МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ

Зав. кафедрой АОИ
д.т.н. профессор

_____Ю.П. Ехлаков

Учебно-методическое пособие к выполнению самостоятельной и лабораторных работ по дисциплинам **«РАСПРЕДЕЛЕННЫЕ ИНФОРМАЦИОННЫЕ СИСТЕМЫ»**

для студентов направления:
09.03.04 - Программная инженерия
«РАСПРЕДЕЛЕННЫЕ СИСТЕМЫ»
для студентов направления:
38.03.05 - Бизнес-информатика

РАЗРАБОТЧИК
доцент кафедры АОИ
_____ П.В. Сенченко
«___»_____2016 г.

Сенченко П.В.

Распределенные информационные системы: Учебно-методическое пособие к выполнению самостоятельной и лабораторных работ по дисциплинам: «Распределенные информационные системы» и «Распределенные системы». – Томск: Томск. гос. ун-т систем управления и радиоэлектроники, 2016. – 48 с.

Учебно-методическое пособие может быть востребовано студентами направления бакалавров «Бизнес-информатика» и «Программная инженерия» при подготовке к лабораторным работам, а также при выполнении самостоятельной работы.

СОДЕРЖАНИЕ

Введение

Цели изучения дисциплины «Распределенные системы»:

- формирование базовых знаний, позволяющих студентам в условиях стремительного развития и совершенствования информационных технологий эффективно применять теоретические знания в области архитектуры и методов управления распределенными информационными системами (РИС) для организации хранения, доступа, обработки информации, основ построения распределенных систем различными программными средствами;
- развитие творческих способностей для проектирования оптимальных архитектурных моделей распределенной обработки информации.

Задачи изучения дисциплины является:

- освоение студентами классификации распределенных систем, их архитектуры, областей применения;
- овладение средствами и способами построения и организации распределенных систем;
- приобретение навыков работы с различными методами работы в распределенных системах;
- овладение средствами проектирования и создания эффективных приложений, обеспечивающих возможность принятия ключевых решений на ранних этапах создания проекта.

Процесс изучения дисциплины направлен на формирование следующих профессиональных компетенций:

Для направления 09.03.04 - Программная инженерия:

владением навыками использования различных технологий разработки программного обеспечения (ПК-3).

Для направления 38.03.05 - Бизнес-информатика:

выбор рациональных информационных систем и информационно-коммуникативных технологий решения для управления бизнесом (ПК-3)

В результате изучения дисциплины **студент должен**:

знать:

- классификацию распределенных информационных систем (РИС);
- методологии и технологии поставки распределенных решений;
- жизненные циклы РИС;
- архитектурные подходы построения распределенных систем (РС);

- технологические основы многозвенных архитектур;
- методы и средства управления транзакциями, обеспечения многопользовательского доступа к информационным ресурсам предприятия;
- основы сервисно-ориентированной архитектуры (SOA);
- методы и средства интеграции прикладных систем предприятия;
- архитектуру интеллектуальных ресурсов предприятия;

уметь:

- проектировать и обоснованно выбирать архитектуру программных решений;
- анализировать и разрабатывать элементы распределенных систем;
- анализировать требования;
- использовать современные программные средства для создания и внедрения распределенных систем;

владеть:

- навыками сбора и анализа требований РИС;
- языком управления транзакциями;
- навыками разработки и отладки элементов распределенных приложений;
- навыками использования современных сред разработки.

Настоящее учебно-методическое пособие разработано на основе материалов лабораторного практикума, предложенного ст. преподавателем каф. АОИ ТУСУР И.И. Веберовой.

Лабораторная работа № 1 «Создание объектов базы данных ORACLE»

Продолжительность: 4 часа.

Целью настоящей работы является изучение объектов БД и получение практических навыков проектирования, создания и изменения объектов в соответствии с заданной проблемной областью.

База данных ORACLE может содержать различные структуры данных или объекты, например – таблицы, представления, последовательности, индексы.

СОЗДАНИЕ ТАБЛИЦ

Команда Create Table

Создать таблицу, как, впрочем, и любой другой объект БД, можно с помощью команды SQL – Create, которая принадлежит языку определения данных DDL. Команды DDL немедленно воздействуют на БД и заносят информацию в словарь данных.

Для изучения таблиц пользователь должен обладать привилегией Create Table и областью хранения данных (схемой), где можно создавать объекты. Предоставляя пользователям привилегии, администратор БД использует команды языка DCL (Grant, Revoke).

Синтаксис

CREATE TABLE [схема.] таблица
(столбец тип данных [Default выражение]
[ограничение столбца],
…
[ограничение таблицы]);
где схема – то же, что имя владельца;
таблица – имя таблицы;
Default выражение – задает значение по умолчанию;
столбец – имя столбца;
тип данных - тип данных и длина столбца;
ограничение - столбца – правило целостности как часть определения столбца;
ограничение - таблицы – правило целостности как часть определения таблицы.

Схема – это набор объектов данного пользователя. Указывается в том случае, если таблица создается в схеме, не принадлежащей пользователю, но пользователю даны права доступа к объектам схемы.

Типы данных в Oracle

Основными типами данных являются символ, число, дата и RAW (табл. 1).

Таблица 1

Примеры типов данных в Oracle

Типы данных	Описание
VARCHAR2 (L)	Символьные значения переменной длины, не превышающей заданного размера $L_{\min}=1, L_{\max}=2000,$ где L - длина)
CHAR (L)	Символьные значения фиксированной длины равной L , $L_{\min}=1, L_{\max}=255$
Number	Число с плавающей точкой с точностью 38 значащих цифр
Number (p, s)	Число с фиксированной точкой, $1 \le p \le 38,$ p - общее количество десятичных цифр, S – количество цифр справа от десятичной точки
DATE	Значение даты и времени между 1 января 4712 г. до н.э. и 31 декабря 4712 г. н.э.
Long	Символьные значения переменной длины размером до 2 гигабайтов. В таблице допускается только один столбец типа Long
RAW и Long RAW	Аналогичны соответственно типам данных VARCHAR2 и Long, но используются для хранения двоичных данных.

Параметр DEFAULT

Параметр **DEFAULT** задает значения столбца по умолчанию при вставке строк, например:

… start-date DATE DEFAULT SYSDATE …

Допускаются строковые константы, выражения и функции SQL SYSDATE и USER (имя пользователя).

Этот параметр исключает появление неопределенных значений при вставке строки без конкретного значения в данном столбце.

Ограничения

Ограничения реализуют правила по обеспечению целостности данных на уровне таблицы при вставке, обновлении и удалении строк.

Если ограничение задано, успешное выполнение операции без его соблюдения невозможно.

В Oracle существуют следующие виды ограничений:

NOT NULL – означает, что данный столбец не может содержать неопределенных значений;

UNIQUE – столбец или набор столбцов содержит значения, которые должны быть уникальны для всех строк таблицы;

PRIMARY KEY – уникально идентифицирует каждую строку таблицы;

FOREIGN KEY – устанавливает и поддерживает отношения между данным столбцом и столбцом таблицы, на которую делается ссылка с помощью внешнего ключа;

CHECK – задает условие, которое должно выполняться для каждой строки таблицы.

Синтаксис

Ограничение на уровне столбца:

column [constraint < constraint_name>] constraint_type;

Ограничение на уровне таблицы:

column …,

[constraint < constraint_name>] constraint_type

(column, column…),

Имена ограничений должны соответствовать стандартным правилам присвоения имен объектам. Обычно ограничения создаются одновременно с созданием таблицы. Но добавлять их можно и после создания таблицы. Кроме того, ограничения могут быть временно запрещены.

Пример создания таблицы

Таблицы создаются с помощью команды CREATE TABLE на основе бланка экземпляра таблицы, заполненного при проектировании базы данных (каждый бланк соответствует сущности ER-модели, составляется на основе методики отображения ERD на бланки экземпляров). Пример создания таблицы на основе бланка экземпляра:

Бланк экземпляра таблицы S_DEPT

Имя столбца	**ID**	**NAME**	**REGION_ID**
Тип ключа	PK		FK
NN/UK	NN, U_1	NN, U_2	U_2
Таблица FK			REGION
Столбец FK			ID
Тип данных	NUMBER	CHAR	NUMBER
Длина	7	25	7
Пример данных	10 31 32 32	Finance Sales Sales Sales	1 1 2 3

Синтаксис Create для таблицы S_DEPT:

```
CREATE TABLE     S_dept
  (id            number (7)
     constraint  S_dept_id_pk    PRIMARY KEY,
     name        varchar2 (25)
     constraint  S_dept_name_nn  NOT NULL,
     region      Number (7)
     constraint S_dept_region_id_fk
                    REFERENCES    S_region (id),
     constraint S_dept_name_region_id_uk
```

Unique (name, region_id));

Вторым методом создания таблицы является использование предложения «AS <подзапрос>». При этом создается таблица и в нее вставляются строки, возвращенные в результате выполнения подзапроса.

Синтаксис

CREATE TABLE таблица
[столбец (, столбец …)]
AS подзапрос;

где подзапрос – команда Select.

Порядок выполнения работы

1. Изучите структуру представленных ниже таблиц.

