

# ОСНОВНЫЕ ОБЪЕКТЫ БАЗЫ ORACLE

Графеева Н.Г.

2016

# Основные объекты

- Таблицы
- Правила целостности
- Индексы
- Представления
- Синонимы
- Секвенции
- Триггеры
- Процедуры
- Функции
- Пакеты
- Типы
- Соединения
- Задания

# Таблицы

- При определении таблицы главное – определение имен столбцов таблицы и задание их типов.
- **Основные типы данных, используемые при создании таблиц:**
- CHAR(размер)
- Символьные данные фиксированной длины. Максимальный размер - 255 байт. Умалчиваемый размер - 1 байт.
- 
- VARCHAR2 (размер)
- Символьные данные переменной длины. Максимальный размер - 2000 байт. Умалчиваемый размер - 1 байт.
- 
- NUMBER(p,s)
- Числовые данные переменной длины. Точность p (общее количество цифр) может задаваться от 1 до 38. Масштаб s (число цифр после десятичной точки) может быть от -84 до 127
- 
- DATE
- Значение даты и времени (с точностью до секунд) в интервале от 1 января 4712 г. до н.э. до 31 декабря 4712 г. н.э.
- 
- TIMESTAMP
- Дата и время с точностью до миллисекунд
- =====
- *Над всеми типами данных предусмотрен достаточный набор операций, но не все операции над типами данных так очевидны. Наиболее часто возникают проблемы с типом DATE.*

# Пример (основные операции с датами)

- SELECT SYSDATE
  - SYSDATE + 1
  - SYSDATE + 1/24
  - EXTRACT(YEAR FROM SYSDATE)
  - EXTRACT(MONTH FROM SYSDATE)
  - TO\_CHAR(SYSDATE, 'DD/MM/YYYY HH24:MI:SS') ,
  - TO\_DATE('01/01/2015 12:20:25', 'DD/MM/YYYY HH24:MI')
  - FROM DUAL
- CURRENT\_DATE,
  - CURRENT\_DATE\_PLUS\_1\_DAY,
  - CURRENT\_DATE\_PLUS\_1\_HOUR,
  - CURRENT\_YEAR,
  - CURRENT\_MONTH,

# Работа с таблицами (SQL DDL)

- CREATE TABLE table-name (field-name type [ options ]
- [, field-name type [ options ] ]\*)
- =====
- ALTER TABLE table-name {ADD|DROP|ALTER}[COLUMN]
- (field-name [type] [ options ])
- =====
- DROP TABLE table\_name [CASCADE CONSTRAINTS]
- -----
- Примечание : опции – это правила целостности, значения по умолчанию и пр.

# Системные представления (словарь данных) для просмотра таблиц

- USER\_TABLES
- ALL\_TABLES
- DBA\_TABLES (только для администраторов)

# Задание 1

- Взгляните на содержимое системных табличных представлений, доступных в вашей схеме.

# Правила целостности

- В Oracle используются следующие правила целостности:
- 
- NOT NULL - запрет пустых значений
- UNIQUE - контроль уникальности
- PRIMARY KEY - первичный ключ
- FOREIGN KEY - внешний ключ
- CHECK - контроль допустимых значений

# Работа с правилами целостности (SQL DDL)

- ALTER TABLE table\_name {ADD|DROP|MODIFY} CONSTRAINT [const\_name] [const\_definition]
- ALTER TABLE table\_name RENAME CONSTRAINT const\_name TO const\_name
- ALTER TABLE table\_name {ENABLE|DISABLE} [CONSTRAINT] [const\_name] [const\_definition]

# Примеры

- ALTER TABLE emp ADD CONSTRAINT emp\_pk PRIMARY KEY (empno);
- =====
- ALTER TABLE EMPLOYEES ADD CONSTRAINT EMP\_JOB\_FK FOREIGN KEY (JOB\_ID) REFERENCES JOBS (JOB\_ID)
- =====
- ALTER TABLE emp DROP CONSTRAINT dept\_fkey
- =====
- ALTER TABLE dept DROP UNIQUE (dname, loc)
- =====
- ALTER TABLE dept RENAME CONSTRAINT dname\_ukey TO dname\_unikey
- =====
- ALTER TABLE dept DISABLE CONSTRAINT dname\_ukey
- =====
- ALTER TABLE dept ENABLE dname\_ukey

