

Dani Schnider

# Die Evolution von View-Optimierungen in der Oracle-Datenbank



**Callista**

# About me

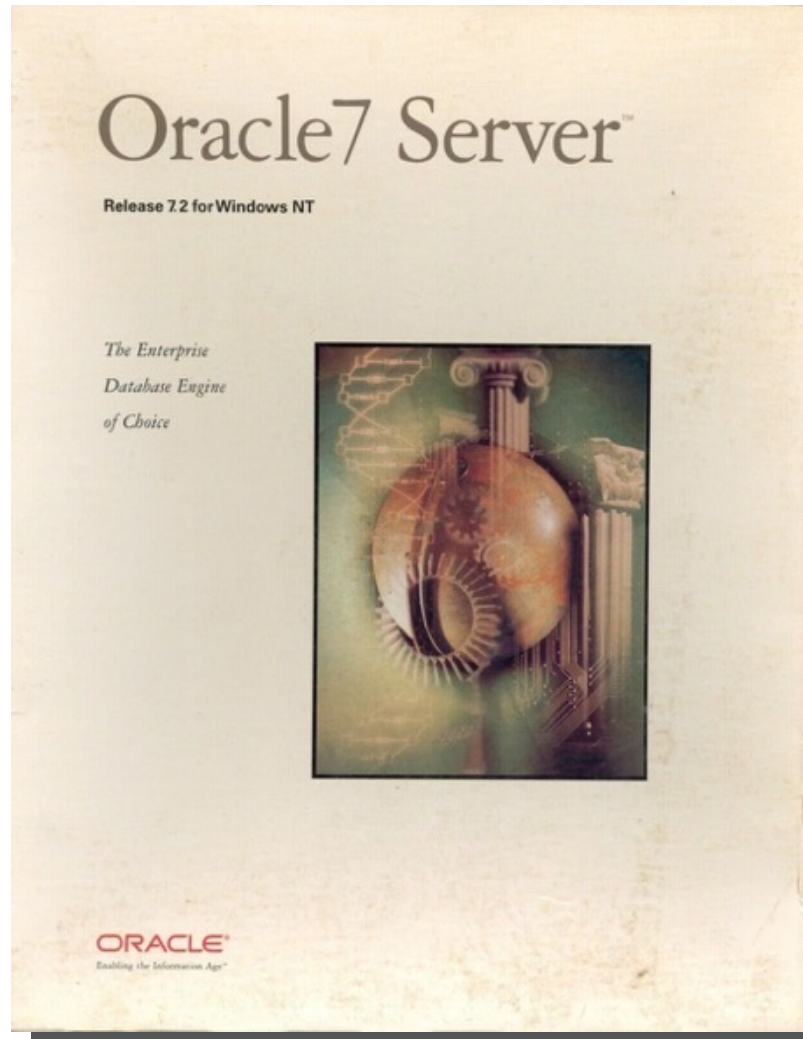


## Dani Schneider



- Working for Callista
- Oracle ACE Director
- Member of Symposium 42
- Hobby: Craft Beer Brewing




# 30 Years Back in Time...



# BACK IN 1994

<b>AVERAGE COST</b> A first class stamp \$0.29 A gallon of gas \$1.09 A gallon of milk \$2.88 A movie ticket \$4.18 A new house \$119,050.00	<b>U.S. PRESIDENT</b> ★ <i>Bill Clinton</i> ★ 	 <b>World POPULATION</b> 5.670 BILLION	<b>TECHNOLOGY</b> Netscape Navigator released and quickly became the market leader for browsing the web.  The world's first satellite digital television service is launched.  Amazon.com is founded in Bellevue, Washington by Jeff Bezos.
---	---	---	--

