



Common Table Expressions in Firebird

Tutorial



COMMON TABLE EXPRESSIONS



• Syntax

- **WITH [RECURSIVE]** *-- new keywords*
 - **CTE_A** *-- first table expression's name*
 [(a1, a2, ...)] *-- fields aliases, optional*
 AS (SELECT ...), *-- table expression's definition*
 - **CTE_B** *-- second table expression*
 [(b1, b2, ...)]
 AS (SELECT ...),
 - ...
- **SELECT ...** *-- main query, used both*
 FROM **CTE_A**, **CTE_B**, *-- table expressions*
 TAB1, **TAB2** *-- and regular tables*
 WHERE ...



ONE SIMPLE CTE



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep')
```

SELECT *

```
FROM SALE_REPS
```



TWO SIMPLE INDEPENDENT CTE's



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep'),
```

MILAN_DEPT AS

```
(SELECT DEPT_NO, DEPARTMENT  
  FROM DEPARTMENT  
  WHERE LOCATION = 'Milan')
```

SELECT *

```
FROM SALE_REPS JOIN MILAN_DEPT  
  ON SALE_REPS.DEPT_NO = MILAN_DEPT.DEPT_NO
```



TWO DEPENDENT CTE's



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep'),
```

```
SALE_REPS_BY_DEPT AS  
(SELECT DEPT_NO, COUNT(*) AS CNT_EMPS  
  FROM SALE_REPS  
  GROUP BY DEPT_NO)
```

```
SELECT *  
FROM SALE_REPS_BY_DEPT NATURAL JOIN DEPARTMENT
```



DERIVED TABLE VS CTE



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep')
```

```
SELECT * FROM SALE_REPS
```

SELECT * FROM

```
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep') AS  
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO)
```



DERIVED TABLE VS CTE



WITH

```
DEPT_YEAR_BUDGET AS (  
    SELECT FISCAL_YEAR, DEPT_NO, SUM(PROJECTED_BUDGET) AS BUDGET  
    FROM PROJ_DEPT_BUDGET  
    GROUP BY FISCAL_YEAR, DEPT_NO  
)
```

```
SELECT D.DEPT_NO, D.DEPARTMENT,  
       B_1993.BUDGET AS B_1993, B_1994.BUDGET AS B_1994,  
       B_1995.BUDGET AS B_1995, B_1996.BUDGET AS B_1996  
FROM DEPARTMENT D  
     LEFT JOIN DEPT_YEAR_BUDGET B_1993  
ON D.DEPT_NO = B_1993.DEPT_NO AND B_1993.FISCAL_YEAR = 1993  
     LEFT JOIN DEPT_YEAR_BUDGET B_1994  
ON D.DEPT_NO = B_1994.DEPT_NO AND B_1994.FISCAL_YEAR = 1994  
     LEFT JOIN DEPT_YEAR_BUDGET B_1995  
ON D.DEPT_NO = B_1995.DEPT_NO AND B_1995.FISCAL_YEAR = 1995  
     LEFT JOIN DEPT_YEAR_BUDGET B_1996  
ON D.DEPT_NO = B_1996.DEPT_NO AND B_1996.FISCAL_YEAR = 1996
```



DERIVED TABLE VS CTE



```
SELECT D.DEPT_NO, D.DEPARTMENT,  
       B_1993.BUDGET AS B_1993, B_1994.BUDGET AS B_1994,  
       B_1995.BUDGET AS B_1995  
FROM DEPARTMENT D  
     LEFT JOIN (SELECT DEPT_NO, SUM(PROJECTED_BUDGET) AS BUDGET  
                FROM PROJ_DEPT_BUDGET WHERE FISCAL_YEAR = 1993  
                GROUP BY FISCAL_YEAR, DEPT_NO) AS B_1993  
ON D.DEPT_NO = B_1993.DEPT_NO  
  
     LEFT JOIN (SELECT DEPT_NO, SUM(PROJECTED_BUDGET) AS BUDGET  
                FROM PROJ_DEPT_BUDGET WHERE FISCAL_YEAR = 1994  
                GROUP BY FISCAL_YEAR, DEPT_NO) AS B_1994  
ON D.DEPT_NO = B_1994.DEPT_NO  
  
     LEFT JOIN (SELECT DEPT_NO, SUM(PROJECTED_BUDGET) AS BUDGET  
                FROM PROJ_DEPT_BUDGET WHERE FISCAL_YEAR = 1995  
                GROUP BY FISCAL_YEAR, DEPT_NO) AS B_1995  
ON D.DEPT_NO = B_1995.DEPT_NO
```



