mip Management Informationspartner GmbH

SQL in Action – SQL Traps and Solutions

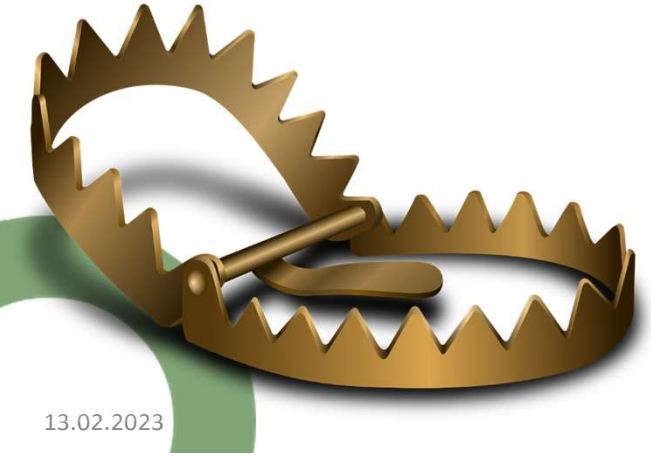
Michael Tiefenbacher

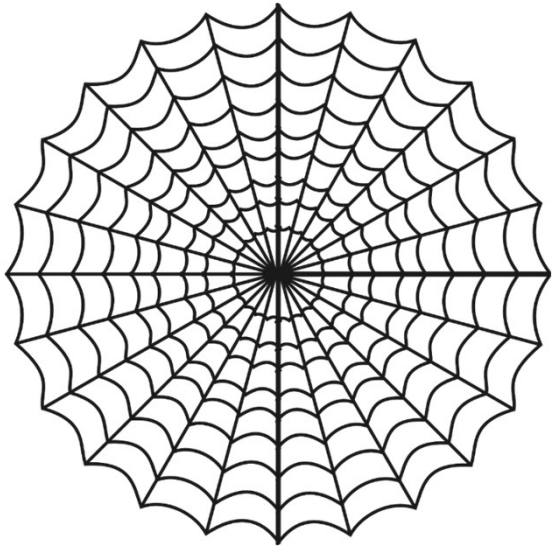
Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS

Introduction & Motivation

- Looking for errors is a daily task for most of us
  - Whether is self-made or supporting others
- Small changes to SQLs can have a huge effect
- Practical experience is helpful to find hidden errors
- Some problems cause troubles more than once
  - these should be discussed here to help you to avoid them and to learn about SQL





Introduction & Motivation

- What‘s wrong?

```
SELECT IN_ID,
       CREATION_DATE
       CHANGE_DATE,
       RENEWAL_DATE
  FROM INSURANCE
 WHERE CHANGE_DATE IS NULL
   AND DIVISION <> 'CAR'
   AND LOADMODE != ''
```

NSURANCE 1

SELECT IN_ID, CREATION_DATE CHANGE_DATE, RENEV | Enter a SQL expression

	IN_ID	CHANGE_DATE	RENEWAL_DATE
1	75099332414	2020-08-01-17.47.34.337000	2022-08-01-17.47.34.337000
2	75091227642	2021-06-26-17.50.13.613000	2022-06-26-17.50.13.613000
3	75099677621	2018-09-26-17.50.13.624000	2021-09-26-17.50.13.624000
4	75092387003	2020-06-11-17.50.13.627000	2022-06-11-17.50.13.627000

Introduction & Motivation

- Syntax errors are rapidly reported by Db2
  - But can sometimes be quite strange
- i.e. complex SQL returns error:
  SQL0183 A datetime arithmetic operation or a datetime scalar function has a result that is not within the valid range of dates
  - only when executed with
    `WHERE datum = '2021-09-12'`
    and datum is a date column
  - Without that it runs successfully
- What happened?
  - Within the SQL following condition existed
    `next_week(datum - 1 day) as datum`
  - In a cte the date range was limited to the current year
    due to optimization the evaluation sequence changed
  - the datum column had also values of `0001-01-01` which led to the error

not our focus today

Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS

GROUP BY Problem

```
SQL0119N  An expression starting with "<expression-start>" specified in
      a SELECT clause, HAVING clause, or ORDER BY clause is not
      specified in the GROUP BY clause or it is in a SELECT clause,
      HAVING clause, or ORDER BY clause with a column function and no
      GROUP BY clause is specified.
```

- Column functions need a GROUP BY clause for all columns without column function
- I guess we all have forgotten the GROUP BY clause the other day
- It seems pretty basic but maybe it‘s worth another look

GROUP BY – Example I

- Will this work?

```
SELECT sum(salary)
FROM DB2ADMIN.EMPLOYEE
```

```
SELECT workdept, sum(salary)
FROM DB2ADMIN.EMPLOYEE
```

```
SELECT workdept, sum(salary)
FROM DB2ADMIN.EMPLOYEE
GROUP BY 1
```

```
SELECT 2 AS ID, 2 * 3 + 4 AS calc, sum(salary)
FROM DB2ADMIN.EMPLOYEE
```

```
SELECT 2 AS ID, 2 * 3 + 4 AS calc, sum(salary)
FROM DB2ADMIN.EMPLOYEE
GROUP BY 1, 2
```

