ViennaDB Meetup

Exasol Database
Why Is It Our Number One Choice for Analytics Projects?

Dr. Thomas Petrik, 7. April 2025

sphinx

About Sphinx

Joy, Performance & Passion for Perfection

- **Founded:**
  1994 Sphinx IT-Consulting GmbH
  2003 Sphinx Managed Services GmbH
  owner-managed
- **Head Office:** Vienna, Austria
- **What we do:** Realising ground-breaking & sustainably effective IT solutions in an astonishingly short time.
- **Our Philosophy:** Holistic consulting, customized solutions, appreciation of existing systems & for the people behind them.
- **Business Areas:**
  - Analytics & Business Intelligence
  - Infrastructure & Database Technology
  - IT- & Data Security
  - IT- & Process Automation
  - AI in Business Solutions
  - Strategic Consulting & External IT



Exasol in a minute

Exasol

MADE IN GERMANY

The Worlds Fastest Analytical Database

for secure, time critical analytics supporting data services/products

- More than 20 years of experience in the field
- 200+ employees
- Public company since 2020
- Columnar analytics
- Low TCO
- Deploy anywhere (on-prem & cloud)
- ML at scale



200+

PROVEN SUCCESS
Customers

100+

PARTNERSHIPS
Technology or consultancy partners

30+ COUNTRIES
Worldwide presence: North America, EMEA and Asia

DELL

Raiffeisenlandesbank Niederösterreich-Wien

SONY MUSIC

badoo

XING

Mercedes-Benz

CINEPLEXX GROSSES KINO

bank 99

Piedmont HEALTHCARE

OTTO

T-Mobile

ÖAMTC

ZGONC

vodafone

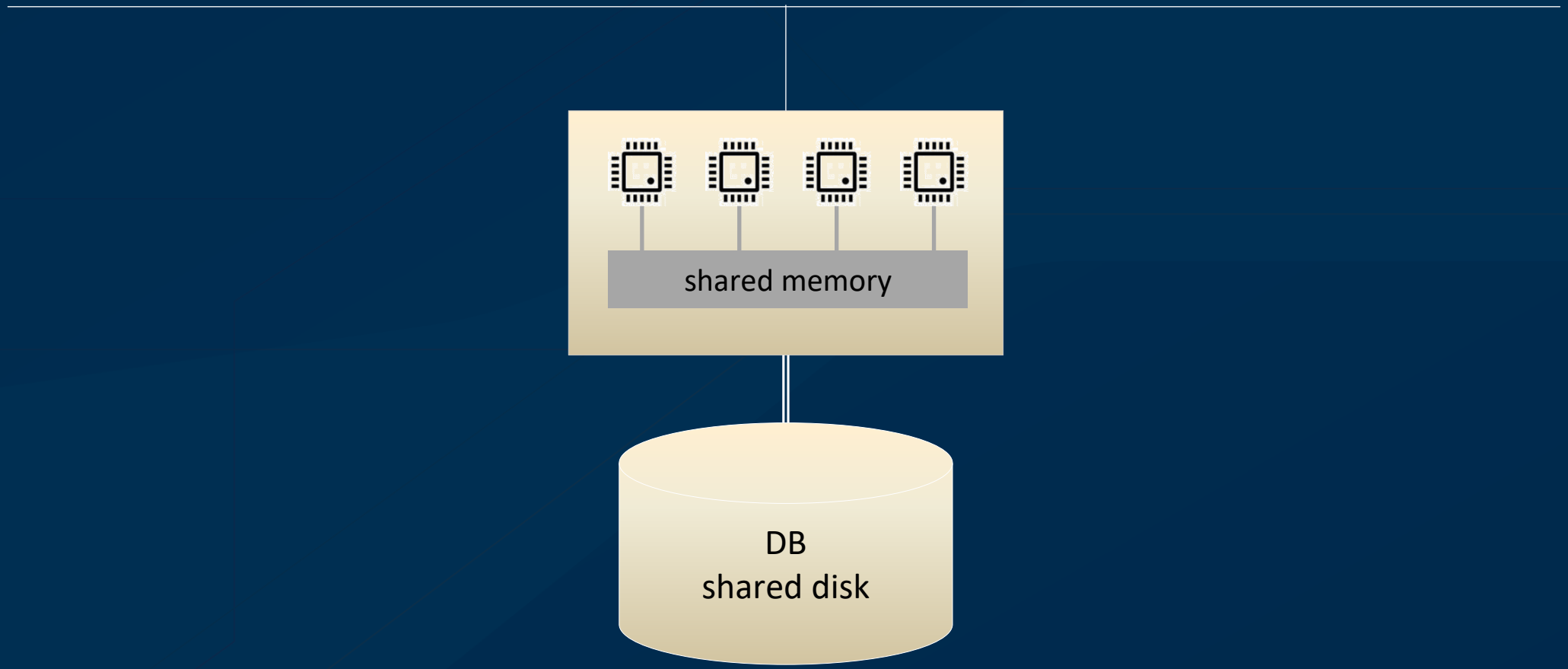
MONSOON

Architecture:

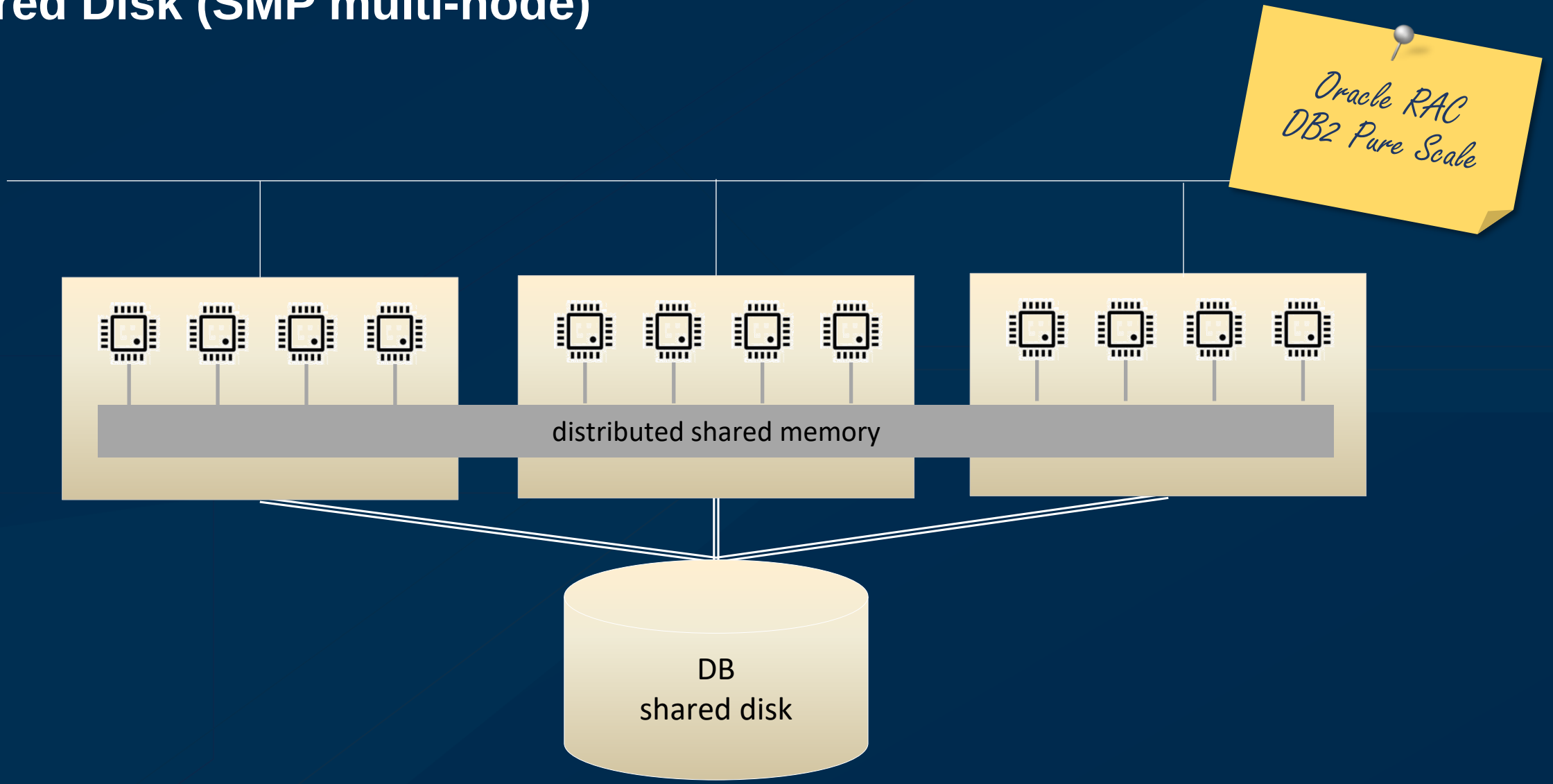
SMP vs. MPP



Shared Everything (SMP)

