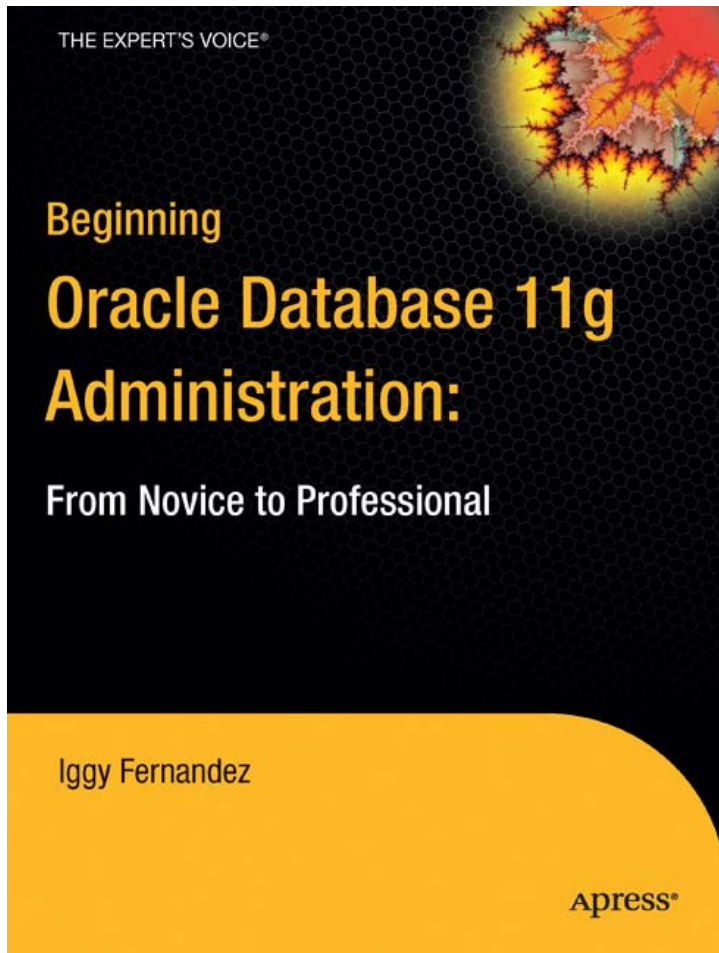


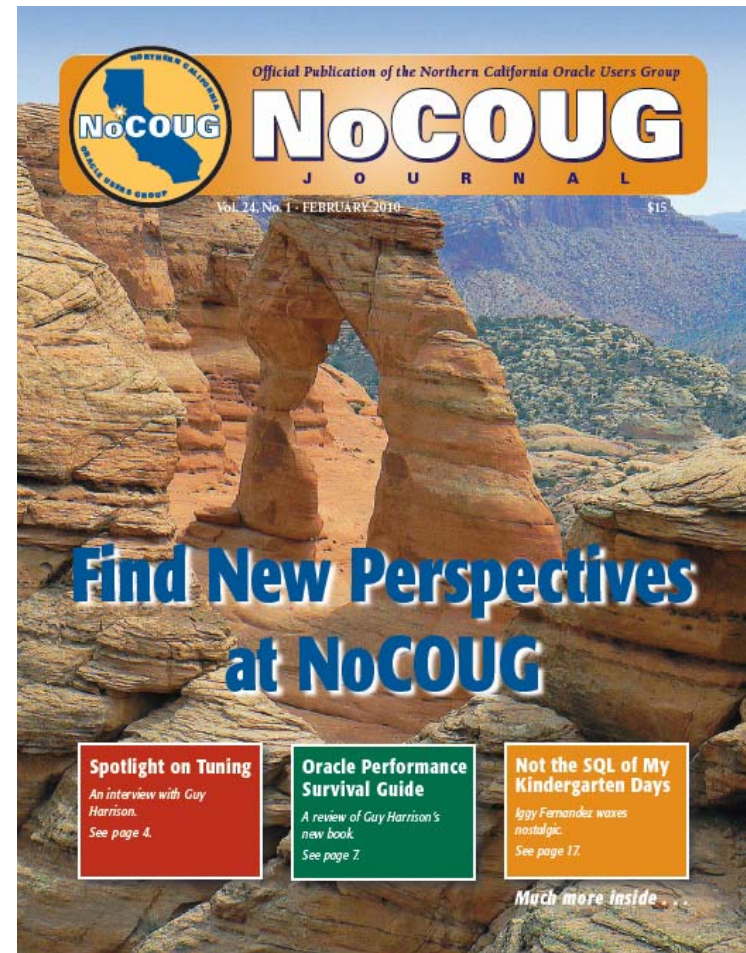
Recursive Common Table Expressions in Oracle 11gR2