OUTER BOOLEANS PUSHED INSIDE



Table PROJ_DEPT_BUDGET :

Total rows = 24

Indexed reads = 24

PLAN

```
JOIN (
  JOIN (
    JOIN (
      JOIN (
        D NATURAL,
        SORT (B_1993 PROJ_DEPT_BUDGET INDEX (RDB$FOREIGN18, RDB$PRIMARY17))
      ),
      SORT (B_1994 PROJ_DEPT_BUDGET INDEX (RDB$FOREIGN18, RDB$PRIMARY17))
    ),
    SORT (B_1995 PROJ_DEPT_BUDGET INDEX (RDB$FOREIGN18, RDB$PRIMARY17))
  ),
  SORT (B_1996 PROJ_DEPT_BUDGET INDEX (RDB$FOREIGN18, RDB$PRIMARY17))
)
```

SELECT...

FROM DEPARTMENT D

LEFT JOIN DEPT_YEAR_BUDGET B_1993

ON D.DEPT_NO = B_1993.DEPT_NO

AND B_1993.FISCAL_YEAR = 1993

...

Used indices :

ALTER TABLE PROJ_DEPT_BUDGET ADD PRIMARY KEY (FISCAL_YEAR, PROJ_ID, DEPT_NO);

ALTER TABLE PROJ_DEPT_BUDGET ADD FOREIGN KEY (DEPT_NO)

REFERENCES DEPARTMENT (DEPT_NO);



PLAN : ONE SIMPLE CTE



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep')
```

SELECT *

```
  FROM SALE_REPS
```

```
PLAN (SALE_REPS EMPLOYEE INDEX (RDB$FOREIGN9))
```

SALE_REPS EMPLOYEE - concatenated name, built from
derived table name (*SALE_REPS*) and
real table name (*EMPLOYEE*)



PLAN : TWO DEPENDENT CTE's



WITH

```
SALE_REPS (EMP_NO, FULL_NAME, PHONE_EXT, DEPT_NO) AS  
(SELECT EMP_NO, FIRST_NAME || ' ' || LAST_NAME, PHONE_EXT, DEPT_NO  
  FROM EMPLOYEE  
  WHERE JOB_CODE = 'SRep'),
```

```
SALE_REPS_BY_DEPT AS  
(SELECT DEPT_NO, COUNT(*) AS CNT_EMPS  
  FROM SALE_REPS  
  GROUP BY DEPT_NO)
```

SELECT *

```
FROM SALE_REPS_BY_DEPT NATURAL JOIN DEPARTMENT
```

```
PLAN JOIN (SALE_REPS_BY_DEPT SALE_REPS EMPLOYEE ORDER RDB$FOREIGN8  
INDEX (RDB$FOREIGN9), DEPARTMENT INDEX (RDB$PRIMARY5))
```

SALE_REPS_BY_DEPT SALE_REPS EMPLOYEE - triple name, as derived table
SALE_REPS_BY_DEPT used another derived table *SALE_REPS*



RECURSIVE TABLE EXPRESSIONS



• Syntax

- **WITH RECURSIVE** *-- mandatory keyword **RECURSIVE***
- **CTE_R AS (** *-- recursive table expression*
 - **SELECT ...** *-- non recursive query (anchor member)*
 FROM T1
 - **UNION ALL** *-- mandatory UNION ALL*
 - **SELECT ...** *-- recursive query*
 FROM CTE_R *-- recursive reference*
- **)**
- **...** *-- another table expressions*
- **SELECT ...** *-- main query*
 FROM CTE_R ...
 WHERE ...



SIMPLE RECURSIVE CTE



