Методы сбора, хранения, обработки и анализа данных

Лекция 2

Иерархические данные

Иерархии

- организационные графики
- структуры предприятий
- списки файлов и папок
- каталоги продуктов
- ветки форумов

Иерархии

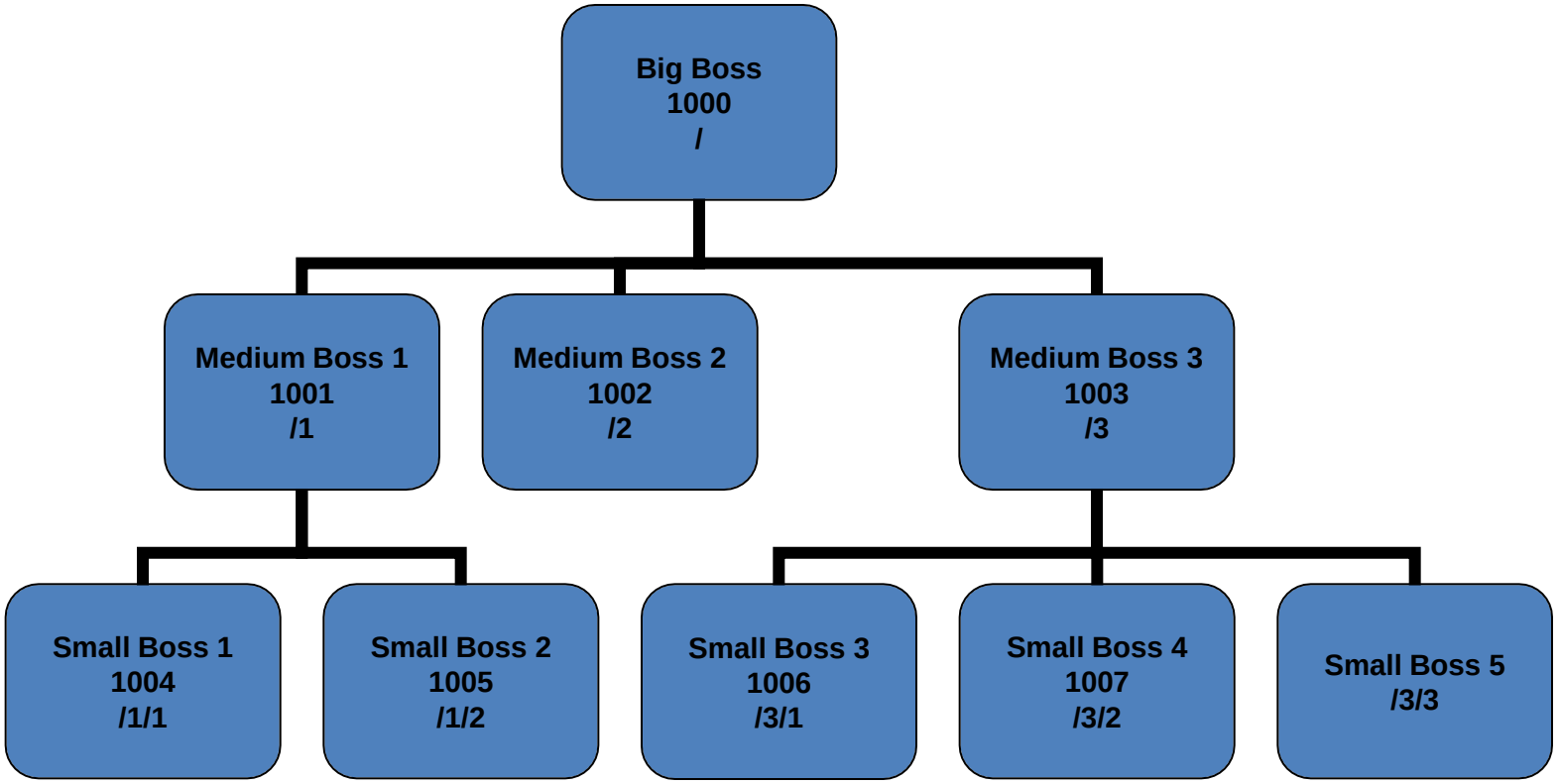
- Adjacency List – список смежных вершин (parent_id)
- Nested Set – указание вложенности (left/right lower/upper)
- Materialized Path – полный путь (1/2/1)
- Добавление уровня – level

Иерархический тип данных

- hierarchyid
- системный тип данных переменной длины
- используется для представления положения в иерархии
- путь логически представлен в виде последовательности меток всех посещенных дочерних узлов, начиная с корня

Иерархический тип данных

- hierarchyid



Корневой элемент

```
CREATE TABLE Employee(
    node hierarchyid PRIMARY KEY CLUSTERED,
    LEVEL AS node.GetLevel () PERSISTED,
    EMPLOYEE_id INT UNIQUE,
    EMPLOYEE_NAME VARCHAR(30) NOT NULL);
```

```
INSERT INTO Employee VALUES
(hierarchyid::GetRoot(), 1000, 'Big Boss');
SELECT node.ToString () AS NodeAsString,
    node as NodeAsBinary,
    node.GetLevel() AS Level,
    employee_id,
    employee_name
FROM Employee;
go
```

%

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss

Добавление дочернего узла

```
DECLARE @ManagerNode hierarchyid;
DECLARE @Level hierarchyid;
SELECT @ManagerNode=node FROM Employee WHERE employee_id=1000;
INSERT INTO Employee VALUES (@ManagerNode.GetDescendant(NULL, NULL),1001, 'Medium Boss 1');
go
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1

```
DECLARE @ManagerNode hierarchyid;
DECLARE @Level hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1001;
SELECT @ManagerNode=node FROM Employee WHERE employee_id=1000;
INSERT INTO Employee VALUES (@ManagerNode.GetDescendant(@Level, NULL),1002, 'Medium Boss 2');
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/2/	0x68	1	1002	Medium Boss 2

GetDescendant (child1, child2)

- Если родитель — NULL, возвращает NULL
- Если родитель — не NULL, а потомки child1 и child2 — NULL, возвращает одного потомка
- Если родитель и потомок child1 — не NULL, а child2 NULL, возвращает потомка, который больше чем child1, и наоборот
- Если родитель, child1 и child2 не NULL, возвращает дочерний узел, больше child1 и меньше child2
- Если child1 (child2) не NULL и не является дочерним узлом, то возникает исключение
- Если child1 >= child2, то возникает исключение

Добавление дочернего узла

```
DECLARE @ManagerNode hierarchyid;
DECLARE @Level hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1002;
SELECT @ManagerNode=node FROM Employee WHERE employee_id=1000;
INSERT INTO Employee VALUES (@ManagerNode.GetDescendant(@Level, NULL),1003, 'Medium Boss 3');
go
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3

```
DECLARE @Level hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1001;
INSERT INTO Employee VALUES (@Level.GetDescendant(NULL, NULL),1004, 'Small Boss 1');
go
```

Results | Messages

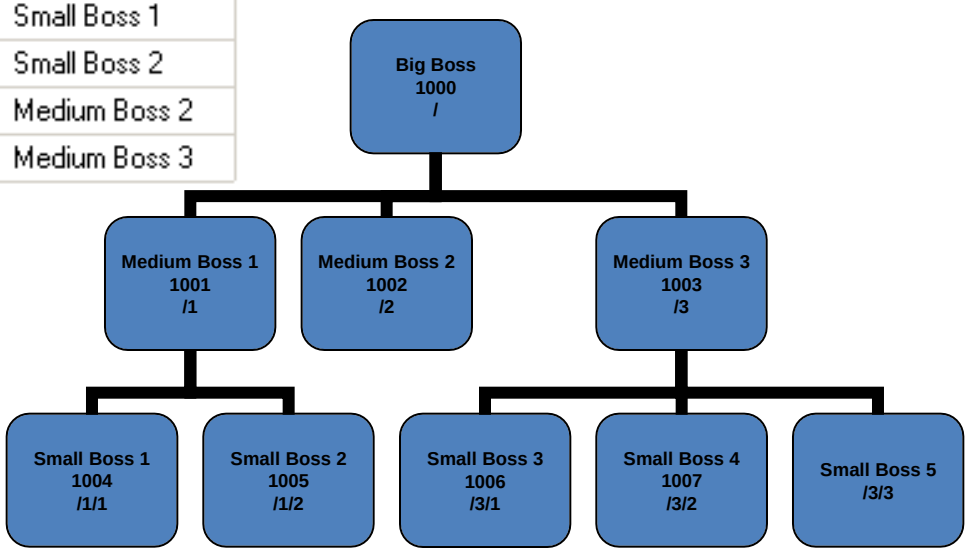
NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/1/1/	0x5AC0	2	1004	Small Boss 1
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3

Иерархический тип данных