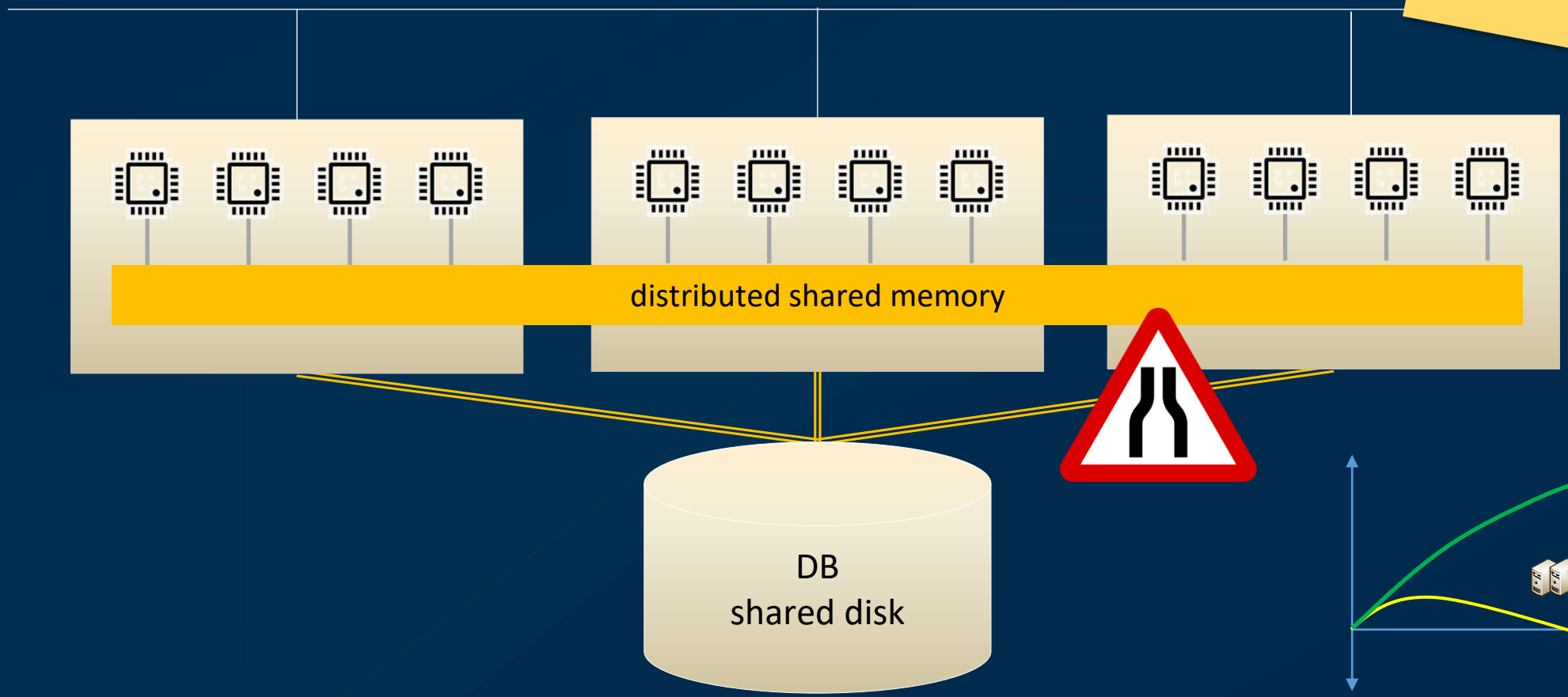


Shared Disk (SMP multi-node)

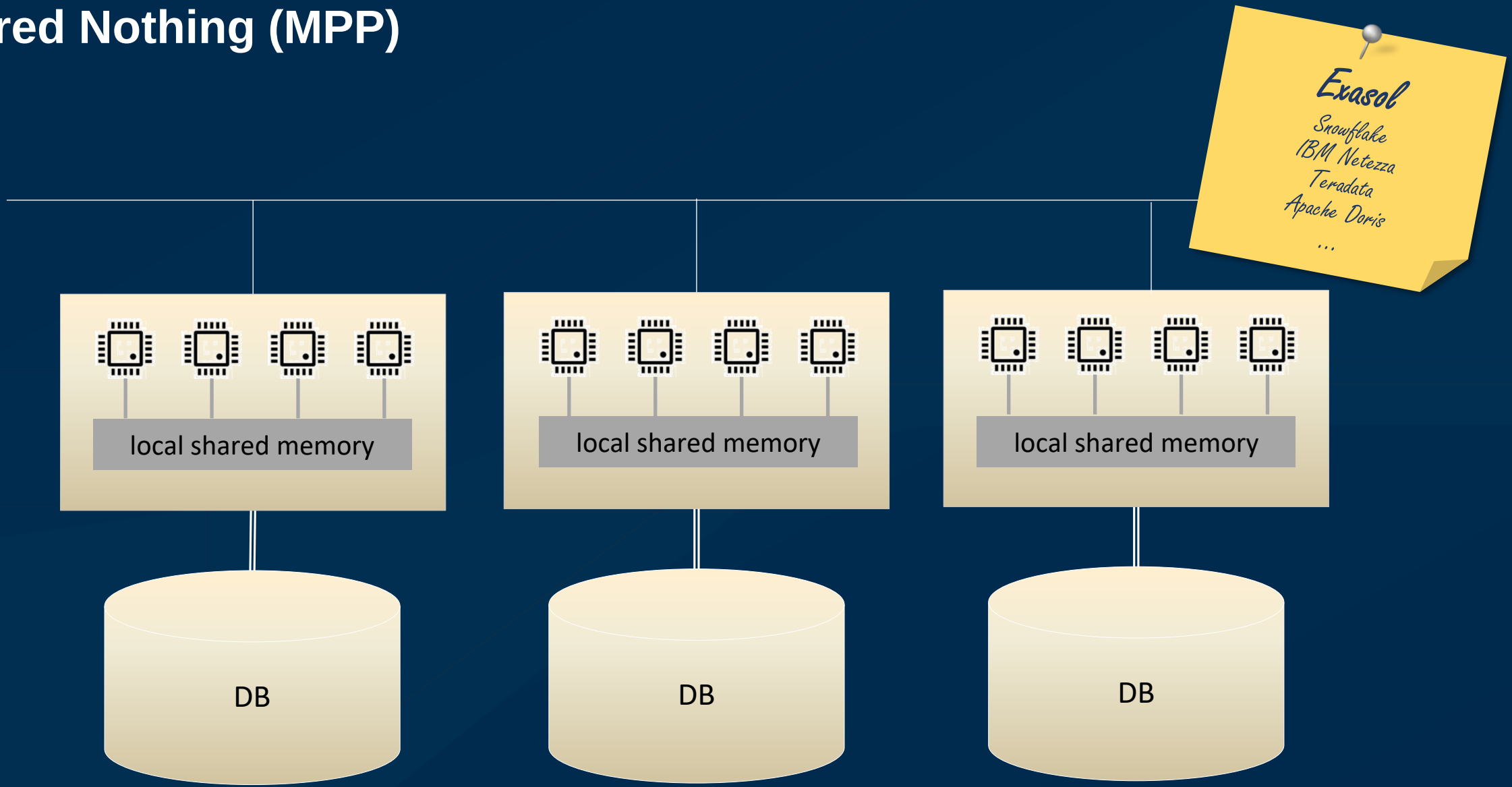


Shared Disk (SMP multi-node)

Oracle RAC
DB2 Pure Scale



Shared Nothing (MPP)



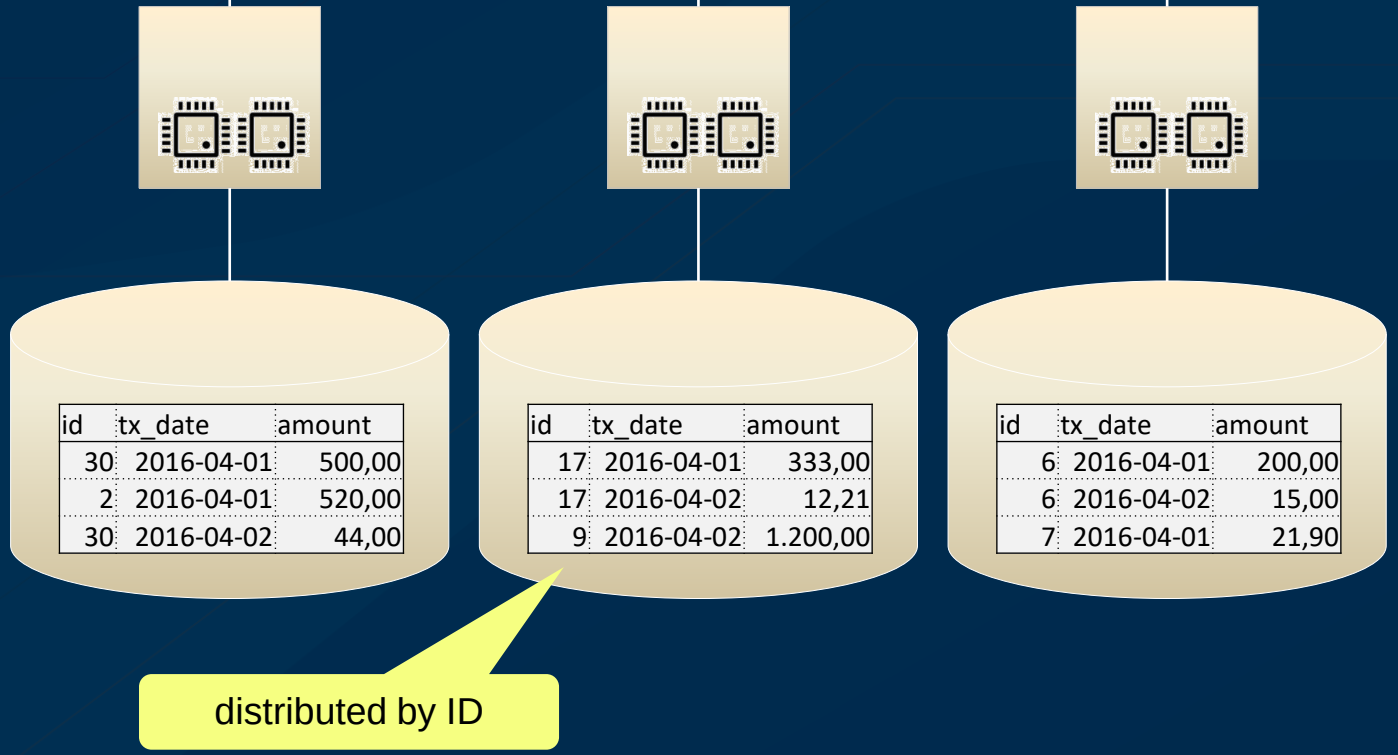
Architecture:

MPP at work …



MPP: Data Distribution

```
select sum(amount) from sales
where tx_date = to_date('2016-04-01','yyyy-mm-dd');
```



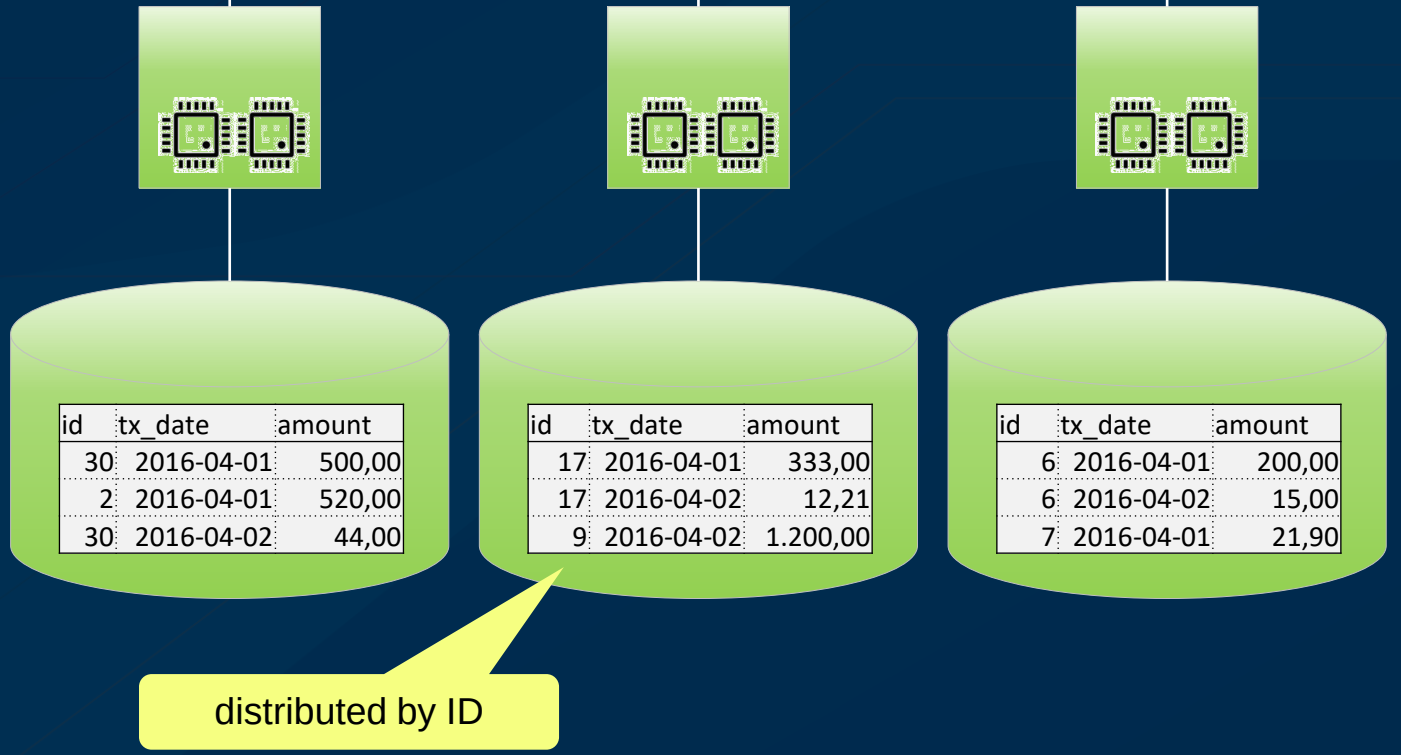
id	tx_date	amount
30	2016-04-01	500,00
2	2016-04-01	520,00
30	2016-04-02	44,00

id	tx_date	amount
17	2016-04-01	333,00
17	2016-04-02	12,21
9	2016-04-02	1.200,00

id	tx_date	amount
6	2016-04-01	200,00
6	2016-04-02	15,00
7	2016-04-01	21,90

MPP: Data Distribution

```
select sum(amount) from sales
where tx_date = to_date('2016-04-01','yyyy-mm-dd');
```



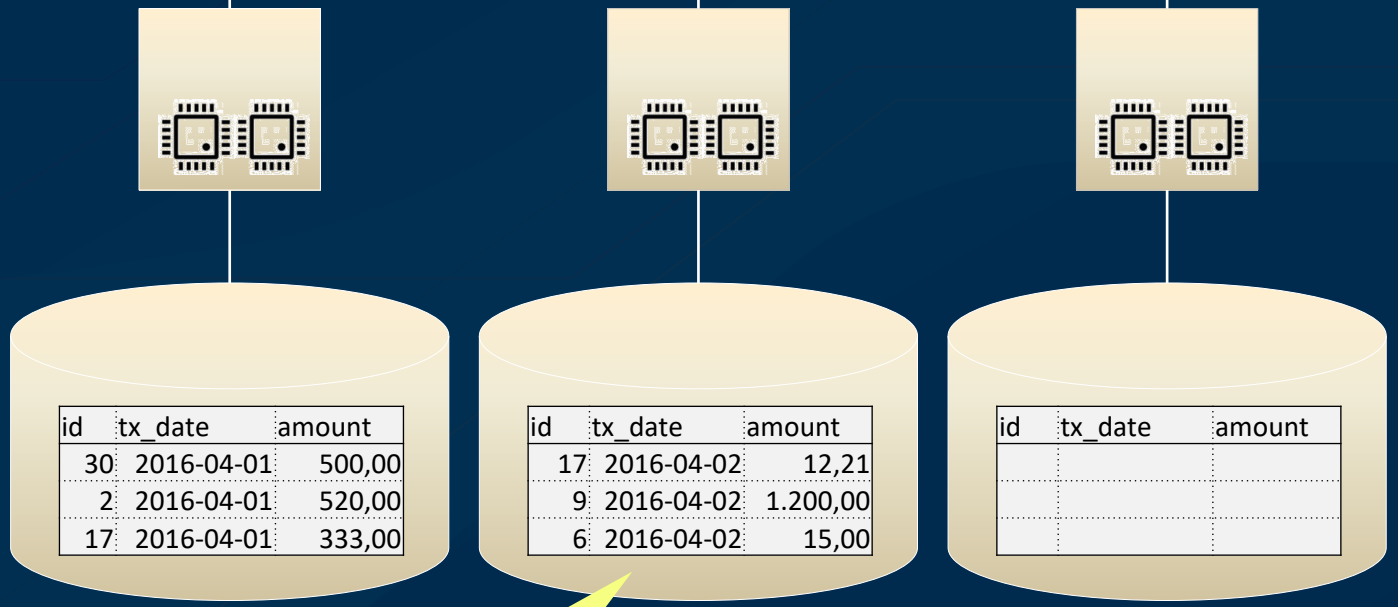
id	tx_date	amount
30	2016-04-01	500,00
2	2016-04-01	520,00
30	2016-04-02	44,00

id	tx_date	amount
17	2016-04-01	333,00
17	2016-04-02	12,21
9	2016-04-02	1.200,00

id	tx_date	amount
6	2016-04-01	200,00
6	2016-04-02	15,00
7	2016-04-01	21,90

MPP: Data Distribution

```
select sum(amount) from sales
where tx_date = to_date('2016-04-01','yyyy-mm-dd');
```



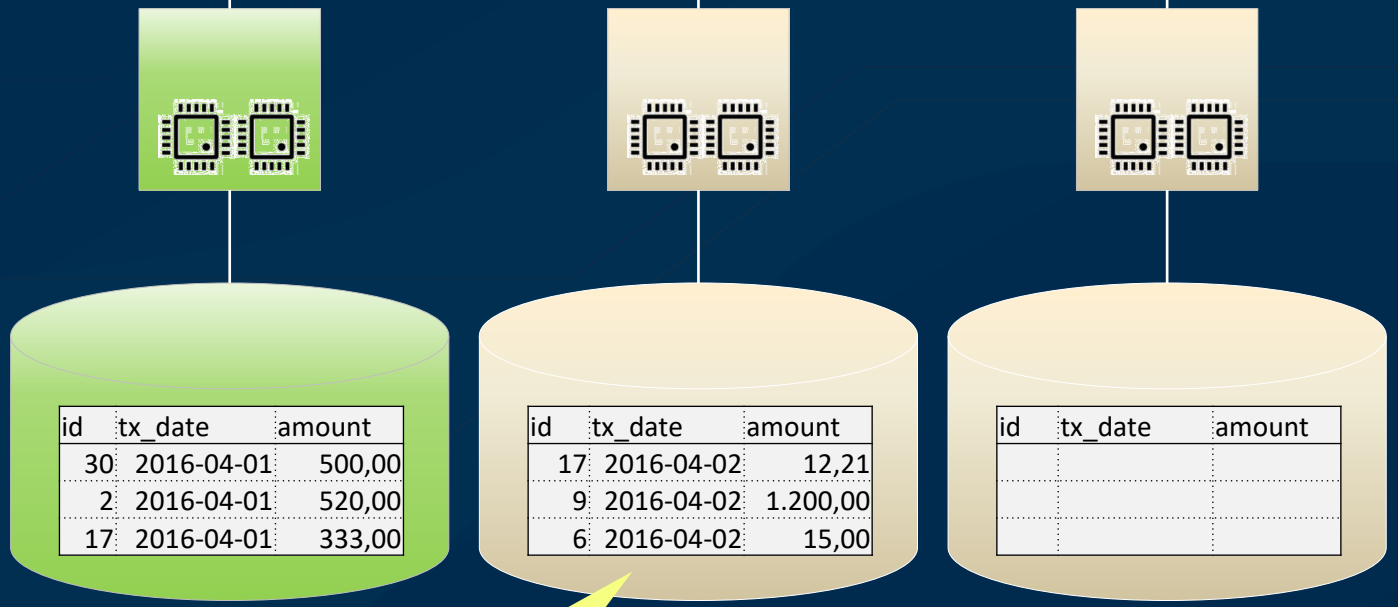
id	tx_date	amount
30	2016-04-01	500,00
2	2016-04-01	520,00
17	2016-04-01	333,00