# Системные представления для просмотра правил целостности

- USER\_CONSTRAINTS
- ALL\_CONSTRAINTS
- DBA\_CONSTRAINTS (только для администраторов)

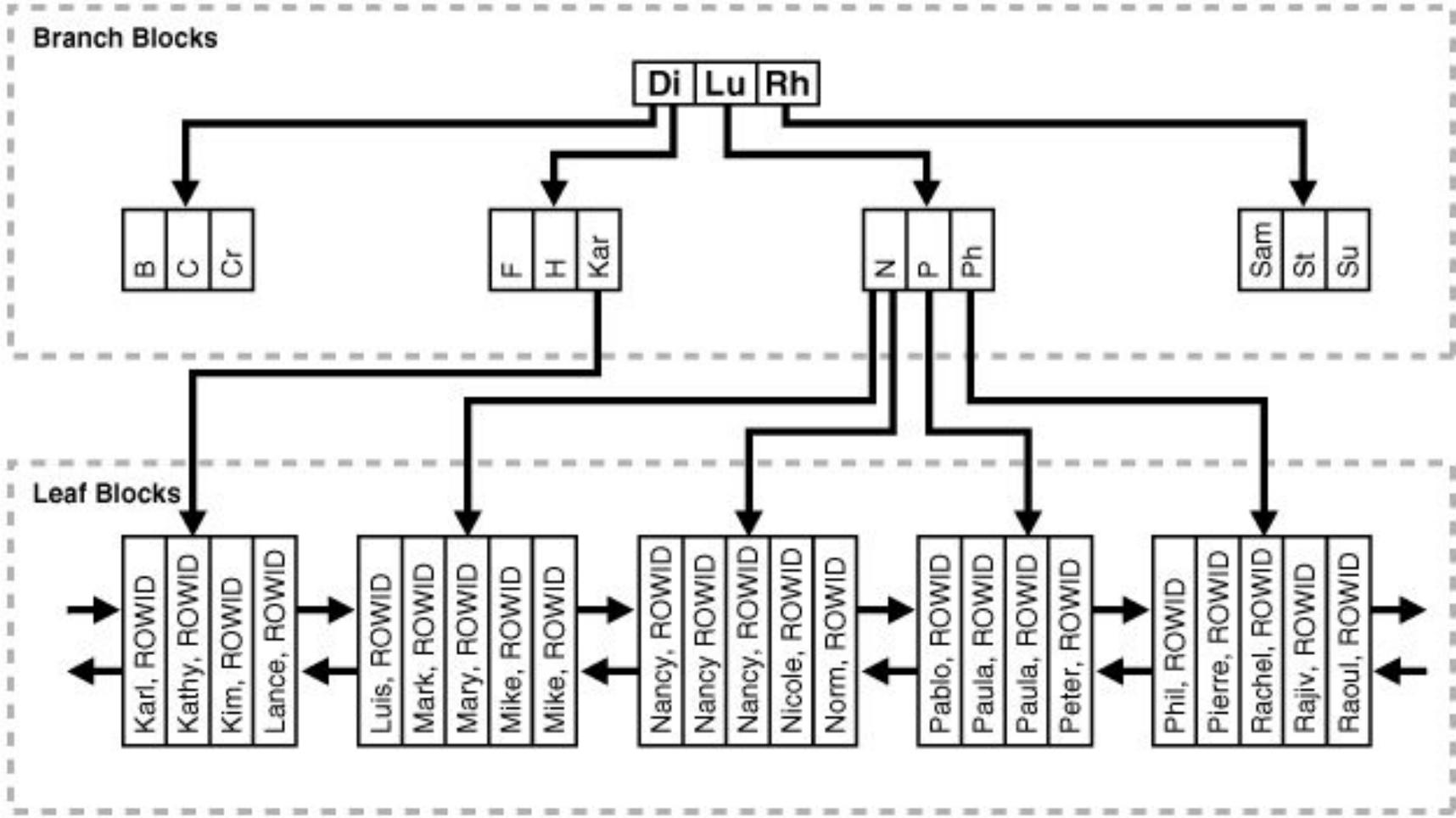
# Задание 2

- Взгляните на содержимое системных представлений правил целостности в вашей схеме. Сколько правил целостности?