Таблица TITLE

Информация о художественных фильмах

Название	Описание	Рейтинг	Категория	Дата выпуска
Willie and christmas TOO	.....	**G**	**CHILD**	**05-OCT-97**
Alien Again	.....	**R**	**SCIFI**	**19-MAY-95**
THL	.....	**NR**	**SCIFI**	**12-AVG-96**
My Day OFF	.....	**PG**	**Comedy**	**12-JUL-97**
Mirucles on Ice	.....	**PG**	**Action**	**01-Jun-95**

Таблица MEMBER *(Члены клуба)*

Имя	Фамилия	Адрес	Штат	Телефон	Дата вступле-ния
Carmen	Velasquer	283 King Street	Seattle	206-844-6666	08-MAR-90
La Doris	Ngao	5 Modrany	Bratisla-va	586-355-8882	08-MAR-90
Midori	Nagagama	68 Via Central	SanPaolo	254-852-5764	17-Juni-91
Mark	Qvick-To-Sec	6921 King Way	Lagos	63-554-7777	07-APR-90

Таблица TITLY_COPY (копии фильмов)

Название	**Номер копии**	***Статус***
Willie and Christmas Too	1	Имеется
Alien Again	1	Имеется
	2	Выдано
The Glob	1	Имеется
My Day Off	1	Имеется
	2	Имеется
	3	Выдано
Miracles on Ice	1	Имеется
Soda Sang	1	Имеется

Таблица RENTAL

Название	**Номер копии**	**Клиент**	**Дата выдачи**	**Ожидаемая дата возврата**	**Фактическая дата возврата**
92	1	101	3 дня назад	Вчера	2 дня назад
93	2	101	Вчера	Завтра	
95	3	102	2 дня назад	Сегодня	
97	1	106	4 дня назад	2 дня назад	2 дня назад

2. Напишите скрипт для создания таблиц в БД на основе бланков экземпляров:

- во время создания таблицы TITLE;
- создайте скрипт для удаления созданных таблиц.

3. По словарю данных Oracle убедитесь в том, что таблицы и ограничения созданы правильно.

4. Создайте последовательность для однозначной идентификации каждой строки в таблицах MEMBER и TITLE:

- номера членов клуба для таблицы MEMBER начните со значения 101. Не разрешайте запись значений в кэш-память;
- начните нумерацию фильмов для таблицы TITLE с числа 92; Кэширование не производится;
- проверьте по словарю данных, что последовательности созданы.

5. Создайте скрипт-файл для добавления данных в таблицы с приглашением пользователю ввести информацию.

Лабораторная работа № 2 «Структурированный язык запросов SQL: манипулирование данными. работа в среде SQL*Plus»

Целью настоящей работы является изучение подмножества языка запросов SQL (манипулирование данными) и получение практических навыков взаимодействия с базой данных под управлением сервера ORACLE в среде SQL*PLUS.

SQL (Structured Query Language) – структурированный язык запросов, позволяющий создавать реляционные базы данных и оперировать ими. Благодаря своей элегантности и независимости от специфики компьютера язык SQL стал и в обозримом будущем останется международным стандартом, пригодным для использования на множестве современных компьютерных платформ.

Существует ANSI (American National Standards Institute) – стандарт языка SQL, однако, подавляющее большинство реляционных систем управления базами данных (РСУБД) расширяют возможности стандарта SQL.

СТРУКТУРИРОВАННЫЙ ЯЗЫК ЗАПРОСОВ SQL. МАНИПУЛИРОВАНИЕ ДАННЫМИ

SQL состоит из трех подмножеств:

- язык манипулирования данными DML (Data Manipulation Language) – это подмножество команд, определяющих какие данные представлены в таблицах в любой момент времени. К ним относятся:
  - SELECT – выборка данных,
  - INSERT – ввод новых данных,
  - UPDATE – изменение существующих данных,
  - DELETE – удаление ненужных строк из таблиц;
- язык определения данных DDL (Data Definition Language) – подмножество команд, которые обеспечивают определение данных, хранящихся в базе данных:
  - CREATE – создание объекта БД,
  - ALTER – изменение объекта БД,
  - DROP – удаление объекта,
  - RENAME – переименование,
  - TRUNCATE – удаление строк таблицы;
- язык управления данными DCL (Data Control Language) определяет, может ли пользователь выполнить определенное действие над БД. Включает команды: Commit, Rollback, Savepoint – управления транзакциями и Grant, Revoke – управления правами доступа. Настоящая работа посвящена освоению языка DML.

1.1. Команда SELECT

Команда SELECT используется для реализации информационных запросов пользователей к БД и имеет следующий *синтаксис:*

```
SELECT [DISTINCT]   <COLUMN LIST>
FROM   <TABLE – REFERENCE>
[WHERE  < SEARH_CONDITION>]
[ORDER  BY  <ORDER_LIST>  [DESC]]
[GROUP  BY  <GROUP_LIST>]
```

[HAVING <HAVINNG_CONDITION>],

где SELECT – ключевое слово, которое сообщает базе данных, что команда является запросом;

<COLUMN_LIST> – список столбцов таблицы, которые должны быть представлены в результате выполнения запроса;

опция DISTINCT исключает дублирование строк в выборке;

FROM – ключевое слово, за которым следует имя таблицы или имена нескольких таблиц, которые используют как источник информации.

Предложение WHERE позволяет определить условие, которое может быть либо истинным, либо ложным для каждой строки таблицы. Команда извлекает только те строки таблицы, для которых условие имеет значение «истина»;

ORDER BY определяет условие сортировки по одному или группе столбцов таблицы. Опция DESC (DESCENDING) задает сортировку в убывающем порядке;

GROUP BY определяет столбец или группу столбцов, на основании значений которых строки объединяются в группы, к которым применяются групповые функции;

HAVING определяет условие включения групп в результат запроса.

Рассмотрим несколько примеров синтаксиса SELECT для реализации запросов.

В примерах используются связанные таблицы SALESPEOPLE (Продавцы), CUSTOMERS (Покупатели) и ORDERS (Заказы), структуры которых приведены в приложении 1.

Внимание! Для успешного выполнения работы и приобретения необходимых навыков необходимо углубленное изучение возможностей команды SELECT [1, 3].

Пример 1

Например, следующий запрос позволяет узнать имена всех продавцов в Лондоне:

```
SELECT  sname, city
    FROM  Salespeople
            WHERE city='London';
```

Предложение WHERE может использовать операторы сравнения (=, <, > ...) , булевы операторы AND, OR, NOT, кроме того, специальные операторы IN, BETWEEN, LIKE и IS NULL .

IN полностью определяет множество, которому данное значение может принадлежать или не принадлежать. Если нужно найти всех продавцов, работающих либо в Барселоне, либо в Лондоне, то можно выполнить следующий запрос.

Пример 2

```
SELECT * FROM  Salespeople
        WHERE city IN ('Barselona', 'London') ;
```

BETWEEN задает границы интервала, в который должно попадать значение, чтобы условие было истинным.

Пример 3

```
SELECT * FROM  Salespeople
```

```
WHERE comm BETWEEN 10 AND 15;
```

Оператор BETWEEN является включающим, т.е. граничные условия (в нашем примере это .10 и .15) делают условие истинным.

LIKE применим только к символьным полям и используется для поиска подстрок. Существует два типа шаблонов, используемых с LIKE:

- символ ”подчеркивание” (_) заменяет один любой символ. Например, образцу ‘b_t’ соответствуют ‘bat’ или ‘bit’, но не соответствует ‘brat’;
- символ “процент” (%) заменяет последовательность символов произвольной длины, в том числе и нулевой. Например, образцу ‘ % p%t’ соответствуют ‘put’ , ‘posit’ , ‘opt‘, но не ’spite’.

Пример 4

Определим продавцов, фамилии которых начинаются на ‘G’:

```
SELECT * FROM  Salespeople
        WHERE sname  Like  ‘G%’;
```

Пример 5

Определим продавцов, заказы которых больше $100:

```
SELECT Orders.amt, Salespeople.sname
      FROM Orders, Salepeople
      WHERE(Orders.snum=Salepeople.snum) AND (amt>100)
```

Результат запроса:

AMT	SNAME
767.16	Peel
1900.10	Monica
5160.45	Alex
1098.43	Alex
4723.00	Peel
1309.09	Serres

Пример 6

Определим продавцов, у которых в Orders нет ни одного заказа:

```
SELECT snum, sname FROM Salepeople
      WHERE snum NOT IN
          (SELECT snum FROM Orders)
```

Результат запроса:

SNUM	SNAME
1001	Peel
1002	Serres
1004	Monica
1003	Alex

Запросы могут обобщать не только группы значений, но и значения одного поля. Для этого применяются агрегатные функции SUM, AVG, MIN, MAX, COUNT. Они дают единственное значение для целой группы строк таблицы.

Пример 7

SELECT SUM (amt) FROM Orders находит сумму всех заявок из таблицы Oreders.

ORDER BY используют для упорядочения строк в результирующем наборе.

Пример 8

SELECT * FROM Salespeople

ORDER BY sname выведет записи в алфавитном порядке.

GROUP BY – необязательная опция, которая группирует строки в запросе на основании значения в одной или более колонках.

Пример 9

SELECT MAX(amt) ” max ”, sname

FROM Orders

GROUP BY sname

Запрос показывает максимальную сумму продажи у каждого продавца.

Результат запроса:

MAX	SNAME
1900.10	1004
1098.43	1003
4723.00	1001
1309.09	1002

HAVING сообщает о необходимости включения групп в результате запроса.

< having_condition > не может содержать подзапрос.

HAVING используется с GROUP BY для указания критерия, по которому группа включается в результат запроса. Несколько условий связываются операторами AND и OR.

Пример 10

SELECT snum, SUM(amt) “sum” FROM Orders

HAVING odate > 10/03/1997

GROUP BY snum

ORDER BY snum

Результат запроса:

SNUM	SUM
1001	4821.56
1002	1384.99
1003	6258.48
1004	1900.10

UNION комбинирует окончательный результат одной команды SELECT с окончательным результатом другой SELECT команды.

Функции

Функции увеличивают мощность простого блока запроса и используются для манипулирования значениями данных. Существует два типа функций в SQL:

- однострочные:
  - символьные,
  - числовые,
  - функции дата,

- функции преобразования;

• многострочные:

- групповые.

Однострочные функции

Однострочные функции работают только с одной строкой и возвращают по одному результату на строку. Многострочные функции работают с группами строк и выдают по одному результату на каждую группу строк.

Рассмотрим функции для работы с датами.

Даты в системе ORACLE хранятся во внутреннем числовом формате, где представлено следующее: столетие, год, месяц, день, часы, минуты, секунды. По умолчанию вывод даты производится в формате DD-MM-YY.

SYSDATE – функция даты, возвращающая текущие дату и время.

Обычно выборка SYSDATE производится из фиктивной таблицы DUAL, которая принадлежит пользователю SYS и доступна всем пользователям. Она содержит один столбец с именем DUMMY и одну строку со значением X.

