Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

SAP HANA Overview

This presentation is for internal use only and not intended for external distribution. Its contents must be held in confidence.

Lesson Objectives

- **On Completing this course, participants will be able to:**
  - Understand SAP HANA
  - Understand ABAP ON HANA
  - Understand SAP HANA System Architecture
  - To know SAP IN-Memory Strategy and Technology
  - Understand Row Store and Column Store

Contents

- SAP HANA System Landscape Connectivity Overview
- Technology Innovations and HANA Architecture
- In-Memory Architecture
- Parallel Processing
- Row Store, Column Store and Dictionary Compression
- ABAP Development shortcuts
- SAP HANA Coding Pattern

Contents

- Introduction to SAP HANA Studio
- SAP HANA Mandatory Adaptations
- SAP HANA Coding Pattern
- ABAP Development shortcuts

AB1011 - ABAP ON HANA

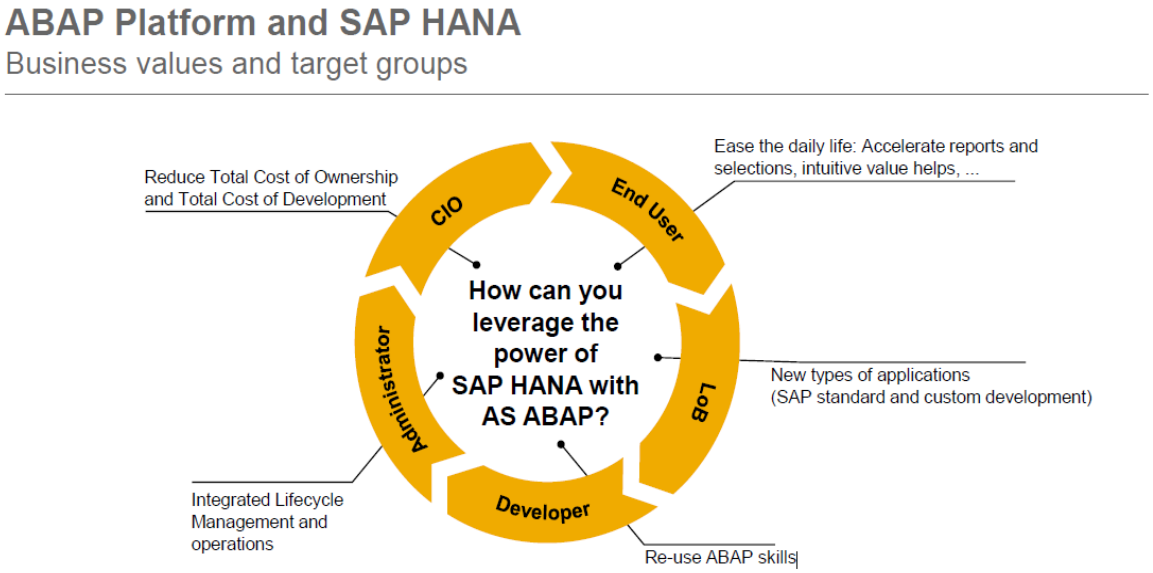
HANA – **H**igh-performance **AN**alytic **A**ppliance

HANA – High-performance ANalytic Appliance

SAP Business Suite Powered by SAP HANA

What makes HANA so Unique?

- 100% In-Memory computing (Allow OLTP & OLAP processing in Real-Time) Memory
- 5-20x Compression (Column based compression)
- Up to 10,000x Faster (Massive parallel scaling)



ABAP Platform and SAP HANA

Business values and target groups

How can you leverage the power of SAP HANA with AS ABAP?

