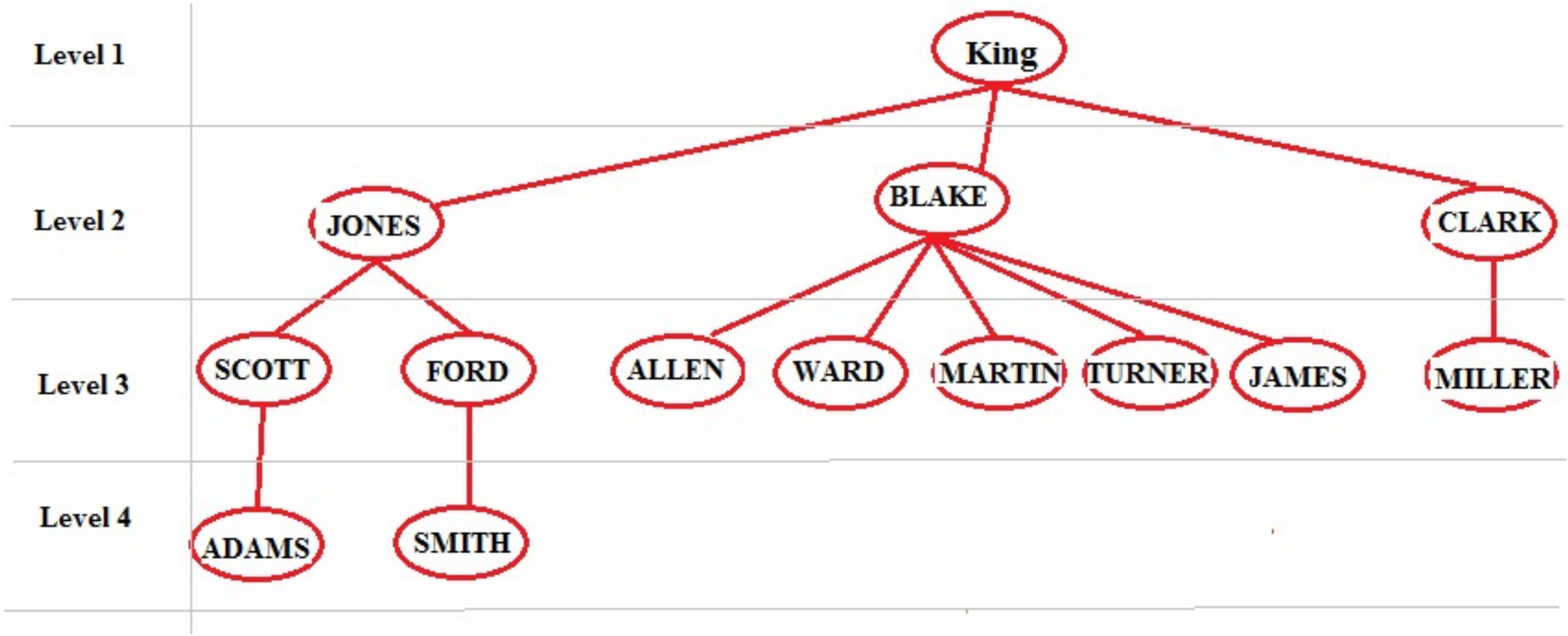
СБЕР УНИВЕРСИТЕТ

Перезапуск Data Engineer

Решаем тест по предыдущей лекции до 18.40

Иерархический (рекурсивный) запрос



Иерархический запрос

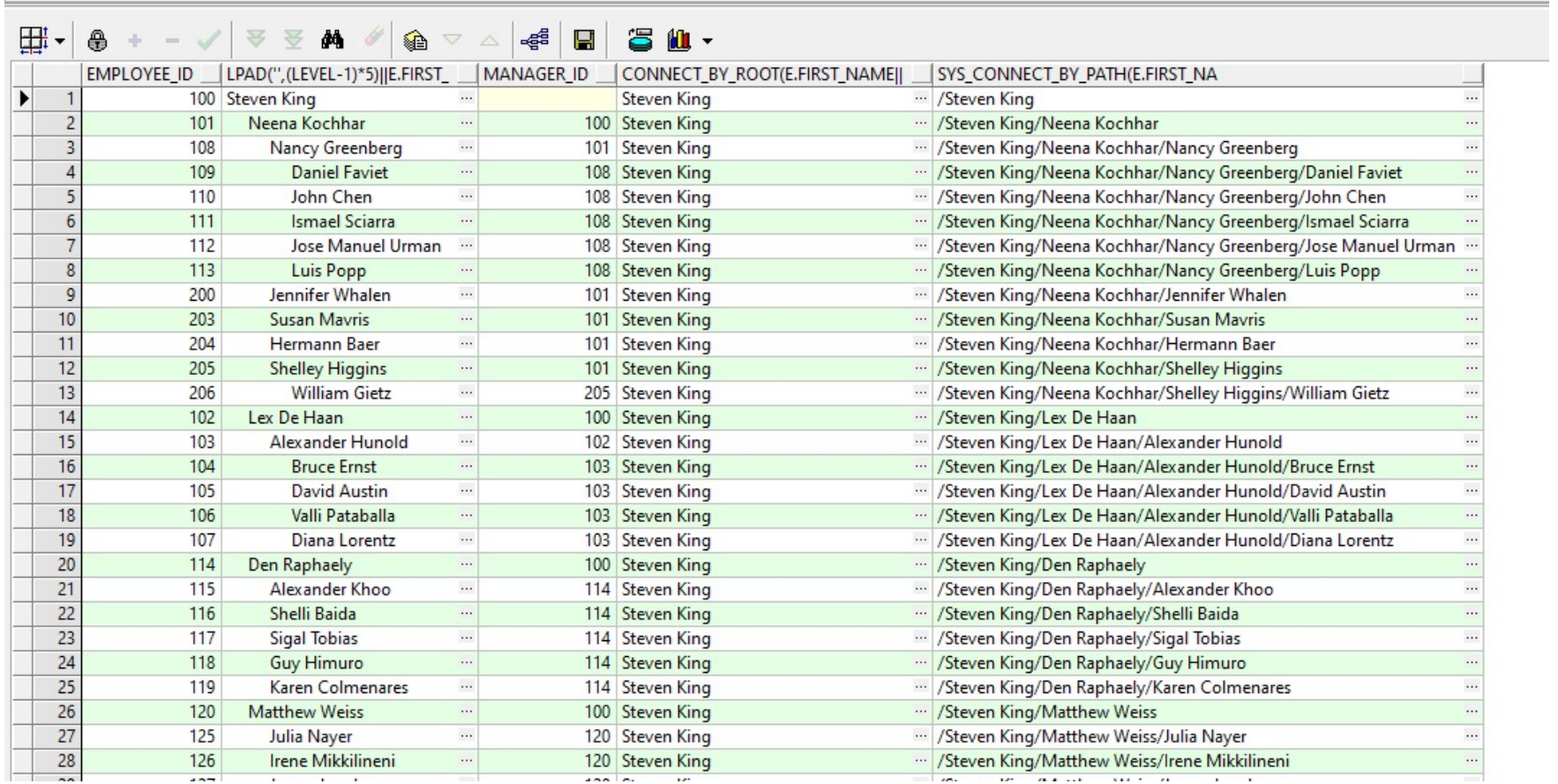
```
select e.employee_id, e.first_name, e.last_name,
e.manager_id, m.first_name, m.last_name
from employees e
    join employees m on e.manager_id =
m.employee_id;
```

```
select e.employee_id, e.first_name, e.last_name, e.manager_id, m.first_name, m.last_name
from employees e
     join employees m on e.manager_id = m.employee_id;
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	MANAGER_ID	FIRST_NAME	LAST_NAME
1	201	Michael	Hartstein	100	Steven	King
2	149	Eleni	Zlotkey	100	Steven	King
3	148	Gerald	Cambrault	100	Steven	King
4	147	Alberto	Errazuriz	100	Steven	King
5	146	Karen	Partners	100	Steven	King
6	145	John	Russell	100	Steven	King
7	124	Kevin	Mourgos	100	Steven	King
8	123	Shanta	Vollman	100	Steven	King