Арифметические операции с датами:

• результатом прибавления числа к дате и вычитания числа из даты является дата;

• результатом вычитания одной даты из другой является количество дней, разделяющих эти даты;

• прибавление часов к дате производится путем деления количества часов на 24.

Примеры обращения к функциям:

MONTH_BETWEEN (′01-SEP-98′, ′11-JAN-99′) → 16.6774

ADD_MONTHS (′11-JAN-99′, 6) → ′11_JUL-99′

NEXT_DAY (′01-SEP-99′, FRIDAY′) → ′08-SEP-99′

LAST_DAY (′01-SEP-99′) → ′30-SEP-99′

Функция TO_CHAR с датами

Формат: TO_CHAR (DATE, ′fmt′)

где fmt – модель формата, преобразующая дату на выводе, например:

TO_CHAR (DATE, ′MM/YY′).

Элементы формата даты

Элемент	Описание
YYY или YY или Y	Последние 3,2 или 1 цифра года
YEAR	Год словами
MM	Месяц в виде двузначного числа
MONTH	Название месяца
WW или W	Неделя года или месяца
DDD или DD или B	День года, месяца или недели
DAY	Название дня

Примеры обращения

Пример 11

Вывод даты заказа в формате типа «1of February 1999»

to_char (odate, ′fm DD “of” Month ′YYYY′)

Здесь использован префикс fm, который подавляет конечные пробелы в названиях месяцев и дней недели, оставляя результат переменной длины. Повтор префикса fm отменяет подавление.

Пример 12

Тот же самый пример, но с выводом даты в формате

«Sevent February 1997 08 : 00 : 00 AM»

to_char (0date, ′fm Dd spth “of” Month ′YYYY′ fm HH : MI : SS AM′).

Здесь использованы:

числовой суффикс для вывода числительных словами – dd spth (seventh);

HH, MI, SS – форматы времени;

AM/PM – индикатор «до полудня / после полудня».

Групповые функции

Групповые функции работают с множеством строк и возвращают один результат на группу. Групповые функции используются в списке SELECT и предложении HAVING (приложение 3). Вариант ALL в синтаксисе групповой функции рассматривает все значения столбца таблицы БД и принимается по умолчанию.

Все групповые функции, кроме count(*), игнорируют неопределенные значения. Для подстановки значения вместо неопределенного используйте функцию NVL.

Команда INSERT

Добавить новую строку в таблицу можно с помощью команды INSERT:

INSERT INTO таблица [(столбец [, столбец…])]

VALUES (значение [, значение…]);

где таблица – имя таблицы;

столбец – имена столбцов таблицы, в которые вносятся значения;

значение – соответствующие значения столбцов.

Поскольку можно вставить новую строку, содержащую значения для каждого столбца, перечисление столбцов в предложении INSERT необязательно. Но последовательность самих значений должна соответствовать стандартной последовательности столбцов в этой таблице.

Пример 13

```
INSERT INTO ORDERS
    VALUES (3014, 14.7 , SYSDATE , 2009, 1006);
```

Пример 14

```
INSERT INTO ORDERS (onum, amt, odate, snum)
    VALUES (3014, 14.7 , SYSDATE , 2009)
```

Вставка неопределенных значений возможна двумя способами:

1 способ – неявный: необходимо опустить столбец в списке столбцов.

2 способ – явный: нужно либо задать ключевое слово NULL в списке VALUES, либо задать пустую строку в списке VALUES (только для символьных строк и дат).

Пример 15

```
INSERT INTO SALESPEOPLE (SNUM, SNAME)
```

VALUES (1008, ′MIS′)

Пример 16

INSERT INTO SALESPEOPLE
VALUES (1008, ′Ron′, Null, Null)

Внимание! Проверив статус Null с помощью команды DESCRIBE SQL*PLUS, убедитесь в том, что столбец допускает неопределенное значение.

Вставка значений с помощью переменных подстановки:

команда INSERT позволяет пользователю вводить значения в интерактивном режиме с помощью переменных подстановки SQL*PLUS.

Пример 17

INSERT INTO SALESPEOPLE (SNUM, SNAME, CITY, COMM)
VALUES (′&salsnum′, ′&salsname′, ′&salscity′, ′&salscomm′);

Для дат и символьных значений амперсант и имя переменной должны быть заключены в апострофы.

Команду INSERT можно использовать для вставки в таблицу новых строк, значения которых копируются из уже существующих таблиц. Вместо предложения VALUES используется подзапрос.

Пример 18

INSERT INTO ORDERS_HISTORY
(SELECT * FROM ORDERS
WHERE ODATE < ′05/10/1997′)

Команда UPDATE

Команда UPDATE служит для обновления существующих строк в таблице и имеет *синтаксис*:

UPDATE <table>
SET column = value [, column = value: …]
[WHERE <condition>];
где table – имя таблицы;
column – имя обновляемого столбца;
value (значение) – новое значение или подзапрос;
condition (условие) – задает строки, которые необходимо изменить.

Пример 19

UPDATE ORDERS SET SNAME = NULL

Этот пример обновляет имя продавца для всех заказов значением NULL. Предложение SET включает в себя выражение для обновления таблицы. Если Вы хотите обновить конкретные записи, используйте предложение WHERE.

Пример 20

UPDATE ORDERS SET SNAME = 1004 WHERE SNAME = 1001.

В выражении SET оператора UPDATE можно также использовать подзапрос.

Пример 21

UPDATE ORDERS
SET cnum = (SELECT cnum FROM Customer WHERE sname = ′Lui′)
WHERE cnum = 3008

Этот оператор обновляет данные покупателя Lui с покупкой 3008.

Команда DELETE

Команда SQL DELETE используется для удаления из таблицы строк:

Например: DELETE FROM ORDERS.

Будьте аккуратны: этот пример удаляет все строки таблицы Orders. Если Вы хотите удалить конкретные строки таблицы, то в оператор нужно включить предложение WHERE.

Пример 22

DELETE FROM ORDERS WHERE sname = 1001.

Внимание! При попытке удалить строку с первичным ключом, который одновременно является внешним ключом в другой таблице, выдается сообщение об ошибке в связи с нарушением ограничения целостности.

РАБОТА В СРЕДЕ SQL*PLUS

Вход в SQL*PLUS. Возможности среды SQL*PLUS

SQL*PLUS – это среда для выполнения команд SQL и PL/ SQL с дополнительными возможности. С помощью команд SQL*PLUS Вы можете:

- получать информацию о структуре таблицы;
- редактировать команды SQL в буфере;
- сохранять файлы командами SQL в буфере для редактирования;
- выполнять файлы;
- получать оперативные справки.

Вход в SQL*PLUS из среды WINDOWS:

в Менеджере программ дважды щелкнуть на пиктограмме SQL*PLUS, после чего ввести имя пользователя, пароль и имя базы данных.

Вход в SQL*PLUS из командной строки:

Sqlplus [имя пользователя[.пароль @ базы данных]]

Для сохранения секретности своего пароля не следует вводить его в ответ на приглашение операционной системы. Лучше ввести только имя пользователя, а пароль ввести позже в ответ на приглашение «Password». После входа в SQL*PLUS на экране появится приглашение: SQL>.

Вывод структуры таблицы

Структуру таблицы можно получить с помощью команды

DESC[RIBE] <имя таблицы>,

где <имя таблицы> – имя существующей таблицы, представления или синонима, доступных пользователю.

Пример 23

SQL> Describe S_dept

Name	Null?	Type
ID	Not Null	Number(7)
Name	Not Null	Number(25)
Region_ID		Number(7)

Null? означает, что столбец должен содержать данные.

Команды редактирования

При вводе команды SQL она записывается в области памяти, называемой буфером SQL, и остается там до ввода новой команды. Команды SQL*PLUS вводятся по одной строке и не хранятся в буфере SQL. Ввод в буфер прекра-

щается вводом одного из символов окончания (точки с запятой или дробной черты). После чего на экран выводится приглашение SQL.

Команды редактирования SQL*PLUS

Команды	Описание
A[PPEND] текст	Добавить текст в конец текущей строки
C[HANGE] /старый/новый	*Заменить в текущей строке «старый» текст на «новый»*
C[HANGE] /текст/	Удалить текст из текущей строки
CL[EAR] BUFF[ER]	*Удалить все строки из буфера SQL*
DEL	Удалить текущую строку
DEL n	*Удалить строку с номером n*
DEL n m	*Удалить строки от m до*
I[NPUT]	Вставить неопределенное количество строк
I[NPUT] текст	*Вставить строку, состоящую из текста*
L[IST]	*Вывести список всех строк в буфере SQL*
L[IST] n	*Вывести одну строку с номером n*
L[IST] n m	*Вывести диапазон строк от n до*
R[UN]	*Вывести и выполнить команду из буфера SQL*
n	*Указать строку, которая должна стать текущей*
n текст	*Заменить строку n текстом*
0 текст	*Вставить строку перед строкой 1*

Команды SQL*PLUS для работы с файлами и оперативная справка

Для сервера Oracle команды SQL*PLUS являются вспомогательным средством. Они используются для управления средой, форматирования результатов запросов и работы с файлами.

Команды для работы с файлами

Команды	Описание
SAVE имя_файла.sql [REP[LACE]/APP[END]	Сохраняет в файле текущее содержимое буфера *SQL* . APPEND используется для добавления информации в существующий файл. REPLACE перезаписывает существующий файл
GET имя_файла	*Вызывает содержимое файла в буфер*
START имя_файла	Запускает выполнение файл
@ имя_файла	**Запускает выполнение файл**
SPOOL [имя_файла.ext/OFF/OUT]	*Записывает результаты запроса в файл. OFF – закрывает спул-файл (буферный файл). OUT – посылает результаты из спул-файла на системный принтер*
EXIT	*Выход из SQL*PLUS*

Создание отчета

Управление столбцом отчета осуществляется с помощью команды COLUMN.

Синтаксис: COL[UMN] [[column/alias] [option….]]