<b>AVERAGE INCOME PER YEAR</b> <b>\$37,070</b>	<b>IN THE NEWS...</b> After many years, the English Channel is opened, joining England to France for the first time.  Tonya Harding wins the national Figure Skating championship title but is stripped of her title following an attack on her rival Nancy Kerrigan.  O.J. Simpson flees from the police in his white Ford Bronco.  Major League Baseball Players Association begin a 232 day strike causing 1994 season to be cancelled.  Lisa Marie Presley marries Michael Jackson.	<b>IN THEATERS...</b> The Shawshank Redemption Pulp Fiction Forrest Gump Lion King  True Lies
---	--	---

<b>'90s SLANG</b> Aiiight - Alright, okay Da bomb! - Really cool As if! - Lack of interest Word - In agreement Boo ya! - In your face Dope - Something cool Talk to the hand - I don't want to hear it	<b>ON THE RADIO...</b> The Sign - Ace Of Base I Swear - All-4-One Regulate - Warren G Baby I Love Your Way - Big Mountain On Bended Knee - Boyz II Men	<b>ON Television...</b> Friends ER Seinfeld Frasier The Simpsons
---	---	---

*«Verwende keine Views in Oracle!  
Sie sind furchtbar langsam und  
verursachen eine Menge  
Performanceprobleme.»*

Ehemaliger Arbeitskollege, 1994

# Optimizer Transformations for Views

Transformation	Oracle Version
Simple View Merging	7.0
Filter Predicate Pushing	7.1
Complex View Merging	8.1.6
Join Predicate Pushing	10g Release 1
Join Elimination	10g Release 1
Subsumption of Views and Subqueries	23ai

# Mergeable View

```
CREATE OR REPLACE VIEW v_swiss_customers AS
SELECT a.cust_id, c.first_name, c.last_name
      , a.street, a.zip_code, a.city
FROM customers c
JOIN addresses a ON (a.cust_id = c.id)
WHERE a.ctr_code = 'CH'
```

❶ view definition

```
SELECT * FROM v_swiss_customers
WHERE last_name = 'Young'
```

❷ query against view

```
SELECT a.cust_id, c.first_name, c.last_name
      , a.street, a.zip_code, a.city
FROM customers c
JOIN addresses a ON (a.cust_id = c.id)
WHERE a.ctr_code = 'CH'
      AND last_name = 'Young'
```

❸ merged query to be optimized

# Non-Mergeable View

```
CREATE OR REPLACE VIEW v_swiss_customers AS
SELECT ROWNUM rn
      , a.cust_id, c.first_name, c.last_name
      , a.street, a.zip_code, a.city
FROM customers c
JOIN addresses a ON (a.cust_id = c.id)
WHERE a.ctr_code = 'CH'
```

```
SELECT * FROM v_swiss_customers
WHERE last_name = 'Young'
```



❶ view definition

❷ query against view

temporary  
row source



WHERE last\_name = 'Young'

❸ apply filter on view result

# Live Demo



View Merging  
Predicate Pushing  
Join Elimination



# Simple View Merging

```
SELECT * FROM v_swiss_customers
WHERE last_name = 'Young'
```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	NESTED LOOPS	
3	TABLE ACCESS BY INDEX ROWID BATCHED	CUSTOMERS
* 4	INDEX RANGE SCAN	CUST_LAST_FIRST_NAME
* 5	INDEX RANGE SCAN	ADR_CUST_ID
* 6	TABLE ACCESS BY INDEX ROWID	ADDRESSES

Predicate Information (identified by operation id):

```
4 - access("C"."LAST_NAME"='Young')
5 - access("A"."CUST_ID"="C"."ID")
6 - filter("A"."CTR_CODE"='CH')
```

## Notes:

- Simple view merging automatically combines the view definition with the query on the view
- Views containing only joins and WHERE conditions are usually mergeable views
- When a view is not visible in the execution plan, it is a mergeable view
- Available since Oracle 7.0

# No Simple View Merging

```
SELECT * FROM v_swiss_customers
WHERE last_name = 'Young'
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	<b>VIEW</b>	<b>V_SWISS_CUSTOMERS</b>
2	COUNT	
* 3	HASH JOIN	
* 4	TABLE ACCESS FULL	ADDRESSES
5	TABLE ACCESS FULL	CUSTOMERS

Predicate Information (identified by operation id):

