

# Методы сбора, хранения, обработки и анализа данных

Лекция 5

Преобразование данных в SQL

# Типичные задачи

- Использование фразы MODEL
- Применение оператора MERGE
- Применение операторов PIVOT и UNPIVOT

# Оператор MERGE

- Типичные задачи:
  - Необходимость слияния старых и новых данных в случае их расхождения
- Заменяет INSERT и UPDATE
- Требуется commit / rollback для фиксации/отката

# Оператор MERGE – пример

- Есть таблица со списком сотрудников emp
- Необходимо произвести ее слияние с данными из emp\_import

```
CREATE TABLE EMP_IMPORT (  
  EMPNO      NUMBER(4) NOT NULL,  
  ENAME      VARCHAR2(10) NOT NULL,  
  JOB        VARCHAR2(9),  
  MGR        NUMBER(4),  
  HIREDATE   DATE,  
  SAL        NUMBER(7,2),  
  COMM       NUMBER(7,2),  
  DEPTNO     NUMBER(2)  
);  
  
INSERT INTO emp_import VALUES (7839, 'KING', 'PRESIDENT', NULL, '17-NOV-01', 5000, NULL, 10);  
INSERT INTO emp_import VALUES (7698, 'BLAKE', 'MANAGER', 7839, '1-MAY-01', 2850, NULL, 30);  
INSERT INTO emp_import VALUES (7782, 'CLARK', 'MANAGER', 7839, '9-JUN-01', 2450, NULL, 10);  
INSERT INTO emp_import VALUES (8654, 'MART', 'SALESMAN', 7698, '28-SEP-01', 1250, 400, 30);  
INSERT INTO emp_import VALUES (8499, 'ALIEN', 'SALESMAN', 7698, '20-FEB-81', 1600, 300, 30);  
INSERT INTO emp_import VALUES (8844, 'TURNUR', 'SALESMAN', 7698, '8-SEP-81', 1500, 0, 30);  
COMMIT;
```

# Оператор MERGE – пример

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839	KING	PRESIDENT	(null)	17.11.81	5000	(null)	10
2	7698	BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
3	7782	CLARK	MANAGER	7839	09.06.81	2450	(null)	10
4	7566	JONES	MANAGER	7839	02.04.81	2975	(null)	20
5	7654	MARTIN	SALESMAN	7698	28.09.81	1250	400	30
6	7499	ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7	7844	TURNER	SALESMAN	7698	08.09.81	1500	0	30
8	7900	JAMES	CLERK	7698	03.12.81	950	(null)	30
9	7521	WARD	SALESMAN	7698	22.02.81	1250	500	30
10	7902	FORD	ANALYST	7566	03.12.81	3000	(null)	20
11	7369	SMITH	CLERK	7902	17.12.80	800	(null)	20
12	7788	SCOTT	ANALYST	7566	09.12.82	3000	(null)	20
13	7876	ADAMS	CLERK	7788	12.01.83	1100	(null)	20
14	7934	MILLER	CLERK	7782	23.01.82	1300	(null)	10

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839	KING	PRESIDENT	(null)	17.11.01	5000	(null)	10
2	7698	BLAKE	MANAGER	7839	01.05.01	2850	(null)	30
3	7782	CLARK	MANAGER	7839	09.06.01	2450	(null)	10
4	8654	MART	SALESMAN	7698	28.09.01	1250	400	30
5	8499	ALIEN	SALESMAN	7698	20.02.81	1600	300	30
6	8844	TURNUR	SALESMAN	7698	08.09.81	1500	0	30

# Оператор MERGE – пример

```
MERGE INTO emp e
  USING emp_import i
  ON (e.empno = i.empno)
WHEN MATCHED THEN
  UPDATE SET
    e.ename = i.ename,
    e.job = i.job,
    e.mgr = i.mgr,
    e.hiredate = i.hiredate,
    e.sal = i.sal,
    e.comm = i.comm,
    e.deptno = i.deptno
WHEN NOT MATCHED THEN
  INSERT (e.empno, e.ename, e.job, e.mgr, e.hiredate, e.sal, e.comm, e.deptno)
  VALUES (i.empno, i.ename, i.job, i.mgr, i.hiredate, i.sal, i.comm, i.deptno);
```

6 rows merged.

# Оператор MERGE – пример

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839 KING	PRESIDENT	(null)	17.11.81	5000	(null)	10
2	7698 BLAKE	MANAGER	7839	01.05.81	2850	(null)	30
3	7782 CLARK	MANAGER	7839	09.06.81	2450	(null)	10
4	7566 JONES	MANAGER	7839	02.04.81	2975	(null)	20
5	7654 MARTIN	SALESMAN	7698	28.09.81	1250	400	30
6	7499 ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7	7844 TURNER	SALESMAN	7698	08.09.81	1500	0	30
8	7900 JAMES	CLERK	7698	03.12.81	950	(null)	30
9	7521 WARD	SALESMAN	7698	22.02.81	1250	500	30
10	7902 FORD	ANALYST	7566	03.12.81	3000	(null)	20
11	7369 SMITH	CLERK	7902	17.12.80	800	(null)	20
12	7788 SCOTT	ANALYST	7566	09.12.82	3000	(null)	20
13	7876 ADAMS	CLERK	7788	12.01.83	1100	(null)	20
14	7934 MILLER	CLERK	7782	23.01.82	1300	(null)	10

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839 KING	PRESIDENT	(null)	17.11.01	5000	(null)	10
2	7698 BLAKE	MANAGER	7839	01.05.01	2850	(null)	30
3	7782 CLARK	MANAGER	7839	09.06.01	2450	(null)	10
4	8654 MART	SALESMAN	7698	28.09.01	1250	400	30
5	8499 ALIEN	SALESMAN	7698	20.02.81	1600	300	30
6	8844 TURNUR	SALESMAN	7698	08.09.81	1500	0	30

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7839 KING	PRESIDENT	(null)	17.11.01	5000	(null)	10
2	7698 BLAKE	MANAGER	7839	01.05.01	2850	(null)	30
3	7782 CLARK	MANAGER	7839	09.06.01	2450	(null)	10
4	7566 JONES	MANAGER	7839	02.04.81	2975	(null)	20
5	7654 MARTIN	SALESMAN	7698	28.09.81	1250	400	30
6	7499 ALLEN	SALESMAN	7698	20.02.81	1600	300	30
7	7844 TURNER	SALESMAN	7698	08.09.81	1500	0	30
8	7900 JAMES	CLERK	7698	03.12.81	950	(null)	30
9	7521 WARD	SALESMAN	7698	22.02.81	1250	500	30
10	7902 FORD	ANALYST	7566	03.12.81	3000	(null)	20
11	7369 SMITH	CLERK	7902	17.12.80	800	(null)	20
12	7788 SCOTT	ANALYST	7566	09.12.82	3000	(null)	20
13	7876 ADAMS	CLERK	7788	12.01.83	1100	(null)	20
14	7934 MILLER	CLERK	7782	23.01.82	1300	(null)	10
15	8654 MART	SALESMAN	7698	28.09.01	1250	400	30
16	8499 ALIEN	SALESMAN	7698	20.02.81	1600	300	30
17	8844 TURNUR	SALESMAN	7698	08.09.81	1500	0	30

# Ключевое слово PIVOT

- Типичные задачи:
  - Необходимость получения сводных отчетов
  - Подсчет итогов и промежуточных итогов
- Выполняется обращение строк в столбцы



# Пример – PIVOT

```
CREATE TABLE Sales.SalesByMonth
(
    year char(4),
    month char(3),
    amount money,
    primary key (year, month)
)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Jan', 789.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Apr', 778.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'May', 78.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Jun', 9.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Jul', 987.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Aug', 866.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Sep', 7787.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Oct', 85576.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Nov', 855.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2017', 'Dec', 5878.0000);
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES ('2016', 'Jan', 7.0000);
```