Iggy Fernandez

RMOUG Training Days 2010



Iggy Fernandez



Database Specialists

CTE Recap

Inline Views

```
SELECT *
FROM (SELECT *
      FROM Suppliers
      MINUS
      SELECT *
      FROM (SELECT SupplierName
            FROM (SELECT *
                  FROM (SELECT *
                        FROM Suppliers,
                        Parts)
                  MINUS
                  SELECT *
                  FROM (SELECT SupplierName,
                        PartName
                        FROM Quotes)))));
```

CTE Recap

All Supplier Part Pairs

WITH

AllSupplierPartPairs AS

```
(  
  SELECT *  
  FROM Suppliers, Parts  
)
```

CTE Recap

Valid Supplier Part Pairs

ValidSupplierPartPairs AS

```
(  
  SELECT SupplierName, PartName  
  FROM Quotes  
)
```

CTE Recap

Invalid Supplier Part Pairs

```
InvalidSupplierPartPairs AS  
(  
  SELECT *  
  FROM AllSupplierPartPairs  
  MINUS  
  SELECT *  
  FROM ValidSupplierPartPairs  
),
```

CTE Recap

Suppliers Who Don't Supply All Parts

SuppliersWhoDontSupplyAllParts AS

```
(  
  SELECT SupplierName  
  FROM InvalidSupplierPartPairs  
),
```

CTE Recap

Suppliers Who Supply All Parts

SuppliersWhoSupplyAllParts AS

```
(  
  SELECT *  
  FROM Suppliers  
  MINUS  
  SELECT *  
  FROM SuppliersWhoDontSupplyAllParts  
)
```

CTE Recap

Suppliers Who Supply All Parts

```
SELECT *  
FROM SuppliersWhoSupplyAllParts;
```

Traditional Hierarchical Queries

Managers and Employees

SELECT

```
    LPAD (' ', 4 * (LEVEL - 1)) || first_name || ' ' ||  
last_name AS name
```

FROM employees

START WITH manager_id IS NULL

CONNECT BY manager_id = PRIOR employee_id;

Traditional Hierarchical Queries

Managers and Employees

Name

Steven King

 Neena Kochhar

 Nancy Greenberg

 Daniel Faviet

 John Chen

 Ismael Sciarra

 Jose Manuel Urman

 Luis Popp

 Jennifer Whalen

 Susan Mavris

 Hermann Baer

 Shelley Higgins

 William Gietz

Traditional Hierarchical Queries

Managers and Employees

WITH

```
RCTE (employee_id, first_name, last_name, lvl) AS  
(
```

SELECT

```
    employee_id,  
    first_name,  
    last_name,  
    1 AS lvl
```

FROM

```
    employees
```

WHERE manager_id IS NULL

Traditional Hierarchical Queries

Managers and Employees

UNION ALL

SELECT

```
e.employee_id,  
e.first_name,  
e.last_name,  
lvl + 1 AS lvl
```

FROM

```
RCTE INNER JOIN employees e  
ON (RCTE.employee_id = e.manager_id)  
)  
-- SEARCH DEPTH FIRST BY employee_id ASC SET seq#
```

Traditional Hierarchical Queries

Managers and Employees

```
SELECT LPAD (' ', 4 * (lvl - 1)) || first_name || ' ' ||  
last_name AS name  
FROM RCTE  
--ORDER BY seq#;
```

Traditional Hierarchical Queries

Breadth First Search

Steven King

Michael Hartstein

Neena Kochhar

Lex De Haan

Den Raphaely

Matthew Weiss

Adam Fripp

Payam Kaufling

Shanta Vollman

Kevin Mourgous

John Russell

Karen Partners

Alberto Errazuriz

Gerald Cambrault

Eleni Zlotkey

Pat Fay

Jennifer Whalen

Algorithm

1. Split the CTE expression into anchor and recursive members.
2. Run the anchor member(s) creating the first invocation or base result set (T_0).
3. Run the recursive member(s) with T_i as an input and T_{i+1} as an output.
4. Repeat step 3 until an empty set is returned.
5. Return the result set. This is a UNION ALL of T_0 to T_n .