```
1 - filter("LAST_NAME"='Young')
3 - access("A"."CUST_ID"="C"."ID")
4 - filter("A"."CTR_CODE"='CH')
```

## Notes:

- There are several reasons that a view is non-mergeable, e.g. GROUP BY, DISTINCT, set operations, outer joins, aggregations, analytic functions, etc.
- When a view is displayed in the execution plan, it is a non-mergeable view

# Example View for Complex View Merging

```
CREATE OR REPLACE VIEW v_countries
AS
SELECT ctr_code, name AS country_name
       , SUM(CASE WHEN adr_type LIKE '%D%' THEN 1 END) num_delivery_adr
       , SUM(CASE WHEN adr_type LIKE '%P%' THEN 1 END) num_payment_adr
FROM countries c
LEFT JOIN addresses a ON (c.code = a.ctr_code)
GROUP BY 1, 2
```

# Complex View Merging

```
SELECT * FROM v_countries
WHERE country_name = 'Canada'
```

Id	Operation	Name
0	SELECT STATEMENT	
1	HASH GROUP BY	
2	NESTED LOOPS OUTER	
* 3	TABLE ACCESS FULL	COUNTRIES
4	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 5	INDEX RANGE SCAN	ADR_CTR_CODE

Predicate Information (identified by operation id):

```
3 - filter("C"."NAME"='Canada')
5 - access("C"."CODE"="A"."CTR_CODE" (+))
```

## Notes:

- Views with DISTINCT or GROUP BY are non-mergeable views
- With complex view merging, they can be converted to mergeable views
- Optimizer evaluates costs with and without complex view merging and uses execution plan with lower costs
- Available since Oracle 8.1.6

# Example View for Predicate Pushing

```
CREATE OR REPLACE VIEW v_extended_addresses AS
SELECT cust_id, adr_type, zip_code, city, ctr_code, plz.canton_code
  FROM addresses adr
  JOIN swiss_postal_codes plz on (adr.zip_code = plz.postal_code)
 WHERE ctr_code = 'CH'
UNION ALL
SELECT cust_id, adr_type, zip_code, city, ctr_code, 'n/a'
canton_code
  FROM addresses adr
 WHERE ctr_code != 'CH'
```

The view is non-mergeable because of the set operator (UNION ALL)

# Filter Predicate Pushing

```
SELECT * FROM v_extended_addresses
WHERE city = 'Burgdorf'
```

Id	Operation	Name
0	SELECT STATEMENT	
1	<b>VIEW</b>	<b>V_EXTENDED_ADDRESSES</b>
2	UNION-ALL	
3	NESTED LOOPS	
4	NESTED LOOPS	
* 5	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 6	<b>INDEX RANGE SCAN</b>	<b>ADR_CITY</b>
* 7	INDEX RANGE SCAN	PLZ_CODE
8	TABLE ACCESS BY INDEX ROWID	SWISS_POSTAL_CODES
* 9	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 10	<b>INDEX RANGE SCAN</b>	<b>ADR_CITY</b>

```
...
6 - access("ADR"."CITY"='Burgdorf')
10 - access("CITY"='Burgdorf')
```

## Notes:

- With predicate pushing, the WHERE condition of the query is applied within the non-mergeable view
- In this example, the filter on city can be used in lines 6 and 10, so the number of rows can already be reduced within the view
- Available since Oracle 7.1

# Join Predicate Pushing