# Пример – PIVOT

```
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES( 2017 , NOV , 855.0000)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES('2017', 'Dec', 5878.0000)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES('2016', 'Jan', 7.0000)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES('2016', 'Feb', 6868.0000)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES('2016', 'Mar', 688.0000)
INSERT INTO Sales.SalesByMonth (year, month, amount) VALUES('2016', 'Apr', 9897.0000)
```

```
SELECT Year,[Jan],[Feb],[Mar],[Apr],[May],[Jun],[Jul],[Aug],[Sep],[Oct],[Nov],[Dec]
FROM (
  SELECT year, amount, month
  FROM Sales.SalesByMonth ) AS SalesByMonth
PIVOT ( SUM(amount) FOR month IN
  ([Jan] , [Feb], [Mar], [Apr], [May], [Jun],
  [Jul],[Aug],[Sep],[Oct],[Nov],[Dec]) ) AS ourPivot
ORDER BY Year;
```

Year	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
2016	7.00	6868.00	688.00	9897.00	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2017	789.00	NULL	NULL	778.00	78.00	9.00	987.00	866.00	7787.00	85576.00	855.00	5878.00

# Обращение данных – PIVOT

```
-----  
DECLARE @months nvarchar(max)  
SELECT @months =  
    STUFF( (SELECT distinct ',[' + month + ']'  
    FROM Sales.SalesByMonth for xml path('') ), 1,1, '');  
  
DECLARE @SQL nvarchar(max);  
SELECT @SQL = N'SELECT Year, ' + @months +  
    ' FROM ( SELECT year, amount, month FROM Sales.SalesByMonth) AS SalesByMonth  
    PIVOT (SUM(amount) FOR month IN ( ' + @months + ')) AS PivotTable ORDER BY Year';  
  
EXEC sp_executesql @SQL;
```

Results							
Year	Apr	Aug	Jan	Jul	Jun	May	Sep
2017	778.00	866.00	789.00	987.00	9.00	78.00	7787.00

# UNPIVOT

```
select * FROM Population;
```

%																			
Результаты																			
Сообщения																			
region	1996	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018
Брестская	1494.3	1477.4	1469.8	1461	1450.2	1439.3	1426.8	1417.8	1409.7	1404.5	1399.2	1394.8	1391.4	1390.4	1388.5	1388.9	1387	1386.4	1384.5
Витебская	1415.8	1354.6	1340	1323.3	1306.4	1289.5	1273.8	1259.4	1247.3	1237.5	1229.4	1221.8	1214.1	1208	1202.1	1198.5	1193.5	1188	1180.2
г. Минск	1669.5	1689.9	1699.4	1709.7	1722.1	1744.6	1758.8	1775.5	1794.7	1814.3	1843.7	1864.1	1885.1	1901	1921.8	1938.2	1959.8	1974.8	1982.4
Гомельская	1566.3	1532.2	1523.1	1509.7	1496.8	1484.2	1471	1459.4	1449.5	1443.2	1439.2	1435	1429.7	1427.7	1425.6	1424	1422.9	1420.7	1415.7
Гродненская	1204.1	1170.1	1160.2	1147.9	1135.4	1122.1	1108	1096.2	1086	1076.7	1071.3	1065.9	1061.2	1058.4	1054.9	1052.6	1050.1	1047.4	1043.7
Минская	1589.1	1535.4	1521.6	1505.8	1491.7	1470.5	1457.6	1447.6	1440.7	1431.1	1419.9	1411.5	1403.6	1401.9	1402.7	1407.9	1417.4	1423.1	1426.5
Могилевская	1238.2	1197.1	1186.3	1173.3	1160.2	1147.3	1134.4	1123.6	1114.5	1106.3	1097.3	1088.1	1080.1	1076.4	1072.6	1070.8	1067.7	1064.3	1058.8

```

create view v2
as
select region, year_count, pinfo
FROM
    (select region,
        "2001",
        "2002",
        "2003",
        "2004",
        "2005",
        "2006",
        "2007",
        "2008",
        "2009",
        "2010",
        "2011",
        "2012",
        "2013",
        "2014",
        "2015",
        "2016",
        "2017"
        from Population) as st1
UNPIVOT
( pinfo FOR year_count IN ( "2001",
    "2002",
    "2003",
    "2004",
    "2005",
    "2006",
    "2007",
    "2008",
    "2009",
    "2010",
    "2011",
    "2012",
    "2013",
    "2014",
    "2015",
    "2016",
    "2017")) as st2;

```

select \* from v1;

100 %

Результаты Сообщения

	region	year_count	pinfo
1	Брестская	2001	21.4
2	Брестская	2002	21.7
3	Брестская	2003	22.1
4	Брестская	2004	22.5
5	Брестская	2005	23.1
6	Брестская	2006	23.4
7	Брестская	2007	23.8
8	Брестская	2008	24
9	Брестская	2009	24.6
10	Брестская	2010	25.2
11	Брестская	2011	25.7
12	Брестская	2012	26
13	Брестская	2013	26.3
14	Брестская	2014	26.7
15	Брестская	2015	27.3
16	Брестская	2016	27.6
17	Брестская	2017	28.1
18	Витебская	2001	22.2
19	Витебская	2002	22.6
20	Витебская	2003	22.9
21	Витебская	2004	23.3
22	Витебская	2005	23.2
23	Витебская	2006	23.6
24	Витебская	2007	23.8
25	Витебская	2008	24.2
26	Витебская	2009	24.6

Запр... | 172.16.193.87 (11.0 RTM) | blinova (55) | OLYMP2019 | 00:00:00 | 119 строк

# PIVOT

```
CREATE TABLE visit (visit_date DATE, visitor VARCHAR2(10), country VARCHAR2(10));

INSERT INTO visit VALUES (to_date('01-01-2022', 'DD-MM-YYYY'), 'KING', 'GERMANY');
INSERT INTO visit VALUES (to_date('01-02-2022', 'DD-MM-YYYY'), 'KING', 'FRANCE');
INSERT INTO visit VALUES (to_date('01-06-2022', 'DD-MM-YYYY'), 'BLAKE', 'GERMANY');
INSERT INTO visit VALUES (to_date('01-07-2022', 'DD-MM-YYYY'), 'BLAKE', 'FRANCE');
INSERT INTO visit VALUES (to_date('01-03-2022', 'DD-MM-YYYY'), 'KING', 'SUDAN');
INSERT INTO visit VALUES (to_date('01-04-2022', 'DD-MM-YYYY'), 'KING', 'EPHIOPIA');
INSERT INTO visit VALUES (to_date('01-05-2022', 'DD-MM-YYYY'), 'CLARK', 'EPHIOPIA');
INSERT INTO visit VALUES (to_date('01-08-2022', 'DD-MM-YYYY'), 'CLARK', 'SUDAN');

COMMIT;
```

	VISIT_DATE	VISITOR	COUNTRY
1	01.01.22	KING	GERMANY
2	01.02.22	KING	FRANCE
3	01.06.22	BLAKE	GERMANY
4	01.07.22	BLAKE	FRANCE
5	01.03.22	KING	SUDAN
6	01.04.22	KING	EPHIOPIA
7	01.05.22	CLARK	EPHIOPIA
8	01.08.22	CLARK	SUDAN

# PIVOT

```
-- pivot
SELECT * FROM
(
  SELECT visitor, country, visit_date
  FROM visit
)
PIVOT
(
  COUNT(visit_date)
  FOR country IN ('GERMANY', 'FRANCE', 'SUDAN', 'EPHIOPIA')
)
ORDER BY visitor;
```

	VISITOR	'GERMANY'	'FRANCE'	'SUDAN'	'EPHIOPIA'
1	BLAKE	1	1	0	0
2	CLARK	0	0	1	1
3	KING	1	1	1	1

```
SELECT * FROM visit
PIVOT
(
  COUNT(visit_date)
  FOR country IN ('GERMANY', 'FRANCE', 'SUDAN', 'EPHIOPIA')
)
ORDER BY visitor;
```