CIO — Reduce Total Cost of Ownership and Total Cost of Development

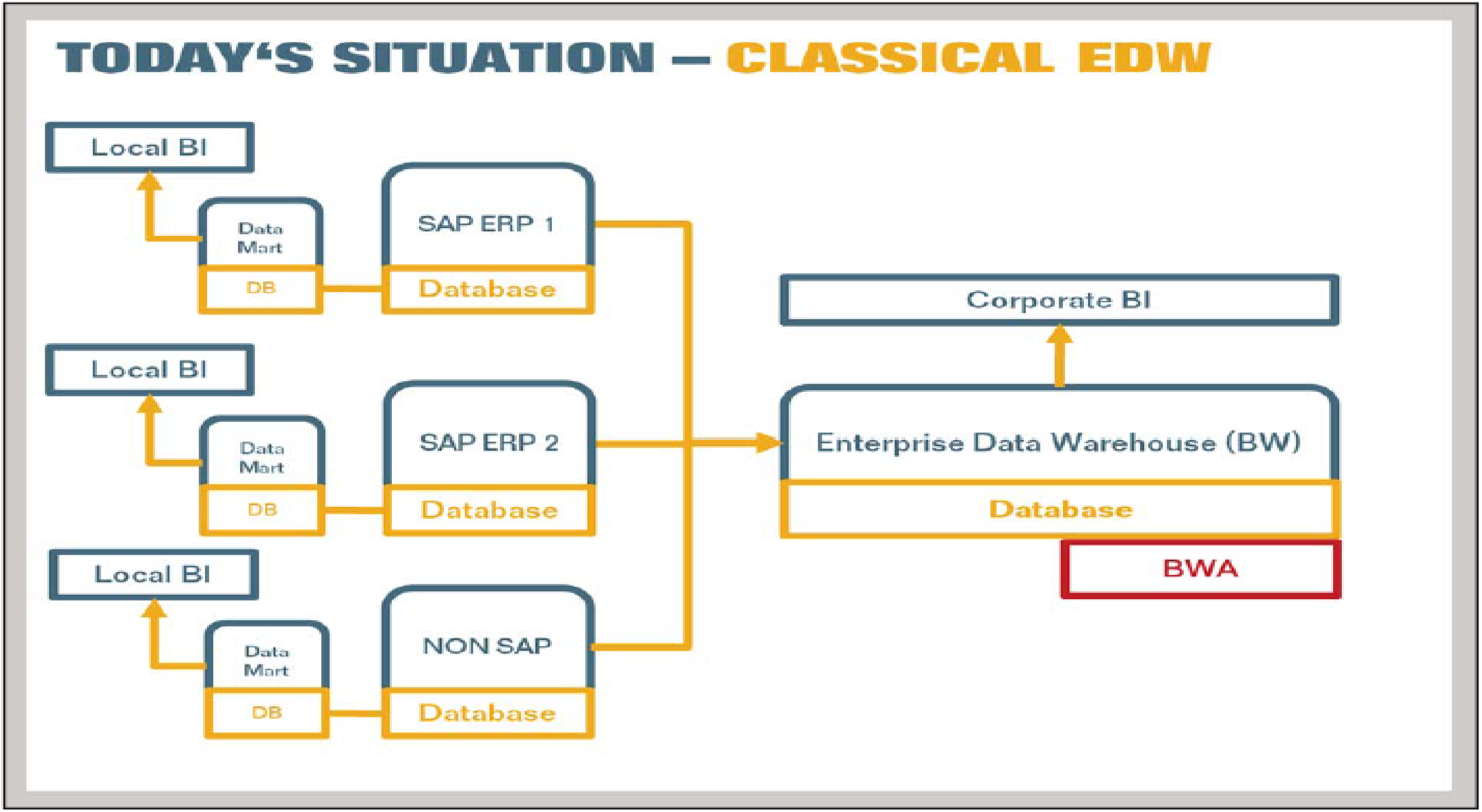
End User — Ease the daily life: Accelerate reports and selections, intuitive value helps, ...

LoB — New types of applications (SAP standard and custom development)