# Индексы

- Индекс – это структура, связанная с таблицей и используемая в первую очередь для ускорения доступа к строкам таблицы.
- Основные формы организации индексов в ORACLE:
- B-tree индексы
- BITMAP индексы

# B-tree индекс



# ВІТМАР индекс (содержимое исходной таблицы)



CUSTOMER #	MARITAL_ STATUS	REGION	GENDER
101	single	east	male
102	married	central	female
103	married	west	female
104	divorced	west	male
105	single	central	female
106	married	central	female

# ВІТМАР индекс (структура індекса)

REGION='east'	REGION='central'	REGION='west'
1	0	0
0	1	0
0	0	1
0	0	1
0	1	0
0	1	0

# BITMAP индекс (обработка запроса)

```
select count(*) from FROM CUSTOMER WHERE MARITAL_STATUS = 'married' AND REGION IN ('central','west')
```

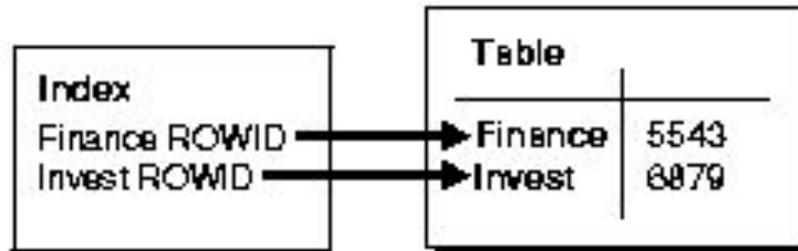
status = 'married'		region = 'central'		region = 'west'	=		AND	=	
0		0	OR	0	=	0	AND	=	0
1		1	OR	0	=	1	AND	=	1
1	AND	0	OR	1	=	1	AND	=	1
0		0	OR	1	=	0	AND	=	0
0		1	OR	0	=	0	AND	=	0
1		1	OR	0	=	1	AND	=	1

# Создание и удаление индексов (SQL DDL)

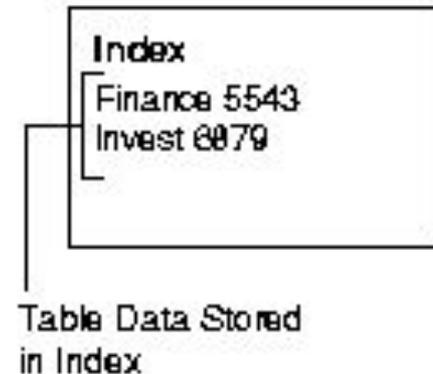
- CREATE [UNIQUE] [BITMAP] INDEX index-name
- ON table-name(field-name {,field-name}\*)
  
- DROP INDEX index-name

# Таблицы, организованные как ИНДЕКСЫ

## Regular Table and Index



## Index-Organized Table



# Создание таблиц-индексов (SQL DDL)

- CREATE TABLE table-name
- (field-name type [options] [UNIQUE (field-list)])
- {,field-name type[options] [UNIQUE (field-list)]}\* )
- ORGANIZATION INDEX

# Пример

- CREATE TABLE COUNTRIES
- (- COUNTRY\_ID CHAR(2) CONSTRAINT COUNTRY\_ID\_NN NOT NULL,
- COUNTRY\_NAME VARCHAR2(40),
- REGION\_ID NUMBER,
- CONSTRAINT COUNTRY\_C\_ID\_PK PRIMARY KEY (COUNTRY\_ID)
- )
- ORGANIZATION INDEX