9	122	Payam	Kaufling	100	Steven	King
10	121	Adam	Fripp	100	Steven	King
11	120	Matthew	Weiss	100	Steven	King
12	114	Den	Raphaely	100	Steven	King
13	102	Lex	De Haan	100	Steven	King
14	101	Neena	Kochhar	100	Steven	King
15	205	Shelley	Higgins	101	Neena	Kochhar
16	204	Hermann	Baer	101	Neena	Kochhar
17	203	Susan	Mavris	101	Neena	Kochhar
18	200	Jennifer	Whalen	101	Neena	Kochhar
19	108	Nancy	Greenberg	101	Neena	Kochhar
20	103	Alexander	Hunold	102	Lex	De Haan
21	107	Diana	Lorentz	103	Alexander	Hunold
22	106	Valli	Pataballa	103	Alexander	Hunold
23	105	David	Austin	103	Alexander	Hunold
24	104	Bruce	Ernst	103	Alexander	Hunold
25	113	Luis	Popp	108	Nancy	Greenberg
26	112	Jose Manuel	Urman	108	Nancy	Greenberg
27	111	Ismael	Sciarra	108	Nancy	Greenberg
28	110	John	Chen	108	Nancy	Greenberg
29	109	Daniel	Faviet	108	Nancy	Greenberg
30	119	Karen	Colmenares	114	Den	Raphaely
31	118	Guy	Himuro	114	Den	Raphaely

Иерархический запрос