```
WITH RECURSIVE
  DEPT_TREE AS (
    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT
      FROM DEPARTMENT D
     WHERE D.HEAD_DEPT IS NULL

    UNION ALL

    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT
      FROM DEPT_TREE H JOIN DEPARTMENT D
     ON D.HEAD_DEPT = H.DEPT_NO
  )

SELECT * FROM DEPT_TREE
```

Root item

Next level



HOW IT WORKS : ALGORITHM



1. Combine all *non-recursive* members into *anchor* query
2. Combine all *recursive* members into *recursive* query
3. Remove *self-reference* from recursive query and replace all its fields by parameters
4. Execute anchor query
5. For each row in result set :
 - a) Return row back to application
 - b) Push current result set into stack
 - c) Execute recursive query with just fetched values as parameters
 - d) If new result set is not empty, go to step (5) with new result set
 - e) Else pop result set from stack and continue to process its next row



HOW IT WORKS : PREPARE



1. Combine all *non-recursive* members into *anchor query*

```
SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT
FROM DEPARTMENT D
WHERE D.HEAD_DEPT IS NULL
```

2. Combine all *recursive* members into *recursive query*

```
SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT
FROM DEPT_TREE H JOIN DEPARTMENT D
ON D.HEAD_DEPT = H.DEPT_NO
```

3. Remove self-reference from recursive query and replace all its fields by parameters

```
SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT
FROM DEPARTMENT D
WHERE D.HEAD_DEPT = :HEAD_DEPT
```



HOW IT WORKS : EXECUTE



4. Execute anchor query

```
SELECT DEPT_NO HEAD_DEPT DEPARTMENT
FROM   EMPLOYEES
WHERE  D.HEAD_DEPT IS NULL
```

5.1.1 a) return row (DEPT_NO = 000) to client

5.1.1 b) push current result set into stack

5.1.1 c) execute recursive query using
fetched row values as parameters

```
SELECT DEPT_NO HEAD_DEPT DEPARTMENT
FROM   EMPLOYEES
WHERE  D.HEAD_DEPT = 000
```

5.2.1 a) return row (DEPT_NO = 100) to client

5.2.1 b) push current result set into stack

5.2.1 c) execute recursive query using
fetched row values as parameters

```
SELECT DEPT_NO HEAD_DEPT DEPARTMENT
FROM   EMPLOYEES
WHERE  D.HEAD_DEPT = 100
```



HOW IT WORKS : RESULTS



DEPT_NO	HEAD_DEPT	DEPARTMENT
000		Corporate Headquarters

DEPT_NO	HEAD_DEPT	DEPARTMENT
100	000	Sales and Marketing
600	000	Engineering
900	000	Finance

DEPT_NO	HEAD_DEPT	DEPARTMENT
180	100	Marketing
130	100	Field Office: East Coast
140	100	Field Office: Canada
110	100	Pacific Rim Headquarters
120	100	European Headquarters

DEPT_NO	HEAD_DEPT	DEPARTMENT
115	110	Field Office: Japan
116	110	Field Office: Singapore

DEPT_NO	HEAD_DEPT	DEPARTMENT
121	120	Field Office: Switzerland
123	120	Field Office: France
125	120	Field Office: Italy

DEPT_NO	HEAD_DEPT	DEPARTMENT
000		Corporate Headquarters
100	000	Sales and Marketing
180	100	Marketing
130	100	Field Office: East Coast
140	100	Field Office: Canada
110	100	Pacific Rim Headquarters
115	110	Field Office: Japan
116	110	Field Office: Singapore
120	100	European Headquarters
121	120	Field Office: Switzerland
123	120	Field Office: France
125	120	Field Office: Italy
600	000	Engineering
620	600	Software Products Div.
621	620	Software Development
622	620	Quality Assurance
623	620	Customer Support
670	600	Consumer Electronics Div.
671	670	Research and Development
672	670	Customer Services
900	000	Finance



SIMPLE : LEVEL OF TREE



WITH RECURSIVE