GROUP BY – Example II

- How will this work?

```
SELECT CASE
         WHEN hiredate < '1990-01-01' THEN 'Before 1990'
         WHEN hiredate < '2000-01-01' THEN 'Before 2000'
         WHEN hiredate < '2010-01-01' THEN 'Before 2010'
       END AS hire_range
     , sum(SALARY) Salary
     , sum(BONUS) AS Bonus
  FROM DB2ADMIN.EMPLOYEE
 GROUP BY ??
```

```
hire_range
```

```
CASE
  WHEN hiredate < '1990-01-01' THEN 'Before 1990'
  WHEN hiredate < '2000-01-01' THEN 'Before 2000'
  WHEN hiredate < '2010-01-01' THEN 'Before 2010'
END
```

GROUP BY Examples

- Better…

```
SELECT hire_range
     , sum(SALARY) AS Salary
     , sum(BONUS) AS Bonus
  FROM (
         SELECT CASE
                  WHEN hiredate < '1990-01-01' THEN 'Before 1990'
                  WHEN hiredate < '2000-01-01' THEN 'Before 2000'
                  WHEN hiredate < '2010-01-01' THEN 'Before 2010'
                END AS hire_range
              , SALARY
              , BONUS
           FROM DB2ADMIN.EMPLOYEE)
 GROUP BY hire_range
```

… and necessary if non-deterministic functions are used

GROUP BY Examples

There are even more things related to GROUP BYs – check out

- Grouping Sets
  - Same result set other GROUP BY clause – avoiding an union all
  - i.e. GROUP BY GROUPING SET ((a), (b))
- Super-Groups (Rollup, Cube)
  - can be thought of as abbreviation for a GROUPING SET specification
  - Efficient through only a single scan of data & reuse of sort results
- Details and examples see student notes
  - Great presentation from Calisto Zuzarte at IDUG NA 2022

Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS

OR Problem

- Is there a difference?
- If so – what is the difference?

```
SELECT *
  FROM DB2ADMIN.EMPLOYEE
 WHERE edlevel = '18' OR sex = 'M' AND NOT(workdept = 'A00')
```

26 rows

```
SELECT *
  FROM DB2ADMIN.EMPLOYEE
 WHERE edlevel = '18' AND NOT(workdept = 'A00') OR sex = 'M'
```

27 rows

Conditions

- Difference exists due to evaluation sequence
- Rule: NOT precedes AND precedes OR

```
SELECT *
  FROM DB2ADMIN.EMPLOYEE
 WHERE edlevel = '18' OR (sex = 'M' AND NOT(workdept = 'A00'))
```

	EMPNO	FIRSTNME	LASTNAME	WORKDEPT	EDLEVEL	SEX
1	200120	GREG	ORLANDO	A00	14	M
2	000120	SEAN	O'CONNELL	A00	14	M
3	000110	VINCENZO	LUCCHESSI	A00	19	M

```
SELECT *
  FROM DB2ADMIN.EMPLOYEE
 WHERE (edlevel = '18' AND NOT(workdept = 'A00')) OR sex = 'M'
```

	EMPNO	FIRSTNME	LASTNAME	WORKDEPT	EDLEVEL	SEX
1	200010	DIAN	HEMMINGER	A00	18	F
2	000010	CHRISTINE	HAAS	A00	18	F

Conditions with Logical Operators

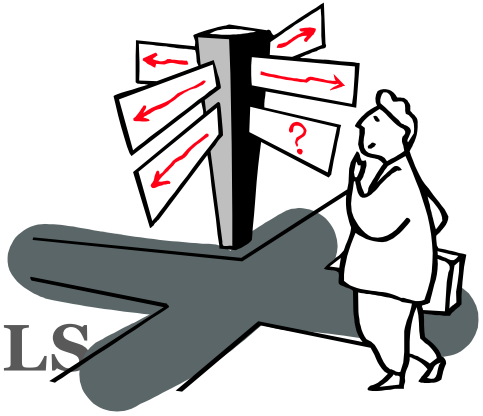
- Hint:
  Always use parentheses for easier understanding and readability ORs should be kept in parentheses

- Side Information
  - Did you know following conditions are equal:

```
WHERE workdept = 'A00' AND edlevel = '19' AND sex = 'M'
WHERE (workdept, edlevel, sex) = ('A00', '19', 'M')
```

Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS



OLAP Functions

- Very useful
- No need for GROUP BY
- Mentioned in most of my presentations – cause I love them
  - Check out my other “SQL in Action“ presentations

Hint: Great performance!

