Оглавление

Лекции по курсу
БД САПР

Преподаватель: Власов Андрей Игоревич
Кафедра: ИУ4

Лекция №1,2

07.02.07 г

Цель курса: подготовка в области проектирования и эксплуатации конструкторско-технологических баз данных на основе реляционной СУБД, в частности СУБД Oracle, MySQL.

СУБД Oracle ценна реализацией процедурного расширения SQL. Главная цель – вся логика приложения должна быть на сервере.

Решаемые задачи:

- Изучение основ, принципов и методологии применения информационных технологий и автоматизированного проектирования ЭС.
- Формализация объектов проектирования.
- Разработка автоматизированной системы управления конструкторско-технологическим проектированием (АСУ КТП).
- Построение АСУ КТП на основе архитектуры клиент-сервер.

ДЗ. К следующему семинару принести бланк задания.

Литература:

1. Норенков. Основы автоматизированного проектирования.
2. Иванова. Технология методов программирования.
3. Oracle. В 10 томах. Том Программирование на PL/SQL.
4. Краткое практическое руководство разработчика по языку PL/SQL. Лежит на citforum.ru.
5. Гуччи, Якобсон.

Учебный план:

- Экзамен. Для допуска надо 60 баллов. 90 баллов и выше – это пять.
- ДЗ – 20 баллов. Срок 10 мая.
- КР основы программирования на PL/SQL. Это будет электронный тест PL/SQL programmer. 5 баллов. Все вопросы задаются на английском языке.
- Экзамен 15 баллов.
- Защита фирмы 10 баллов.

СУБД Oracle. Ориентация на профессиональные СУБД

Сложность – в командной работе. Необходимо программировать в одном стиле.

Экзамен автоматом, если принести личный сертификат Oracle DBA.)))

Что отличает профессиональные СУБД от пользовательских:

1. Оптимизированный многопользовательский режим работы с развитой системой транзакционной системой транзакционной обработки, что обеспечивается многочисленным пользователям возможность работы с базой данных, не мешая друг другу.
2. Надёжные средства защиты информации (трёхзвенная архитектура защиты на уровне сети, на уровне БД, на уровне клиентской ОС).
3. Эффективные инструменты для разграничения доступа к БД.
4. Поддержка широкого диапазона аппаратно-программных платформ.
5. Реализация распределённой обработки данных.
6. Возможность построения гетерогенных и распределённых сетей.
7. Развитые средства управления, контроля, мониторинга и администрирования сервера БД.
8. Поддержка таких эффективных инструментариев как: словари данных, триггеры, функции, процедуры, пакеты и т.п.

Требования к выполнению ДЗ.

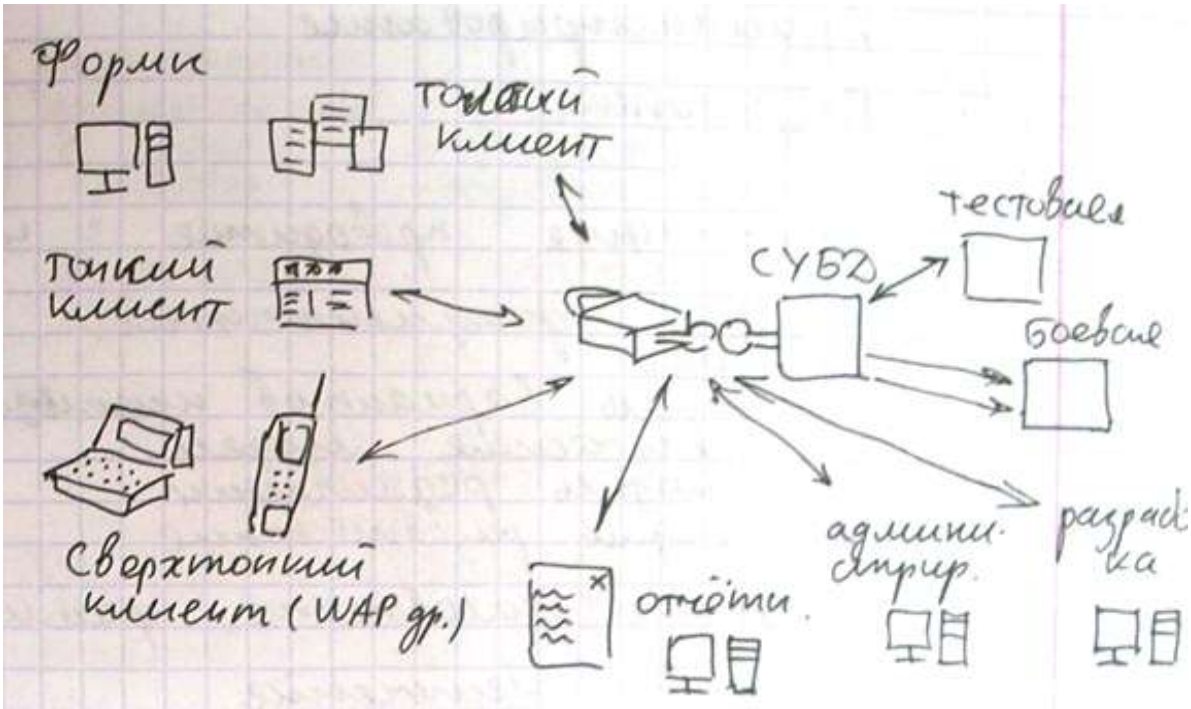


Рис. 1.1

Разработать тестовую и боевую БД. Боевая БД развёртывается на oracle.iu4.ru прогоном SQL скриптов.

БД определяется SIDом (System IDentificator). На боевом разделе SID=ORCL.

Первая задача – задача авторизации.

Предназначение информационной системы. **Наши задачи**:

1. Предоставление доступа стороннего пользователя к ресурсам информационной системы посредством интерфейсных форм (тонкий, толстый клиент).
2. Предоставление сводной отчётности по бизнес-процессу.
3. Администрирование.
4. Разработка.

Структура представления информации в ДЗ.

Содержание документации соответствует перечню моделей методики RUP (5 моделей). В каждой модели представлено содержание.

1. Модель вариантов использования (аналог контекстной диаграммы, показывает, как система работает с пользователем).
2. Логическая модель (описывает пакеты и классы).
3. Модель-реализация. Описывает из каких физических файлов состоит система.
4. Модель развёртывания. Связь железа и софта.
5. Пятую не назвал)))

Заключение.

Список источников.

В РПЗ включается методическое обеспечение (инструкция работы пользователя).

Логин и пароль пользователя: guest-guest.

Данные документы создаются на основе соглашения по разработке программного кода, которое лежит на сервере.

В основе программного кода, воспринимаемого всеми членами группы разработчиков лежит соглашение по разработке программного кода (смотри oracle.iu4.bmstu.ru раздел «Проекты»).

Последовательность подготовки специалистов для работы с СУБД Oracle.

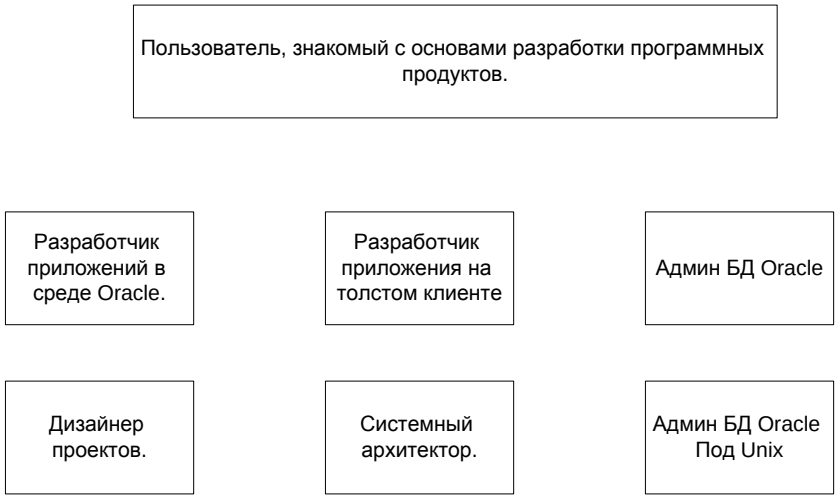


Рис. 1.2

Установка СУБД Oracle 9i:

Требования к установке – ОЗУ 1 ГБ (swap file 1 ГБ), XP pro.

Устанавливаем серверный вариант.

1. Запускаем Oracle installer.
2. Далее и т.д.
3. Указание файловой структуры установленной СУБД. Куда ставить. Объём – 9 ГБ.
4. Выбор варианта установки. Выбираем Oracle Database (СУБД и клиентская часть). Обязательно проверить, чтобы в локализации продукта было 2 языка (русский, английский).
5. Выбор типа устанавливаемой СУБД. Выбираем Enterprise.
6. Выбор типа создаваемой БД. Выбираем General.
7. Выбор доступных компонентов.
8. Осуществляется инициализация статуса компонентов. Проверка.
9. Создание БД. Обязательно сразу создать БД.
10. Установка параметров БД. Name и SID одинаковы – ORCL.
11. Анализ отчёта по установке. В течение 4 часов пьём кофе)
12. Инициализация компонентов конфигурации. Настройки параметров конфигурации.

Создание тестовой конфигурации БД (только для любителей Customize):

1. Создаём БД.

2. Выбираем тип.
3. Указываем SID.
4. Указываем устанавливаемые расширения (везде ставим галки).
5. Вариант сервера (dedicated server mode).
6. Параметры. Объём ОЗУ и т.п. Режим архивирования (не надо).
7. Файлы БД. Файлы конфигурации init.ora. Его не трогать.
8. Сохранить отчёт в HTML. Подключить под номером v0010. Это первая задача.

Итого на дом:

- Бланк задания на подпись.
- Поставить Oracle. Выложить отчёт.

Лекция №3

14.02.07 г

Замена. Смотри презентацию.

Лекция №4, 5

22.02.07 г

Объектно-ориентированное моделирование.

Методология RUP

RUP – методология проектирования информационных систем фирмы Rational Software, в основу которой положен язык UML (unified modeling language – унифицированный язык моделирования). UML развивается под эгидой консорциума OMG (Object Management Group – сообщество по технологии управления объектами).

Цель моделирования – получение исходного кода заголовочных файлов ПО.

В основе методологи лежит ряд диаграмм классов, диаграммы процессов: сценариев и взаимодействий объектов (отражают поведенческий аспект), диаграммы использования.

Интерфейс нужно делать из множества последовательных и простых форм.

В основе UML лежат методы объектно-ориентированного проектирования.

Объекты, сообщения и классы в ООП.

Класс – это структурный тип данных, который включает описание полей данных (атрибуты), а также процедур и функций (методы), работающих с этими полями данных.

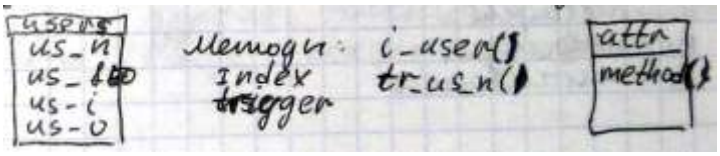


Рис. 3.1 Пример класса типа сущность – таблица.

В ПО используются переменные типа класс. Такие переменные принято называть объектами.

Методы таблицы – индексы, триггеры и др.

Класс и объект – элементы представления данных.

Основные свойства ОО проектирования.

- Наследование – объектам потомкам присущи свойства объектов родителей.
- Полиморфизм – изменение свойств объектов при сохранении названия объекта.
- Инкапсуляция – вхождение частных конструкций в более общие композиции.
- Наполнение – механизм подключения объекта или структуры объектов к некоторому классу, реализующему управление всей структурой.

Цель ООП – возможность конструирования сложных объектов из совокупности простых.

Создание систем на базе IP блоков.

Здесь IP – intelligent properties. Технология называется IP core.

Основные принципы ООП.

- Абстрагирование.
- Ограничение доступа.
- Модульность.
- Иерархичность.
- Типизация.
- Параллелизм.
- Устойчивость.

Модель – представляет собой совокупность диаграмм, описывающих отдельные аспекты структуры и поведения ИС.

- **Модель вариантов использования** – представляет собой описание функциональности ПО с точки зрения пользователя (система).
- **Логическая модель** – описывает ключевые абстракции ПО (классы, интерфейсы), т.е. набор логических средств.

- **Модель реализации** (модель компонентов) определяет реальную организацию программных модулей в среде разработки.
- **Модель процессов** (модель деятельности) отражает организацию вычислений и оперирует понятиями «процессы» и «нити».
- **Модель развёртывания** показывает особенности размещения программных компонентов на конкретном оборудовании.

Конкретное представление логики системы осуществляется посредством диаграмм, каждая из которых имеет графическую часть и текстовую часть в виде спецификации.

Спецификация моделей

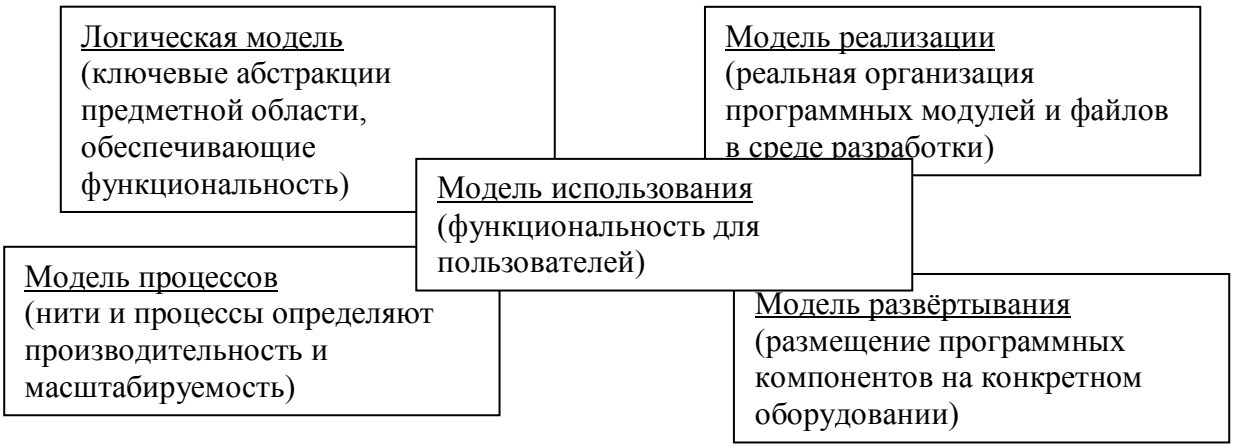


Рис. 3.2 Выводы.

Структура разработки РПЗ к программному проекту.

Содержание.

Введение.

1. Модель вариантов использования ИС управления процессами изготовления *Имя Прибора*.
   1.1 Диаграмма вариантов использования.
   1.2 Диаграмма взаимодействия.
      1.2.1 Диаграмма кооперации
      1.2.2 Диаграмма последовательности.
2. Логическая модель ИС управления процессами изготовления *Имя Прибора*.
      2.1 Диаграмма пакетов.
      2.2 Диаграмма классов.
      2.3 Диаграмма состояний.
      2.4 Диаграмма деятельности.
3. Модель реализации ИС управления процессами изготовления *Имя Прибора*.

3.1 Диаграмма компонентов.

3.1.1 Диаграмма компонентов для моделирования ???

3.1.2

4. Модель развёртывания ИС управления процессами изготовления *Имя Прибора.*

5. Генерация заголовочных файлов.

6. Методическое обеспечение.

6.1 Инструкция администратора.

6.2 Инструкция пользователя.

7. Модель тестирования (методика разработки тест-программ лежит на сервере в разделе «Проекты»).

Наименование и представление заголовочных файлов оформляются в соответствии с документом «Соглашение о разработке ПО».

Замечание.

ДЗ реализуется в указанной структуре в бумажном и PDF вариантах. Документация фирмы реализуется средствами Wiki.

Диаграммы UML.

- **Диаграммы вариантов использования** (use cases) – описывают функции системы для каждого типа пользователей.
- **Диаграммы классов** – делятся на контекстные, описания интерфейсов и реализации. Демонстрируют отношение классов между собой.
- **Диаграммы пакетов –** демонстрируют связи наборов классов, объединённых в пакеты между собой.

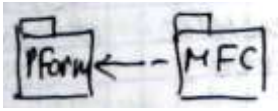


Рис. 3.3

- **Диаграммы деятельности** (activity) представляют собой схему потоков управления для решения некоторой задачи по отдельным действиям, допускают наличие параллельных и альтернативных действий.
- **Диаграммы компонентов** – показывают из каких компонентов состоит ПО и их связи.
- **Диаграммы взаимодействия:**

- **Диаграммы последовательностей действий** (sequence) отображает упорядоченные по времени взаимодействия объектов в процессе выполнения варианта использования.
- **Диаграммы кооперации** (collaboration diagrams) представляют ту же информацию, что и предыдущая диаграмма, но подробнее представляет ответственности классов в целом.

- **Диаграммы состояния** объектов – показывают состояние объекта и условия перехода из одного состояния в другое.
- **Диаграммы размещения –** показывает связь программного и аппаратного обеспечения системы.

Взаимосвязь диаграмм моделей.

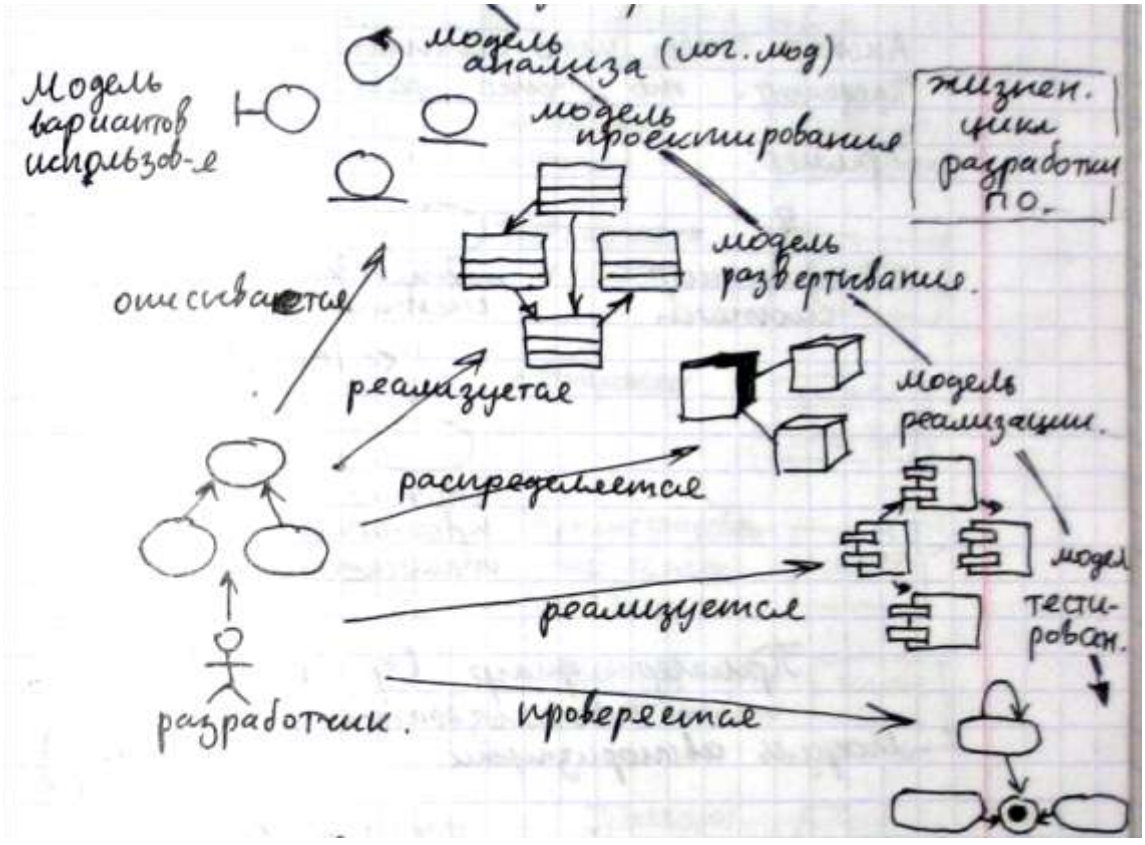


Рис. 3.4

Элементы унифицированной системы обозначений для диаграммы вариантов использования.

Компонент модели	Условное обозначение	Компонент модели	Условное обозначение
Действующее лицо (Actor)		Связь	

Вариант использования или прецедент		Связи «расширение» и «использование»	

Выделяют обычных и абстрактных актеров (например пользователь).

Актёр – тот, кто действует.

Прецедент – то, что делает.

Пример:

Рис.3.5

Пример диаграммы вариантов использования.

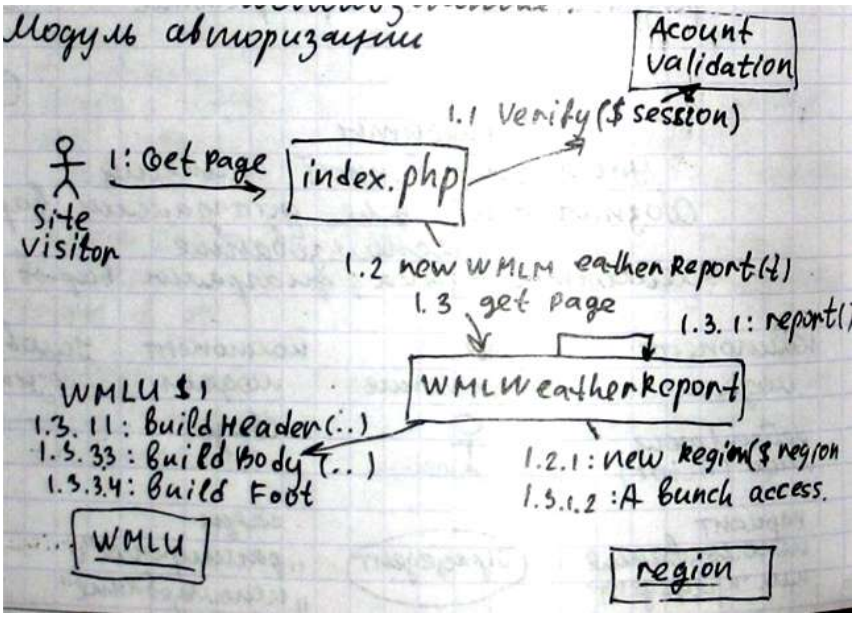
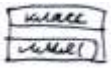
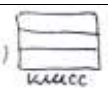
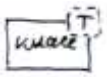
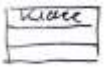
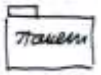
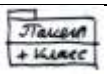
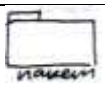
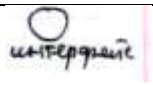
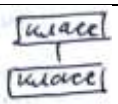


Рис. 3.6 ~~Диаграмма вариантов использования~~. Только это не она)

Элементы унифицированной системы обозначений для диаграммы классов и пакетов

Компонент модели	Условное обозначение	Компонент модели	Условное обозначение
Класс со скрытыми секциями		Абстрактный класс	
Класс с раскрытыми секциями		Абстрактный ?	
Класс (пиктограмма)		Параметризованный класс	
Активный класс		Настроенный класс (шаблон)	
Видимость атрибутов класса	+общий -скрытый #защищённый	Пакет	
Граничный класс		Пакет с раскрытой секцией	
Управляющий класс		Пакет (пиктограмма)	
Класс-сущность		Интерфейс	
Обобщение		Реализация интерфейса классом	
Двунаправленная ассоциация		Реализация интерфейса пакетом	

Граничный класс – определяет взаимодействие с внешними объектами.

Управляющий класс – класс с основной логикой приложения.

Класс-сущность – IDEF1X.

Класс типа интерфейс – представление формы взаимодействия пользователя с программой.

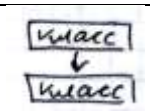
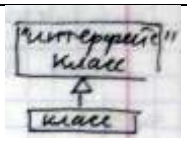
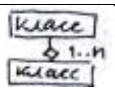
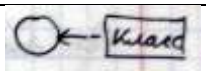
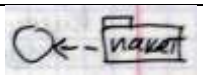
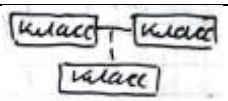
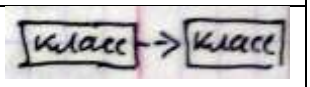
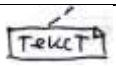
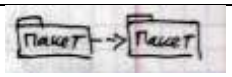
Обобщение используется, когда в класс предок выносится некая общая функциональность.

Двунаправленная ассоциация – взаимное использование классов.

Пример. Смотри диаграмму классов пакет MFC.

Замечание. Желательно диаграмму класс-сущность выделять в отдельную диаграмму.

Виды связей между классами

Однонаправленная ассоциация		Реализация интерфейса (раскрытая)	
Агрегация		Использование интерфейса классом	
Композиция		Использование интерфейса пакетом	
Отношение ассоциации класса		Зависимость классов	
Примечание		Связь пакетов	

Композиция – реализация свойства инкапсуляции.

Агрегация – частный случай композиции, когда отдельные части сгруппированы по ролям.

Задание на дом.

По фирме: к следующему семинару сделать диаграммы вариантов использования, диаграммы пакетов, диаграммы классов.

Лекция №6

28.02.07 г

Диаграмма пакетов.

Это одна из диаграмм логической модели. Находится в Logical View. Представляет собой обобщение тех модулей, которые мы разрабатываем.

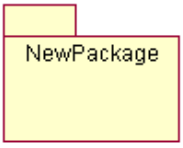


Рис. 5.1 Условное обозначение диаграммы пакета.

Отношения и другое для диаграммы пакетов такие же как в диаграммах классов.

В основном с её помощью описывается архитектура построения информационных систем, чтобы показать модульную структуру.

В нашем случае должны быть модули: авторизации пользователя, функциональный модуль. Вот и все два пакета (пока, в следующем семестре добавится пакет экспертной системы).

Диаграмма взаимодействия (collaboration diagram)

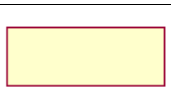
Компонент модели	Условное обозначение	Компонент модели	Условное обозначение
Объект		Фокус управления	
Линия жизни		Разрушение объекта	
Вызов процедур		Асинхронный поток	
Синхронный поток управления		Возврат управления	

Рис. 5.2 Диаграмма взаимодействия.

Диаграмма используется для описания интерфейса.

Если для отслеживания операций над потоками данных диаграммы последовательностей (sequence) недостаточно, то используются диаграммы взаимодействий (collaboration).

Выводы (критерий выбора определённых диаграмм):

1. Диаграмма последовательностей используется если необходимо показать элементы кода и объекты, а также их отношение к временным и межпроцессным взаимодействием.
2. Диаграмма взаимодействия используется, если акцент ставится на указании потоков обработки данных без временной привязки.
3. Диаграмма последовательностей используется, если надо отразить время жизни процессов и их синхронизацию.
4. Диаграмма взаимодействий используется если отражается логика передачи сообщений между небольшим количеством объектов.

Рис. 5.3 Диаграмма кооперации.

Диаграммы взаимодействия и диаграммы последовательностей являются расширением диаграммы вариантов использования или логической модели.

ДЗ.

Разработать для фирмы диаграммы взаимодействия и последовательностей.

Диаграммы деятельности (activity diagram)

Это аналог представления алгоритмов.

Начало		Переход	
Конец		Линейки синхронизации	
Деятельность		Состояние	
Выбор		Составное состояние	

Линейка синхронизации – всё, что есть до них должно быть выполнено, лишь потом будет выполнено то, что после них. Аналог ключей IDEF3.

Очень просто перевести IDEF3 в диаграмму деятельности.

Диаграмма деятельности используется для отражения логических схем, расчётных моделей, т.е. всего, что можно назвать алгоритмом.

Диаграмма компонентов (component diagram).

Физические элементы диаграммы.

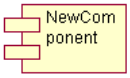
Программный компонент (файл)		База данных	
Текстовый файл (в основном комментарии)		Таблица	
Динамическая библиотека DLL		Узел (компьютер).	

Диаграмма компонентов может состоять из файлов, частей файлов, кусочков кода.

Рис. 5.4 Пример диаграммы компонентов.

Диаграмма размещения.

Показывает структуру и состав аппаратного обеспечения, необходимого для развёртывания создаваемой информационной системы. Набор инструментов достаточно простой.

ДЗ.

Разработать все диаграммы в UML, описывающие модуль АСУ вашей фирмы и разместить на сервере.

Этапы проектирования автоматизированной информационной системы

Основные этапы:

1. Описание структуры предприятия, описание производственных процессов, материальных и информационных потоков, разработка информационной модели. СФ модель – источник разработки. На неё нужно дать ссылку в источниках.
2. Объектно-ориентированное проектирование. Перевод терминов между предметными областями. Работы: разработка вариантов использования (преобразование деятельности каждого рабочего), разработка структуры АСУ (диаграммы пакетов и компонентов), логика работы (диаграмма деятельности, классов), интерфейс пользователя (разрабатывается в Visio на основе диаграмм последовательности и деятельности), определяются конкретные компоненты

системы (диаграммы компонентов), системные требования (диаграммы развёртывания).

Связь моделей и диаграмм проектирования

В основе всего лежит понятие модели, модель состоит из диаграмм, диаграммы описывают объекты и их состояния, объекты связаны связями.

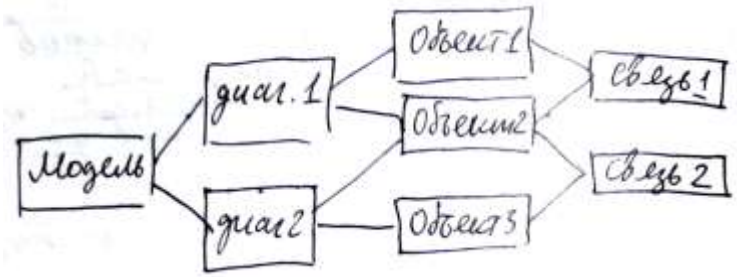


Рис. 5.5 Связь моделей и диаграмм проектирования.

Структура ДЗ.

Базовая объектно-ориентированная модель создаваемого АСУ.

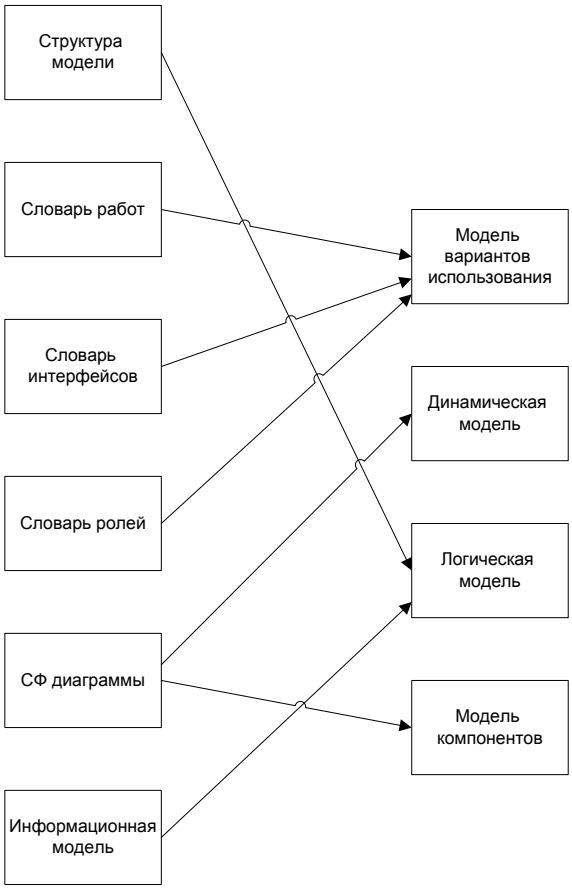


Рис. 5.6 Базовая объектно-ориентированная модель АСУ предприятия.

Информационная модель – набор скриптов и заголовочных файлов.

Перечень вопросов, подлежащих разработке.

1. Разработка модели вариантов использования.
2. Разработка логической модели.
3. Разработка динамической модели.
4. Разработка модели компонентов.
5. Разработка модели развёртывания.
6. Формирование шаблонов классов и методов разработанного ПО.
7. Методологическое обеспечение.
8. Обеспечение автоматической генерации.

Замечания по ДЗ.

Диаграмма вариантов использования.

Сюда входят диаграмма вариантов использования, разработка диаграммы последовательности действий либо диаграммы деятельности. Для каждой роли нужна своя диаграмма. На каждую диаграмму составляется спецификация.

Разработка логической модели.

Диаграмма пакетов. Надо сделать её более конкретной.

Диаграмма классов. В двух вариантах. Первый показывает последовательность взаимодействий между классами. Вторая диаграмма показывает структуру классов.

Модель компонентов.

Модель развёртывания одна на всех. Надо получить конфигурацию сервера.

Лекция №7, 8

07.03.07 г

Автоматизированная информационная система

Это структура, обеспечивающую принятие конкретных решений при обработке данных.

Информационный перерабатывающий завод.

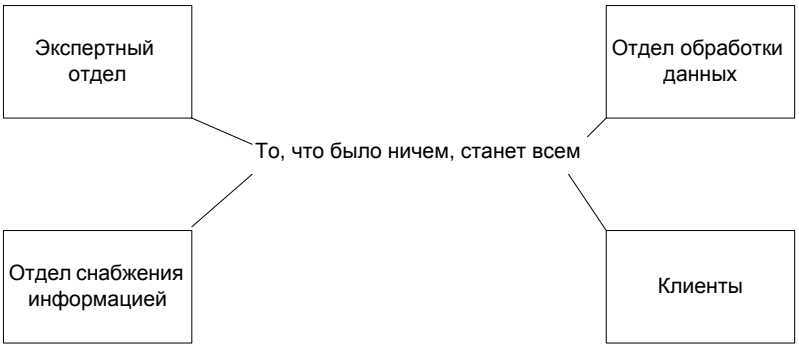


Рис. 5.1 Автоматизированная информационная система.

Знания – это некое понятие, смысл которого раскрывается через набор его специфических характеристик данных.

Данные – это конкретные значения свойств (характеристик) знаний.

Парадигмы решения задач.

1. Традиционная. Данные+Алгоритм=Программа.
2. Знания+Стратегия=Решение проблемы. Применяется в экспертных системах.

Основные определения.

БД – электронные хранилища информации, доступ к которым осуществляется с помощью одного или несколько компьютеров.

СУБД – программное обеспечение для создания, наполнения, обновления и удаления БД.

Реляционная модель знаний: организует и представляет данные в виде таблиц или реляций.

Реляционная БД – БД, построенная на реляционной модели.

Реляция – двумерная таблица, содержащая строки и столбцы данных.

Степень реляции – количество атрибутов реляции.

Кортежи – строки реляции, соответствуют объекту, конкретному событию и явлению.

Атрибуты – столбцы, характеризующие признаки, параметры объекта, события, явления.