```
SELECT * FROM customers c
  JOIN v_extended_addresses a ON (a.cust_id = c.id)
 WHERE c.last_name = 'Young'
```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	TABLE ACCESS BY INDEX ROWID BATCHED	CUSTOMERS
* 3	INDEX RANGE SCAN	CUST_LAST_FIRST_NAME
4	<b>VIEW</b>	<b>V_EXTENDED_ADDRESSES</b>
5	UNION ALL <b>PUSHED PREDICATE</b>	
6	NESTED LOOPS	
7	NESTED LOOPS	
* 8	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 9	<b>INDEX RANGE SCAN</b>	<b>ADR_CUST_ID</b>
* 10	INDEX RANGE SCAN	PLZ_CODE
11	TABLE ACCESS BY INDEX ROWID	SWISS_POSTAL_CODES
* 12	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 13	<b>INDEX RANGE SCAN</b>	<b>ADR_CUST_ID</b>

## Notes:

- Predicate pushing is not only available for filter conditions in the query, but also for joins columns of the view
- In this example, the join column cust\_id in the view reduces the result set, although the values depend on the join with table customers in the query
- Available since Oracle 10.1

# Join Elimination

```
SELECT first_name, last_name, street, zip_code, city
FROM v_swiss_customers
WHERE cust_id = 2573
```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	TABLE ACCESS BY INDEX ROWID	CUSTOMERS
* 3	INDEX UNIQUE SCAN	CUST_PK
* 4	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 5	INDEX RANGE SCAN	ADR_CUST_ID

```
SELECT street, zip_code, city
FROM v_swiss_customers
WHERE cust_id = 2573
```

Id	Operation	Name
0	SELECT STATEMENT	
* 1	TABLE ACCESS BY INDEX ROWID BATCHED	ADDRESSES
* 2	INDEX RANGE SCAN	ADR_CUST_ID

## Notes:

- Optimizer is able to remove tables from a query when the result is not affected
- Typical optimization for views containing joins
- Only possible when foreign key constraints are defined
- Available since Oracle 10.1





# Examples from the Real World



# New Optimizer Transformations in Oracle 23ai

## Grouping, Subsumption, and Disjunctive Join Optimizations in Oracle

Rafi Ahmed

Oracle America Inc.  
Redwood City, CA, USA  
rafi.ahmed@oracle.com

Krishna Kantikiran Pasupuleti

Oracle America Inc.  
Redwood City, CA, USA  
kanti.kiran@oracle.com

Sriram Tirupattur

Oracle America Inc.  
Redwood City, CA, USA  
sriram.tirupattur@oracle.com

Lei Sheng

Oracle America Inc.  
Redwood City, CA, USA  
lei.sheng@oracle.com

Hong Su

Oracle America Inc.  
Redwood City, CA, USA  
hong.su@oracle.com

Mohamed Ziauddin

Oracle America Inc.  
Redwood City, CA, USA  
mohamed.ziauddin@oracle.com

### ABSTRACT

Query optimization must evolve with new workloads. As analytic and data warehouse workloads become more ubiquitous, optimization techniques that reduce the amount of data processed during query execution, enable shared computation and avoid expensive data access and joins must be rigorously explored. In this paper, we present aggregate-decomposition techniques as enhancements to an existing query transformation that performs grouping before joins. Consequently, the techniques reduce the amount of data processed during query execution, enable shared computation and avoid expensive data access and joins. Consequently, the techniques reduce the amount of data processed during query execution, enable shared computation and avoid expensive data access and joins.

rewriting these constructs such that the new equivalent forms i) prune data during query execution as early as possible so that the overhead of subsequent computation is reduced, ii) avoid redundant computation through sharing of common parts, iii) open up access paths and operations that would otherwise not be possible, etc. In this paper we present some techniques built into the Oracle query optimizer that achieve the above stated objectives of query optimization.

Grouping rows before joins is a well-known technique [5, 22] used by query optimizers to improve the efficiency of joins by reducing the amount of data processed during query execution. This paper presents new techniques for grouping rows before joins that achieve the above stated objectives of query optimization.

# Subsumption of Views and Subqueries

