Some Iterations Over Recursion

Michael Tiefenbacher

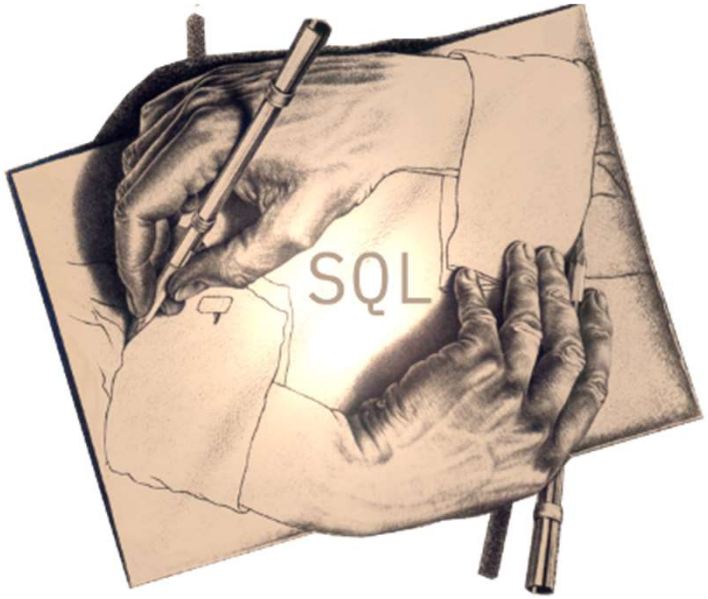
Agenda

- Introduction and Motivation
- Common Table Expressions
- Recursion
  - Simple Example
  - Structure
  - Further Examples
  - Debugging
  - Alternative Syntax
- Appendix

Introduction & Motivation

- Recursion is a problem-solving strategy
- A function, which references itself in its definition

To loop is human,
to recurse is divine



Source: Henrik Loeser Blog
https://blog.4loeser.net/2018/04/db2-cte-and-connect-by-two-kinds-of.html

Kudos for Henrik who created the presentation with me

Use Cases

- Test data generation
- Step through hierarchies of undefined depth
  - Organizational hierarchies
  - Referential Integrity hierarchy
  - View hierarchy
- Route calculation
- Bill of materials
- Mathematical calculations
  - Faculty
- **Solving a Sudoku**

Whenever the number of joins is not defined or not previously known recursion is the solution

Common Table Expression (CTE)

- Common Tables Expression is part of the SQL standard (ANSI SQL99)
- Generates a temporary table within a SQL statement
- Can be used to
  - Improve readability of statements
  - Reuse of parts of a statement
  - Implement recursion
- Multiple CTEs can be defined within a single SQL statement when separated with a comma (“,”)
  - WITH keyword is only necessary at the beginning

```
WITH <cte-table name> [(<column-list>)] AS
( <full-select> )
```

Common Table Expression (CTE) – Sample

Optional, if all columns are named

```
with temp (date) as (
select date('29.06.2018') as date
  from sysibm.sysdummy1
)
select * from temp
```

Multiple CTEs can be used – separated by comma

```
with t1 (date) as (
select …
)
, t2 as (
select …
)
select * from t1, t2
```