```
DECLARE @Level hierarchyid;
DECLARE @child hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1001;
SELECT @child=node FROM Employee WHERE employee_id=1004;
INSERT INTO Employee VALUES (@Level.GetDescendant(@child, NULL),1005, 'Small Boss 2');
go
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/1/1/	0x5AC0	2	1004	Small Boss 1
/1/2/	0x5B40	2	1005	Small Boss 2
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3

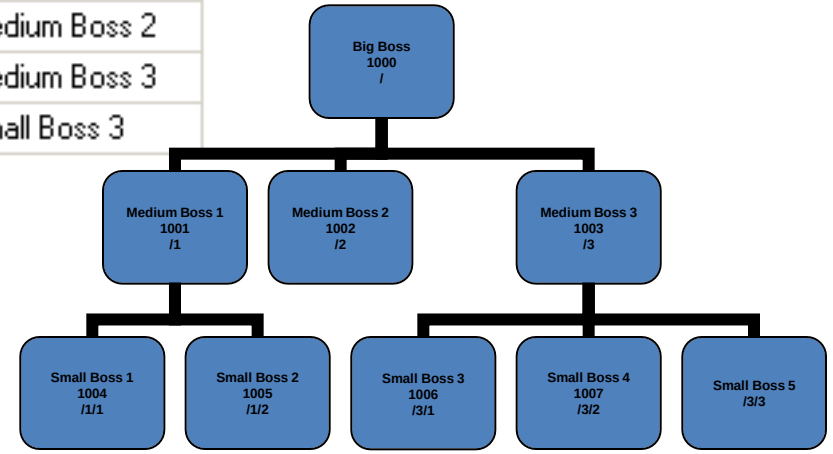


Иерархический тип данных

```
DECLARE @Level hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1003;
INSERT INTO Employee VALUES (@Level.GetDescendant(NULL, NULL),1006, 'Small Boss 3');
go
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/1/1/	0x5AC0	2	1004	Small Boss 1
/1/2/	0x5B40	2	1005	Small Boss 2
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3
/3/1/	0x7AC0	2	1006	Small Boss 3

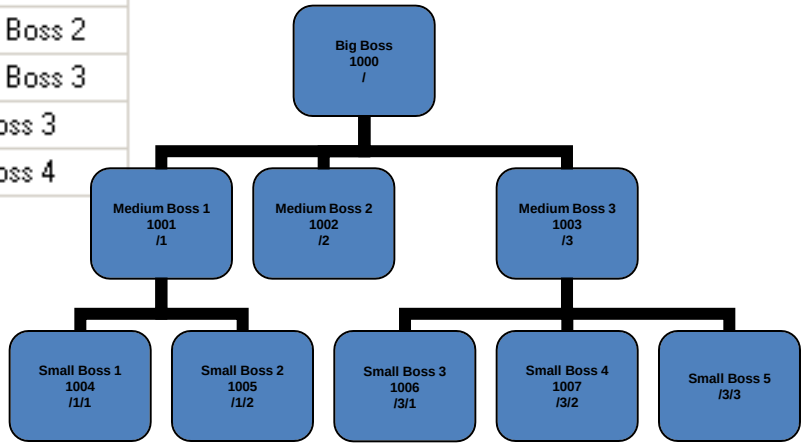


Иерархический тип данных

```
DECLARE @Level hierarchyid;
DECLARE @child hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1003;
SELECT @child=node FROM Employee WHERE employee_id=1006;
INSERT INTO Employee VALUES (@Level.GetDescendant(@child, NULL),1007, 'Small Boss 4');
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/1/1/	0x5AC0	2	1004	Small Boss 1
/1/2/	0x5B40	2	1005	Small Boss 2
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3
/3/1/	0x7AC0	2	1006	Small Boss 3
/3/2/	0x7B40	2	1007	Small Boss 4



Иерархический тип данных

```
DECLARE @Level hierarchyid;
DECLARE @child hierarchyid;
SELECT @Level=node FROM Employee WHERE employee_id=1003;
SELECT @child=node FROM Employee WHERE employee_id=1007;
INSERT INTO Employee VALUES (@Level.GetDescendant(@child, NULL),1008, 'Small Boss 5');
go
```

Results | Messages

NodeAsString	NodeAsBinary	Level	employee_id	employee_name
/	0x	0	1000	Big Boss
/1/	0x58	1	1001	Medium Boss 1
/1/1/	0x5AC0	2	1004	Small Boss 1
/1/2/	0x5B40	2	1005	Small Boss 2
/2/	0x68	1	1002	Medium Boss 2
/3/	0x78	1	1003	Medium Boss 3
/3/1/	0x7AC0	2	1006	Small Boss 3
/3/2/	0x7B40	2	1007	Small Boss 4
/3/3/	0x7BC0	2	1008	Small Boss 5

Иерархический тип данных

- GetRoot
- GetLevel
- GetDescendant (child1, child2)
- GetAncestor (n)
- IsDescendantOf
- GetReparentedValue (oldRoot, newRoot)
- Parse
- ToString

Индекс по иерархическому типу

- Depth-first
- Breadth-first

```
CREATE INDEX Breadth ON Employee (node, level);
```

Иерархические запросы в Oracle

```
1  ----- 0 -- SINPLE SELECT
2  SELECT  *
3  FROM EMP;
```

Query Result

SQL | All Rows Fetched: 14 in 0,046 seconds

	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

Иерархические запросы в Oracle

```
5  ----- 1 -- CONNECT BY
6  SELECT   EMPNO,
7           ENAME,
8           MGR
9  FROM EMP
10 START WITH MGR IS NULL
11 CONNECT BY prior EMPNO = MGR
12
```

Query Result

SQL | All Rows Fetched: 14 in 0,016 seconds

	EMPNO	ENAME	MGR
1	7839	KING	(null)
2	7566	JONES	7839
3	7788	SCOTT	7566
4	7876	ADAMS	7788
5	7902	FORD	7566
6	7369	SMITH	7902
7	7698	BLAKE	7839
8	7499	ALLEN	7698
9	7521	WARD	7698
10	7654	MARTIN	7698
11	7844	TURNER	7698
12	7900	JAMES	7698
13	7782	CLARK	7839
14	7934	MILLER	7782

- Start with – начальный узел иерархии
- Connect by – связь текущего с родительским
- Порядок прохода записей – от корня обход в глубину

Иерархические запросы в Oracle

```
15  ----- 2 -- START WITH
16  SELECT   EMPNO,
17           ENAME,
18           MGR
19  FROM EMP
20  START WITH ENAME ='KING'
21  CONNECT BY PRIOR EMPNO = MGR;
22
```

Script Output | Query... | SQL | All Rows Fetched: 14 in 0 seconds

	EMPNO	ENAME	MGR
1	7839	KING	(null)
2	7566	JONES	7839
3	7788	SCOTT	7566
4	7876	ADAMS	7788
5	7902	FORD	7566
6	7369	SMITH	7902
7	7698	BLAKE	7839
8	7499	ALLEN	7698
9	7521	WARD	7698
10	7654	MARTIN	7698
11	7844	TURNER	7698
12	7900	JAMES	7698
13	7782	CLARK	7839
14	7934	MILLER	7782