# PIVOT

- Нельзя сформировать список столбцов подзапросом

```
SELECT LISTAGG( ' '||country , ',' ) WITHIN GROUP (ORDER BY country)|| ' '
FROM ( select distinct country from visit);
```

LISTAGG(' '  COUNTRY,',')WITHINGROUP(ORDERBYCOUNTRY)  '
1 'EPHIOPIA, 'FRANCE, 'GERMANY, 'SUDAN'

```
SELECT * FROM
(
  SELECT visitor, country, visit_date
  FROM visit
)
PIVOT
(
  COUNT(visit_date)
  FOR country IN (SELECT LISTAGG( ' '||country , ',' ) WITHIN GROUP (ORDER BY country)|| ' '
    FROM ( SELECT DISTINCT country FROM visit))
)
ORDER BY visitor; ---- error
```



# PIVOT

- Можно сформировать динамическим SQL

```
----- dynamic sql
SET autoprint ON;
var cur refcursor;
DECLARE
  cols          VARCHAR2(4000):='';
  ex_string VARCHAR2(4000):='';
BEGIN
  SELECT listagg( ' '
    ||country , ' ', ' ' ) WITHIN GROUP (
ORDER BY country)
    ||' '
  INTO cols
  FROM
    ( SELECT DISTINCT country FROM visit
    );
  dbms_output.put_line(cols);
  ex_string := 'SELECT * FROM visit PIVOT (COUNT(visit_date) FOR country IN ('||cols||')) ORDER BY visitor';
  dbms_output.put_line(ex_string);
  OPEN :cur FOR ex_string;
END;
```

VISITOR	'EPHIOPIA'	'FRANCE'	'GERMANY'	'SUDAN'
BLAKE	0	1	1	0
CLARK	1	0	0	1
KING	1	1	1	1

# PIVOT

- Или через временную таблицу

```
----- with temp table
```

```
DECLARE
```

```
cols          VARCHAR2(4000):='';
```

```
cols_for_table VARCHAR2(4000):='';
```

```
ex_string     VARCHAR2(4000):='';
```

```
BEGIN
```

```
SELECT LISTAGG( country , ' varchar2(10),' ) WITHIN GROUP (
ORDER BY country)
```

```
INTO cols_for_table
```

```
FROM
```

```
( SELECT DISTINCT country FROM visit
);
```

```
cols_for_table:='create global temporary table country(visitor varchar2(10), '||cols_for_table||' varchar2(10))';
```

```
EXECUTE IMMEDIATE cols_for_table;
```

```
SELECT LISTAGG( '''
```

```
||country , ',' ) WITHIN GROUP (
ORDER BY country)
```

```
||'''
```

```
INTO cols
```

```
FROM
```

```
( SELECT DISTINCT country FROM visit
);
```

```
ex_string := 'insert into country
```

```
SELECT * FROM visit
```

```
PIVOT (COUNT(visit_date) FOR country IN ('||cols||'))
```

```
ORDER BY visitor';
```

```
EXECUTE IMMEDIATE ex_string;
```

```
END;
```

	VISITOR	EPHIOPIA	FRANCE	GERMANY	SUDAN
1	BLAKE	0	1	1	0
2	CLARK	1	0	0	1
3	KING	1	1	1	1

	VISITOR	BELARUS	EPHIOPIA	FRANCE	GERMANY	POLAND	RUSSIA	SUDAN	US
1	BLAKE	0	0	1	1	0	1	0	0
2	CLARK	0	1	0	0	1	0	1	0
3	JOBS	0	0	0	0	0	0	0	1
4	KING	1	1	1	1	0	0	1	0

# MODEL

- Создает многомерный массив на основе результатов запроса
- Позволяет анализировать данные в рамках многомерного куба
- Применяет правила для вычисления новых значений
- Нет передачи больших наборов данных в PL/SQL

# MODEL

- Столбцы запроса разделяются на три группы:
  - Столбцы секционирования – PARTITION BY
  - Измерения – DIMENSIONS
  - Меры – MEASURES

# MODEL

- Столбцы секционирования – PARTITION BY
- Определяют логические блоки результирующего набора аналогично секции PARTITION BY аналитических функций

# MODEL

- Измерения – DIMENSIONS
- Определяют многомерный массив и используются для идентификации ячеек

# MODEL

- Меры – MEASURES
- Меры содержат числовые значения, которые необходимо вычислить

# MODEL

- Рассмотрим на примере

	YEAR	MONTH	PRD_TYPE_ID	EMP_ID	AMOUNT
1	2022	1	1	21	10034,84
2	2022	2	1	21	15144,65
3	2022	3	1	21	20137,83
4	2022	4	1	21	25057,45
5	2022	5	1	21	17214,56
6	2022	6	1	21	15564,64
7	2022	7	1	21	12654,84
8	2022	8	1	21	17434,82
9	2022	9	1	21	19854,57
10	2022	10	1	21	21754,19
11	2022	11	1	21	13029,73
12	2022	12	1	21	10034,84
13	2022	1	2	21	1034,84
14	2022	2	2	21	1544,65
15	2022	3	2	21	2037,83
16	2022	4	2	21	2557,45
17	2022	5	2	21	1714,56
18	2022	6	2	21	1564,64
19	2022	7	2	21	1264,84
20	2022	8	2	21	1734,82
21	2022	9	2	21	1854,57
22	2022	10	2	21	2754,19
23	2022	11	2	21	1329,73

-- история продаж за 2022

```
select * from all_sales;
```

-- 2022

```
select distinct year from all_sales;
```

-- 1-12

```
select distinct month from all_sales;
```

-- 1-5

```
select distinct prd_type_id from all_sales;
```

-- 21-26

```
select distinct emp_id from all_sales;
```



# MODEL

- Есть фактические данные о продажах за 2022 год
- Необходимо дать сотрудникам план продаж на:
  - 1) январь 2023,
  - 2) на январь - март 2023,
  - 3) на весь 2023;
- причем этот план продаж строится для каждого сотрудника по всем товарам и может быть одним из четырех вариантов:
  - а) такой же, как и в январе 2022;
  - б) на 10% выше, чем январе 2022 на товары 1-3, а на остальные такой же;
  - в) должен быть равен среднему за январь-март 2022;
  - г) должен быть максимум на 10% меньше продаж того же товара самого лучшего продавца за январь 2022.

# 1 – данные по одному столбцу

```
-- 1 - формируем один столбец по правилам  
-- 1a такой же, как и в январе 2022  
select prd_type_id, emp_id, 1, 2023, amount plan_sales  
from all_sales  
where year = 2022 and month = 1;
```

	PRD_TYPE_ID	EMP_ID	1	2023	PLAN_SALES
1	1	21	1	2023	10034,84
2	2	21	1	2023	1034,84
3	3	21	1	2023	6034,84
4	4	21	1	2023	3034,84
5	1	22	1	2023	11034,84
6	2	22	1	2023	1234,84
7	3	22	1	2023	6334,84
8	4	22	1	2023	3334,84
9	1	23	1	2023	4034,84
10	2	23	1	2023	1234,84
11	3	23	1	2023	6334,84
12	4	23	1	2023	3334,84
13	1	24	1	2023	7034,84
14	2	24	1	2023	1934,84
15	3	24	1	2023	2334,84
16	4	24	1	2023	3364,84

# 1 – данные по одному столбцу

```
-- 16 на 10% выше, чем январе 2022 на товары 1-3, а на остальные такой же
select prd_type_id, emp_id, 1, 2023, amount *
  (CASE
    WHEN prd_type_id between 1 and 3 THEN 1.10
    ELSE 1
  END) plan_sales
from all_sales
where year = 2022 and month = 1;
```

