчета.

Рассмотрим процесс встраивания отчета в информационную панель BI.

1. Перейдите на веб-сайт Oracle BI по ссылке http://[machine_name]:port/analytics. Зайдите как пользователь с правом создания отчетов и других BI-объектов. Откройте существующую информационную панель или создайте новую, выбрав New/Dashboard. Укажите имя созданной информационной панели и убедитесь, что сохранили ее в /Shared/Folders.

2. В редакторе информационной панели перейдите в область Catalog, найдите отчет и перетащите его в макет информационной панели.

3. Для того чтобы просмотреть отчет на странице информационной панели, требуется сохранить информационную панель (кнопка Save) и запустить его щелчком по кнопке Run.

Oracle BI Publisher позволяет создать собственный шаблон, который, например, будет содержать логотип компании. Для формирования шаблона требуется создать специальный отчет, называемый Boilerplates, и сохранить его в отдельной директории Components в разделяемой области каталога. Каждый шаблон макета, который добавляется к этому отчету, становится доступным всем пользователям при создании отчетов. Для создания разделяемого шаблона макета требуется выполнить следующие действия:

1) из меню на сайте Oracle BI выберите New/Report. В окне, предлагающем выбрать модель данных, щелкните кнопку Cancel;

2) выберите один из предлагаемых базовых шаблонов, например Header and Footer (Portrait);

3) далее предоставляется пустой макет, но без элементов данных, перечисленных на панели Data Source. Добавьте компоненты макета к вашему макету;

4) сохраните макет, указав его название, щелкните по кнопке Return и сохраните отчет. Указывая имя отчета, введите Boilerplates и сохраните его в новую директорию /Shared Folders/Components.

Теперь при создании каждого нового отчета разделяемый шаблон будет присутствовать в списке доступных шаблонов макета.

ВОПРОСЫ ДЛЯ САМОПРОВЕРКИ

1. Охарактеризуйте Oracle Business Intelligence Publisher.
2. Дайте определение понятию модели данных.
3. Перечислите этапы создания модели данных.
4. Назовите этапы создания макетов отчетов с помощью Online Layout Editor.

СПИСОК ЛИТЕРАТУРЫ

Исаченко, *А. Н.* Модели данных и системы управления базами данных / А. Н. Исаченко, С. П. Бондаренко. Минск : БГУ, 2007.

Becker, *B.* Relentlessly Practical Tools for Data Warehousing and Business Intelligence / B. Becker, J. Mundy, W. Thornthwaite. Indiana : Wiley Publishing, 2010.

Howson, *C.* Successful Business Intelligence / C. Howson. N. Y. : Mc Graw Hill, 2014.

Moss, *L.* Business Intelligence Roadmap: The Complete Project Lifecycle for Decision-Support Applications / L. Moss, Sh. Atre. Boston : Addison-Wesley, 2003.

Rittman, *M.* Oracle Business Intelligence 11g Developers Guid / M. Rittman. N. Y. : Mc Graw Hill, 2013.

Vlamis, *D.* Data Visualization for Oracle Business Intelligence 11g / D. Vlamis. N. Y. : Mc Graw Hill, 2015.

СОДЕРЖАНИЕ