Пустое значение (NULL) – значение, приписываемое атрибуту в кортеже, если атрибут неприменим или его значение неизвестно. Создание числового поля без NOT NULL в таблице – грубейшая ошибка.

Ключ – любой набор атрибутов, однозначно определяющий каждый кортеж реляционной таблицы.

Родительская реляция – таблица, поля которой входят в другую таблицу.

Дочерняя реляция – таблица, поля которой используют информацию из полей другой таблицы. Мы не можем вставить данные, пока они не появятся в родительской. Здесь часто бывает логическая ошибка!

Представления (VIEW) – виртуальные таблицы, атрибуты которых составляются на основе запросов к другим таблицам. На VIEW делается публичный синоним, и через него осуществляется доступ других пользователей к информации.

Отношение один к одному – одна запись в таблице родителя соответствует одной записи к дочерним.

Отношение один ко многим - одна запись в таблице родителя соответствует нескольким записям к дочерним.

Хранимая процедура – программа, которая выполняется внутри СУБД и может предпринимать сложные действия на основе информации, задаваемой пользователем.

В дз базовая логика приложений будет на PL/SQL.

Операция соединения (join) – процесс, позволяющий объединить данные из двух таблиц посредством сопоставления содержимого двух аналогичных столбцов.

Индекс – специальный механизм БД, обеспечивающий лучший поиск информации в данной таблице. Физически представляет собой специальную таблицу с полями-индексами, которые заранее отсортированы либо по возрастанию, либо по убыванию.

Триггер – специальная наименованная программа БД, которая обрабатывает различные действия.

Язык структурированных запросов SQL

SQL основан на реляционной алгебре и является языком манипулирования данными.

SQL позволяет описывать условия поиска информации без того, чтобы задавать последовательность действий, нужных для получения ответа на запрос.

SQL – стандартное средство доступа к реляционным БД.

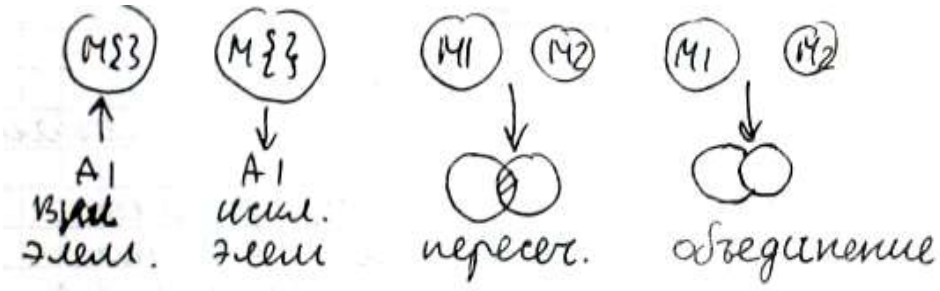


Рис. 5.2 Основа SQL – операции над множествами.

Буковки М – множества, построенные по реляционной схеме, т.е. таблицы.

В основе базовых операций лежит реляционная модель. Реляционная модель, основана на логических отношениях данных. Это набор двумерных таблиц определяемый четырьмя понятиями: таблица, столбец, строка, поле.

Оперирование с реляционной моделью данных основано на понятиях раздела математики, называемого реляционной алгеброй. В основе её два понятия: есть набор объектов (отношения), есть набор операторов, который устанавливает связи между отношениями.

Эти понятия включают: выбор объектов, называют отношениями, выбор операторов.

Отличия традиционных (файловых) и реляционных БД.

Операция	**Традиционная БД**	**Реляционная БД**
Разработка приложений	Необходимо определить, какая информация требуется различным приложениям и создать ряд общих файлов	Необходимо определить виды хранимых данных и взаимосвязи между ними
Реализация приложений	Поступающие данные записываются в основные файлы, в каждую информационную ячейку	Различные данные записываются в таблицу данных, соответствующую этим видам
Модификация приложений	Требуется пересмотр структуры БД	Достаточно найти и модифицировать таблицу

Типы данных обрабатываемых СУБД Oracle

Основные шаги по разработке реляционной базы данных были рассмотрены при разработке модели IDEF1X.

Типы данных делятся на 2 класса: логические (общий для всех СУБД; number, string, date), физические (СУБД ориентированные, см. далее).

CHAR (size) и CHARACTER– символьная строка до 255 символов.

DATE – правильные даты.

LONG – символьные данные до 2 Гб.

LONGRAW – двоичные данные переменные до 2 Гб.

MISLABEL – используется в trusted Oracle.

NUMBER(p,s) – число, имеющее p значащих цифр и масштаб S. P=1..38, S=-84..127.

В состав языка SQL входит операторный язык определения данных DDL (Data Defenition Language) и операторы языка обработки данных DML (Data Manipulation Language).

Рис. 5.3 Очень важная картинка.

Основные компоненты PL/SQL

Операторы DDL:

CREATE – создание объекта БД.

ALTER – изменение элемента БД.

DROP – удаление объекта БД.

AUDIT – команды протоколирования действий объекта схемы.

“USER” – команды управления пользователями.

“CONTROL” – команды (служебные и вспомогательные) управления.

Специальный тип данных – **псевдостолбцы**. Это элементы, которые позволяют получать определённое состояние атрибутов таблицы.

CURRVAL – текущее значение объекта. Например, можно получить значение счётчика (sequence): SELECT <имя сиквенса>.CURRVAL FROM DUAL;

NEXTVAL – следующее значение в текущем сеансе.

LEVEL – 1 для корня дерева, 2 для узлов второго уровня и т.д.

ROWID – значение, которое идентифицирует строку таблицы уникальным атрибутом.

Требования к именам объектов БД

Длина 1..30 байт.

Длина имени БД не более 8 байт.

Не может содержать кавычки.

Не могут совпадать с именами других объектов.

Имена начинаются с A-Z, могут содержать A-Z, 0-9.

Не могут дублировать зарезервированные слова SQL.

Различие в прописных и строчных буквах учитывается в двойных кавычках.

Операции и их приоритеты

Арифметические операции	Символьные операции	Логические операции	Операции сравнения
+- (один операнд)	|| (склеивание строковых переменных)	NOT	=
*/		AND	;=, ^=, _=, <>
+- (два операнда)		OR	:=, >=, <=, <
			IN
			NOT IN
			ANY, SOME
			ALL

В SQL запросах и при наименовании объектов необходимо использовать собственные конструкции, отличающиеся от зарезервированных слов SQL.

Вставить таблицу из методы! Будет в приложении

Документирование скриптов

Все объекты БД и обращения к ним создаются посредством SQL скриптов. Это файлы типа: имя.sql. Для их запуска в SQL+ используется команда @.

Все скрипты SQL должны быть документированы и оформлены на основе солгашения о разработке ПО.

Комментарий может стоять в любом месте - /*…*/

Оформленный по всем правилам набор скриптов, выполняющихся последовательно, называется инсталляционным комплектом.

Выдержки из соглашения о разработке

Создание таблиц.

1. Имена sql-программ должны включать следующие атрибуты: **NNXXXX.SQL**, где **NN –** порядковый номер sql-файла при запуске; **XXXX –** имя подсистемы разработчика (3-4 символа). Например: 02exam.sql
2. Необходимо предусмотреть возможность запуска sql-файлов в назначенной последовательности несколько раз.
3. При выполнении sql-программы должен включаться вывод результатов в файл. Для этого необходимо добавить: **SPOOL xxxxxxxx.lst** в начало файла, где **xxxxxxxx** имя файла sql-программы; **SPOOL OFF** в конец файла.
4. Необходимо, чтобы в sql-программе при выполнении действий с объектом базы данных отражалось имя этого объекта, например: **PROMPT создаётся TABLE1; CREATE TABLE1…**
5. Этого нет в соглашении о разработке, но надо делать так. Перед командой создания таблицы должна ставиться команда удаления таблицы **DROP TABLE.** В соглашении записано следующее: «Перед созданием нового объекта (база данных, таблица, хранимая процедура, и.т.п) необходимо выполнять проверку на существование объекта с таким же именем». Вероятно так и надо будет поступать, когда будем уметь делать такую проверку.
6. Имена таблиц модуля должны начинаться с префикса (2-4 буквы), за которым следует знак подчеркивания.
7. Все ограничения кроме NULL (NOT NULL, CONSTRAINT, PRIMARY KEY, INDEX) нельзя создавать при создании таблицы (т.е. их надо создать потом), т.к. все ограничения должны иметь собственное осмысленное имя с префиксом.

Создание индексов.

1. Следует включить в команду **CREATE INDEX** опцию **TABLESPACE** и предусмотреть возможность интерактивной подстановки имени табличного пространства администратору БД в момент выполнения sql-программы: **CREATE INDEX i_table1 ON table1 ...TABLESPACE &&tablespace;**
2. Имена индексов должны начинаться с символов «I_», далее имя таблицы (или его сокращение), после чего следует смысловая часть. Например: для первичных ключей рекомендуется применять аббревиатуру «_PK», для FOREIGN KEY – имена столбцов или сокращений, на которые осуществляется ссылка.

Создание первичных ключей.

1. Первичный ключ должен создаваться с помощью CONSTRAINT, но не с префиксом C_ , а с префиксом I_ (также как у индекса).

Создание элементов ссылочной целостности.

1. Имена ограничений (CONSTRAINT) должны создаваться с префиксом C_, далее идёт смысловая часть.

Визуальное описание SQL скрипта; понятие синтаксической диаграммы.

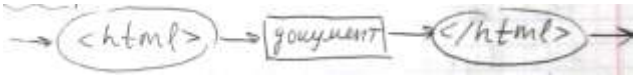
Синтаксическая диаграмма – это графическая диаграмма, это графическое представление правил построения конструкций языка в наглядной форме.

Символы алфавита изображаются блоками в овальных рамках.

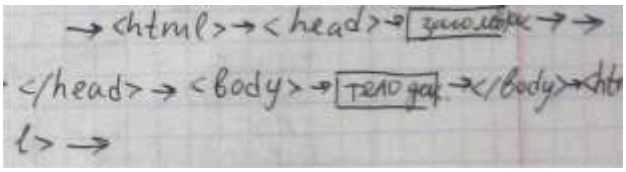
Названия конструкций – в прямоугольниках.

Правила построения конструкций в виде линий со стрелками, также выделяют осевую линию. На осевой линии расположены обязательные элементы, все дополнительные элементы располагаются на выносных линиях.

Пример.



Пример разработки синтаксической диаграммы:



Обязательным условием составления синтаксической диаграммы является соблюдение правил структурного программирования.

Основные требования структурного подхода:

- SQL команды могут задаваться на одной или нескольких строках;
- Отдельные функциональные компоненты команды (предложения) обычно задаются на разных строках;
- Допускается использовать табуляцию;
- Отдельные слова в команде не могут быть разделены;
- SQL команда.

Примеры.

1. SELECT * FROM EMP;

2. SELECT

*

FROM EMP;

3 SELECT *

FROM EMP;

Второй вариант лучший. Чем сложнее скрипт, тем более структурирован должен быть запрос.

Операторы группы CREATE

CREATE DATABASE – создание БД.

CREATE USER – создание пользователя.

CREATE TABLE – создание таблицы.

CREATE SYNONYM – создание некого глобального представления объекта с тем же именем для доступа к нему извне (из других схем).

CREATE INDEX – создание индексов.

CREATE VIEW – создание представлений.

CREATE SEQUENCE – создание последовательностей.

Создание базы данных. CREATE DATABASE

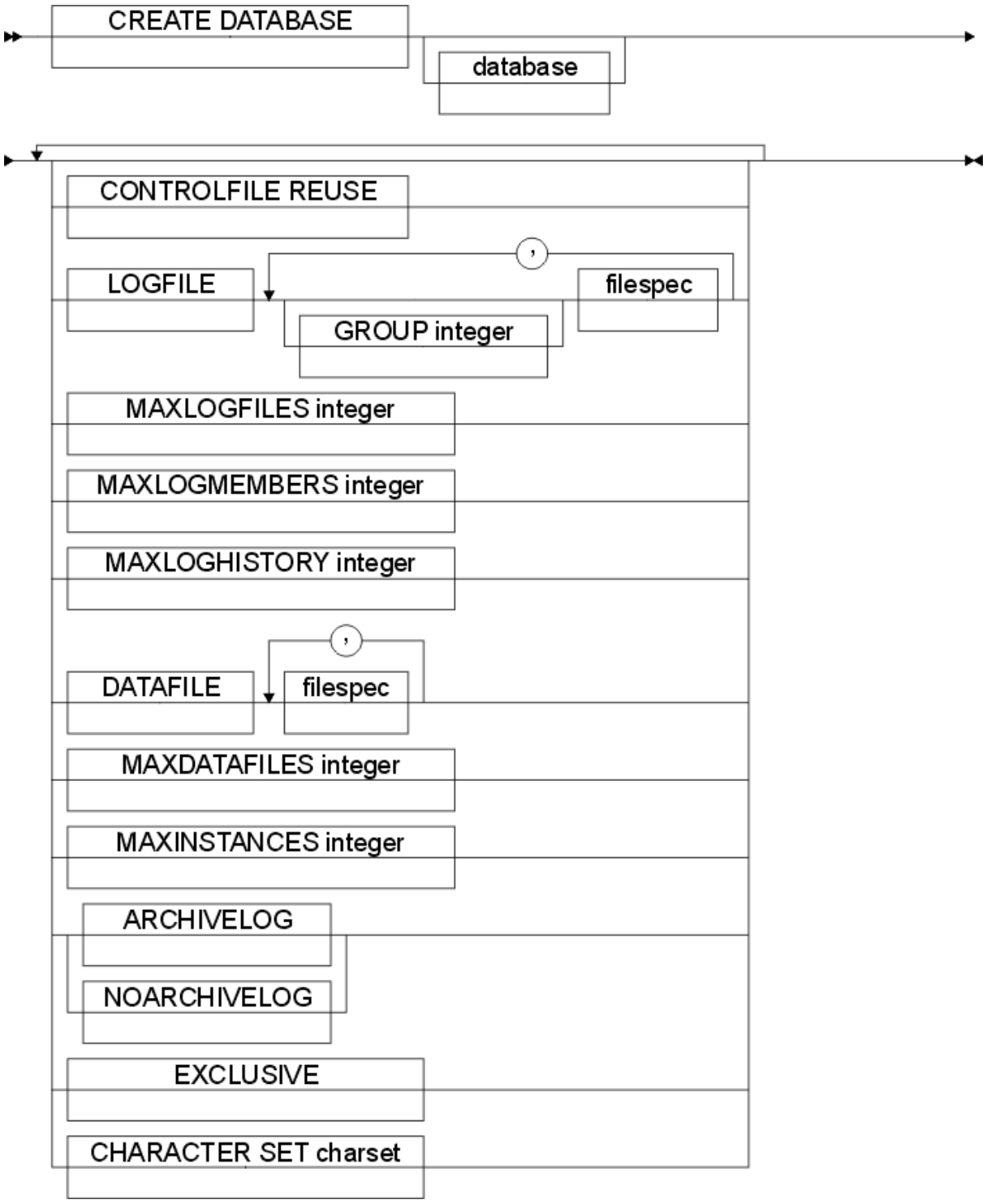


Рис. 5.4 CREATE DATABASE.

Пример:

PROMPT Скрипт создания БД CLINICS

SPOOL clinic.log

CONNECT internal

STARTUP NOMOUNT pfile=/oracle/dbs/initclinic.ora

PROMPT Создаём БД с именем CLINICS

```
CREATE DATABASE “clinics”
 MAXINSTANCES 1
 MAXLOGFILES 10
 CHARACTERSET “RUSPC866”
 DATAFILE
 ‘/oracle/db/systemIO.dbf’
 LOGFILE
 ‘/oracle/db/log01.dbf’
 ‘/oracle/db/log02.dbf’
DISCONNECT
SPOOL OFF
```

Лекция №9

14.03.07 г

Базу данных можно создать с помощью DataBase Assistant.

Создание пользователя. CREATE USER

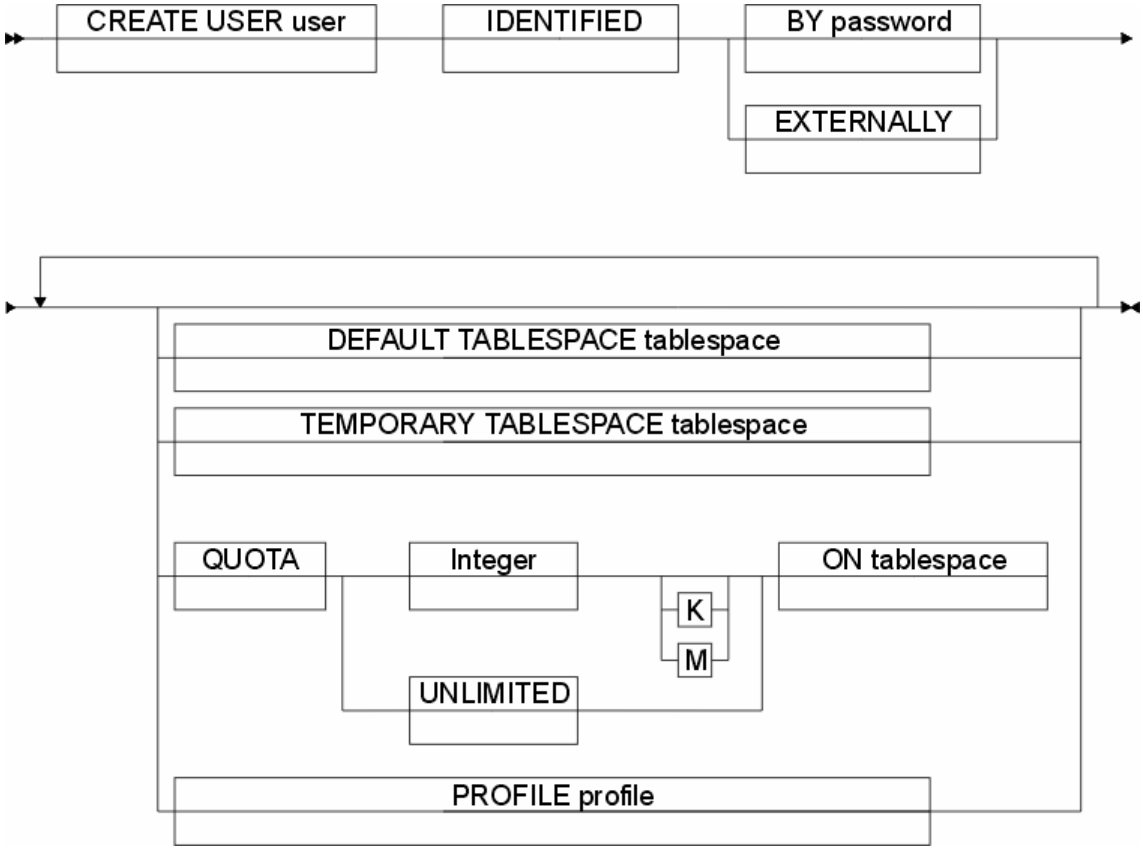


Рис. 9.1 CREATE USER

PROMPT создание пользователя test

```
CREATE USER “test”
PROFILE “DEFAULT”
IDENTIFIED BY “123456”
DEFAULT TABLESPACE “USERS”
        QUOTA UNLIMITED
        ON CWMLITE

        QUOTA UNLIMITED
        ON DRSYS
        QUOTA UNLIMITED
        ON EXAMPLE
        QUOTA UNLIMITED
        ON INDEX
        QUOTA UNLIMITED
        ON SYSTEM
        QUOTA UNLIMITED
        ON TEMP
        QUOTA UNLIMITED
        ON TOOLS
        QUOTA UNLIMITED
        ON USERS;
ACCOUNT UNLOCK;
GRANT CONNECT TO “test”;
```

(Дома надо сделать ещё и админа с правами DBA: GRANT DBA TO “test”)

ДЗ. Создать пользователя с правами DBA, создать пользователя с правами CONNECT, научиться входить под пользователями SYS и SYSTEM.

Создание последовательности.

Это отдельный объект БД, который хранит текущее значение счётчика. До выполнения INSERT нужно поймать её последовательностью и сгенерить номер очередной.

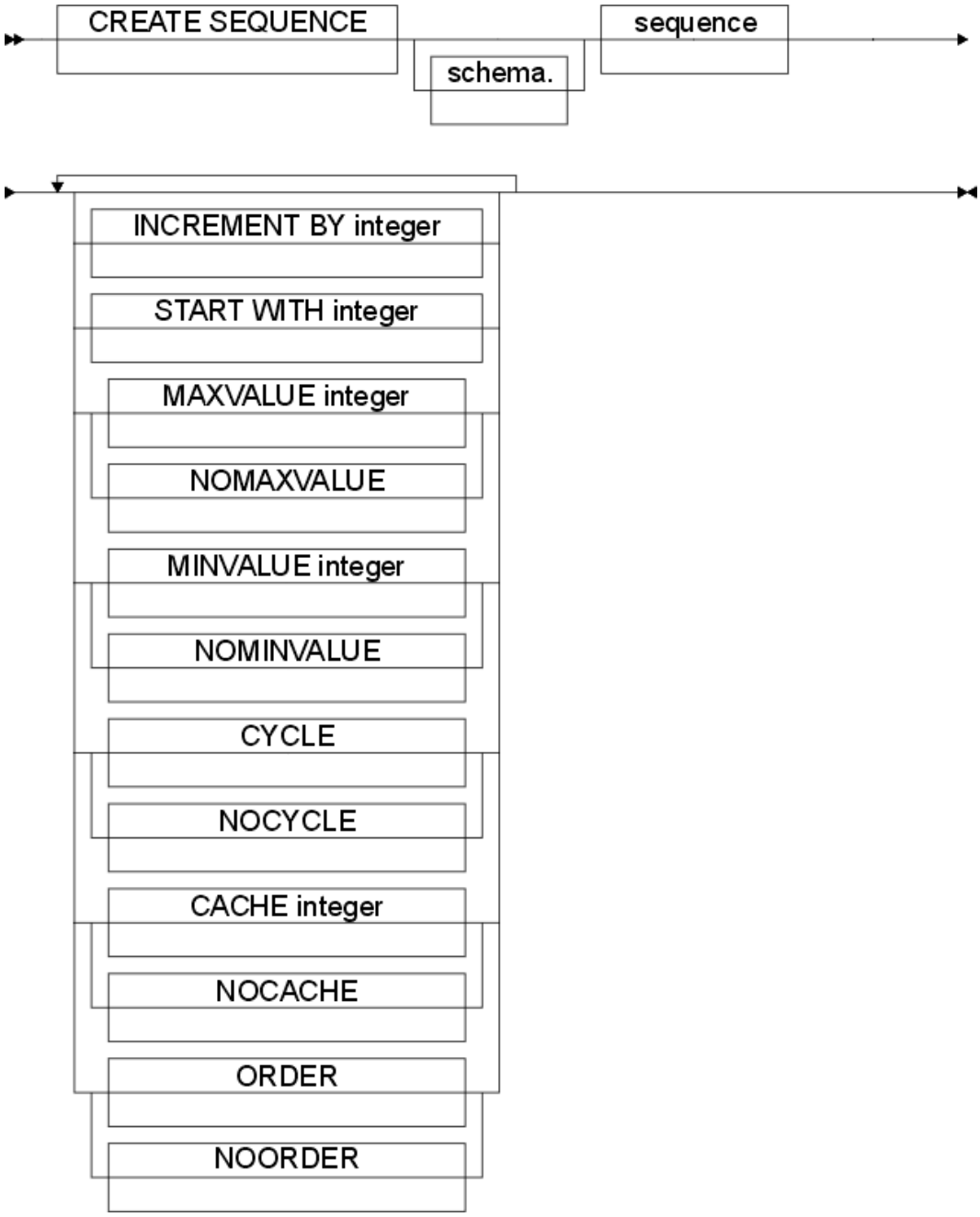


Рис. 9.2 CREATE SEQUENCE.

Каждый сиквенс имеет префикс S_, далее имя поля. Хотя в соглашении вроде имя таблицы.

Пример.

PROMPT создаём сиквенс для автоматической генерации номера задачи

CREATE SEQUENCE s_prj_nnn

START WITH 1

***

PROMPT получаем номер текущей задачи

SELECT s_prj_nnn .NEXTVAL FROM dual;

Можно сделать сиквенс как счётчик номера задач. Директора фирм должны сделать сиквенс, который ведёт номер задачи.

Для каждой таблицы мы сделаем минимум один сиквенс.

Создание таблиц.

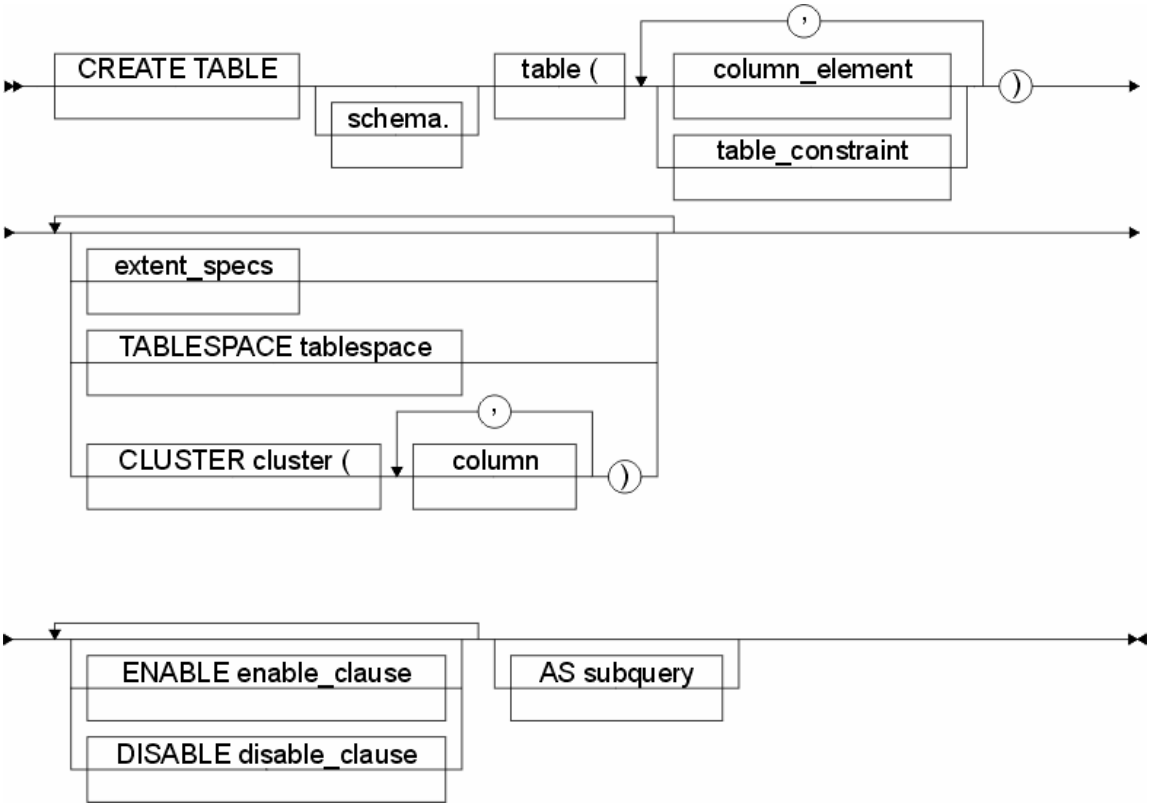


Рис. 9.3 CREATE TABLE.

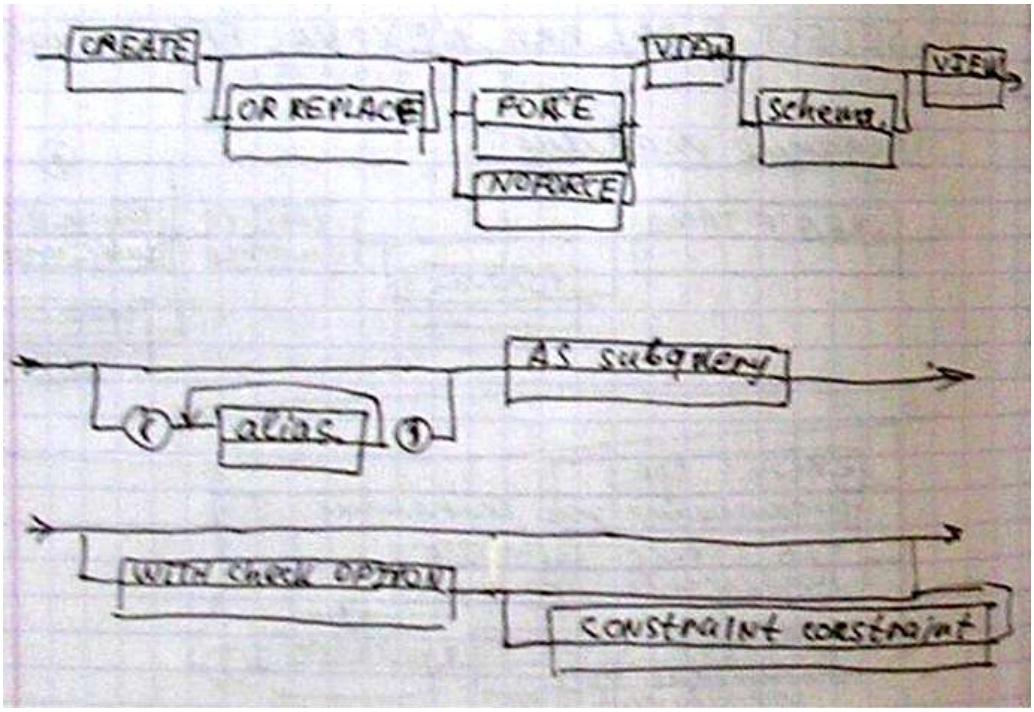
Пример.

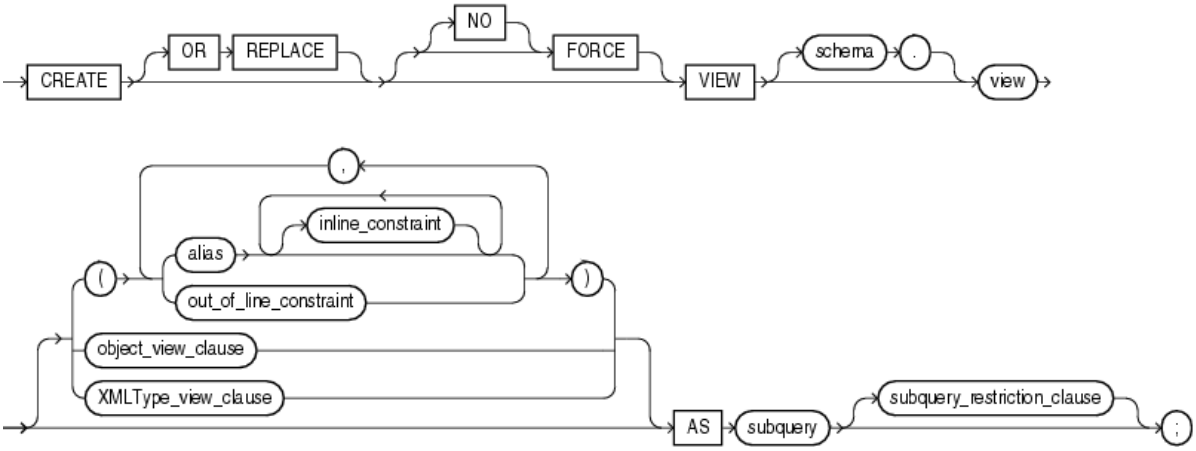
```
PROMPT создание таблицы doctors
CREATE TABLE doctors(
dc_nnn          NUMBER(12,0)
,dc_dc_nnn      NUMBER(12,0)
,dc_name        VARCHAR2(255)
…
) TABLESPACE USERS;
```

Сначала создаётся тело таблицы, а потом добавляются все ограничения (NOT NULL, CONSTRAINT, PRIMARY KEY).

Создание представлений.

Представления (VIEW) – это виртуальная таблица, в которую нельзя вставлять данные. В ней под SELECT-запросами понимается SUBQUERY.





subquery_restriction_clause::=

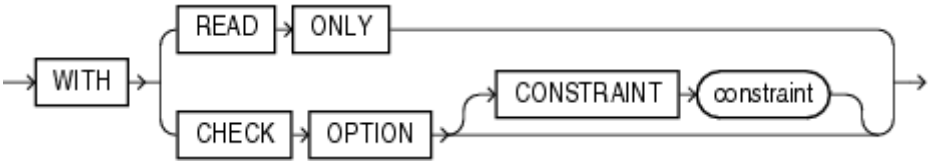


Рис. 9.4 CREATE VIEW.

Пример. См. oracle.iu4.bmstu.ru. Вставить самому.

```
create view EMP DATA as
select EMPNO, ENAME, JOB, MGR, SAL, DEPTNO
from EMP
where SAL BETWEEN 1000 AND 2000
and MGR IN (select DISTINCT EMPNO FROM EMP)
```

and DEPTNO IN (select DEPTNO FROM DEPT)

whith check option;

Общие требования к разработке БД

К следующему разу подготовить расширенное ТЗ на ДЗ. Пример ТЗ на сайте, почти всё кроме раздела «описание функциональности», можно скопировать.

Все требования к разработке БД изложены в документации на сайте:

1. Требования к ТЗ.
2. Требования соглашения по разработке ПО.
3. Требования к методике тестирования.

Требования по функциональности модуля должны быть разработаны самостоятельно на основе материалов информационного аудита.

Пример оформления исходного кода скрипта: по-моему, не очень удачный.

Лекция №10,11

21.03.07 г

Задание ограничений на сущности и атрибуты

Все ограничения кроме NULL рекомендуется создавать при помощи ALTER.

Типы ограничений:

1. **NULL/NOT NULL.** Число + NULL=NULL. Любой INSERT, не вставляющий ничего в поле NOT NULL закончится ошибкой.
2. **UNIQUE**. Все записи в данном атрибуте должны быть уникальны.
3. **PRIMARY KEY.**
4. **FOREIGN KEY**
5. **CHECK**. Ограничение на изменение.

Уровни, на которых накладываются ограничения:

1. Ограничения на отдельные или объединённые атрибуты при создании таблицы.
2. Ограничение на все элементы таблицы после её создания. **ТАК РЕКОМЕНДУЕТСЯ.**

Все ограничения фиксируются в словаре данных. В случае незадания явно имени ограничений вручную ему присвается имя, генерируемое автоматически SYS_.... Такие

имена – **ошибки**. Все ограничения должны иметь собственное уникальное заданное разработчиком имя с учётом префиксного наименования.

Ограничения целостности

Их два: PRIMARY KEY, FOREIGN KEY.