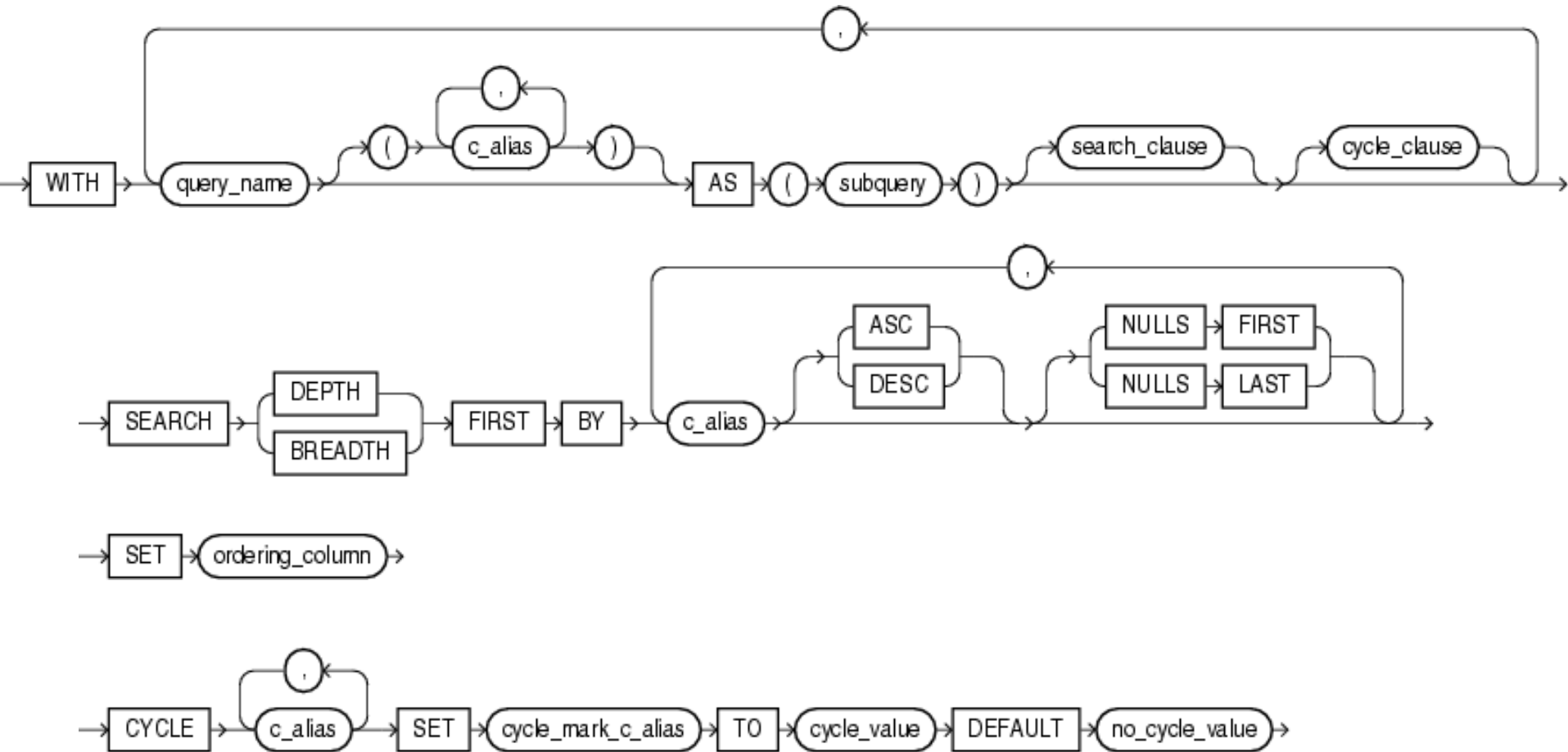
Number Generator New Style

```
WITH numbers(n) AS  
(  
  SELECT 1 FROM dual  
  UNION ALL  
  SELECT n + 1 FROM numbers WHERE n < 100  
)  
SELECT * FROM numbers;
```