id	tx_date	amount
17	2016-04-02	12,21
9	2016-04-02	1.200,00
6	2016-04-02	15,00

id	tx_date	amount

MPP: Data Distribution

```
select sum(amount) from sales
where tx_date = to_date('2016-04-01','yyyy-mm-dd');
```



id	tx_date	amount
30	2016-04-01	500,00
2	2016-04-01	520,00
17	2016-04-01	333,00

id	tx_date	amount
17	2016-04-02	12,21
9	2016-04-02	1.200,00
6	2016-04-02	15,00

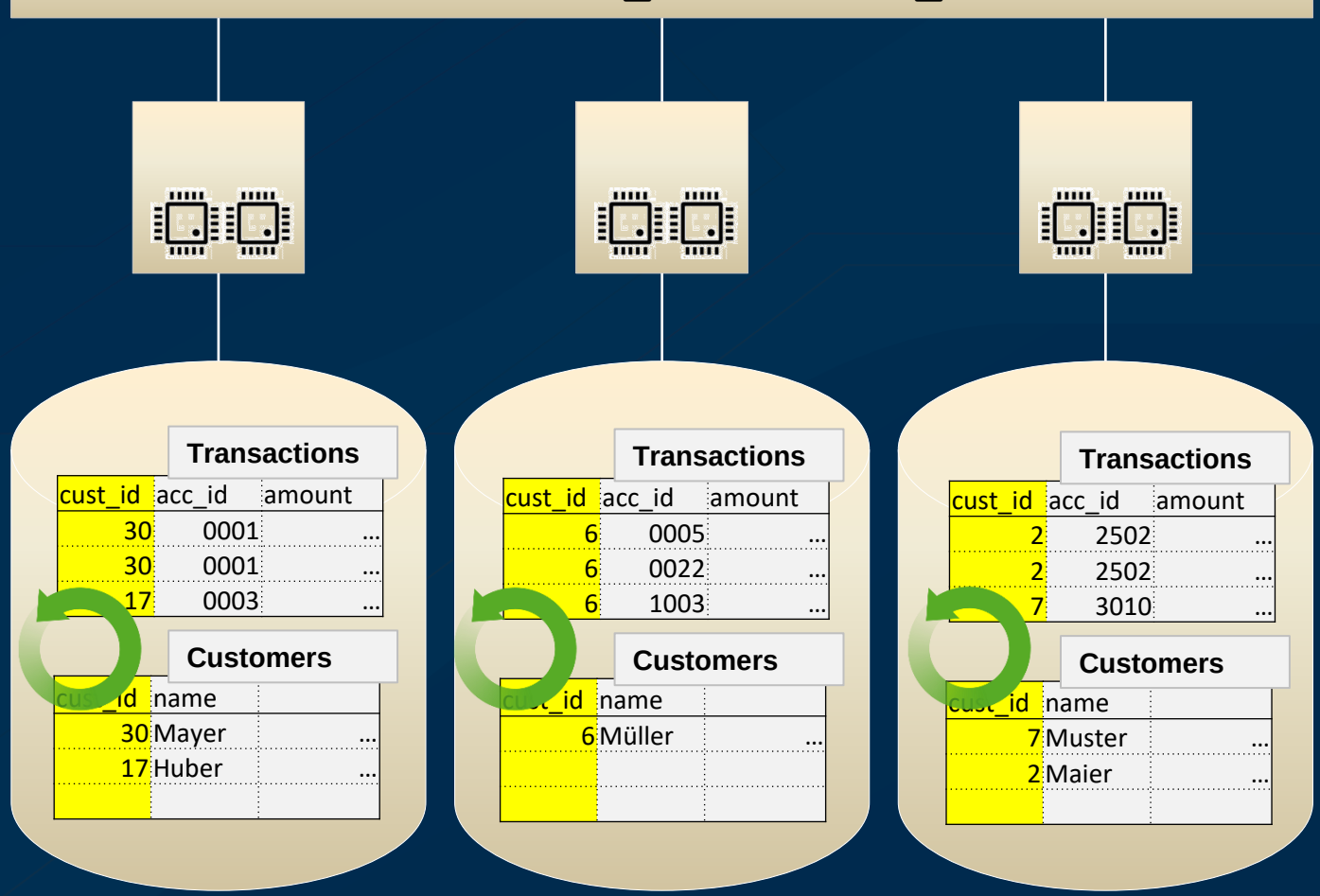
id	tx_date	amount



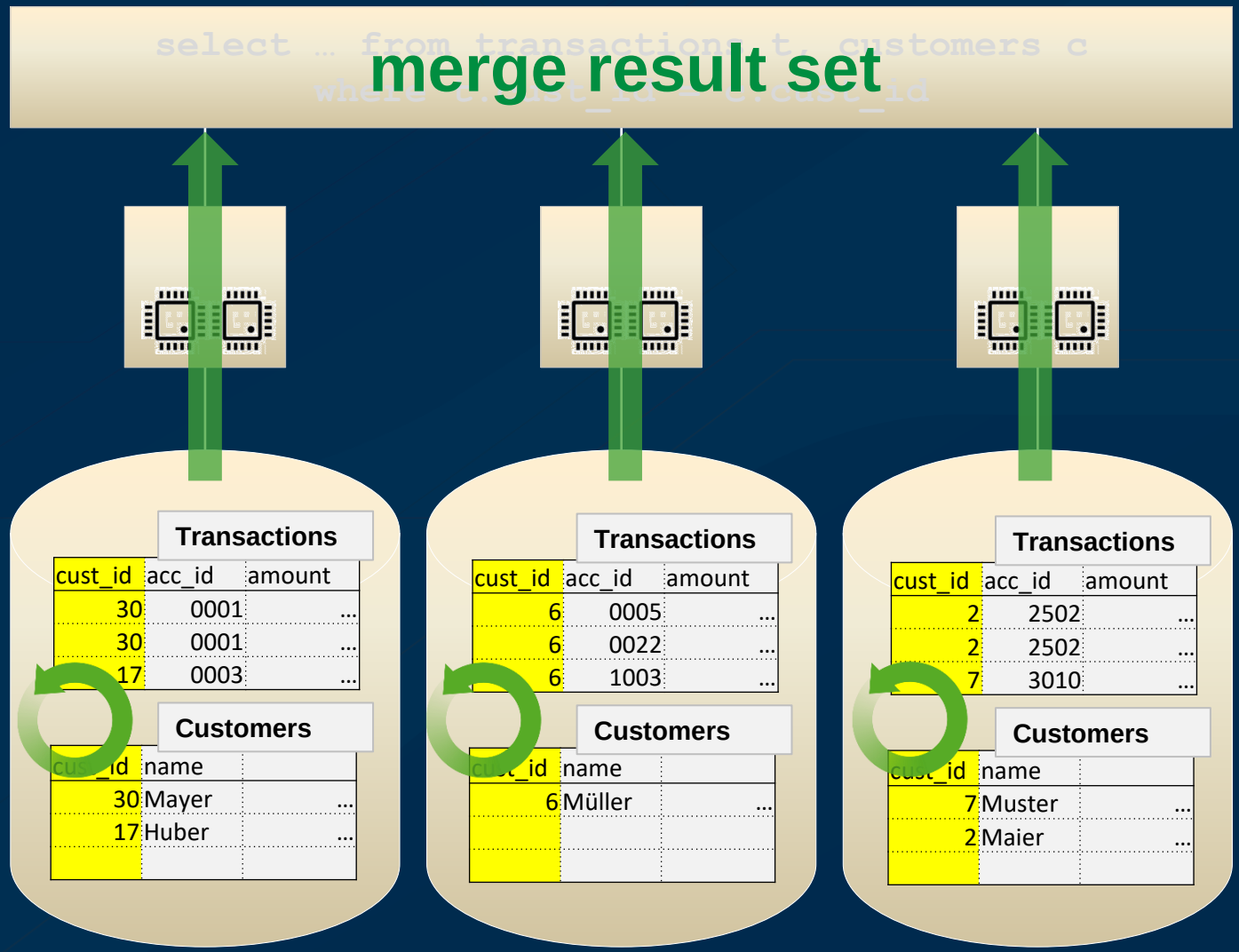
distributed by tx_date

MPP: Local Joins

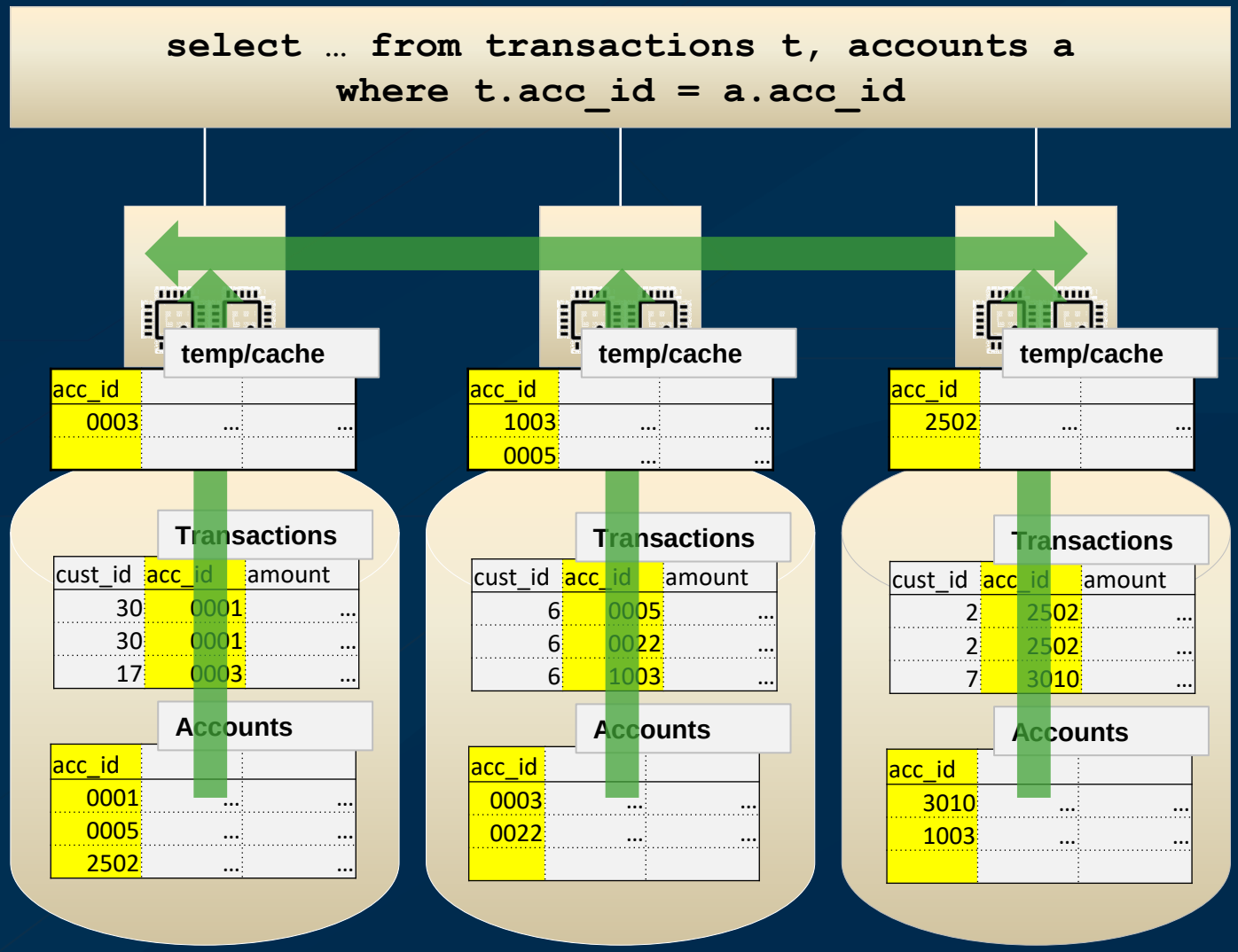
```
select … from transactions t, customers c