- Start with – начальный узел иерархии

Иерархические запросы в Oracle

```
23  ----- 3 -- LEVEL
24  SELECT  LEVEL,
25          EMPNO,
26          ENAME,
27          MGR
28  FROM EMP
29  START WITH MGR IS NULL
30  CONNECT BY PRIOR EMPNO = MGR;
31
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 14 in 0 seconds

	LEVEL	EMPNO	ENAME	MGR
1	1	7839	KING	(null)
2	2	7566	JONES	7839
3	3	7788	SCOTT	7566
4	4	7876	ADAMS	7788
5	3	7902	FORD	7566
6	4	7369	SMITH	7902
7	2	7698	BLAKE	7839
8	3	7499	ALLEN	7698
9	3	7521	WARD	7698
10	3	7654	MARTIN	7698
11	3	7844	TURNER	7698
12	3	7900	JAMES	7698
13	2	7782	CLARK	7839
14	3	7934	MILLER	7782

- LEVEL – уровень иерархии

Иерархические запросы в Oracle

```
32  ----- 4 -- LEVEL
33  SELECT  LPAD(' ', 3*LEVEL)||ENAME AS Employee_Name
34  FROM EMP
35  START WITH MGR IS NULL
36  CONNECT BY PRIOR EMPNO = MGR;
37
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 14 in 0,016 seconds

	EMPLOYEE_NAME
1	KING
2	JONES
3	SCOTT
4	ADAMS
5	FORD
6	SMITH
7	BLAKE
8	ALLEN
9	WARD
10	MARTIN
11	TURNER
12	JAMES
13	CLARK
14	MILLER

- LEVEL – уровень иерархии

Иерархические запросы в Oracle

```
38  ----- 5 -- ORDER SIBLINGS BY
39  SELECT LPAD(' ', 3*LEVEL)||ENAME AS Employee_Name
40  FROM EMP
41  START WITH MGR IS NULL
42  CONNECT BY PRIOR EMPNO = MGR
43  ORDER SIBLINGS BY ENAME;
44
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query R

SQL | All Rows Fetched: 14 in 0,016 seconds

	EMPLOYEE_NAME
1	KING
2	BLAKE
3	ALLEN
4	JAMES
5	MARTIN
6	TURNER
7	WARD
8	CLARK
9	MILLER
10	JONES
11	FORD
12	SMITH
13	SCOTT
14	ADAMS

- ORDER SIBLINGS – упорядочение в рамках уровня

Иерархические запросы в Oracle

```
45  ----- 6 -- SYS_CONNECT_BY_PATH
46  SELECT SYS_CONNECT_BY_PATH(ENAME, '/') as Path
47  FROM EMP
48  START WITH MGR IS NULL
49  CONNECT BY PRIOR EMPNO = MGR;
50
```

Script Output | Query Result | Query Result 1 | Query Result 2

SQL | All Rows Fetched: 14 in 0,016 seconds

	PATH
1	/KING
2	/KING/JONES
3	/KING/JONES/SCOTT
4	/KING/JONES/SCOTT/ADAMS
5	/KING/JONES/FORD
6	/KING/JONES/FORD/SMITH
7	/KING/BLAKE
8	/KING/BLAKE/ALLEN
9	/KING/BLAKE/WARD
10	/KING/BLAKE/MARTIN
11	/KING/BLAKE/TURNER
12	/KING/BLAKE/JAMES
13	/KING/CLARK
14	/KING/CLARK/MILLER

- SYS_CONNECT_BY_PATH – материализованный путь в иерархии

Иерархические запросы в Oracle

```
51  ----- 7 -- CONNECT_BY_ISLEAF
52  SELECT   EMPNO,
53           ENAME,
54           LPAD(' ', 3*LEVEL)||ENAME AS Employee_Name,
55           CONNECT_BY_ISLEAF IS_LEAF
56  FROM EMP
57  START WITH MGR IS NULL
58  CONNECT BY PRIOR EMPNO = MGR;
```

Script Output | Query Result | Query Result 1 | Query Result 2 | Query Re

SQL | All Rows Fetched: 14 in 0 seconds

	EMPNO	ENAME	EMPLOYEE_NAME	IS_LEAF
1	7839	KING	KING	0
2	7566	JONES	JONES	0
3	7788	SCOTT	SCOTT	0
4	7876	ADAMS	ADAMS	1
5	7902	FORD	FORD	0
6	7369	SMITH	SMITH	1
7	7698	BLAKE	BLAKE	0
8	7499	ALLEN	ALLEN	1
9	7521	WARD	WARD	1
10	7654	MARTIN	MARTIN	1
11	7844	TURNER	TURNER	1
12	7900	JAMES	JAMES	1
13	7782	CLARK	CLARK	0
14	7934	MILLER	MILLER	1

- CONNECT_BY_ISLEAF – является ли узел листовым

Иерархические запросы в Oracle

```
60  ----- 8 -- CONNECT_BY_ROOT PRIOR
61  SELECT   EMPNO,
62           ENAME,
63           LPAD(' ', 3*LEVEL)||ENAME AS Employee_Name,
64           PRIOR ENAME PARENT_NODE,
65           CONNECT_BY_ISLEAF IS_LEAF,
66           CONNECT_BY_ROOT ENAME ROOT_NODE
67  FROM EMP
68  START WITH MGR IS NULL
69  CONNECT BY PRIOR EMPNO = MGR;
70
```

Query Result

SQL | All Rows Fetched: 14 in 0 seconds

	EMPNO	ENAME	EMPLOYEE_NAME	PARENT_NODE	IS_LEAF	ROOT_N
1	7839	KING	KING	(null)	0	KING
2	7566	JONES	JONES	KING	0	KING
3	7788	SCOTT	SCOTT	JONES	0	KING
4	7876	ADAMS	ADAMS	SCOTT	1	KING
5	7902	FORD	FORD	JONES	0	KING
6	7369	SMITH	SMITH	FORD	1	KING
7	7698	BLAKE	BLAKE	KING	0	KING
8	7499	ALLEN	ALLEN	BLAKE	1	KING
9	7521	WARD	WARD	BLAKE	1	KING
10	7654	MARTIN	MARTIN	BLAKE	1	KING
11	7844	TURNER	TURNER	BLAKE	1	KING
12	7900	JAMES	JAMES	BLAKE	1	KING
13	7782	CLARK	CLARK	KING	0	KING
14	7934	MILLER	MILLER	CLARK	1	KING

- PRIOR – родительский узел
- CONNECT_BY_ROOT – корневой узел иерархии

Иерархические запросы в Oracle

```
----- 9 -- CONNECT_BY_ISCYCLE
CREATE TABLE lngcode(lang_code VARCHAR2(8) NOT NULL,
                     next_lang VARCHAR2(8) NOT NULL,
                     CONSTRAINT tld_pk PRIMARY KEY (lang_code));

INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('RUS', 'ENG');
INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('ENG', 'RUS');
INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('UA', 'RUS');
INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('KZ', 'RUS');
INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('BY', 'UA');
INSERT INTO lngcode (lang_code, next_lang)
     VALUES ('EST', 'ENG');

COMMIT;
```

	LANG_CODE	NEXT_LANG
1	RUS	ENG
2	ENG	RUS
3	UA	RUS
4	KZ	RUS
5	BY	UA
6	EST	ENG

Иерархические запросы в Oracle

```
 96  -- 1
 97  SELECT  lang_code,
 98          next_lang
 99  FROM lngcode
100  START WITH lang_code = 'RUS'
101  CONNECT BY PRIOR lang_code = next_lang;
102
```

Query Result | Script Output | Query R...

SQL | Executing:SELECT lang_code, next_langFROM lngcode

ORA-01436: цикл CONNECT BY в данных пользователя
01436. 00000 - "CONNECT BY loop in user data"
*Cause:
*Action:

- CONNECT_BY_LOOP – зацикливание

Иерархические запросы в Oracle