```
select e.employee_id
      , lpad(' ',(level-1)*5)||e.first_name||' '|| e.last_name
      , e.manager_id
      , connect_by_root(e.first_name||' '|| e.last_name)
      , sys_connect_by_path(e.first_name||' '|| e.last_name, '/')
from employees e
connect by prior employee_id = manager_id
start with manager_id is null;
```



	EMPLOYEE_ID	LPAD('',(LEVEL-1)*5)||E.FIRST_	MANAGER_ID	CONNECT_BY_ROOT(E.FIRST_NAME||	SYS_CONNECT_BY_PATH(E.FIRST_NA
1	100	Steven King		Steven King	/Steven King
2	101	Neena Kochhar	100	Steven King	/Steven King/Neena Kochhar
3	108	Nancy Greenberg	101	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg
4	109	Daniel Faviet	108	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg/Daniel Faviet
5	110	John Chen	108	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg/John Chen
6	111	Ismael Sciarra	108	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg/Ismael Sciarra
7	112	Jose Manuel Urman	108	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg/Jose Manuel Urman
8	113	Luis Popp	108	Steven King	/Steven King/Neena Kochhar/Nancy Greenberg/Luis Popp
9	200	Jennifer Whalen	101	Steven King	/Steven King/Neena Kochhar/Jennifer Whalen
10	203	Susan Mavris	101	Steven King	/Steven King/Neena Kochhar/Susan Mavris
11	204	Hermann Baer	101	Steven King	/Steven King/Neena Kochhar/Hermann Baer
12	205	Shelley Higgins	101	Steven King	/Steven King/Neena Kochhar/Shelley Higgins
13	206	William Gietz	205	Steven King	/Steven King/Neena Kochhar/Shelley Higgins/William Gietz
14	102	Lex De Haan	100	Steven King	/Steven King/Lex De Haan
15	103	Alexander Hunold	102	Steven King	/Steven King/Lex De Haan/Alexander Hunold
16	104	Bruce Ernst	103	Steven King	/Steven King/Lex De Haan/Alexander Hunold/Bruce Ernst