OLAP Functions

- SUM Example
  Show all Managers with the overall sum
  of department salaries and the number of employees
  in the department

```
SELECT empno, lastname, salary
     , count(*)    OVER (PARTITION BY workdept) AS ANZMA
     , sum(salary) OVER (PARTITION BY workdept) AS DEPTSALARY
  FROM EMPLOYEE
 WHERE job = 'MANAGER'
```

	EMPNO	LASTNAME	SALARY	ANZMA	DEPTSALARY
1	000020	THOMPSON	94,250	1	94,250
2	000030	KWAN	98,250	1	98,250
3	000060	STERN	72,250	1	72,250
4	000070	PULASKI	96,170	1	96,170

- Wrong result – WHERE condition gets evaluated before the OLAP function

OLAP Functions – Solution

- Common Table Expression to evaluate OLAP function upfront
  - If you want to apply the OLAP function on all rows rows (globally) but return only a subset
- Final SELECT to filter resultset

```
WITH cte AS (
SELECT empno, Lastname, job, workdept, SALARY
     , count(*) OVER (PARTITION BY WORKDEPT) AS ANZMA
     , sum(SALARY) OVER (PARTITION BY WORKDEPT) AS DEPTSALARY
  FROM EMPLOYEE
)
SELECT *
  FROM cte
 WHERE JOB = 'MANAGER'
 ORDER BY deptsalary desc
```

	EMPNO	LASTNAME	JOB	WORKDEPT	SALARY	ANZMA	DEPTSALARY
1	000060	STERN	MANAGER	D11	72,250	11	646,620
2	000070	PULASKI	MANAGER	D21	96,170	7	358,680
3	000090	HENDERSON	MANAGER	E11	89,750	7	317,140
4	000030	KWAN	MANAGER	C01	98,250	4	308,890
5	000100	SPENSER	MANAGER	E21	86,150	6	282,520

Example – Business Problem

- Select the sum of amount
  and the last values for sale_date
  and unit_price
  per customer and product

	ID	CUSTOMER_NO	PRODUCT_NO	AMOUNT	SALE_DATE	UNIT_PRICE
1	1,000	3,355	567,890	120	2022-08-03	14.79
2	1,001	3,355	567,890	120	2022-08-12	14.85
3	1,002	3,355	567,890	90	2022-08-24	15.5
4	1,003	3,355	567,890	150	2022-09-02	16.2
5	1,004	4,810	567,890	120	2022-08-05	23.9
6	1,005	4,810	567,890	120	2022-09-10	25.6

Solution using OLAP Functions

- First approach

```
SELECT customer_no
     , product_no
     , sum(amount) OVER (PARTITION BY customer_no, product_no) AS SUM_AMOUNT
     , last_value(sale_date) OVER (PARTITION BY customer_no, product_no
                                   ORDER BY sale_date) AS Last_sale
     , last_value(unit_price) OVER (PARTITION BY customer_no, product_no
                                    ORDER BY sale_date) AS price
  FROM product_sales
```

- Works for sum
- Last_sale and price seem to have a problem

	CUSTOMER_NO	PRODUCT_NO	SUM_AMOUNT	LAST_SALE	PRICE
1	3,355	567,890	480	2022-08-03	14.79
2	3,355	567,890	480	2022-08-12	14.85
3	3,355	567,890	480	2022-08-24	15.5
4	3,355	567,890	480	2022-09-02	16.2
5	4,810	567,890	240	2022-08-05	23.9
6	4,810	567,890	240	2022-09-10	25.6

Solution using OLAP Functions

- alternative approach

```
SELECT customer_no
     , product_no
     , sum(amount) OVER (PARTITION BY customer_no, product_no) AS SUM_AMOUNT
     , first_value(sale_date)  OVER (PARTITION BY customer_no, product_no
                                     ORDER BY sale_date desc) AS Last_sale
     , first_value(unit_price) OVER (PARTITION BY customer_no, product_no
                                     ORDER BY sale_date desc) AS price
  FROM product_sales
```

- Works
  Distinct has been left out for the show case

	CUSTOMER_NO	PRODUCT_NO	SUM_AMOUNT	LAST_SALE	PRICE
1	3,355	567,890	480	2022-09-02	16.2
2	3,355	567,890	480	2022-09-02	16.2
3	3,355	567,890	480	2022-09-02	16.2
4	3,355	567,890	480	2022-09-02	16.2
5	4,810	567,890	240	2022-09-10	25.6
6	4,810	567,890	240	2022-09-10	25.6

Solution using OLAP Functions

- Why does first_value work and last_value doesn‘t?
- Is this a bug?