	PRD_TYPE_ID	EMP_ID	1	2023	PLAN_SALES
1	1	21	1	2023	11038,324
2	2	21	1	2023	1138,324
3	3	21	1	2023	6638,324
4	4	21	1	2023	3034,84
5	1	22	1	2023	12138,324
6	2	22	1	2023	1358,324
7	3	22	1	2023	6968,324
8	4	22	1	2023	3334,84
9	1	23	1	2023	4438,324
10	2	23	1	2023	1358,324
11	3	23	1	2023	6968,324
12	4	23	1	2023	3334,84
13	1	24	1	2023	7738,324
14	2	24	1	2023	2128,324
15	3	24	1	2023	2568,324
16	4	24	1	2023	3364,84
17	1	25	1	2023	1358,324

# 1 – данные по одному столбцу

--lv должен быть равен среднему за январь-март 2022

```
with avg_in_1_3_2022
```

```
as
```

```
(
```

```
    select emp_id, prd_type_id, trunc(avg(amount)) avg_1_3
```

```
    from all_sales
```

```
    where year = 2022 and month between 1 and 3
```

```
    group by emp_id, prd_type_id
```

```
    order by emp_id, prd_type_id
```

```
)
```

```
select avg_in_1_3_2022.emp_id,
```

```
avg_in_1_3_2022.prd_type_id,
```

```
1,
```

```
2023,
```

```
avg_in_1_3_2022.avg_1_3 plan_sales
```

```
from avg_in_1_3_2022;
```

	EMP_ID	PRD_TYPE_ID	1	2023	PLAN_SALES
1	21	1	1	2023	15105
2	21	2	1	2023	1539
3	21	3	1	2023	3505
4	21	4	1	2023	3839
5	21	5	1	2023	(null)
6	22	1	1	2023	17105
7	22	2	1	2023	1605
8	22	3	1	2023	3539
9	22	4	1	2023	3605
10	22	5	1	2023	(null)
11	23	1	1	2023	7772
12	23	2	1	2023	1872
13	23	3	1	2023	3339
14	23	4	1	2023	3672
15	23	5	1	2023	(null)
16	24	1	1	2023	15439
17	24	2	1	2023	2539
18	24	3	1	2023	4405
19	24	4	1	2023	4015
20	24	5	1	2023	(null)

# 1 – данные по одному столбцу

--lg должен быть максимум на 10% меньше продаж того же товара самого лучшего продавца за январь 2022

```
with max_sales_in_1_2022
```

```
as
(
    select prd_type_id, max(amount) max_1
    from all_sales
    where year = 2022 and month = 1
    group by prd_type_id
    order by prd_type_id
)
```

```
select a.emp_id,
       a.prd_type_id,
       1,
       2023,
       m.max_1 plan_sales
from max_sales_in_1_2022 m join all_sales a
on m.prd_type_id = a.prd_type_id
where a.month = 1 and a.year = 2022;
```

	EMP_ID	PRD_TYPE_ID	1	2023	PLAN_SALES
1	21	1	1	2023	11034,84
2	22	1	1	2023	11034,84
3	23	1	1	2023	11034,84
4	24	1	1	2023	11034,84
5	25	1	1	2023	11034,84
6	26	1	1	2023	11034,84
7	21	2	1	2023	5434,84
8	22	2	1	2023	5434,84
9	23	2	1	2023	5434,84
10	24	2	1	2023	5434,84
11	25	2	1	2023	5434,84
12	26	2	1	2023	5434,84
13	21	3	1	2023	6334,84
14	22	3	1	2023	6334,84
15	23	3	1	2023	6334,84
16	24	3	1	2023	6334,84
17	25	3	1	2023	6334,84
18	26	3	1	2023	6334,84
19	21	4	1	2023	3364,84
20	22	4	1	2023	3364,84

## 2 – данные по набору столбцов

- При переходе от фиксированного значения (января 2023) к набору значений (январь – март 2023) запросы усложняются

```
-- 2 - формируем три столбца по правилам
-- 2а такие же, как на январь - март 2023
select prd_type_id,
       emp_id,
       1 month,
       2023 year,
       amount plan_sales
from all_sales
where year = 2022 and month = 1
union
select prd_type_id,
       emp_id,
       2 month,
       2023 year,
       amount plan_sales
from all_sales
where year = 2022 and month = 2
union
select prd_type_id,
       emp_id,
       3 month,
       2023 year,
       amount plan_sales
from all_sales
where year = 2022 and month = 3;
```

	PRD_TYPE_ID	EMP_ID	MONTH	YEAR	PLAN_SALES
1	1	21	1	2023	10034,84
2	1	21	2	2023	15144,65
3	1	21	3	2023	20137,83
4	1	22	1	2023	11034,84
5	1	22	2	2023	16144,65
6	1	22	3	2023	24137,83
7	1	23	1	2023	4034,84
8	1	23	2	2023	7144,65
9	1	23	3	2023	12137,83
10	1	24	1	2023	7034,84
11	1	24	2	2023	17144,65
12	1	24	3	2023	22137,83
13	1	25	1	2023	1234,84
14	1	25	2	2023	6144,65
15	1	25	3	2023	8137,83
16	1	26	1	2023	5534,84
17	1	26	2	2023	8844,65
18	1	26	3	2023	5137,83
19	2	21	1	2023	1034,84
20	2	21	2	2023	1544,65

## 2 – данные по набору столбцов

```
-- 26 формируем на 10% выше, чем январе-марте 2022 на товары 1-3, а на остальные такие же
select prd_type_id, emp_id, 1 year, 2023 year, amount *
  (CASE
    WHEN prd_type_id between 1 and 3 THEN 1.10
    ELSE 1
  END) plan_sales
from all_sales
where year = 2022 and month = 1
union
select prd_type_id, emp_id, 2 month, 2023 year, amount *
  (CASE
    WHEN prd_type_id between 1 and 3 THEN 1.10
    ELSE 1
  END) plan_sales
from all_sales
where year = 2022 and month = 2
union
select prd_type_id, emp_id, 3 month, 2023 year, amount *
  (CASE
    WHEN prd_type_id between 1 and 3 THEN 1.10
    ELSE 1
  END) plan_sales
from all_sales
where year = 2022 and month = 3;
```

	PRD_TYPE_ID	EMP_ID	YEAR	YEAR_1	PLAN_SALES
1	1	21	1	2023	11038,324
2	1	21	2	2023	16659,115
3	1	21	3	2023	22151,613
4	1	22	1	2023	12138,324
5	1	22	2	2023	17759,115
6	1	22	3	2023	26551,613
7	1	23	1	2023	4438,324
8	1	23	2	2023	7859,115
9	1	23	3	2023	13351,613
10	1	24	1	2023	7738,324
11	1	24	2	2023	18859,115
12	1	24	3	2023	24351,613
13	1	25	1	2023	1358,324
14	1	25	2	2023	6759,115
15	1	25	3	2023	8951,613
16	1	26	1	2023	6088,324
17	1	26	2	2023	9729,115
18	1	26	3	2023	5651,613
19	2	21	1	2023	1138,324
20	2	21	2	2023	1699,115
21	2	21	3	2023	2241,613
22	2	22	1	2023	1250,324

# 2 – данные по набору столбцов

```
--2в план на январь 2023 должен быть равен среднему за январь-март 2022,  
-- план на февраль 2023 должен быть равен среднему за февраль-апрель 2022,  
-- план на март 2023 должен быть равен среднему за март-май 2022
```

```
with  
avg_in_1_3_2022  
as  
(  
    select emp_id, prd_type_id, trunc(avg(amount)) avg_1_3  
    from all_sales  
    where year = 2022 and month between 1 and 3  
    group by emp_id, prd_type_id  
    order by emp_id, prd_type_id  
)  
avg_in_2_4_2022  
as  
(  
    select emp_id, prd_type_id, trunc(avg(amount)) avg_2_4  
    from all_sales  
    where year = 2022 and month between 2 and 4  
    group by emp_id, prd_type_id  
    order by emp_id, prd_type_id  
)  
,
```