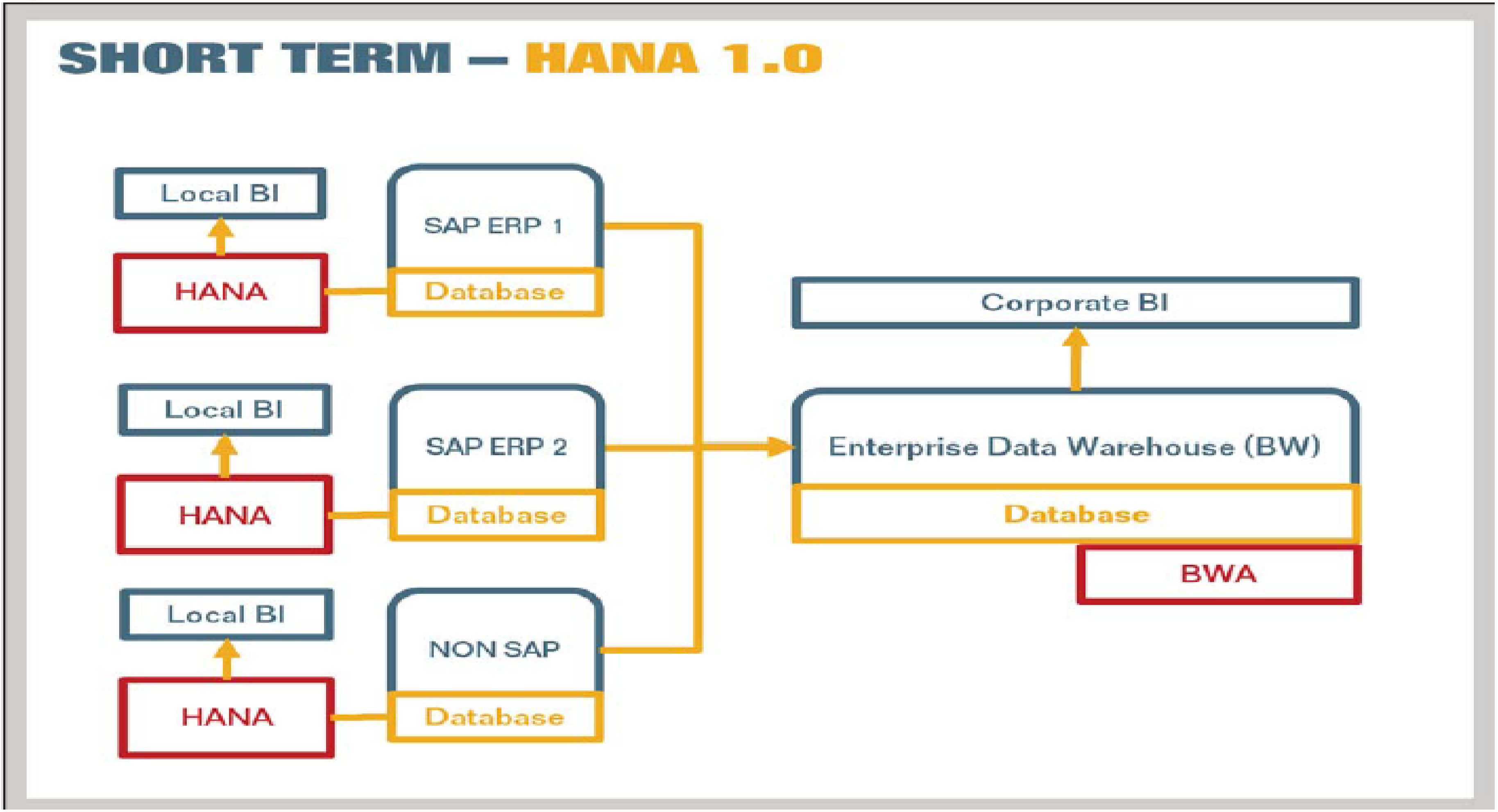
Developer — Re-use ABAP skills

Administrator — Integrated Lifecycle Management and operations

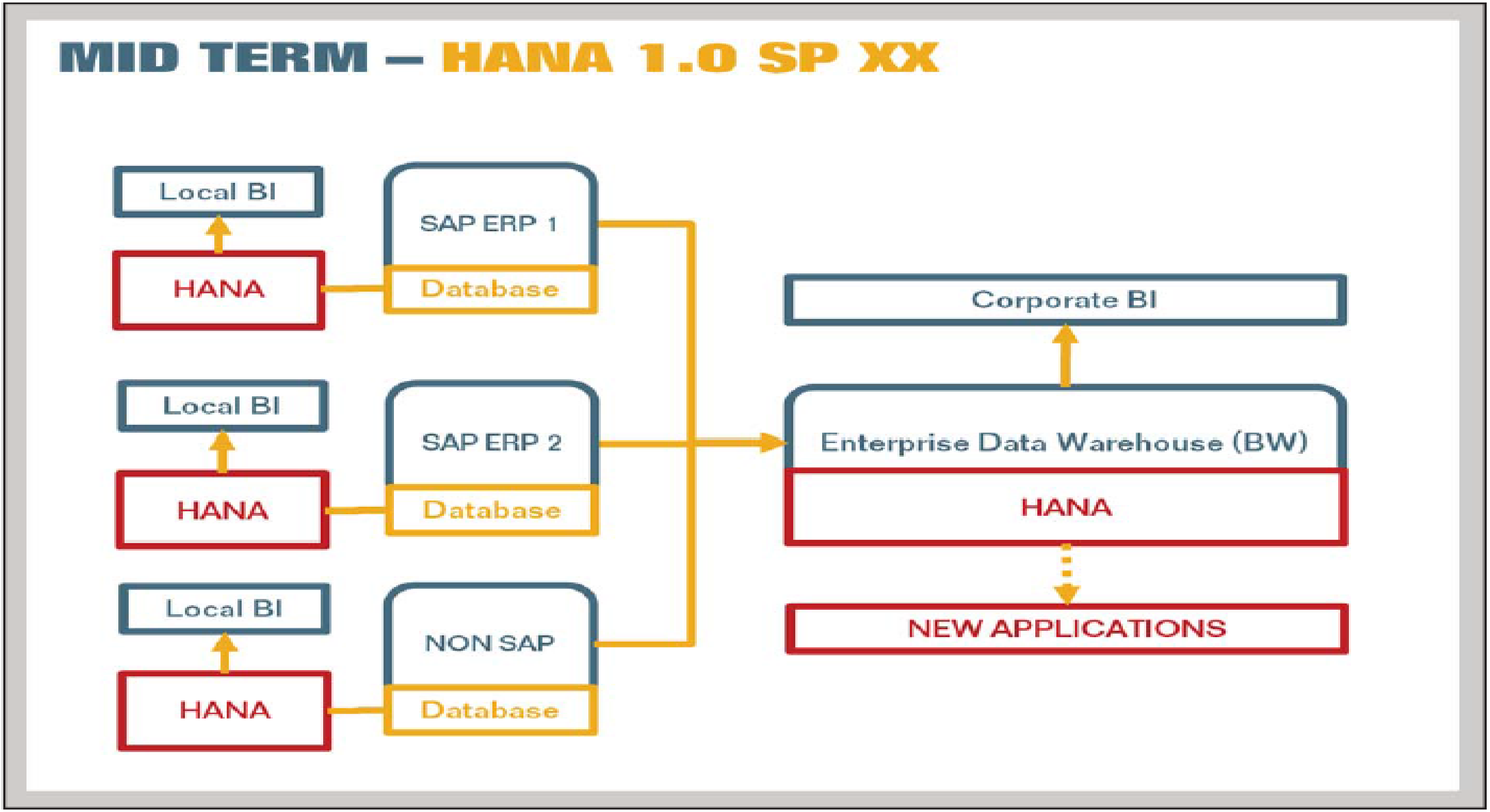
SAP HANA EVOLUTION



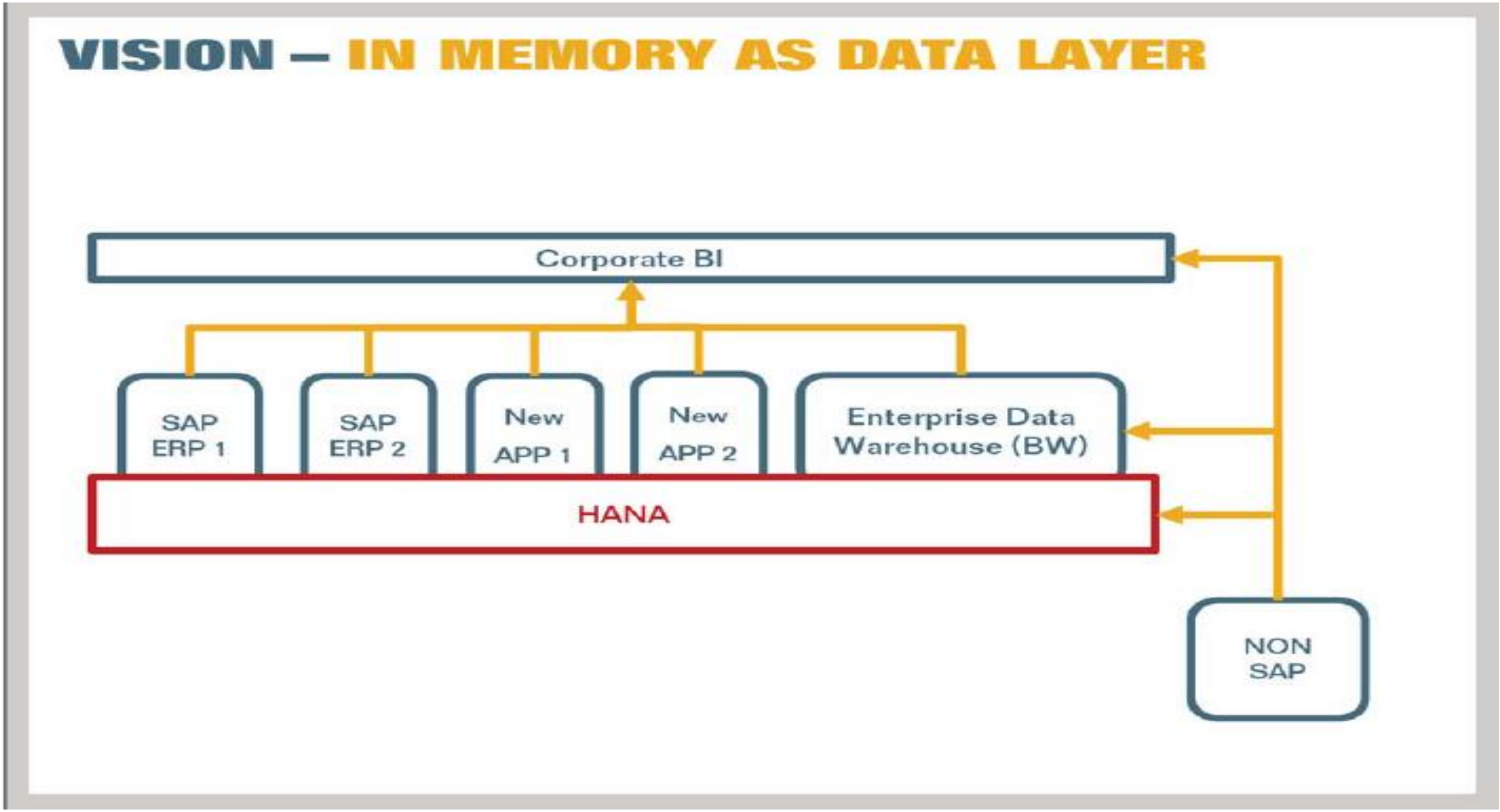
SAP HANA EVOLUTION



SAP HANA EVOLUTION

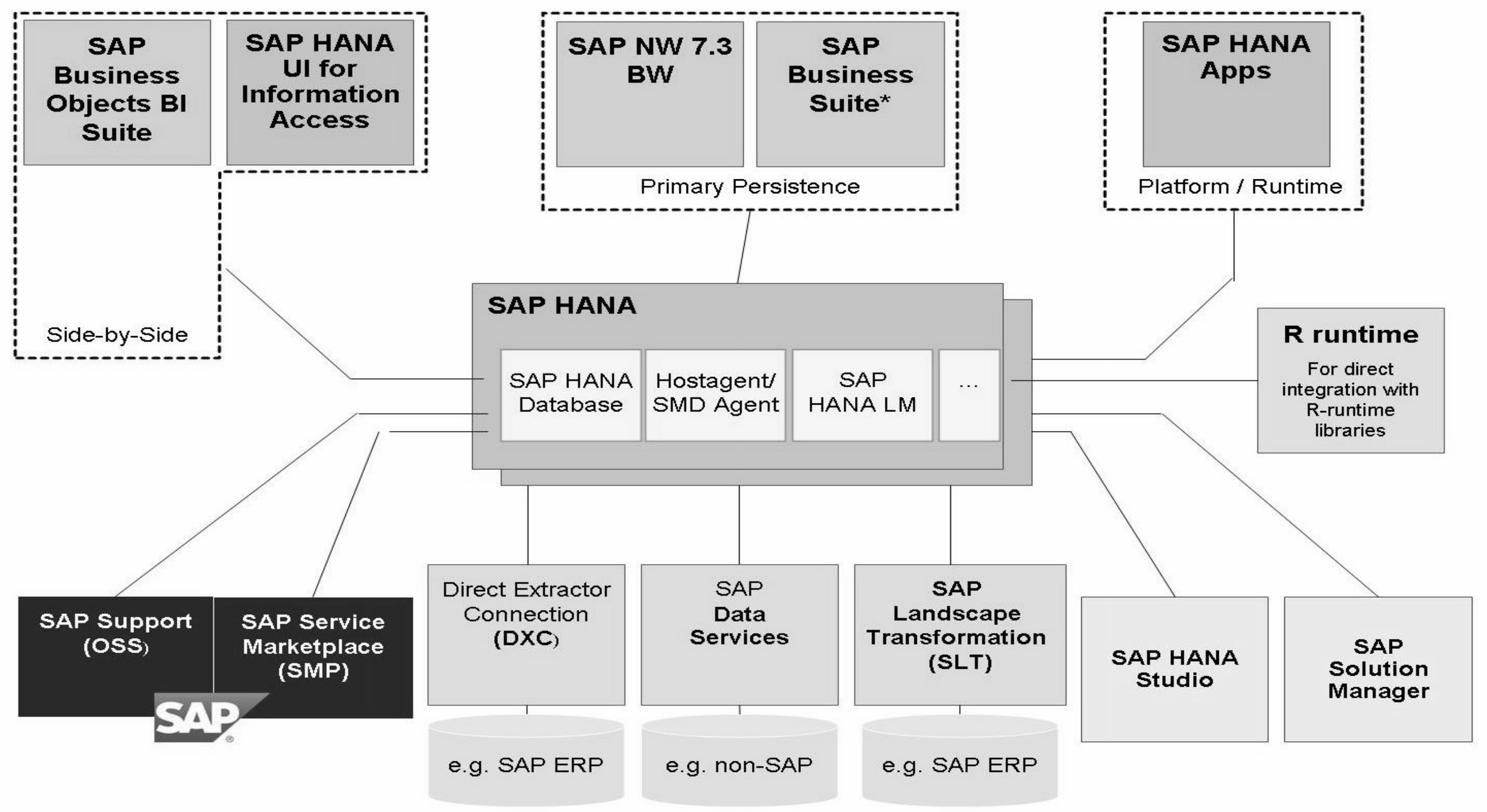