```
DEPT_TREE AS (  
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, 0 AS LVL  
    FROM DEPARTMENT D  
   WHERE D.HEAD_DEPT IS NULL  
  
  UNION ALL  
  
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, H.LVL + 1  
    FROM DEPT_TREE H JOIN DEPARTMENT D  
     ON D.HEAD_DEPT = H.DEPT_NO  
)
```

```
SELECT * FROM DEPT_TREE
```

DEPT_NO	HEAD_DEPT	DEPARTMENT	LVL
000		Corporate Headquarters	0
100	000	Sales and Marketing	1
180	100	Marketing	2
130	100	Field Office: East Coast	2
140	100	Field Office: Canada	2
110	100	Pacific Rim Headquarters	2
115	110	Field Office: Japan	3
116	110	Field Office: Singapore	3
120	100	European Headquarters	2
121	120	Field Office: Switzerland	3
123	120	Field Office: France	3
125	120	Field Office: Italy	3
600	000	Engineering	1
620	600	Software Products Div.	2
621	620	Software Development	3
622	620	Quality Assurance	3
623	620	Customer Support	3
670	600	Consumer Electronics Div.	2
671	670	Research and Development	3
672	670	Customer Services	3
900	000	Finance	1



SIMPLE : STOP AT SOME LEVEL



WITH RECURSIVE

```
DEPT_TREE AS (  
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, 0 AS LVL  
  FROM DEPARTMENT D  
  WHERE D.HEAD_DEPT IS NULL
```

UNION ALL

```
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, H.LVL + 1  
  FROM DEPT_TREE H JOIN DEPARTMENT D  
  ON D.HEAD_DEPT = H.DEPT_NO  
  WHERE H.LVL + 1 <= 1
```

)

```
SELECT * FROM DEPT_TREE
```

Result rows : 4
Reads : 11

DEPT_NO	HEAD_DEPT	DEPARTMENT	LVL
000		Corporate Headquarters	0
100	000	Sales and Marketing	1
600	000	Engineering	1
900	000	Finance	1

WITH RECURSIVE

```
DEPT_TREE AS (  
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, 0 AS LVL  
  FROM DEPARTMENT D  
  WHERE D.HEAD_DEPT IS NULL
```

UNION ALL

```
  SELECT D.DEPT_NO, D.HEAD_DEPT,  
         D.DEPARTMENT, H.LVL + 1  
  FROM DEPT_TREE H JOIN DEPARTMENT D  
  ON D.HEAD_DEPT = H.DEPT_NO
```

)

```
SELECT * FROM DEPT_TREE  
WHERE LVL <= 1
```

Result rows : 4
Reads : 21

Outer booleans **not pushed** inside recursive query !



SIMPLE : PATH FROM ROOT



WITH RECURSIVE

```
DEPT_TREE AS (  
  SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
         CAST('.' AS VARCHAR(255)) AS PATH1,  
         CAST('.' || D.DEPT_NO || '.' AS VARCHAR(255)) AS PATH2,  
         CAST('.' || D.DEPT_NO || '.' AS VARCHAR(255)) AS PATH3  
  FROM DEPARTMENT D  
  WHERE D.HEAD_DEPT IS NULL
```

UNION ALL