Команды для работы с файлами

Команды	Описание
CLE[AR]	Отменяет форматы столбцов
FOR[MAT] format	*Меняет отображение данных столбца*
HEA[DING] text	Задает заголовки столбца. Вертикальная линия | задает переход на новую строку в заголовке, если вы не используете выравнивание
JUS[TIFY] {align}	*Выравнивает заголовок столбца (не данные!)слева,справа, по центру*
NOPR[NT]	*Прячет данные*
NUL[L] text	*Задает text, который должен отображаться в случае неопределенных значений*
PRI[NT]	*Показывает столбец*
WRA[PPED]	*Переходит на следующую строку в конце строки*

! Длинную команду можно перенести на следующую строку. Для этого текущую строку следует закончить символом переноса (-).

Форматные модели Column

Элемент	Описание	Пример	Результат
An	*Задает ширину столбца n символов для вывода символьных данных и дат*	*Нет*	*Нет*
9	Один цифровой разряд	*999999*	*123456*
0	Вставляет ведущий ноль	*099999*	*012345*
$	Плавающий знак доллара	*$9999*	*$1234*
.	*Обозначает позицию десятичной точки*	*9999.99*	*1234.00*
,	*Обозначает разделитель тысяч*	*9,999*	*1,234*

! Вместо целого числа, количество цифр в котором превышает количество цифр в форматной модели, сервер Oracle7 выводит строку символов #.

Примеры:

```
COLUMN  last_name MEADING ‘Employee Name’ Format A15
COLUMN  salary JUSTIFY LEFT FORMAT $ 99,990.0
COLUMN  start_date FORMAT A& NULL ‘Not hired’
```

Сброс установок для столбца last_name :

```
COLUMN last_name CLEAR.
```

Заключение

SQL*PLUS – это исполнительная среда, которую можно использовать для посылки команд SQL серверу баз данных, а также для редактирования и

сохранения команд SQL. Команды могут выполняться из командной строки или командного файла.

Форматирование выдаваемых результатов в SQL*PLUS.

Создание отчета

Существует множество параметров настройки SQL*PLUS. Рассмотрим наиболее существенные из них.

1. Длина строк и страниц

Set linesize 80 (или 132) устанавливает ширину страницы;

Set pagesize 55 (или 60 устанавливает длину страницы.

2. Заголовки (верхние колонтитулы страниц)

Title ′заголовок′

Для переноса заголовка на следующую строку используется вертикальная черта | в месте переноса.

3. Нижние колонтитулы страниц

Btitle ′текст′ [{left, rigth}]

4. Сохранение выдачи SQL*PLUS в файле

Пример 25

Spool c:\report\out.lis

Для прекращения записи результатов в файл – spool off, spool out (с выдачей на печать).

5. Форматирование столбцов данных в распечатке

Пример 26

column last_name format a8 wrap heading′;

формат a8 ограничивает поле вывода восемью символами;

wrap – неумещающееся в 8 символов поле переносится на следующую строку;

heading – заголовок столбца размещенный на двух строках отчета.

Пример 27

column sales format 999, 999, 999, 999.99 hearding ′Sales′

Устанавливается выдача до 12 десятичных знаков целой части с выделением запятыми тысяч, миллионов и т.д.

6. Управление повторами

Break <имя_поля>

Устанавливают для подавления выдачи одинаковых значений некоторого поля в результате сортировки по этому полю.

Пример 28

```
break on dept_id;
select dept_id, last_name
from s_emp
orler by dept_id, last_name;
```

7. Пропуски и повторы

Skin n

Вставляется n пустых строк перед выдачей нового значения при подавлении повторов.

break on dept_id skip l;

8. Подавление повторов и вычисление промежуточных итогов

compute sum of <поле> on report – вычисление в отчете итоговой суммы

compute sum of <поле> on <поле2> – вычисление промежуточного итога по полю <поле1>, поле сортировки – <поле2>.

Переустановите условия подавления командой:

break on report skip 1 on <поле2>skip l;

Порядок выполнения работы

1. Ознакомьтесь с методическими указаниями к лабораторной работе.
2. Изучите дополнительные источники информации (список рекомендуемой литературы).
3. Запустите на выполнение продукт SQL*PLUS. Имя и пароль пользователя и строку связи с сервером ORUCL получите у преподавателя.
4. Ознакомьтесь с командами SQL*PLUS.
5. Изучите ER-модель базы данных в приложении 4.
6. Выполните в среде SQL*PLUS все запросы, формулировки которых приведены в приложении 5.
7. Выполните форматирование результатов запроса. Задание получите у преподавателя.

Лабораторная работа № 3 «Разработка приложений в ORACLE FORMS. Основные понятия и компоненты. Создание формы»

Продолжительность: 4 часа.

Целью настоящей работы является ознакомление с понятиями и методами продукта Forms и получение навыков создания одно- и многоформных приложений с помощью средств базы данных и GUI:

- создание модулей форм, включая компоненты для взаимодействия с БД и элементы управления GUI;
- создание триггеров для:
  - дополнительной проверки данных;
  - расширения функциональных возможностей при работе с элементами GUI;
  - реакции на события мыши;
  - взаимодействия с пользователем.
- связывание форм для создания многоформных приложений;
- изменение стандартного меню приложения и присоединения нового меню к модулям форм.

КОНЦЕПЦИИ И КОМПОНЕНТЫ

Модули Forms и их связи

Forms – это основной компонент Developer/2000. Forms поможет Вам быстро сконструировать forms-based приложение для просмотра и манипулирования данными произвольным способом. Основные модули Forms и связи продемонстрированы на рис. 1.

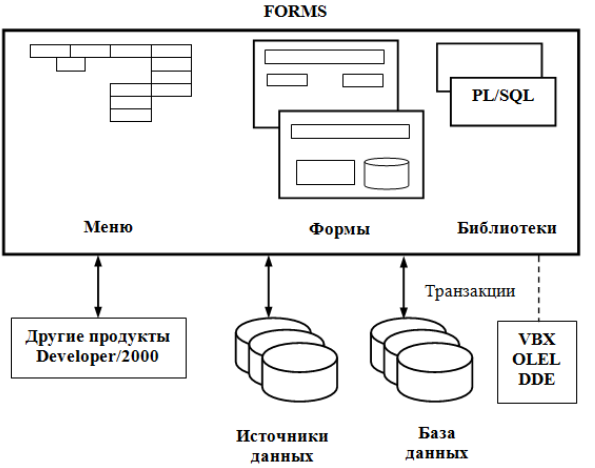


Рис. 1. Модули Forms и связи

Forms-приложение, как ясно видно из рис. 1, может компоноваться из многих модулей (файлов) следующих 3-х типов:

- Form (форма) включает в себя объекты интерфейса и данные из таблиц БД;
- Menu (меню);
- …..
- Library – коллекция PLISQL программных единиц и другие средства Developer/2000.

Компоненты Forms

Все компоненты Forms наглядно представим на рис. 2.

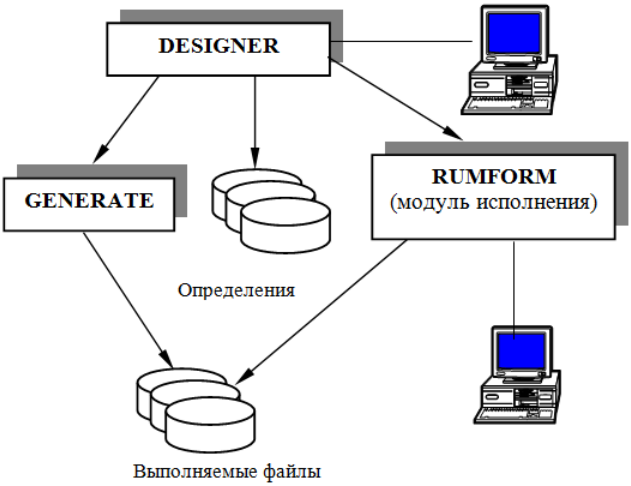


Рис. 2. Компоненты Forms

Пояснения к рис. 1.2:

- Forms Rumform – это программа, которая запускает исполнимые forms-приложения и модули вместе с ним. Безусловно, прежде чем запускать форму, ее необходимо сгенерировать с помощью компонента Generate (генератор);
- Forms Generate читает описание модуля, созданное дизайнером и создает исполнимый код;
- Forms Designer – инструментальная среда для выполнения всех этапов разработки приложения.

ORACLE FORMS DESIGNER
Структура дизайнера

Интерфейс программы Oracle Forms Designer состоит из четырех основных компонентов: навигатора объектов (Object Navigator), таблицы свойств (Property Sheet), редактора форматов (Layout editor) и редактора PL/SQL.

Навигатор объектов

В первую очередь Object Navigator (ON) служит для быстрого переключения между тремя другими компонентами интерфейса Oracle Forms Designer.

Навигатор позволяет работать с объектами и библиотеками, как находящимися на диске Вашего компьютера, так и хранящиеся в базе данных.

Буксируя мышью объекты в рабочую область Oracle Forms Designer, Вы можете добавить соответствующие функциональные возможности в создавшуюся форму.

Верхняя строка с инструментами интерфейса называется панелью инструментов. Инструментальные панели, идущие вдоль боковых частей экрана, носят название «Палитры инструментов».

Таблица свойств предназначена для задания атрибутов объекта. Конкретный набор атрибутов зависит от типа разрабатываемого объекта: модуль, макетный стол, блок. Существует три способа определения характеристик в таблице свойств:

- ввод текста в области ввода;
- щелчком мыши на стрелке раскрывающегося списка;
- нажатием кнопки <More>.

Набрав на клавиатуре новое значение атрибута, не забудьте нажать Enter для внесения изменения в таблицу.

Редактор PL/SQL

В редакторе осуществляется реализация основного функционального содержания, разрабатываемого приложения. В частности в редакторе производится редактирование и компиляция всех триггеров и процедур, используемых формой. Редактор запускается из навигатора объектов.

Порядок выполнения работы

По умолчанию при запуске дизайнера автоматически отображается новая форма с именем MODULE1. Вы можете открыть новую форму (после завершения работы со старой) двумя способами:

меню File/New;

выделитьForms в Объектном навигаторе, дважды щелкнуть мышью на клавише Create палитры навигатора.