SAP HANA EVOLUTION

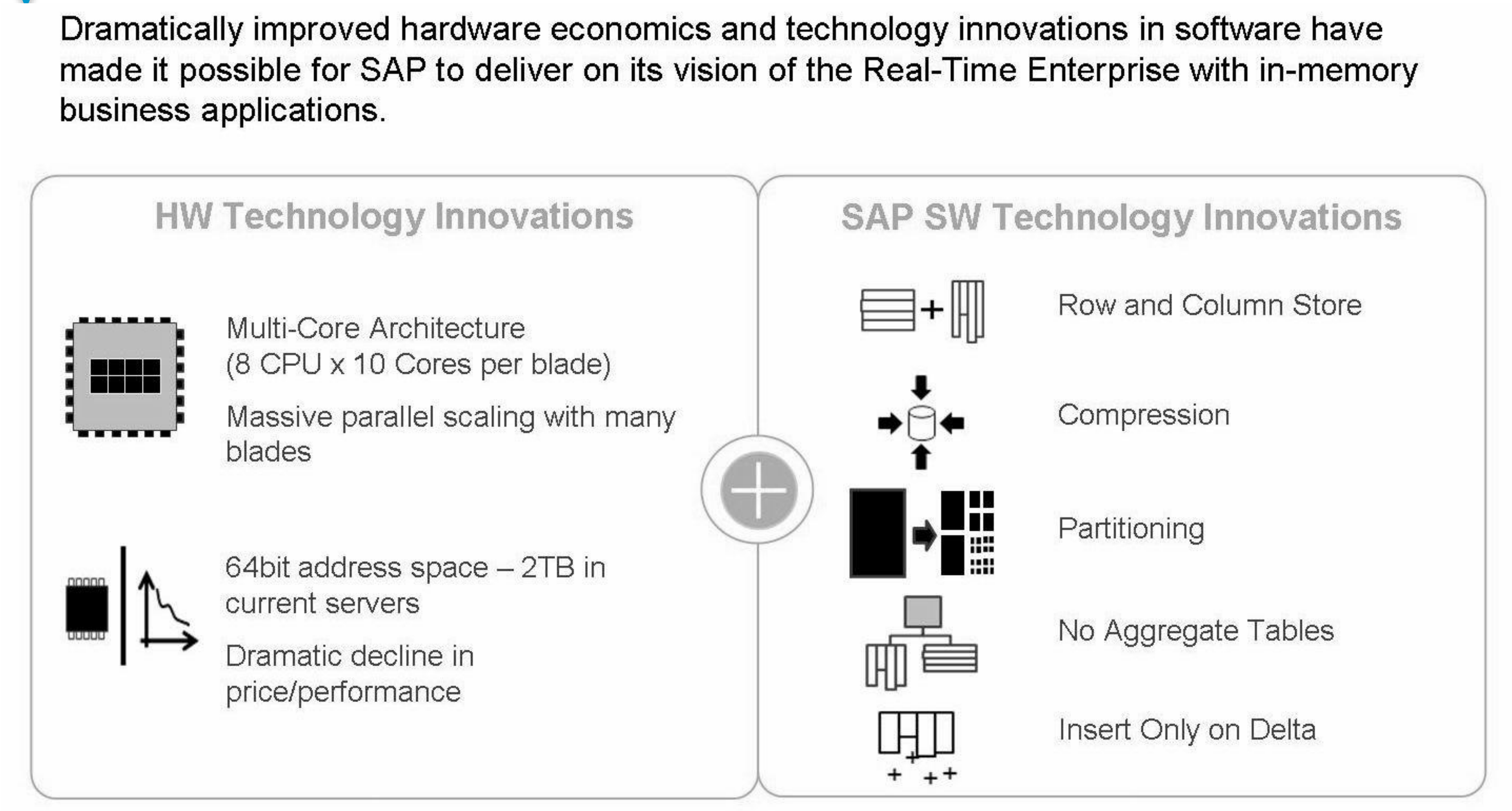


VISION – IN MEMORY AS DATA LAYER

SAP HANA System Landscape Connectivity Overview



Technology Innovations



Dramatically improved hardware economics and technology innovations in software have made it possible for SAP to deliver on its vision of the Real-Time Enterprise with in-memory business applications.