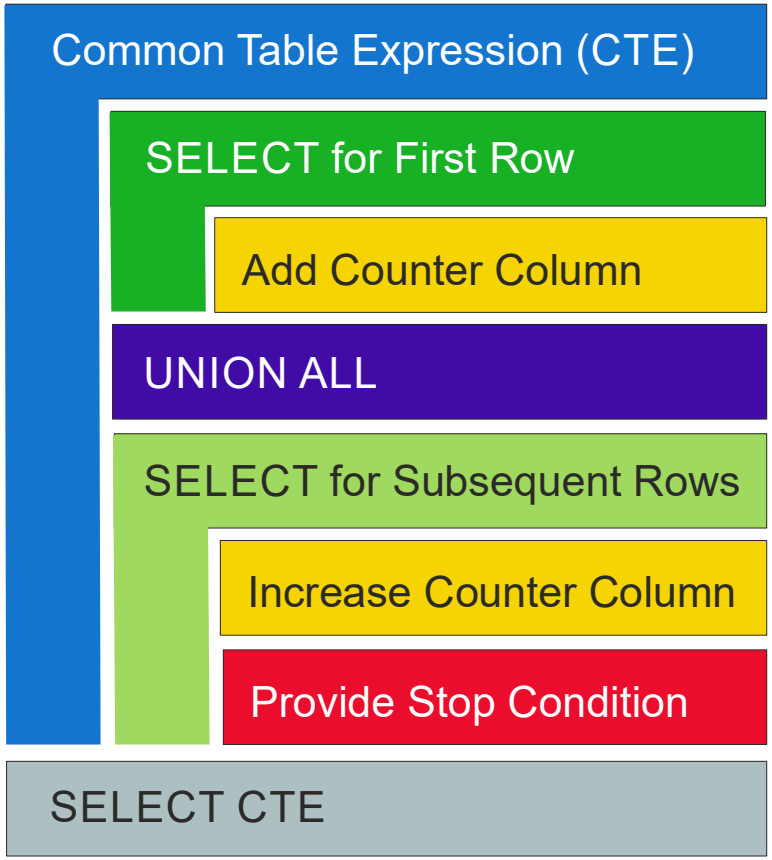
Recursion – Simple Example

- Generation of Time Series (Dates)

```
with temp (date) as (
select date('29.06.2018') as date
  from sysibm.sysdummy1
union all
select date + 1 day
  from temp
 where date < date('19.10.2018')
)
select * from temp
```

DATE
2018-06-29
2018-06-30
2018-07-01
2018-07-02
2018-07-03
2018-07-04
2018-07-05
2018-07-06
2018-07-07
...
2018-10-15
2018-10-16
2018-10-17
2018-10-18
2018-10-19

Structure of a Recursive Query




```
WITH cte_recursion (…) AS (
SELECT …
     , 1 as counter
  FROM …
UNION ALL
SELECT …
  FROM cte_recursion,…
 WHERE …
   AND counter < 5)
select * from cte_recursion
```

Introduce counter

Heart of recursion

Use counter to limit loops

Potential Problems

- DISTINCT is not allowed
  - SQL0342N The common table expression "<name>" **cannot use SELECT DISTINCT** and **must use UNION ALL** because it is recursive.
- Columns need to be named in CTE (only for recursive queries)
  - SQL0343N The column names are required for the recursive common table expression "<name>".
- Joins in recursive CTE need the old syntax and further restrictions
  - SQL0345N The fullselect of the recursive common table expression "<name>" must be the UNION of two or more fullselects and **cannot include column functions, GROUP BY clause, HAVING clause, ORDER BY clause, or an explicit join including an ON clause.**

Limitation does not exist in Db2 for z/OS

Recursion – Hierarchies

- Hierarchies of Departments
  - From Db2 SAMPLE database

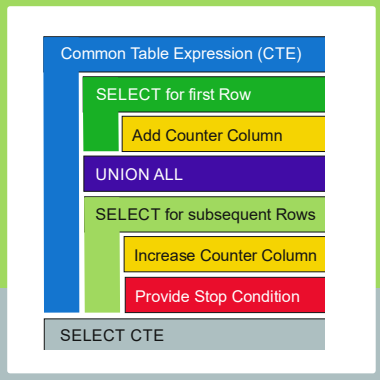
DEPTNO [CHAR(3 OCTETS)]	DEPTNAME [VARCHAR(36 OCTETS)]	MGRNO [CHAR(6 OCTETS)]	ADMRDEPT [CHAR(3 OCTETS)]
A00 ...	SPIFFY COMPUTER SERVICE DIV.	000010	A00
B01	PLANNING	000020	A00
C01	INFORMATION CENTER	000030	A00
D01	DEVELOPMENT CENTER		A00
D11	MANUFACTURING SYSTEMS	000060	D01
D21	ADMINISTRATION SYSTEMS	000070	D01
E01	SUPPORT SERVICES	000050	A00
E11	OPERATIONS	000090	E01
E21	SOFTWARE SUPPORT	000100	E01
F22	BRANCH OFFICE F2		E01
G22	BRANCH OFFICE G2		E01
H22	BRANCH OFFICE H2		E01
I22	BRANCH OFFICE I2		E01
J22	BRANCH OFFICE J2		E01