# Системные представления для просмотра индексов

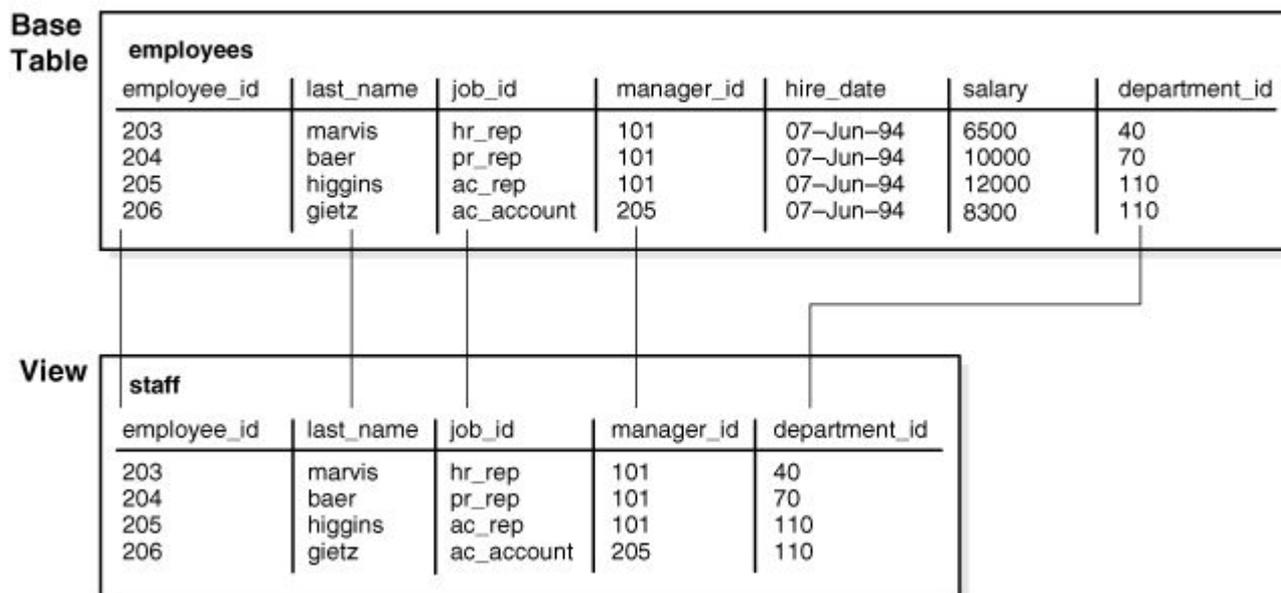
- USER\_INDEXES
- ALL\_INDEXES
- DBA\_INDEXES (доступно администратору)

# Задание 3

- Взгляните на содержимое представлений, связанных с индексами в своей схеме (в APEX, SQL Commands).  
Сколько их?

# Представления

Представление - это именованное правило выборки данных.



# Создание и удаление представлений (SQL DDL)

- **Синтаксис:**
- CREATE VIEW view-name [(field-list)]
- AS {SELECT-statement| UNION-statement} [ WITH CHECK OPTION ]
- DROP VIEW view-name

# Примеры

- CREATE VIEW STAFF
- AS SELECT employee\_id, last\_name, job\_id, manager\_id, department\_id  
FROM EMPLOYEES
- =====
- CREATE VIEW JOB\_IDS
- AS SELECT DISTINCT job\_id FROM EMPLOYEES
- =====
- CREATE VIEW STAFF\_MANAGER\_101
- AS SELECT employee\_id, last\_name, job\_id, manager\_id, department\_id  
FROM EMPLOYEES WHERE manager\_id = 101
- =====

# Системные представления

- USER\_VIEWS
- ALL\_VIEWS
- DBA\_VIEWS (доступно администратору)

# Задание 4

- Создайте представление с номерами, именами и максимальными зарплатами по каждой должности (из таблицы EMP демо базы ORACLE).