```
103  -- 2
104  SELECT  lang_code,
105          next_lang
106  FROM lngcode
107  START WITH lang_code = 'RUS'
108  CONNECT BY NOCYCLE
109  PRIOR lang_code = next_lang;
110
```

Script Output | Query... | SQL | All Rows Fetched: 6 in 0 seconds

	LANG_CODE	NEXT_LANG
1	RUS	ENG
2	ENG	RUS
3	EST	ENG
4	KZ	RUS
5	UA	RUS
6	BY	UA

- CONNECT BY NOCYCLE – соединение до зацикливания

Иерархические запросы в Oracle

```
111  -- 3
112  SELECT  lang_code,
113          next_lang,
114          CONNECT_BY_ISCYCLE ISCYCLE
115  FROM lngcode
116  START WITH lang_code = 'RUS'
117  CONNECT BY NOCYCLE
118  PRIOR lang_code = next_lang;
119
```

Script Output | Query Result | Query Result 1

SQL | All Rows Fetched: 6 in 0,016 seconds

	LANG_CODE	NEXT_LANG	ISCYCLE
1	RUS	ENG	0
2	ENG	RUS	1
3	EST	ENG	0
4	KZ	RUS	0
5	UA	RUS	0
6	BY	UA	0

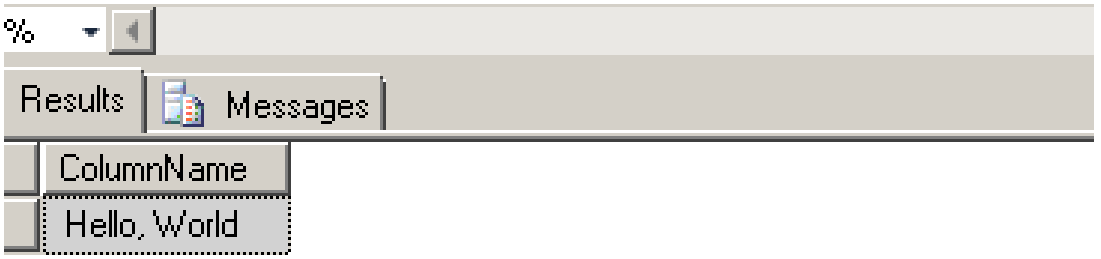
- CONNECT_BY_ISCYCLE – 1, если вызывает зацикливание, 0 – в остальных случаях

CTE

Общие табличные выражения

- Common Table Expressions
- Не создается как объект БД

```
WITH SimpleExample AS

(SELECT 'Hello, World' AS ColumnName
)
SELECT ColumnName from SimpleExample;
```



Общие табличные выражения

- Обобщенные табличные выражения – временные результирующие наборы, определенные в области выполнения инструкций SELECT
- Не сохраняются в базе данных в виде объектов
- Время жизни ограничено продолжительностью запроса
- Могут ссылаться сами на себя
- Один и тот же запрос может ссылаться на CTE несколько раз

Общие табличные выражения

- Нельзя использовать:
  - ORDER BY (только вместе с TOP)
  - INTO
  - OPTION (для хинтов)
- Можно использовать:
  - Несколько CTE
  - UNION (ALL), INTERSECT, EXCEPT
  - Сослаться на предварительно объявленный CTE

Общие табличные выражения

```
WITH
 CountCust (TotalCust, TableName) AS
    (
     SELECT COUNT(CUST_NUM), 'CUSTOMERS' FROM CUSTOMERS
    ) ,
 CountEmpl (TotalEmpl, TableName) AS
    (
     SELECT COUNT(EMPL_NUM), 'SALESREPS' FROM SALESREPS
    )
SELECT TableName, TotalCust FROM CountCust
UNION ALL
SELECT TableName, TotalEmpl FROM CountEmpl;
```

00 %

Results | Messages

	TableName	TotalCust
1	CUSTOMERS	22
2	SALESREPS	12

Общие табличные выражения

```
---------------------- two CTE in SELECT with JOIN
WITH
 MaxCustOrder (Cust, MaxCustOrder) AS
    (
     SELECT CUST, MAX(AMOUNT) FROM ORDERS GROUP BY CUST
    ) ,
 MaxOrder (MaxOrder) AS
    (
     SELECT MAX(AMOUNT) FROM ORDERS
    )
SELECT Cust, MaxCustOrder, MaxOrder, cast (MaxCustOrder/MaxOrder as decimal(4,2)) Percentage
FROM MaxCustOrder, MaxOrder
ORDER BY MaxCustOrder;
```

	Cust	MaxCustOrder	MaxOrder	Percentage
1	2118	1420.00	45000.00	0.03
2	2101	1458.00	45000.00	0.03
3	2106	2130.00	45000.00	0.05
4	2124	2430.00	45000.00	0.05
5	2111	3745.00	45000.00	0.08
6	2120	3750.00	45000.00	0.08
7	2102	3978.00	45000.00	0.09
8	2108	5625.00	45000.00	0.13
9	2114	15000.00	45000.00	0.33
10	2113	22500.00	45000.00	0.50
11	2107	22500.00	45000.00	0.50
12	2103	27500.00	45000.00	0.61
13	2109	31350.00	45000.00	0.70
14	2117	31500.00	45000.00	0.70
15	2112	45000.00	45000.00	1.00

Общие табличные выражения

```
WITH
  MaxCustOrder (Cust, MaxCustOrder) AS
     (
      SELECT CUST, MAX(AMOUNT) FROM ORDERS GROUP BY CUST
     ) ,
  Top3Order (TopOrder) AS
     (
       SELECT TOP 3 MaxCustOrder FROM MaxCustOrder ORDER BY MaxCustOrder DESC
     )
SELECT Cust, MaxCustOrder FROM MaxCustOrder
WHERE MaxCustOrder < ALL(SELECT * FROM Top3Order)
ORDER BY MaxCustOrder;
```

	Cust	MaxCustOrder
1	2118	1420.00
2	2101	1458.00
3	2106	2130.00
4	2124	2430.00
5	2111	3745.00
6	2120	3750.00
7	2102	3978.00
8	2108	5625.00
9	2114	15000.00
10	2113	22500.00
11	2107	22500.00

Common Table Expressions

- Рекурсивность – обращение через CTE к самому себе

```
WITH rec_cte AS (
    SELECT  0  AS data
    UNION all
    SELECT data + 1 FROM rec_cte
    WHERE data < 10)
SELECT data FROM rec_cte;
```

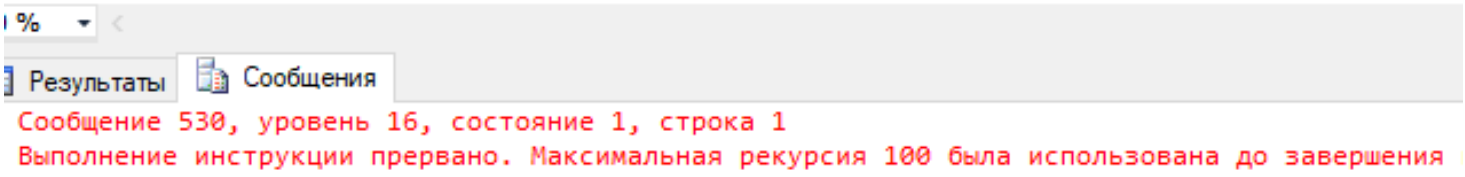
100 %

Результаты | Сообщения

	data
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9
11	10

Common Table Expressions

```
WITH rec_cte AS (
     SELECT  0  AS data
     UNION all
     SELECT data + 1 FROM rec_cte
     )
SELECT data FROM rec_cte;
```



Рекурсивные запросы

```
SELECT * FROM EMP;