Hierarchies Example

```
WITH temp(DEPTNO, DEPTNAME, ADMRDEPT, LEVEL) as (
SELECT DEPTNO, DEPTNAME, ADMRDEPT, 0 as LEVEL
  FROM DEPARTMENT
 WHERE DEPTNO = 'D21'
UNION ALL
SELECT d.DEPTNO, d.DEPTNAME, d.ADMRDEPT, t.LEVEL + 1 as LEVEL
  FROM temp t,
       DEPARTMENT d
 WHERE t.ADMRDEPT = d.DEPTNO
   AND t.LEVEL < 5
   AND t.DEPTNO <> t.ADMRDEPT
)
SELECT * FROM temp
```

Structure – Color Coding

```
WITH temp(DEPTNO, DEPTNAME, ADMRDEPT, LEVEL) as (
SELECT DEPTNO, DEPTNAME, ADMRDEPT, 0 as LEVEL
  FROM DEPARTMENT
 WHERE DEPTNO = 'D21'
UNION ALL
SELECT d.DEPTNO, d.DEPTNAME, d.ADMRDEPT, t.LEVEL + 1 as LEVEL
  FROM temp t,
       DEPARTMENT d
 WHERE t.ADMRDEPT = d.DEPTNO
   AND t.LEVEL < 5
   AND t.DEPTNO <> t.ADMRDEPT
)
SELECT * FROM temp
```



Recursion – View Hierarchies Example

```
with temp (tabname, bname, level) as (
SELECT tabname, bname, 0 as level
  FROM SYSCAT.TABDEP
 WHERE tabname = 'V_LEVEL5'
 UNION ALL
SELECT td.tabname, td.bname, t.level + 1 as level
  FROM temp t,
       SYSCAT.TABDEP td
 WHERE t.bname = td.tabname
   AND t.level < 9
)
SELECT * FROM temp
```

Top-Down

Recursion – View Hierarchies Example

```
with temp (tabname, bname, level) as (
SELECT tabname, bname, 0 as level
  FROM SYSCAT.TABDEP
 WHERE bname = 'EMPLOYEE'
 UNION ALL
SELECT td.tabname, td.bname, t.level + 1 as level
  FROM temp t,
       SYSCAT.TABDEP td
 WHERE td.bname = t.tabname
   AND t.level < 9
)
SELECT * FROM temp
```

Bottom-Up

Template

```
with cte (<columnlist>, <joincolumn>, level) as (
SELECT <columnlist>, <joincolumn>, 0 as level
  FROM <basetable>
 WHERE <start-condition>
 UNION ALL
SELECT b.<columnlist>, b.<joincolumn>
     , c.level + 1 as level
  FROM cte c,
       <basetable> b
 WHERE b.<joincolumn> = c.colname
   AND c.level < 9
)
 SELECT * FROM cte
```

Debugging Options for Recursive Queries

- How can a recursive statement be debugged?
  - First part of the UNION ALL can be executed by its own
  - Comment UNION ALL and SELECT first part of the CTE with the second one
  - Use a simple SQL to SELECT the complete CTE

Debugging Options for Recursive Queries

```
with temp (tabname, bname, level) as (

SELECT tabname, bname, 0 as level
  FROM SYSCAT.TABDEP
 WHERE bname = 'EMPLOYEE'

) -- UNION ALL
SELECT td.tabname, td.bname, t.level + 1 as level
  FROM temp t,
       SYSCAT.TABDEP td
 WHERE td.bname = t.tabname
   AND t.level < 9
)
 SELECT * FROM temp
```

Level1

Level2

Alternative Syntax

I would like to hear a recursion joke!

Oh, I know a good one!