Взгляните на содержимое соответствующих системных представлений в своей схеме.

# Секвенции

- Секвенция - это объект базы данных, который генерирует целые числа в соответствии с правилами, установленными во время его создания. Для последовательности можно указывать как положительные, так и отрицательные целые числа. Последовательности применяют для самых разных целей, но в основном для автоматической генерации первичных ключей. Тем не менее к первичному ключу таблицы последовательность никак не привязана. При определении секвенции указывается следующая информация:
  - имя последовательности
  - стартовое значение (опционально)
  - интервал между числами
  - максимальное значение (опционально)
  - минимальное значение (опционально)
- • размер кэша для очередного набора сгенерированных чисел (опционально)

# Создание и удаление секвенций (SQL DDL)

- CREATE SEQUENCE seq-name
- [START WITH start-value]
- INCREMENT BY step-value
- [MAXVALUE max-value]
- [MINVALUE min-value]
- [CYCLE]
- [CACHE cache-size]
  
- ALTER SEQUENCE seq-name
- [START WITH start-value]
- INCREMENT BY step-value
- [MAXVALUE max-value]
- [MINVALUE min-value]
- [CYCLE]
- [CACHE cache-size]
  
- DROP SEQUENCE seq-name

# Примеры

- CREATE SEQUENCE sequence\_1 INCREMENT BY 10
- =====
- CREATE SEQUENCE sequence\_2  
START WITH 20  
INCREMENT BY -1  
MAXVALUE 20  
MINVALUE 0  
CYCLE  
CACHE 2
- =====
- DROP SEQUENCE sequence\_1

# Использование секвенций

- Seq-name.NEXTVAL - генерирует очередное значение секвенции
- Seq-name.CURRVAL - текущее значение секвенции

# Примеры

- `SELECT departments_seq.NEXTVAL FROM DUAL`
- =====
- `INSERT INTO DEPARTMENTS(DEPARTMENT_ID, DEPARTMENT_NAME, LOCATION_ID )  
VALUES  
(  
DEPARTMENTS_SEQ.NEXTVAL,  
'DEPARTMENT' || TO_CHAR(DEPARTMENTS_SEQ.CURRVAL),  
1700  
)`

# Системные представления для секвенций

- USER\_SEQUENCES
- ALL\_SEQUENCES
- DBA\_SEQUENCES (доступно только для администраторов)

# Задание 5

- Создайте подходящую секвенцию для таблицы DEPT1.
- Сгенерируйте 3 значения с использованием этой секвенции (используйте вспомогательную таблицу DUAL)
- Найдите эту секвенцию в соответствующем системном представлении (в APEX, SQL Commands).

# Процедуры, функции, пакеты

- Хранимые процедуры (stored procedure) и функции — это подпрограммы, которые выполняют некоторые действия с информацией в базе данных и при этом сами хранятся в базе данных. В Oracle хранимые процедуры и функции можно писать на языках PL/SQL (процедурное расширение SQL) и Java. Хранимые процедуры и функции никогда не передаются на клиентские компьютеры. Они всегда находятся в базе данных и выполняются СУБД на том компьютере, где располагается сервер базы данных.
- Процедуры и функции могут быть с параметрами и без параметров. Способы передачи параметров:
  - 
  - **IN** – параметр используется как параметр, передающий начальное значение от фактического параметра формальному при старте процедуры. Этот способ передачи параметра используется по умолчанию, т.е. когда способ передачи в явном виде не задан.
  - **OUT** – параметр передает значение в конце работы процедуры/функции от формального параметра фактическому
  - **IN OUT** – при старте процедуры/функции передает начальное значение от фактического параметра формальному, а в конце работы процедуры/функции от формального параметра фактическому

# Определение функций

- CREATE OR REPLACE FUNCTION name[(parameters...)]
- RETURN type
- IS
- [variables....]
- BEGIN
- ...
- RETURN ...
- ...
- [EXCEPTION
- WHEN ... THEN
- .....]
- END [name]

# Пример

- **Определение функции:**
  - CREATE OR REPLACE FUNCTION summ (a IN NUMBER, b IN NUMBER)
  - RETURN NUMBER
  - IS
  - var\_result NUMBER;
  - BEGIN
  - var\_result := a + b;
  - RETURN var\_result;
  - END summ;
- **Вызов функции:**
  - SELECT summ(2,3) FROM DUAL

# Задание 6

- Создайте и вызовите хранимую функцию, вычисляющую факториал натурального числа.