WITH X (ENAME, EMPNO)
        AS (
        SELECT ENAME, EMPNO FROM EMP WHERE ENAME = 'JONES'
            UNION ALL
        SELECT E.ENAME, E.EMPNO FROM EMP E, X WHERE X.EMPNO = E.MGR
            )
SELECT ENAME FROM X;
```

Результаты | Сообщения

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	7369	SMITH	CLERK	7902	2012-09-17	800	NULL	20
2	7499	ALLEN	SALESMAN	7698	2017-09-11	1600	300	30
3	7521	WARD	SALESMAN	7698	2015-07-17	1250	500	30
4	7566	JONES	MANAGER	7839	2017-09-19	2975	NULL	20
5	7654	MARTIN	SALESMAN	7698	2017-09-11	1250	400	30
6	7698	BLAKE	MANAGER	7839	2014-01-31	2850	NULL	30
7	7782	CLARK	MANAGER	7839	2013-02-21	2450	NULL	10
8	7788	SCOTT	ANALYST	7566	2017-01-11	3000	NULL	20
9	7839	KING	PRESIDENT	NULL	2011-09-11	5000	NULL	10
10	7844	TURNER	SALESMAN	7698	2017-03-22	1500	0	30
11	7876	ADAMS	CLERK	7788	2018-07-13	1100	NULL	20
12	7900	JAMES	CLERK	7698	2016-11-11	950	NULL	30
13	7902	FORD	ANALYST	7566	2017-03-11	3000	NULL	20
14	7934	MILLER	CLERK	7782	2018-03-12	1300	NULL	10

Результаты

	ENAME
1	JONES
2	SCOTT
3	FORD
4	SMITH
5	ADAMS

ORACLE CTE

```
1  -- CTE Example
2  WITH CTE_year AS -- cte name
3     (
4     SELECT EXTRACT( YEAR FROM hiredate) AS hireyear, -- select query
5            empno
6     FROM emp
7     )
8  SELECT hireyear,
9         COUNT(DISTINCT empno) AS emp_count
10 FROM CTE_year -- use of CTE query
11 GROUP BY hireyear
12 ORDER BY hireyear;
```

	HIREYEAR	EMP_COUNT
1	1980	1
2	1981	10
3	1982	3
4	1983	1
5	2001	4
6	2018	13

ORACLE CTE

```
14  -- Recursive CTE 1
15  WITH
16  numbers ( N ) -- cte name with column list as parameters
17  AS (
18        SELECT 0 AS N FROM DUAL          -- start from one line
19         UNION ALL                       -- union lines
20        SELECT N + 2 AS N                -- with previous
21         FROM   NUMBERS                  -- use CTE as data source
22         WHERE  n < 10                    -- where to stop recursion
23  )
24  SELECT N FROM numbers;                 -- use of CTE query
```

	N
1	0
2	2
3	4
4	6
5	8
6	10

ORACLE CTE

```
27  -- Recursive CTE 2
28  WITH EmpOrg_CTE (empno, mgr, ename) AS
29  (
30      SELECT  empno,
31              mgr,
32              ename
33        FROM EMP
34        WHERE mgr is null              -- starting point
35    UNION ALL
36      SELECT  child_table.empno,
37              CHILD_TABLE.MGR,
38              child_table.ename      -- recursive member
39        FROM EmpOrg_CTE parent_table
40        JOIN Emp child_table
41        ON child_table.mgr=parent_table.empno
42   )
43  SELECT EMPNO, MGR, ENAME FROM EMPORG_CTE; -- use of CTE query
```

	EMPNO	MGR	ENAME
1	7839	(null)	KING
2	7698	7839	BLAKE
3	7782	7839	CLARK
4	7566	7839	JONES
5	7654	7698	MARTIN
6	7499	7698	ALLEN
7	7844	7698	TURNER
8	7900	7698	JAMES
9	7521	7698	WARD
10	8002	7698	Jonson
11	8499	7698	ALIEN
12	8654	7698	MART
13	8844	7698	TURNUR
14	7901	7698	JOHN

ORACLE CTE

```
45  -- table
46  CREATE TABLE ROUTE (
47     CITY_FROM VARCHAR2(20),
48     CITY_TO   VARCHAR2(20),
49     DISTANCE NUMBER(5)
50  );
51
52  insert into ROUTE values ('Minsk', 'Warsaw', 1021);
53  insert into ROUTE values ('Minsk', 'Kiev', 634);
54  insert into ROUTE values ('Kiev', 'Odessa', 390);
55  insert into ROUTE values ('Warsaw', 'Praha', 435);
56  insert into ROUTE values ('Praha', 'Kadis', 758);
57  insert into ROUTE values ('Kadis', 'Barselona', 622);
58  COMMIT;
```

ORACLE CTE

```
60  -- summarize km
61  WITH CITY_CYCLE_CTE (city_to, way, distance)
62  AS
63  (
64      SELECT CITY_TO,
65             CITY_FROM || '-' || CITY_TO way,
66             DISTANCE
67        from   ROUTE
68        WHERE  CITY_FROM = 'Minsk'
69    UNION ALL
70      SELECT R.CITY_TO,
71             C.WAY || '-' || R.CITY_to,
72             R.DISTANCE + C.DISTANCE
73      FROM ROUTE R JOIN CITY_CYCLE_CTE C
74           ON ( C.CITY_TO = R.CITY_FROM )
75    )
76  SELECT way, distance FROM CITY_CYCLE_CTE;
77
```

	WAY	DISTANCE
1	Minsk-Warsaw	1021
2	Minsk-Kiev	634
3	Minsk-Kiev-Odessa	1024
4	Minsk-Warsaw-Praha	1456
5	Minsk-Warsaw-Praha-Kadis	2214
6	Minsk-Warsaw-Praha-Kadis-Barselona	2836

ORACLE CTE

```
79  -- cycle
80  insert into ROUTE values ('Barselona', 'Kadis', 622);
81
82  WITH CITY_CYCLE_CTE (city_to, way, distance)
83  AS
84  (
85      SELECT CITY_TO,
86             CITY_FROM || '-' || CITY_TO way,
87             DISTANCE
88        from   ROUTE
89        WHERE  CITY_FROM = 'Minsk'
90    UNION ALL
91      SELECT R.CITY_TO,
92             C.WAY || '-' || R.CITY_to,
93             R.DISTANCE + C.DISTANCE
94      FROM ROUTE R JOIN CITY_CYCLE_CTE C
95           ON ( C.CITY_TO = R.CITY_FROM )
96    )
97    cycle CITY_TO set IS_CYCLED to 'X' default '-'
98  select WAY, DISTANCE, IS_CYCLED  from CITY_CYCLE_CTE;
```

	WAY	DISTANCE	IS_CYCLED
1	Minsk-Warsaw	1021	-
2	Minsk-Kiev	634	-
3	Minsk-Kiev-Odessa	1024	-
4	Minsk-Warsaw-Praha	1456	-
5	Minsk-Warsaw-Praha-Kadis	2214	-
6	Minsk-Warsaw-Praha-Kadis-Barselona	2836	-
7	Minsk-Warsaw-Praha-Kadis-Barselona-Kadis	3458	X

```
rollback;
```

ORACLE CTE

```
103 -- order by
104 WITH CITY_CYCLE_CTE (city_to, way, distance)
105 AS
106 (
107     SELECT CITY_TO,
108            CITY_FROM || '-' || CITY_TO way,
109            DISTANCE
110       from   ROUTE
111       WHERE  CITY_FROM = 'Minsk'
112   UNION ALL
113     SELECT R.CITY_TO,
114            C.WAY || '-' || R.CITY_to,
115            R.DISTANCE + C.DISTANCE
116     FROM ROUTE R JOIN CITY_CYCLE_CTE C
117          ON ( C.CITY_TO = R.CITY_FROM )
118   )
119 SEARCH DEPTH FIRST BY CITY_TO ASC SET sort_order -- BREADTH/DEPTH
120 SELECT CITY_TO, WAY, DISTANCE  FROM CITY_CYCLE_CTE
121 ORDER BY sort_order;
```

	CITY_TO	WAY	DISTANCE
1	Kiev	Minsk-Kiev	634
2	Odessa	Minsk-Kiev-Odessa	1024
3	Warsaw	Minsk-Warsaw	1021
4	Praha	Minsk-Warsaw-Praha	1456
5	Kadis	Minsk-Warsaw-Praha-Kadis	2214
6	Barselona	Minsk-Warsaw-Praha-Kadis-Barselona	2836

ORACLE CTE