	EMP_ID	PRD_TYPE_ID	MONTH	YEAR	PLAN_SALES
1	21	1	1	2023	15105
2	21	1	2	2023	20113
3	21	1	3	2023	20803
4	21	2	1	2023	1539
5	21	2	2	2023	2046
6	21	2	3	2023	2103
7	21	3	1	2023	3505
8	21	3	2	2023	3013
9	21	3	3	2023	3603
10	21	4	1	2023	3839
11	21	4	2	2023	4013
12	21	4	3	2023	3936
13	21	5	1	2023	(null)

```
avg_in_3_5_2022  
as  
(  
    select emp_id, prd_type_id, trunc(avg(amount)) avg_3_5  
    from all_sales  
    where year = 2022 and month between 3 and 5  
    group by emp_id, prd_type_id  
    order by emp_id, prd_type_id  
)  
select a1.emp_id,  
       a1.prd_type_id,  
       1 month,  
       2023 year,  
       a1.avg_1_3 plan_sales  
from avg_in_1_3_2022 a1  
union  
select a2.emp_id,  
       a2.prd_type_id,  
       2 month,  
       2023 year,  
       a2.avg_2_4 plan_sales  
from avg_in_2_4_2022 a2  
union  
select a3.emp_id,  
       a3.prd_type_id,  
       3 month,  
       2023 year,  
       a3.avg_3_5 plan_sales  
from avg_in_3_5_2022 a3;
```



# 2 – данные по набору столбцов

--2г должен быть максимум на 10% меньше продаж того же товара самого лучшего продавца  
-- за январь-март 2022 соответственно

```
with max_sales_in_1_2022
as
(
    select prd_type_id, max(amount) max_1
    from all_sales
    where year = 2022 and month = 1
    group by prd_type_id
    order by prd_type_id
),
max_sales_in_2_2022
as
(
    select prd_type_id, max(amount) max_2
    from all_sales
    where year = 2022 and month = 2
    group by prd_type_id
    order by prd_type_id
),
max_sales_in_3_2022
as
(
    select prd_type_id, max(amount) max_3
    from all_sales
    where year = 2022 and month = 3
    group by prd_type_id
    order by prd_type_id
)
select a.emp_id,
       a.prд_type_id,
       1 month,
       2023 year,
       m.max_1 plan_sales
from all_sales a join max_sales_in_1_2022 m
on m.prд_type_id = a.prд_type_id
where a.month = 1 and a.year = 2022
union
```

```
select a.emp_id,
       a.prд_type_id,
       2 month,
       2023 year,
       m.max_2 plan_sales
from all_sales a join max_sales_in_2_2022 m
on m.prд_type_id = a.prд_type_id
where a.month = 2 and a.year = 2022
union
select a.emp_id,
       a.prд_type_id,
       3 month,
       2023 year,
       m.max_3 plan_sales
from all_sales a join max_sales_in_3_2022 m
on m.prд_type_id = a.prд_type_id
where a.month = 3 and a.year = 2022
;
```

	EMP_ID	PRD_TYPE_ID	MONTH	YEAR	PLAN_SALES
1	21	1	1	2023	11034,84
2	21	1	2	2023	17144,65
3	21	1	3	2023	24137,83
4	21	2	1	2023	5434,84
5	21	2	2	2023	3844,65
6	21	2	3	2023	5137,83
7	21	3	1	2023	6334,84
8	21	3	2	2023	4544,65
9	21	3	3	2023	6337,83
10	21	4	1	2023	3364,84
11	21	4	2	2023	4344,65
12	21	4	3	2023	6337,83
13	21	5	1	2023	(null)
14	21	5	2	2023	(null)
15	21	5	3	2023	(null)
16	22	1	1	2023	11034,84

# 3 – данные по набору столбцов

- При увеличении количества значений запросы усложняются до полной нечитаемости

# MODEL

- Решим те же задачи с использованием фразы MODEL:
- Измерения – месяц и год
- Значения не зависят от типа товара или номера сотрудника – секции
- Мерой является значение **amount**
- Правила задаются для ячеек

```
-- 1 - формируем один столбец по правилам
-- 1a такой же, как и в январе 2022
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id, emp_id)
  DIMENSION BY (month, year)
  MEASURES (amount sales_amount)
  RULES (sales_amount[1, 2023] = sales_amount[1, 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21		1 2023	1	10034,84
2	21		2 2023	1	1034,84
3	21		3 2023	1	6034,84
4	21		4 2023	1	3034,84
5	21		5 2023	1	(null)
6	22		1 2023	1	11034,84
7	22		2 2023	1	1234,84
8	22		3 2023	1	6334,84
9	22		4 2023	1	3334,84
10	22		5 2023	1	(null)
11	23		1 2023	1	4034,84
12	23		2 2023	1	1234,84
13	23		3 2023	1	6334,84

# 2а – данные по набору столбцов

- Для ссылки на ячейку можно использовать функцию **currentv()**

```
-- 2а такие же, как и в январе-марте 2022
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id, emp_id)
  DIMENSION BY (month, year)
  MEASURES (amount sales_amount)
  RULES (
    sales_amount[1, 2023] = sales_amount[1, 2022],
    sales_amount[2, 2023] = sales_amount[2, 2022],
    sales_amount[3, 2023] = sales_amount[3, 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21	1 2023	1	10034,84
2	21	2 2023	1	1034,84
3	21	3 2023	1	6034,84
4	21	4 2023	1	3034,84
5	21	5 2023	1	(null)
6	22	1 2023	1	11034,84
7	22	2 2023	1	1234,84
8	22	3 2023	1	6334,84
9	22	4 2023	1	3334,84
10	22	5 2023	1	(null)
11	23	1 2023	1	4034,84
12	23	2 2023	1	1234,84
13	23	3 2023	1	6334,84

```
-- или с помощью функции currentv()
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id, emp_id)
  DIMENSION BY (month, year)
  MEASURES (amount sales_amount) (
    sales_amount[1, 2023] = sales_amount[currentv(), 2022],
    sales_amount[2, 2023] = sales_amount[currentv(), 2022],
    sales_amount[3, 2023] = sales_amount[currentv(), 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
28	26	3 2023	1	2434,84
29	26	4 2023	1	3164,23
30	26	5 2023	1	(null)
31	21	1 2023	2	15144,65
32	21	2 2023	2	1544,65
33	21	3 2023	2	1944,65
34	21	4 2023	2	2944,65
35	21	5 2023	2	(null)
36	22	1 2023	2	16144,65
37	22	2 2023	2	1044,65
38	22	3 2023	2	1544,65
39	22	4 2023	2	2344,65
40	22	5 2023	2	(null)
41	22	1 2023	2	2144,65

# За – данные по набору столбцов

- Если необходимо пройти по диапазону используется конструкция FOR ... FROM ... TO ... INCREMENT

```
-- За такие же, как и в 2022 соответственно
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id, emp_id)
  DIMENSION BY (month, year)
  MEASURES (amount sales_amount) (
    sales_amount[FOR month FROM 1 TO 12 INCREMENT 1, 2023] = sales_amount[currentv(), 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21	1	2023	1	10034,84
2	21	2	2023	1	1034,84
3	21	3	2023	1	6034,84
4	21	4	2023	1	3034,84
5	21	5	2023	1	(null)

181	21	1	2023	7	12654,84
182	21	2	2023	7	1264,84
183	21	3	2023	7	21264,84
184	21	4	2023	7	1264,84
185	21	5	2023	7	(null)
186	22	1	2023	7	13654,84
187	22	2	2023	7	1964,84

# 16 – данные по столбцу

- Есть зависимость изменения значения меры от вида товара
- Вид товара – измерение