# Задание 7

- Создайте и вызовите хранимую функцию, определяющую количество служащих в демонстрационной базе ORACLE (таблица EMP)

.

# Определение процедур

- CREATE OR REPLACE PROCEDURE name [(parameters...)]
- IS
- [variables...]
- BEGIN
- .....
- [EXCEPTION
- WHEN ... THEN
- .....]
- END [name]

# Пример

- Определение процедуры:
- CREATE OR REPLACE PROCEDURE avgnumbers
- ( a IN NUMBER,
- b IN NUMBER,
- ar OUT NUMBER,
- geom OUT NUMBER
- )
- IS
- BEGIN
- ar := (a + b) / 2;
- geom := sqrt(a \* b);
- END avgnumbers;

# Пример

- Вызов процедуры:
- DECLARE
- A     NUMBER;
- B     NUMBER;
- AR    NUMBER;
- GEOM  NUMBER;
- BEGIN
- A := 2;
- B := 3;
- AVGNUMBERS ( A, B, AR, GEOM );
- dbms\_output.put\_line('AR=' || AR);
- dbms\_output.put\_line('GEOM =' || GEOM);
- END;

# Задание 8

- Создайте хранимую процедуру, выдающую возможную статистику по сотрудникам и департаментам (сколько всего сотрудников, сколько департаментов, сколько различных должностей, какая суммарная зарплата). Для вывода используйте пакет `dbms_output`.

# Как сохранять в базе результаты работ процедур и функций?

- Создать вспомогательную таблицу, например:
- `debug_log(id, LogTime, Message, inSource)`
- 
- `id` - идентификатор записи,  
`LogTime` – дата и время появления записи,
- `Message` – сообщение,
- `inSource` – имя процедуры или функции (от которой пришло сообщение)
- Такую таблицу **`debug_log`** можно использовать как журнал для фиксации результатов работы процедур и функций.

# Пример

- Вызов процедуры:
- DECLARE
- A     NUMBER;
- B     NUMBER;
- AR    NUMBER;
- GEOM NUMBER;
- BEGIN
- A := 2;
- B := 3;
- AVGNUMBERS ( A, B, AR, GEOM );
- INSERT INTO debug\_log(id, LogTime, Message, inSource)
- VALUES(debug\_log\_seq.nextval, sysdate, 'AR =' || AR || ' GEOM=' || GEOM, 'AVGNUMBERS');
- END;

# Задание 9

- Создайте таблицу **debug\_log** и подходящую для нее секвенцию.
- Создайте и вызовите процедуру, определяющую, даты приема на работу сотрудника, который работает дольше всех и сотрудника, который работает меньше всех. Результаты работы процедуры зафиксируйте в **debug\_log**.
- Просмотрите содержимое **debug\_log** после вызова процедуры.

# Системные представления для процедур и функций

- USER\_PROCEDURES
- ALL\_PROCEDURES
- DBA\_PROCEDURES

# Обработка динамических ошибок

- ....
- EXCEPTION
- [WHEN NO\_DATA\_FOUND       THEN....]
- [WHEN ZERO\_DIVIDE        THEN... ]
- [WHEN TOO\_MANY\_ROWS     THEN ...]
- .....
- [WHEN OTHERS             THEN...]
- 
- END