```
103 -- order by
104 WITH CITY_CYCLE_CTE (city_to, way, distance)
105 AS
106 (
107     SELECT CITY_TO,
108            CITY_FROM || '-' || CITY_TO way,
109            DISTANCE
110       from   ROUTE
111       WHERE  CITY_FROM = 'Minsk'
112   UNION ALL
113     SELECT R.CITY_TO,
114            C.WAY || '-' || R.CITY_to,
115            R.DISTANCE + C.DISTANCE
116     FROM ROUTE R JOIN CITY_CYCLE_CTE C
117          ON ( C.CITY_TO = R.CITY_FROM )
118   )
119 SEARCH DEPTH FIRST BY CITY_TO ASC SET sort_order -- BREADTH/DEPTH
120 SELECT CITY_TO, WAY, DISTANCE  FROM CITY_CYCLE_CTE
121 ORDER BY sort_order;
```

	CITY_TO	WAY	DISTANCE
1	Kiev	Minsk-Kiev	634
2	Odessa	Minsk-Kiev-Odessa	1024
3	Warsaw	Minsk-Warsaw	1021
4	Praha	Minsk-Warsaw-Praha	1456
5	Kadis	Minsk-Warsaw-Praha-Kadis	2214
6	Barselona	Minsk-Warsaw-Praha-Kadis-Barselona	2836

ORACLE CTE

```
103 -- order by
104 WITH CITY_CYCLE_CTE (city_to, way, distance)
105 AS
106 (
107     SELECT CITY_TO,
108            CITY_FROM || '-' || CITY_TO way,
109            DISTANCE
110       from   ROUTE
111       WHERE  CITY_FROM = 'Minsk'
112   UNION ALL
113     SELECT R.CITY_TO,
114            C.WAY || '-' || R.CITY_to,
115            R.DISTANCE + C.DISTANCE
116     FROM ROUTE R JOIN CITY_CYCLE_CTE C
117          ON ( C.CITY_TO = R.CITY_FROM )
118   )
119 SEARCH BREADTH FIRST BY CITY_TO ASC SET sort_order -- BREADTH/DEPTH
120 SELECT CITY_TO, WAY, DISTANCE  FROM CITY_CYCLE_CTE
121 ORDER BY sort_order;
```

	CITY_TO	WAY	DISTANCE
1	Kiev	Minsk-Kiev	634
2	Warsaw	Minsk-Warsaw	1021
3	Odessa	Minsk-Kiev-Odessa	1024
4	Praha	Minsk-Warsaw-Praha	1456
5	Kadis	Minsk-Warsaw-Praha-Kadis	2214
6	Barselona	Minsk-Warsaw-Praha-Kadis-Barselona	2836

Используются для

- Создания рекурсивных запросов
- Многократных ссылок на результирующую таблицу из одной и той же инструкции
- Замены представлений
- Группирования по столбцу, производного от скалярного подзапроса выборки или функции, которая недетерминирована
- https://www.mssqltips.com/sqlservertip/5379/sql-server-common-table-expressions-cte-usage-and-examples/

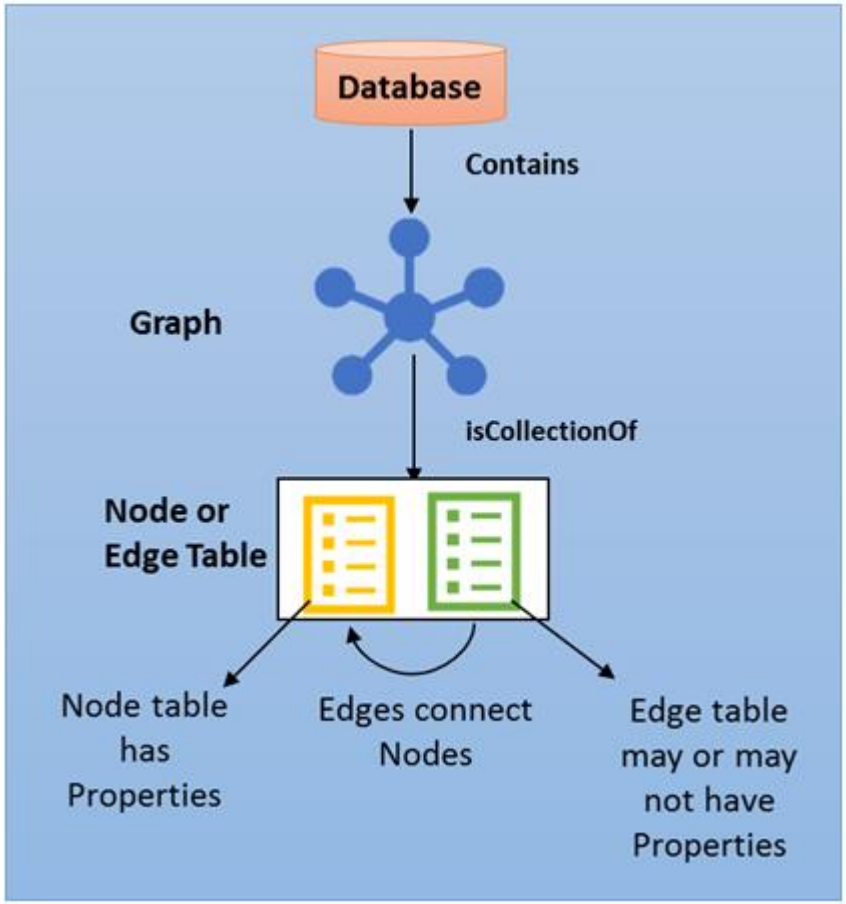
CTE

- Позволяет значительно повысить читаемость и упростить работу со сложными запросами
- Можно составлять промежуточные CTE
- Могут быть определены в пользовательских подпрограммах
- Повышает ли CTE производительность запроса?

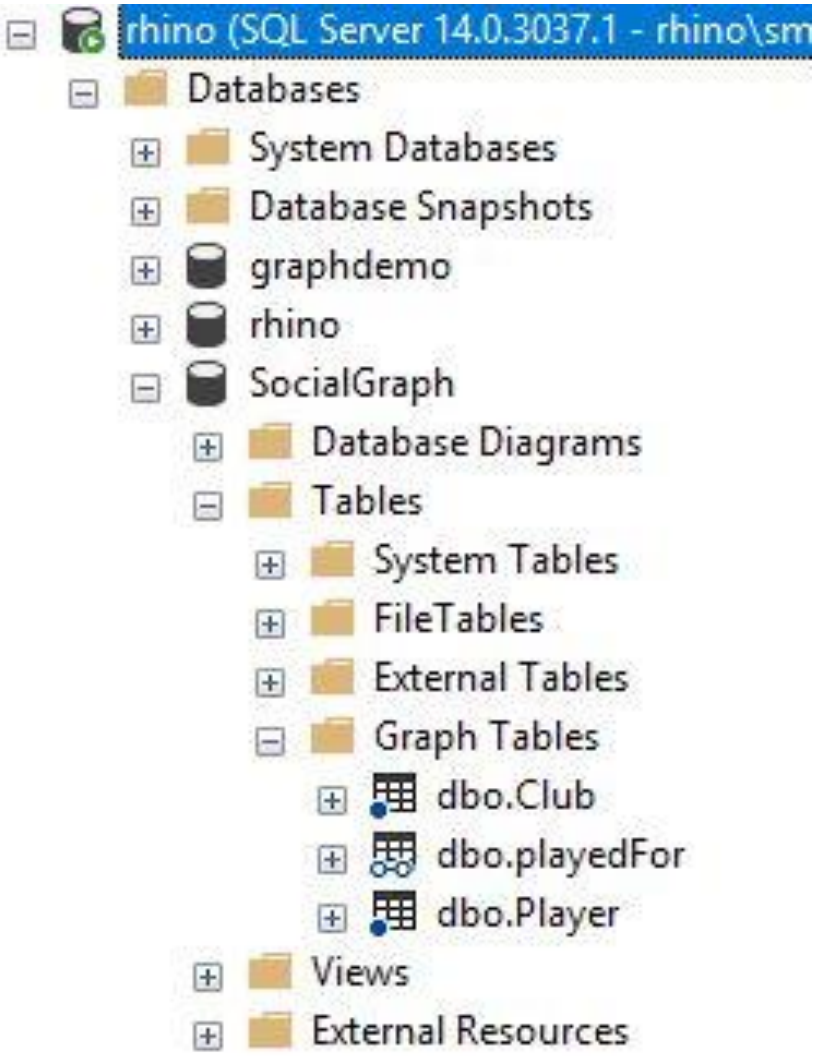
Графы

Графовые базы данных

- С SQL Server 2017
- Можно создать один граф на базу данных
- Граф состоит из таблиц узлов и ребер



Простой пример



Простой пример