PRIMARY KEY включает в себя ограничений: NOT NULL, UNIQUE, автоматом создаёт уникальный индекс. Предназначение – однозначная идентификация строк таблицы. Синтаксис:

CONSTRAINT имя ограничения PRIMARY KEY (столбец, столбец,…)

FOREIGN KEY в основном используется для поддержки целостности на уровне связи реляционных таблиц. Используется как правило вместе с первичным ключом. Синтаксис:

CONSTRAINT имя ограничения FOREIGN KEY (столбец, столбец,…) REFERENCES таблица (столбец, столбец, …)

В результате установки ограничений строка основной таблицы не может быть удалена, пока не будут удалены записи из дочерних таблиц. Для совместного удаления используется конструкция ON DELETE CASCADE CONSTRAINT.

CHECK – ограничение на фильтрацию значений, вставляемых в определённые строки таблиц. Ограничения аналогичны ограничениям WHERE в SELECT запросах. Синтаксис:

CONSTRAINT имя ограничения CHECK условие

В CHECK запрещено использовать подзапросы и обращение к псевдостолбцам.

Пример.

CREATE TABLE DEPT (имя столбца)

AS SELECT команда;

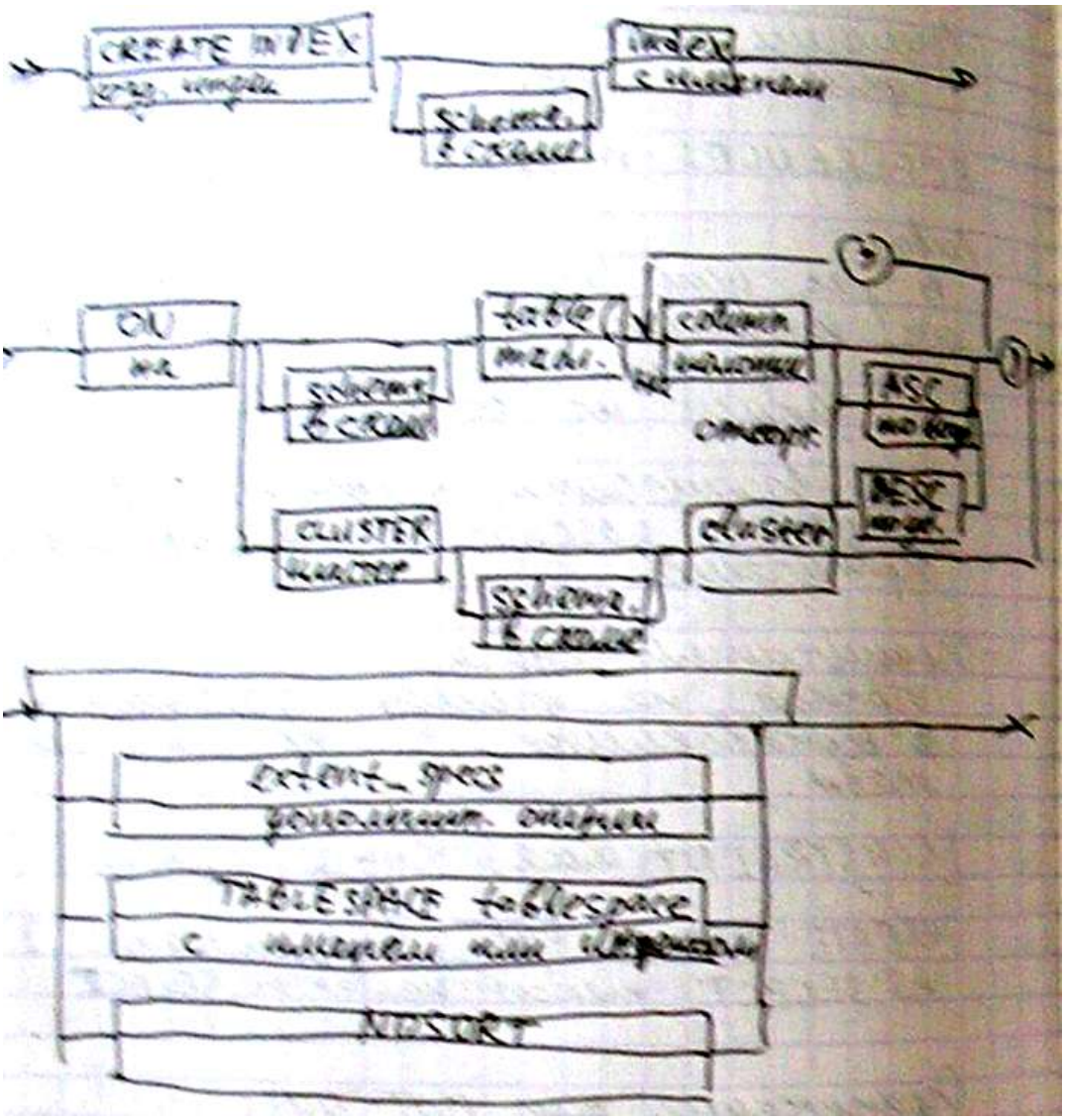
Ограничение **NOT NULL.**

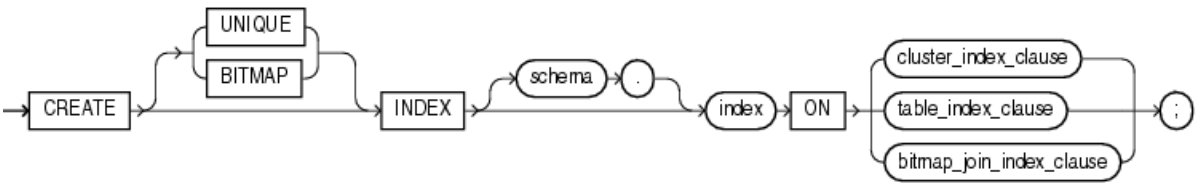
Данное поле не может быть пустым.

Ограничение **UNIQUE.**

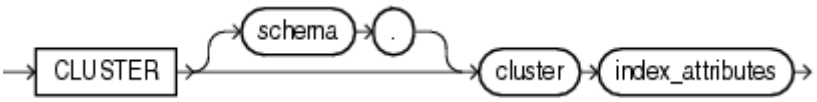
Определяет требования к уникальности столбца либо комбинации столбцов.

Создание индексов. CREATE INDEX.

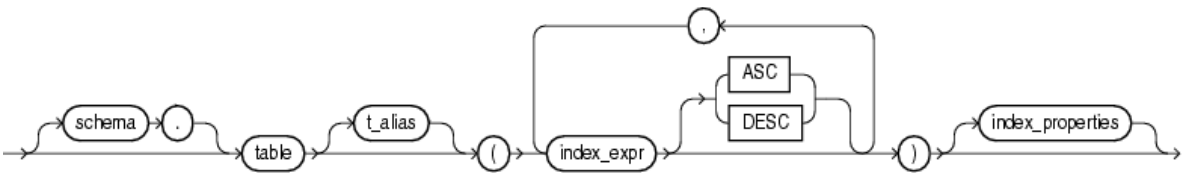




cluster_index_clause ::=



table_index_clause ::=



bitmap_join_index_clause ::=

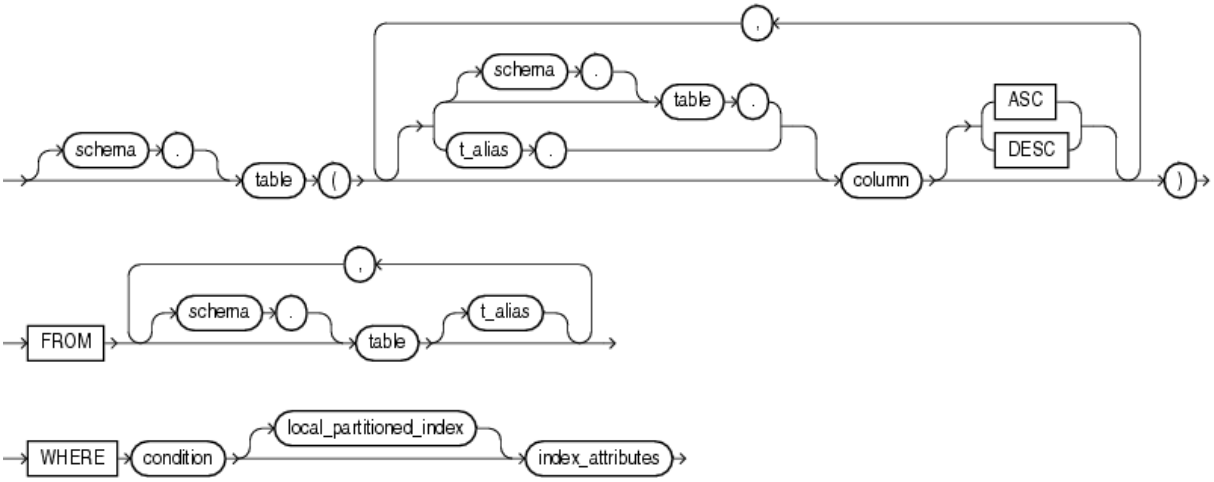


Рис. 10.1 CREATE INDEX.

При создании индекса обязательно указание параметра TABLESPACE, причём отличного от USERS. Т.е. создавать индексы надо в отдельном табличном пространстве. Надо их создавать в табличном пространстве INDEX. Но вообще-то можно в любом. Нам надо писать EXAMPLE.

Пример.

PROMPT удаляем и создаём индекс на таблицу doctors.

DROP INDEX i_doc_nnn;

CREATE UNIQUE INDEX i_doc_nnn

ON doctors;

Группа операторов ALTER. Внесение изменений в таблицы и атрибуты

Изменение таблицы. ALTER TABLE.

Используется для задания всех типов ограничений на таблицы. Основные команды:

1. ADD – добавить в таблицу новый столбец. Пример: ALTER TABLE EMP ADD (first_name char (5)).
2. MODIFY – для изменения столбца таблицы. Пример: ALTER TABLE имя таблицы MODIFY (столбец тип). Таким образом надо создавать ограничения NOT NULL.
3. DROP – удаление ограничений либо свойств таблицы. Пример: ALTER TABLE имя таблицы DROP CONSTRAINT имя ограничения;

4. Свойство ENABLE/DISABLE – позволяет на время отключать ограничений. Может быть очень удобно. Пример: ALTER TABLE dept DISABLE CONSTRAINT c_dept prim CASCADE; - удаление ограничений на таблицу prim из dept.

Ограничение видимости объекта. SYNONYM.

Синонимы нужны для доступа к таблицам другого пользователя. Бывают обычные и публичные.

Для обращения к таблице другого пользователя необходимо к имени таблицы добавить слева в качестве префикса имя её хозяина, отделённое точкой.

Как альтернатива – можно создать синоним на таблицу или представление.

CREATE PUBLIC SYNONYM FOR хозяин;

Требование : имя синонима должно совпадать с именем объекта.

Операторы манипулирования данными. DML

INSERT, SELECT, DELETE, UPDATE.

INSERT – построчно вставляет данные в таблицу.

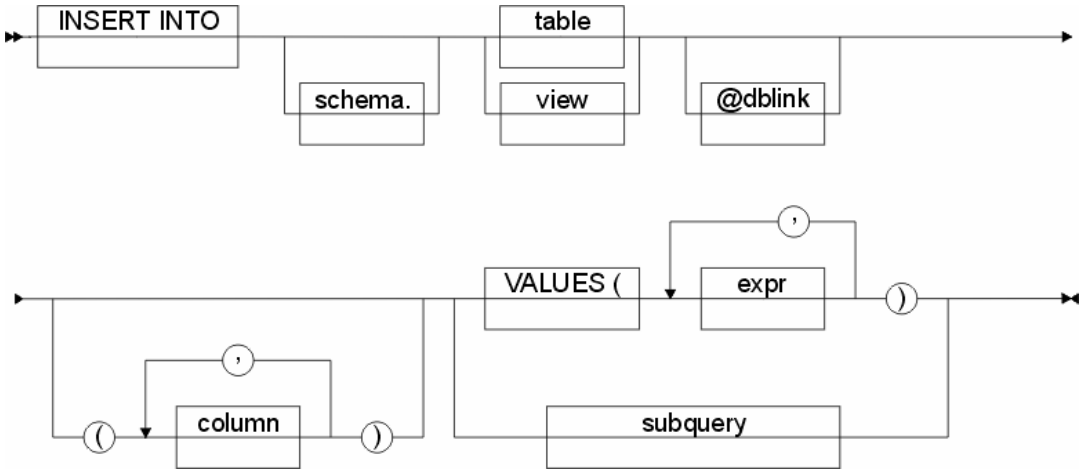


Рис. 10.2 INSERT.

expr – любое выражение или синтаксическое описание из синтаксической диаграммы. Синтаксическая диаграмма будет на следующей лекции.

DELETE – удаляет строки из таблицы.

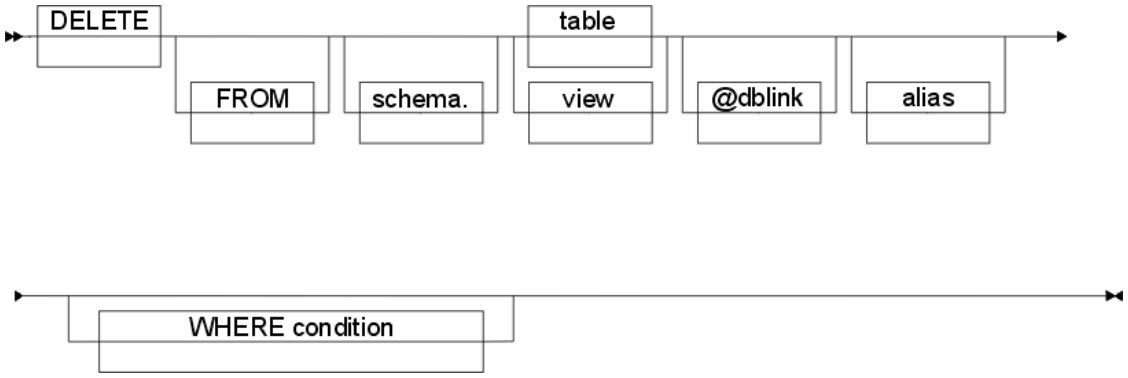


Рис. 10.3 DELETE.

UPDATE.

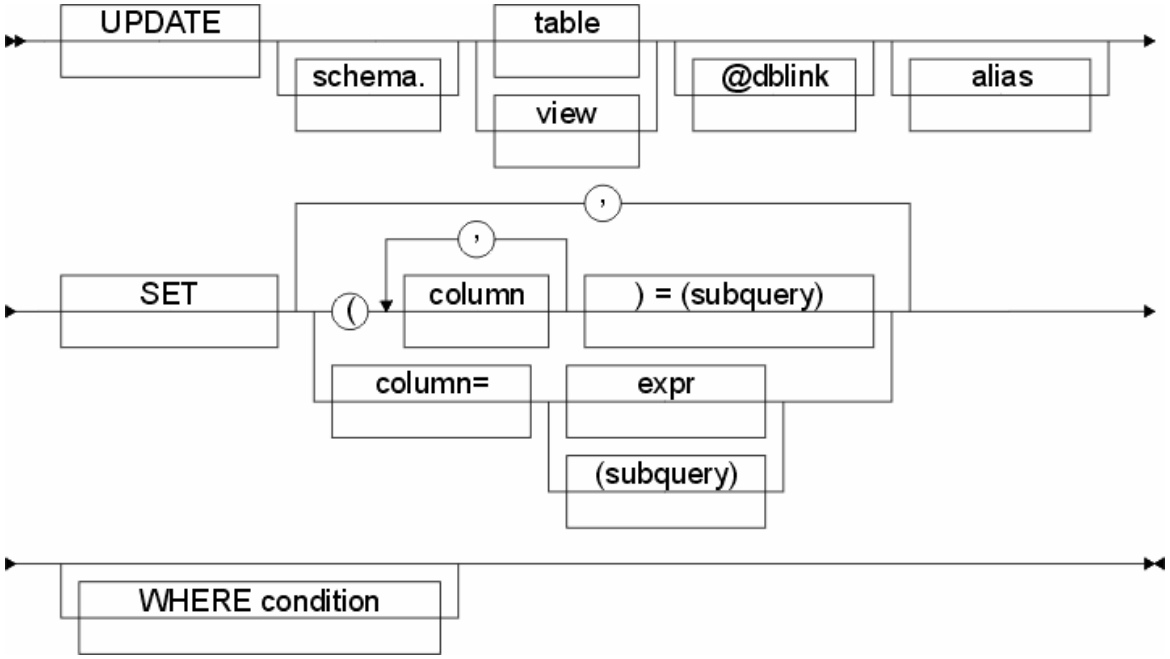


Рис. 10.4 UPDATE.

Лекция №12

28.03.07 г

Expr и conditions – расширения синтаксических диаграмм.

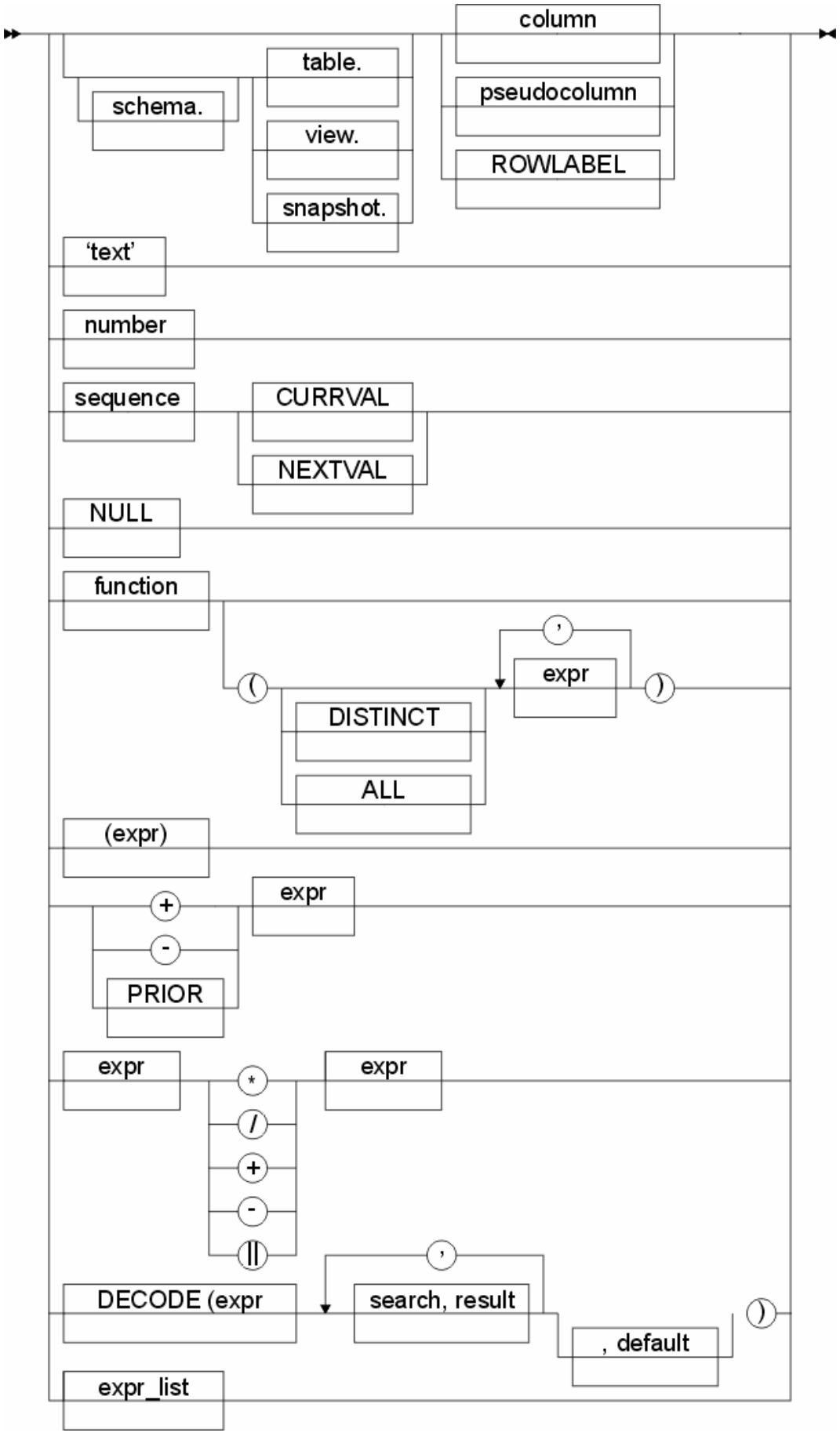

column
table.
schema.
view.
pseudocolumn
ROWLABEL
snapshot.
'text'
number
sequence
CURRVAL
NEXTVAL
NULL
function
,
expr
(
)
DISTINCT
ALL
(expr)
expr
+
-
PRIOR
expr
expr
*
/
+
-
||
,
DECODE (expr
search, result
, default
)
expr_list

Рис. 12.1 EXPR.

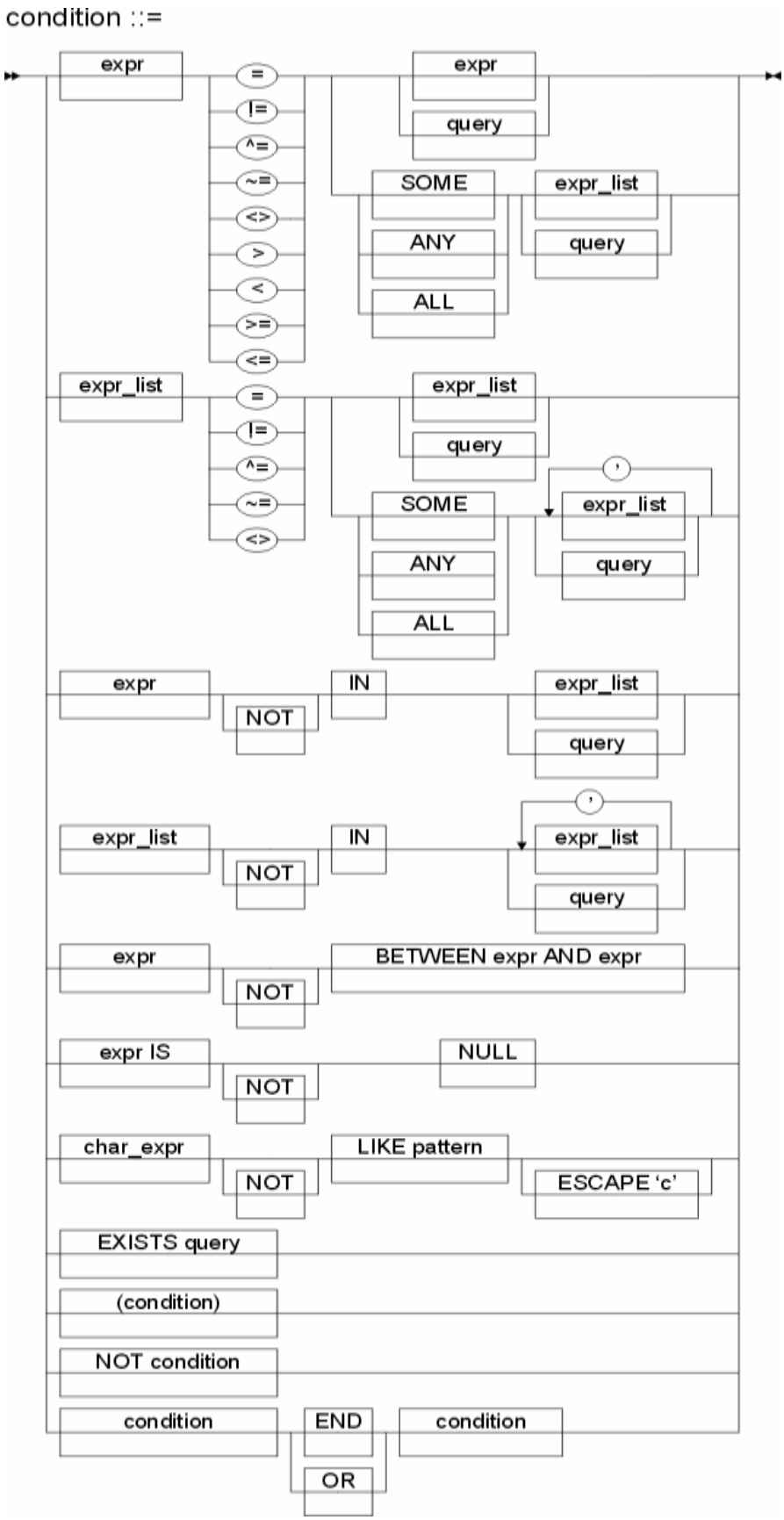


Рис. 12.2 CONDITIONS.

… - возвращает TRUE, если указанная операция сравнения выполняется хотя бы для одного из элементов списка, либо множества возвращаемого запросом.

Формирование запросов к базе данных. SELECT

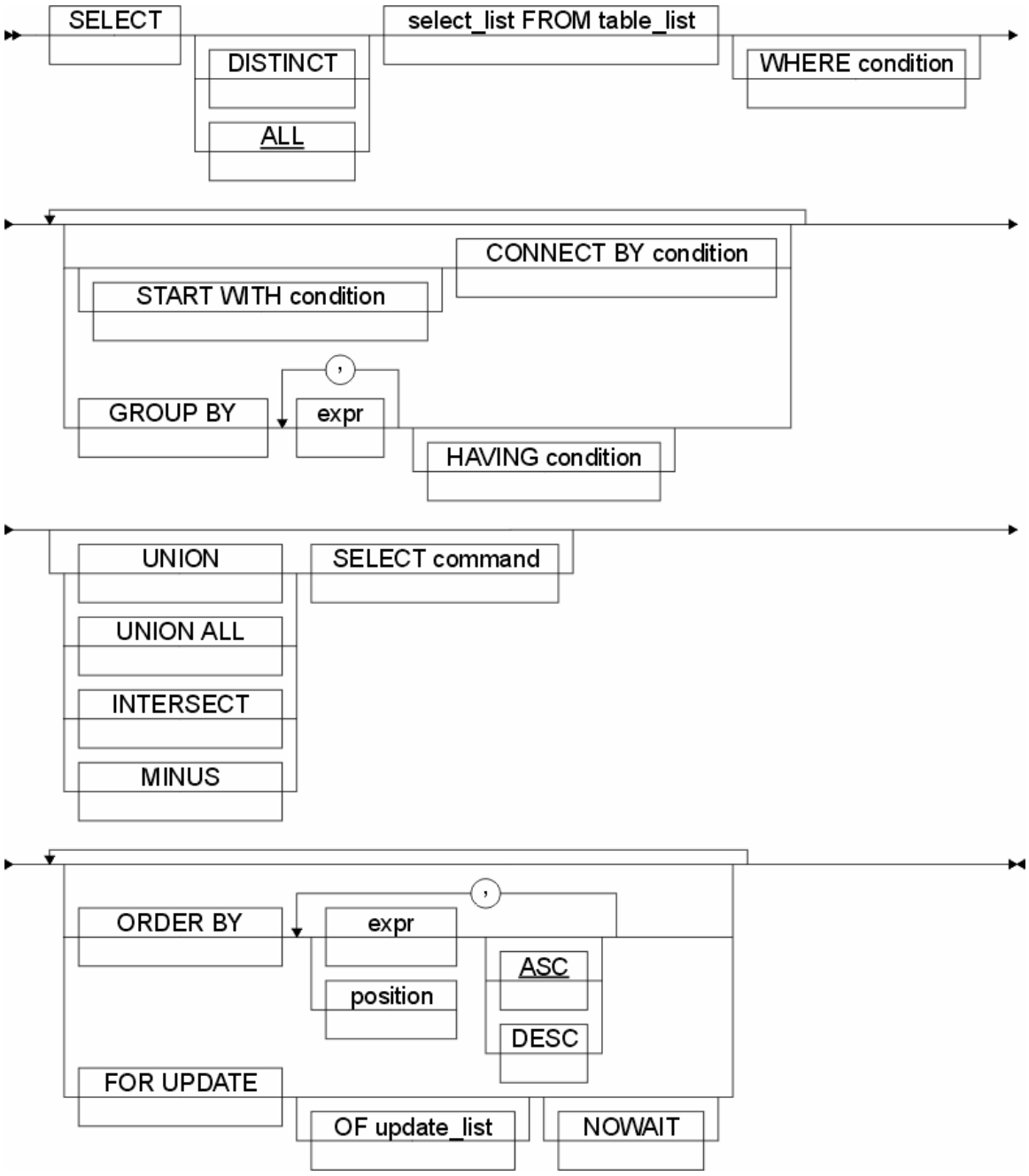


Рис. 12.3 SELECT.

Примеры.

*Показать всех врачей заведенных в БД.

SELECT * FROM doctors ORDER BY dc_name;

Результат: все записи из таблицы DOCTORS отсортированные по полю dc_name по алфавиту (по убыванию).

* Показать всех врачей с кодом специальности равным 111.

SELECT dc_name

```
FROM doctors
WHERE dc_speciality_nnn = 111
ORDER BY dc_name;
```

* Показать всех врачей с кодом специальности равным 111 или 112.

```
SELECT dc_name
FROM doctors
WHERE dc_speciality_nnn = 111
      OR dc_speciality_nnn = 112
ORDER BY dc_name;
```

2-ой способ. Он лучше!

```
SELECT dc_name
FROM doctors
WHERE dc_speciality_nnn in (111, 112)
ORDER BY dc_name;
```

Операции над множествами

Результат работы SELECT – множество из значений атрибутов. Над этими множествами можно производить 4 базовых операции: UNION, UNION ALL, INTERSECT, MINUS.

UNION – комбинирует два запроса, возвращает все неповторяющиеся строки, извлечённые хотя бы одним из запросов.

UNION ALL – также включает повторяющиеся строки.

INTERSECT – возвращает все неповторяющиеся строки извлечённые каждым из запросов.

MINUS – возвращает строки извлечённые первым запросом и неизвлечённые вторым запросом.

Операции внутри SELECT запросов

***** - выбрать всё.

(+) – указывает, что предшествующий столбец является столбцом внешнего соединения.

PRIOR – при построении иерархических древовидных запросов.

ALL – выбрать всё.

DISTINCT – выбрать неповторяющиеся записи.

Примеры.

*Показать всех врачей с кодом специальности равным 111 и работающих в подразделении №2.

```
SELECT dc_name
FROM doctors
WHERE dc_speciality_nnn = 111
    AND dc_shtat_nnn = 2
ORDER BY dc_name;
```

* Показать всех пациентов врача Иванова А. А.

```
SELECT pt.pt_name
FROM patients pt
,doctors dc
WHERE dc.dc_nnn = pt.pt_dc_nnn
AND dc.dc_name = 'ИВАНОВ А. А.'
ORDER BY pt.pt_name;
```

dc.dc_nnn = pt.pt_dc_nnn – это JOIN, связь, которая строит правильное произведение при обращении к двум таблицам (или более).

Лекция №13, 14

04.04.07 г

Системные привилегии и роли

Это механизмы, позволяющие распределять права доступа пользователей к возможностям проводить действия в БД, обращаться к объектам.

Рис. 13.1 Системные привилегии

Привилегия – возможность пользователя выполнять действия над объектами в схемах пользователя.

Системная привилегия – привилегия на объекты схем SYS и SYSTEM.

Роль – специальный тип объекта БД (группировка), который используется для упрощения назначения списка привилегий.

Описание картинки: пользователь владеет объектами, которые располагаются в схемах. Пользователь владеет объектами посредством ролей, которые включают системные или объектные привилегии.

Пользователь владеет объектами БД, которые располагаются в его схеме, либо в другой, к которой у него есть доступ. Доступ пользователя к объектам осуществляется согласно ролям, которые включают системные и объектные привилегии. Системные привилегии назначаются на действия пользователя с объектами (например CREATE TABLE, CREATE ANY TABLE, DROP TABLE и т.д.). Объектные привилегии назначаются на доступ к объектам в разных схемах.

Можно создать схему под свою БД и дать пользователю права даже на отдельные поля! Тогда не нужны ни триггеры, которые будут следить за пользователем. Легко реализовать защиту БД.

Роль CONNECT – минимальная роль, позволяющая подключиться к БД.

Роль может включать роли (можно засунуть роль DBA в CONNECT), системные привилегии, объектные привилегии и пользователей групп.

Системные привилегии нужно знать. Их легко запомнить. CREATE SEQUENCE – в своей схеме, CREATE ANY SEQUENCE – в любой схеме. Назначаются так: GRANT ALTER ANY INDEX TO USER1.

Объектные привилегии … дальше отвлекли

Для удобства управления ролями и привилегиями можно использовать Enterprise Manager (начиная с 10 версии: localhost:5500/em).

Команды управления привилегиями и ролями

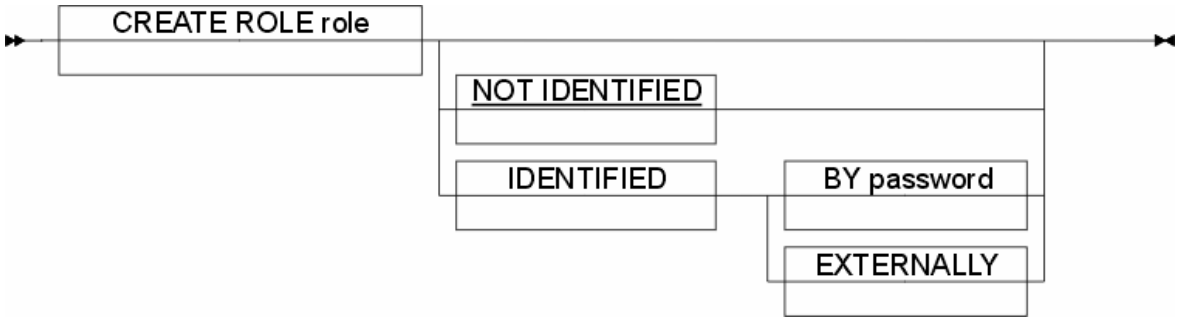


Рис. 13.2 CREATE ROLE.

Расшифровка: создать роль с авторизацией (без авторизации) по паролю (с внешней авторизацией).