# Пример (процедура для фиксации динамических ошибок)

- create or replace procedure LogInfo
- ( inInfoMessage in varchar2, inSource in varchar2 )
- is
- **PRAGMA AUTONOMOUS\_TRANSACTION;**
- begin
- insert into debug\_log(id, LogTime, Message, inSource)
- values (seq\_debug\_log.nextval, sysdate, inInfoMessage, inSource);
- commit;
- exception
- when others then
- return;
- end LogInfo;

# Пример (вызовы процедуры LogInfo)

- create or replace procedure Calculate(...)
- is
- .....
- begin
- .....
- LogInfo('A=' || A, 'Calculate');
- .....
- exception
- when others then
- LogInfo(substr(sqlerrm, 1, 100), 'Calculate');
- 
- end;

# Задание 10

- Создайте процедуру для фиксации динамических ошибок.
- Создайте функцию или процедуру, которая может привести к появлению динамической ошибки.
- Спровоцируйте появление и фиксацию динамической ошибки в журнале **debug\_log**.

# Пакеты

- ПАКЕТ - это объект базы данных, который группирует логически связанные типы, . Пакеты обычно состоят из двух частей, спецификации и тела, хотя иногда в теле нет необходимости.
- СПЕЦИФИКАЦИЯ пакета - объявляет типы, переменные, константы и подпрограммы, доступные для использования в пакете.
- ТЕЛО пакета полностью определяет подпрограммы, тем самым реализует спецификацию пакета.

# Определение спецификации и тела пакета

- PACKAGE имя AS -- спецификация (видимая часть)
- -- объявления общих типов и объектов
- -- спецификации подпрограмм
- END [имя];
  
- =====
- PACKAGE BODY имя AS -- тело (скрытая часть)
- -- тела подпрограмм
- END [имя];

# Пример

- `CREATE PACKAGE emp_actions AS` -- спецификация пакета  
    `TYPE EmpRecType IS RECORD (emp_id INTEGER, salary REAL);`  
    `PROCEDURE hire_employee`  
        `(`  
            `ename VARCHAR2,`  
            `job VARCHAR2,`  
            `mgr NUMBER,`  
            `sal NUMBER,`  
            `comm NUMBER,`  
            `deptno NUMBER`  
        `);`  
    `PROCEDURE fire_employee`  
        `(`  
            `emp_id NUMBER`  
        `);`
- `END emp_actions;`

# Пример

- CREATE PACKAGE BODY emp\_actions AS -- тело  
    PROCEDURE hire\_employee  
        (ename VARCHAR2,  
         job VARCHAR2,  
         mgr NUMBER,  
         sal NUMBER,  
         comm NUMBER,  
         deptno NUMBER  
        )  
    IS  
    BEGIN  
        INSERT INTO emp VALUES (empno\_seq.NEXTVAL, ename, job, mgr, SYSDATE, sal, comm, deptno);  
    END hire\_employee;  
    PROCEDURE fire\_employee  
        (  
         emp\_id NUMBER  
        )  
    IS  
    BEGIN  
        DELETE FROM emp WHERE empno = emp\_id;  
    END fire\_employee;
- END emp\_actions;

# Преимущества пакетов

- Модульность
- Облегчение проектирования
- Скрытие информации
- Совместное использование
- Улучшенная производительность
- Облегчение администрирования

# Обращение к содержимому пакета

- имя\_пакета.имя\_типа      имя\_пакета.  
имя\_объекта      имя\_пакета.  
имя\_подпрограммы

# Задание 11

- Создайте пакет EMP\_PACK с набором процедур и функций, выдающим информацию по сотрудникам.
- Создайте пакет DEPT\_PACK с набором процедур и функций, выдающим информацию по департаментам.
- В каждой процедуре и функции пакета предусмотрите вызов процедуры для фиксации динамических ошибок.

# Домашнее задание 1(11 баллов)

- Оформите все задания в виде одного переиспользуемого скрипта.

Результат отправьте по адресу N.Grafeeva@spbu.ru. Тема письма – Modern\_DB\_2015\_job1.

*Примечание: задание должно быть отправлено в течение 2 недель. За более позднее отправление будут сниматься штрафные баллы ( по баллу за каждые 2 недели).*