-- 16 на 10% выше, чем январе 2022 на товары 1-3, а на остальные такой же

```
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
PARTITION BY (emp_id)
DIMENSION BY (prd_type_id, month, year)
MEASURES (amount sales_amount)
RULES (sales_amount[FOR prd_type_id FROM 1 TO 3 INCREMENT 1, 1, 2023] =
        sales_amount[currentv(), 1, 2022] * 1.1,
        sales_amount[FOR prd_type_id FROM 4 TO 5 INCREMENT 1, 1, 2023] = sales_amount[currentv(), 1, 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21	1	2023	1	11038,324
2	21	2	2023	1	1138,324
3	21	3	2023	1	6638,324
4	21	4	2023	1	3034,84
5	21	5	2023	1	(null)
6	22	1	2023	1	12138,324
7	22	2	2023	1	1358,324
8	22	3	2023	1	6968,324
9	22	4	2023	1	3334,84
10	22	5	2023	1	(null)
11	23	1	2023	1	4438,324
12	23	2	2023	1	1358,324
13	23	3	2023	1	6968,324

# 26 – данные по набору столбцов

```
-- 26 формируем на 10% выше, чем январе-марте 2022 на товары 1-3, а на остальные такие же
-- циклы по двум измерениям, замена функции currentv() на такую же cv()
-- 36 год аналогично
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (emp_id)
  DIMENSION BY (prd_type_id, month, year)
  MEASURES (amount sales_amount)
  RULES (sales_amount[FOR prd_type_id FROM 1 TO 3 INCREMENT 1,
    FOR month FROM 1 TO 3 INCREMENT 1, 2023] =
    sales_amount[cv(), cv(), 2022] * 1.1,
    sales_amount[FOR prd_type_id FROM 4 TO 5 INCREMENT 1,
    FOR month FROM 1 TO 3 INCREMENT 1, 2023] =
    sales_amount[cv(), cv(), 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21	1	2023	1	11038,324
2	21	2	2023	1	1138,324
3	21	3	2023	1	6638,324
4	21	4	2023	1	3034,84
5	21	5	2023	1	(null)
6	22	1	2023	1	12138,324
7	22	2	2023	1	1358,324
8	22	3	2023	1	6968,324
9	22	4	2023	1	3334,84
10	22	5	2023	1	(null)
11	23	1	2023	1	4438,324
12	23	2	2023	1	1358,324

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
28	26	3	2023	1	2678,324
29	26	4	2023	1	3164,23
30	26	5	2023	1	(null)
31	21	1	2023	2	16659,115
32	21	2	2023	2	1699,115
33	21	3	2023	2	2139,115
34	21	4	2023	2	2944,65
35	21	5	2023	2	(null)
36	22	1	2023	2	17759,115
37	22	2	2023	2	1149,115
38	22	3	2023	2	1699,115
39	22	4	2023	2	2344,65
40	22	5	2023	2	(null)

# 1в – данные по столбцу

- Есть зависимость изменения значения меры от диапазона значений
- От сотрудника или товара значение меры не зависит

--1в должен быть равен среднему за январь-март 2022

```
SELECT emp_id, prd_type_id, year, month, sales_amount
```

```
FROM all_sales
```

## MODEL

PARTITION BY (prd type id, emp id)

DIMENSION BY (month, year)

**MEASURES** (amount sales amount)

**RULES** (sales amount[1, 2023] = AVG(sales amount) [month BETWEEN 1 AND 3, 2022])

```
ORDER BY year DESC, month, emp id, prd type id;
```

[illegible]



## 2в – данные по набору столбцов

- 2в план на январь 2023 должен быть равен среднему за январь-март 2022,
- план на февраль 2023 должен быть равен среднему за февраль-апрель 2022,
- план на март 2023 должен быть равен среднему за март-май 2022
- 3в год аналогично

```
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id, emp_id)
  DIMENSION BY (month, year)
  MEASURES (amount sales_amount)
  RULES (sales_amount[FOR month FROM 1 TO 3 INCREMENT 1, 2023] =
    AVG(sales_amount)[month BETWEEN cv() AND (cv() + 2), 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

[illegible]

# 1г – данные по столбцу

- Есть зависимость изменения значения меры от сотрудника

```
--1г должен быть максимум на 10% меньше продаж того же товара
--самого лучшего продавца за январь 2022
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id)
  DIMENSION BY (emp_id, month, year)
  MEASURES (amount sales_amount)
  RULES (sales_amount[FOR emp_id FROM 21 TO 26 INCREMENT 1, 1, 2023] =
    MAX(sales_amount)[emp_id BETWEEN 21 AND 26, 1, 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

	EMP_ID	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21		1 2023	1	11034,84
2	21		2 2023	1	5434,84
3	21		3 2023	1	6334,84
4	21		4 2023	1	3364,84
5	21		5 2023	1	(null)
6	22		1 2023	1	11034,84
7	22		2 2023	1	5434,84
8	22		3 2023	1	6334,84
9	22		4 2023	1	3364,84

# 2г – данные по набору столбцов

```
--2г должен быть максимум на 10% меньше продаж того же товара самого лучшего продавца
-- за январь-март 2022 соответственно
-- 3г за год аналогично
SELECT emp_id, prd_type_id, year, month, sales_amount
FROM all_sales
MODEL
  PARTITION BY (prd_type_id)
  DIMENSION BY (emp_id, month, year)
  MEASURES (amount sales_amount)
  RULES (sales_amount[FOR emp_id FROM 21 TO 26 INCREMENT 1,
                      FOR month FROM 1 TO 3 INCREMENT 1, 2023] =
          MAX(sales_amount)[emp_id BETWEEN 21 AND 26,
                           cv(), 2022])
ORDER BY year DESC, month, emp_id, prd_type_id;
```

E...	PRD_TYPE_ID	YEAR	MONTH	SALES_AMOUNT
1	21	1 2023	1	11034,84
2	21	2 2023	1	5434,84
3	21	3 2023	1	6334,84
4	21	4 2023	1	3364,84
5	21	5 2023	1	(null)

31	21	1 2023	2	17144,65
32	21	2 2023	2	3844,65
33	21	3 2023	2	4544,65
34	21	4 2023	2	4344,65
35	21	5 2023	2	(null)

# MODEL – анализ планов запросов

- Результаты по стандартным SELECT

	а	б	в	г
1	3	3	3	5
2	9	9	9	15
3	?	?	?	?

- Результаты по SELECT с использованием MODEL

	а	б	в	г
1	3	3	3	3
2	3	3	3	3
3	3	3	3	3

# MODEL – обзор возможностей

- **Partitions** – секции куба
- **Dimensions** – измерения куба
- **Measures** – меры куба
- **Rules** – правила вычисления ячеек
- Символьная, позиционная и смешанная нотации
- **Nested references** – существует возможность вложенных ссылок
- **Upsert (all), update** – выдача измененных/всех строк
- **Order by** – сортировка при вычислении значений
- **Sequential / automatic order** – вычисления производятся по столбцам
- **Iterate [until]** – задается количество итераций
- **Previous** – получение предыдущего значения ячейки
- **Reference model** – модель, которая может быть использована как вспомогательная
- **Unique single reference** – возможность использовать неуникальную адресацию ячеек

# MODEL nested references

```
-- dimensions
-- measures
-- rules
-- return
-- nested cell references

with t (id, value) as
(
  select 1, 3 from dual
    union all
  select 3, 8 from dual
    union all
  select 5, 5 from dual
    union all
  select 10, 4 from dual
)
select * from t
  model
    -- return updated rows
    dimension by (id)
    measures (value, 0 result)
    rules
    (
      result[0] = value[10] + value[value[1]],
      result [id >= 5] = sum(value) [id <= cv(id)]
    );
```

	ID	VALUE	RESULT
1	1	3	0
2	3	8	0
3	5	5	16
4	10	4	20
5	0 (null)		12

# MODEL – upsert (all), update

- Update только обновляет существующие строки
- Upsert (используется по умолчанию) обновляет существующие и добавляет пропущенные, если использована позиционная нотация
- Upsert all возвращает также строки, если использована комбинированная нотация и ячейки для измерений с символьной нотацией существуют