```
SELECT *
FROM (SELECT SUM(i.quantity) sum_quantity
      FROM order_items i
      JOIN products p ON (i.prod_id = p.id)
      WHERE p.id = 1) v1,
      (SELECT SUM(i.price_per_unit) sum_amount
      FROM order_items i
      JOIN products p ON (i.prod_id = p.id)
      WHERE p.id = 1) v2;
```

Id	Operation	Name
0	SELECT STATEMENT	
1	NESTED LOOPS	
2	VIEW	
3	SORT AGGREGATE	
* 4	TABLE ACCESS FULL	ORDER_ITEMS
5	VIEW	
6	SORT AGGREGATE	
* 7	TABLE ACCESS FULL	ORDER_ITEMS

ORACLE Database 23<sup>ai</sup>

Id	Operation	Name
0	SELECT STATEMENT	
1	VIEW	<b>VW_SVS_A18161FF</b>
2	SORT AGGREGATE	
* 3	TABLE ACCESS FULL	ORDER_ITEMS

# Live Demo



Subsumption of Views  
JSON Relational Duality Views

# JSON Relational Duality View

```
CREATE OR REPLACE JSON RELATIONAL DUALITY VIEW v_orders_order_items_json AS
orders @insert @update @delete
{
  _id          : id
  cust_id      : cust_id
  order_date   : order_date
  order_items @insert @update @delete
  {
    _id          : id
    line_no      : line_no
    prod_id      : prod_id
    delivery_date : delivery_date
    quantity     : quantity
    price_per_unit : price_per_unit
  }
}
```

# JSON Relational Duality View

```
SELECT JSON_SERIALIZE(dv.data PRETTY)
  FROM v_orders_order_items_json dv
 WHERE dv.data."_id" = 12345
```

Id	Operation	Name
0	SELECT STATEMENT	
1	SORT GROUP BY	
2	TABLE ACCESS BY INDEX ROWID BATCHED	ORDER_ITEMS
* 3	INDEX RANGE SCAN	ORDI_ORDER_ID
4	TABLE ACCESS BY INDEX ROWID	ORDERS
* 5	INDEX UNIQUE SCAN	ORD_PK

Predicate Information (identified by operation id):

```
3 - access("OUTER_ALIAS1"."ORDER_ID"=:B1)
5 - access("ID"=12345)
```

## Notes:

- JSON Relational Duality Views contains only joins between tables
- They can be converted to mergeable views
- Available since Oracle 23ai

# Conclusion

*«Verwende keine Views in Oracle!  
Sie sind furchtbar langsam und  
verursachen eine Menge  
Performanceprobleme.»*

Ehemaliger Arbeitgeber, 1994

## Wrong!

Mergeable views never cause performance problems

Several query transformations help to reduce / avoid performance problems with non-mergeable views

Views are a wonderful database feature! I am a big fan of views 😊

# Further Information

<https://docs.oracle.com/en/database/oracle/oracle-database/23/tgsql/query-transformations.html>

<https://blogs.oracle.com/optimizer/post/optimizer-transformations-view-merging-part-1>

<https://blogs.oracle.com/optimizer/post/optimizer-transformations-view-merging-part-2>

<https://blogs.oracle.com/optimizer/post/optimizer-transformation-join-predicate-pushdown>

[https://jonathanlewis.wordpress.com/2014/12/12/push\\_pred-evolution](https://jonathanlewis.wordpress.com/2014/12/12/push_pred-evolution)

[https://jonathanlewis.wordpress.com/2023/10/11/no\\_merge-joelkallmanday](https://jonathanlewis.wordpress.com/2023/10/11/no_merge-joelkallmanday)

<https://www.vldb.org/pvldb/vol17/p4200-pasupuleti.pdf>



A wide-angle landscape photograph of a mountain range. The foreground features a large, calm lake with a small island in the middle. The middle ground shows rolling hills and a winding road. The background is dominated by towering, rugged mountains with significant snow cover and patches of glaciers. The sky is filled with soft, grey clouds.

**DRIVEN BY EXCELLENCE**