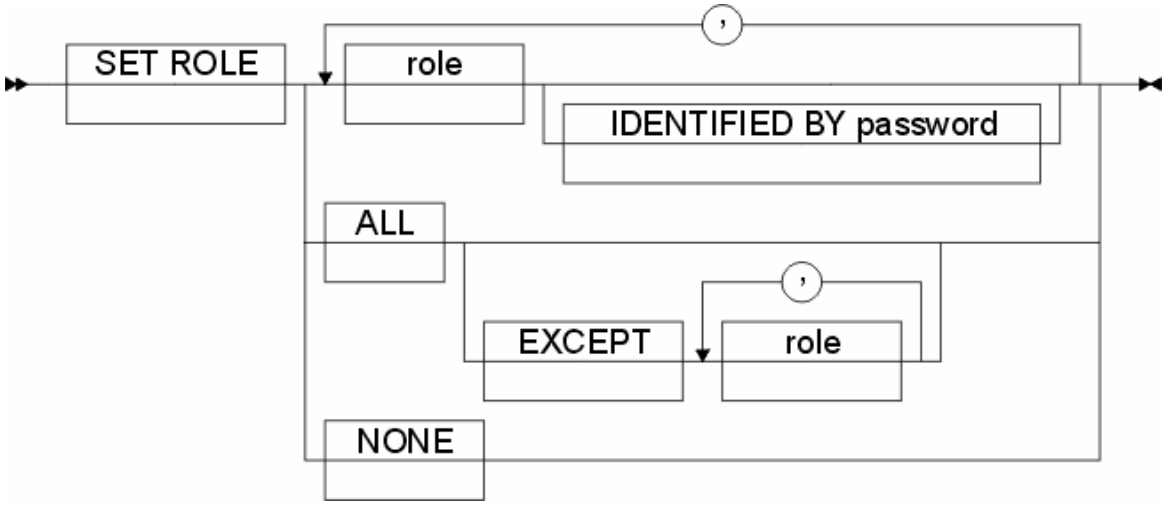


Рис. 13.3 SET ROLE.

Разрешает заданную роль только в текущем сеансе и запрещает все другие роли. Нужна, чтобы дать роль временно.

Расшифровка: дать роль имя_роли, параметры идентификации, обобщения (с отключением группировок роли).

Команда GRANT используется в двух вариантах:

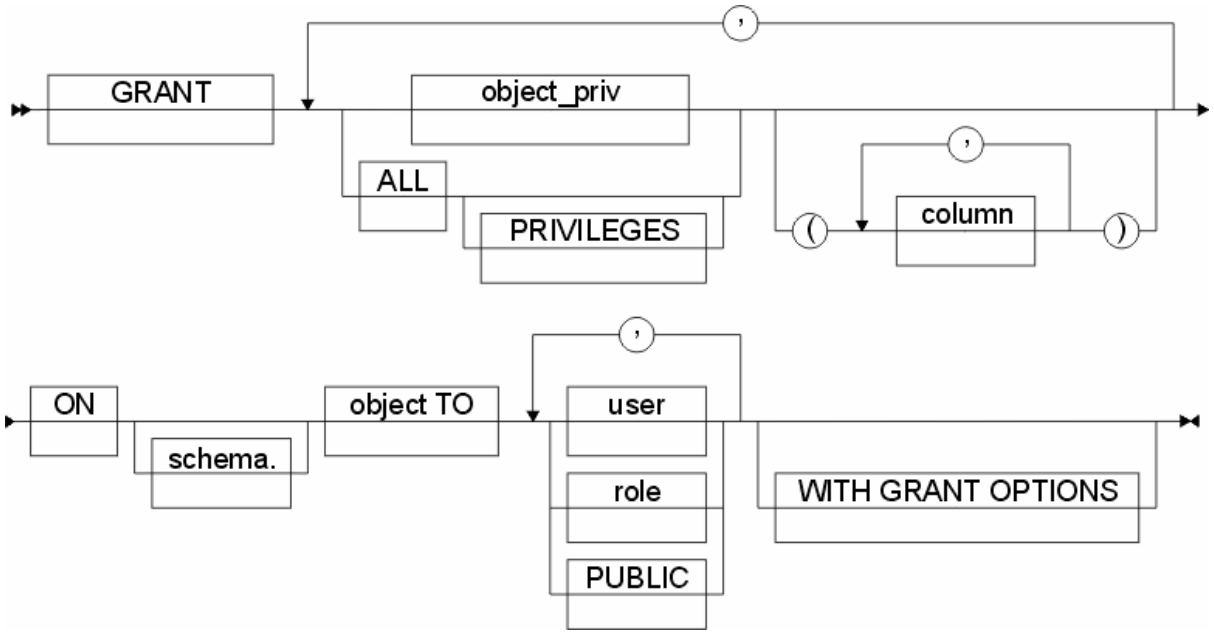


Рис. 13.4 GRANT. Объектные привилегии

Расшифровка: дать имя_привилегии (либо все привилегии) на конкретные объекты (колонки, атрибуты) на объект схемы пользователю (роли, публичному синониму) с опцией назначения.

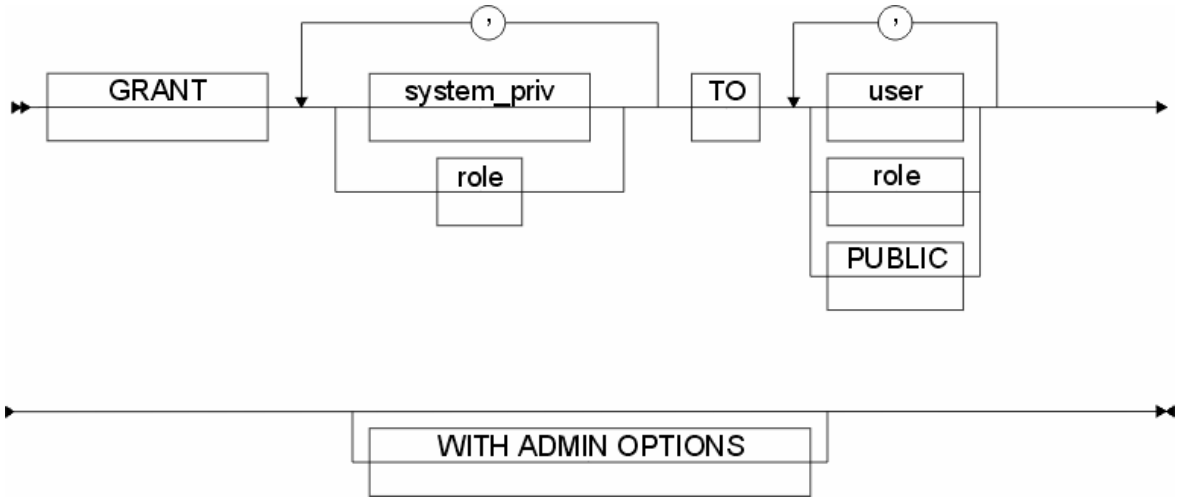


Рис. 13.5 GRANT. Системные привилегии

Чтобы забрать роль:

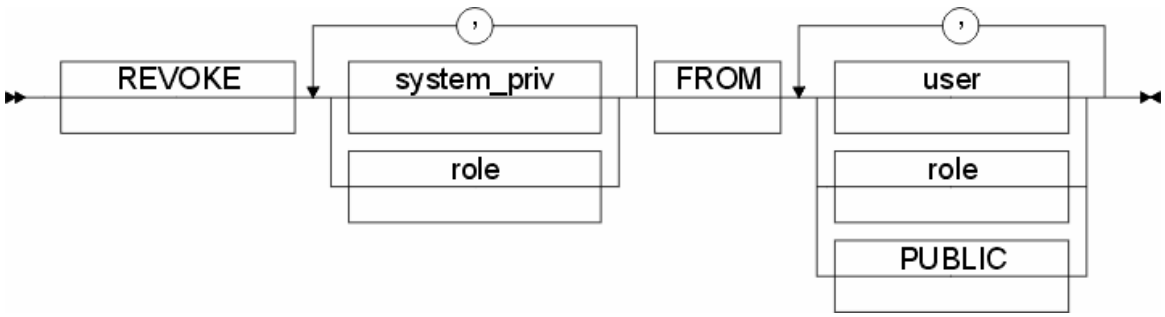


Рис. 13.6 REVOKE.

Выполнять операции с ролями и привилегиями может только привилегированный пользователь (DBA).

ДЗ.

Для системы, развёрнутой дома, сделать отдельную схему пользователя с ролью только CONNECT. Все задания выполнять только в ней, последовательно добавляя необходимые привилегии. В конце концов все назначенные привилегии собрать в роль STUDENT и её назначить себе))).

Классификация системных привилегий

Структура наименования системных привилегий:

[Функциональность привилегии][ANY (для любой схемы)][Имя объекта]

Таблица 13.1 Соответствия между объектами БД и привилегиями

	таблица	представление	последоват.	функции	мом. копии
ALTER	x		x		
DELETE	x	x			
EXECUTE				x	
INSERT	x	x			
REFER	x				
SELECT	x	x	x		x
UPDATE	x	x			x
INDEX	x				

Все привилегии также хранятся в системной таблице.

Автоматическая настройка окружения рабочей среды

Инфа лежит на сервере. Освоить можно только на практике.

В настройке выделяются 2 режима: глобальный и локальный.

Глобальный: размещается на сервере, glogin.sql. Расположен: $ORACLE_HOME/sqlplus/admin. В этом файле представлен набор команд SET, которые будут соответствовать любому сеансу SQLPLUS.

Локальный: найти у себя файл slogin.sql. Там настраиваем.

Подробная документация по всем настройкам находится на сайте в разделе isqlplus.

ДЗ

Настроить локальный файл, чтобы для редактирования sql-скриптов (команда edit) автоматом запускался блокнот.

Основы PL/SQL

Это процедурное расширение языка SQL. Позволяет реализовать динамические конструкции на сервере.

Свойства:

- Позволяет передавать на сервер программный блок, содержащий логику приложения, и выполнять его одним запросом.
- Блоки позволяют использовать библиотечные повторяемые конструкции.
- Вся логика приложения делится на клиентскую и серверную часть. Вся логика приложения должна быть на сервере.

Основные элементы PL/SQL

1. Ненаименованный PL/SQL блок. Конструкция BEGIN … END, которая работает только в текущем сеансе.
2. Наименованный PL/SQL блок.
    a. Функции – часть логики приложения, ориентированная на выполнения комплекса операций на сервере, результат которой представляется в виде значения функции. Возвращает значение присваемое имени функции.
    b. Процедура возвращает значение через параметры.
    c. Пакет – часть логики приложений, которая объединяет функции и процедуры. Аналог роли.

Ненаименованные PL/SQL блоки

Структура программы:

DECLARE – определяет начало объявления переменных
-- - комментарии

BEGIN – тело блока

EXCEPTION – операторы обработки исключительных ситуаций
Команда, которая ловит код ошибки и выводит пояснение.
END;

Обязательным параметром настройки среды окружения является SERVEROUTPUT ON (SET SERVEROUTPUT ON)

***Пример.**
SET SERVEROUTPUT ON -- определяет вывод SQL*Plus всю информацию
--возвращаемую сервером.
BEGIN
DBMS_OUTPUT.enable; -- включение механизма вывода

DBMS_OUTPUT.put_line('IU4-SUX'); -- печать строки
END;
/ -- указание к выполнению блока PL/SQL
DBMS_OUTPUT.enable и DBMS_OUTPUT.put_line – функции пакета **DBMS_OUTPUT** (пакет для вывода в выходной поток SQL*Plus)

/=RUN

Чтобы увидеть встроенные процедуры и функции надо зайти в PACKAGE в схему SYS. Часть пакетов зашифрована – защита интеллектуальной собственности.

***Пример.** Вычисление длины окружности и площади круга.

```
DECLARE
PI CONSTANT REAL := 3.141519265359;
LOKR REAL;
SKRG REAL;
RADIUS REAL := &RADIUS; -- переменная вводится с клавиатуры
BEGIN
LOKR := PI * RADIUS * 2.0;
SKRG := PI *RADIUS ** 2;
DBMS_OUTPUT.put_line('Радиус = ' || to_char(RADIUS)|| ', ДЛИНА
ОКРУЖНОСТИ =' || to_char(LOKR) || ', ПЛОЩАДЬ КРУГА =' ||
to_char(SKRG));
END;
/
```

Условные конструкции PL/SQL

IF условие BEGIN … END THEN ELSE … END IF;

Циклы.

Безусловные.

Условные.

Итеративные.

Основные встроенные функции

Целочисленные функции.

Обращение: SELECT func FROM DUAL;

Символьные функции.

CHR(n)	Вывести символ с кодом n
CONCAT(char1,char2)	Конкатенация двух строк
INITCAP(char)	Символьная строка char, первые буквы всех слов в которой преобразованы в прописные.
LOWER(char)	Все буквы преобразованы в строчные
LPAD(char1.n [,char2})	Дополнить строку слева
LTRIM(char[,set])	Удаление части строки слева
REPLACE(char, search_string [,replacement_string])	Заменить

SUBSTRB(char,m[,n])	Вырезать часть строки. Часто используется в WHERE
UPPER(char)	Преобразование в прописные. Обязательно нужна для сравнения строк!!!

Функции с префиксом NLS обрабатывают пустые строки.

Символьные функции, возвращающие числовые значения.

ASCII(char)	Возвращает код первого символа строки
INSTR(char1.char2[,n[,m]])	Вычисление позиции символа. Полезна для вычисления длины строки
LENGTH(char)	

Групповые функции.

Это функции в операторах SELECT.

AVG([DISTINCT|ALL]n)	Среднее
COUNT([ALL]*)	
MAX([DISTINCT|ALL] expr)	
STDDEV([DISTINCT|ALL] n)	Отклонение
SUM([DISTINCT|ALL] n)	
VARIANCE([DISTINCT|ALL]n)	Дисперсия

Примеры использования функций агрегирования (дз).

Лекция №15

11.04.07 г

Функции для работы с датами.

SYSDATE	Возвращает текущую дату и время сервера
TO_DATE	Преобразование в дату
TO_CHAR	Преобразование в символ
ADD-MONTHS (d,n)	Добавить n месяцев к дате d
LAST-DAY (d)	Узнать последнее число месяца d
MONTHS-BETWEEN (d,e)	Узнать число месяцев между датами
NEW-TIME (d,a,b)	
NEW-DAY (d,char)	Дата следующего дня
ROUND(d[,fmt])	Округление
TRUNC(d[,fmt])	Усечение

fmt – форматная маска

Форматные маски – в журнале лабораторных работ.

Форматная маска по умолчанию: DD-MON-YY.

Форматные маски при использовании TO_CHAR

1. 9 – возвращаемая цифра. **Пример:**

TO_CHAR(9999,’999’)

‘0999’ – добавление 0 перед числом;

’99.99’

2. 0 – добавляет 0 перед числом;

3. $ - добавляет $;
4. B – заменяет 0 пробелами;
5. MI – добавляет минус после отрицательных величин
6. S – добавляет плюс или минус.

и т.д. Смотри журнал лабораторных работ.

ДЗ

По каждой маске сделать: SELECT TO_CHAR(…) FROM DUAL;

Разместить это как 211 задачу.

Функции преобразования.

TO_CHAR (expr [,fmt[,'nls_num_fmt']]) – 'nls_num_fmt' – значение, которое возварщается, если expr null.

TO_DATE (char[,fmt [,'nls_lang']])

TO_NUMBER (char [,fmt[,'nls_lang']])

Дополнительные вспомогательные функции Oracle.

DECODE (expr, search1, return1, [search2, return2,]...[default]) – проводит сравнение expr и search. Если они равны, то она возвращает return, иначе false. Часто используется для запросов SELECT.

DUMP(expr[, return_format [,start_position[, length]]]) – не принципиально))))

LEAST(expr[, expr]...) – наименьшее значение выражений

NVL(expr1, expr2) – заменяет NULL значения на expr2. Если в таблицах не хватает ограничений – эта функция используется при вычислениях.

USER – имя текущего пользователя

VSIZE(expr) – длина представления expr.

Управляющие структуры PL/SQL

Позволяют обрабатывать логические ситуации. Условные операторы и циклы.

IF лог. выр1 THEN

…

[ELSEIF лог.выр2 THEN …]

ELSE …

END IF;

***Пример.**

```
DECLARE
v_num1 NUMBER;
v_num2 NUMBER;
v_result VARCHAR(7);
BEGIN
IF v_num1>v_num2 THEN
v_result:=’NO’;
ELSE
v_result:=’YES’;
END IF;
END;
```

ДЗ

Модифицировать для замены пустых значений и оформить 212 задачей.

Виды циклов.

Простой цикл. LOOP.

Выполняется, пока не выполнятся все операторы.

```
LOOP
…
END LOOP;
```

Условный цикл. WHILE.

```
WHILE условие LOOP
…
END LOOP;
```

Счётный цикл. FOR.

```
FOR счётчик
IN min..max
LOOP
…
END LOOP;
```

***Пример.**

Разработать цикл для последовательного вычисления длины окружности с радиусом, значение которого изменяется в заданных пределах.

```
DECLARE
dt DATE;
radius REAL:=&radius;
BEGIN
dt:=SYSDATE;
FOR i IN 1..5
LOOP
DBMS_OUTPUT.PUT_LINE(TO_CHAR(lokrug(radius)||'время'||TO_CHAR(SYSDATE-dt));
radius:=radius+1;
END LOOP;
END;
/
```

***Пример.**

Условный цикл.

Практически идентично. Лучше в данном случае лучше 1 вариант.

Обработка исключительных ситуаций

Исключительная ситуация – специальная конструкция, которой передается управление при возникновении ошибки.

```
DECLARE
x REAL:=&x;
n INTEGER:=0;
BEGIN
dbms_output.enable;
dbms_output.put_line('Значение функции');
dbms_output.put_line('abs(x)='||TO_CHAR… n=n+1
EXCEPTION
WHEN others THEN BEGIN
dbms_output.put_line('ERRORS');
IF n=0 THEN
```

dbms_output.put_line(‘ошибка в …’);

end;

END;

Создание наименованных PL/SQL блоков

К ним относятся:

- процедуры
- функции
- пакеты
- триггеры

Вопрос на экзамен на засыпку: когда используется функция, а когда процедура? Ответ: если надо вернуть много значений – процедура, одно – функция.

В ДЗ должна быть процедура, функция, засунутые в пакет.

Триггер – наименованный блок, который вызывается при обработке определённых DML операторов

Лекция №16, 17

25.04.07 г

Создание функции. CREATE FUNCTION

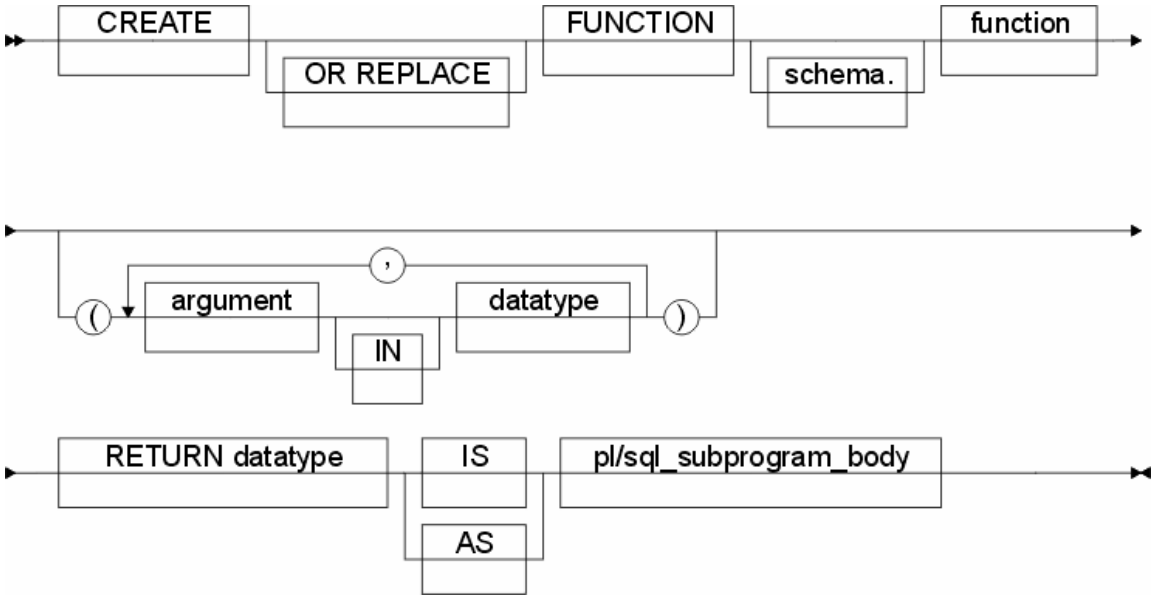


Рис. 15.1 CREATE FUNCTION

Расшифровка: создать или заменить функцию в схеме (аргументы (входная переменная) типа) возвращает тип.

Функция используется, если необходимо вернуть одно значение, присвоенное имени функции.

Вызов функции:

1. SELECT FROM DUAL;
2. BEGIN y=f(x) END;

***Пример.**

```
CREATE OR REPLACE FUNCTION lokrug (rad REAL) RETURN REAL
IS
BEGIN
return (3.14*rad*2);
END;
/
```

Создание процедур. CREATE PROCEDURE

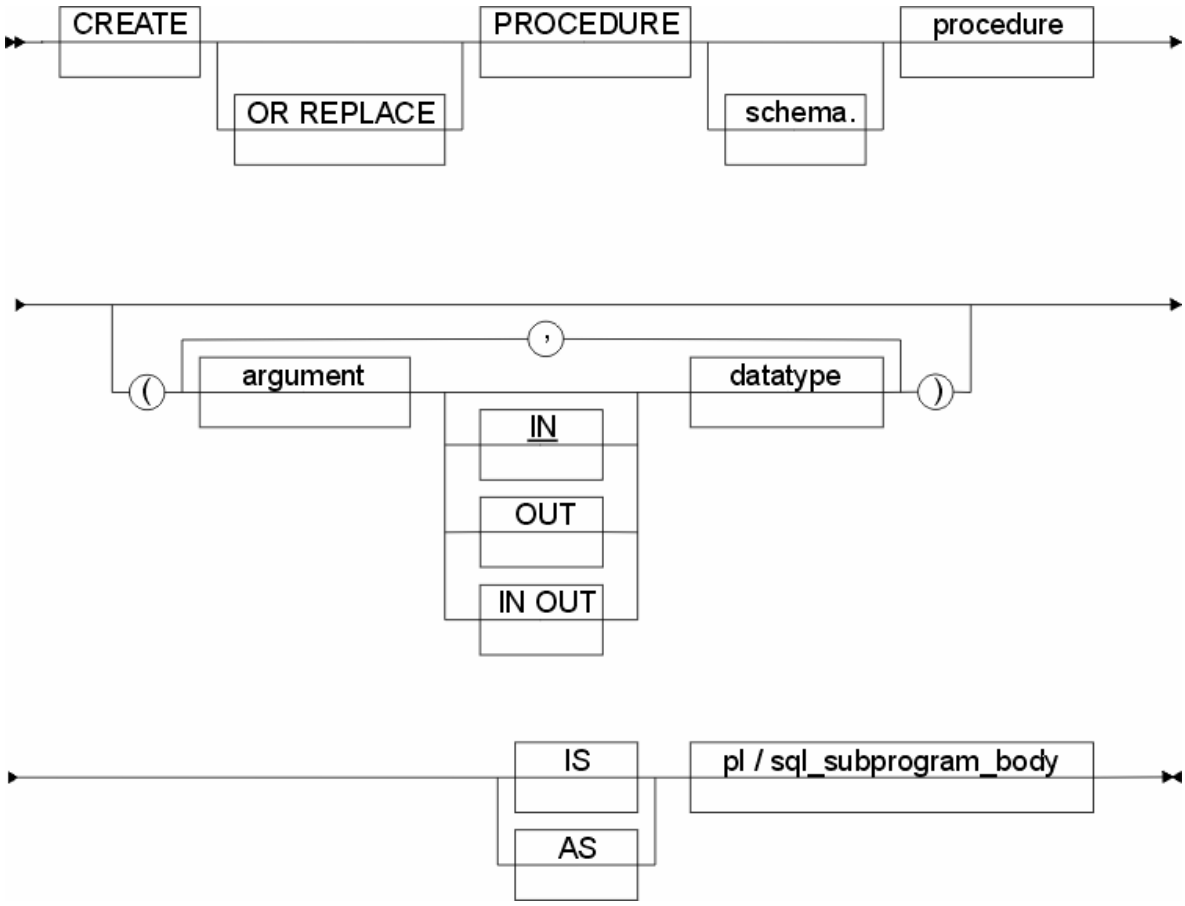


Рис. 15.2 CREATE PROCEDURE

Обращение к процедуре:

```
BEGIN
proc(params);
END;

BEGIN
EXEC proc(params);
END;

BEGIN
proc(params).exec;
END;
```

Замечание: все параметры процедуры должны быть предварительно объявлены.

*Пример.

PROMPT Создание отчёта по всем объектам созданным сегодня

```
CREATE OR REPLACE PROCEDURE proc_user
IS
db DATE :=&db;
dl DATE:=&dl;
a CHAR(20);
b CHAR(20);
c DATE;
e DATE;
BEGIN
SELECT  substr(object_name, 1, 20) name, substr(object_type, 1, 20) type_obj, created, last_ddl_time
INTO a, b, c, e
FROM user_objects
WHERE last_ddl_time>=db
AND
last_ddl_time<=dl
END;
/
```

Фишка: если известно, что в переменную будут загружены данные из таблицы, то лучше объявить её так: **db user_objects.last_ddl_time%type**

Таким образом она будет того же типа, что и данные в таблице. И нам не надо париться о совместимости.

Создание специального типа данных типа таблица.

Определение своих типов данных осуществляется через ключевое слово type

***Пример.**

Хранение и обработка многомерного массива данных с помощью временной таблицы.

```
DECLARE /* специальный тип данных - таблица */
TYPE t_table IS table OF VARCHAR2(10)
INDEX BY binary_integer;
v_table t_table;
vt NUMBER;
BEGIN
```

```
FOR v_c IN 1..50
LOOP
v_table(v_c):=v_c;
END LOOP;
v_t:=v_table.count;
dbms_output.enable;
dbms_output.put_line(v_t);
FOR i IN 1..100
LOOP
IF v_table(i).exists
THEN
dbms_output.put_line('Элемент с номером'||TO_CHAR(i)||'???';)
END IF;
END LOOP;
END;
/
```

Создание пакетов.

Пакет – группировка логических элементов (функций и процедур). Он необходим для упрощения поиска функций.

Пример. dbms_output.put_line(); - обращение к функции put_line пакета dbms_output.

Пакет разделяется на спецификацию и тело для сокрытия логики. Спецификация незашифрована, а тело – может быть зашифровано.

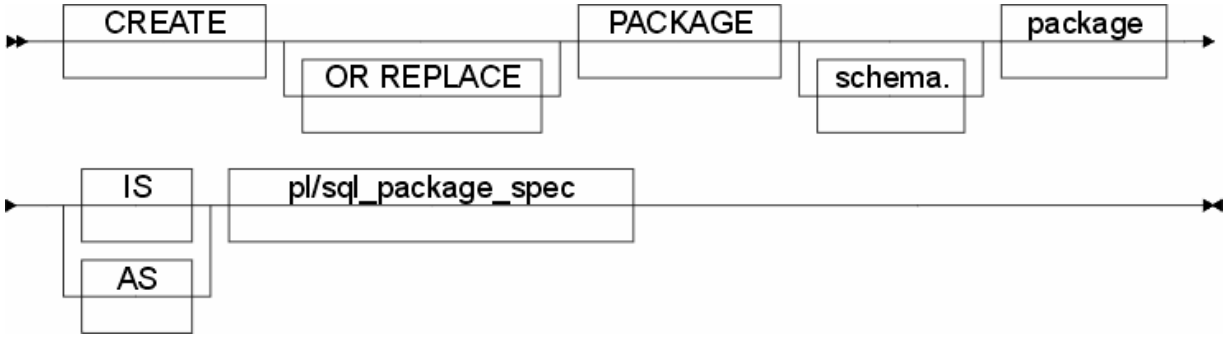


Рис. 15.3 CREATE PACKAGE. Создание спецификации пакета

Создаём или перезаписываем пакет в схеме. Далее блок PL/SQL.

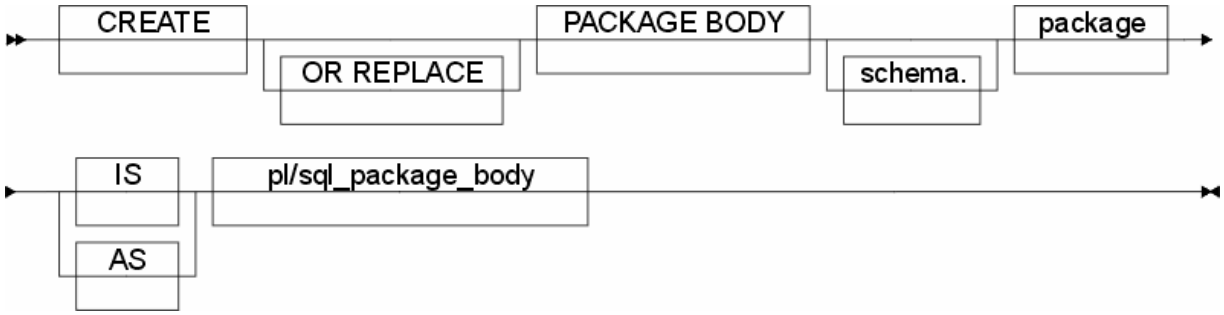


Рис. 15.4 CREATE PACKAGE BODY. Создание тела пакета

***Пример.**

Взять из журнала лаб.

```
/* ******Пакет обработки ошибок ********************* */
create or replace package app_err as
/* Получение текста аварийного завершения */
function get_err return varchar2;
/* Установка текста аварийного завершения */
procedure set_err ( error_text in varchar2);
/* Установка текста аварийного завершения и само завершение */
procedure raise_err ( error_text in varchar2);
end app_err;
create or replace package body app_err as
app_err_text varchar2(32767);
/* ********************************************************** */
function get_err return varchar2
is
A varchar (32767);
Begin
A := app_err_text;
app_err_text := null;
Return ( A );
End;
/* ********************************************************** */
procedure set_err(error_text in varchar2)
is
Begin
app_err_text := error_text;
```

```
End;
/* ************************************************************** */
procedure raise_err ( error_text in varchar2)
is
Begin
app_err_text := error_text;
raise_application_error (-20000, error_text);
End;
end app_err;
/* ** Функция осуществляющая расшифровку пароля пользователя в БД ** */
create or replace function password return varchar2 is
name varchar2(23);
pass varchar2(23);
begin
select ORANAME, CRYPT_PASSWORD
into name, pass
from USERS
where USER_ORANAME = user;
pass := encrypt( pass, name );
return( pass );
end;
/*Удаляется публичный синоним*/
drop public synonym password;
/*Создается публичный синоним*/
create public synonym password for password;
/* устанавливаем привелегии*/
grant all on password to admin;
```

Разработка прикладного клиентского программного обеспечения

Клиентское ПО	
Borland Delphi	CGI/ISAPI
Borland C++	PHP/FI
MC++	JAVA
…	ActiveX (ADO)
Толстый клиент (часть логики реализована на стороне клиента)	**Тонкий клиент**

CGI – common gateway interface (стандартный интерфейс шлюза). ISAPI – конкурент от MS.

PHP/FI – personal home page/form interface

PHP появился как упрощённая замена CGI.

Сделать краткий конспект со слайда по обзору всех технологий.

Краткий конспект по обзору технологий.

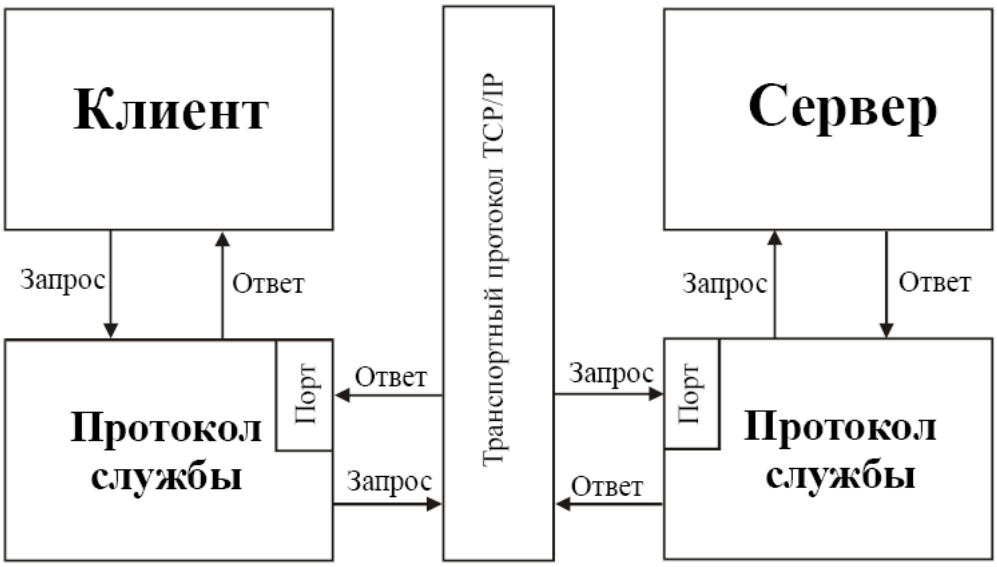


Рис. 15.5 Функциональные взаимосвязи клиента и сервера в сетях TCP/IP

Архитектура бывает 2 и 3 уровневая. В ДЗ – 2 уровневая (PHP – не уровень).

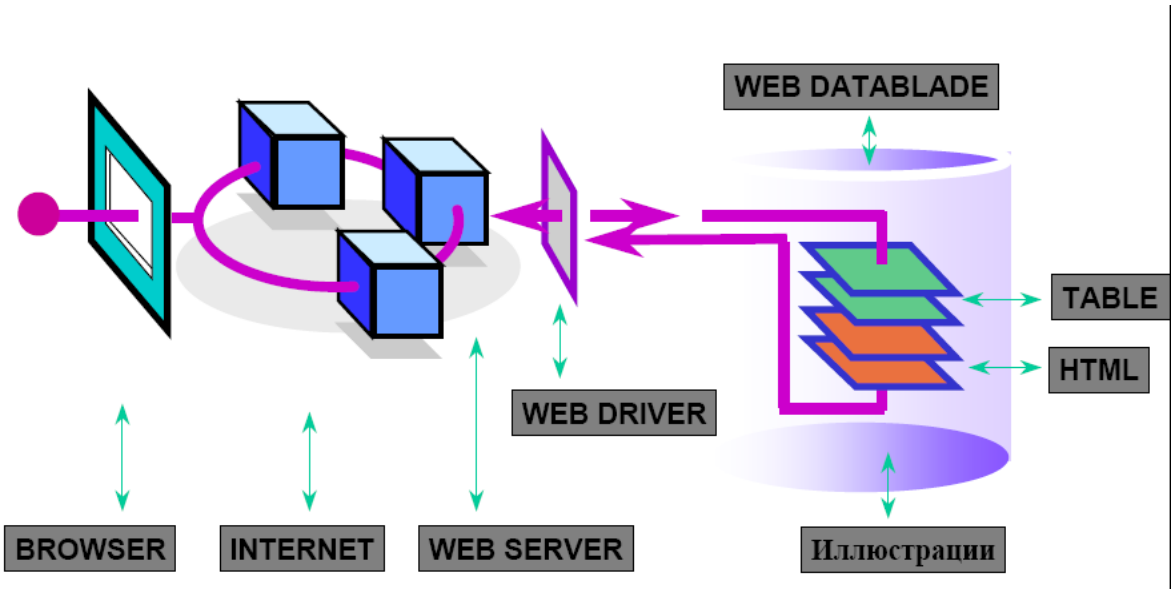


Рис. 15.6 АРХИТЕКТУРА «ИНТЕРНЕТ-ИНТРАНЕТ»

Это (выше) схема размещения. Для нас – диаграмма размещения.