- No it is not a bug!
- The OLAP window is missing
- Gives great option to explain it with this use case

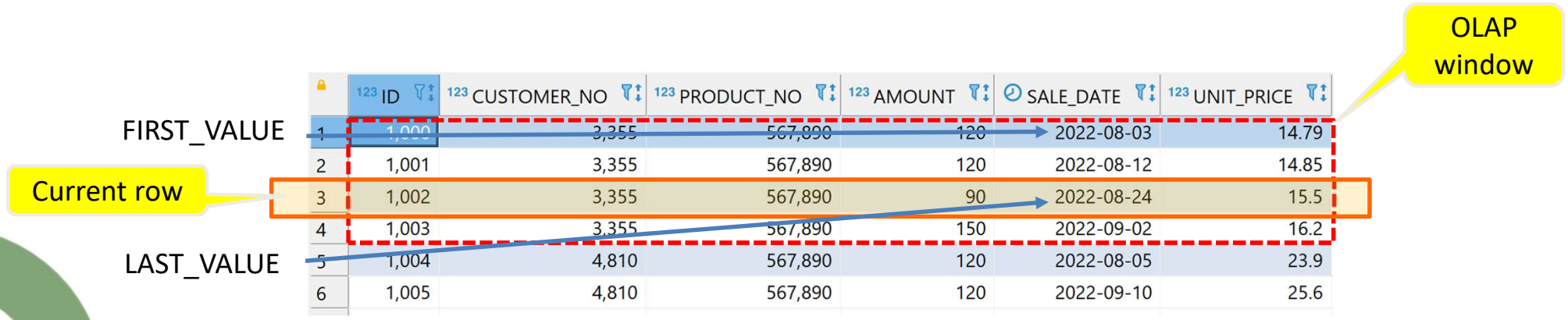
OLAP Function – Solution

```
SELECT customer_no
     , product_no
     , sum(amount) OVER (PARTITION BY customer_no, product_no) AS SUM_AMOUNT
     , last_value(sale_date) OVER (PARTITION BY customer_no, product_no
                                   ORDER BY sale_date
                                   ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
       AS last_sale
     , last_value(unit_price) OVER (PARTITION BY customer_no, product_no
                                    ORDER BY sale_date
                                    ROWS BETWEEN UNBOUNDED PRECEDING AND UNBOUNDED FOLLOWING)
       AS price
  FROM product_sales
```

	CUSTOMER_NO	PRODUCT_NO	SUM_AMOUNT	LAST_SALE	PRICE
1	3,355	567,890	480	2022-09-02	16.2
2	3,355	567,890	480	2022-09-02	16.2
3	3,355	567,890	480	2022-09-02	16.2
4	3,355	567,890	480	2022-09-02	16.2
5	4,810	567,890	240	2022-09-10	25.6
6	4,810	567,890	240	2022-09-10	25.6

OLAP Function – Solution – Reason

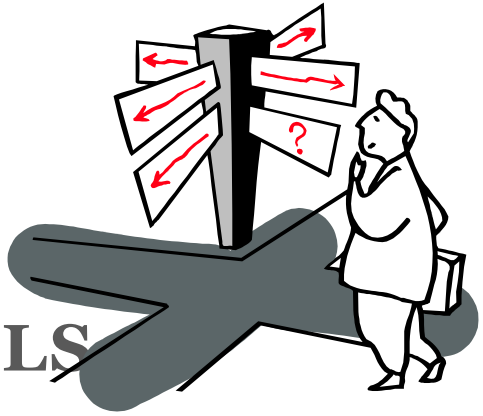
- The OLAP window clause is missing
  - If not specified it defaults to
    - when ORDER BY is NOT used: all rows in the window partition
    - when ORDER BY is used: all rows preceding
  - Because FIRST_VALUE was within the OLAP window it worked



	ID	CUSTOMER_NO	PRODUCT_NO	AMOUNT	SALE_DATE	UNIT_PRICE
1	1,000	3,355	567,890	120	2022-08-03	14.79
2	1,001	3,355	567,890	120	2022-08-12	14.85
3	1,002	3,355	567,890	90	2022-08-24	15.5
4	1,003	3,355	567,890	150	2022-09-02	16.2
5	1,004	4,810	567,890	120	2022-08-05	23.9
6	1,005	4,810	567,890	120	2022-09-10	25.6

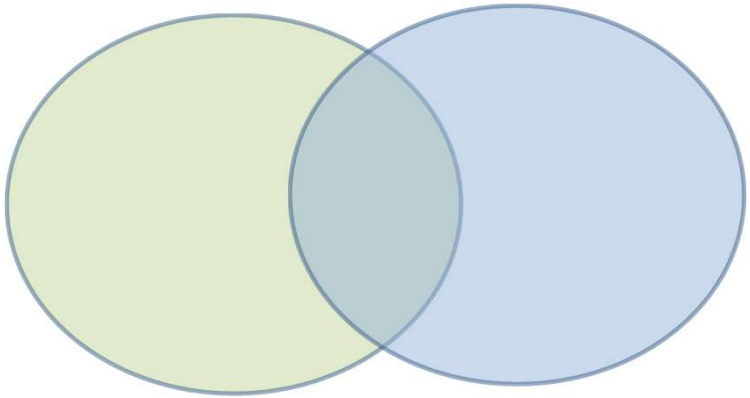
Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS



Outer Join

- LEFT, RIGHT or FULL OUTER Join
- Helpful to enrich Data
- Often used in Data Warehouse environments