```
  SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
         H.PATH1 || D.HEAD_DEPT || '.',  
         H.PATH2 || D.DEPT_NO || '.',  
         '.' || D.DEPT_NO || H.PATH3  
  FROM DEPT_TREE H JOIN DEPARTMENT D  
    ON D.HEAD_DEPT = H.DEPT_NO  
)
```

SELECT * FROM DEPT_TREE

PATH1 - path from the root down to current node

PATH2 - path from the root down to current node including current node

PATH3 - path from the current node up to the root



SIMPLE : PATH FROM ROOT



DEPT_NO	HEAD_DEPT	DEPARTMENT	PATH1	PATH2	PATH3
000		Corporate Headquarters	.	.000.	.000.
100	000	Sales and Marketing	.000.	.000.100.	.100.000.
180	100	Marketing	.000.100.	.000.100.180.	.180.100.000.
130	100	Field Office: East Coast	.000.100.	.000.100.130.	.130.100.000.
140	100	Field Office: Canada	.000.100.	.000.100.140.	.140.100.000.
110	100	Pacific Rim Headquarters	.000.100.	.000.100.110.	.110.100.000.
115	110	Field Office: Japan	.000.100.110.	.000.100.110.115.	.115.110.100.000.
116	110	Field Office: Singapore	.000.100.110.	.000.100.110.116.	.116.110.100.000.
120	100	European Headquarters	.000.100.	.000.100.120.	.120.100.000.
121	120	Field Office: Switzerland	.000.100.120.	.000.100.120.121.	.121.120.100.000.
123	120	Field Office: France	.000.100.120.	.000.100.120.123.	.123.120.100.000.
125	120	Field Office: Italy	.000.100.120.	.000.100.120.125.	.125.120.100.000.
600	000	Engineering	.000.	.000.600.	.600.000.
620	600	Software Products Div.	.000.600.	.000.600.620.	.620.600.000.
621	620	Software Development	.000.600.620.	.000.600.620.621.	.621.620.600.000.
622	620	Quality Assurance	.000.600.620.	.000.600.620.622.	.622.620.600.000.
623	620	Customer Support	.000.600.620.	.000.600.620.623.	.623.620.600.000.
670	600	Consumer Electronics Div.	.000.600.	.000.600.670.	.670.600.000.
671	670	Research and Development	.000.600.670.	.000.600.670.671.	.671.670.600.000.
672	670	Customer Services	.000.600.670.	.000.600.670.672.	.672.670.600.000.
900	000	Finance	.000.	.000.900.	.900.000.

PATH1 - path from the root down to current node

PATH2 - path from the root down to current node including current node

PATH3 - path from the current node up to the root



SIMPLE : PATH FROM ROOT



WITH RECURSIVE

```
DEPT_TREE AS (  
    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
           CAST(' ' AS VARCHAR(255)) AS HEAD  
    FROM DEPARTMENT D  
    WHERE D.HEAD_DEPT IS NULL  
  
    UNION ALL  
  
    SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
           H.HEAD || '\ ' || H.DEPARTMENT  
    FROM DEPT_TREE H JOIN DEPARTMENT D  
    ON D.HEAD_DEPT = H.DEPT_NO  
)  
SELECT * FROM DEPT_TREE
```



SIMPLE : PATH FROM ROOT



DEPT_NO	HEAD_DEPT	DEPARTMENT	HEAD
000		Corporate Headquarters	
100	000	Sales and Marketing	\Corporate Headquarters
180	100	Marketing	\Corporate Headquarters\Sales and Marketing
130	100	Field Office: East Coast	\Corporate Headquarters\Sales and Marketing
140	100	Field Office: Canada	\Corporate Headquarters\Sales and Marketing
110	100	Pacific Rim Headquarters	\Corporate Headquarters\Sales and Marketing
115	110	Field Office: Japan	\Corporate Headquarters\Sales and Marketing\Pacific Rim Headquarters
116	110	Field Office: Singapore	\Corporate Headquarters\Sales and Marketing\Pacific Rim Headquarters
120	100	European Headquarters	\Corporate Headquarters\Sales and Marketing
121	120	Field Office: Switzerland	\Corporate Headquarters\Sales and Marketing\European Headquarters
123	120	Field Office: France	\Corporate Headquarters\Sales and Marketing\European Headquarters
125	120	Field Office: Italy	\Corporate Headquarters\Sales and Marketing\European Headquarters
600	000	Engineering	\Corporate Headquarters
620	600	Software Products Div.	\Corporate Headquarters\Engineering
621	620	Software Development	\Corporate Headquarters\Engineering\Software Products Div.
622	620	Quality Assurance	\Corporate Headquarters\Engineering\Software Products Div.
623	620	Customer Support	\Corporate Headquarters\Engineering\Software Products Div.
670	600	Consumer Electronics Div.	\Corporate Headquarters\Engineering
671	670	Research and Development	\Corporate Headquarters\Engineering\Consumer Electronics Div.
672	670	Customer Services	\Corporate Headquarters\Engineering\Consumer Electronics Div.
900	000	Finance	\Corporate Headquarters



SIMPLE : TREE-STYLE REPORT



WITH RECURSIVE