where t.cust_id = c.cust_id
```



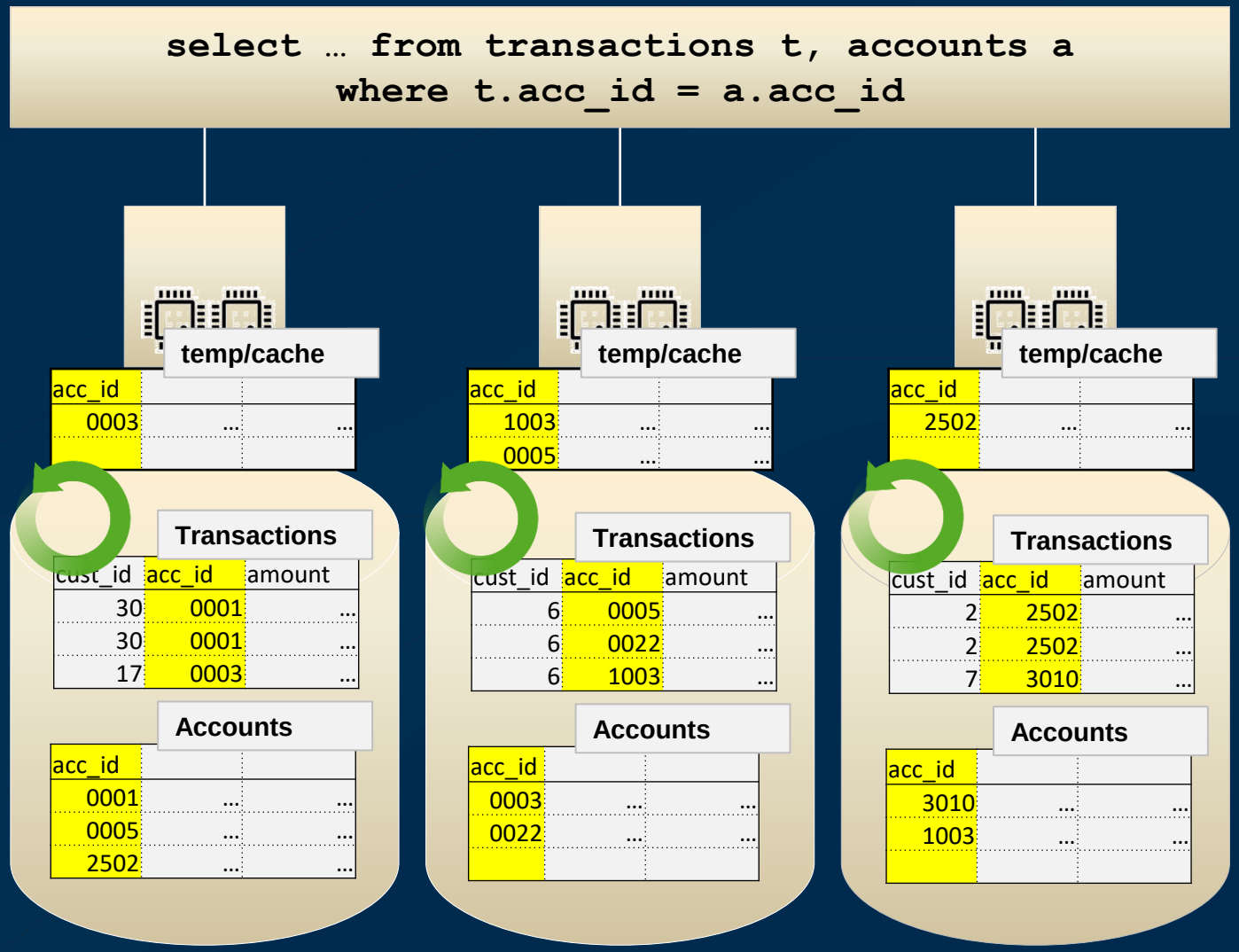
MPP: Local Joins



MPP: Global Joins / Redistribution



MPP: Global Joins / Redistribution



ACID Without Any OLTP Overhead …

minimum locking
table locks – no row locks

compression

column oriented design

small table redundancy
replication instead of broadcast

no redo-/archive Logs

update = delete + insert

no undo-segments

MPP Use Cases

best choice for ...

- DWH
- every kind of reporting (ad-hoc and batch)
- analytics
- ad-hoc load
- an accelerator (data mart)
- small up to extremely large data volumes

do not use for ...