Прежде чем мы создадим простую форму в Oracle Forms Designer, рассмотрим процедуру создания новых блоков.

Создание блока по умолчанию

Выберите в меню Tools строку New Block. На экране появится диалоговое окно, содержащее четыре страницы с закладками:

- страница General (общие). Укажите имя блока, таблицу БД, имя макетного стола, где будет находиться данный блок;

• страница Items (элементы). Выбирайте конкретные поля, включаемые в форму и устанавливайте атрибуты (Label - метка, Width – ширина, Object Type – тип объекта);

• страница Layout (формат). Устанавливайте ориентацию полей (Tabular – для блоков, состоящих из нескольких записей; Form – для блоков, содержащих только одну запись), а также ориентацию записей (вертикальную или горизонтальную, отступ между полями формы, количество записей, одновременно выдаваемое на экран, характеристики окон);

• страница Master/Detail – на этой странице устанавливайте взаимосвязи между блоками.

Создание новой формы с нуля

1. ON/Windows/ щелчок на знаке «+». На экране появилось название окна Windows Ø.

2. Выделите название Windows Ø и замените его на что-нибудь осмысленное.

3. Создайте макетный стол. ON/ два щелчка мышью на знаке «+» в строке Canvas – Views. Переименуйте Canvas1 в название окна (п. 2).

4. Создайте новый блок Меню Tools/New Block или в ON. Сделайте все необходимые установки на страницах.

5. Сохраните созданную форму Cntr/S.

6. Запустите форму нажатием Cntr/R. Когда форма появится на экране, для выполнения запроса выберите в меню Guery строку Execute. Получив из БД результат, для возврата в Designer дважды щелкните мышью на кнопке управляющего меню в заголовке окна Oracle Forms Rumform.

Управление атрибутами объектов макетного стола

1. Перейдите на соответствующий макетный стол.

2. Удерживая клавишу Ctrl, щелкните мышью на каждом поле рабочего стола. После того как все поля будут выбраны, вокруг каждого из них появится шесть маркеров управления.

3. Переведите курсор на макетный стол и нажмите правую кнопку мыши. В появившемся контекстном меню выберите Properties (свойства).

4. Выберите атрибут Background Color и установите его, например, в Green.

5. Закройте таблицу свойств двойным нажатием кнопки управляющего меню ее окна.

6. Сохраните применения Ctrl/S.

Создание отношений Master-detail

1. Выберите пиктограмму Oracle Forms Designer в программной группе диспетчера программ Microsoft Windows.

2. Создадим главный блок для таблицы S_emp.

3. Создадим подчиненный блок. На странице General введем имя таблицы S_dept. С помощью клавиши TAB перейдите к полю Block Name. Откройте закладку Items и нажмите кнопку Select Columns.

4. Откройте страницу Master/Detail и укажите в поле Master Block название блока emp.

5. В окне Join Condition укажите условие emp.dept_id = dept.dept_id.

6. Нажмите OK и Ctrl/S.

7. Запустите форму. В меню Query выберите Execute.

8. Перейдите к следующей записи меню Record/Next. Вы увидите вторую запись таблицы S_emp и соответствующую ей запись из таблицы S_dept.

Если Вы хотите сделать S_dept главным блоком, а S_emp – детальным, с отображением записей в табличном виде, тогда:

1) сделайте dept главным блоком;

2) на странице Layout в списке Style выберите Form;

3) сделайте emp детальным блоком;

4) на странице Layout в графе Style должно стоять Tabular, а в графе Orientation –Vertical.

В графе Records укажите число отображаемых записей, в графе Spacing введите S. Установите флажки Integrity Constraints и Scrollbar;

5) Откройте страницу Master/Detail и нажмите Select. Подтвердите единственное приведенное в диалоговом окне условие связи. Вернитесь в Master/Detail и OK;

6) сохраните созданную форму, запустите ее, выполните запрос.

Лабораторная работа № 4 «Создание приложений в ORACLE FORMS. Управляющие элементы интерфейса»

Продолжительность: 4 часа.

Целью настоящей работы является дальнейшее изучение возможностей Oracle Designer в части создания управляющих элементов данных в формах. В ходе выполнения работы Вы научитесь делать следующее:

- создавать новые элементы данных;
- управлять свойствами элементов данных;
- изменять навигационные характеристики;
- расширять возможности взаимодействия между элементами данных и базой данных;
- расширять функциональные возможности элементов данных;
- включить подсказки в приложения.

УПРАВЛЯЮЩИЕ ЭЛЕМЕНТЫ ИНТЕРФЕЙСА

Текстовый элемент Text Item 

Text Item – интерфейсный элемент, необходимый для выборки, вставки, обновления и удаления данных. Поведение элемента задается в окне свойств.

Вы можете создать Text Item в редакторе макета:

- щелчком мыши (click) активизируйте инструмент ;
- click макет. Отобразится текстовый элемент;
- double-click Text Item. Отобразится палитра свойств;
- установите свойства элемента;

Создание Text Item в Объектном Навигаторе (ON):

- локализируйте блок, в котором создается элемент;
- click поле ввода;
- щелкните мышью на пиктограмме Greate;
- double-click пиктограмму слева от поля ввода;
- установите свойство типа для Text Item;
- установите остальные свойства по умолчанию.

Свойства элемента разделены на 3 группы:

- визуальные (Display);
- типа (TYPE);
- записи (Record).

Визуальные свойства:

Canvas – канва, в которой отображается элемент;
Displayd – True/False – отображение;
X Position – X-координата по канве или на экране;
Y Position- Y-координата по канве или на экране;
Width – ширина;
High – высота;

Space Between Record – расстояние между строками;

Bevel – вид поля отображения (вдавленное, плоское, выдавленное);

Rendered – True/False. Экономит системные ресурсы (True), которые не отводятся, если элемент не виден;

Visual Attribute Name - атрибуты визуализации:

- Default – используются по умолчанию;
- Custom – индивидуальная настройка (можно изменить цвет и шрифт отдельных объектов);
- Numed – индивидуальная настройка нескольких объектов.

Даталогические свойства Text Item:

Mirror Item – специфицирует имя зеркального элемента блока, значение которого копируется в текстовый элемент Text Item. Таким образом все корректировки в элементах выполняются синхронно.

Date Type – тип вводимых значений;

Maximum Length – максимальная длина элемента (обычно соответствует длине базового столбца, если элемент с ним связан);

Format Mask – формат вывода элемента (форматная маска);

Range Low Value – минимальное значение;

Range High Value – максимальное значение;

Default Value – значение по умолчанию;

Copy Value from Item – специфицирует блок и элемент, используемый как источник копирования значения в Text Item, когда фокус установлен на одной из записей блока.

В приложении Summit это свойство имеет Item.Ord_id. Текстовый элемент никогда не будет использоваться для ввода, однако пользователь может контролировать с помощью этого элемента связь по внешнему ключу (лучше разместить его в этом случае на «пустую канву»).

Значения по умолчанию всегда включаются во вновь созданную запись.

Примеры значений по умолчанию

Системные переменные:

$$DATE $$DD-MON-YY

$$ DATETIME $$ DD-MON-YYYY hh:mi[:SS] – дата/время операционной системы;

$$ DBDATE $$ DD-MON-YY

$$ DBDATETIME $$ DD-MON-YYYY hh:mi[:SS] – дата/время базы данных.

Глобальные переменные:

: GLOBAL.Customer_id.

Параметры формы:

: PARAMETER.SALES_REP_ID.

Элемент формы:

ORDER.ID.

Последовательность:

: SEQUENCE.S_ORD_ID.Nextval.

Свойства навигации:

Enabled – определяет, можете ли Вы навигировать в элемент и манипулировать мышью;

Navigable – определяет, можете ли Вы навигировать в элемент по умолчанию с помощью функциональных клавиш и меню. Эффективнее всего установить:

Enabled - True

Navigable - False;

Next Navigation Item - устанавливает следующий активируемый элемент при навигации;

Previous Navigation Item – устанавливает предыдущий элемент при навигации.

Вы можете изменить или усилить возможность взаимодействия текстового элемента с базой данных при установлении соответствующих свойств базы данных.

Свойства базы данных:

Base Table Item – устанавливает базовый элемент, связанный со столбцом базовой таблицы;

Primary Key – устанавливают в True, если соответствующий столбец является частью первичного ключа;

Insert Allowed – устанавливает допустимость ввода в текстовом элементе;

Query Allowed – устанавливает допустимость запроса;

Query Length – устанавливает максимальную длину выражения запроса;

Update Allowed - устанавливает возможность изменения;

Updute Only if Null - устанавливает возможность изменения только если текущее значение в записи блока есть Null;

Lock Record – блокируется запись в блоке, используется только для небазовых элементов..

Дополнительные возможности

Вы можете расширить функциональность Text Item за счет некоторых дополнительных возможностей:

Case Restiction – устанавливает регистр ввода символов ;

Aligment – выравнивание;

Multi-Line – устанавливает, может ли текстовый элемент быть многострочным (только для символьного типа данных);

Wrap Style – стиль завершения строки (перенос строки);

Secure – устанавливает возможность визуализации значения элемента для пользователя, используется для ввода пароля;

Keep Position- сохраняет местоположение курсора при повторном входе;

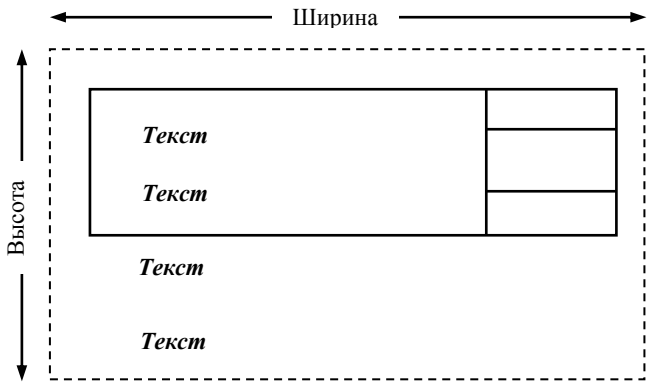
Auto Skip – True/False устанавливает автоматически ли курсор переходит на следующий элемент, используется совместно с Fixed Lenght свойством;

Vertical Scroll Bar – True/False включает линейку прокрутки в текстовый элемент.