Number Generator Old Style

```
SELECT level AS n  
FROM dual  
CONNECT BY level <= 100;
```

Railroad Diagram



Restrictions

The recursive member cannot contain any of the following elements:

- The DISTINCT keyword or a GROUP BY clause
- The model_clause
- An aggregate function. However, analytic functions are permitted in the select list.
- Subqueries that refer to the recursive member.
- Outer joins that refer to recursive member as the right table.

Other Goodies

- `SYS_CONNECT_BY_PATH`
- `CONNECT_BY_ROOT`
- `CONNECT_BY_CYCLE`
- `CONNECT_BY_ISLEAF`
- `ORDER SIBLINGS BY`

Managers and Employees

EMPLOYEE_ID	NAME	SALARY	LVL	EMPLOYEES	TOTAL_SALARY
100	Steven King	24000	3	106	667416
101	Neena Kochhar	17000	2	11	92816
148	Gerald Cambrault	11000	1	6	51900
145	John Russell	14000	1	6	51000
146	Karen Partners	13500	1	6	51000
149	Eleni Zlotkey	10500	1	6	50000
147	Alberto Errazuriz	12000	1	6	46600
108	Nancy Greenberg	12008	1	5	39600
102	Lex De Haan	17000	2	5	28800
123	Shanta Vollman	6500	1	8	25900
121	Adam Fripp	8200	1	8	25400
122	Payam Kaufling	7900	1	8	23600
124	Kevin Mourgos	5800	1	8	23000
120	Matthew Weiss	8000	1	8	22100
103	Alexander Hunold	9000	1	4	19800
114	Den Raphaely	11000	1	5	13900
205	Shelley Higgins	12008	1	1	8300
201	Michael Hartstein	13000	1	1	6000

Managers and Employees

```
WITH CTE (employee_id, last_name, first_name, lvl,  
manager_salary, employee_salary, manager_id) AS  
(  
  
SELECT employee_id, last_name, first_name, 0, salary,  
salary, manager_id  
FROM employees  
  
UNION ALL  
  
SELECT e.employee_id, e.last_name, e.first_name, c.lvl+1,  
e.salary, c.employee_salary, e.manager_id  
FROM employees e INNER JOIN CTE c ON (e.employee_id =  
c.manager_id)  
  
)
```

Managers and Employees

```
SELECT
    employee_id,
    first_name||' '||last_name AS name,
    manager_salary AS salary,
    max(lvl) AS lvl,
    count(*) - 1 AS employees,
    sum(employee_salary) - manager_salary AS total_salary
FROM CTE
GROUP BY employee_id, last_name, first_name,
manager_salary
HAVING max(lvl) > 0
ORDER BY 6 desc, 1;
```

Coupon Clipping

Given a list of products and a list of discount coupons, we needed to find the minimum price for all the products based on certain rules. Here are those rules:

- A maximum of ten coupons can be applied on the same product.
- The discount price can not be less than 70% of the original price.
- The total amount of the discount can not exceed 30\$.

Coupon Clipping

Id	Name	Price	Discounted Price	Discount Amount	Discount Rate	Coupon Names
1	PROD 1	100.00	72.00	28.00	28.00	CP 1 : -15\$ + CP 2 : -5\$ + CP 3 : -10%
2	PROD 2	220.00	193.00	27.00	12.27	CP 1 : -15\$ + CP 4 : -12\$
3	PROD 3	15.00	13.50	1.50	10.00	CP 3 : -10%
4	PROD 4	70.00	49.50	20.50	29.29	CP 1 : -15\$ + CP 3 : -10%
5	PROD 5	150.00	121.50	28.50	19.00	CP 1 : -15\$ + CP 3 : -10%