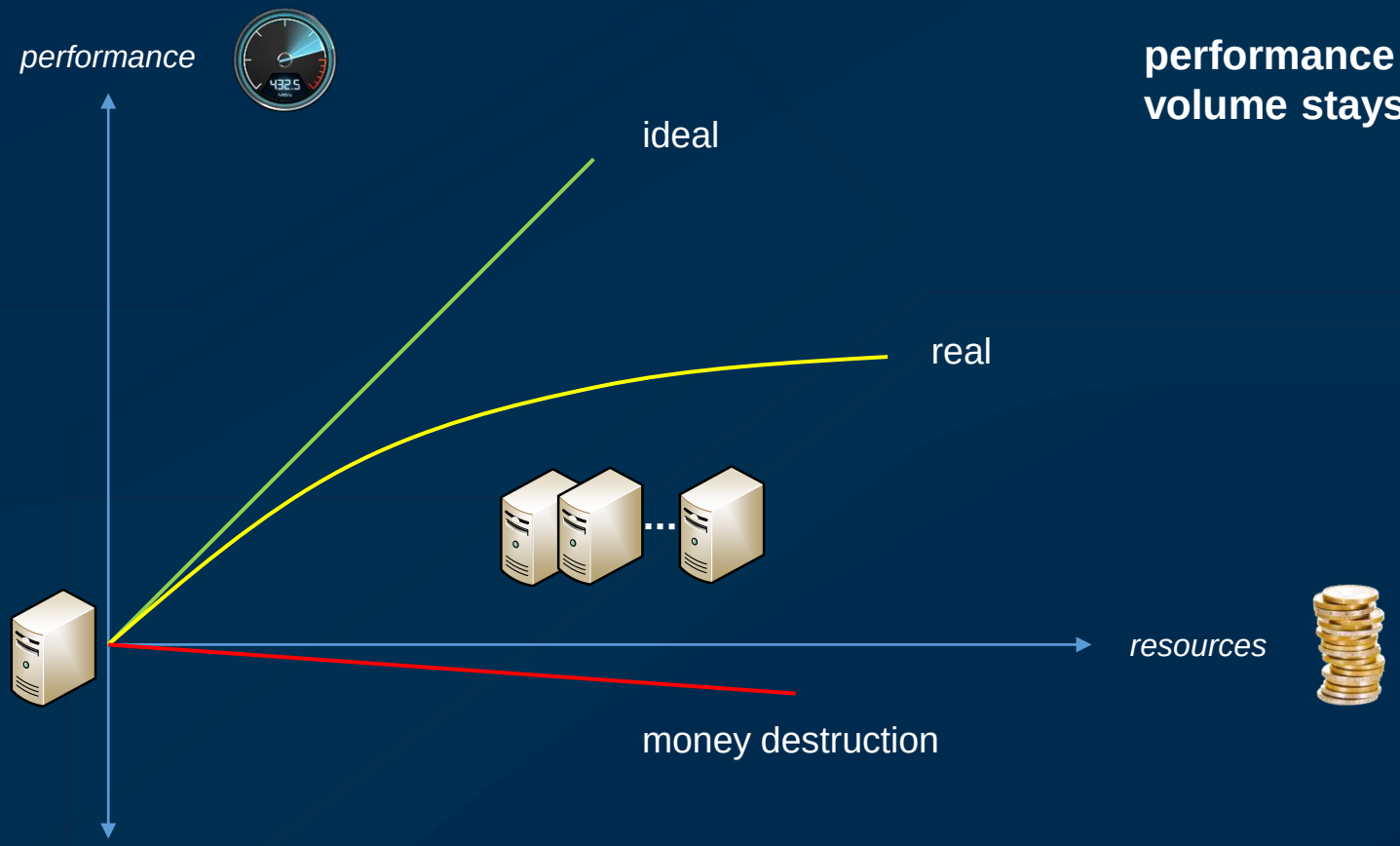
- OLTP
- single row processing

Architecture:

Scalability



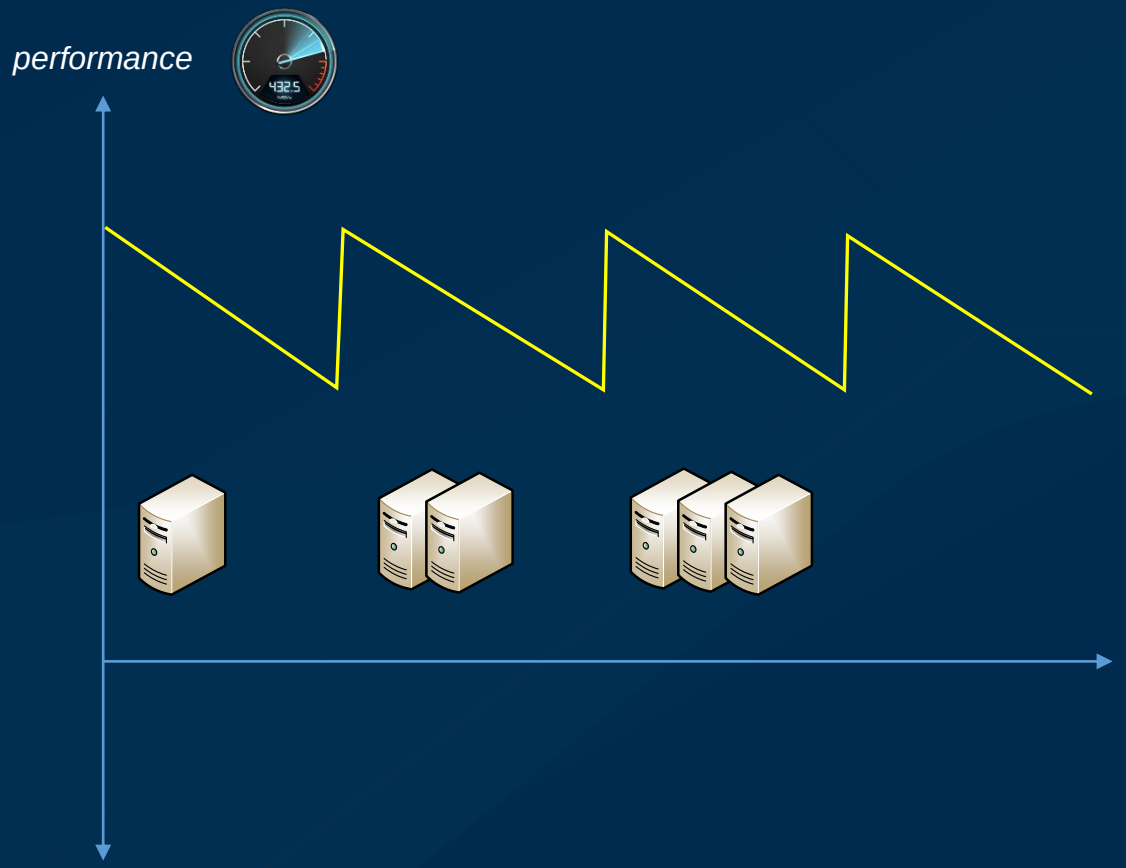
Scalability



Goal:

performance increase while data volume stays constant

Scalability



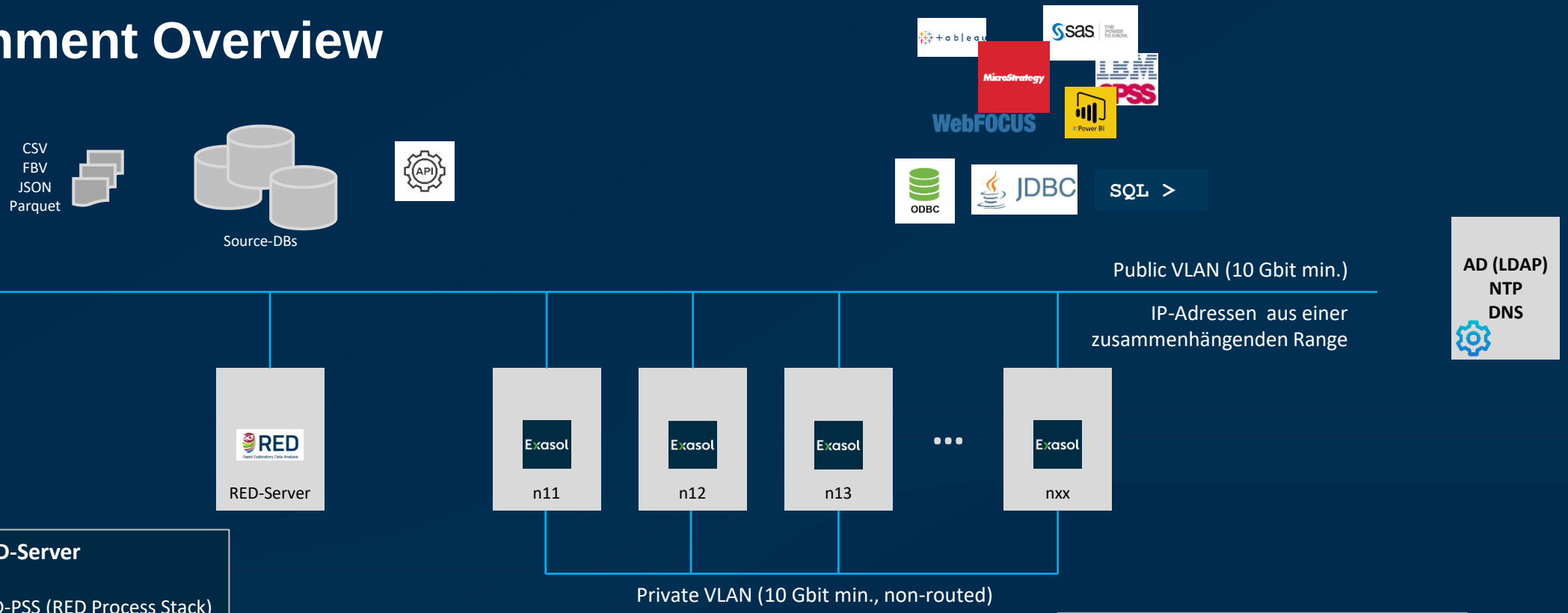
Goal:

Keep average performance while data volume is growing (scale-out)