Создание многострочного текстового элемента

Многострочный элемент (рис. 3) удобно использовать для вывода на экран длинных строк (например, комментарии, адрес и т.д.). Если Вы создади-

те такой элемент, непременно задайте ширину, высоту и максимальную длину текста. Кроме того, установите группу свойств Alignment.



Общая длина текста=Максимальная длина

Рис. 3. Многострочный элемент данных

Контекстно-зависимая справка

Оперативная справка на уровне элемента делается просто, с помощью Miscellaneous-свойств:

Hint - сообщение помощи;

Automatic Hint - True/False. Устанавливает автоматическое изображение Hint-сообщения.

Списки значений и редакторы

List of Values (LOV) – списки значений обеспечивают простой механизм выбора значения из списка элементов.

Список значений содержит значения элементов данных, может быть динамическим или постоянным, не зависеть от отдельных элементов данных. Список значений эффективен и может настраиваться. Список значений может быть создан на основе группы записей или SQL-запроса.

Порядок выполнения работы

Создание новых элементов данных, изменение их поведения и удаление

1. Удалите элемент Region_id и его статичный текст.

2. Сделайте так, чтобы в элементе Comments можно было отобразить многострочный текст.

3. Обеспечьте автоматический вывод уникального номера клиента для каждой новой записи, который невозможно было бы изменить. Используйте Sefuence.S – Customer_ID.NextVal

4. В форме Custx измените размеры, расположение элементов.

Используйте образец экрана (рис. 4) и таблицу (табл. 2).

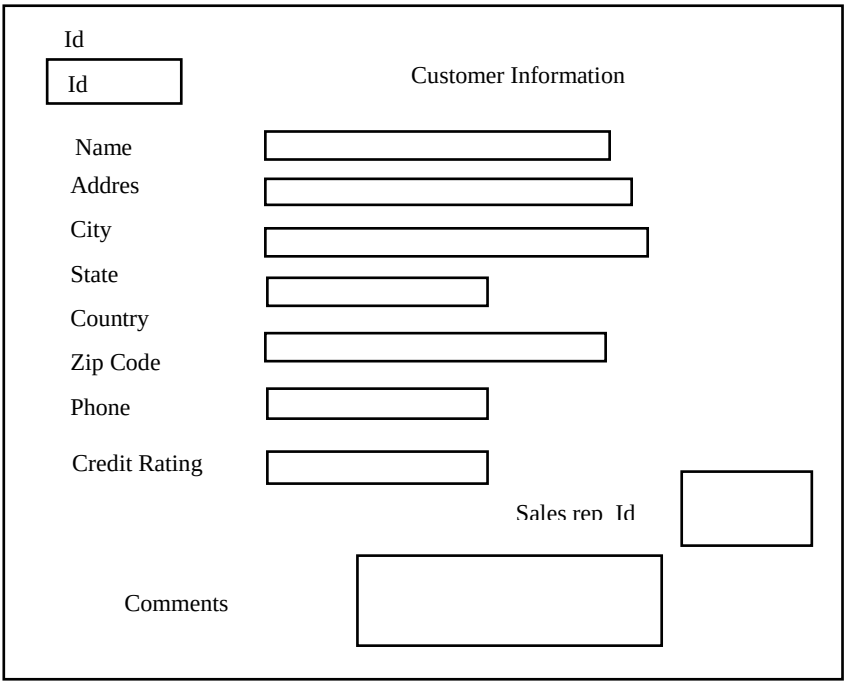


Рис. 4. Образец экрана

Таблица 2

Размеры элементов формы CUSTX

Элемент	Рекомендуемая ширина
ID	60
NAME	195
ADDRESS	195
CITY	195
STATE	130
COUNTRY	195
ZIP_CODE	85
PHONE	160
CREDIT RATING	65
DATES_REP_ID	65
COMMENTS	195

Форма ORDX

5. В блоке Order создайте новый элемент данных Customer_Name, который не должен быть связан с таблицей S_ord. Вставка, обновление и запрос данных в отношении этого элемента должны быть запрещены, а навигация возможна только с помощью мыши.

6. В блоке Order создайте элемент Sales_Rep_Name. Свойства аналогичны предыдущему.

7. Для TOTAL задайте также свойства, чтобы навигация была возможна только с помощью мыши, а вставка и обновление данных были запрещены. Формат для вывода значения – 9,999, 990.99 с правым выравниванием.

8. Для Date_Ordered сделайте так, чтобы при вводе каждой новой записи отображалась текущая дата.

9. В блоке Item создайте новый элемент данных Item_Total. Он не должен быть связан с таблицей S_Item. Вставка, обновление или запрос данных в отношекнии этого элемента должны быть запрещены, а навигация возможна только с помощью мыши. Разрешите только цифровые данные с выводом в формате 999, 990.99 .

10. Значения Price, Quantity_Shipped должны быть выравнены справа.

11. Измените элемент Quantity_Shipped. Задайте такие свойства, чтобы навигация была возможна только с помощью мыши и обновление данных запрещено.

12. Сохраните, сгенерируйте и запустите формы для проверки правильности внесенных изменений.

Самостоятельная работа

Согласно рабочей программе самостоятельная работа направлена на выполнение следующих мероприятий:

- подготовка к контрольным работам;
- подготовка к лабораторным работам;
- самостоятельное изучение разделов дисциплины.

Форма контроля и проверка достижения заявленных компетенций: проведение контрольных работ (в том числе тестовых), опрос перед проведением лабораторных работ, проверка отчетов, проверка конспектов самоподготовки.

Для проработки лекционного материала студентам, помимо конспектов лекций, рекомендуются изучить учебное пособие [1].

Для подготовки к лабораторным работам рекомендуется повторить соответствующие тематике разделы учебно-методического пособия [1], а также ознакомиться с порядком выполнения лабораторных работ, по настоящему руководству.

Для изучения тем теоретической части курса, отводимых на самостоятельную проработку, рекомендуется ознакомление со всеми разделами [1]. Кроме того, рекомендуется повторить разделы предложенной литературы [2-6], посвященные проектированию данных и построению пользовательских приложений.

Электронные варианты УМПО находятся в открытом доступе в компьютерных классах.

Для организации работы студентов требуется свободный доступ в компьютерные классы с наличием ОС Windows, MS Office, СУБД Oracle.

Необходимые базы данных, информационно-справочные и поисковые системы: образовательный портал университета (http://edu.tusur.ru), электронный каталог библиотеки http://lib.tusur.ru); электронные информационно-справочные ресурсы вычислительных

Рекомендуемая литература

1.Сенченко П. В. Организация баз данных: учеб. пособие / П.В. Сенченко. — Томск: факультет дистанционного обучения ТУСУРа, 2015. — 170 с. ил. [Электронный ресурс]. – URL: https://edu.tusur.ru/training/publications/5179

2.Веберова, И.И. Распределенные информационные системы : Учебное пособие / И. И. Веберова. – Томск : ТУСУР, 2001. – 348 с. : ил. (гриф СИБРУМЦ) (Наличие в библиотеке ТУСУР: экземпляры всего: 16, из них: аунл (14), счз1 (1), счз5 (1))

3.Дейт К. Дж. Введение в системы баз данных: Пер. с англ./ К. Дж. Дейт. - 6-е изд. - Киев; М.: Диалектика, 1998. - 784 с.: ил. - (Системное программирование). – (в пер.): Б.ц. (наличие в библиотеке ТУСУР: АНЛ – 1 экз.)

4.Саймон, Алан Р. Стратегические технологии баз данных: менеджмент на 2000 год: Пер. с англ./ Алан Р. Саймон; Ред. М. Р. Когаловский, Пер. М. Р. Когаловский, Пер. Н. И. Вьюкова, Пер. Г. Т. Никитина. - М.: Финансы и статистика, 1999. - 480 с.: ил. (наличие в библиотеке ТУСУР: счз1(1), счз5(1))

5.Кузнецов С.Д. Основы современных баз данных. Информационно-аналитические материалы Центра Информационных технологий. М.– режим доступа к сайту http://citforum.ru/database/osbd/contents.shtml свободный (дата обращения: 8.10.2016)

6.Кириллов В.В. Основы проектирования реляционных баз данных. Учебное пособие режим доступа к сайту http://citforum.ru/database/dbguide/index.shtml свободный (дата обращения: 11.10.2016)

ПРИЛОЖЕНИЕ 1

СТРУКТУРЫ ТАБЛИЦ

Таблица Salespeople (Продавцы)

SNUM	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	Barcelona	.13
1004	Monica	New York	.11
1007	Rifkin	London	.14
1003	Alex	San Diego	.10

Таблица Customers (Покупатели)

CNUM	SNAME	CITY	SNUM
2001	Hoffman	London	1001
2002	Giovanni	Rome	1003
2003	Lui	San Diego	1003
2004	Grass	Berlin	1002
2006	Clemens	London	1001
2008	Hill	Rome	1003
2007	Pereira	New York	1004

Таблица Orders (Заказы)

ONUM	AMT	ODATE	CNUM	SNUM
3001	10.29	10/03/1997	2007	1003
3003	767.16	10/03/1997	2001	1001
3002	1900.10	10/05/1997	2007	1004
3005	5160.45	10/05/1997	2003	1003
3006	1098.43	10/05/1997	2002	1003
3009	75.90	10/06/1997	2004	1002
3007	4723.00	10/07/1997	2006	1001
3008	1309.09	10/08/1997	2004	1002
3010	98.56	10/08/1997	2006	1001