Left Outer Join

Table 1

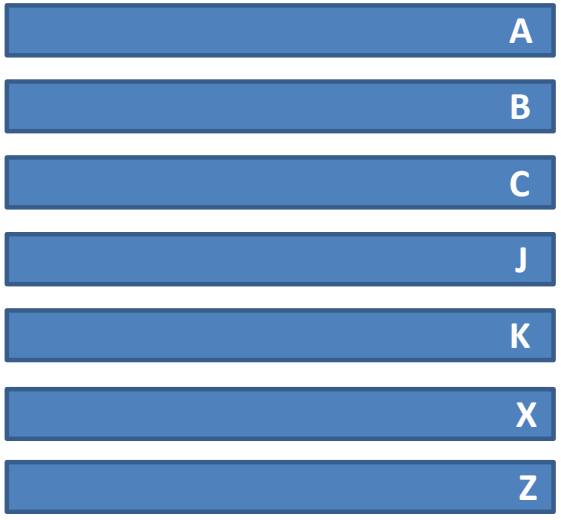
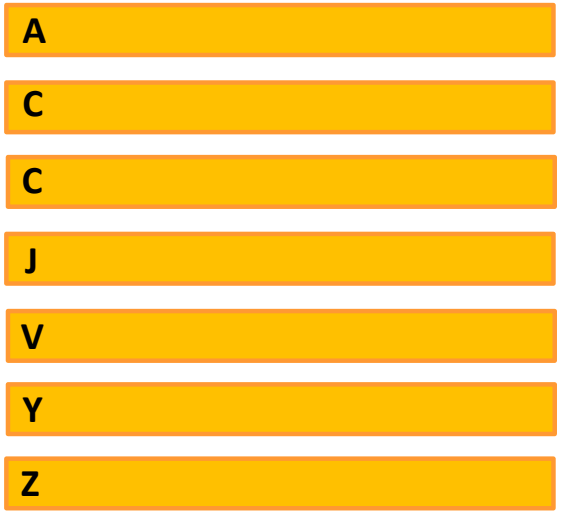
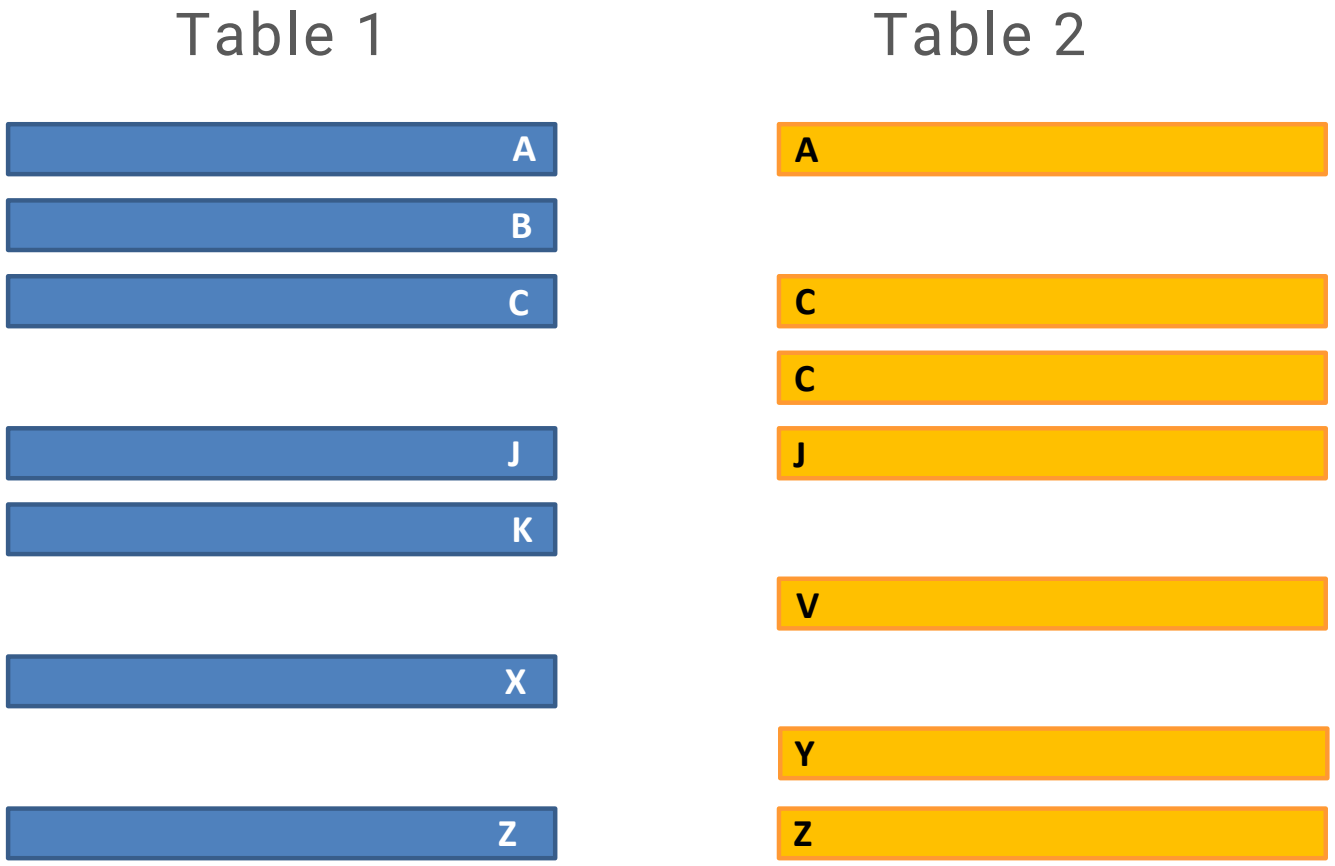