Let me start off: I would like to hear…

Alternative Syntax for Recursion

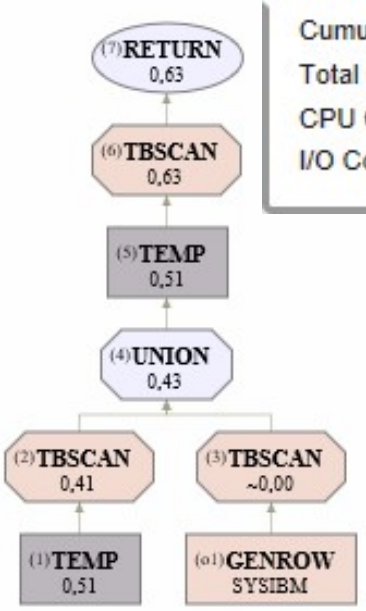
- Alternative Syntax with CONNECT BY
  - Oracle style
  - Prerequisite: DB2_COMPATIBILITY_VECTOR = 08
    - Instance restart is sufficient
- Note
  - This is not only a different syntax but also the internal processing is different
    - Depth first
  - Limited to a depth of 64 levels of recursion
    - SQL20450N Recursion limit exceeded within a hierarchical query.

Oracle Syntax Example

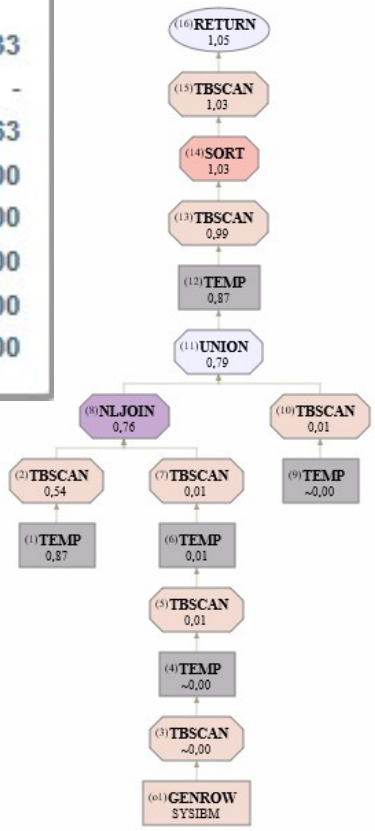
```
SELECT date('20.04.2018') + level days  as dt
  FROM sysibm.sysdummy1
CONNECT BY date('20.04.2018') + level days
              <= date('29.05.2018')
```

Comparison of the Syntax Alternatives

Operation	Return
Estimated Cardinality	334.333
Actual Cardinality	-
Cumulative Total Cost(timeron)	0.63
Cumulative CPU Cost(timeron)	4,334,759.00
Cumulative I/O Cost(timeron)	0.00
Total Cost(timeron)	0.00
CPU Cost(timeron)	0.00
I/O Cost(timeron)	0.00