ПРИЛОЖЕНИЕ 2

ОДНОСТРОЧНЫЕ ФУНКЦИИ SQL В ORACLE

Таблица 1

Числовые SQL-функции

SQL-функции	Описание
ABS (x)	*Возвращает абсолютное значение аргумента x*
CEIL (x)	*Возвращает наименьшее целое, большее или равное x*
COS (x)	*Возвращает косинус x*
EXP (x)	*Возвращает e в степени x*
FLOOR (x)	*Возвращает наибольшее целое, меньшее или равное x*
LN (x)	*Возвращает натуральный логарифм x*
LOG (x,y)	*Возвращает основание x логарифма y*
MOD (x, y)	*Возвращает остаток деления x на y*
POWER (x, y)	*Возвращает x в степени y*
ROUND (x, [y])	*Возвращает x, округленный до десятичных знаков справа от десятичной точки. По умолчанию y равен 0. Если y имеет отрицательное значение, то x округляется справа от десятичной точки*
SIGN (x)	*Возвращает знак x: -1, если он отрицательный, и 1, если положительный*
SIN (x)	*Возвращает синус x*
SQRT (x)	*Возвращает квадратный корень из x*
TAN (x)	*Возвращает тангенс x*
TRUNC (x, [y])	*Возвращает значение x, усеченное до y десятичных знаков. По умолчанию y равен 0.*

Таблица 2

Символьные SQL-функции

SQL-функции	Описание
ASCII (x)	*Возвращает десятичное представление символа x*
CHR (x)	*Возвращает символ, соответствующий целому x*
COSNCAT(x,y)	*Конкатенирует строку x со строкой y*
INITCAP(x)	*Возвращает строку x, в которой первая буква в каждом слове преобразована в верхний регистр*
INSTR (w,x[,y[,z]])	*Возвращает позицию первого символа строки x в строке w (если она в ней присутствует). Если задается y, то поиск начинается с позиции y в строке w. Если задается x, то ищется вхождение z строки x в строку w, начиная с символа y*

Продолжение табл. 2

SQL-функции	Описание
INSTRB (w,x[,y[,z]])	*Эквивалентна INSTR, но y и возвращаемое значение возвращаются в байтах*
LENGTH (x)	*Возвращает длину строки x*
LENGTHB (x)	*Возвращает длину строки x в байтах*
LOWER (x)	*Возвращает строку x, все буквы которой преобразованы в нижний регистр*
LPAD (x, y[,z])	*Возвращает строку x, дополненную слева пробелами до длины y. Строку-заполнитель можно задать с помощью аргумента z*
LTRIM(x[,y])	*Отбрасывает строки слева от строки x. Необязательный набор отсекаемых символов задается аргументом y. По умолчанию y пуст, что означает удаление пробелов, дополняющих строку слева*
REPLACE(x,y[,z])	*Возвращает строку x, в которой все вхождения y заменены на z. Если z не задается, то все вхождения y удаляются*
NVL (x, y)	*Возвращает y, если x – неопределенное значение*
RPAD (x, y[,z])	*Возвращает строку x, дополненную справа пробелами до длины y. Строку-заполнитель можно задать с помощью аргумента z*
RTRIM (x[,y])	*Отбрасывает символы справа от строки x. Необязательный набор отсекаемых символов задается аргументом y. По умолчанию y пуст, что означает удаление пробелов, дополняющих строку справа*
SUBSTR (x, y[,z])	*Возвращает подстроку строки x, начиная с символа y. Если вы задаете z, то возвращает часть длиной z символов*
UPPER (x)	*Возвращает строку x, преобразовав ее символы в верхний регистр*
LOWER (x)	*Возвращает строку x, преобразовав ее символы в нужный регистр*

Таблица 3

SQL-функции Oracle7 для работы с датой

SQL-функции	Описание
ADD_MONTHS (x,y)	*Возвращает дату x, плюс y месяцев*
LAST_DAY (x)	*Возвращает последний день месяца, заданного датой x*
MONTHS_BETWEEN (x,y)	*Возвращает число месяцев между датами x и y*

Продолжение табл. 3

SQL-функции	Описание
NEW_TIME(x,y,z)	*Возвращает соответствующее время во временной зоне z времени x во временной зоне y*
NEXT_DAY (x,y)	*Возвращает дату первого дня недели, заданного y, после даты x*
ROUND (x[,y])	*Возвращает значение x, округленное до ближайшего числа в формате, заданном y (например, месяц года)*
SYSDATE	*Возвращает текущую дату и время*
TRUNC (x, [y])	*Возвращает x с временем дня, усеченным по спецификации формата y*

Таблица 4

SQL-функции преобразования

SQL-функции	Описание
TO_CHAR (x[,y])	*Преобразует дату или число в символьную строку. Если x – это дата, то y задает формат*
TO_DATE (x[,y])	*Преобразует символьную строку x в дату формата y*
TO_NUMBER (x[,y])	*Используя формат y, преобразует символьную строку в число*

Таблица 5

Прочие SQL-функции

SQL-функции	Описание
UID	*Возвращает целое число, уникальным образом идентифицирующее текущего пользователя*
USER	*Возвращает имя текущего пользователя базы данных*

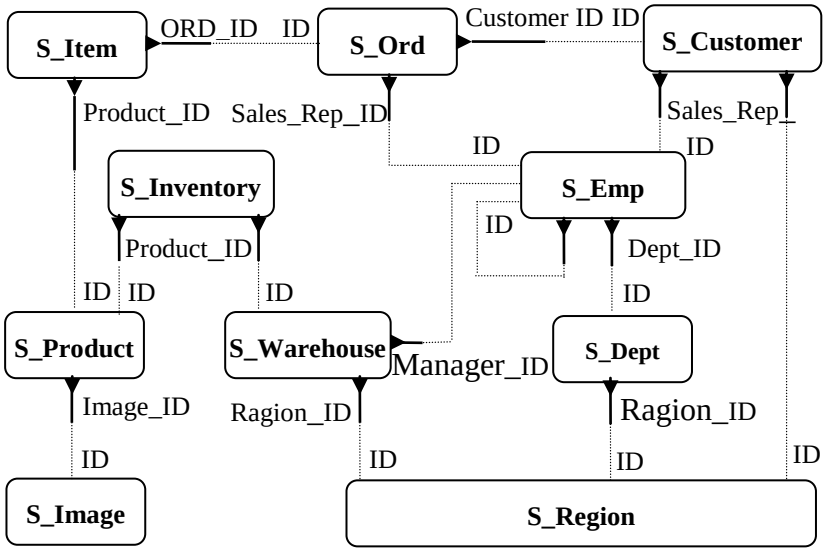
ПРИЛОЖЕНИЕ 3

ГРУППОВЫЕ ФУНКЦИИ

Функция	Описание
AVG (Distinct | ALL| n)	Среднее значение n без учета неопределенных значений
Count (Distinct|ALL| выражение|*)	Количество строк только с определенными результатами вычисления выражения. По “*” подсчитываются все строки, включая повторяющиеся и строки с неопределенными значениями
Max (Distinct|All| выражение*)	Максимальное значение выражения
MIN (Distinct|All| выражение*)	Минимальное значение выражения
STDDEV(Distinct|All| n)	Стандартное отклонение n без учета неопределенных значений
SUM (Distinct|All| n)	Сумма значений n без учета неопределенных значений
VARIANCE (Distinct|All| n)	Дисперсия n без учета неопределенных значений

ПРИЛОЖЕНИЕ 4

ER-МОДЕЛЬ ДЛЯ ФИРМЫ «Summit Sporting Goods»



ОПИСАНИЕ ИНФОРМАЦИОННЫХ ПОТРЕБНОСТЕЙ

“Я – менеджер оптовой фирмы по продаже спортивных товаров, которая выполняет заказы предприятий розничной торговли по всему миру. Нашими заказчиками являются магазины (некоторые из наших служащих предпочитают называть их клиентами). Сейчас у нас 15 клиентов по всему миру, и мы стараемся увеличить их количество. Самыми крупными из них являются магазины “Big John’s Sports Emporium” в Сан-Франциско, Калифорния, США, и “Womansport” в Сиэттле, Вашингтон, США. Мы должны знать индентификационный номер и имя каждого клиента. Можно также хранить его адрес (включая город, штат, почтовый индекс и страну) и номер телефона. Для наилучшего обслуживания клиентов у нас есть склады в различных регионах. Прежде всего, нам необходимо знать номер каждого заказа. Но дата заказа, дата отгрузки и способ платежа тоже могут быть получены, если эта информация имеется. Весь мир мы условно поделим на пять регионов: Северная Америка, Южная Америка, Африка/Средний Восток, Азия и Европа. Здесь нам достаточно иметь номер региона и его название. Чтобы знать, откуда лучше всего доставлять товары по каждому заказу, мы стараемся закрепить каждого клиента за каким-либо регионом. Каждый склад должен иметь номер. Можно также хранить его адрес (включая город, штат, почтовый индекс и страну) и

номер телефона. Сейчас в каждом регионе у нас только один склад, но мы надеемся, что вскоре их станет больше.”

“На нашей оптовой фирме по продаже спорттоваров я заведую отделом приема заказов. Отдел отвечает за размещение и контроль выполнения заказов клиентов. Нам необходимо знать номер и название каждого отдела. Иногда, если это не срочно, клиенты присылают заказ по почте, но чаще всего звонят или присылают факс. Мы надеемся расширить свой бизнес за счет немедленной информационной обработки каждого заказа. При наличии нужного товара на одном из наших складов мы можем обещать отгрузить его на следующий день. Если у нас есть информация, мы отслеживаем размер товарного запаса, минимальное количество, при котором необходимо пополнить запас, максимальное количество, причину отсутствия товара на складе и дату восполнения конкретного товара. Мы планируем автоматически отправлять файлы об отгрузке товаров через нашу систему отгрузки.”

“Мой отдел просто следит за тем, чтобы клиенты получили правильную информацию по оплате и чтобы на их счетах было достаточно средств для кредита. Кроме этого, мы можем хранить общие сведения о клиенте.”

“Мы должны следить за тем, чтобы все товары, заказанные клиентами, присутствовали на складе. Для каждого товара мы храним его номер. Можно также хранить цену товара, количество в наличии и отгруженное количество, если такая информация имеется. Если нужный товар на складе есть, мы хотим обработать заказ и сообщить нашему заказчику номер заказа и его итоговую сумму. Если нужного количества товара на складе нет, заказчик должен сказать, что нам делать, и ждать, пока мы сможем отгрузить заказанный товар полностью или выполнить заказ частично.”