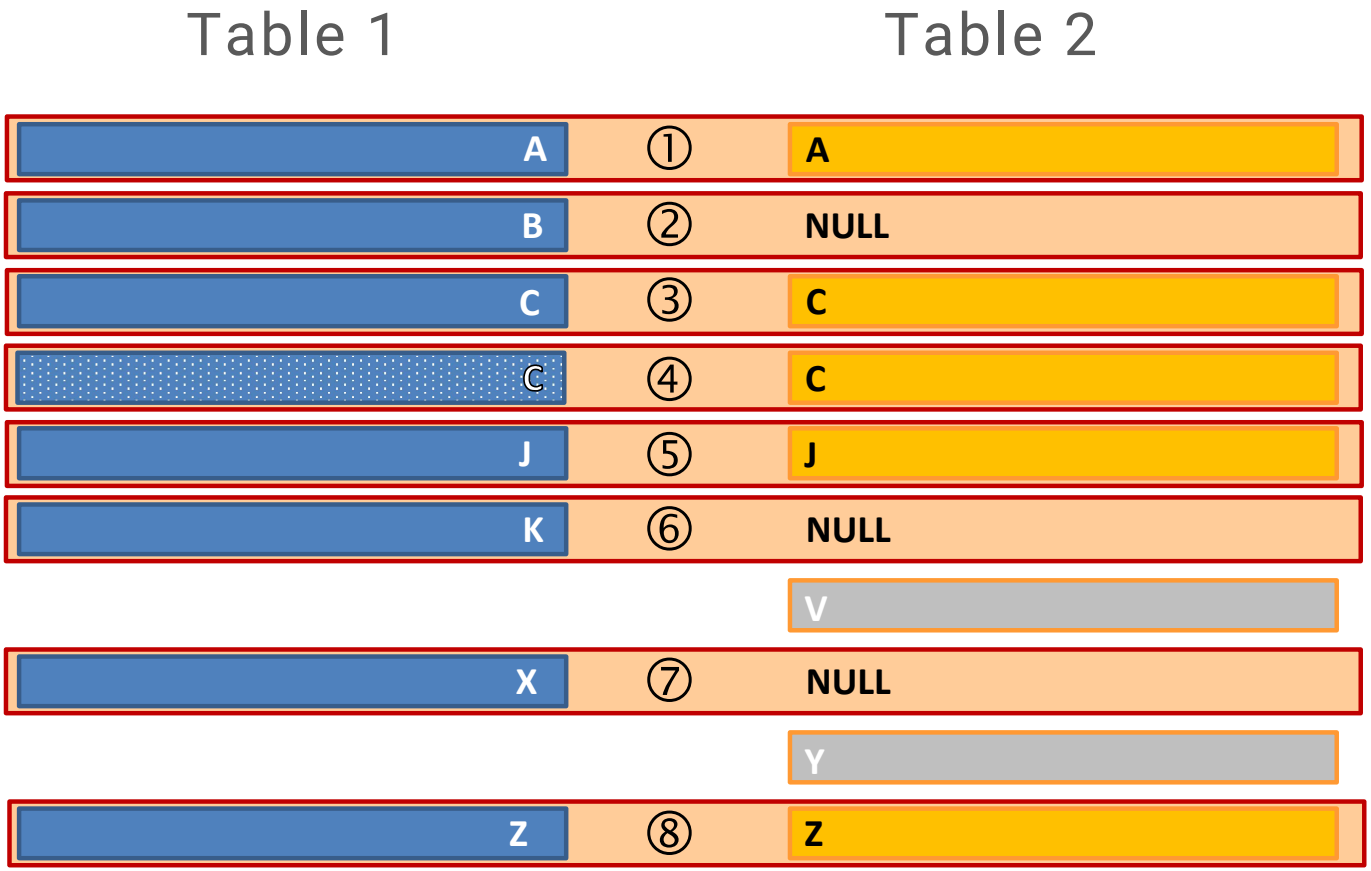
Table 2



Left Outer Join



Left Outer Join



Outer Join

- What is the difference between Join predicate and WHERE condition?
- Is there a difference between those two statements?
- If so – what is it?

```
select *
  from employee
  left join department
         on workdept = deptno
            and deptno = 'A00'
```