Exasol:
Environment Management

Environment Overview



RED-Server

RED-PSS (RED Process Stack)
File Landing Zone

x86 64-Bit
RHEL 8

Basis Hardware

x86 64-Bit (SSSE3 featured)
AMD oder Intel
1-4 Sockel

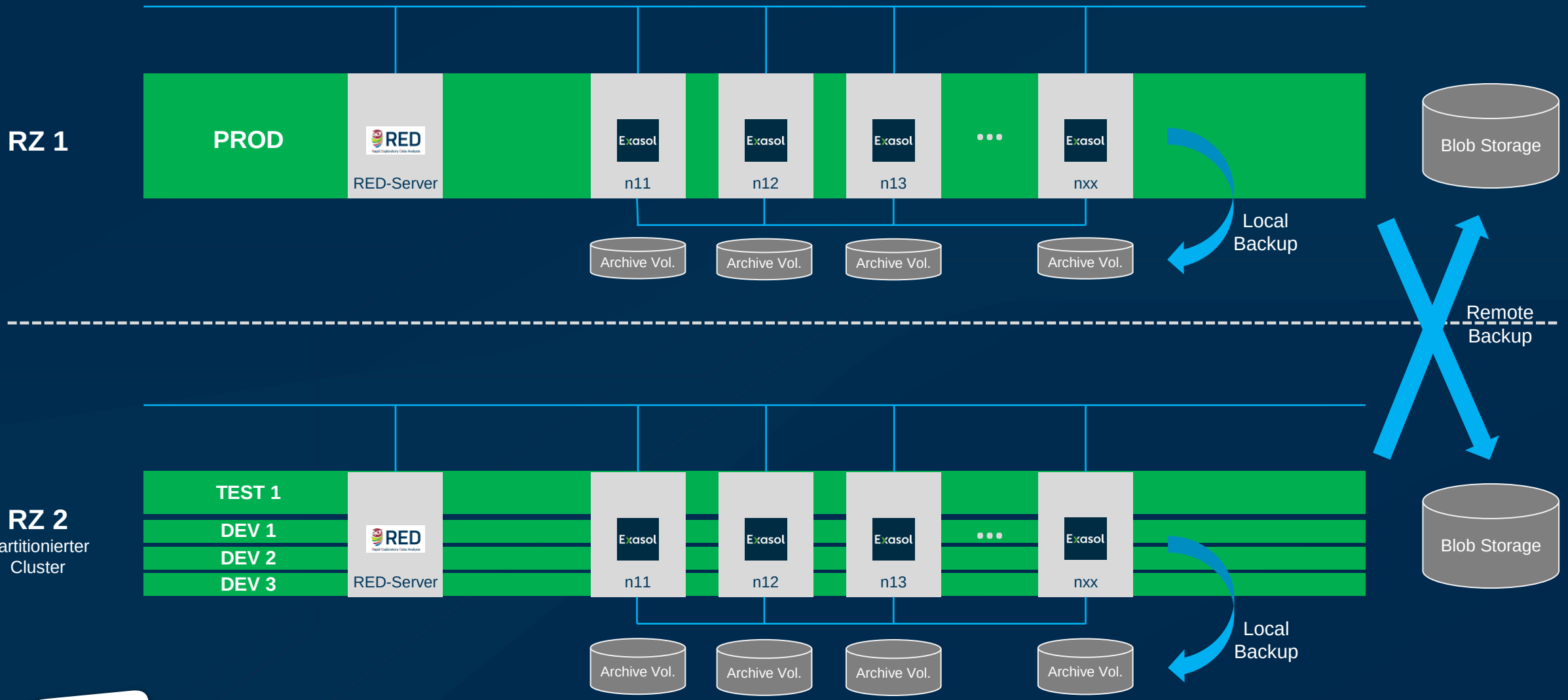
Storage:
Shared Nothing Architektur: KEIN Shared Storage
NVMe, SSD, HDD mit mind. 250 MBps (r/w)
local / SAN / iSCSI
Redundanz: RAID 1, 5 oder 6

Knoten (VMs)

Anzahl: 1 (Single Node) bis mehrere tausend
CPU / RAM: NICHT überprovisioniert

RHEL 8 oder 9
Ubuntu 20.04 oder 22.04 LTS

Environment Management

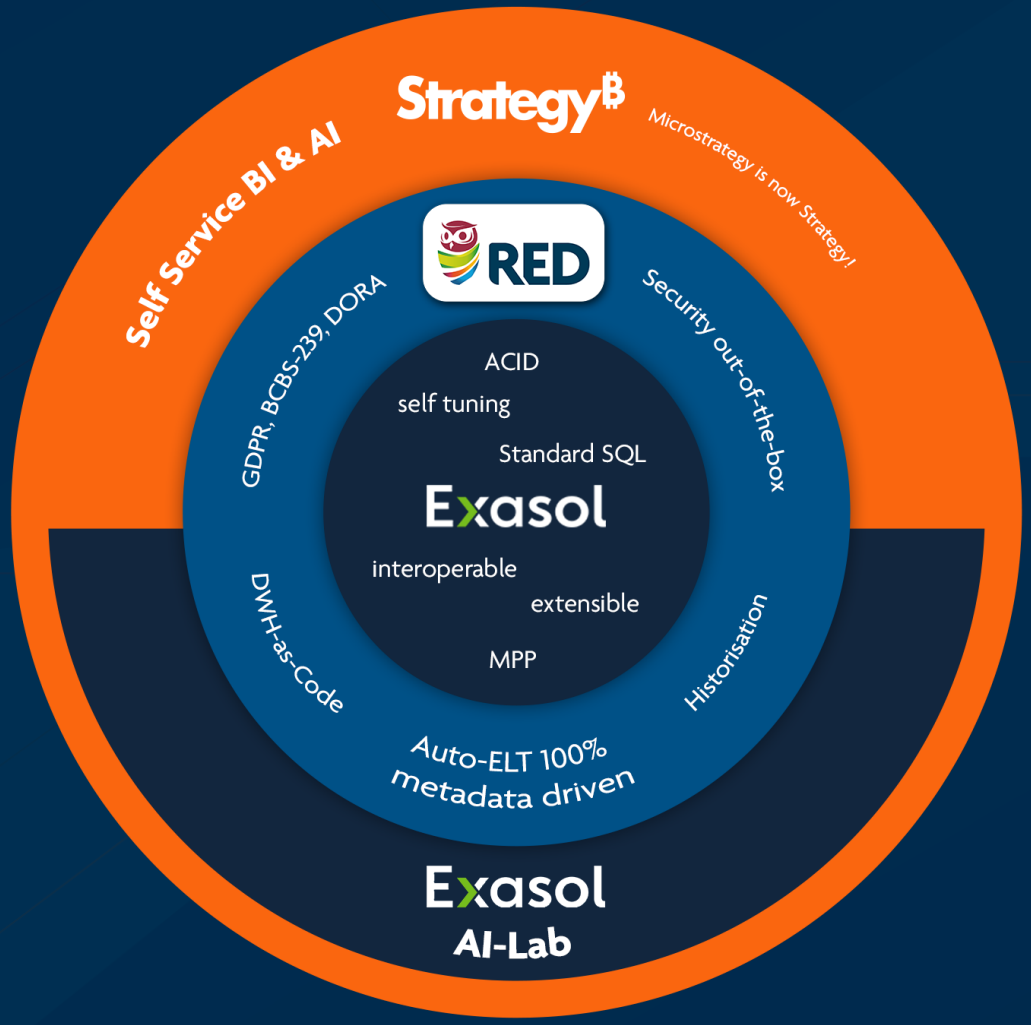


Analytics @Sphinx: Integrated Solution Stack

Exasol – Sphinx - Strategy

SQL-only Architecture

No precedural ELT
insert/select + merge



DWH-as-Code

Infrastructure
Data extraction (ELT)
Historization
Consolidation
Security



Historisation

An example for the SQL-only paradigm

Historisation – what for?

100% reproducable Reporting

business point of view

On which business date was the row valid?

Business Day Logic

Business Date (BDate)

bi-temporal
2-dimensional

technical point of view

When did the row arrive in the DWH?

systimestamp

Technical Date (TDate)

Master Data & Event Data

Master Data

- valid for a period of time
- e.g. account data, customer data

KundenNr	Name	BDate
123	Müller	2023-11-21
123	Müller	2023-11-22
123	Mayer	2023-11-23

KundenNr	Name	BDate_From	BDate_To
123	Müller	2023-11-21	2023-11-22
123	Mayer	2023-11-23	9999-12-31

business-related:
extend valifity to a period of time

technical:
deduplication

Event Data

- valid only at a specific point in time
- z.B. transactions, purchases
- BDate results from the transaction date

RE_Nr	Name	RE_Datum	BDate
100	Popcorn	2023-11-21 17:21	2023-11-21
100	Nachos	2023-11-21 17:21	2023-11-21
...	...	...	...
187	Cola	2023-11-23 20:11	2023-11-23

The Business (Surrogate) Key

Business Key (BK)

Business Surrogate Key
MD5 Hash of BK

BLZ	AccountNr	Opening_Date	...	BSK
32105	0101246	2008-04-30	...	99f7558a58ace36441e256b6ddecbb94
34200	0100222	1999-05-17	...	abf7dc25b0cf875d218646e025ba4891

- BSK is the basis for historisation
  - monitor the changes over time for the same business key
- BK mostly results from the primary key (PK) in the source
  - but: not always!
  - sometimes no PK is defined
  - PK in the source may contain time dependend information
    - especially if the source is already somehow historised itself

Technical Historisation

Source

CustNr	Nam	BDate
123	Müller	2023-11-15

CustNr	Nam	BDate
123	Müller	2023-11-16

no change
row ignored!

CustNr	Nam	BDate
123	Mayer	2023-11-17

Correction

CustNr	Nam	BDate
123	Meier	2023-11-17

DWH

CustNr	Nam	BDate_From	BDate_To	TDate_From	TDate_To
123	Müller	2023-11-15	9999-12-31	2023-11-15 23:15:41	9999-12-31

CustNr	Nam	BDate_From	BDate_To	TDate_From	TDate_To
123	Müller	2023-11-15	9999-12-31	2023-11-15 23:15:41	**2023-11-17 23:16:00**
123	**Müller**	**2023-11-15**	**2023-11-16**	**2023-11-17 23:16:00**	**9999-12-31**
123	**Mayer**	**2023-11-17**	**9999-12-31**	**2023-11-17 23:16:00**	**9999-12-31**

CustNr	Nam	BDate_From	BDate_To	TDate_From	TDate_To
123	Müller	2023-11-15	9999-12-31	2023-11-15 23:15:41	2023-11-17 23:16:00
123	Müller	2023-11-15	2023-11-16	2023-11-17 23:16:00	9999-12-31
123	Mayer	2023-11-17	9999-12-31	2023-11-17 23:16:00	**2023-11-18 08:00:00**
123	**Meier**	**2023-11-17**	**9999-12-31**	**2023-11-18 08:00:00**	**9999-12-31**

Technical Historisation

Ask for the status as of the booking date 16.11., which was valid at 8:30 am on 17.11.

```
SELECT *
  FROM kunde
 WHERE      '2023-11-16' BETWEEN bdate_from AND bdate_to
       AND '2023-11-17 08:30:00' BETWEEN tdate_from and tdate_to;
```