```
--- table creation
CREATE TABLE dbo.Club(
    ID int IDENTITY(1,1) NOT NULL,
    name varchar(100) NULL)
AS NODE;

CREATE TABLE dbo.playedFor
AS EDGE;

CREATE TABLE dbo.Player(
    ID int IDENTITY(1,1) NOT NULL,
    name varchar(100) NULL)
AS NODE;

--- insert into nodes


INSERT INTO Club(name)
VALUES('Tottenham Hotspur'), ('Chelsea'), ('Manchester City'), ('Arsenal'), ('West Ham United');
GO
INSERT INTO Player(name)
VALUES('Frank Lampard'), ('Petr Cech'), ('Cesc Fabregas'), ('Gael Clichy'), ('William Gallas');
GO
```

Простой пример

```
--- insert into edges

-------Frank Lampard
INSERT playedFor ($to_id,$from_id)
    VALUES
    ((SELECT $node_id FROM dbo.Club WHERE ID = 1), (SELECT $node_id FROM dbo.Player WHERE ID = 1)),
    ((SELECT $node_id FROM dbo.Club WHERE ID = 2), (SELECT $node_id FROM dbo.Player WHERE ID = 1)),
    ((SELECT $node_id FROM dbo.Club WHERE ID = 5), (SELECT $node_id FROM dbo.Player WHERE ID = 1))

-------William Gallas
INSERT playedFor ($to_id,$from_id)
    VALUES
    ((SELECT $node_id FROM dbo.Club WHERE ID = 1), (SELECT $node_id FROM dbo.Player WHERE ID = 5)),
    ((SELECT $node_id FROM dbo.Club WHERE ID = 3), (SELECT $node_id FROM dbo.Player WHERE ID = 5)),
    ((SELECT $node_id FROM dbo.Club WHERE ID = 2), (SELECT $node_id FROM dbo.Player WHERE ID = 5))
```

Пример посложнее

```
-- Create NODE tables
CREATE TABLE Person (
  ID INTEGER PRIMARY KEY,
  name VARCHAR(100)
) AS NODE;

CREATE TABLE Restaurant (
  ID INTEGER NOT NULL,
  name VARCHAR(100),
  city VARCHAR(100)
) AS NODE;

CREATE TABLE City (
  ID INTEGER PRIMARY KEY,
  name VARCHAR(100),
  stateName VARCHAR(100)
) AS NODE;

-- Create EDGE tables.
CREATE TABLE likes (rating INTEGER) AS EDGE;
CREATE TABLE friendOf AS EDGE;
CREATE TABLE livesIn AS EDGE;
CREATE TABLE locatedIn AS EDGE;
```

Графовые базы данных

```
-- Insert data into node tables. Inserting into a node table is same as inserting into a regular table
INSERT INTO Person VALUES (1,'John');
INSERT INTO Person VALUES (2,'Mary');
INSERT INTO Person VALUES (3,'Alice');
INSERT INTO Person VALUES (4,'Jacob');
INSERT INTO Person VALUES (5,'Julie');

INSERT INTO Restaurant VALUES (1,'Taco Dell','Bellevue');
INSERT INTO Restaurant VALUES (2,'Ginger and Spice','Seattle');
INSERT INTO Restaurant VALUES (3,'Noodle Land', 'Redmond');

INSERT INTO City VALUES (1,'Bellevue','wa');
INSERT INTO City VALUES (2,'Seattle','wa');
INSERT INTO City VALUES (3,'Redmond','wa');
```

Графовые базы данных

```
-- Insert into edge table. While inserting into an edge table,
-- you need to provide the $node_id from $from_id and $to_id columns.
INSERT INTO likes VALUES ((SELECT $node_id FROM Person WHERE id = 1),
       (SELECT $node_id FROM Restaurant WHERE id = 1),9);
INSERT INTO likes VALUES ((SELECT $node_id FROM Person WHERE id = 2),
      (SELECT $node_id FROM Restaurant WHERE id = 2),9);
INSERT INTO likes VALUES ((SELECT $node_id FROM Person WHERE id = 3),
      (SELECT $node_id FROM Restaurant WHERE id = 3),9);
INSERT INTO likes VALUES ((SELECT $node_id FROM Person WHERE id = 4),
      (SELECT $node_id FROM Restaurant WHERE id = 3),9);
INSERT INTO likes VALUES ((SELECT $node_id FROM Person WHERE id = 5),
      (SELECT $node_id FROM Restaurant WHERE id = 3),9);

INSERT INTO livesIn VALUES ((SELECT $node_id FROM Person WHERE id = 1),
      (SELECT $node_id FROM City WHERE id = 1));
INSERT INTO livesIn VALUES ((SELECT $node_id FROM Person WHERE id = 2),
      (SELECT $node_id FROM City WHERE id = 2));
INSERT INTO livesIn VALUES ((SELECT $node_id FROM Person WHERE id = 3),
      (SELECT $node_id FROM City WHERE id = 3));
INSERT INTO livesIn VALUES ((SELECT $node_id FROM Person WHERE id = 4),
      (SELECT $node_id FROM City WHERE id = 3));
INSERT INTO livesIn VALUES ((SELECT $node_id FROM Person WHERE id = 5),
      (SELECT $node_id FROM City WHERE id = 1));

INSERT INTO locatedIn VALUES ((SELECT $node_id FROM Restaurant WHERE id = 1),
      (SELECT $node_id FROM City WHERE id =1));
INSERT INTO locatedIn VALUES ((SELECT $node_id FROM Restaurant WHERE id = 2),
      (SELECT $node_id FROM City WHERE id =2));
INSERT INTO locatedIn VALUES ((SELECT $node_id FROM Restaurant WHERE id = 3),
      (SELECT $node_id FROM City WHERE id =3));
```

Графовые базы данных

```
-- Insert data into the friendof edge.
INSERT INTO friendof VALUES ((SELECT $NODE_ID FROM person WHERE ID = 1), (SELECT $NODE_ID FROM person WHERE ID = 2));
INSERT INTO friendof VALUES ((SELECT $NODE_ID FROM person WHERE ID = 2), (SELECT $NODE_ID FROM person WHERE ID = 3));
INSERT INTO friendof VALUES ((SELECT $NODE_ID FROM person WHERE ID = 3), (SELECT $NODE_ID FROM person WHERE ID = 1));
INSERT INTO friendof VALUES ((SELECT $NODE_ID FROM person WHERE ID = 4), (SELECT $NODE_ID FROM person WHERE ID = 2));
INSERT INTO friendof VALUES ((SELECT $NODE_ID FROM person WHERE ID = 5), (SELECT $NODE_ID FROM person WHERE ID = 4));
```

Графовые базы данных

```
-- What is in the NODE table?
SELECT *   FROM [graphdemo].[dbo].[Person]