Db2 Syntax

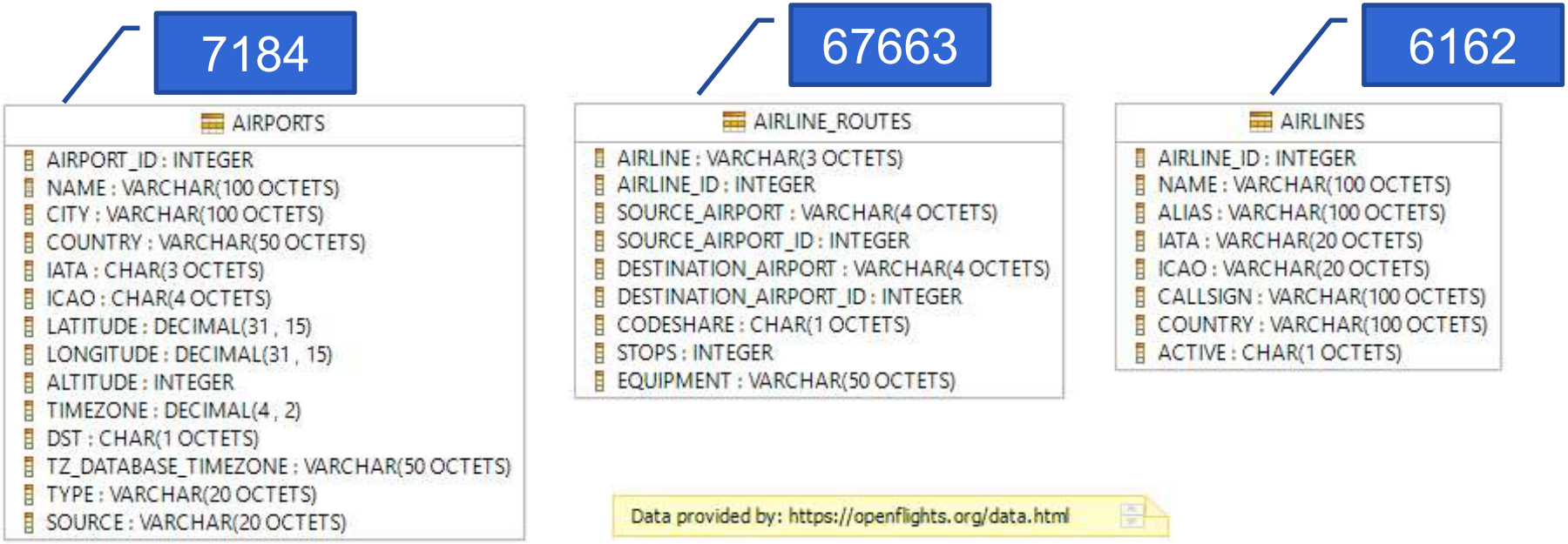


Operation	Return
Estimated Cardinality	334.333
Actual Cardinality	-
Cumulative Total Cost(timeron)	1.05
Cumulative CPU Cost(timeron)	7,225,548.00
Cumulative I/O Cost(timeron)	0.00
Total Cost(timeron)	0.02
CPU Cost(timeron)	149,223.50
I/O Cost(timeron)	0.00

Oracle Syntax

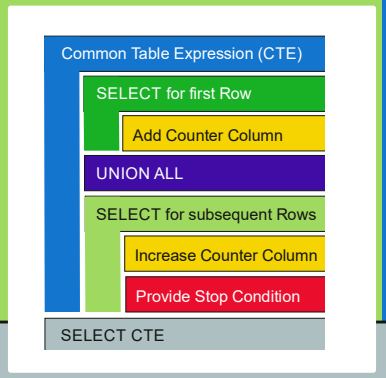
Recursion – Example Flight Route

- Find a route from A to B with a certain number of hops
  - i.e. Frankfurt (FRA) to Philadelphia (PHL)



Flight Route Solution

```
with temp (source_airport, destination_airport, airline_id, hops, route) as (
SELECT source_airport, destination_airport, airline_id, 0 as hops,
cast(source_airport || '->' || destination_airport as varchar(500)) as route
  FROM airline_routes ar
 WHERE source_airport = 'FRA'
   AND exists (SELECT 1 FROM airline_routes WHERE destination_airport = 'PHL'
                                              AND airline_id = ar.airline_id)
 UNION ALL
SELECT ar.source_airport, ar.destination_airport, ar.airline_id, t.hops + 1 as hops
     , cast(route || '->' || ar.destination_airport as varchar(500)) as route
  FROM temp t, airline_routes ar
 WHERE ar.source_airport = t.destination_airport
   AND ar.airline_id = t.airline_id
   AND LOCATE_IN_STRING(t.route, ar.destination_airport) = 0
   AND ar.source_airport <> 'PHL'
   AND t.hops < 3          )
SELECT * FROM temp WHERE destination_airport = 'PHL'
```



Flight Route – Details

- Concatenation of the airports on the route

```
cast(source_airport || '->' || destination_airport as varchar(500)) as route
```

- Only airlines that approach the destination airport

```
and exists (SELECT 1 FROM airline_routes WHERE destination_airport = 'PHL'
                                               AND airline_id = ar.airline_id)
```

- The next destination must not be part of the current route (avoid loops)

```
and LOCATE_IN_STRING(t.route, ar.destination_airport) = 0
```

- Stop if PHL is the origin or more than three hops are involved

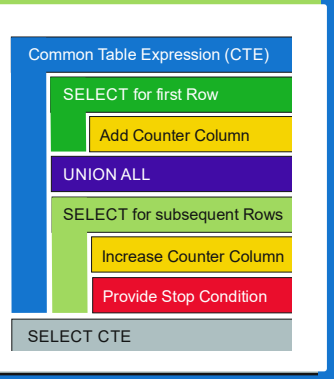
```
and ar.source_airport <> 'PHL' and t.hops < 3
```