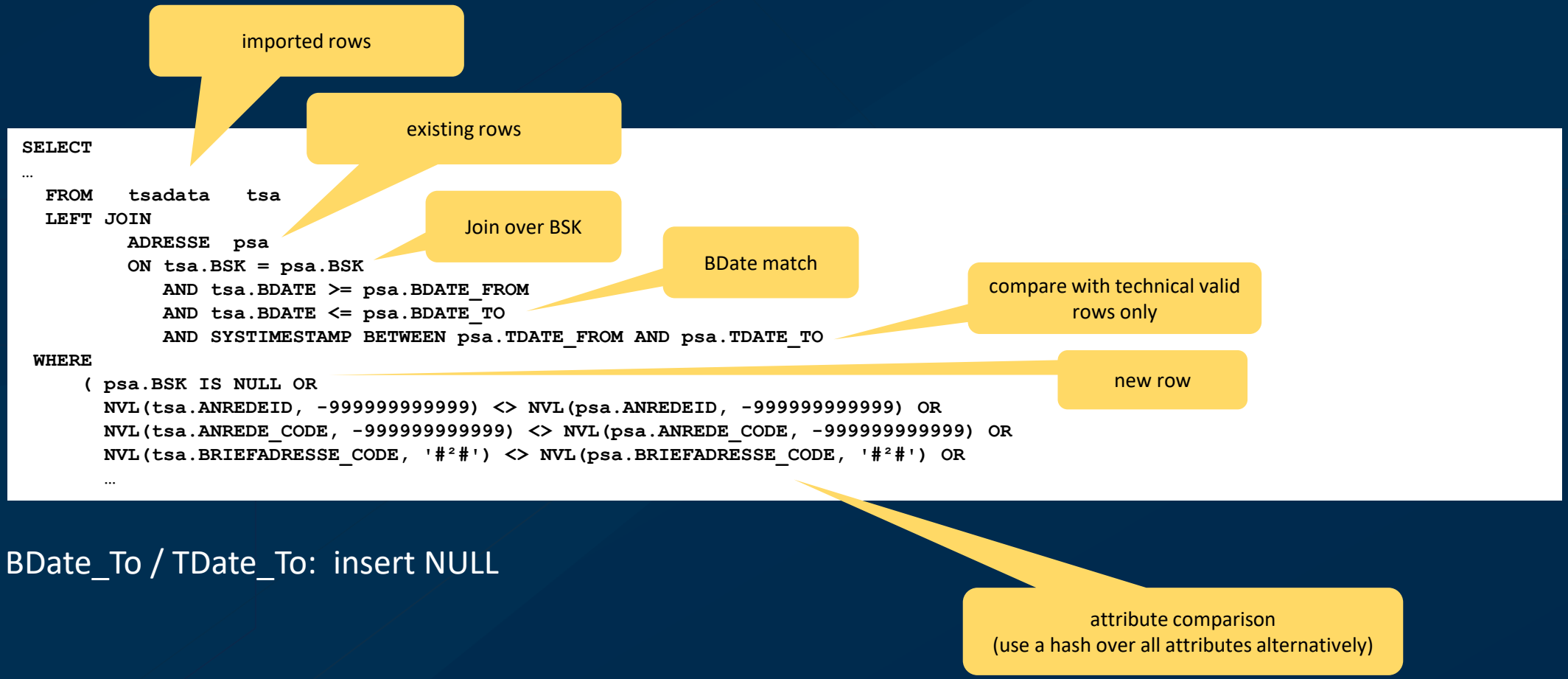
CustNr	Nam	BDate_From	BDate_To	TDate_From	Tdate_To
123	**Müller**	**2023-11-15**	**9999-12-31**	**2023-11-15 23:15:41**	**2023-11-17 23:16:00**
123	Müller	2023-11-15	2023-11-16	2023-11-17 23:16:00	9999-12-31
123	Mayer	2023-11-17	9999-12-31	2023-11-17 23:16:00	2023-11-18 08:00:00
123	Meier	2023-11-17	9999-12-31	2023-11-18 08:00:00	9999-12-31

100% reproducable



Historisation: SQL-only

Step 1: Change new and updated Rows (insert / select)




```
SELECT
…
  FROM   tsadata   tsa
  LEFT JOIN
         ADRESSE  psa
         ON tsa.BSK = psa.BSK
            AND tsa.BDATE >= psa.BDATE_FROM
            AND tsa.BDATE <= psa.BDATE_TO
            AND SYSTIMESTAMP BETWEEN psa.TDATE_FROM AND psa.TDATE_TO
 WHERE
     ( psa.BSK IS NULL OR
       NVL(tsa.ANREDEID, -999999999999) <> NVL(psa.ANREDEID, -999999999999) OR
       NVL(tsa.ANREDE_CODE, -999999999999) <> NVL(psa.ANREDE_CODE, -999999999999) OR
       NVL(tsa.BRIEFADRESSE_CODE, '#²#') <> NVL(psa.BRIEFADRESSE_CODE, '#²#') OR
       …
```

BDate_To / TDate_To: insert NULL

sphinx

Historisation: SQL-only

Step 2: Close time slices (BDate_To, TDate_To – merge)

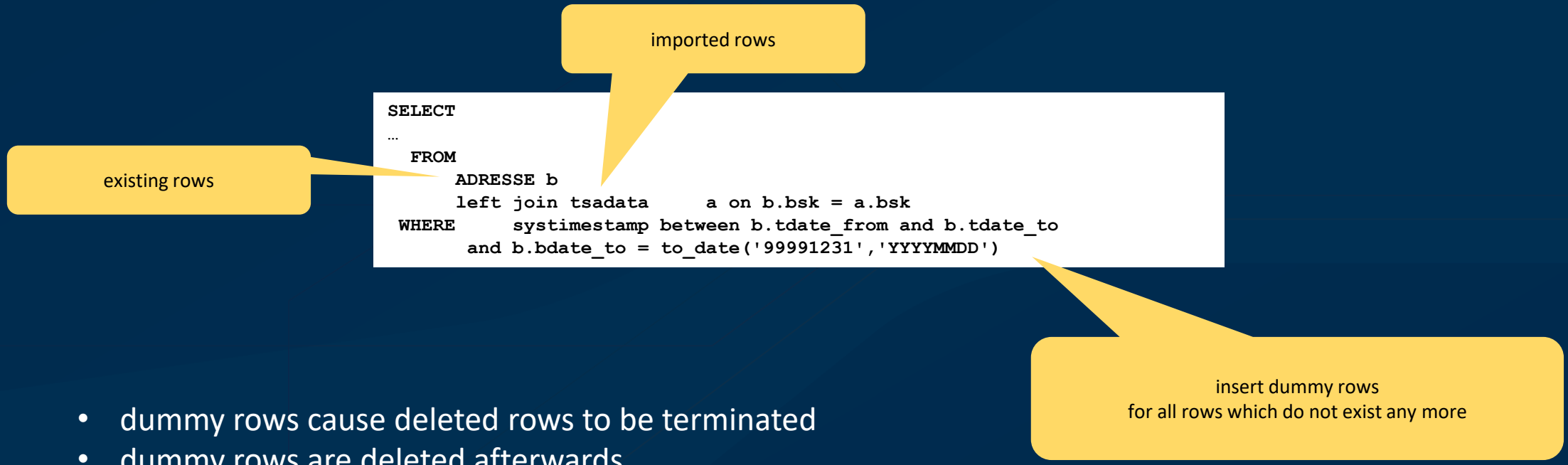
```
SELECT  BDATE_FROM
       ,BDATE_TO_OLD
       ,TDATE_FROM
       ,TDATE_TO_OLD
       ,CASE
           WHEN TDATE_FROM_NEXT IS NULL THEN TO_TIMESTAMP('99991231', 'YYYYMMDD')
           ELSE ADD_SECONDS(TDATE_FROM_NEXT, - 0.001)
       END AS TDATE_TO
       ,CASE
           WHEN LOCAL.TDATE_TO <> TDATE_TO_OLD THEN BDATE_TO_OLD
           WHEN BIS_BDATE_FROM = BDATE_FROM_NEXT
               THEN
                  CASE
                       WHEN MAX(BDATE_FROM_NEXT) over(partition BY BSK, BDATE_FROM) = BDATE_FROM
                           THEN to_date('99991231', 'YYYYMMDD')
                       ELSE add_days(MAX(BDATE_FROM_NEXT) over(partition BY BSK, BDATE_FROM), -1)
                   END
           WHEN BDATE_FROM_NEXT IS NULL
               THEN to_date('99991231', 'YYYYMMDD')
           ELSE add_days(BDATE_FROM_NEXT, -1)
       END AS BIS_BDATE_TO
FROM (
      SELECT  BDATE_FROM                                                          AS BDATE_FROM
             ,BDATE_TO                                                            AS BDATE_TO_OLD
             ,BSK                                                                 AS BSK
             ,TDATE_FROM                                                          AS TDATE_FROM
             ,TDATE_TO                                                            AS TDATE_TO_OLD
             ,LEAD(BDATE_FROM) OVER (PARTITION BY BSK ORDER BY BDATE_FROM, PK)    AS BDATE_FROM_NEXT
             ,LEAD(TDATE_FROM) OVER (PARTITION BY BSK, BDATE_FROM ORDER BY PK)    AS TDATE_FROM_NEXT
      FROM   ADRESSE
      WHERE  systimestamp BETWEEN TDATE_FROM AND NVL(TDATE_TO, systimestamp)
     )
```

close time slice if a successor exists

get successor

Historisation: SQL-only

Step 3: Terminate deleted rows



- dummy rows cause deleted rows to be terminated
- dummy rows are deleted afterwards
  - could also stay with an INACTIVE-flag
- not needed if the source itself delivers deleted rows (in case of a delta-load)



I have a question.

Sphinx IT Consulting GmbH

T +43 1 599 31 - 0
M +43 664 155 83
thomas.petrik@sphinx.at

Aspernbrückengasse 2
A-1020 Wien

www.sphinx.at