# MODEL – upsert (all), update

```
-- update, upsert, upsert all - отображение строк
-- Символьная (с названием измерения) и позиционная нотация
-- смешанная нотация
```

```
with t (dim1, dim2, value) as
```

```
(
  select 0, 0, 1 from dual
    union all
  select 0, 1, 2 from dual
    union all
  select 1, 0, 3 from dual
)
```

```
select * from t
```

```
  model
```

```
    dimension by (dim1, dim2)
```

```
    measures (value, cast(NULL as number) result)
```

```
    rules
```

```
    upsert all -- upsert all -- update -- upsert
```

```
  (
```

```
    result[0, 0] = -1,
```

```
    result[dim1 = 1, dim2 = 0] = -3,
```

```
    result[-1, for dim2 in (select count(*) from dual)] = -4,
```

```
    result[-2, dim2 = 1] = -10, -- dim2 = 1 exists
```

```
    result[-3, dim2 = -1] = -20, -- dim2 = -1 not exists
```

```
    result[-4, -1] = -30
```

```
  )
```

```
order by dim1, dim2;
```

	DIM1	DIM2	VALUE	RESULT
1	-4	-1	(null)	-30
2	-2	1	(null)	-10
3	-1	1	(null)	-4
4	0	0	1	-1
5	0	1	2	(null)
6	1	0	3	-3



# MODEL – upsert (all), update

```
-- update, upsert, upsert all - отображение строк
-- символьная (с названием измерения) и позиционная нотация
-- смешанная нотация
with t (dim1, dim2, value) as
(
  select 0, 0, 1 from dual
    union all
  select 0, 1, 2 from dual
    union all
  select 1, 0, 3 from dual
)
select * from t
  model
    dimension by (dim1, dim2)
    measures (value, cast(NULL as number) result)
    rules
      upsert -- upsert all -- update -- upsert
      (
        result[0, 0] = -1,
        result[dim1 = 1, dim2 = 0] = -3,
        result[-1, for dim2 in (select count(*) from dual)] = -4,
        result[-2, dim2 = 1] = -10, -- dim2 = 1 exists
        result[-3, dim2 = -1] = -20, -- dim2 = -1 not exists
        result[-4, -1] = -30
      )
    order by dim1, dim2;
```

	DIM1	DIM2	VALUE	RESULT
1	-4	-1	(null)	-30
2	-1	1	(null)	-4
3	0	0	1	-1
4	0	1	2	(null)
5	1	0	3	-3

# MODEL – upsert (all), update

```
-- update, upsert, upsert all - отображение строк
-- символьная (с названием измерения) и позиционная нотация
-- смешанная нотация
with t (dim1, dim2, value) as
(
  select 0, 0, 1 from dual
    union all
  select 0, 1, 2 from dual
    union all
  select 1, 0, 3 from dual
)
select * from t
  model
    dimension by (dim1, dim2)
    measures (value, cast(NULL as number) result)
    rules
    update -- upsert all -- update -- upsert
    (
      result[0, 0] = -1,
      result[dim1 = 1, dim2 = 0] = -3,
      result[-1, for dim2 in (select count(*) from dual)] = -4,
      result[-2, dim2 = 1] = -10, -- dim2 = 1 exists
      result[-3, dim2 = -1] = -20, -- dim2 = -1 not exists
      result[-4, -1] = -30
    )
order by dim1, dim2;
```

	DIM1	...	VALUE	RESULT
1	0	0	1	-1
2	0	1	2	(null)
3	1	0	3	-3

# MODEL – order by

- При вычислении всех значений ячеек можно использовать адресацию [any] или [... is any]
- При такой адресации можно установить порядок вычисления ячеек – сортировку
- Используется, когда следующее значение зависит от предыдущего

# MODEL – order by

```
-- any позиционная
-- is any символьная
-- сортировка определяет порядок вычисления номеров ячеек
with t (id, value) as
  (select rownum, rownum*2 from dual connect by level <=3)
select * from t
  model
    dimension by (id)
    measures (value, 10 planned_2023, 20 planned_2024)
    rules
      (
        planned_2023[any] order by id asc = nvl(planned_2023[cv(id) - 1], 0) + value[cv(id)],
        planned_2024[id is any] order by id desc = nvl(planned_2024[cv(id) - 1], 0) + value[cv(id)]
      )
order by id;
```

	ID	VALUE	PLANNED_2023	PLANNED_2024
1	1	2	2	2
2	2	4	6	24
3	3	6	12	26

# MODEL – order by

```
-- если сортировка не указана
-- возможны ошибки заикливания
with t (id, value) as
    (select rownum, rownum from dual connect by level <=3)
select * from t
    model
        dimension by (id)
        measures (value, 10 result)
        rules
        (
            result[any] -- order by id asc
                        = sum(result)[any]
        )
order by id;
```

ORA-32637: Правило заикливания в MODEL с последовательным порядком  
32637. 00000 - "Self cyclic rule in sequential order MODEL"

\*Cause: A self-cyclic rule was detected in the sequential order MODEL.  
Sequential order MODELS cannot have self cyclic rules  
to guarantee that the results do not depend on the order  
of evaluation of the cells that are updated or upserted.

\*Action: Use ordered rule evaluation for this rule.

# MODEL – order by

```
-- если сортировка не указана
-- возможны ошибки зацикливания
with t (id, value) as
  (select rownum, rownum from dual connect by level <=3)
select * from t
  model
    dimension by (id)
    measures (value, 10 result)
    rules
      (
        result[any] order by id asc
          = sum(result)[any]
      )
order by id;
```

	ID	VALUE	RESULT
1	1	1	30
2	2	2	50
3	3	3	90

# MODEL – sequential order

- При sequential order вычисляется вначале значения полностью по первому правилу, потом по второму и т.д.
- При automatic order учитываются связи между правилами
- Однако вычисление значений происходит по правилам (столбцу)

# MODEL – sequential order

```
-- sequential order (default)
-- automatic order - учитываются зависимости между правилами

with t as
  (select rownum id from dual connect by level <= 5)
select * from t
  model
    dimension by (id)
    measures (0 t1, 0 x, 0 t2)
    rules
      -- automatic order -- sequential order
      (
        t1[id] = x[cv(id)-1],
        x[id] = cv(id),
        t2[id] = x[cv(id)-1]
      )
  end select;
```

	ID	T1	X	T2
1	1	(null)	1	(null)
2	2	0	2	1
3	3	0	3	2
4	4	0	4	3
5	5	0	5	4



# MODEL – automatic order

```
-- sequential order (default)
-- automatic order - учитываются зависимости между правилами

with t as
  (select rownum id from dual connect by level <= 5)
select * from t
model
  dimension by (id)
  measures (0 t1, 0 x, 0 t2)
  rules
    automatic order -- sequential order
  (
    t1[id] = x[cv(id)-1],
    x[id] = cv(id),
    t2[id] = x[cv(id)-1]
  )
order by id;
```

	ID	T1	X	T2
1	1	(null)	1	(null)
2	2		1	2
3	3		2	3
4	4		3	4
5	5		4	5

# MODEL – columns

```
-- вычисления по столбцам!  
with t as  
  (select rownum id from dual connect by level <= 5)  
select * from t  
  model  
    dimension by (id)  
    measures (0 t1, 1 x)  
    rules  
    automatic order  
    --sequential order  
  (  
    t1[any] = sum(x)[any],  
    x[any] order by id asc = sum(x)[id <= cv(id)]  
  )  
order by id;
```

	ID	T1	X
1	1	31	1
2	2	31	2
3	3	31	4
4	4	31	8
5	5	31	16

# MODEL – columns

```
-- вычисления по столбцам!
with t as
  (select rownum id from dual connect by level <= 5)
select * from t
  model
    dimension by (id)
    measures (0 t1, 1 x)
    rules
      --automatic order
      sequential order
    (
      t1[any] = sum(x)[any],
      x[any] order by id asc = sum(x)[id <= cv(id)]
    )
order by id;
```