```
DEPT_TREE AS (  
  SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT,  
         CAST(' ' AS VARCHAR(255)) AS INDENT  
  FROM DEPARTMENT D  
  WHERE D.HEAD_DEPT IS NULL
```

UNION ALL

```
  SELECT D.DEPT_NO, D.HEAD_DEPT, D.DEPARTMENT, H.INDENT || ' ' AS INDENT  
  FROM DEPT_TREE H JOIN DEPARTMENT D  
    ON D.HEAD_DEPT = H.DEPT_NO  
)
```

```
SELECT T.INDENT || T.DEPARTMENT AS DEPARTMENT, BUDGET, LOCATION, PHONE_NO  
FROM DEPT_TREE T JOIN DEPARTMENT D  
  ON T.DEPT_NO = D.DEPT_NO
```



SIMPLE : TREE-STYLE REPORT



DEPARTMENT	BUDGET	LOCATION	PHONE_NO
Corporate Headquarters	1 000 000,00	Monterey	(408) 555-1234
Sales and Marketing	2 000 000,00	San Francisco	(415) 555-1234
Marketing	1 500 000,00	San Francisco	(415) 555-1234
Field Office: East Coast	500 000,00	Boston	(617) 555-1234
Field Office: Canada	500 000,00	Toronto	(416) 677-1000
Pacific Rim Headquarters	600 000,00	Kuauai	(808) 555-1234
Field Office: Japan	500 000,00	Tokyo	3 5350 0901
Field Office: Singapore	300 000,00	Singapore	3 55 1234
European Headquarters	700 000,00	London	71 235-4400
Field Office: Switzerland	500 000,00	Zurich	1 211 7767
Field Office: France	400 000,00	Cannes	58 68 11 12
Field Office: Italy	400 000,00	Milan	2 430 39 39
Engineering	1 100 000,00	Monterey	(408) 555-1234
Software Products Div.	1 200 000,00	Monterey	(408) 555-1234
Software Development	400 000,00	Monterey	(408) 555-1234
Quality Assurance	300 000,00	Monterey	(408) 555-1234
Customer Support	650 000,00	Monterey	(408) 555-1234
Consumer Electronics Div.	1 150 000,00	Burlington, VT	(802) 555-1234
Research and Development	460 000,00	Burlington, VT	(802) 555-1234
Customer Services	850 000,00	Burlington, VT	(802) 555-1234
Finance	400 000,00	Monterey	(408) 555-1234



NOT USUAL : DATA GENERATOR



Integer numbers :

```
WITH RECURSIVE
  NUMBERS (NUM) AS (
    SELECT 1 FROM RDB$DATABASE

    UNION ALL

    SELECT NUM + 1 FROM NUMBERS
      WHERE NUM < 100
  )
SELECT NUM FROM NUMBERS
```

Random numbers :

```
WITH RECURSIVE
  NUMBERS (SEQ, NUM) AS (
    SELECT 1, CAST(RAND() * 1000 AS INTEGER)
      FROM RDB$DATABASE

    UNION ALL

    SELECT SEQ + 1, CAST(RAND() * 1000 AS INTEGER)
      FROM NUMBERS WHERE SEQ < 100
  )
SELECT * FROM NUMBERS
```

Consecutive dates :

```
WITH RECURSIVE
  DATES (DT) AS (
    SELECT CURRENT_DATE
      FROM RDB$DATABASE

    UNION ALL

    SELECT DT + 1 FROM DATES
      WHERE DT < CURRENT_DATE + 31
  )
SELECT * FROM DATES
```

Weekend's in next year :

```
WITH RECURSIVE
  WEEKENDS (WK) AS (
    SELECT CURRENT_DATE + 7 -
      EXTRACT(WEEKDAY FROM CURRENT_DATE)
      FROM RDB$DATABASE

    UNION ALL

    SELECT WK + 7 FROM WEEKENDS
      WHERE WK < CURRENT_DATE + 365
  )
SELECT * FROM WEEKENDS
```



ADVANCED : FIBONACCI



$$n_1 = n_2 = 1, \quad n_i = n_{i-1} + n_{i-2}, \quad i \geq 2$$

1, 1, 2, 3, 5, 8, ...