“Бухгалтерия отвечает за ведение информации о клиентах – особенно за присвоение им новых номеров. Мой отдел может разрешить внести изменение в информацию о клиенте только в случае, если он сделал заказ, а его платежные реквизиты или адреса грузополучателей изменились. Нет, за сбор платежей мы не отвечаем. Этим занимается отдел дебиторских счетов. Думаю также, что в этом участвуют и торговые представители, так как размер их комиссионных зависит от клиентов, которые платят деньги. Нам необходимо знать номер и фамилию каждого торгового представителя или служащего. Иногда требуется его имя, имя пользователя (в базе данных), дата начала работы в компании, должность и месячный оклад. Можно также хранить данные о проценте комиссионных служащего и любые замечания о нем.”

“Наш персонал по приему заказов прекрасно разбирается в нашей продукции. Мы часто проводим совещания с представителями отдела маркетинга, где они информируют нас о новых товарах. Это возможно, благодаря тому, что мы заключаем сделки с небольшим количеством специально подобранных клиентов и поддерживаем для них специализированные линии товаров. Мы должны знать номер и наименование каждого продукта. Время от времени может потребоваться описание, предполагаемая цена и минимальное количество товара, которое можно хранить. В случае необходимости хотелось бы также иметь возможность получить очень длинные описания наших товаров и их фотографии.”

СТРУКТУРА ТАБЛИЦ

1. S_CUSTOMER

Name	Null?	Type
ID	Not Null	Number (7)
NAME	Not Null	Varchar 2 (50)
PHONE		Varchar 2 (25)
ADDRESS		Varchar 2 (400)
CITY		Varchar 2 (30)
STATE		Varchar 2 (20)
COUNTRY		Varchar 2 (30)
ZIP_code		Varchar 2 (75)
Credit_Rating		Varchar 2 (9)
Sales_Rep_Id		Number (7)
Region_Id		Number (7)
Comments		Varchar 2 (255)

2. S_DEPT

Name	Null?	Type
ID	Not Null	Number (7)
NAME	Not Null	Varchar 2 (25)
Region_Id		Number (7)

3. S_EMP

Name	Null?	Type
ID	Not Null	Number (7)
LAST_NAME	Not Null	Varchar 2 (25)
FIRST_NAME		Varchar 2 (25)
USERID	Not Null	Varchar 2 (8)
START_DATE		Date
COMMENTS		Varchar 2 (255)
MANAGER_Id		Number (7)
TITLE		Varchar 2 (25)
Dept_Id		Number (7)
SALARY		Number (11,2)
COMMISSION_PCT		Number (4,2)

4. S_IMAGE

Name	Null?	Type
ID	Not Null	Number (7)
Format		Varchar 2 (25)
USE-FILENAME		Varchar 2 (25)
FILENAME		Varchar 2 (25)
Image		Long Row

5. S_INVENTORY

Name	Null?	Type
Product_Id	Not Null	Number (7)
Warehouse_Id		Number (7)
Amovnt_In_Stock		Number (9)
Reorder_Point		Number (9)
Max_In_Stock		Number (9)
Out_Of_Stock_Explanation		Varchar 2 (255)
Restock_Date		Date

6. S_ITEM

Name	Null?	Type
ORD_Id	Not Null	Number (7)
ITEM_Id	Not Null	Number (7)
PRODUCT_Id	Not Null	Number (7)
PRICE		Number (11,2)
QUANTITY		Number (9)
QUANTITY_SHIPPED		Number (9)

7. S_ORD

Name	Null?	Type
ID	Not Null	Number (7)
CUSTOMER_Id	Not Null	Number (7)
DATE_ORDERED		Date
DATE_SHIPPED		Date
SALES_REP_ID		Number (7)
TOTAL		Number (11,2)
PAYMENT_TYPE		Varchar 2 (6)
ORDER_FILLED		Varchar 2 (1)

8. S_PRODUCT

Name	Null?	Type
ID		Number (7)
NAME		Varchar 2 (50)
SHORT_DESC		Varchar 2 (255)
LONGTEXT_ID		Number (7)
IMAGE_ID		Number (7)
SUGGESTED_WHLSL_PRICE		Number (11,2)
WHLSL_UNITS		Varchar 2 (25)

9. S_REGION

Name	Null?	Type
ID	Not Null	Number (7)
NAME	Not Null	Varchar 2 (50)

10. S_TITLE

Name	Null?	Type
TITLE	Not Null	Varchar 2 (25)

ПРИЛОЖЕНИЕ 5

СПИСОК ЗАПРОСОВ

1. Покажите структуру таблицы S_Customer:

а) выберите всю информацию из таблицы;

б) получите список названий и номеров телефонов всех фирм-клиентов;

в) получите список названий и номеров телефонов всех фирм-клиентов, причем номер телефона должен быть на строке первым.

2. Выполните следующие действия с таблицей S_Customer:

а) создайте запрос для вывода названия, номера и кредитного рейтинга всех фирм-клиентов, имеющих торгового представителя под номером 11. Сохраните команду QSL в файле L1q2;

б) выполните запрос у файла L1q2.

3. Загрузите файл L1q2 в буфер QSL. Присвойте столбцам заголовки Company, Company Id и Rating. Выполните запрос еще раз. Сохраните отредактированный запрос в файле p2q2.

4. Загрузите файл p2q2. Отсортируйте результат запроса в порядке убывания номеров клиентов. Выполните запрос.

5. Покажите структуру таблицы S_Emp. Получите список имен, фамилий, номеров отделов для служащих отделов 10 и 50. Отсортируйте список по фамилиям в алфавитном порядке. Объедините имя с фамилией и назовите столбец «Emploees».

6. Получите информацию по всем служащим, в фамилии которых имеется буква «s».

7. Получите имя пользователя и дату начала работы всех служащих, занятых с 14 мая 1990 г. и 26 мая 1991 года. Результаты запроса отсортируйте по убыванию дат начала работы.

8. Таблица S_Emp.

Составьте запрос для вывода фамилии и зарплаты всех служащих, месячный заработок которых не лежит в интервале от 1000 до 2500.

9. Получите список фамилий и заработной платы всех служащих отделов 31, 42 и 50, зарабатывающих более 1350. Назовите столбец фамилий «Employee Name», а столбец заработной платы – «Monthly Salary».

10. Получите список фамилий и дату найма всех служащих, пришедших в 1991 году.

11. Получите список имен и фамилий всех служащих, не имеющих менеджера.

12. Покажите структуру таблицы S_Product.

Перечислите в алфавитном порядке все товары, названия которых начинаются с «Pro».

13. Получите номера служащих, фамилии и заработную плату, повышенную на 15 % и округленную до целого.

14. Получите фамилии, номера отделов и заработную плату всех служащих. Результат сортируется по номерам отделов, а внутри отделов – в порядке убывания заработной платы.

15. Получите имена и фамилии всех служащих с фамилией Patel.

16. Для каждого служащего вычислить количество месяцев со дня начала работы до настоящего времени. Результаты отсортируйте по количеству отработанных месяцев. Количество месяцев округлите до ближайшего целого.

17. Покажите фамилию каждого служащего, дату и день недели, когда он был нанят на работу. Результаты отсортируйте по дням недели, начиная с понедельника.

18. Получите текущую дату с помощью фиктивной таблицы DUAL.

19. Получите фамилии и количества отработанных недель для служащих отдела 43.

20. Получите в хронологическом порядке номера каждого товара, запас которого пополняется, первой пятницы после даты пополнения запасов и последнего дня месяца, когда пополнялся запас.

21. Сформируйте сообщение о выполнении заказа с указанным номером для каждого заказа, сделанного 21 сентября 1992 года. Сообщение должно содержать сумму заказа.

Например:

Note:

Order 107 was filled for a total of $142.71.

22. Напишите запрос для получения следующей информации по каждому служащему:

<имя служащего> зарабатывает <зарплата в месяц>, но желает <утроенная зарплата>.

Например:

ALLEN зарабатывает $1100 в месяц, но желает $3300.

23. Составьте отчет, содержащий для каждого служащего фамилию, номер отдела и название отдела. Используйте префиксы в виде имен таблиц.

24. Составьте отчет, содержащий наименование клиента, номера региона и названия региона для всех клиентов. Используйте названия столбцов и псевдонимы таблиц. Названия столбцов: «Customr Name», «Region_Id», «Region_Name».

25. Вывести для каждого клиента его наименование, а также фамилии и идентификационные номера торгового представителя. В список включаются наименования даже тех клиентов, которые не имеют торгового представителя.

! Используйте внешнее соединение.

26. Вывести имена сотрудников и их менеджеров.

27. Получите список заказчиков, общая сумма заказа которых превышает 100.00. Список должен включать наименование заказчика, заказанные им товары и их количество.

28. Покажите наибольшую и наименьшую сумму заказа из таблицы S_Ord. Озаглавьте столбцы Highest и Lowest.

29. Составьте запрос для вывода минимальной и максимальной заработной платы по всем должностям в алфавитном порядке.

30. Определите количество менеджеров без их перечисления.

31. Получите номер каждого заказа и количество позиций в нем. Столбец с количеством позиций озаглавьте «Number of Items».

32. Получите номер каждого менеджера и заработную плату самого низкооплачиваемого из его подчиненных. Исключите группы с минимальной заработной платой менее 1000. Отсортируйте по размеру заработной платы.

33. Получите имя, фамилию и дату начала работы всех служащих, работающих в одном отделе с Magee.

34. Для каждого служащего с заработной платой выше средней покажите номер служащего, имя, фамилию и имя пользователя.

35. Получите наименование и краткое описание всех товаров, незаказанных ни разу в сентябре 1992 г.

36. Получите наименование и кредитный рейтинг всех клиентов, чьим торговым представителем является Andre Dummus.

37. По регионам 1 и 2 покажите имя каждого торгового представителя, фамилии его заказчиков и общую сумму заказов каждого заказчика.