	ID	T1	X
1	1	5	1
2	2	5	2
3	3	5	4
4	4	5	8
5	5	5	16

# MODEL – ITERATE

- **Iterate** – задает количество итераций
- Номера итераций от 0

```
-- iterate from 0
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x1, 1 x2)
    rules iterate (10)
      (
        x2[iteration_number] = (nvl(x2[iteration_number-1], 0) + 1) * 2,
        x1[iteration_number] = x2[iteration_number-1] + 1
      )
order by id;
```

	ID	X1	X2
1	0	(null)	2
2	1	3	6
3	2	7	14
4	3	15	30
5	4	31	62
6	5	63	126
7	6	127	254
8	7	255	510
9	8	511	1022
10	9	1023	2046

# MODEL – ITERATE UNTIL

```
-- iterate until
-- checked at the end of each iteration
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x1, 1 x2)
    rules iterate (10) until (abs(x1[iteration_number] - x2[iteration_number]) > 40)
    (
      x2[iteration_number] = (nvl(x2[iteration_number-1], 0) + 1) * 2,
      x1[iteration_number] = x2[iteration_number-1] + 1
    )
order by id;
```

	ID	X1	X2
1	0	(null)	2
2	1	3	6
3	2	7	14
4	3	15	30
5	4	31	62
6	5	63	126

# MODEL – ITERATE UNTIL

```
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x1, 1 x2)
    rules iterate (10) until (iteration_number = 5)
      (
        x2[iteration_number] = iteration_number,
        x1[iteration_number] = x2[iteration_number-1] + 1
      )
order by id;
```

	ID	X1	X2
1	0	(null)	0
2	1	1	1
3	2	2	2
4	3	3	3
5	4	4	4
6	5	5	5

# MODEL – ITERATE UNTIL

```
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x1, 1 x2)
    rules iterate (10) until (x2 [iteration_number] > 5)
    (
      x2[iteration_number] = iteration_number,
      x1[iteration_number] = x2[iteration_number-1] + 1
    )
order by id;
```

	ID	X1	X2
1	0	(null)	0
2	1	1	1
3	2	2	2
4	3	3	3
5	4	4	4
6	5	5	5
7	6	6	6

# MODEL – PREVIOUS()

- **Previous** – получение предыдущего значения ячейки

```
-- previous() - previous value of a cell
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x)
    rules iterate (10) until (abs(x[0] - previous(x[0])) > 6)
    (
      x[0] = x[0] + iteration_number * 2 -- 0, 2, 6, 12, 20, 30, 42, 56, 72, 90
    )
order by id;
```

ID	X
1	0 20



# MODEL – PREVIOUS()

```
-- previous() - only in until section
with t as
  (select 0 id from dual)
select * from t
  model
    dimension by (id)
    measures (0 x)
    rules iterate (10) until (abs(x[0] - previous(x[0])) > 6)
    (
      x[0] = previous(x[0]) + iteration_number * 2 -- err
    )
order by id;
```

ORA-32618: некорректное использование функции MODEL PREVIOUS  
32618. 00000 - "incorrect use of MODEL PREVIOUS function"  
\*Cause: The MODEL PREVIOUS function was used outside of MODEL  
"ITERATE UNTIL" clause, or was nested.  
\*Action: Check the SQL statement and rewrite if necessary.  
Error at Line: 207 Column: 20

# MODEL – REFERENCE MODELS

```
with
sales(year, currency, value)
as
    (select '2015', 'GBP', 100 from dual
     union all
     select '2015', 'USD', 200 from dual
     union all
     select '2015', 'EUR', 300 from dual
     union all
     select '2016', 'GBP', 400 from dual
     union all
     select '2016', 'EUR', 500 from dual) ,
usd_rates(currency, rate)
as
    (select 'GBP', 1.45 from dual
     union all
     select 'USD', 1 from dual
     union all
     select 'EUR', 1.12 from dual)
select * from sales
model
reference usd_rates_model on (select * from usd_rates)
    dimension by (currency)
    measures (rate)
main sales_model
    dimension by (year, currency)
    measures (value, 0 usd_value)
(
    usd_value[any, any] order by year, currency =
    value[cv(year), cv(currency)] * usd_rates_model.rate[cv(currency)]
)
order by 1, 2;
```

**Reference model –**  
модель, которая  
может быть  
использована как  
вспомогательная

	YEAR	CURRENCY	VALUE	USD_VALUE
1	2015	EUR	300	336
2	2015	GBP	100	145
3	2015	USD	200	200
4	2016	EUR	500	560
5	2016	GBP	400	580

# MODEL – UNIQUE SINGLE REFERENCE

- **Unique single reference** – возможность использовать неуникальную адресацию ячеек

```
-- unique single reference
-- by default all values in dimensions must be unique
with t(id, value)
as
(
    select trunc(rownum/2), rownum
    from dual
    connect by level <= 3
)
select * from t
model unique single reference
    dimension by (id)
    measures (value, 0 result)
    ( result[0] = 111,
      result[1] = 222 )
order by id;
```

	ID	VALUE	RESULT
1	0	1	111
2	1	3	222
3	1	2	222

# MODEL – UNIQUE SINGLE REFERENCE

```
-- unique single reference
-- by default all values in dimensions must be unique
with t(id, value)
as
(
    select trunc(rownum/2), rownum
    from dual
    connect by level <= 3
)
select * from t
model -- unique single reference
    dimension by (id)
    measures (value, 0 result)
    ( result[0] = 111,
      result[1] = 222 )
order by id;
```

ORA-32638: Неуникальная адресация в измерениях MODEL  
32638. 00000 - "Non unique addressing in MODEL dimensions"

\*Cause: The address space defined for the MODEL (partition by and dimension by expressions) do not uniquely identify each cell.

\*Action: Rewrite the MODEL clause. Using UNIQUE SINGLE REFERENCE option might help.

# MODEL – применяется для:

- Spreadsheet-like вычислений, т.е. получение значений ячеек с помощью выражений, использующих значения других ячеек
- Внешних отчетных систем: когда имеются только привилегии SELECT
- Для материализованных представлений

# MODEL – не применяется для:

- Генерации последовательностей независимых значений – connect by
- Генерации последовательностей зависимых значений – with (recursive)
- Обработки строковых значений
- Определения последовательностей в наборе данных – аналитические функции
- Подсчета итогов – group by rollup / grouping sets / cube
- Транспонирования – pivot / unpivot

# MODEL – проверочная работа:

- Используется таблица all\_sales
- Построить план продаж на каждый месяц 2023 года, причем:
- а) для сотрудников 21-22 должен быть на 10% больше, чем за аналогичный месяц 2022 года, для остальных - на 5% больше, чем за аналогичный месяц 2022 года.
- б) для всех сотрудников должен быть равен среднему значению продаж за предыдущие 3 месяца;
- в) для каждого сотрудника должен быть вычислен как половина разницы между продажами этого же товара для аналогичного периода этого сотрудника и сотрудника, который продал тот же товар в аналогичном периоде на наибольшую сумму.
- т.е. сотрудник 21 продал в 1 месяце 2022 года 1 товар на сумму 10034,84, а максимальную продажу в этом периоде по этому товару сделал сотрудник 22 на сумму 11034,84, разница 1000, поэтому 21 сотруднику на 1 месяц 2023 года установлен план:  
$$(11034,84 - 10034,84) / 2 + 10034,84 = 10534,84;$$
- а сотрудник 23 продал в 1 месяце 2022 года 1 товар на сумму 4034,84, а максимальную продажу в этом периоде по этому товару сделал сотрудник 22 на сумму 11034,84, разница 7000, поэтому 23 сотруднику на 1 месяц 2023 года установлен план:  
$$(11034,84 - 4034,84) / 2 + 4034,84 = 7534,84;$$

Вопросы?