HW Technology Innovations

Multi-Core Architecture
(8 CPU x 10 Cores per blade)

Massive parallel scaling with many blades

64bit address space – 2TB in current servers

Dramatic decline in price/performance

SAP SW Technology Innovations

Row and Column Store

Compression

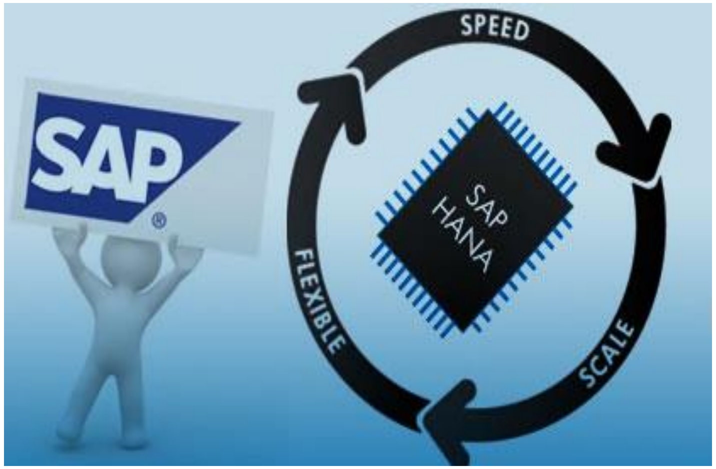
Partitioning

No Aggregate Tables

Insert Only on Delta

HANA Architecture

Flexibility
Columnar Data Storage
SAP Hardware partners
In-Memory Hardware Innovations
Huge Volume of Data
SAP HANA
Real-time Access to Data
High-Performance Analytic Appliance
SAP Software components



What is SAP HANA?