17	105	David Austin	103	Steven King	/Steven King/Lex De Haan/Alexander Hunold/David Austin
18	106	Valli Pataballa	103	Steven King	/Steven King/Lex De Haan/Alexander Hunold/Valli Pataballa
19	107	Diana Lorentz	103	Steven King	/Steven King/Lex De Haan/Alexander Hunold/Diana Lorentz
20	114	Den Raphaely	100	Steven King	/Steven King/Den Raphaely
21	115	Alexander Khoo	114	Steven King	/Steven King/Den Raphaely/Alexander Khoo
22	116	Shelli Baida	114	Steven King	/Steven King/Den Raphaely/Shelli Baida
23	117	Sigal Tobias	114	Steven King	/Steven King/Den Raphaely/Sigal Tobias
24	118	Guy Himuro	114	Steven King	/Steven King/Den Raphaely/Guy Himuro
25	119	Karen Colmenares	114	Steven King	/Steven King/Den Raphaely/Karen Colmenares
26	120	Matthew Weiss	100	Steven King	/Steven King/Matthew Weiss
27	125	Julia Nayer	120	Steven King	/Steven King/Matthew Weiss/Julia Nayer
28	126	Irene Mikkilineni	120	Steven King	/Steven King/Matthew Weiss/Irene Mikkilineni

PL/SQL

PL/SQL — это сокращение от «Procedural Language extensions to the Structured Query Language», что в переводе с английского означает «**процедурные языковые расширения для SQL**».

Язык PL/SQL является собственным расширением языка SQL от Oracle и предлагает функциональность серьезного языка программирования.

PL/SQL

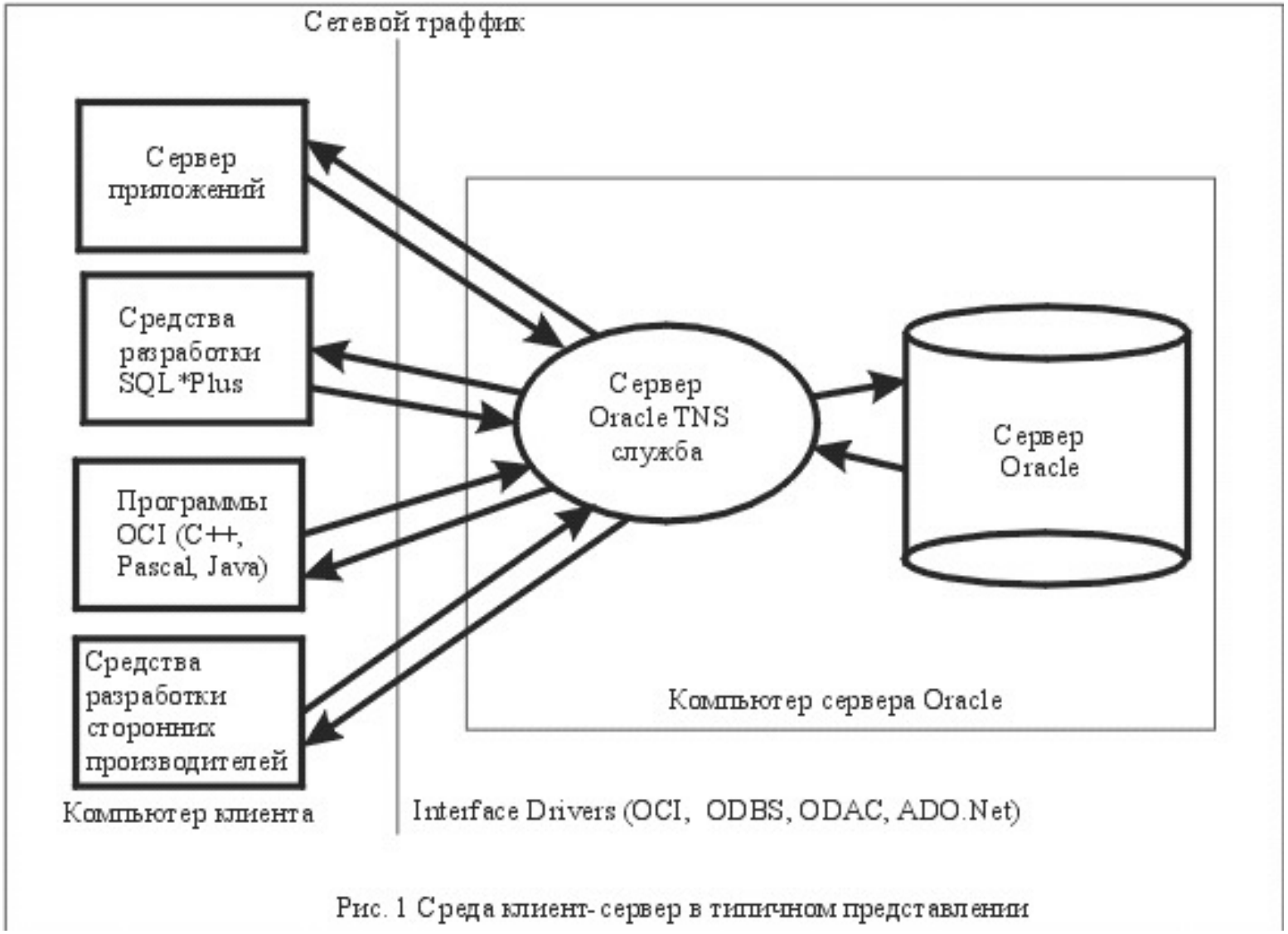


Рис. 1 Среда клиент-сервер в типичном представлении

PL/SQL

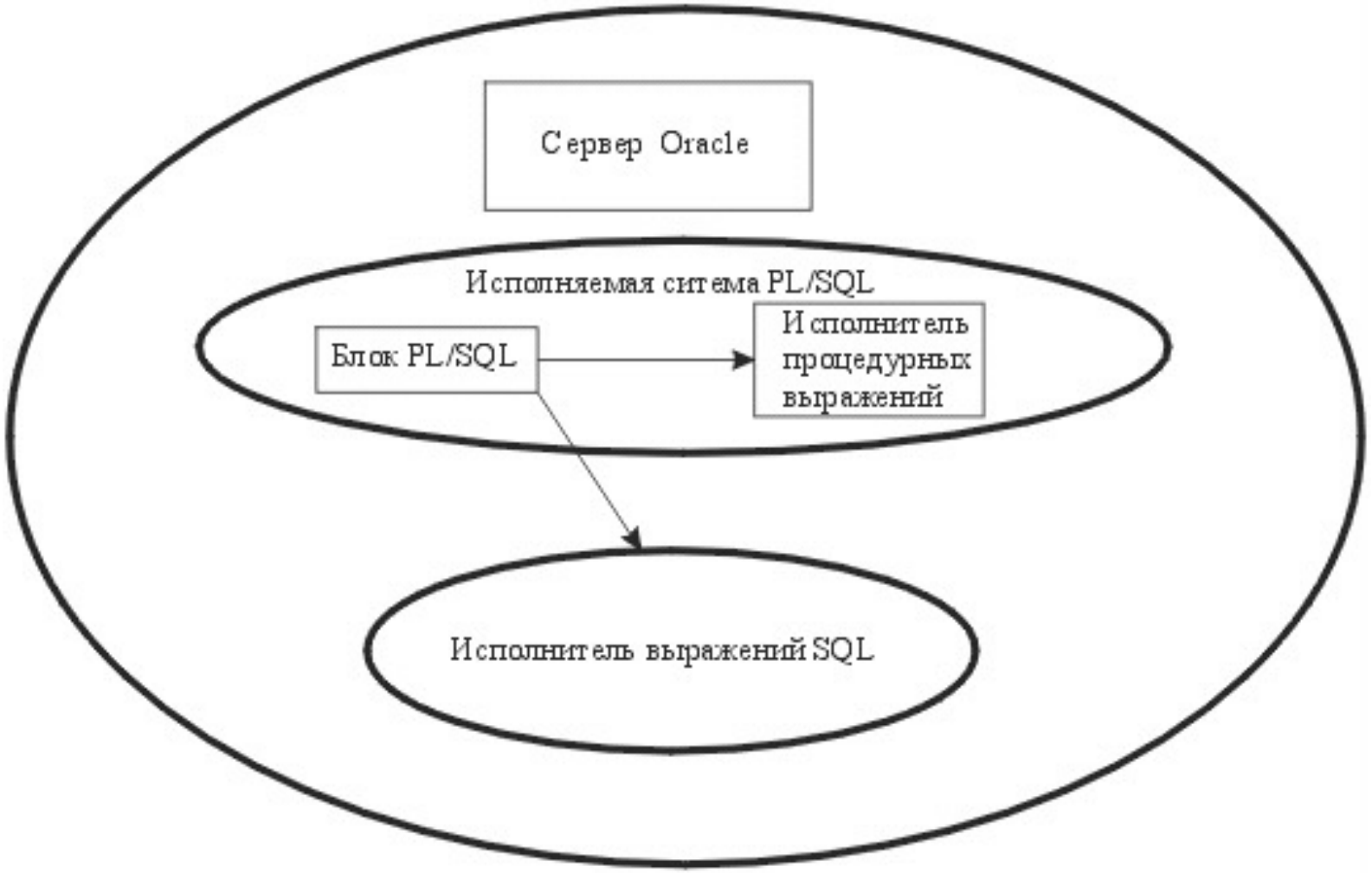


Рис. 2 Система исполнитель языка PL/SQL (является компонентом сервера БД Oracle

PL/SQL

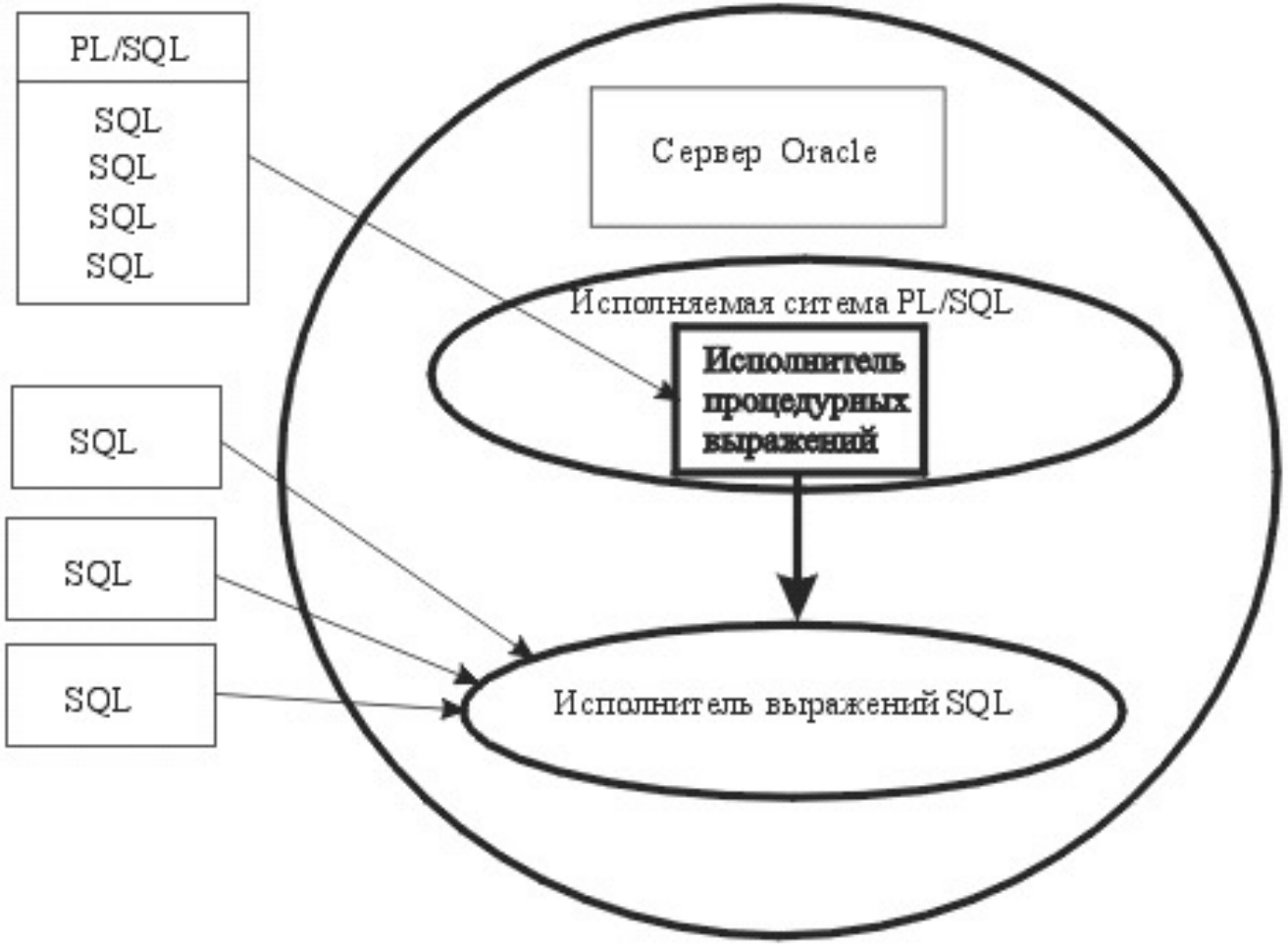


Рис. 3 Группировка SQL кода в единый блок языка PL/SQL, позволяет значительно уменьшить загрузку сети

PL/SQL

Стандартная конструкция неименованного блока:

```
DECLARE
        -- объявления

BEGIN
        -- выполняемый код

EXCEPTION
        -- обработка исключений
END;
```

PL/SQL

```
DECLARE id_c NUMBER(9);
BEGIN
  id_c := 99;
  insert into countries values (id_c, 'olololo', 2);
  COMMIT;
END;
```

PL/SQL

```
IF(некоторое условие справедливо)
THEN
        -- условие справедливо, выполнять это.
ELSE  -- условие не выполняется
        -- выполнять оператор в этой части.
END IF; -- конец условного оператора.
```

Типы данных в PL\SQL

Учимся читать оракловую документацию

https://docs.oracle.com/cd/E11882_01/appdev.112/e25519/datatypes.htm#LNPLS003

Пример скрипта на PL\SQL

```
DECLARE
TYPE emp_det IS RECORD
(
EMP_NO NUMBER,
EMP_NAME VARCHAR2(150),
MANAGER NUMBER,
SALARY NUMBER
);
TYPE emp_det_tbl IS TABLE OF emp_det;
g_emp_rec emp_det_tbl:= emp_det_tbl();

BEGIN
INSERT INTO employees (employee_id, last_name, salary, manager_id) VALUES (1000,'AAA',25000,100);
INSERT INTO employees (employee_id, last_name, salary, manager_id) VALUES (1001,'XXX',10000,100);
INSERT INTO employees (employee_id, last_name, salary, manager_id) VALUES (1002,'YYY',15000,100);
INSERT INTO employees (employee_id, last_name, salary, manager_id) VALUES (1003,'ZZZ',7500,100);
COMMIT;

SELECT employee_id,last_name,manager_id,salary BULK COLLECT INTO g_emp_rec
FROM employees
where employee_id in (1000,1001,1002,1003);

dbms_output.put_line ('Employee Detail');
FOR i IN g_emp_rec.FIRST..g_emp_rec.LAST
LOOP
dbms_output.put_line('Employee Number:'||g_emp_rec(i).emp_no);
dbms_output.put_line('Employee Name:'||g_emp_rec(i).emp_name);
dbms_output.put_line('Employee Salary:'|| g_emp_rec(i).salary);
dbms_output.put_line('Employee Manager Number:'||g_emp_rec(i).manager);
dbms_output.put_line('****************');
END LOOP;
END;
```

Связанные переменные в PL\SQL

Если бы мне пришлось писать книгу о том, как создавать немасштабируемые приложения Oracle, первая и единственная ее глава называлась бы «Не используйте связываемые переменные».

Том Кайт, вице-президент Oracle

Связываемые переменные, они же prepared statements, они же подготовленные выражения — это часть функциональности SQL-баз данных, предназначенная для отделения данных запроса и собственно выполняемого SQL-запроса.

Связанные переменные в PL\SQL

Указывается через : или &

```
DECLARE
  s pls_integer;
BEGIN
s:=50;
   IF (s>&g) and (s>:e) THEN dbms_output.put_line('S bigger');
   ELSE dbms_output.put_line('S lower');
   END IF;
END;
```

```
select *
from employees e
where e.employee_id = :6;
```

Циклы

1. Безусловные циклы (выполняемые бесконечно)
2. Интерактивные циклы (**FOR**)
3. Условные циклы (**WHILE**)

Выход из циклов

1. EXIT - Безусловный выход из цикла. Используется посредством применения оператора **IF**.
2. EXIT WHEN - Выход при выполнении условия.
3. GOTO - Выход из цикла во внешний контекст.

Самый простой тип цикла в языке **PL/SQL**:

```
LOOP
    NULL;
END LOOP
```

LOOP

```
DECLARE
        i NUMBER := 0;