CGI – программы на С на сервере.

CGI - это механизм для выбора, обработки и форматирования информации. CGI используется, чтобы принять ввод пользователя и передать его с сервера Web базе данных.

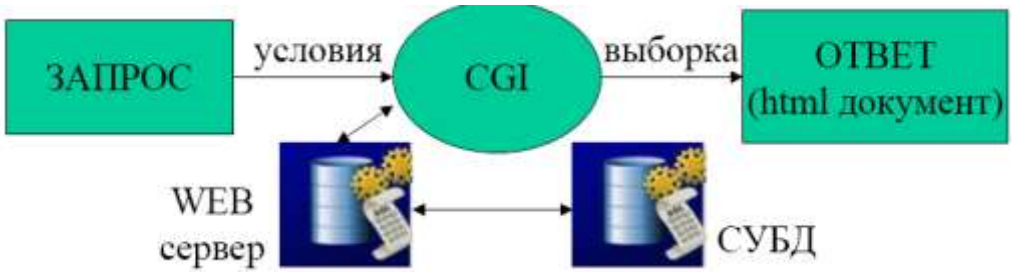


Рис. 15.7 CGI

CGI - это стандарт. Он определяет, как серверы Web передают информацию, используя приложения, исполняемые на сервере. Это способ расширения возможностей сервера.

При написании CGI можно использовать почти любой язык программирования. Самые популярные - shell, Perl, C и C++.

Порядок его работы:

1. Браузер принимает информацию.
2. Браузер помещает информацию в URL.
3. Браузер подключается к серверу и запрашивает URL. Сервер запускает соответствующий сценарий.
4. Сценарий обрабатывает данные.
5. Сценарий формирует веб-страницу.

6. Сервер возвращает страницу браузеру.
7. Браузер её отображает.

XML

XML не содержит никаких тэгов, предназначенных для разметки, он просто определяет порядок их создания. Одним из важных преимуществ XML является возможность использования его в качестве универсального языка запросов к хранилищам информации. Сегодня рассматривается рабочий вариант стандарта XML-QL(или XQL), который, возможно, в будущем составит серьезную конкуренцию SQL.

PHP технологии

Технология придумана для упрощения программирования серверных скриптов в виде набора серверных сценариев.

Это набор функций, который передаётся интерпретатору, подключенному к Apache. Это некомпилирумая вещь – это сценарий! (в отличие от CGI).

Свойства PHP:

1. Устраняет потребность в многочисленных небольших CGI программах.
2. Позволяет размещать пользовательский сценарий в теле HTML.
3. Позволяет обеспечить портальную реализацию веб-системы.
4. Встроенная поддержка для работы с БД (для работы с Oracle используется OCI (oracle connect interface) и ORA).

Разработка ППО под PHP под Oracle

Доступ к БД средствами PHP.

Чтобы обеспечить взаимодействие с БД будем использовать библиотеку OCI. Она должна быть установлена.

Установка PHP под Windows и Apache

С СУБД уже ставится Apache. Он настроен уже на работу с Oracle. Надо будет найти отдельный пакет PHP, ставить и руками править конфиги Оракловского Апача.

1. Распаковать дистрибутив в каталог C:/php. Найти файлы: php4ts.dll, php_oci8.dll, php_oracle.dll, php_openssl.dll и кинуть их в system32

2. Установить Apache. Изменить файл **httpd.conf .** В нём хранятся настройки окружения сервера.

#for the apache module

LoadModulephp4_modulec:/php/sapi/php4apache.dll

AddType application/x-httpd-php.php4

#for the cgi binary(you can use that one compiled with force cgi redirect too)ScriptAlias/php4/ “c:/php/"Action application/x-httpd-php4 "/php4/php.exe"AddType application/x-httpd-php4 .php

2. Вносим изменения:

Скопировать php.ini-dist в папку Виндовс, и переименовать его в php.ini. В нём раскомментировать строки: extension=php_oci8.dll
extension=php_openssl.dll

extension=php_oracle.dll

3. Запустить Апач.
4. Проверить функциональность сервера – localhost. Должна пявиться страница.

Установка PHP под UNIX и Apache

1. Проверить, включена ли в Апач поддержка модулей – httpd –l. Ответ: http_core.c, mod_so.c – поддержка модулей включена.
2. Распаковать архив: gunzip php-4.0.2.ta.gz.|tar xf; cd php- 4.0.2
3. Компиляция. make install
4. Настройка Apache
    a. Скопировать php.ini-dist в каталог с httpd.conf, переименовав его в php.ini.
    b. Добавить строку: addtype application/x-httpd-php.php.phtm
    c. Перезапустить Апач: /etc/rc.d/init.d/httpd restart

Проверка работоспособности внесённых изменений

Разработать страницу index.php, разместить её в корневой директории сервера:

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<?
-- выводим комментарии
echo “PHP”;
phpinfo(); -- вызываем функцию информации о PHP
?>
```

Основы PHP

Для Апача необходимо правильно указывать расширение файла. Объявление PHP скрипта в файле:

1. <? … ?>
2. <?php … ?>
3. <% … %>
4. <script language=”php”> … </script>

Синтаксис PHP скрипта.

<объявление PHP

//объявление переменных

$a=1;

$b=2;

/* тело скрипта */

$c=$a+$b;

echo “$c”;

закрытие PHP>

Требования к стилю программирования на PHP

1. Каждый блок кода должен иметь отступ в 4 пробела:

```
{
 some code;
        {
        some code;
        }
}
```

2. Для всех описаний определений и вызовов функций:
    a. Скобка после имени функции следует без пробела:

sin(x);

if(something) {do};

for(inti=1; i<10; i++);

    b. После открывающейся скобки и перед открывающей скобкой обязательно ставится пробел: sin((x+(y-1))). Если количество скобок чётно – без пробела

3. Каждый файл исходного текста программы должен содержать заголовок в виде комментария с кратким описанием функциональности, а также информация об авторе.

Пример.

```
////////////////////////////////
// form.php:command list class
//
//© 2002 SAS
```

4. Критические места комментируются.

5. Функции представленные для внешнего доступа должны быть снабжены кратким описанием возможностей, входных параметров и возвращаемого значения.

Переменные в PHP.

Типы переменных:

- integer – целые;
- double –действ. переменные;
- string – строковая переменная;
- array – массивы;
- object – объектная переменная.

Виды переменных:

- глобальные (доступны из разных PHP модулей);
- локальные (доступны только внутри модуля).

Инициализация переменных происходит путём присваивания.

Инициализация массивов:

1. Поэлементное задание элементов (начиная с 0): $array[]="0 element";
2. Используя array()

Инициализация объектов:

1. Создаётся класс
2. Создаётся объект – экземпляр класса

```
class br
{
        function make_br ()
        {
                echo"<br>";
        }
}
$br1 = new br;
$br1 -> make_br(); //Объект br1 вызывает метод make_br
```

Области видимости переменных:

```
$a = 1;
$b = 2;
Function Sum ()
  {
    global $a, $b;
    $b = $a +$b;
  }
Sum();
echo $b;
```

```
$a = 1;

Function Test ()
        {
         echo $a;
        }
Test();
```

Слева пример правильный, справа – неправильный (надо **а** внести в функцию).

Статические переменные в PHP.

Статическая переменная – аналог последовательности (некорректная аналогия). Доступна внутри конкретного модуля и хранит текущее значиение. Используется, если надо сохранить значение и потом его использовать.

*Пример.

```
Function Test ()
{
        $a = 0;
        cho $a;
        $a++;
}
```

В результате всегда будет 0

```
Function Test ()
{
        static $a = 0;
        echo $a;
        $a++;
}
```

Результат – число обращений.

Передача переменных из форм.

***Пример**:

передать значение, которое будет введено в поле Name данной формы.

<form action=“index.php” method=“post”>

Name: <input type=“text” name=“name”>

<input type=“submit”>

</form>

Итог: значение переменной $name = значению поля NAMEи оно передано файлу index.php на обработку.

***Пример. Передача нескольких переменных как элементов массива.**

передать значения, которое будет введено в поле Name данной формы.

<form action=“index.php” method=“post”>

Name: <input type=“text” name=“pers[name]”>

E-mail: <input type=“text” name=“pers[e-mail]”>

Beer:

<select multiple name=“beer[]”>

<option value=“светлое”>W

<option value=“темное”>G</select>

<input type=“submit”>

</form>

***Пример. Передача нескольких переменных как элементов массива.**

```
// Имя базы данных, к которой производится подключение
        $db= "test";
// Подключение к серверу базы данных
        $conn=ocilogon("test","test",$db);
// Формирование запроса к базе
$statement=ociparse($conn,"Select * from TEST"); // отпарсить запрос)))
OCIExecute($statement); // выполнение запроса на сервере
echo“<table border= 1>";
while(OCIFetchInto($statement, $row, OCI_ASSOC))
        {
        echo"<tr border= 1>";
        echo $row['FIO'];
        echo"</tr>";
        }
echo"</table>";
// Подтверждение транзакции
OCICommit($conn);
```

```
// Разрыв соединения с сервером
ocilogoff($conn);
?>
```

Результат – таблица ФИО из test.

Вся авторизационная инфа лежит на серваке в одном файле, который лежит ВЫШЕ каталогом, чем индекс.

ДЗ.

Отработать пример. Выложить отдельной задачей.

Основные примеры реализации отдельных конструкций на PHP.

Подходы к программированию

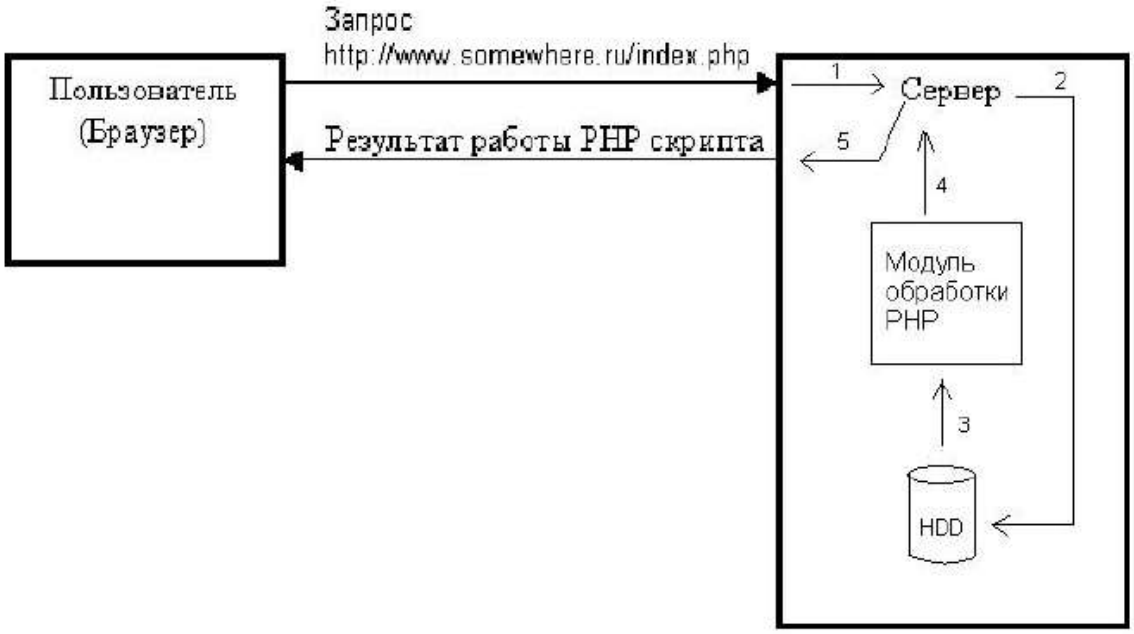
Структурное программирование

```
<?php
function ShowAdminInterface() {...}
function ShowInteface($user_rank)
{
  ShowHeader();
  switch ($user_rank)
  {
    case “admin”: ShowAdminInterface();
    ……..
  }
 ShowFooter();
}
ShowUserInterface(_POST[“user_rank”]);
?>
```

Объектный подход

```
abstract class UserInterface
{
 function __construct (){}
 private  function ShowHeader(){..}
 private function ShowFooter() {...}
 public ShowInterface()
 {
  ShowHeader();
  ShowUserInterface();
  ShowFooter();
 }
 abstract protected function ShowUserInterface();
}
class AdminInterface extends UserInterface
{
 protected ShowUserInterface{//определяем здесь
поведение}
}
```

Обработка запроса пользователя



Приведение типов

(int), (integer) -приведение к целому числу

(bool), (boolean) -приведение к булеву типу

(float), (double), (real) -приведение к числу с плавающей точкой (float)(string) -приведение к строке

(array) -приведение к массиву(object) -приведение к объекту

Пример№1

$TotalFullPacks= (int) ($AllApples/ $NumberOfApplesInPack);

Пример№2

$points = (float) gamer[“points”] //пришло из БД в виде string

Переменные и синтаксис

- Переменные автоматически определяют свой тип по содержанию
- Для выявления типа переменной можно пользоваться специальными функциями: bool is_int (mixed var)...

```
<?php
$var = «Bonjour»;
$var = 5;
while (5 == $var)
{
   $var = True;
   if ($var) $var = «Bye»;
echo $var;
}
?>
```

Глобальные массивы

Г.М. -массив, генерируемый PHP и содержащий определенный набор переменных

- $_POST-массив переменных, переданных методом POST
- $_GET -массив переменных, переданных методом GET
- $_SERVER -массив серверных переменных
- $_SESSION -массив переменных сессии
- $_ENV -массив переменных окружения
- $_COOKIE -массив переменных cookie
- $_FILES -массив переменных файлов, загружаемых на сервер, переданных методом POST

Функции

- Функции возвращают тот тип, который имеет переменная после ключевого слова return
- Начинаются с ключевого слова functionfunc_name(args)
- Параметры передаются по значению, либо по ссылке
- & -передача параметра по ссылкеfunction

***Пример.**

```
function
dummy(&$is_dummy)
{
        if (is_bool($is_dummy))
                $is_dummy = true;
```

```
        else if(is_int($is_dummy))
                return ++$is_dummy;
        else
                echo«void»;
}
```

***Пример. Почти из ДЗ.**

Config.php

```
<?php$db_username= «user»;
$db_pwd= «pwd»;
?>
```

DB.php

```
<?phprequire_once(«Config.php»)
$conn= @ocilogon(«user», «pwd», «orcl»);
if (!$conn) die(«UnabletoconnecttoDB!»);
?>
```

Login.php

```
<?php
require_once(«db.php»);
require_once
if ( (! isset($username) || ! isset($pwd)) ||empty(username) ||
empty($pwd))ShowLoginForm();
else if (Login($username, $pwd))
{
session_start();
$valid_user= $username;
session_register(«valid_user»);
Header(«Location:ui.php»);
}
else
{
echo«Пользователь не найден!<br>»;
ShowLoginForm();}
?>
Login.php::ShowLoginForm()
function Login($username, $pwd)
```

```
{
global$conn;
$md5pwd = md5($pwd);
$stmt =ociparse($conn, «SELECT * FROM LoginData where username= '$username'
pwd= 'md5pwd'»);
if (!$stmt) return false;
ociexecute($stmt);
$result = array();
if (ocifetchstatement($stmt, $result) > 0)
        return true;
else
        return false;
}
```

UI.php

```
<?php
session_start();
if (!session_is_registred(«valid_user»)) Header(«Location:login.php»);
function ShowUserInterface($user)
{
switch($user)case
 ……...
}
ShowUserInterface($valid_user);
?>
```

Report.php

```
<?php
session_start();
if (!session_is_registred(«valid_user»)) Header(«Location:login.php»);
function ShowReport($report_type)
{
switch($report_type)
case «ShowDefectMakers»: ShowDeffectMakers();
…...
```

```
}
ShowReport($report_type);
?>
```

Report.php:: ShowDeffectMakers()

```
<?php
…..
function ShowDeffectMakers()
{
$stmt=ociparse($conn, «SELECT * FROM Woker where defectCount> 5 );
if(!$stmt)return false;
ociexecute($stmt);
print «<table><tr><tdalign = center colspan = 4> Бракоделы</td></tr>»;
print ”<tr><td>Имя</td><td>Должность</td><td>Кол-во
деффектов</td><td>Уволить</td>”
while(ocifetchinto($stmt, $woker, OCI_ASSOC))
        print «<tr>
        <td>$woker[«name»]</td><td>$woker[«position»]</td><td>$woker[«deffectCo
unt»]</td>
        <td><a href =
ui.php?op=delete_woker&woker_id=$woker[«woker_id»]>Пока!</a></td>»;
        print«</table>»;
}
?>
```

Сеансы (сессии)

Сессия-метод сохранения состояния между HTTP-транзакциями

Переменные сессии не могут быть изменены при помощи GET и POST

- cookie-набор-файл с данными, который сохраняется на пользовательской машине
- session_start() -запуск сессии
- session_register (“var1”, “var2”) -регистрация переменной
- session_is_registred (“var”) -проверка регистрации переменной
- session_unregister(“var”) -отмена регистрации переменной

- session_unset() -отмена регистрации всех переменных
- session_destroy() -уничтожение сессии

Использование XML при работе с БД.

XML (extensible markup language) – это язык разметки, описывающий, как можно представить информацию в структурированном виде xml документа.

XML – это платформо-, аппаратно- и программонезависимое средство для представления информации в строгом, структурированном виде, предназначенное для её хранения и передачи.

XML – это набор правил и рекомендаций, официально утверждённых W3C, которые указывают, как можно описать информацию.

XML – это язык разметки, предоставляющий возможность работать с данными как с ветвями деревьев.

XML предназначен для:

- хранения структурированных данных
- для обмена информацией
- просмотра информации в нужном виде.

В XML теги не определены заранее.

Теги должны соответствовать правилам структурированного программирования.

XML документ предназначен для создания унифицированных шаблонов, представлений, формализованных структур данных.

Лекция №18

16.05.07 г

Курсоры

Курсор – объект, который обеспечивает операции над данными на уровне строк предложений языка SQL.

Объявление курсора – это объявление НЕ переменной, а указателя на область памяти, которая используется курсором (зарезервирована для его данных).

Время жизни курсора – от объявления до его закрытия.

Свойства курсора:

1. Объявление курсора определяет, какое предложение будет передано далее для разбора.
2. Вызов и обработка курсора осуществляется логикой приложения.
3. Делятся на явные и неявные (например SELECT).

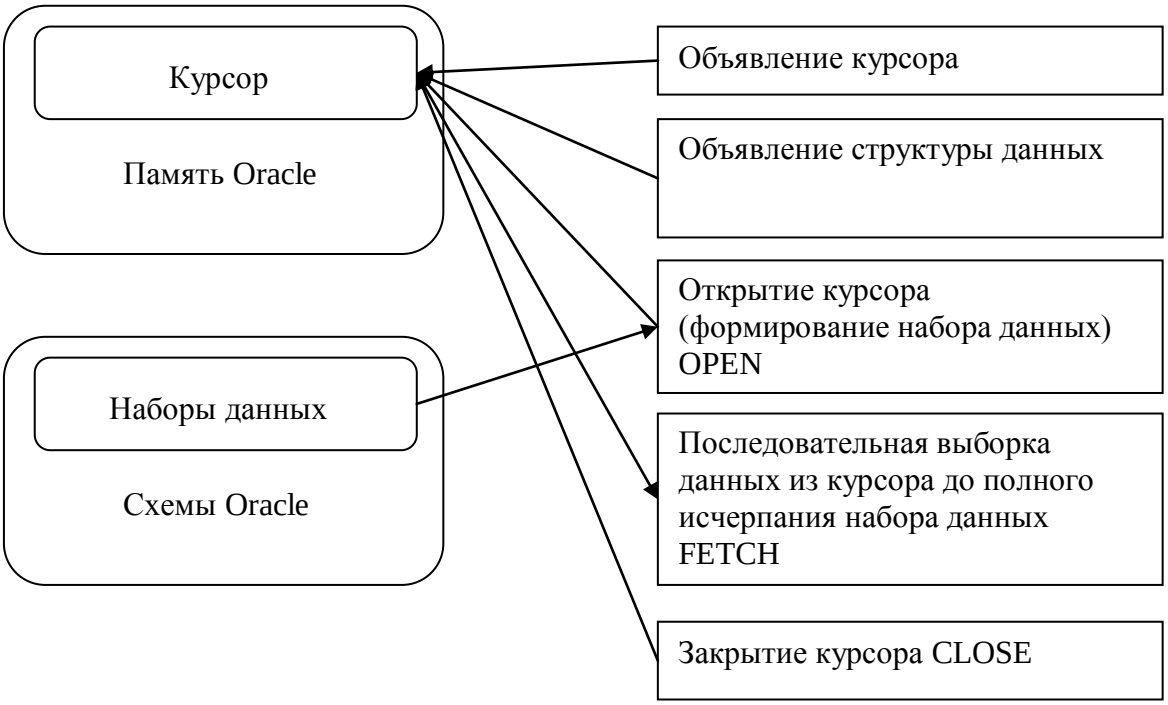


Рис. 18.1

Все наборы данных хранятся в схеме. Курсор – зарезервированная область оперативной памяти, куда можно загружать построчно результат запроса и обрабатывать его.

Последовательность работы:

1. Объявление.
2. Объявление структуры данных (какого типа данные, в каком виде представлены).
3. Открытие курсора (OPEN).
4. Выборка данных.
5. Закрытие. Память освобождается.

Объявление:

DECLARE

CURSOR с_имя_курсора IS SELECT … FROM … WHERE;

По этой команде сформируется курсор. Теперь с ним можно работать.

Обращение:

BEGIN

OPEN c_имя_курсора; Анализируются значения переменных, возвращаемых запросом; формируется активный набор данных в зарезервированной области памяти; указатель помещается на 1 строку.

FETCH c_имя_курсора INTO список_переменных;

FETCH c_имя_курсора INTO PL/SQL блок;

CLOSE c_имя_курсора;

END;

***Пример.**

Циклы обработки курсоров.

1. Использование простого цикла.

OPEN c_имя_курсора;

LOOP

FETCH c_имя_курсора INTO переменная;

EXIT WHEN c_имя_курсора%NOTFOUND;

END LOOP;

CLOSE c_имя_курсора;

2. Курсорный цикл WHILE.

OPEN c_имя_курсора;

FETCH c_имя_курсора INTO переменная;

WHILE c_имя_курсора %FOUND

LOOP

--Обработка выбираемых строк построчно.

FETCH c_имя_курсора INTO переменная;

END LOOP;

CLOSE c_имя_курсора;

3. Курсорный цикл FOR. Используется для работы с записями.

FOR имя_записи IN имя_курсора

LOOP

--Открытие курсора происходит неявно.

END LOOP;

CLOSE c_имя_курсора;

Курсор типа SELECT FOR UPDATE.

Позволяет одновременно выбирать данные и изменять их в таблице.

SELECT … FROM … FOR UPDATE [OF ссылка на столбец] [NOWAIT]

Объявление курсора:

```
CURSOR c_student IS
        SELECT id, first_name, last_name
        FROM students
        WHERE profession=v_profession
        ORDER BY id
        FOR UPDATE;
```

Обращение к курсору:

```
BEGIN
--курсорный цикл
UPDATE students
SET current_credit=current_credits+v_numcredits
WHERE CURRENT OF c_имя_курсора
--конец цикла
END;
```

*Пример.

Выбрать всех студентов конкретной специальности и изменить её:

```
DECLARE
        /* Вых. переменные*/
        studentid students.id%type;
        firstname students.firstname%type;
        lastname students.lastname%type;
        /* Переменные привязки */
        v_proffesion student.profession%type:=’computer science’;
        CURSOR c_student IS
                SELECT id, firstname, lastname
                FROM students
                WHERE profession=v_profession;
```

```
BEGIN
        OPEN c_students;
        LOOP
        /* Выберем каждую строку активного набора в переменные */
        FETCH c_student INTO studentid, firstname, lastname;
        /* Если строки, которые нужно выбрать, закончились, выйдем из цикла */
        EXIT WHEN c_student%NOTFOUND;
                -- курсорные атрибуты
                -- %FOUND
                -- %NOTFOUND
                -- %ISOPEN
                -- %ROWCOUNT
        END LOOP;
/* Освобождаем ресурсы, используемые запросом */
```

ДЗ.

Выполнить с 2 dbms_output (до и после end).

Курсорный атрибут – значение, которое может принимать курсор.

Атрибуты курсора.

FOUND – в результате действия что-то найдено

NOTFOUND – ничего не найдено

ISOPEN – «открыт»

ROWCOUNT – порядковый номер найденной строки

***Пример.**

Вывод номера текущей строки.

```
BEGIN
        IF (c_student%FOUND) THEN
        OPEN item_cur (c_student.studentid); -- неявное определение курсора
        LOOP
                EXIT WHEN item_cur%NOTFOUND;
                dbms_output.put_line( 'On row'||TO_CHAR(item_cur%ROWCOUNT);
        END LOOP;
END;
```

Основные свойства и работа с курсором.

1. Можно определять или не определять тип возвращаемого значения, который может быть указан пользователем. Типы данных: запись (REC), отдельными переменными.
2. Столбцы, определённые в SELECT запросе, должны совпадать с используемой для получения структурой данных.

```
CURSOR c_student IS SELECT
        id, firstname, lastname …
FETCH c_student INTO studentid, firstname …
```

Примеры.

Циклы FOR для курсоров.

Ограничения: открытие, выборка и закрытие курсора происходит автоматически в области видимости цикла FOR. Возвращаемая строка определяется неявно и на неё нельзя сослаться извне области видимости цикла.

```
DECLARE
        CURSOR c_get_table IS
        SELECT * FROM user_tables;
BEGIN
        FOR get_tables_string IN c_get_tables
        LOOP
        dbms_output.put_line(get_tables_string.table_name);
        END LOOP;
END;
```

Результат работы данного курсора – последовательный вывод имён всех таблиц, принадлежащих пользователю.

2 вариант:

```
BEGIN
        FOR get_tables_string IN
        -- определяем переменную строку
        (SELECT * FROM user tables)
        LOOP
        dbms_output.put_line(get_tables_string.table_name);
```

```
        END LOOP;
END;
```

3 вариант:

```
DECLARE
        CURSOR c_get_tables(user_name all_tables.owner%type) IS
        SELECT * FROM user_tables
        WHERE owner=user_name;
        user_name all_tables.owner%type:=&user_name;
BEGIN
        FOR get_tables_string IN c_get_tables(user_name)
        LOOP
        dbms_output.put_line(get_tables_string.table_name);
        END LOOP;
END;
```

Управление курсором.

Через курсорные атрибуты

```
DECLARE
-- предыдущий пример (объявление)+
--определяем переменную записи
get_tables_rec c_get_tables%ROWTYPE; -- присвоить переменной записи
```

get_tables_rec тип строки курсора c_get_tables

```
-- задаём критерий поиска
user_name all_tables.owner%type:=&user_name;

BEGIN
OPEN c_get_tables(user_name);
-- ищем все таблицы пользователя
LOOP
   FETCH c_get_tables INTO get_tables_res;
-- получаем строку
   EXIT WHEN c_get.tables%NOTFOUND;
-- до конца списка
   dbms_output.put_line(get_tables_rec.table_name);
```

```
END LOOP;
CLOSE c_get_tables;
```

Пример курсоров при работе с файлами.

Файл запишется в директорию, описанную в oracle.ini

```
DECLARE
v_nnn NUMBER;
v_name VARCHAR2(10);
v_outfile UTL_FILE.FILE_TYPE;
CURSOR c_temp IS
SELECT nnn, name
FROM srv_temp;
-- Получить данные из курсора и записать в файл
BEGIN
dbms_output.enable;
v_outputfile:=UTL_FILE.FOPEN('c:/temp','report.txt', 'w'); -- Файл открыт на запись (w). Путь должен быть указан в ини файле! Иначе не сработает.
OPEN c_temp;
LOOP
FETCH c_temp INTO v_nnn, v_name;
EXIT WHEN c_temp%NOTFOUND;
dbms_output.put_line(TO_CHAR( v_nnn )|| TO_CHAR( v_name ));
UTL_FILE.PUT_LINE(v_outfile, TO_CHAR( v_nnn )||TO_CHAR( v_name));
END LOOP;
UTL_FILE.FCLOSE(v_outfile);
CLOSE c_temp;
END;
```

Неявные курсоры.

Плохи тем, что их не видно сразу. Используются, если мы ожидаем появление одной строки (если больше – ошибки).