- In the Conventional Data base, transmitting data from that server to all the application servers and back created a huge performance bottleneck. This is due to I/O operations become very high on multiple APP servers with Single DB.

- SAP HANA combines database, data processing, and application platform capabilities in-memory. The platform provides libraries for predictive, planning, text processing, spatial, and business analytics.

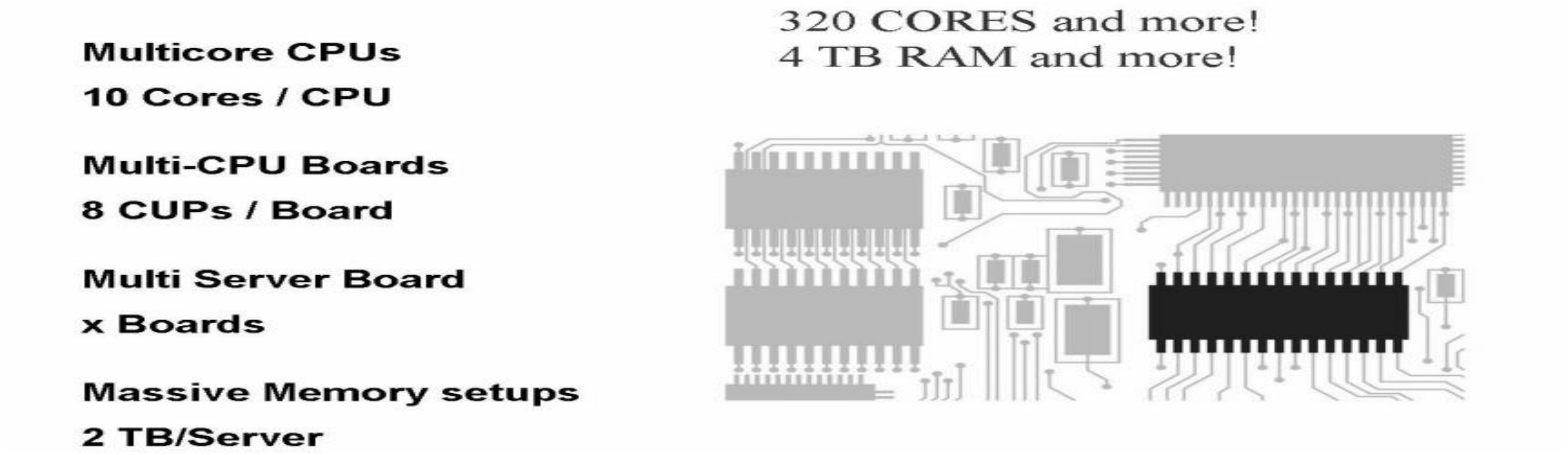
- This new architecture enables converged OLTP and OLAP data processing within a single in-memory column-based data store with ACID compliance, while eliminating data redundancy and latency.

- By providing advanced capabilities, such as predictive text analytics, spatial processing, data virtualization, on the same architecture, it further simplifies application development and processing across big data sources and structures.

- This makes SAP HANA the most suitable platform for building and deploying next-generation, real-time applications and analytics.

The Solution: In-Memory Architecture

- SAP HANA runs on multi-core CPUs with fast communication between processor cores, and containing terabytes of main memory. With SAP HANA, all data is available in main memory, which avoids the performance penalty of disk I/O. Either disk or solid-state drives are still required for permanent persistency in the event of a power failure or some other catastrophe. This does not slow down performance, however, because the required backup operations to disk can take place asynchronously as a background task.