Flight Route – Result

SOURCE_AIRPORT	DESTINATION_AIRPORT	AIRLINE_ID	HOPS	ROUTE
LHR	PHL	1355	3	FRA->ATL->MIA->LHR->PHL
LHR	PHL	1355	3	FRA->ATL->ORD->LHR->PHL
LHR	PHL	1355	3	FRA->DFW->ATL->LHR->PHL
LHR	PHL	1355	3	FRA->DFW->CDG->LHR->PHL
LHR	PHL	1355	3	FRA->DFW->MAD->LHR->PHL
LHR	PHL	1355	3	FRA->DFW->MEX->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->ABZ->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->AGP->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->AMS->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->ARN->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->DUS->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->EDI->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->GLA->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->IBZ->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->MAD->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->NCE->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->PMI->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->RTM->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->VCE->LHR->PHL
LHR	PHL	1355	3	FRA->LCY->ZRH->LHR->PHL
DTW	PHL	2009	3	FRA->ATL->ABE->DTW->PHL
MSP	PHL	2009	3	FRA->ATL->ABQ->MSP->PHL
SLC	PHL	2009	3	FRA->ATL->ABQ->SLC->PHL
DTW	PHL	2009	3	FRA->ATL->ALB->DTW->PHL

Nested XML test data (1|2)

```
with mytable(mydoc,depth,fanout,level) as
(SELECT xmlelement(name "first",'Hello recursive world'),5,10
      , 1 as level
 FROM sysibm.sysdummy1
UNION ALL
SELECT xmlelement(name "in-between",
          xmlquery('<a>{for $i in (1 to $FANOUT) return
<b>{$MYDOC}</b>}</a>')),
       depth,fanout, level+1 as level
FROM mytable WHERE level<depth)
SELECT xmlelement(name "root",mydoc) as doc
  FROM mytable
 WHERE level = depth
```

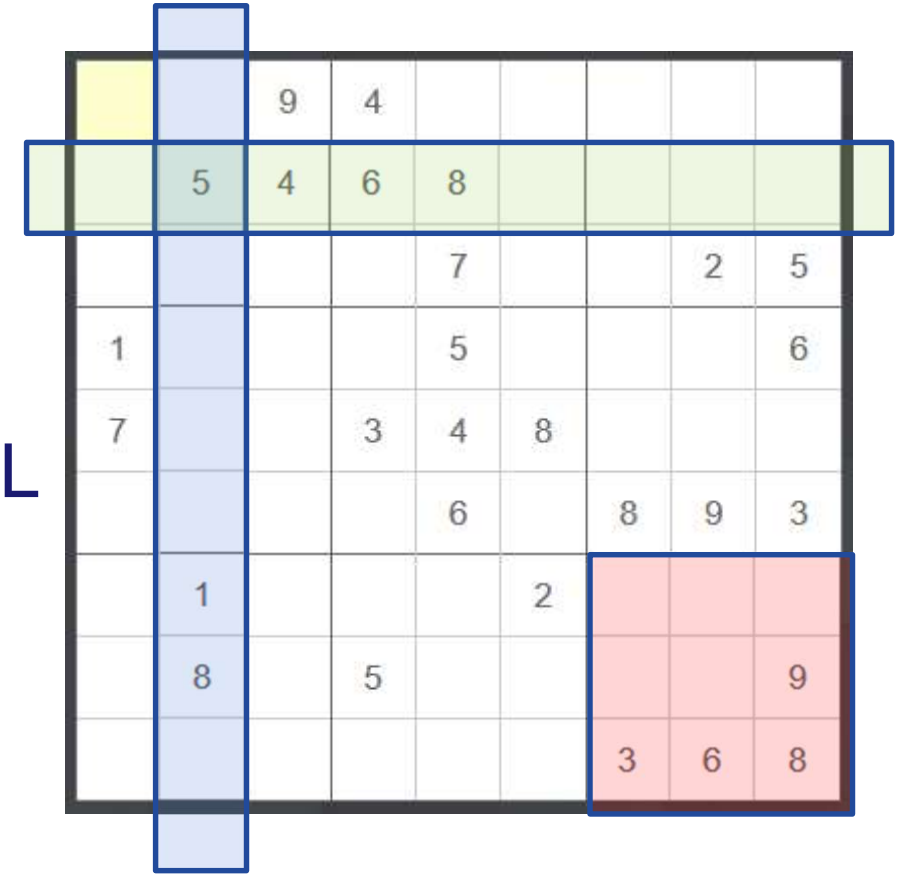


Nested XML test data (2|2)

```
<?xml version="1.0" encoding="UTF-16" ?><root><in-between><a><b><in-
between><a><b><in-between><a><b><in-between><a><b><first>Hello
recursive world</first></b><b><first>Hello recursive
world</first></b><b><first>Hello recursive
world</first></b><b><first>Hello recursive world</first></b> … and
so on
```