BEGIN

        LOOP                        -- start loop 1
        i := i + 1;
        IF (i >= 100) THEN
                i := 0;
                EXIT;               -- exit when i >= 0
        END IF;
        END LOOP;                           -- end loop 1
END;
```

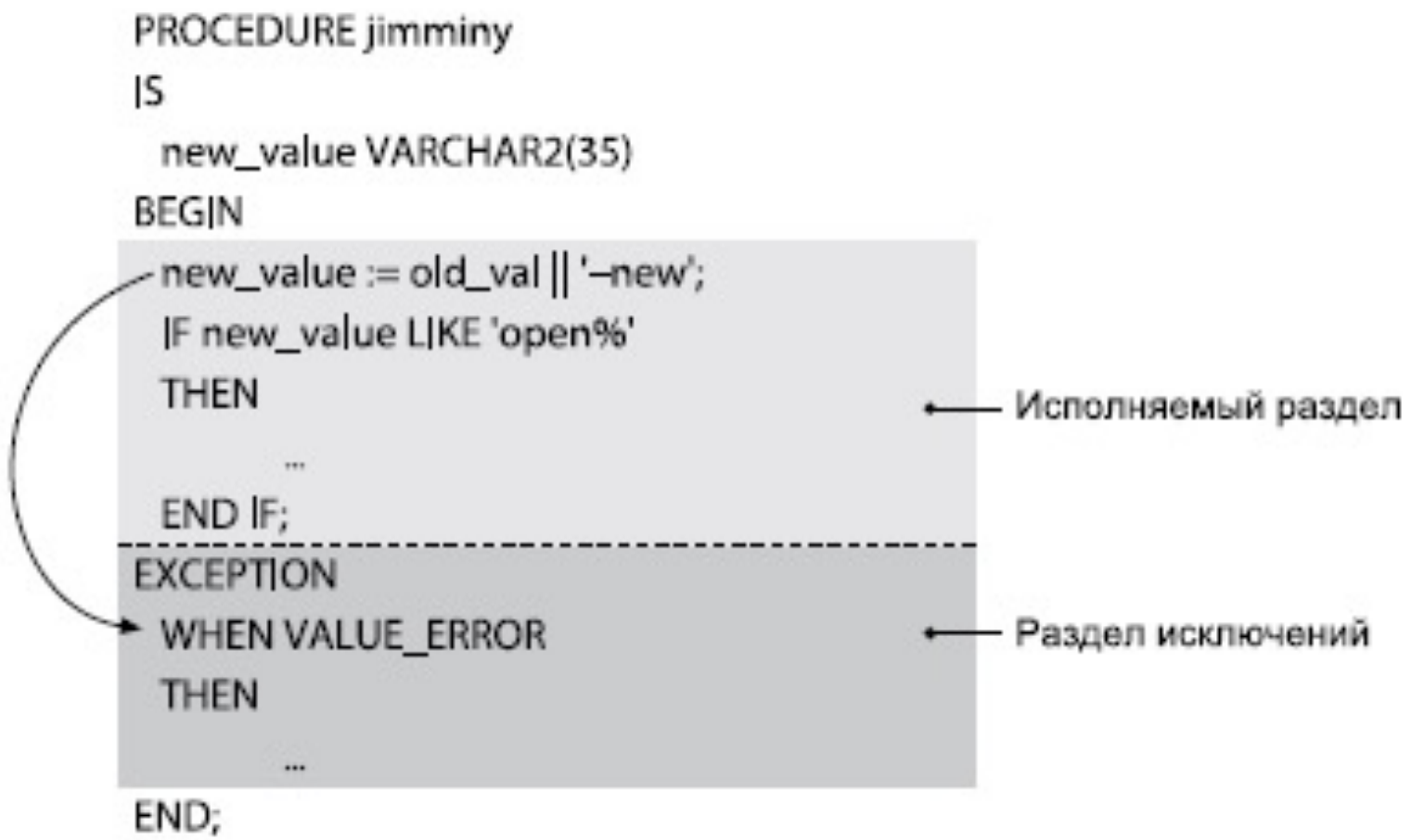
WHILE (цикл с предусловием)

```
DECLARE
        k NUMBER := 0;

BEGIN

        WHILE (k < 10) LOOP
                k := k + 1;
        END LOOP;

END;
```

FOR (фиксированное количество итераций)

```
DECLARE
        s NUMBER := 0;

BEGIN
        FOR i IN 1..20 LOOP
         IF(MOD(i, 2) = 1) THEN
                DBMS_OUTPUT.put_line(TO_CHAR(i)||' is even!');
                s := i;
         END IF;
        END LOOP;
        DBMS_OUTPUT.put_line('last odd number was '||TO_CHAR(s));
END;
```

Исключения


PROCEDURE jimminy
IS
new_value VARCHAR2(35)
BEGIN
new_value := old_val || '–new';
IF new_value LIKE 'open%'
THEN
...
END IF;
EXCEPTION
WHEN VALUE_ERROR
THEN
...
END;
Исполняемый раздел
Раздел исключений

Исключения

```
CREATE [OR REPLACE] PROCEDURE procedure_name [ (parameter [,parameter]) ]
IS
[declaration_section] exception_name EXCEPTION;

BEGIN

  executable_section

RAISE exception_name;   -- Вызвать исключение

EXCEPTION

    WHEN exception_name THEN  [statements]

    WHEN OTHERS THEN  [statements]

END [procedure_name];
```

Исключения

```
CREATE OR REPLACE PROCEDURE add_new_order
  (order_id_in IN NUMBER, prod_id in NUMBER, sales_in IN NUMBER)
IS
  no_sales EXCEPTION;
BEGIN
  IF sales_in = 0 THEN RAISE no_sales;
  ELSE
    INSERT INTO orders (id, productid, customerid, createdat, productcount, price)
    VALUES   ( order_id_in, prod_id, 101,to_date('10.04.2021','DD.MM.YYYY'),sales_in, 100);
    COMMIT;
  END IF;
EXCEPTION
  WHEN no_sales THEN
    raise_application_error (-20001,'You should have Order to close position');
  WHEN OTHERS THEN
    raise_application_error (-20002,'Some error happed!!!');
END;
```