```
DECLARE
get_tables_rec all_tables%ROWTYPE;
user_name all_table.owner%TYPE:=&table_name;
BEGIN
```

SELECT * INTO get_tables_rec FROM all_tables

WHERE owner=user_name AND table_name=user_table;

dbms_output.put_line(gettables_rec.tablespace_name);

END;

Оператор DESC для таблицы all_tables – позволяет определить неявные атрибуты.

Выводы:

1. Курсоры используются там, где необходимо обеспечить управление явным образом SQL операторов.
2. Для задания последовательности обработки наборов данных через курсорные атрибуты.
3. Позволяют проводить построчную обработку записей.

Лекция №18

30.05.07 г

Архитектура Oracle

Физическое размещение элементов БД осуществляется в табличных пространствах, которые находятся на диске в виде файлов БД. На логическом уровне – конкретные БД размещаются в конкретных табличных пространствах.

Имя файла данных – имя табличного пространства+01.dbf.

Замечания:

Копировать просто эти файлы нельзя.

Последовательность действий для создания логической структуры БД.

Экстент – некая группа смешанных блоков Oracle. Это физическая реализация способа хранения данных в Oracle. Т.е. – элемент памяти.

Создание объекта:

1. Пользователь инициирует команду CREATE.
2. НА основе команды в указанной табличной области создаётся необходимый сегмент объекта, который состоит из экстента (достаточного по объёму).
3. Копируется в файл данных.

Обобщенная структура:

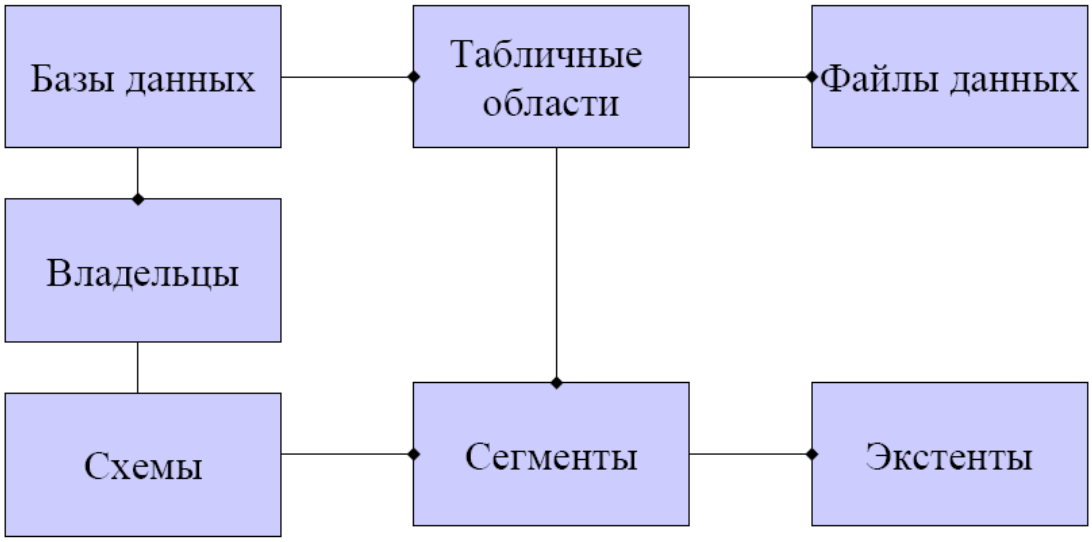


Рис. 18.1

БД имеет владельца объектов и имеет табличные области для хранения объектов. Владельцы имеют свои схемы (как правило одну). На уровне оперативной памяти элементы схемы реализуются в сегментах (куски оперативной памяти). Каждый сегмент состоит из экстентов. Связи с точками: один-ко-многим. Табличные области физически реализуются в виде многих файлов данных.

Табличные пространства .

Оптимальная гибкая архитектура - Optimal Flexible Architecture -OFA

По большому счёту для АСУ создаётся БД, можно и для каждого модуля сделать отдельное табличное пространство.

Основные табличные пространства:

SYSTEM – аналог корневого каталога. В нём хранятся основные схемы (таблицы словаря данных SYS, сегмент отката SYSTEM). В нём можно найти инфу о процедурах, функциях.

DATA – по умолчанию не создаётся. Для хранения таблиц для работающих ваших приложений. Там можно хранить результаты своей работы. Достоинства: можно изменять размеры сегментов и экстентов, что позволяет увеличить скорость доступа.

INDEXES – для хранения индексов.

TOOLS – для хранения объектов сторонних пакетов (от сторонних разработчиков).

RBS – сегмент отката

TEMP – временные сегменты, временное преобразование данных

USERS – хранение объекта пользователя

Требования к оптимальной логической структуре

- однотипно используемые сегменты должны храниться вместе.
- система должна быть спроектирована для типовогоиспользования (без прибамбасов).
- для исключительных ситуаций должны существовать отдельные области
- конкуренция за использование табличных областей должнабыть сведена к минимуму
- словарь данных должен быть изолирован

Приложение 1. Дополнительные конспекты

GRID

Термин GRID вычисления (Computing grid) появился по аналогии с термином Power grid (единая энергосистема). Т. е. его можно перевести как единая компьютерная система.

Часто в связи с концепцией GRID также используют термин “виртуализация”. Действительно, в GRID мы работаем не с множеством мелких компьютеров, а с одним виртуальным суперкомпьютером, не с множеством дисков, на которых лежат наши файлы и базы данных, а с единой виртуальной областью хранения данных (огромным виртуальным диском), которая образуется из множества отдельных дисков.

Итак, с точки зрения пользователя GRID не важно, где размещаются данные и какой компьютер будет обрабатывать его запросы. Главное – это то, что пользователь потребовал информацию или выполнение вычислений и получил результат.

Концепцию GRID описали в своих статьях “Анатомия GRID” и “Физиология GRID” [1, 2] американские ученые Фостер, Кессельман, Ник и Тукке. Они так определили термин Computing Grid в 1998 г.: “Вычислительная Grid – это программно-аппаратная инфраструктура, которая обеспечивает из любого места в мире надежный, согласованный и недорогой доступ к высокоэффективным вычислительным ресурсам”. Отметим слово “недорогой” в этом определении, поскольку появившаяся сегодня возможность использовать в качестве элементов GRID недорогие вычислительные элементы с недорогой операционной системой дала толчок развитию коммерческого использования GRID вычислений.

В 2000 г Фостер и Тукке определили GRID как “Скоординированное разделение ресурсов и решение проблем в динамической, многокомпонентной виртуальной организации ”, где виртуальная организация – это группа предприятий, объединяющих свои вычислительные ресурсы в единую GRID и совместно их использующая.

Однако еще задолго до появления первых работ по GRID один из основоположников интернет Лен Клейнрок (Len Kleinrock) предсказывал в 1969 г. “Мы, возможно, станем свидетелями распространения ‘computer utilities’ (вычислительных коммунальных услуг), которые, также как сегодня телефонные услуги будут доступны во всех домах и офисах по всей стране”. Прошло чуть меньше 30 лет и это предсказание начинает сбываться.

Мы не случайно приводим здесь классические определения концепции GRID. Конечно все вышеописанное – это идеальная картина. Надо различать идеальное понимание термина GRID и его реальную реализацию. Так сегодня невозможно еще создать единый мировой суперкомпьютер, но начать реализовывать эту концепцию в рамках организации уже возможно. Далее мы посмотрим, что из этих идеальных понятий реализуемо уже сегодня, а что может быть реализовано только в далеком будущем.

Если со стороны пользователя GRID все просто (попросил ресурс – получил его), то со стороны организаций, предоставляющих этот единый вычислительный ресурс, необходимо обеспечить ряд требований.

Необходимо обеспечить, что требования на выделение вычислительных ресурсов всегда удовлетворяются, а ресурсы полностью используются, т. е. не должно возникать ситуации, когда пользователь будет ждать выделения ресурса. Еще более сложная задача – это сделать информацию, необходимую для выполнения вычислений, доступной в то время, когда она необходима, и в том месте, где она необходима. Так если речь идет о быстрой переброске огромных баз данных в ту часть света, где есть свободные вычислительные мощности, то сегодня эта задача невыполнима. Скорость и пропускная способность сегодняшних сетей передачи данных этого не позволяет. Но в рамках предприятия и ограниченного числа файлов и баз данных решить эту задачу можно.

Необходимо также обеспечить постоянную доступность и работоспособность системы GRID. Выход из строя отдельных ее элементов не должен останавливать работу приложений. Некоторые решения в этой области, такие как серверный кластер - Real Application Cluster, кластеры серверов приложений, резервные базы данных и т. д. уже сегодня позволяют обеспечить высокую надежность [6].

Основная идея GRID – обеспечить эффективное использование составляющих ее ресурсов. Для этого оборудование и программное обеспечение GRID должно определять загруженность отдельных элементов GRID и балансировать нагрузки, направляя пользователей и приложения на менее загруженные узлы, подключая новые узлы и т. д.

Элементы GRID должны быть дешевыми и простыми, только это позволит оценить экономическую выгоду от внедрения GRID.

Как уже упоминалось выше, создать сегодня мировую коммерческую GRID мы еще не можем. Поэтому выделим три этапа построения GRID.

Самый простой этап – это GRID одного центра обработки данных (ЦОД). ЦОД предприятия уже сегодня может начать объединять свои компьютеры в единую GRID.

Следующим шагом будет объединение различных ЦОД предприятия в единую GRID уровня предприятия. А вот третьим этапом, который наступит не ранее чем через 10

лет, будет объединение GRID предприятий в единую GRID города, страны и т д. Здесь придется решать огромное количество организационных, правовых, финансовых вопросов. Например, вопросы защиты информации и взаиморасчетов между предприятиями могут сильно тормозить эту работу.

Эволюция и стандартизация языка SQL

Стандартизация SQL: Все фирмы провозглашают соответствие стандарту SQL Реализованные диалекты очень близки. Деятельность началась одновременно с появлением первых коммерческих реализаций. В качестве стандарта нельзя было использовать SQL SystemR (не было технической проработки, слишком сложно реализовать). Нельзя было принять за стандарт коммерческий диалект (слишком различались).

Международный стандарт 1989 г.Во многих частях имеет чрезвычайно общий характер и допускает очень широкое толкование Отсутствуют важные разделы (манипулирование схемой БД, динамический SQL, многое определяется в реализации) Наибольшие достижения (стандартизация синтаксиса и семантики операторов выборки и манипулирования данными, фиксация средств ограничений целостности БД: определение первичного и внешнего ключей отношений, проверочные ограничения целостности)

Международный стандарт 1992 г.(SQL2)Расширено манипулирование таблицами (Altertable, Droptable) Манипулирование схемой БД (Users, Tables, Views, Columns, Domains, Table_priveleges, Table_constraints, , ,) Возможность управления доменами (Createdomainимя char(25) . . .и при определении имен столбцов эти имена определяются через имена доменов) Новые типы данных (Date, Time, Datetime, . . .) и новые функции Управление транзакциями и сессиями (сессия -последовательность транзакций, в пределах которой сохраняются временные отношения) Подключение к БД Развитие динамического SQL

Стандарт SQL:1999 (SQL3)Незадолго до завершения работ по определению стандарта SQL2 была начата разработка стандарта SQL3. Реально работу над новым стандартом удалось частично завершить только в 1999 г., и по этой причине (а также в связи с проблемой 2000 года) стандарт получил название SQL:1999.1999 г. были приняты пять первых частей стандарта SQL:1999. Первая часть (SQL/Framework) посвящена описанию концептуальной структуры стандарта: приводится развернутая аннотация следующих четырех частей.

Вторая часть SQL:1999 (SQL/Foundation) образует базис стандарта. Вводится система типов языка, формулируются правила определения функциональных зависимостей и возможных ключей, определяются синтаксис и семантика основных операторов SQL: операторов определения и манипулирования схемой базы данных; операторов манипулирования данными; операторов управления транзакциями (расширенные модели транзакций, контрольные точки, многозвенные транзакций); операторов управления подключениями к базе данных и т. д.

В конце 2003 г. был принят и опубликован новый вариант международного стандарта **SQL:2003**.

Наиболее серьезные изменения языка SQL, специфицированные в части 2 стандарта SQL:2003, касаются следующих аспектов: типы данных; подпрограммы, вызываемые из SQL; расширенные возможности оператора CREATE TABLE; новый объект схемы –генератор последовательностей; новые виды столбцов – идентифицирующие столбцы (identity column) и генерируемые столбцы (generated column); новый оператор MERGE. В SQL:2003 поддерживается механизм табличных функций, т.е. функций, вызываемых из SQL и возвращающих значение-“таблицу”.

Установка Oracle под Linux

Подготовка к установке Oracle

ORAINST-I

Скопировать во временный каталог файлы1) `orainst.pl`, `dbora`.

Сконвертировать все переводы строк в этих файлах в UNIX-стандарт командой `dos2unix *.pl dbora`.

Запустить из этого каталога `perl orainst.pl -i`.

BASEDIRS

Из-под root сделать пользователя oracle и группу oinstall владельцами каталогов, в которых будут храниться файлы БД.

chown -R oracle:oinstall /Base?

ORACLE-INSTALL: Установка Oracle

Подготовка

Выполнить действия, требуемые скриптом `orainst.pl`:

Зайти как `oracle` на другой консоли.

Запустить X Window (`startx`).

Вставить и смонтировать первый диск дистрибутива Oracle 9i.

Не делать `cd` в каталог со смонтированным компакт-диском (!), иначе потом невозможно будет CD-ROM размонтировать.

Для предотвращения ошибки2) при запуске установщика необходимо выполнить скрипты `gccFedora/gccfedora.sh` и `gccFedora/rpm/rpm.sh` (при этом находясь внутри каталогов, в которых они расположены).

Установка

Запустить `/mnt/cdrom/runInstaller` (либо `/media/cdrom/runInstaller`).

Установить Oracle в каталог `/opt/oracle` (создаётся в `orainst.pl`); Inventory будет в `/opt/oracle/oraInventory`.

Unix Group Name: `oinstall`.

Destination: Oracle Home Name - `Ora920`, Path - `/opt/oracle/product/9.2.0`.

Выбрать установку Oracle Database, Enterprise Edition (2.04 GB).

В качестве модели БД выбрать **только** Data Warehouse.

Имя БД и Service: `orcl`.

Файлы баз данных помещать в каталог `/opt/oracle/oradata`.

Кодировку БД выбрать явно: Cyrillic CL8MSWIN1251.

Проблемы при линковке

ins_oemagent.mk: Около 70%

Ошибка "`Error in invoking target install of makefile /opt/oracle/product/9.2.0/network/lib/ins_oemagent.mk`".

Необходимо выполнить следующие действия (**не нажимая кнопок** в диалоге ошибки!):

Заменить файл `/opt/oracle/product/9.2.0/network/lib/ins_oemagent.mk` готовым, исправленным файлом `ins_oemagent.mk`

или

Открыть в редакторе файл `/opt/oracle/product/9.2.0/network/lib/ins_oemagent.mk`.

В строке 18: `LDFLAGS= …` после `$(STDMODE)` добавить параметр `L$(LIBHOME)/stubs`.

Записать файл.

Нажать **Retry** в диалоге ошибки.

env_ctx.mk: Около 85%

В диалоге с сообщением об ошибке "`Error in invoking target install of makefile /opt/oracle/product/9.2.0/ctx/lib/ins_ctx.mk`" нажать кнопку **Ignore**.

Запуск Oracle Enterprise Manager

Для работы с табличными пространствами необходимо открыть Oracle Enterprise Manager (OEM). Он автоматически запускается после выхода из инсталлятора Oracle. Кроме того, OEM может также быть запущен на рабочей станции с установленным Oracle Client'ом или на сервере (из-под X Window) командой `oemapp console`.

С помощью Oracle Enterprise Manager открыть локальный сервер (логин `SYS`, вход в качестве `SYSDBA`), раздел Storage.

Создание табличных пространств

Создать два табличных пространства:

`NEW` в датафайле `/Base1/NEW.dbf`,

`NEW_I` в датафайле `/Base2/NEW_I.dbf` (при наличии только одного диска - `/Base1/NEW_I.dbf`).

Если база будет очень большой и на диске для этого имеется свободное место, можно создать несколько датафайлов с предельным максимальным размером (32767 МБ).

Для датафайлов обоих табличных пространств **включить Autoextend** (автоматическое расширение) с шагом в 1024 мегабайта (либо 2048, 4096).

Минимальный (изначальный) размер для экономии времени можно **оставить по умолчанию (5 МБ)**. Файлы расширятся до нужного размера при первом восстановлении дампа.

Максимальный размер всех датафайлов рабочих табличных пространств, а также временного и откатного (TEMP и UNDOTBS) нужно ограничить так, чтобы при их одновременном расширении до этого размера на разделе осталось ещё несколько гигабайтов свободного пространства.

Следует учитывать, что ограничения на максимальный размер табличных пространств `NEW`, `NEW_I`, `TEMP` и `UNDOTBS` должны быть **не ниже 5 гигабайтов** каждое.

ORAINST-F

Закрыть Enterprise Management Console. Из-под `root` выполнить завершение установки: `perl orainst.pl -f`.

Создание файла настроек БД

INITORCL.ORA

Переместить файл `initorcl.ora` и открыть его для редактирования:

mv /opt/oracle/admin/orcl/pfile/initorcl.ora.* /opt/oracle/product/9.2.0/dbs/initorcl.ora

mcedit /opt/oracle/product/9.2.0/dbs/initorcl.ora

В разделе `[Optimizer]` исправить:

star_transformation_enabled=FALSE

Добавить:

query_rewrite_integrity=TRUSTED

optimizer_index_cost_adj=75

В разделе Pools исправить:

java_pool_size=150000000

shared_pool_size=150000000

В конец файла добавить:

utl_file_dir=’/FiServ’

CREATE-SPFILE

Под пользователем `oracle` выполнить:

cd /opt/oracle/product/9.2.0/dbs

sqlplus "/ as sysdba"

В запустившемся `sqlplus` выполнить следующие команды

shutdown immediate

create spfile from pfile='initorcl.ora';

Должно появиться сообщение “File created”. После этого надо выйти из sqlplus (`quit` или `exit`).

REBOOT

Перегрузить сервер (`reboot`) и проверить запуск Oracle. После перезагрузки инстанция должна запуститься сама, т. е. `ps ax | grep ora_` должен будет показывать процессы `tnslsnr`, `ora_pmon_orcl`, `ora_dbw0_orcl`, `ora_lgwr_orcl`, `ora_ckpt_orcl`, `ora_smon_orcl`, `ora_reco_orcl`, `ora_cjq0_orcl`, `ora_qmn0_orcl`, `ora_s000_orcl`, `ora_d000_orcl`. В случае установки shared_servers также должны присутствовать процессы серверов `ora_sXXX_orcl`.

Установка патчсета

PATCHSET-INSTALL

Для патчсета 9.2.0.6 нужно исправить версию дистрибутива в /etc/redhat-release, иначе установщик патчсета не запустится:

echo 3 > /etc/redhat-release

Для 9.2.0.8 содержимое файла с версией дистрибутива должно[3)] оставаться без изменений, и содержать версию RHEL ES4. Привести его к этому состоянию можно следующей командой:

echo "Red Hat Enterprise Linux ES release 4 (Nahant)" > /etc/redhat-release

После этого:

Остановить инстанцию и листенер: из под `root`: `/etc/init.d/dbora stop`.

Установить новый Oracle Universal Installer из состава патчсета.

Установить патчсет.

PATCHSET-CATPATCH

Выполнить:

cd /opt/oracle/product/9.2.0

sqlplus "/ as sysdba"

В запустившемся `sqlplus` выполнить:

spool patch.sql

startup migrate

@rdbms/admin/catpatch.sql

Дождаться завершения выполнения скрипта (выполняется более получаса) и проверить отображаемые в консоли результаты установки. Версии установленных продуктов должны соответствовать версии патчсета, состояние должно быть `VALID`, `UPGRADED` либо `OPTION OFF`.

Всё ещё находясь в `sqlplus`, выполнить:

spool off

shutdown

startup

Выйти из sqlplus.

PATCHSET-LOWMEM

Если на сервере менее 1 гигабайта ОЗУ: Отредактировать файл `/opt/oracle/product/9.2.0/dbs/initorcl.ora`:

В разделе Pools:

java_pool_size=50000000 [исправить существующий]

shared_pool_size=50000000 [исправить существующий]

Повторить шаг CREATE-SPFILE.

LOADJAVA

Из-под пользователя oracle выполнить:

loadjava -user sys/sys -r -s -g public /opt/oracle/product/9.2.0/jlib/regexp.jar

DBSHUT-FIX

Подправить скрипт завершения работы БД `dbshut` на использование команды немедленного отключения (`shutdown immediate`).4)

Под пользователем `oracle`:

vi `which dbshut`

Все (два) вхождения `shutdown` (кроме строки `sqldba command=shutdown`) заменить на `shutdown immediate`, либо скопировать готовый (уже исправленный) файл `dbshut` в каталог `/opt/oracle/product/9.2.0/bin/`.

Настройка резервного копирования

BKUP.RPM

Установка RPM-пакета со скриптом backup'а. Выполнить:

rpm -i <путь_к_файлу_bkup-*.rpm>

BKUP.CONF

Скопировать файл по умолчанию `/etc/bkup.conf.sample` в `/etc/bkup.conf` и отредактировать его.

BKUP-CRON

Настройка ночного резервного копирования. В /etc/crontab добавить строку:

00 22 * * * root perl /opt/bkup.pl -b

Можно добавить её командой:

echo "00 22 * * * root perl /opt/bkup.pl -b" >> /etc/crontab

DUMP

Восстановить дамп. Находясь в каталоге с файлом дампа выполнить:

/opt/bkup.pl –rf <имя файла>

Проверить результат закачки - в основном каталоге архивных копий, файл `bkup_import_*.log`.

BKUP-TEST

Проверка резервного копирования:

Выполнить `perl /opt/bkup.pl -b`.

Посмотреть протокол `/var/log/bkup.log`.

Проверить наличие файлов резервных копий в заданных в `bkup.conf` каталогах.

1) Архив с этими и другими необходимыми при установке БД iSZN под Linux файлами находится по адресу http://www.ites.ru/products/files/iszn/ITES-SetupFilesForLinux.exe (13 МБ).

2) "Unable to load native library... libjava.so: symbol __libc_wait, version GLIBC_2.0 not defined in file libc.so.6..."

3) Иначе при запуске установщика патчсета произойдёт ошибка "Segmentation Fault"

4) Если этого не сделать, при выключении или перезагрузке сервер сначала будет ждать отключения всех подключенных к нему клиентов, и только после этого сможет отключиться или перезагрузиться.

Управление сеансами и сессиями. Методы POST и GET

Сессия-метод сохранения состояния между HTTP-транзакциями

Переменные сессии не могут быть изменены при помощи GET и POST

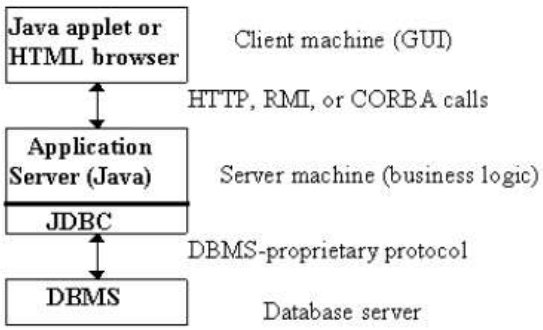
cookie-набор-файл с данными, который сохраняется на пользовательской машине

- session_start() -запуск сессии
- session_register (“var1”, “var2”) -регистрация переменной
- session_is_registred (“var”) -проверка регистрации переменной
- session_unregister(“var”) -отмена регистрации переменной
- session_unset() -отмена регистрации всех переменных
- session_destroy() -уничтожение сессии

Метод GET По умолчанию при запросе используется метод GET. Метод POST используется только тогда, когда это явно указано в запросе формы. Для CGI-программиста очень важно понимать, что при запросе методом GET данные формы передаются серверу вместе с URL. Web-серверы, поддерживающие CGI, копируют эти данные в переменную окружения с именем QUERY_STRING. После этого забота о получении данных из переменной окружения и их обработке возлагается на CGI-программу. Знак ? отделяет строку запроса от собственно URL ресурса

Метод POST Метод POST используется тогда, когда это явно указано в атрибуте формы METHOD. В отличии от метода GET, POST помещает данные не в URL, а в тело запроса. Запрос POST во многом похож на ответ HTTP. Первая строка представляет собой стандартный запрос HTTP, в котором указан метод POST. В ней могут быть необходимые дополнительные заголовки, отделяемые от тела запроса пустой строкой. Тело запроса при использовании метода POST передается программе как стандартный поток ввода.

Java



Общение программ на Javac данными в БД под управлением Oracleo существляется двумя основными способами: через **JDBC** и через **SQLJ**

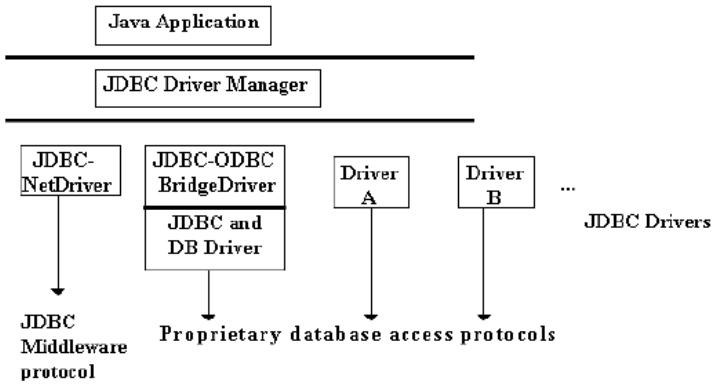
JDBC (Java Database Connectivity) является не протоколом, а интерфейсом и основан на спецификациях SAG CLI (SQL Access Group Call Level Interface-интерфейс уровня вызова группы доступа SQL).

В отличие от интерфейса ODBC, JDBC организован намного проще. Главной его частью является драйвер, поставляемый фирмой JavaSoft для доступа из JDBC к источникам данных. Этот драйвер является самым верхним в иерархии классов JDBC и называется DriverManager. Согласно, установившимся правилам Internet, база данных и средства ее обслуживания идентифицируются при помощи URL.

Фирма Oracle поставляет для работы со своей СУБД следующие драйверы, удовлетворяющие спецификациям JDBC:

- тонкий (thin; тип IV, для работы извне, через браузер, по TCP/IP)
- толстый (thick; тип II, для локальной работы извне)
- родной (тип II, для работы изнутри, из хранимых в БД Java-процедур)

Помимо этого около сотни разных фирм поставляют JDBC-драйверы собственной реализации типов I, II, III и IV, в том числе и для связи с Oracle. Они доступны в интернете.



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

Пакет используется для логический группировки программных единиц Java.

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

Для построения web-приложений на основе Java часто используют подход («модель») Model-View-Controller(MVC), предложенный фирмой Xerox в 80-х годах для своего языка Smalltalk-80. Согласно этому подходу приложение представляет собой в общем случае организованный набор страниц HTML и JSP, а также сервлетов и компонент JavaBeans и EnterpriseJavaBeans(EJB). Задачи, которые решают эти элементы web-приложения, разделяются на задачи моделирования прикладной области (Model), представления данных приложения клиенту (View) и управления работой приложения (Controller).

Сервлет (servlet) Java – это серверная программа, вызываемая web-сервером для обслуживания HTTP-запросов.

Преимущества сервлетов по сравнению с другими методами серверного программирования:

- При каждом вызове сервлета не создается новый процесс, сервлеты вызываются посредством потоков, обслуживающих отдельные HTTP-запросы.
- Это обычные программы Java, транслируемые в не зависящий от платформы байтовый код, поэтому их можно автоматически переносить на любую машину, где применяется Java. Они работают в области, защищенной

границами виртуальной машины Java и не могут быть причиной нарушения правил доступа к памяти и вызывать аварии сервера.

- Серверы имеют полный доступ к мощным прикладным программным интерфейсам Java, таким как EJB, Java Mail, RMI и др.

***Пример. Написание простого сервлета**

```
/** Имя программы: HelloServlet.java
** Назначение: Вывод приветствия и текущего времени
**/
importjavax.servlet.*; // (1)
importjavax.servlet.http.*;
importjava.io.PrintWriter;
importjava.io.IOException;
publicclassHelloServletextendsHttpServlet{ // (2)
publicvoiddoGet(HttpServletRequestrequest,
HttpServletResponseresponse) // (3)
throwsServletException, IOException
{
PrintWriterout= response.getWriter(); // (4)
response.setContentType("text/html"); // (5)
out.println("<HTML>"); // (6)
out.println("<HEAD><TITLE>TheHelloServlet</TITLE></HEAD>");
out.println("<BODY BGCOLOR=\"white\">");
out.println("<H2>Hello" + request.getParameter("user") +
", howareyou?</H2>"); // (7)
out.println("<P><B>Thecurrenttimeis" + newjava.util.Date());
// (8)
out.println("<P>Hopeyouhavea niceday! </B>");
out.println("</BODY>");
out.println("</HTML>");
out.close(); // (9)
}
}
```

***Пример. JSP**

```
<!--имя программы: WelcomeUser.jsp
--Назначение: Поприветствовать пользователя и
--вывести текущее время.
-->
<HTML>
<HEAD>
<TITLE>The WelcomeUserJSP</TITLE>
</HEAD>
<BODY>
<H2>Привет,<%= request.getParameter("user") %>, как дела?</H2>
<P><B> Текущее время <% new java.util.Date() %>.
</BODY>
</HTML>
```

JSP удобнее сервлетов по следующим причинам:

- Сокращенный синтаксис для смешивания статических конструкций с динамической логикой
- Программный текст Java четко выделяется в таких компонентах, как JavaBeans и библиотеки тегов JSP.
- Поддерживается модель быстрой разработки RAD (rapid application development), где исходный JSP-файл автоматически преобразуется двигателем JSP.

XML + Oracle

Oracle Database начиная с версии 9i – это, пожалуй, единственная РСУБД, которая полноценно поддерживает хранение документов XML и поиск по ним. Для этого в ней имеется встроенный тип XMLType (он основан на встроенном типе CLOB), который позволяет хранить документы и осуществлять поиск по ним посредством запросов Xpath

Схема БД для хранения XML документов

Схемы реляционной БД для хранения иерархических структур общеизвестны. Вот один из таких примеров:

```
CREATE TABLE XMLnodes
(
```

DocId INTEGER,

NodeName CHAR(20),

NodeValue VARCHAR(250),

NodeType INTEGER NOT NULL,

Left INTEGER NOT NULL,

Right INTEGER NOT NULL

);

DocId – каждому документу присваивается уникальный идентификатор.

NodeType – тип узла: элемент, атрибут, текст, комментарий, инструкция обработки, пространство имен и корневой узел.

NodeName – имя узла: имя атрибута для атрибута, имя элемента для элемента, получатель для инструкции обработки, префикс для пространства имен, NULL для текста, комментария, корневого узла).

NodeValue – текстовое значение узла; NULL для корневого узла и элементов. Пространство имен представляется узлом, родителем которого является узел элемента, в котором оно объявлено.

Последние два поля Left и Right (а также NodeId) предназначены для сохранения иерархической структуры XML-документа.

Ограничения синтаксиса XML:

- **Все элементы XML должны иметь закрывающий тег**
- Составляя XML документ, вы **не можете опускать закрывающие теги.**
- В отличие от HTML, теги в XML **чувствительны к регистру.**
- Открывающий и закрывающий теги должны быть написаны **одинаково с учетом регистра.**
- **XML элементы должны быть строго вложенными**
- Все XML документы должны содержать **единственную пару тегов, определяющую корневой**
- элемент.
- Все остальные элементы должны быть потомками *этого корневого элемента.*
- В XML считается **ошибкой, если вы напишите значение параметра, не заключив его** в кавычки.

Архитектура Oracle. Управление памятью

Экземпляром называется комплекс структур памяти и процессов операционной системы.

Процесс-задача, выполняемая без вмешательства пользователя.

Свойства:

- имеют отдельный блок памяти, для хранения локальных переменных, стека адресов и др. информации
- используют разделяемую область памяти для доступа к данным общего пользования.

Разделяемая область памяти называется: **SGA -System Global Area**

Если отдельные функции организма - это фоновые процессы, то SGA - это мозг, где запоминается и передается информация от одних органов -другим. SGA-принимает участие во всех процессах обработки информации в системе.

Алгоритм инициализации Экземпляра:

1. Считывается файл параметров init.ora (размер структур памяти, количество и тип фоновых процессов).См. oracle_home/…/init.ora (имя экземпляра, как правило, соответствует переменной ORACLE_SID).

2. Запускаются фоновые процессы.

3. Инициализируется SGA.

Установочная стадия:

1.Значения параметров контрольного файла из init.ora определяют параметры БД.

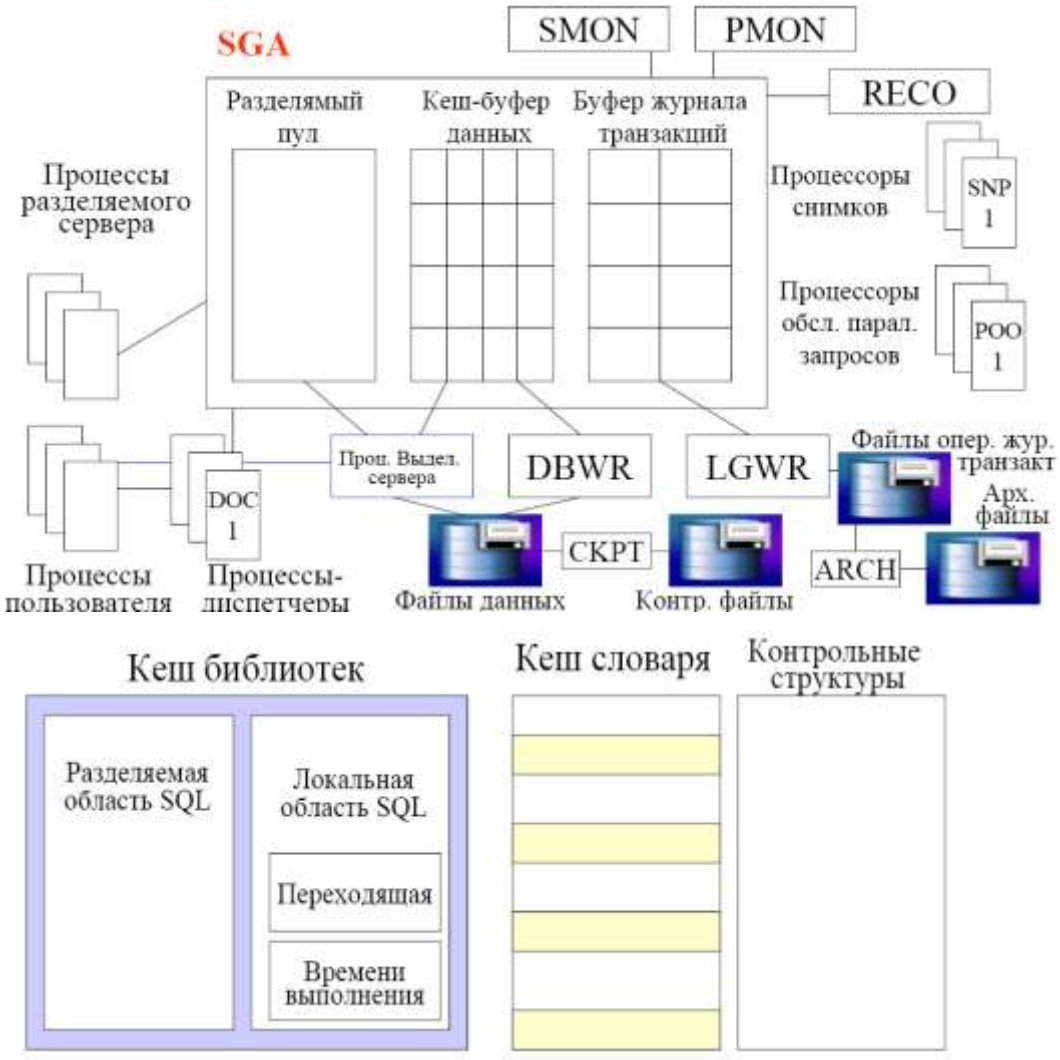
Инициализация:

1. Отрывается БД, экземпляр получает доступ к файлам БД, имена которых хранятся в контрольном файле.

Незанятый экземпляр (idle)-это экземпляр в котором не создана база данных. Он занимает память, но не выполняет никакой работы.

Экземпляр может подсоединиться только к одной БД до тех пор, пока не будет использован Parallel Server. База данныхможет быть подключена только к одному экземпляру Oracle.

Настройка экземпляра производится путем внесения необходимых изменений в значения параметров в файле init.ora.(см. структуру файла init.ora в Enterprise Manager). Назначения параметров представлены в системной таблице **X$KSPPI**.



Размер разделяемого пула задается параметром: SHARED_POL_SIZE в файле init.ora в байтах.

Очистка пула: **ALTER SYSTEM FLUSH SHARED_POOL**

Кеш библиотек - текст, формат лексического анализатора и план выполнения SQL, содержит заголовки пакетов и процедур, выполнявшихся ранее.

Предназначен: для повышения скорости выполнения операторов SQL.(требования к исходному коду -выражения SQL должны быть полностью идентичны, вплоть до регистров символов).

Разделяемая область включает дерево лексического анализа и план выполнения SQL-выражения.

Локальная – информацию по текущему сеансу работы (переменные параметры окружения, стеки, буфера и т.п.).

Пример: дерево лексического анализа SQL операторов в EM.

Свойства:

- Локальные области формируются для каждой инициируемой транзакции и закрывается, после закрытия соотв. курсора.
- Количество локальных областей, которые можно одновременно открыть определяются параметром OPEN_CURSORS в файле init.ora.

Локальная область содержит: **переходящую**(информацию, сохраняющую свое значение, которое может быть использовано несколькими SQL), **область времени выполнения**(информацию только для выражения, выполняемого в текущий момент).

Кеш словаря - хранит строки словаря данных, которые были использованы для лексического анализа предложений SQL.В этой области хранятся данные , касающиеся сегментирования, привилегий доступа и размера свободной памяти.

Кеш-буфер данных:

- Задает производительность системы.
- Состоит из блоков данных того же размера, что и блоки Oracle.

OLTP приложение – Кеш буфер –Swap – данных файл данных // Механизм LRU

Блок данных

Настройка - рациональный выбор объема:

DB_BLOCK_SIZE -задает размер блока Oracle

DB_BLOCK_BUFFERS -задает количество блоков,выделяемых для кеш-буфера данных.

Общий объем кеш-буфера данных (в байтах) является произведением: **DB_BLOCK_SIZE * DB_BLOCK_BUFFERS.**

Буфер журнала транзакций

Расположен между

Настройка - размер задается параметром LOG_BUFFER в байтах.

Оперативный контроль за процессом записи осуществляется с помощью представления: **V$SYSSTAT.**

Количество транзакций, данные о которых заносятся в журнал одновременно ограничиваются параметром **LOG_SMALL_ENTRY_MAX_SIZE**, который задается в байтах.

Мониторинг работы с буфером журнала транзакций может выполняться при помощи представления: **V$LATCH.**

Фоновые процессы Oracle

PMON-Process Monitor-выполняет автоматическую уборку внезапно прекратившихся или завершившихся аварийно процессов, что предусматривает удаление сеанса, удаление блокировок и не принятых транзакций процесса, освобождение ресурсов SGA.

Он также следит за процессами сервера и диспетчерами автоматически перезапускает их в случае останова.

SMON -выполняет автоматическое восстановление экземпляра запуска БД. Автоматически выполняет незавершенные операции, незавершенные до последнего выключения БД. Следит за сегментами БД, фиксирует освобождение пространства во временных сегментах и автоматически объединяет их в непрерывные свободные блоки в файлах данных.

DBWR-DatabaseWriter-отвечает за перенос обновленных блоков их кеш-буфера в файлы данных.Процесс выполняет запись на диск в следующих случаях: обнаружения контрольной точки, достижения заданного количество элементов кеширования, достижение максимального числа буферов Кеша BD_BLOCK_MAX_SCAN.-по истечении заданного для процесса DBRW интервала времени (3 с).

OracleDBWR-DatabaseWriter-отвечает за перенос обновленных блоков их кеш-буфера в файлы данных.

Настройки:

- число процессов DBRW рекомендуется брать равным числу файлов данных БД или число физических дисков, используемых для хранения файлов данных.
- число защелок (LRU) не должно превышать количества процессов более чем вдвое.
- DB_BLOCK_CHECKPOINT_BATCH -чем больше, тем реже DBRW перезаписывает данные.

LGWR-Log Writer-отвечает за перепись информации из буфера журнала транзакций в файлы оперативного журнала.

Условия выполнения:

- транзакция принимается.
- истекает время ожидания процесса LGWR.
- буфер журнала транзакций заполняется на треть.
- DBRW завершает перезапись данных из Кеш - буфера после обнаружения контрольной точки.

Процессы диспетчеры

Dnnn-осуществляет управление разделяемыми процессами. **Разделяемые процессы -** это процессы используемые различными пользовательскими процессами совместно. Для использования разделяемых процессов необходимо сконфигурировать многопоточный сервер MTS -Multi-Threaded Server.

Число диспетчеров определяется параметром: MTS_DISPATCHES = “tsp, 4”.

Другие процессы

ARCH-отвечает за копирование полностью заполненного оперативного журнала транзакций в архивные файлы журналов транзакций.

CKPT-дополнительный фоновый процесс, отвечающий за обработку контрольных точек.

RECO-отвечает за восстановление (recovery) незавершенных транзакций в распределенной системе Бд.

SNPn-выполняет автоматическое обновление снимков (snapshot) БД и запускает процедуры в соответствии с расписанием, зафиксированным в пакете DBMS_JOB.

LCKn-Координация блокировок в режиме с параллельным обслуживанием одной БД множества экземпляров.

Pnnn-процессы параллельных запросов. Каждый процесс пользователя подключается к процессу сервера, который может быть либо жестко связан с одним процессом пользователя, либо распределяться между многими.

Каждому процессу пользователя выделяется область памяти: глобальная область процесса -PGA (Process Global Area).

Приложение 2. Справочные таблицы

Типы обрабатываемых данных Oracle

Тип данных	Описани
CHAR(size), CHARACTER (size)	Символьная строка фиксированной длины, имеющая максимальную длину size символов. Длина по умолчанию 1, максимальная -255.
VARCHAR2(size), VARCHAR(size)	Символьная строка переменной длины, имеющая максимальную длину
LONG	Символьные данные переменной длины до 2 Гигабайт.
LONG RAW	Двоичные данные переменной длины вплоть до 2 Гигабайт.
CLOB	Character Long
BLOB	Binary Long
NUMBER(p,s)	Число, имеющее p значащих цифр и масштаб s. p может быть от 1 до 38. s может принимать значения от -84 до 127.
RAW(size)	Двоичные данные длиной size байт. Максимальное значение для size -
MLSLABEL	Используется в Trusted ORACLE.
RAW MLSLABEL	
ROWID	Значения псевдостолбца ROWID.

Псевдостолбцы

Псевдостолбец	Возвращаемое
Sequence.CURRVAL	Текущее значение sequence в данном сеансе (после
Sequence.NEXTVAL	Следующее значение sequence в текущем сеансе.
[table.]LEVEL	1 - для корня дерева, 2 - для узлов второго уровня и так далее. Используется в операторе SELECT в иерархических запросах.
[table.]ROWID	Значение типа ROWID, которое идентифицируют строку в таблице table уникальным образом.
ROWNUM	Порядковый номер строки среди других строк, выбираемых запросом. ORACLE выбирает строки в произвольном порядке и приписывает значения ROWNUM, прежде чем строки будут отсортированы

Зарезервированные слова SQL

ACCESS*	DEFAULT*	INTEGER	OPTION*	START*
ADD*	DELETE*	INTERSECT*	OR*	SUCCESSFUL
ALL*	DESC*	INTO*	ORDER*	SYNONYM
ALTER*	DISTINCT*	IS*	PCTFREE*	SYSDATE
AND*	DROP*	LEVEL*	PRIOR*	TABLE*
ANY*	ELSE*	LIKE*	PRIVILEGES	THEN*
AS*	EXCLUSIVE	LOCK	PUBLIC*	TO*
ASC*	EXISTS*	LONG	RAW	TRIGGER
AUDIT	FILE	MAXEXTENTS	RENAME*	LTD

BETWEEN*	FLOAT	MINUS*	RESOURCE*	UNION*
BY*	FOR*	MODE	REVOKE	UNIQUE*
CHAR*	FROM*	MODIFY	ROW	UPDATE*
CHECK*	GRANT*	NOAUDIT	ROWID	USER
OCLUSTER*	GROUP*	NOCOMPRESS*	ROWLABEL	VALIDATE
COLUMN	HAVING*	NOT*	ROWNUM*	VALUES*
COMMENT	IDENTIFIED*	NOWAIT	ROWS	VARCHAR*
COMPRESS*	IMMEDIATE	NULL*	SELECT*	VARCHAR2*
CONNECT*	IN*	NUMBER*	SESSION	VIEW*
CREATE*	INCREMENT	OF*	SET*	WHENEVER
CURRENT*	INDEX*	OFFLINE	SHARE	WHERE*
DATE*	INITIAL	ON*	SIZE*	WITH*
DECIMAL*	INSERT*	ONLINE	SMALLINT	

Операции над множествами

Операция	**Выполняемые функции**
UNION	Комбинирует два запроса: возвращает все неповторяющиеся строки, извлеченные хотя бы одним из запросов.
UNION ALL	Комбинирует два запроса: возвращает все строки. извлеченные хотя бы одним из запросов, включая повторяющиеся.
INTERSECT	Комбинирует два запроса: возвращает все неповторяющиеся строки, извлеченные каждым из запросов.
MINUS	Комбинирует два запроса: возвращает все неповторяющиеся строки, извлеченные первым запросом. но не извлеченные вторым.
(+)	Указывает, что предшествующий столбец является столбцом внешнего соединения.
*	Используется вместо имен столбцов при выборке всех столбцов из таблицы или представления.
PRIOR	Используется в иерархическом древовидном запросе для определения зависимости между родительскими и дочерними строками. Смотрити оператор SELECT.
ALL	Оставляет повторяющиеся строки в результате запроса (установлен по умолчанию ALL. но не DISTINCT).
DISTINCT	Удаляет повторяющиеся строки из результата запроса.

Операторы редактирования команд SQL

Команда	**Сокращенный формат**	**Назначение**
APPEND текст	A текст	Добавляет текст в конец строки
CHAGE старый/новый	C старый/новый	Заменяет старый текст в строке новым
CHANGE /текст	C /текст	Удаляет текст из строки
CLEAR BUFFER	CL BUFF	Удаляет все строки
DEL	(Нет)	Удаляет текущую строку из буфера
INPUT	I	Добавляет одну или несколько строк в буфер
INPUT текст	I текст	Добавляет заданную текстовую строку
LIST	L	Выводит содержимое буфера SQL*Plus
LIST n	Ln	Выводит строку n
LIST*	L*	Выводит текущую строку
LIST m n	L m n	Выводит строки от m до n
LIST LAST	L LAST	Выводит последнюю строку буфера

Вспомогательные команды SQL*Plus

Команда	**Описание**
RUNFORM имя файла	Запуск приложения, написанного на Oracle Forms, из SQL*Plus.
SPOOL имя файла	Инициализирует запись всего, что появляется на экране в течение сессии, в текстовый файл.
SPO[OL] OFF|OUT|ON	OFF отключает выдачу протокола работы в файл. ON возобновляет выдачу в тот же файл. OUT - задает выдачу протокола на системный принтер.
DESC[RIBE] таблица	Высвечивает на экране структуру таблицы.
HELP	Вызов системной помощи.
HOST команда ОС	Выполняет команду операционной системы из SQL*Plus
CONNECT	Вход в Oracle под другим именем/паролем.

имя/пароль	
PROMPT текст	Выдает сообщение при работе командного файла.

Команды установки среды SQL*Plus (SET)

Команда	**Описание**
ECHO (OFF ON)	ON задает выдачу на экран текста команд, выполняющихся в командном файле. OFF отключает высвечивание текста команд.
FEED[BACK] (6n OFF ON)	и - задает команду на выдачу сообщения о количестве найденных по SQL-запросу строк, начиная с п найденных строк. ON и OFF включают и выключают выдачу данного сообщения. Задание ON эквивалентно заданию n=1. Задание n=0 эквивалентно заданию OFF.
HEA[DING] (OFF ON)	ON - задает выдачу имен столбцов таблиц результатов. OFF отключает выдачу.
LIN[ESIZE] (80n)	Задает ширину выходного потока в символах, т.е. сколько символов может быть помещено на одну строку. На основании этого значения также определяется правый край для выравненных вправо и середина для центрированных строк. Максимальное значение п равно 500.
NEWP[AGE](ln)	Устанавливает количество пустых строк между двумя высвечиваемыми на экране страницами. Значение 0 задает команду на постановку символа «form feed» между страницами. При печати на принтере это задает переход на новый лист: для большинства видеотерминалов это означает очистку экрана и высвечивание страницы с первой строки.
NUMF[ORMAT] текст	Задает формат по умолчанию для чисел в таблицах результатов. В качестве текста должна использоваться маска форматирования. Допустимые элементы масок рассмотрены в описании опции FORMAT команды COLUMN ниже в этом разделе.
NUM[WIDTHl (10n)	Задает ширину столбцов числового типа (NUMBER) в таблицах результатов.
PAGES[IZE] (24n)	Определяет количество строк на одной странице. Для отчетов, печатаемых на страницах высотой в 11 дюймов (27.5 см) значение 54 (в совокупности с NEWPAGE=6) оставляет по одному дюйму (2,5 см) сверху и снизу страницы.
PAU[SE] (OFF ON текст)	ON устанавливает режим ожидания перед выдачей очередной страницы на экран до нажатия оператором клавиши [Return]. OFF отключает режим ожидания. Если вы хотите высвечивать на экране подсказку в момент ожидания (типа "нажмите Return хля продолжения"), то задайте соответствующий текст. Замечание: если текст содержит пробелы, то он должен быть взят в апострофы. Текст, содержащий апостроф, может быть взят в двойные кавычки: это два равноправных разделителя, и какой разделитель стоит сразу после слова PAUSE, тот и трактуется системой в качестве разделителя.

VER[IFY] (OFF ON)	Параметр ON устанавливает режим выдачи на экран строк, содержащих переменные подстановки, до и после подстановки фактических значений. OFF подавляет выдачу.
TIMI[NG] (OFF ON)	ON задает режим выдачи времени выполнения каждой SQL-команды сразу после выдачи результирующей таблицы. OFF отключает режим.
SPA[CE]{ln}	Определяет количество пробелов между столбцами таблиц результатов. Максимальное значение п равно 10.
TERM[OUT] (OFF ON)	OFF отключает выдачу сообщений на экран терминала при выполнении командного файла. Таким образом, можно выполнять набор команд по изменению содержимого БД или печати результатов в файл без появления сообщений на экране. ON включает режим выдачи на экран.
SQLCASE (MIXED/LOWER/UPPER)	LOWER/UPPER задает режим перевода текста вводимых оператором команд на нижний/верхний регистр. Перевод не отражается на экране, а производится перед выполнением команды и обрабатывает весь текст, включая символьные литералы. MIXED (по умолчанию) оставляет текст таким, каким он был введен оператором.

Числовые функции

Функция	Возвращаемое значение
ABS(n)	Абсолютное значение величины n.
CEIL(n)	Наименьшее целое, большее или равное n,
COS(n)	Косинус л (угла, выраженного в радианах).
COSH(n)	Гиперболический косинус п.
EXP(n)	е в степени n.
FLOOR(n)	Наибольшее целое, меньшее или рапное n.
LN(n)	Натуральный логарифм n, где n>0.
LOG(m.n)	Логарифм n по основанию m.
MOD(m.n;	Остаток от деления m на n.
POWER(w.n)	W в степени n.
ROUND(n[.m])	n, округленное до m позиций после десятичной точки. По умолчанию m равно нулю.
SIGN(n)	Если n<0.-1:если n=0. 0: если n>0. 1.,
SIN(n)	Синус n (угла, выраженного в радианах).
SINHM	Гиперболический синус.
SQRT(n)	Квадратный корень от n. Если n<0. возвращает значение NULL.

TAN(n)	Тангенс n (угла, выраженного в радианах).
TANH(n)	Гиперболический тангенс n.
TRUNC(n[.m])	n, усеченное до m позиций после от десятичной точки По умолчанию m равно нулю.

Символьные функции, возвращающие символьные значения

Функция	Возвращаемое значение
CHR(n)	Символ с кодом n.
CONCAT(charl,cAflr2j	Конкатенация символьных строк char1 и char2.
INITCAP(char)	Символьная строка char, первые буквы всех слов в которой преобразованы в прописные.
LOWER(char)	Символьная строка char, осе буквы которой преобразованы D строчные.
LPAD(charl.n [,char2})	Символьная строка churl, которая дополняется слева последовательностью символов из charl так, чтобы общая длина строки стала равна п. Значение charl по умолчанию -" (один пробел). Если часть многобайтового символа не помешается в добавляемой строке, то конец строки заполняется пробелами.
LTRIM(char[.set])	Символьная строка char, в которой удалены все символы от начала вплоть до первого символа, которого нет в строке set. Значение set по умолчанию -" (один пробел).
NLS_INITCAP(char[.nls_son])	Символьная строка char, в которой первые буквы всех слов преобразованы в прописные. Параметр nIs_sort определяет последовательность сортировки.
NLS_LOWER(char[.nls_sort])	Символьная строка char, все буквы которой преобразованы в строчные. Параметр tils-sort определяет последовательность сортировки.
NLS_UPPER(cliar[.nls_sort])	Символьная строка char, все буквы которой преобразованы в прописные. Параметр ntssort определяет последовательность сортировки.
REPLACE(char. search_string [.replacement_string])	Символьная строка char, в которой все фрагменты search_string заменены на replacement_string. Если параметр replacement string не определен, все фрагменты search-string удаляются.

RPADfcharl. n[, char2J)	Символьная строка charl, которая дополнена справа последовательностью символов из charl так. что общая длина строки равна п. Если часть многобайтового символа не помещается в добавляемой строке, то конец строки заполняется пробелами.
RTRIM(char[.set])	Символьная строка char, в которой удалены все символы справа вплоть до первого символа, которого нет в строке set. Значение параметра set по умолчанию - (один пробел).
SOUNDEX(char)	Символьная строка, содержащая фонетическое представление для char, на английском языке.
SUBSTR(cliar.m[.n])	Фрагмент символьной строки char, начинающийся с символа т, длиной n символов (до конца строки, если параметр n не указан).
SUBSTRB(char.m[.n])	Фрагмент символьной строки char, начинающийся с символа m, длиной л байтов (до конца строки, если параметр п не указан).
TRANSLATE(char.from. to)	Символьная строка char, в которой все символы, встречающиеся в строке from, заменены на соответствующие символы из to.
UPPER(char)	Символьная строка char, в которой все буквы преобразованы в прописные.

Символьные функции, возвращающие числовые значения

Функция	Возвращаемое значение
ASCII(char)	Возвращает десятичный код первого символа строки char в кодировке, принятой в базе данных. (Код ASCII в системах, использующих кодировку ASCII). Возвращает значение первого байта многобайтового символа.
INSTRfcharl. char2[, n[, m]])	Позиция первого символа m-ого фрагмента строки charl, совпадающего со строкой char2, начиная с n-ого символа. По умолчанию n и m равны 1. Номер символа отсчитывается от первого символа строки charl, даже когда n> 1.
TNSTRBfcharl. char2[,n[, m]])	Позиция первого символа m-ого фрагмента строки charl, совпадающего со строкой charl, начиная с m-ого байта. По умолчанию n и m равны 1. Номер байта отсчитывается от первого символа строки charl, даже когда n> 1.
LENGTH(char)	Длина строки char в символах.

LENGTHB(char)	Длина строки char в байтах.
NLSSORT(charl,char2[,n[,m]])	Зависящее от национального языка значение, используемое при сортировке строки char.

Функции группировки

Функция	**Возвращаемое значение**
AVG ([DISTIXCT/ALL]n)	Среднее значение группы полей в столбце, не включая пустых значений
COUNT ([DISTINC/ALL] выражение)	Количество строк в группе, в которых имеет непустое значение
MAX ([DISTINCT/ALL] выражение)	Максимальное значение выражения в группе
MIX ([DISTIXCT/ALL] выражение)	Минимальное значение выражения в группе
STDDEV ([DISTIXCT/ALL] u)	Функция, определяющая математическое ожидание в группе
SUM ([DISTIXCT/ALL] n)	Сумма значений, игнорируя пустые значения
VARIANCE ([DISTIXCT/ALL] n)	Дисперсия в группе.

Функции работы с датами

Функция	**Возвращаемое значение**
ADD-MONTHS (d.n)	Дата d плюс п месяцев.
LAST-DAY (d)	Последнее число месяца, указанного в d
MONTHS-BETWEEN (d.e)	Число месяцев между датами dl и d2.
NEW-TIME (d.a.b)	Дата и время в часовом поясе a. соответствующие дате и времени в часовом поясе b, при этом d,a и b значения типа CHAR, определяющие часовые пояса.
NEW-DAY (d,char)	Дата первого после даты (дня недели, название которого записано в char).
SYSDATE	Текущая дата и время.
ROUND(d[.fmt])	Дата d, округленная до единиц, указанных в форматной маске.
TRUNC(d[.fmt])	Дата d, усеченная по форматной маске fmt.

Форматные маски дат в TO_CHAR и TO_DATE

Элемент формата	Возвращаемое значение
SCC или CC	Столетие: если указано 'S' то перед датами до нашей эры ставится '-‘.
YYYY или SYYYY	Год; если указано ‘S' то перед датами до нашей эры ставится '-‘. YYY или YY или Y] Последние 3. 2. или1 цифра года.
IYYY	4 цифры года по стандарту ISO. IYY или IY или I] Последние 3, 2, или1 цифра года по стандарту ISO.
Y.YYY	Год с запятой в указанной позиции.
SYEAR или YEAR	Год. записанный словами, а не цифрами: если указано ‘S' то перед датами до нашей эры ставится '-'.
RR	Последние 2 цифры года: для указания года в других столетиях.
BC или AD	BC- до нашей эры (до н.э.); AD - нашей эры
B.C. илиA-D.	B.C.- до нашей эры (до н.э.): A.D. - нашей эры
Q	Квартал (1. 2. 3. 4:JAN-MAR=1).
MM	Месяц(01-12; JAN=1).
RM	Нумерация месяцев римскими цифрами(1-XII: JAN=I).
MONTH	Название месяца, дополненное пробелами до 9-ти символов.
MON	Сокращенное название месяца.
WW или W	Неделя года (1-52) или месяца (1-5).
IW	Неделя года (1-52 или 1-53) по стандарту ISO.
DDD или DD или D	День года (1-366) или месяца (1-31) или недели (1-7).
DAY	Название дня. дополненное пробелами до 9-ти символов.
DY	Сокращенное название дня.
J	Дата юлианского календаря: число дней, считая с первого января 4712 года до н.э.
AM или PM	AM -до полудня.PM- после полудня
A.M. или P.M.	A.M. -до полудня.P.M.- после полудня
HHилиHH12	Час дня (1-12).
HH24	Час дня (0-23).
MI	Минута (0-59)

SS или SSSSS	Секунда (0-59) или количество секунд после полуночи (0-86399).
-/,.;:	Знаки пунктуации.
"...текст..."	Текст воспр в возвращенном значении.

Функции преобразования

Функция	Возвращаемое значение
CHARTOROWID(char)	Char преобразуется из типа данных CHAR в тип данных ROWID
CONVERT(char, dest_char_set [,source_char_set])	Преобразует символьную строку из набора символов source_char_set в набор символов dest_char_set
HEXTORAW (char)	Преобразует значение char, содержащее шестнадцатиричные цифры, в значение типа данных RAW
RAWTOHEX (raw)	Преобразует raw в символьное значение. содержащее его шестнадцатиричный эквивалент
ROWIDTOCHAR (rowid)	Преобразует значение типа ROWID в значение типа CHAR
TO_CHAR (expr [.fmt [,'nls_num_fmt']])	Преобразует значение expr типа DATE или NUMBER в значение типа CHAR по формату форматной маски fmt. Если fmt отсутствует. значения типа DATE преобразуются по формату. заданному по умолчанию, и значения типа NUMBER- в значение типа CHAR с шириной. достаточной для того, чтобы вместить все значащие цифры. Значение enls_num_fint' определяет связанные с языком форматные маски. В Trusted ORACLE преобразует значения MLS или MLS LABEL в значение типа VARCHAR2
TO_DATE (char[.fmt [/nlsjang']])	Преобразует char в значение типа DATE с помощью форматной маски fmt. Если fmt опускается, используется форматная маска для даты, принятая по умолчанию.*iils_ang" задает язык, используемый в названиях месяцев и дней
TO_MULTI_BYTE (char)	Преобразует однобайтовые символы, имеющие многобайтовые эквиваленты, в соответствующие многобайтовые символы
TO NUMBER (char [.fmt [,'nls_lang']])	Преобразует char, содержащее число в формате. указанном параметром fmt. в значение типа NUMBER, 'nls_lang ' задает язык, определяющий символы валют и числовые разделители

TO_SINGLE_BYTE (char)	Преобразует многобайтовые символы, имеющие однобайтовые эквиваленты, в соответствующие однобайтовые символы

Функции для обработки данных любого типа

Функция	Возвращаемое значение
DECODE (expr. search 1. return 1 « [search2, return2,]...[default])	Если expr равно search, возвращается соответствующий результат return. Если совпадающей пары не найдено, возвращается default.
DUMP(expr[. return_format [, start position[. length]]])	Expr во внутреннем формате Oracle
GREATEST(expr[. expr]...)	Наибольшее значение expr
LEAST(expr[. expr]...)	Наименьшее значение expr
NVL(exprl.expr2)	Возвращает expr2. если exprl имеет пустое значение, в противном случае возвращает exprl.
UID	Целое число, которое уникально идентифицирует текущего пользователя.
USER	Имя текущего пользователя ORACLE.
USERENV(option)	Возвращает информацию о текущем сеансе. Аргументы помещаются в одиночных кавычках. Аргументы: ENTRYID. SESSIONSID. TERMINAL. LANGUAGE или LABEL.
VSIZE(expr)	Длина в байтах внутреннего представления для expr.

Атрибуты курсора

Имя	Что возвращает
%FOUND	Значение TRUE, если успешно выбрана хотя бы одна строка; в противном случае возвращает значение FALSE
%ROWCOUNT	Количество строк, выбранных из курсора на данный момент времени
%ISOPEN	Значение TRUE, если курсор открыт; в противном случае возвращает значение FALSE
%BULK ROWCOU XT	Коллекцию, в которой для каждого элемента исходной коллекции, заданной в операторе FORALL. указано количество строк, модифицированных SQL-инструкцией

%BULK EXCEPTI ONS	Коллекцию, в которой для каждого элемента исходной коллекции, заданной в операторе FORALL. указано инициированное Oracle исключение

Представления словаря данных

Имя таблицы	Комментарии
ALL_CATALOG	Все таблицы, представления, синонимы, последовательности, доступные пользователю.
ALL_COL_COMMENTS	Комментарии на столбцы доступных пользователю таблиц и представлений-
ALL_COL_PRIVS	Привилегий на отдельные столбцы объектов, данные пользователем другим пользователям; полученные от других персонально, через роль или опцию PUBLIC.
ALL_COL_PRIVS_MADE	Привилегии, данные самим пользователем или другими на столбцы принадлежащих ему объектов.
ALL_COL_PRIVS_RECD	Привилегии на отдельные столбцы объектов, данные пользователю персонально, через роль или опцию PUBLIC.
ALL_CONSTRAINTS	Описания ограничений, заданных на доступные пользователю таблицы.
ALLCONSCOLUMNS	Информация о доступных столбцах, заданных в определениях ограничений на таблицы.
ALL_DB_LINKS	Информация о доступных пользователю связях с другими базами.
ALL_DEF_AUDIT_OPTS	Опции проверки доступа для вновь созданных объектов, доступных пользователю.
ALL_DEPENDENCIES	Сведения обо всех взаимосвязях между объектами, к которым пользователь имеет доступ.
ALL_ERRORS	Текущие ошибки компиляции в хранимых PL/SQL-объекгах. к которым пользователь имеет доступ-
ALL INDEXES	Описание доступных пользователю индексов на таблицы.
ALL_IND_COLUMNS	Перечень индексированных столбцов в доступных пользователю таблицах.
ALL OBJECTS	Перечень всех объектов БД. доступных пользователю.
ALL SEQUENCES	Все последовательности, доступные пользователю.

ALL_SNAPSHOTS	Все снимки таблиц, доступные пользователю.
ALL_SOURCE	Тексты всех доступных пользователю хранимых PL/SQL-объектов (процедур, функций, пакетов).
ALL_SYNONYMS	Все доступные пользователю синонимы на объекты БД.
ALL_TABLES	Описание всех доступных пользователю таблиц.
ALL_TAB_COLUMNS	Столбцы всех таблиц, представлений и кластеров.
ALL_TAB_COMMENTS	Комментарии на таблицы и представления, доступные пользователю.
ALL_TAB_PRIVS	Привилегии на объекты, данные пользователем другим, а также полученные от других персонально, через роль или опцию PUBLIC.
ALL_TAB_PRIVS_MADE	Привилегии, данные самим пользователем или другими на принадлежащие ему объекты.
ALL_TAB_PRIVS_RECD	Привилегии на объекты, данные пользователю персонально, через роль или опцию PUBLIC.
ALL_TRIGGERS	Триггеры базы данных, доступные пользователю.
ALL_TRIGGERS_COLS	Ссылки на столбцы в созданных пользователем триггерах и в триггерах на его таблицы.
ALL_USERS	Сведения обо всех пользователях БД.
ALL_VIEWS	Перечень и описание представлений, доступных пользователю.
AUDIT_ACTIONS	Таблица описания для кодов аудитинга: привязка кодов типов действий к их наименованиям.
CAT	Синоним USER CATALOG
CLU	Синоним USER CLUSTERS
COLS	Синоним USER TAB COLUMNS
C OLUMN_PRIVILEGES	Привилегии на отдельные столбцы таблиц, данные пользователем другим пользователям: полученные от других персонально, через роль или опцию PUBLIC.
DICT	Синоним DICTIONARY
DICTIONARY	Перечень всех объектов словаря данных.
DICT_COLUMNS	Описание столбцов в таблицах и представлениях словаря данных.
GLOBAL_NAME	Глобальное имя базы данных.
IND	Синоним USER INDEXES
INDEX_HISTOGRAM	Статистика по частоте использования индексных ключей.
INDEX_STATS	Статистика на бинарные деревья индексов.
OBJ	Синоним USER OBJECTS
RESOURCE_COST	Таблица стоимости ресурсов

ROLE_ROLE_PRIVS	Роли, включенные в другие роли.
ROLE_SYS_PRIVS	Системные привилегии, данные ролям.
ROLE_TAB_PRIVS	Привилегии на таблицы, данные ролям.
SEQ	Синоним на USER SEQUENCES
SESSION_PRIVS	Текущие привилегии пользователя.
SESSION_ROLES	Роли, доступные пользователю в данный момент.
SYN	Синоним USER SYNONYMS
TABLE_PRIVILEGES	Привилегии на таблицы, данные пользователем другим пользователям, а также полученные от других персонально, через роль или опцию PUBLIC.
TABS	Синоним USER TABLES
USER_AUDIT_OBJECT	Строки журнала проверки команд к объектам БД: таблицам, кластерам, представлениям, индексам, последовательностям, связям баз данных, синонимам, процедурам, триггерам, сегментам отката, табличным пространствам, ролям, пользователям.
USER_AUDIT_SESSION	Строки журнала аудитинга по текущей сессии
USER_AUDIT_STATEME NT	Строки журнала проверки, касающиеся команд GRANT. REVOKE. AUDIT. NO AUDIT и ALTER SYSTEM.
USER_AUDIT_TRAIL	Все доступные пользователю строки журнала аудитинга.
USER_CATALOG	Таблицы, представления, синонимы и последовательности, принадлежащие пользователю.
USER_CLUSTERS	Описание всех кластеров пользователя.
USER_CLU_COLUMNS	привязка столбцов таблиц к столбцам кластеров.
USER_COL_COMMENTS	Комментарии на столбцы таблиц пользователя-
USER_COL_PRIVS	Привилегии на отдельные столбцы объектов, данные пользователем другим пользователям, а также полученные от других.
USER_COL_PRIVS_MAD E	Привилегии на отдельные столбцы объектов, принадлежащих пользователю.
USER_COL_PRIVS_RECD	Привилегии на отдельные столбцы объектов, предоставленные пользователю.
USER_CONSTRAINTS	Определения ограничений на таблицы пользователя.
USER_CONS_COLUMNS	Информация о доступных столбцах в определениях ограничений.
USER_DB_LINKS	Связи с другими БД. созданные пользователями.
USER_DEPENDENCIES	Сведения обо всех взаимозависимостях объектов пользователя.
USER_ERRORS	Текущие ошибки компиляции в хранимых PL/SQL-объектах пользователя.

USER_EXTENTS	Экстенты памяти, содержащие сегменты, которыми владеет пользователь.
USER_FREE_SPACE	Свободные экстенты в доступных пользователю табличных пространствах.
USER_INDEXES	Описание всех созданных пользователем индексов.
USER_IND_COLUMNS	Столбцы, содержащиеся в индексах, созданных пользователем, или на таблицы пользователя.
USER_OBJECTS	Все принадлежащие пользователю объекты.
USER_OBJECT_SIZE	Размеры различных PL/SQL-объектов пользователя в байтах.
USER_OBLAUDIT_OPTS	Опции проверки для принадлежащих пользователю объектов.
USER_RESOURCE_LIMITS	Ограничения на ресурсы для пользователя.
USER_ROLE_PRIVS	Роли, разрешенные пользователю.
USER_SEGMENTS	Память, занятая под сегменты базы данных.
USER_SEQUENCES	Описание созданных пользователем последовательностей.
DICTIONARY	Содержит перечень всех доступных пользователю таблиц. представлений, синонимов в словаре данных.
DICT_COLUMNS	Перечень столбцов в объектах словаря данных, доступных пользователю.
CONSTRAINT_DEFS	Информация об ограничениях, заданных в описаниях всех таблиц, к которым пользователь имеет доступ.
CONSTRAINT_COLUMNS	Перечень всех столбцов, доступных пользователю, и заданных в определениях ограничений на таблицы.

Приложение 3. Пример реализации БД

Создание пользователя:

Шаг 1. Создание пользователя.

```
CREATE USER "EGOROV_AIS"  PROFILE "DEFAULT"
   IDENTIFIED BY "12345" PASSWORD EXPIRE DEFAULT TABLESPACE
"EXAMPLE"
   TEMPORARY TABLESPACE "TEMP"
   QUOTA UNLIMITED
   ON EXAMPLE
   QUOTA UNLIMITED
   ON TEMP
   QUOTA UNLIMITED
   ON USERS
   ACCOUNT UNLOCK;
GRANT "CONNECT" TO "EGOROV_AIS";
```

Шаг 2. Создание роли.

```
CREATE ROLE "EGOROV_ROLE"  NOT IDENTIFIED;
GRANT ALTER ANY INDEX TO "EGOROV_ROLE";
GRANT ALTER ANY PROCEDURE TO "EGOROV_ROLE";
GRANT ALTER ANY SEQUENCE TO "EGOROV_ROLE";
GRANT ALTER ANY TABLE TO "EGOROV_ROLE";
GRANT ALTER ANY TRIGGER TO "EGOROV_ROLE";
GRANT CREATE ANY INDEX TO "EGOROV_ROLE";
GRANT CREATE ANY PROCEDURE TO "EGOROV_ROLE";
GRANT CREATE ANY SEQUENCE TO "EGOROV_ROLE";
GRANT CREATE ANY SYNONYM TO "EGOROV_ROLE";
GRANT CREATE ANY TABLE TO "EGOROV_ROLE";
GRANT CREATE ANY TRIGGER TO "EGOROV_ROLE";
GRANT CREATE ANY VIEW TO "EGOROV_ROLE";
GRANT CREATE PROCEDURE TO "EGOROV_ROLE";
GRANT CREATE PUBLIC SYNONYM TO "EGOROV_ROLE";
GRANT CREATE SEQUENCE TO "EGOROV_ROLE";
GRANT CREATE TABLE TO "EGOROV_ROLE";
GRANT CREATE TRIGGER TO "EGOROV_ROLE";
```

```
GRANT CREATE VIEW TO "EGOROV_ROLE";
GRANT DROP ANY PROCEDURE TO "EGOROV_ROLE";
GRANT DROP ANY SEQUENCE TO "EGOROV_ROLE";
GRANT DROP ANY TABLE TO "EGOROV_ROLE";
GRANT DROP ANY TRIGGER TO "EGOROV_ROLE";
GRANT DROP ANY VIEW TO "EGOROV_ROLE";
GRANT EXECUTE ANY PROCEDURE TO "EGOROV_ROLE";
GRANT "CONNECT" TO "EGOROV_ROLE";
```

Шаг 3. Присвоение роли.