Solving Sudokus

- Sudoku
  - Values 1 to 9 only once per row and per column
  - Values 1 to 9 ony once per square
- Input needs to be provided to SQL
  - As a unformatted string
  - “.“ for empty fields



Sudoku – Solution Part I

- The input string is provided in a CTE

```
WITH input(sud) AS (
values ('..94......5468........7..251...5...67..348.....
..6.893.1...2....8.5....9......368')
```

Sudoku – Solution Part II

- Build a list with possible values (1-9)
  - Recursive query
  - Value as number and as string (as the input is a string)

```
with digits(z, lp) AS (
    SELECT '1' as z, 1 as lp FROM sysibm.sysdummy1
    UNION ALL
    SELECT CAST(lp + 1 AS varchar(1)), lp + 1
     FROM digits WHERE lp < 9
)
select * from digits
```

Z	LP
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

- Alternative

```
VALUES ('1',1),('2',2),('3',3),('4',4),('5',5)
      ,('6',6),('7',7),('8',8),('9',9)
```

Sudoku – Solution Part III

```
,   x(s, ind) AS (
     SELECT sud, instr(sud, '.') FROM input
     UNION ALL
     SELECT
       substr(s, 1, ind-1) || z || substr(s, ind+1),
       instr( substr(s, 1, ind-1) || z || substr(s, ind+1), '.' )
      FROM x, digits AS z
     WHERE ind>0
       AND NOT EXISTS (
             SELECT 1
               FROM digits AS lp
              WHERE z.z = substr(s, ((ind-1)/9)*9 + lp, 1)
                 OR z.z = substr(s, ((ind-1)%9) + (lp-1)*9 + 1, 1)
                 OR z.z = substr(s, (((ind-1)/3) % 3) * 3
                         + ((ind-1)/27) * 27 + lp + ((lp-1) / 3) * 6, 1)
         )
  )
SELECT s FROM x WHERE ind=0;
```

Next empty field

Maths for same row

Maths for same column

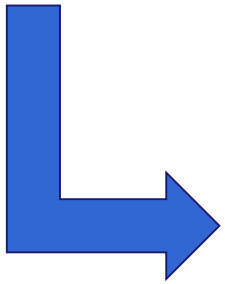
Clever maths for same square

Sudoku – Result

- Result is also returned as a string

```
s
379425681254681937861973425138259746796348152542167893613892574487536219925714368
```

reformatted



379	425	681
254	681	937
861	973	425
138	259	746
796	348	152
542	167	893
613	892	574
487	536	219
925	714	368

Further Information I

- Queries and other materials:
  - https://github.com/data-henrik/sql-recursion
- Blog by Henrik Loeser
  - https://blog.4loeser.net/2018/04/db2-cte-and-connect-by-two-kinds-of.html

- IDUG Session of previous conferences
  - Utilizing DB2 V8 Recursive SQL on all Plattforms
    - Daniel Luksetich – IDUG EMEA 2005 – F13
  - Parlez-Vous Klingon? Recursion SQL for Generating Test Data
    - Alexander Kopac – IDUG 2018 – F15

Further Information II

- Hierarchical Queries with DB2 Connect By (iSeries)
  https://developer.ibm.com/articles/i-db2connectby/
- Migrating Recursive SQL from Oracle to DB2 UDB
  https://www.ibm.com/developerworks/data/library/techarticle/0307steinbach/0307steinbach.html
- SQL Cookbook by Graeme Birchall
  http://db2-sql-cookbook.org/
- KnowledgeCenter
  https://www.ibm.com/support/knowledgecenter/SSEPGG_11.5.0/com.ibm.db2.luw.apdv.porting.doc/doc/r0052877.html

Michael Tiefenbacher
ids-System GmbH

m.tiefenbacher@ids-system.de
Twitter: @globomike