42 rows

```
select *
  from employee
  left join department
         on workdept = deptno
 where deptno = 'A00'
```

6 rows

Outer Join Example

```
create table jointest (id int, text varchar(20));
insert into jointest values (1, 'A'), (2, 'B');

SELECT '1. Query' as Query, a.*, b.*
  FROM jointest a
  LEFT JOIN jointest b
         ON a.id = b.id AND b.id = 1
UNION ALL
SELECT '2. Query' as Query, a.*, b.*
  FROM jointest a
  LEFT JOIN jointest b
         ON a.id = b.id
 WHERE b.id = 1
```

```
QUERY      ID TEXT ID   TEXT
---------- -- ---- ---- ----
1. Query    1 A       1 A
1. Query    2 B    NULL NULL
2. Query    1 A       1 A
```

Outer Join – Practice Example

- Data need to be deleted
- This deletion need to be executed in the warehouse
- A deletion is marked with
  Input Timestamp (ITS) = History Timestamp (HTS)
- An exception to this is a revival where an ID exists
- So revivals have been checked with following query

- Now simply select all that can be deleted

```
select *
  from tab_hist h
  left join tab_akt a
         on h.id = a.id
            and h.its = h.hts
where a.id is not null
```

Outer Join – Practice Example

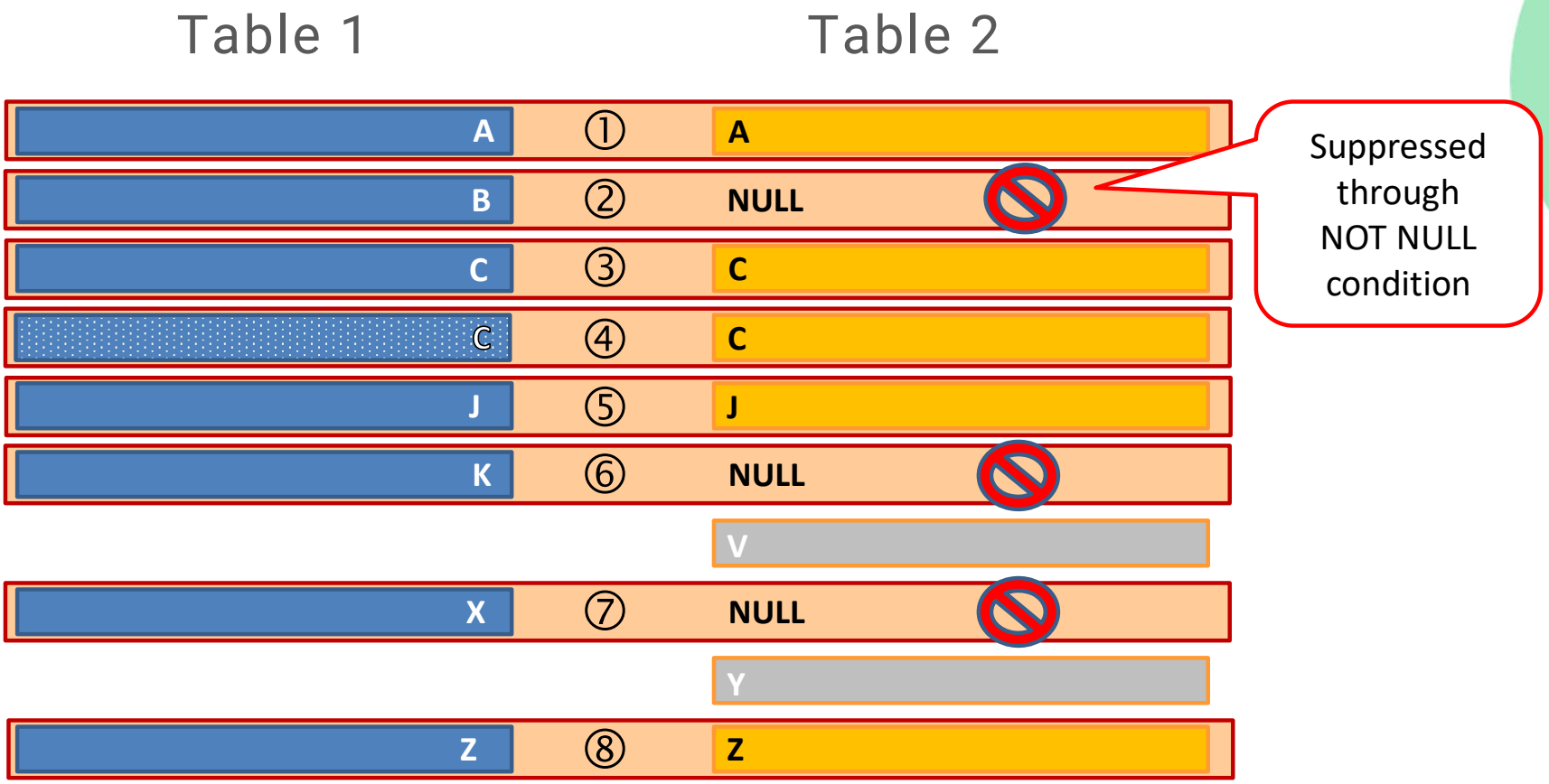
- Simply convert the SQL to return the opposite

```
delete from tab_akt
select *
  from tab_hist h
  left join tab_akt a
         on h.id = a.id
            and h.its = h.hts
where a.id is not null
```

- But this is a mistake!