```
GRANT "EGOROV_ROLE" TO "EGOROV_AIS";

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание таблиц БД
SPOOL 01ais.lst
/* ****************************************** */
/* Модуль администратора */
DROP TABLE ad_user CASCADE CONSTRAINTS;
PROMPT Создание таблицы ad_user
CREATE TABLE ad_user (
        us_id           INTEGER,
        us_name         VARCHAR2(20),
        us_pass         VARCHAR2(20),
        us_rol_id       INTEGER
);

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание ограничений NOT NULL
SPOOL 02ais.lst
/* ****************************************** */
/* Модуль технолога */
PROMPT Ограничения на таблицу pro_equipment
ALTER TABLE pro_equipment
DROP CONSTRAINT c_eq_time_pz_null;
```

```
ALTER TABLE pro_equipment
MODIFY (eq_time_pz INTEGER CONSTRAINT c_eq_time_pz_null NOT NULL);

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание индексов
SPOOL 03ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Удаление и создание индекса на таблицу ad_role
DROP INDEX i_rol_id;
CREATE UNIQUE INDEX i_rol_id
ON ad_role(rol_id);

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание первичных ключей
SPOOL 04ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание первичного ключа для таблицы ad_role
ALTER TABLE ad_role
DROP PRIMARY KEY;
ALTER TABLE ad_role
ADD CONSTRAINT c_ad_role_pk
PRIMARY KEY(rol_id);

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание элементов ссылочной целостности
SPOOL 05ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание внешнего ключа для таблицы ad_user
ALTER TABLE ad_user
```

```
DROP CONSTRAINT c_us_rol_id_fk CASCADE;
ALTER TABLE ad_user
ADD (CONSTRAINT c_us_rol_id_fk
FOREIGN KEY(us_rol_id)
REFERENCES ad_role(rol_id));

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание последовательностей
SPOOL 07ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание последовательности для первичного ключа таблицы ad_user
DROP SEQUENCE s_ad_user;
CREATE SEQUENCE s_ad_user;

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание триггеров
SPOOL 08ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание триггера на вставку данных для таблицы ad_role
CREATE OR REPLACE TRIGGER t_ins_ad_role
BEFORE INSERT ON ad_role FOR EACH ROW
        BEGIN
        SELECT s_ad_role.NEXTVAL
        INTO :NEW.rol_id
        FROM dual;
        END;
/
PROMPT Создание триггера на обновление данных для таблицы ad_role
CREATE OR REPLACE TRIGGER t_upd_ad_role
BEFORE UPDATE ON ad_role FOR EACH ROW
        BEGIN
```

```
        IF (:new.rol_id != :old.rol_id) THEN
        :new.rol_id := :old.rol_id;
        END IF;
        END;
/
PROMPT Создание триггера на удаление данных для таблицы ad_role
CREATE OR REPLACE TRIGGER t_del_ad_role
BEFORE DELETE ON ad_role FOR EACH ROW
        BEGIN
        DELETE FROM ad_user
        WHERE us_rol_id = :old.rol_id;
        END;
/

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Вставка данных в справочники
SPOOL 09ais.lst
/* ****************************************** */
/* Модуль администратора */
INSERT INTO ad_role
        VALUES (1, 'guest', 'Запись');
INSERT INTO ad_role
        VALUES (2, 'chief', 'Запись');
INSERT INTO ad_role
        VALUES (3, 'technologist', 'Запись');
INSERT INTO ad_role
        VALUES (4, 'admin', 'Запись');
INSERT INTO ad_role
        VALUES (5, 'operator', 'Запись');
COMMIT;

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание представлений
```

```
SPOOL 10ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание представления для таблиц ad_user, ad_role
CREATE OR REPLACE VIEW v_ad_user
AS
        SELECT u.us_id, u.us_name, u.us_pass, r.rol_name
        FROM ad_user u, ad_role r
        WHERE u.us_rol_id=r.rol_id ;

PROMPT Автор - Егоров Д.А.
PROMPT Дата обновления  30.04.2007
PROMPT Создание синонимов
SPOOL 11ais.lst
/* ****************************************** */
/* Модуль администратора */
PROMPT Создание синонима v_ad_user
DROP PUBLIC SYNONYM v_ad_user;
CREATE PUBLIC SYNONYM v_ad_user  FOR v_ad_user;
```

Подключение к БД из PHP:

```
<?php
        // Соединение с БД
        $db_sid='//localhost/orcl';
        $db_conn=ocilogon('EGOROV','superais',$db_sid);
        if ($db_conn)                   // delete!!!
                {
                        //echo '<br> connected <br>';
                }
        else
                {
                        echo 'not connected <br>';
                        $page='err1';   // Код ошибки соединения с БД
                }
?>
```

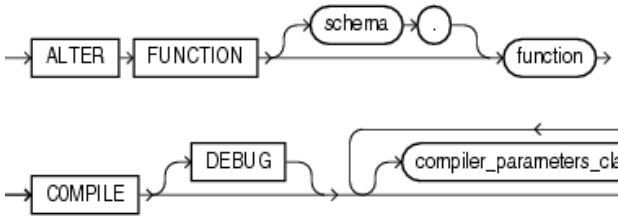
Вывод таблицы в браузер:

```
<?php
$cmdstr="SELECT * FROM ad_role";
$stmt=OCIParse($db_conn,$cmdstr);
        OCIExecute($stmt,OCI_DEFAULT);
echo '</table>';
echo '<table border=1 align=center>';
                echo '<tr>';
                        echo '<td>';
                                echo 'Роль';
                        echo '</td>';
                echo '</tr>';
        while (OCIFetch($stmt))
        {
                echo '<tr>';
                        echo '<td>';
                                echo OCIResult($stmt,'ROL_NAME');
                        echo '</td>';
                echo '</tr>';
        };
echo '</table>';
?>
```

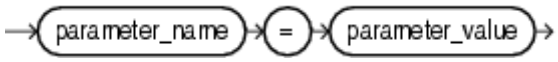
Приложение 4. Синтаксические диаграммы

ALTER FUNCTION

alter_function::=

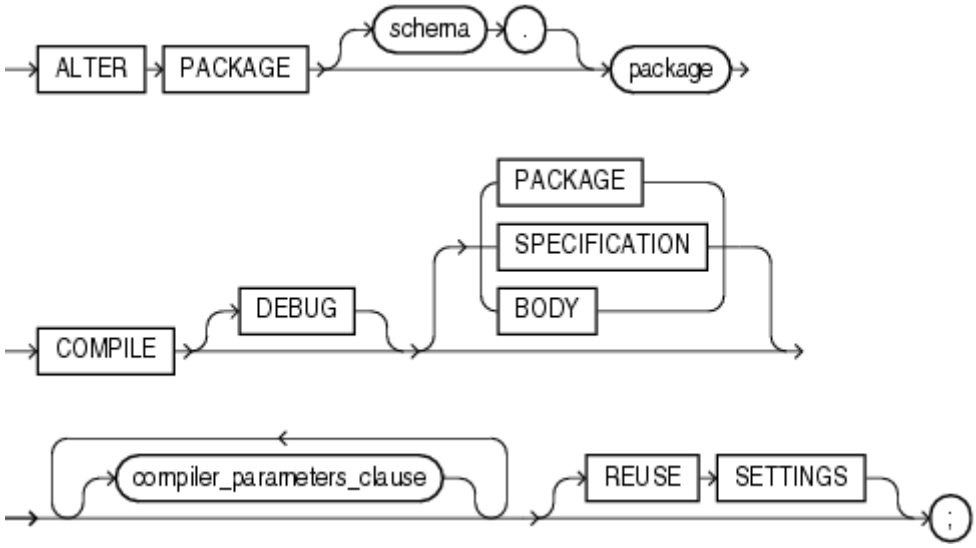


compiler_parameters_clause::=

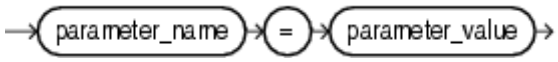


ALTER PACKAGE

alter_package::=

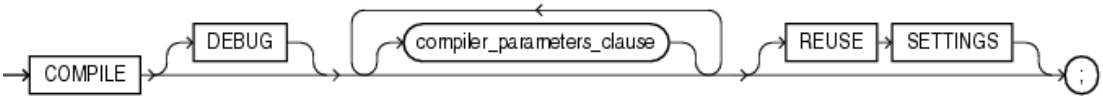


compiler_parameters_clause::=

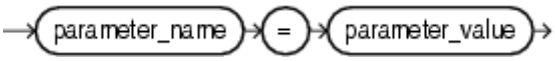


ALTER PROCEDURE

alter_procedure::=

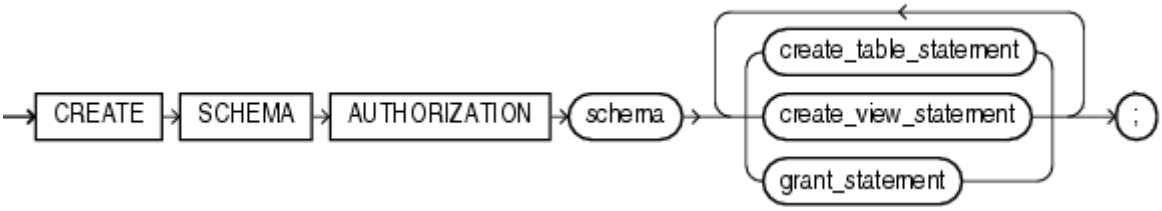


compiler_parameters_clause::=



CREATE SCHEMA

create_schema::=

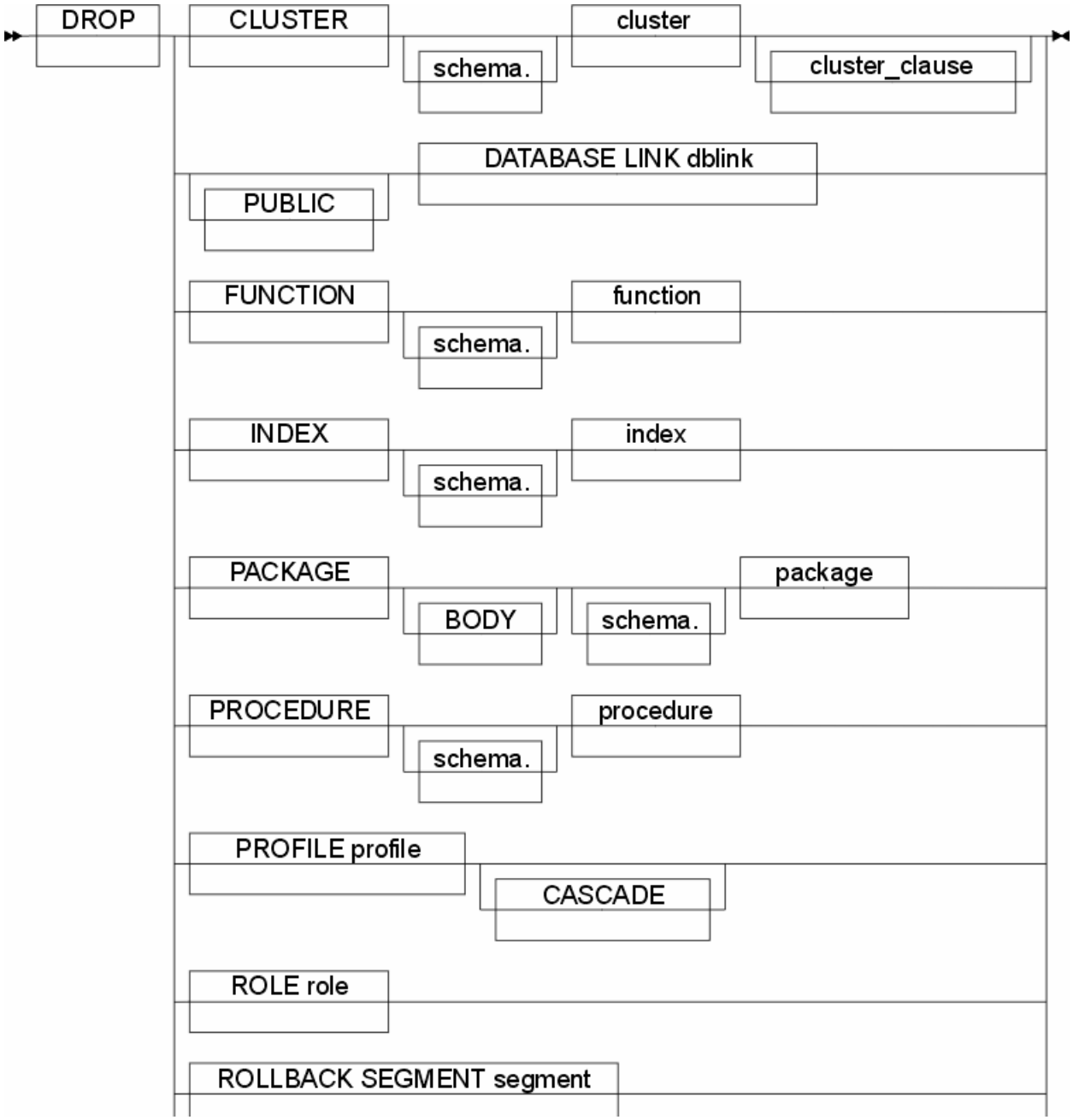


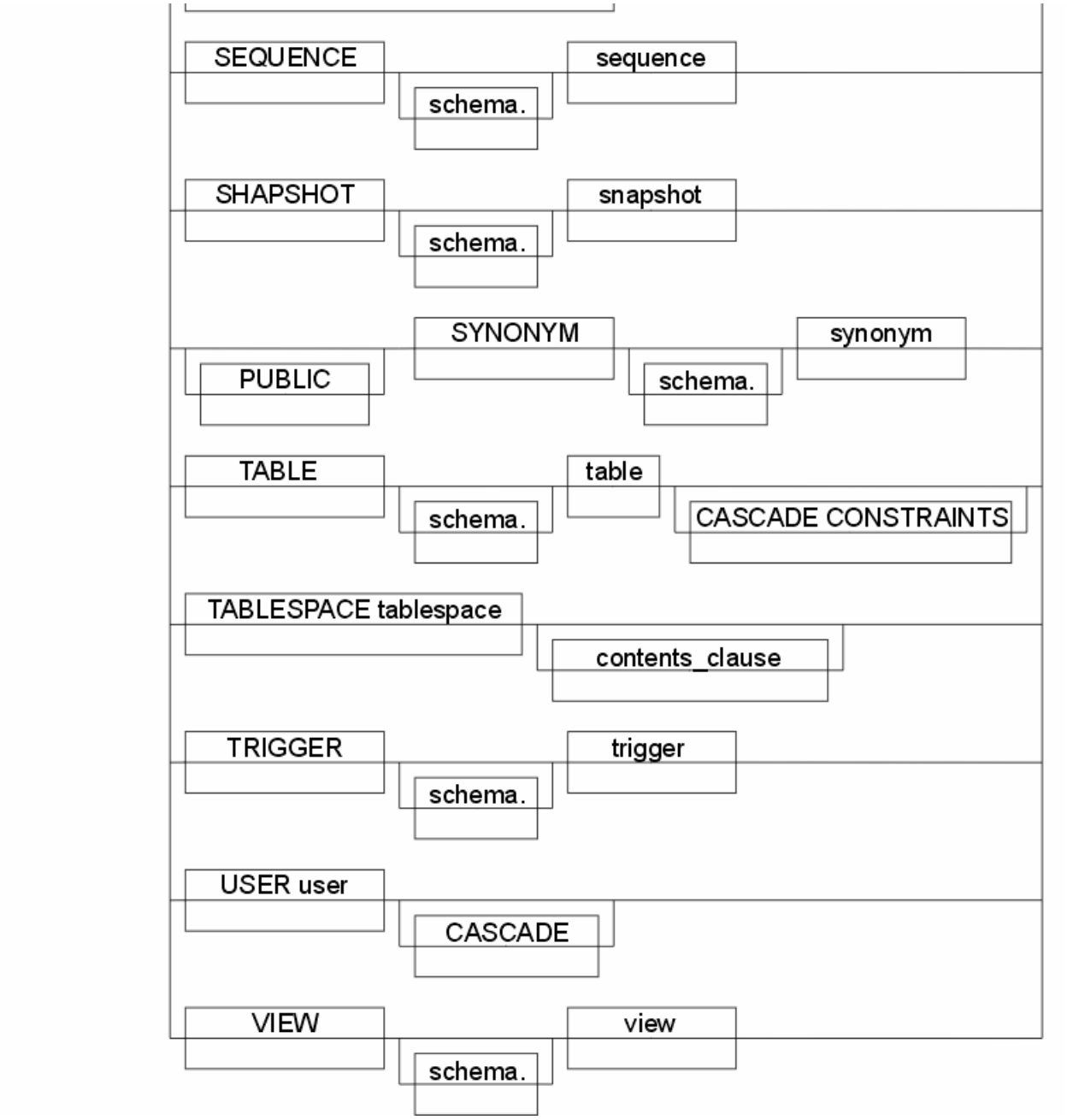
DROP TABLE

drop_table::=

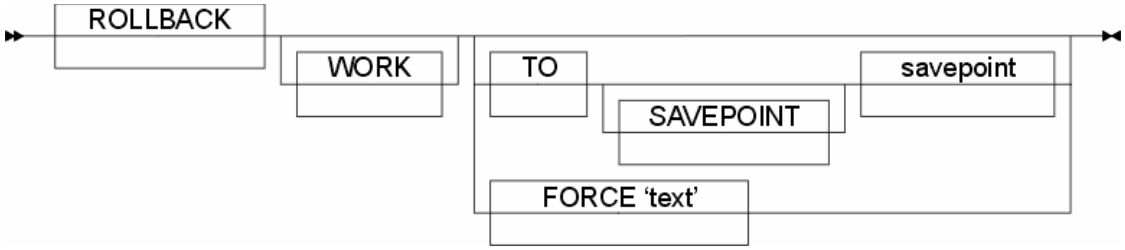
ALTER TABLE имя таблицы		
DROP	CONSTRAINT имя ограничения	[CASCADE]
	PRIMARY KEY	
	UNIQUE (столбец, столбец....)	

DROP


DROP
CLUSTER
schema.
cluster
cluster_clause
DATABASE LINK dblink
PUBLIC
FUNCTION
schema.
function
INDEX
schema.
index
PACKAGE
BODY
schema.
package
PROCEDURE
schema.
procedure
PROFILE profile
CASCADE
ROLE role
ROLLBACK SEGMENT segment

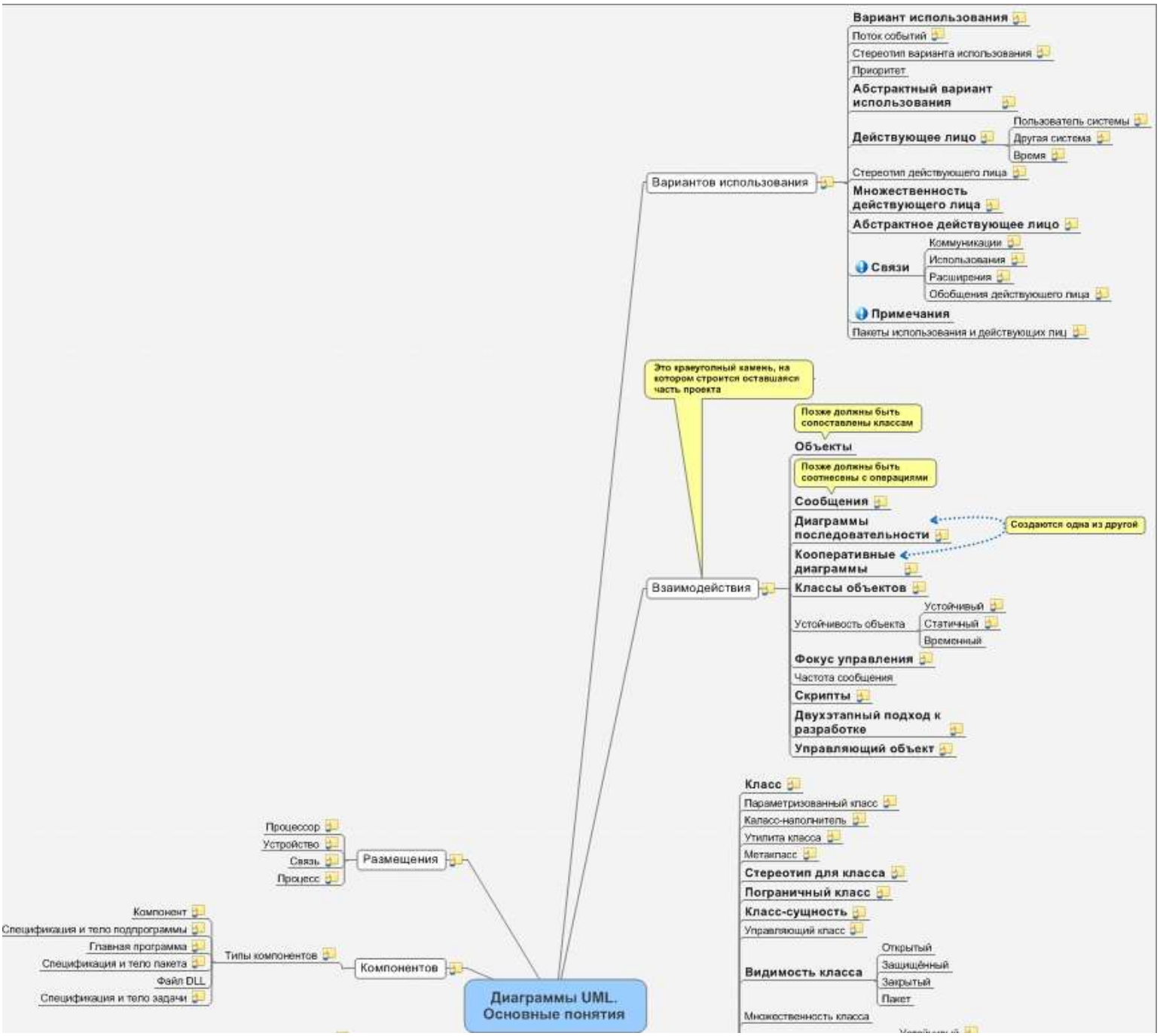

SEQUENCE
schema.
sequence
SHAPSHOT
schema.
snapshot
SYNONYM
PUBLIC
schema.
synonym
TABLE
schema.
table
CASCADE CONSTRAINTS
TABLESPACE tablespace
contents_clause
TRIGGER
schema.
trigger
USER user
CASCADE
VIEW
schema.
view

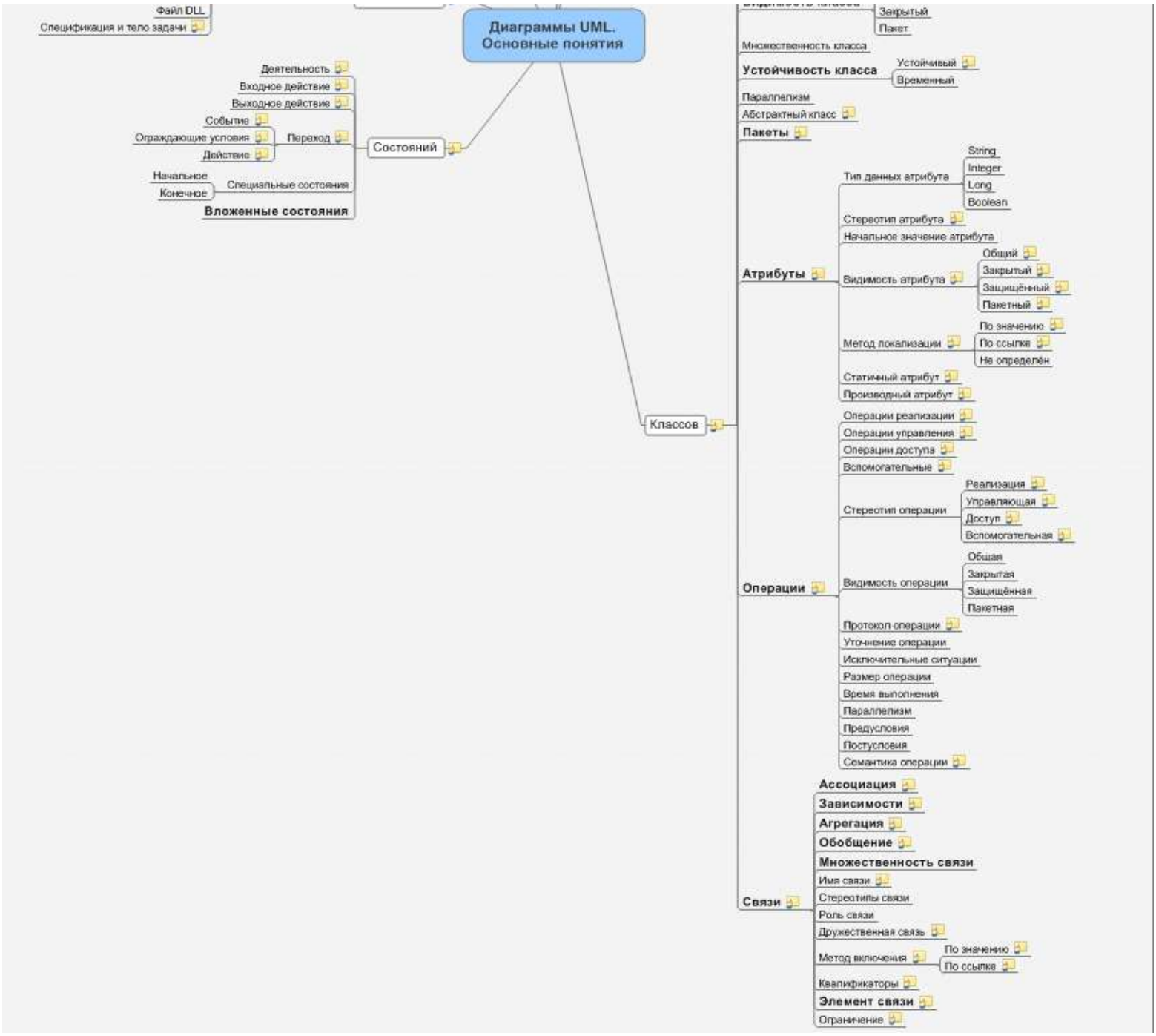
ROLLBACK


ROLLBACK
WORK
TO
SAVEPOINT
savepoint
FORCE 'text'

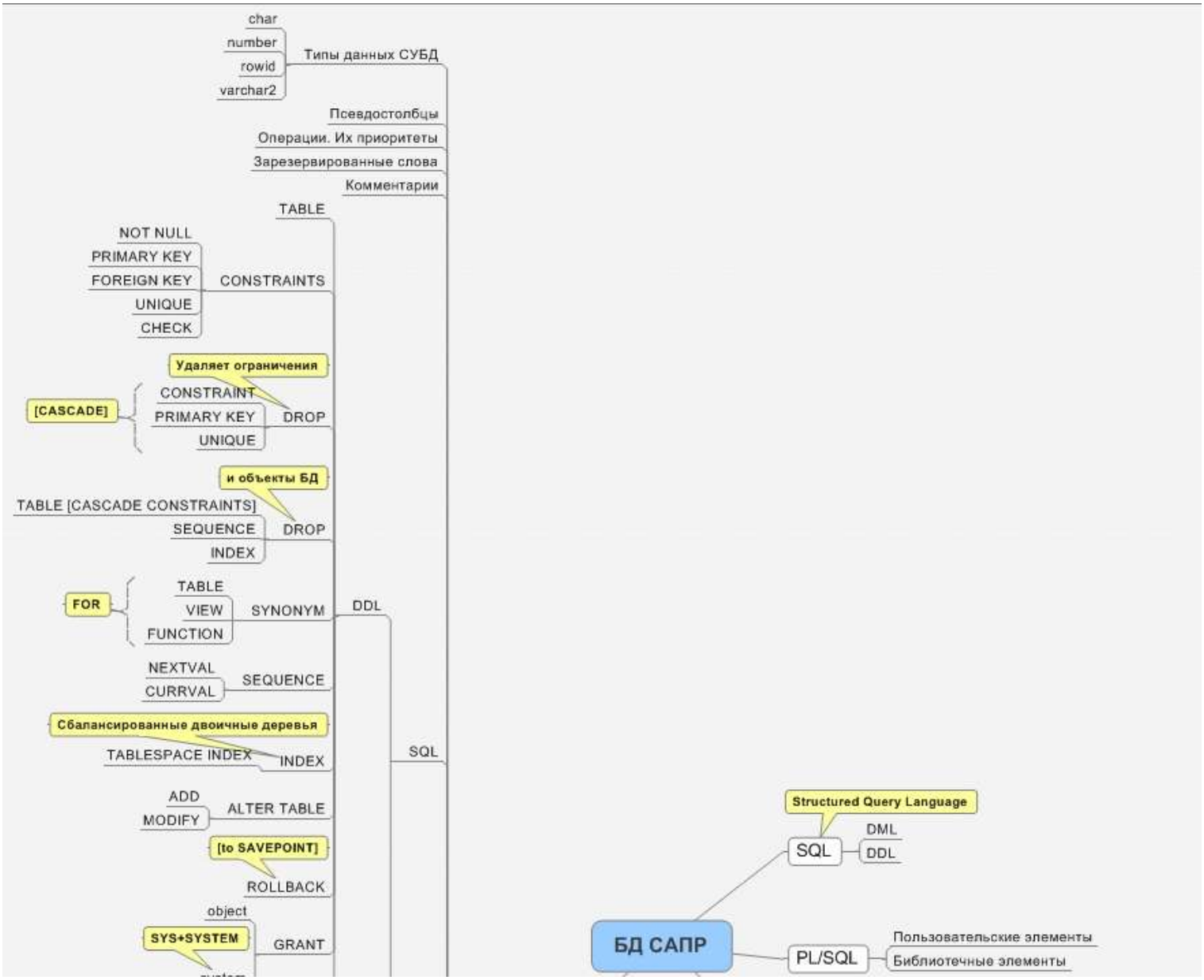
Приложение 5. Mind maps

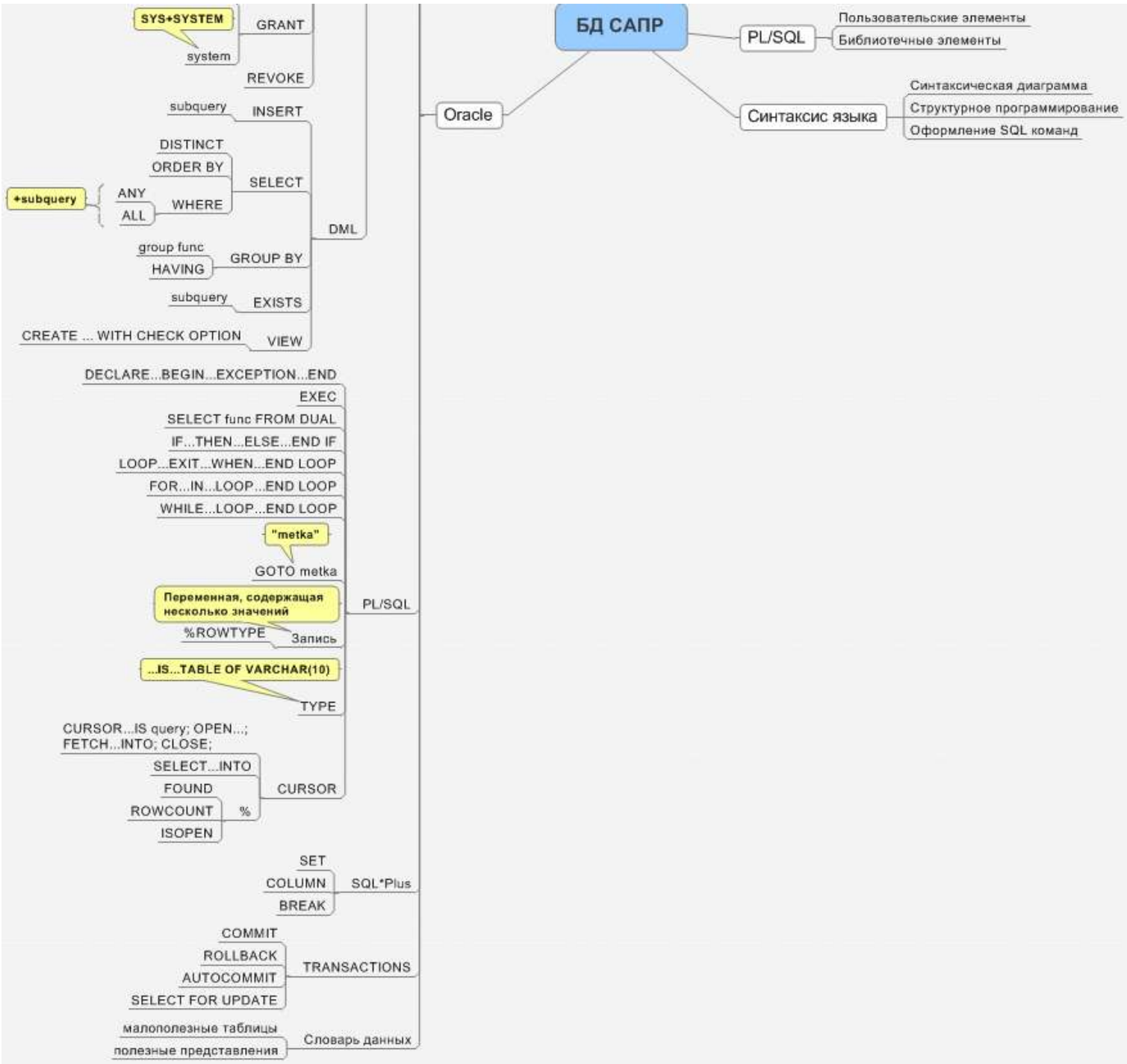
UML mind map




Диаграммы UML. Основные понятия
Файл DLL
Спецификация и тело задачи
Состояний
Деятельность
Входное действие
Выходное действие
Переход
Событие
Ограждающие условия
Действие
Специальные состояния
Начальное
Конечное
Вложенные состояния
Классов
Закрытый
Пакет
Множественность класса
Устойчивость класса
Устойчивый
Временный
Параллелизм
Абстрактный класс
Пакеты
Атрибуты
Тип данных атрибута
String
Integer
Long
Boolean
Стереотип атрибута
Начальное значение атрибута
Видимость атрибута
Общий
Закрытый
Защищённый
Пакетный
Метод локализации
По значению
По ссылке
Не определён
Статичный атрибут
Производный атрибут
Операции
Операции реализации
Операции управления
Операции доступа
Вспомогательные
Стереотип операции
Реализация
Управляющая
Доступ
Вспомогательная
Видимость операции
Общая
Закрытая
Защищённая
Пакетная
Протокол операции
Уточнение операции
Исключительные ситуации
Размер операции
Время выполнения
Параллелизм
Предусловия
Постусловия
Семантика операции
Связи
Ассоциация
Зависимости
Агрегация
Обобщение
Множественность связи
Имя связи
Стереотипы связи
Роль связи
Дружественная связь
Метод включения
По значению
По ссылке
Квалификаторы
Элемент связи
Ограничение

БД САПР mind map




БД САПР
PL/SQL
Пользовательские элементы
Библиотечные элементы
Синтаксис языка
Синтаксическая диаграмма
Структурное программирование
Оформление SQL команд
Oracle
DML
GRANT
SYS+SYSTEM
system
REVOKE
INSERT
subquery
SELECT
DISTINCT
ORDER BY
WHERE
ANY
ALL
+subquery
GROUP BY
group func
HAVING
EXISTS
subquery
VIEW
CREATE ... WITH CHECK OPTION
PL/SQL
DECLARE...BEGIN...EXCEPTION...END
EXEC
SELECT func FROM DUAL
IF...THEN...ELSE...END IF
LOOP...EXIT...WHEN...END LOOP
FOR...IN...LOOP...END LOOP
WHILE...LOOP...END LOOP
GOTO metka
"metka"
Переменная, содержащая несколько значений
Запись
%ROWTYPE
TYPE
...IS...TABLE OF VARCHAR(10)
CURSOR
CURSOR...IS query; OPEN...; FETCH...INTO; CLOSE;
%
SELECT...INTO
FOUND
ROWCOUNT
ISOPEN
SQL*Plus
SET
COLUMN
BREAK
TRANSACTIONS
COMMIT
ROLLBACK
AUTOCOMMIT
SELECT FOR UPDATE
Словарь данных
малополезные таблицы
полезные представления