-- What is in the EDGE table?
SELECT *   FROM [graphdemo].[dbo].likes
```

Results | Messages

	$node_id_E048B68446464460B33F184DA35B8F01	ID	name
1	{"type":"node","schema":"dbo","table":"Person","id":0}	1	John
2	{"type":"node","schema":"dbo","table":"Person","id":1}	2	Mary
3	{"type":"node","schema":"dbo","table":"Person","id":2}	3	Alice
4	{"type":"node","schema":"dbo","table":"Person","id":3}	4	Jacob
5	{"type":"node","schema":"dbo","table":"Person","id":4}	5	Julie

	$edge_id_4885BEF9341F4FC7A3C1F5AEF5D23ADD	$from_id_1A38B6D2842D4D85B3B8B0EE8C08C49F	$to_id_7F5C1B8F909B4B2A9587E7F126554591	rating
1	{"type":"edge","schema":"dbo","table":"likes","id":0}	{"type":"node","schema":"dbo","table":"Person","i...	{"type":"node","schema":"dbo","table":"Restauran...	9
2	{"type":"edge","schema":"dbo","table":"likes","id":1}	{"type":"node","schema":"dbo","table":"Person","i...	{"type":"node","schema":"dbo","table":"Restauran...	9
3	{"type":"edge","schema":"dbo","table":"likes","id":2}	{"type":"node","schema":"dbo","table":"Person","i...	{"type":"node","schema":"dbo","table":"Restauran...	9
4	{"type":"edge","schema":"dbo","table":"likes","id":3}	{"type":"node","schema":"dbo","table":"Person","i...	{"type":"node","schema":"dbo","table":"Restauran...	9
5	{"type":"edge","schema":"dbo","table":"likes","id":4}	{"type":"node","schema":"dbo","table":"Person","i...	{"type":"node","schema":"dbo","table":"Restauran...	9

Графовые базы данных

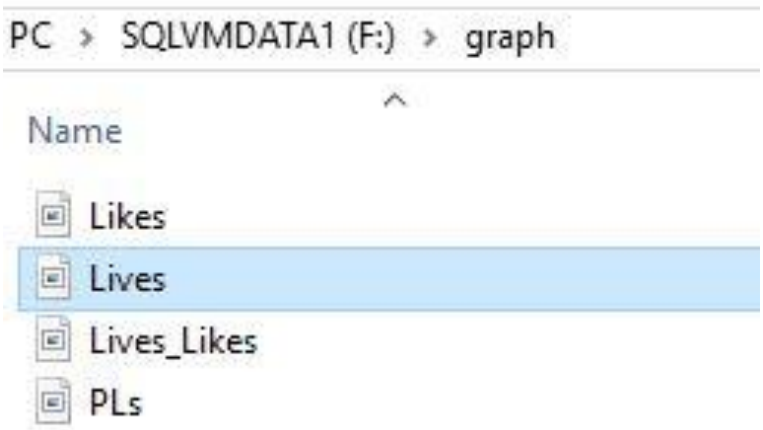
```
-- Find Restaurants that John likes
SELECT Restaurant.name
FROM Person, likes, Restaurant
WHERE MATCH (Person-(likes)->Restaurant)
AND Person.name = 'John';

-- Find Restaurants that John's friends like
SELECT Restaurant.name
FROM Person person1, Person person2, likes, friendOf, Restaurant
WHERE MATCH(person1-(friendOf)->person2-(likes)->Restaurant)
AND person1.name='John';

-- Find people who like a restaurant in the same city they live in
SELECT Person.name, Restaurant.name, City.name
FROM Person, likes, Restaurant, livesIn, City, locatedIn
WHERE MATCH (Person-(likes)->Restaurant-(locatedIn)->City AND Person-(livesIn)->City);
```

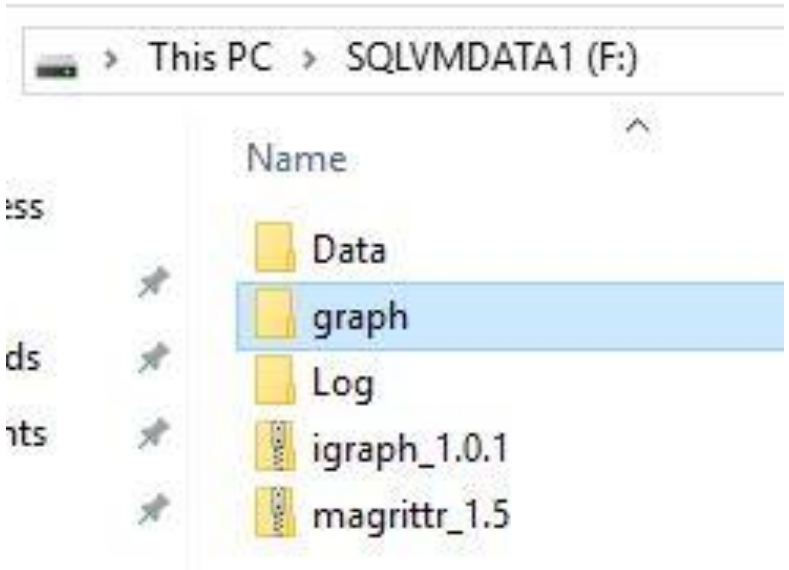
	name
1	Taco Dell

	name
1	Ginger and Spice

	name	name	name
1	John	Taco Dell	Bellevue
2	Mary	Ginger and Spice	Seattle
3	Alice	Noodle Land	Redmond
4	Jac...	Noodle Land	Redmond

Визуализация графа

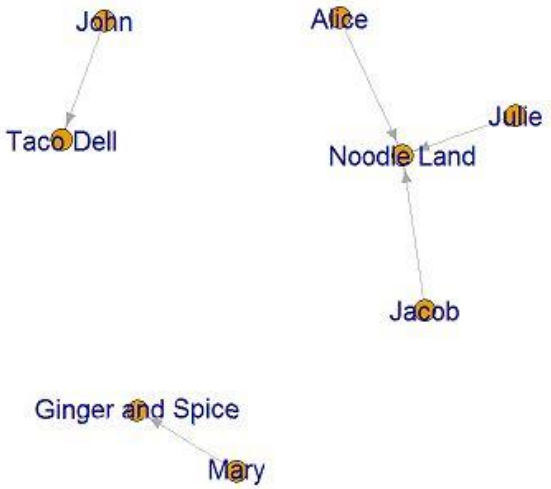





```
--- install packages graph and magrittr to use visualization by R

sp_configure 'external scripts enabled', 1;
RECONFIGURE WITH OVERRIDE;

sp_configure;
```

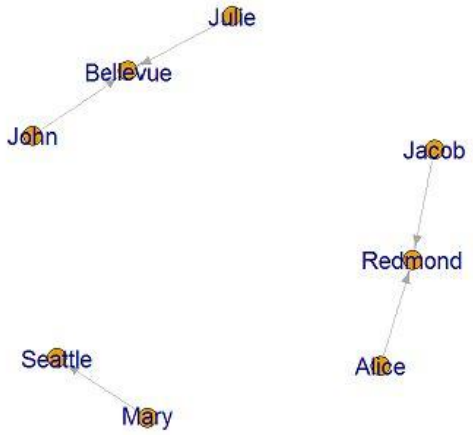
Визуализация графа

```
--- visualize Person-(likes)->Restaurant
EXECUTE sp_execute_external_script @language = N'R',
@script = N'
    require(igraph)
    g <- graph.data.frame(graphdf)
    V(g)$label.cex <- 2
    png(filename = "f:\\graph\\Likes.png", height = 800, width = 1500, res = 100);
    plot(g, vertex.label.family = "sans", vertex.size = 10)
    dev.off()',
@input_data_1 = N'
    SELECT a.name, b.name as Name FROM Person a, likes, Restaurant b
        WHERE MATCH(a-(likes)->b);',
@input_data_1_name = N'graphdf'
GO
```



Визуализация графа

```
-- visualize Person-(lives)->City
EXECUTE sp_execute_external_script @language = N'R',
@script = N'
    require(igraph)
    g <- graph.data.frame(graphdf)
    V(g)$label.cex <- 2
    png(filename = "f:\\graph\\Lives.png", height = 800, width = 1500, res = 100);
    plot(g, vertex.label.family = "sans", vertex.size = 10)
    dev.off()',
@input_data_1 = N'
    SELECT a.name, b.name as Name FROM Person a, livesIn, City b
        WHERE MATCH(a-(livesIn)->b);',
@input_data_1_name = N'graphdf'
GO
```



Вопросы?