- Original Statement worked but was definied sub-optimal
- It was an effective inner join although a left join was coded
- Reversing the condition changed this behavior and led to a wrong result

Outer Join – Practice Example



Outer Join

- Do not filter the inner table in the WHERE Condition
  - It will convert the outer join to an inner one

- WHERE Condition will filter rows
- Join condition will determine which rows to join

Agenda

- Introduction & Motivation
- GROUP BY Problem
- OR Problem
- OLAP Function Evaluation
- Outer Join Confusion
- <> in WHERE Condition with NULLS

NULLs in SQL

- Queries with NULL search
  - WHERE col IS NULL
  - WHERE col IS NOT NULL
  - WHERE col = NULL will not work!
  - Querying < NULL, <= NULL, > NULL and >= NULL are not valid.

- NULLABLE columns in calculations
  - Calculations with NULL return NULL
    - 10 + NULL => NULL
    - Test || NULL => NULL
- Column function
  - SUM(GEHALT) => NULL rows will be **ignored**
  - AVG(GEHALT) => NULL rows will be **ignored** => **biased data!**

Exception:
NULL equals NULL
in a GROUP BY

NULL – Example I

```
create table nulltest(id int not null,name varchar(30),salary dec(12,2));
insert into nulltest values (1, 'HUBER', 10000.00)
                          , (2, 'SCHNEIDER', 5000.00)
                          , (3, 'MEIER', NULL)
                          , (4, 'MUELLER', 45000.00)
                          , (5, NULL, NULL);

select sum(salary) as sum,
       avg(salary) as avg,
       min(salary) as min,
       max(salary) as max
  from nulltest;

 SUM      AVG                          MIN     MAX
 -------- ---------------------------- ------- --------
 60000.00 20000.000000000000000000000 5000.00 45000.00
```

20000 is the average of all NON-NULL rows
The „real“ average would be 12000!

NULL – Example II

```
select *
  from nulltest
 where Name <> 'HUBER';
```

```
 ID NAME       GEHALT
 -- --------- --------
  2 SCHNEIDER  5000.00
  3 MEIER          NULL
  4 MUELLER   45000.00
```

Four rows are <> HUBER but only three get returned because a comparison with NULL will always be invalid

Logical Conditions

ID	A	B	A_and_B	A_or_B
1	True	True	True	True
2	True	False	False	True
3	False	True	False	True
4	False	False	False	False
5	True	NULL	NULL	True
6	False	NULL	False	NULL
7	NULL	True	NULL	True
8	NULL	False	False	NULL
9	NULL	NULL	NULL	NULL

More Conditions

- DISTINCT predicate
  - `is distinct from`
  - `is not distinct from`
- Does not return NULL even when NULL is involved!
  - NULL is not distinct from NULL
- Available since
  - Db2 11.1 Mod 1
  - Db2 for z/OS Version 8

Logical Conditions II

ID	A	B	A_and_B	A_or_B	A_is_distinct_from_B	A_is_not_distinct_from_b
1	True	True	True	True	False	True
2	True	False	False	True	True	False
3	False	True	False	True	True	False
4	False	False	False	False	False	True
5	True	NULL	NULL	True	True	False
6	False	NULL	False	NULL	True	False
7	NULL	True	NULL	True	True	False
8	NULL	False	False	NULL	True	False
9	NULL	NULL	NULL	NULL	False	True

Example

- SubSelect
  - With NOT IN for status attribute

```
…
AND dim_zeit NOT IN (SELECT …
                       FROM tab
                      WHERE status IN ('a', 'b', 'c')
```

- Gets converted to a Join

```
…
 LEFT JOIN tab ON dim_zeit = …
WHERE status NOT IN ('a', 'b', 'c')
```

Example

- Resultset is not identical
- LEFT JOIN leaves option of a NON-match
- So NULL options needs to be added

```
…
 LEFT JOIN tab ON dim_zeit = …
WHERE (status NOT IN ('a', 'b', 'c‘)
       OR status IS NULL)
```

Additional Information

- IDUG 2016: Good Nulls, Bad Nulls by John Maenpaa
- IDUG NA 2022: Harnessing The Power of OLAP Functions by Calisto Zuzarte

Thanks

Michael Tiefenbacher

Principal Consultant

Email: michael.tiefenbacher@mip.de

Twitter: @globomike

mip
A CENIT COMPANY

IBM GOLD Consultant

IBM Champion 2022