Coupon Clipping

```
CREATE TABLE products (ID INTEGER PRIMARY KEY, Name
VARCHAR2(20), Price NUMBER);
```

```
INSERT INTO products VALUES (1, 'PROD 1', 100);
```

```
INSERT INTO products VALUES (2, 'PROD 2', 220);
```

```
INSERT INTO products VALUES (3, 'PROD 3', 15);
```

```
INSERT INTO products VALUES (4, 'PROD 4', 70);
```

```
INSERT INTO products VALUES (5, 'PROD 5', 150);
```

```
CREATE TABLE coupons (ID INTEGER PRIMARY KEY, Name
VARCHAR2(20), Value INTEGER, IsPercent CHAR(1));
```

```
INSERT INTO coupons VALUES (1, 'CP 1 : -15$', 15, 'N');
```

```
INSERT INTO coupons VALUES (2, 'CP 2 : -5$', 5, 'N');
```

```
INSERT INTO coupons VALUES (3, 'CP 3 : -10%', 10, 'Y');
```

```
INSERT INTO coupons VALUES (4, 'CP 4 : -12$', 12, 'N');
```

Coupon Clipping

WITH

```
RCTE(ID, Name, Price, DiscountedPrice, DiscountAmount,  
DiscountRate, CouponNames, CouponCount, CouponID) AS  
(
```

SELECT

```
  ID,  
  Name,  
  Price,  
  Price AS DiscountedPrice,  
  0 AS DiscountAmount,  
  0 AS DiscountRate,  
  CAST(' ' AS VARCHAR2(1024)) AS CouponNames,  
  0 AS CouponCount,  
  -1 AS CouponId
```

FROM

products

Coupon Clipping

UNION ALL

SELECT

```
RCTE.ID, RCTE.Name, RCTE.Price,  
  DECODE(C.IsPercent, 'N', RCTE.DiscountedPrice - C.Value,  
RCTE.DiscountedPrice - (RCTE.DiscountedPrice / 100 *  
C.Value)) DiscountedPrice,  
  RCTE.Price - DiscountedPrice AS DiscountAmount,  
  (RCTE.Price - DiscountedPrice) / RCTE.Price * 100 AS  
DiscountRate,  
  DECODE(RCTE.CouponNames, ' ', C.Name, RCTE.CouponNames  
|| ' + ' || C.Name) AS CouponNames,  
  RCTE.CouponCount + 1 AS CouponCount,  
  C.ID AS CouponID
```

FROM RCTE, coupons C

WHERE

```
c.id > RCTE.CouponID AND CouponCount <= 2 AND  
DiscountAmount <= 30 AND DiscountRate <= 30  
)
```

Coupon Clipping

```
SortedPrices AS
(
    SELECT
        RCTE.*,
        RANK() OVER (PARTITION BY ID ORDER BY DiscountedPrice)
AS Rank
    FROM RCTE
)

SELECT
    ID, Name, Price,
    DiscountedPrice, DiscountAmount, DiscountRate,
    CouponNames
FROM SortedPrices
WHERE Rank = 1
ORDER BY ID;
```

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

Sudoku

```
with x( s, ind ) as
(
select sud, instr( sud, ' ' )
from (select '53 7 6 195 98 6 8 6 34 8 3 17 2 6 6 28 419 5 8 79' sud
from dual )
union all
select substr( s, 1, ind - 1 ) || z || substr( s, ind + 1 ), instr( s, ' ', ind + 1 )
from x, (select to_char( rownum ) z from dual connect by rownum <= 9) z
where ind > 0
and not exists
(
select *
from (select rownum lp from dual connect by rownum <= 9)
where z = substr(s,trunc((ind-1)/9)*9+lp,1)
or z = substr(s,mod(ind-1,9)-8+lp*9,1)
or z = substr(s,mod( trunc(( ind-1)/3),3)*3+ trunc((ind-1)/27)*27+lp+trunc((lp-1)/3)*6,1))
)
select s from x
```

Thanks For Listening

iggy_fernandez@hotmail.com

iggyfernandez.wordpress.com

Please submit evaluation forms