```
WITH RECURSIVE
  FIBONACCI (N1, N2) AS (
    SELECT 0, 1 FROM RDB$DATABASE

    UNION ALL

    SELECT N2, N1 + N2 FROM FIBONACCI
      WHERE N1 + N2 < 1000
  )

SELECT N2 FROM FIBONACCI
```

N2
1
1
2
3
5
8
13
21
34
55
89
144
233
377
610
987



GENERALIZED CASE OF FIBONACCI



$$n_1 = n_2 = \dots n_m = 1$$

$$n_i = n_{i-1} + \dots + n_{i-m}, i > m$$

WITH RECURSIVE

```
FIBONACCI (N1, N2, ..., Nm) AS (
  SELECT 0, 0, ..., 1
  FROM RDB$DATABASE
```

```
  UNION ALL
```

```
  SELECT N2, ..., Nm,
         CASE WHEN N1 = 0
              THEN 1
              ELSE F(N1, N2, ..., Nm)
```

```
         END
```

```
  FROM FIBONACCI
```

```
  WHERE ...
```

```
)
```

```
SELECT Nm FROM FIBONACCI
```

Examples :

a) classical fibonacci numbers :

$$M = 2, F(n1, n2) = n1 + n2$$

b) M = 4,

$$F(n1, n2, n3, n4) = n1 + n2 + n3 + n4$$

N4
1
1
1
4
7
13
25
49
94
181
349
673
1297
2500
4819
9289



ADVANCED : COMBINATIONS



Combinations :

WITH RECURSIVE

T (CH) AS

(

SELECT 'a' FROM RDB\$DATABASE UNION ALL

SELECT 'b' FROM RDB\$DATABASE UNION ALL

SELECT 'c' FROM RDB\$DATABASE UNION ALL

SELECT 'd' FROM RDB\$DATABASE

),

COMBINATIONS AS

(

SELECT CAST(CH AS VARCHAR(4)) COMB FROM T

UNION ALL

SELECT C.COMB || T.CH

FROM COMBINATIONS C JOIN T

ON T.CH > SUBSTRING(C.COMB FROM CHAR_LENGTH(C.COMB) FOR 1)

WHERE CHAR_LENGTH(C.COMB) < 4

)

SELECT * FROM COMBINATIONS

ORDER BY CHAR_LENGTH(COMB), COMB

COMB
a
b
c
d
ab
ac
ad
bc
bd
cd
abc
abd
acd
bcd
abcd



ADVANCED : PERMUTATIONS



Permutations :

WITH RECURSIVE

T (CH) AS

(

SELECT 'a' FROM RDB\$DATABASE UNION ALL

SELECT 'b' FROM RDB\$DATABASE UNION ALL

SELECT 'c' FROM RDB\$DATABASE

),

PERMUTATIONS AS

(

SELECT CAST(CH AS VARCHAR(3)) PERM FROM T

UNION ALL

SELECT P.PERM || T.CH

FROM PERMUTATIONS P JOIN T

ON (1 = 1)

WHERE CHAR_LENGTH(P.PERM) < 3

)

SELECT * FROM PERMUTATIONS

WHERE CHAR_LENGTH(P.PERM) = 3

ORDER BY CHAR_LENGTH(PERM), PERM

PERM
aaa
aab
aac
aba
abb
abc
aca
acb
acc
baa
bab
bac
bba
bbb
bbc
bca
bcb
bcc
caa
cab
cac
cba
cbb
cbc
cca
ccb
ccc



OBRIGADO PELA VOSSA ATENÇÃO !



Questions ?

[Firebird official web site](#)

[Firebird tracker](#)

hvlad@users.sf.net