Multicore CPUs
10 Cores / CPU

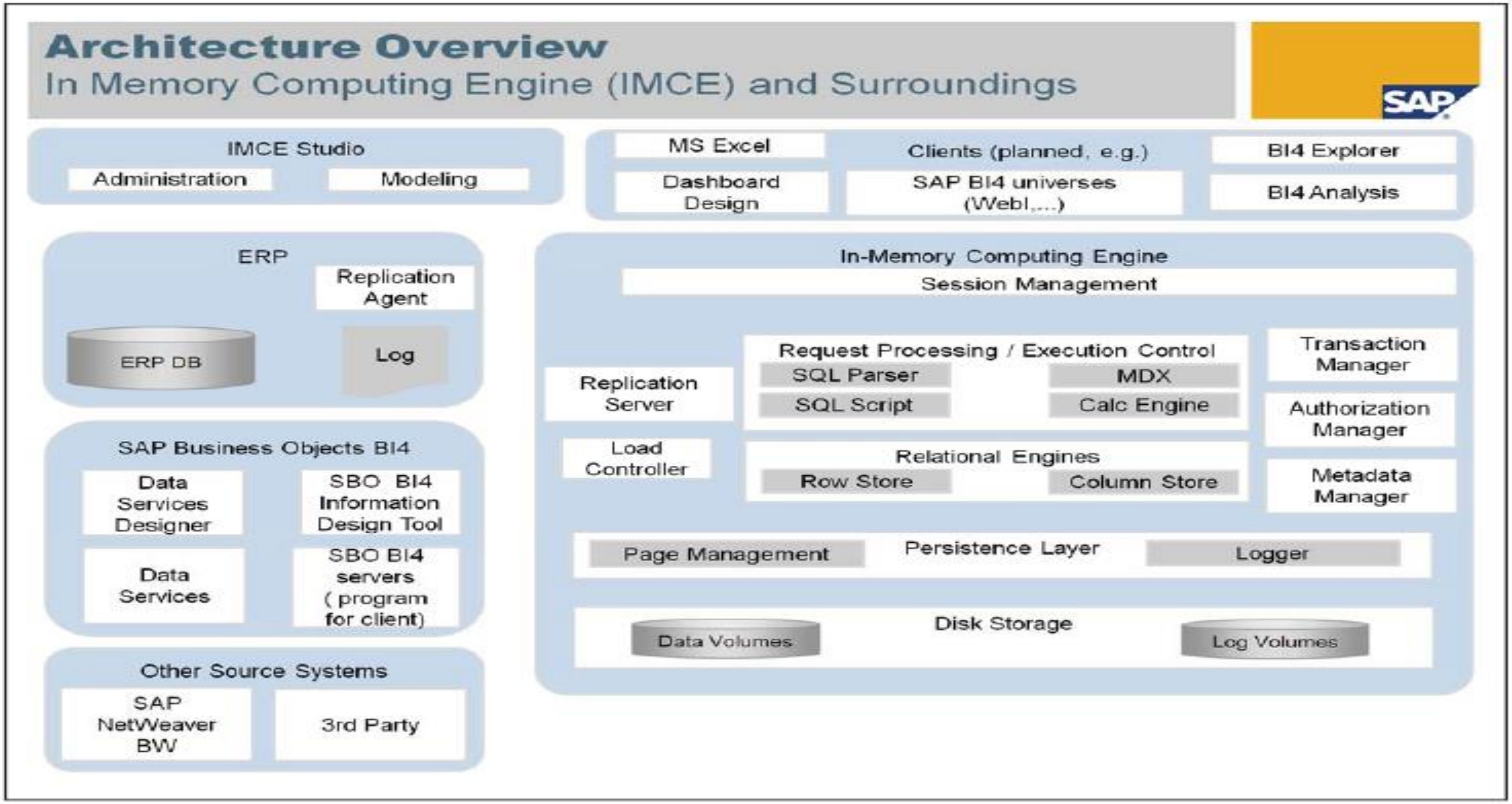
Multi-CPU Boards
8 CUPs / Board

Multi Server Board
x Boards

Massive Memory setups
2 TB/Server

320 CORES and more!
4 TB RAM and more!

Architecture Overview



Architecture Overview

In Memory Computing Engine (IMCE) and Surroundings

SAP

IMCE Studio

Administration

Modeling

MS Excel

Clients (planned, e.g.)

BI4 Explorer

Dashboard Design

SAP BI4 universes (Webl,...)

BI4 Analysis

ERP

Replication Agent

ERP DB

Log

SAP Business Objects BI4

Data Services Designer

SBO BI4 Information Design Tool

Data Services

SBO BI4 servers (program for client)

Other Source Systems

SAP NetWeaver BW

3rd Party

In-Memory Computing Engine

Session Management

Replication Server

Load Controller

Request Processing / Execution Control

SQL Parser

MDX

SQL Script

Calc Engine

Relational Engines

Row Store

Column Store

Transaction Manager

Authorization Manager

Metadata Manager

Page Management

Persistence Layer

Logger

Disk Storage

Data Volumes

Log Volumes

Architecture Overview

- **Session Management:** Connection and Session Management which creates and manages sessions and connections for the database clients. For each session a set of parameters is maintained such as e.g. auto commit settings or the current transaction isolation level.

- **Request Processing and Execution Control:** The client requests are analyzed and executed. Once a session is established, database clients typically use SQL statements to communicate with the in-memory computing engine. For analytical applications the multidimensional query language MDX is supported.

- **Relational Engines:** Are memory based engines in which we store data in Row Store and Column Store. For we need the Persistence layer for data safety to overcome the power cut or OS reboot would permanently erase all data which are currently updating in the tables.

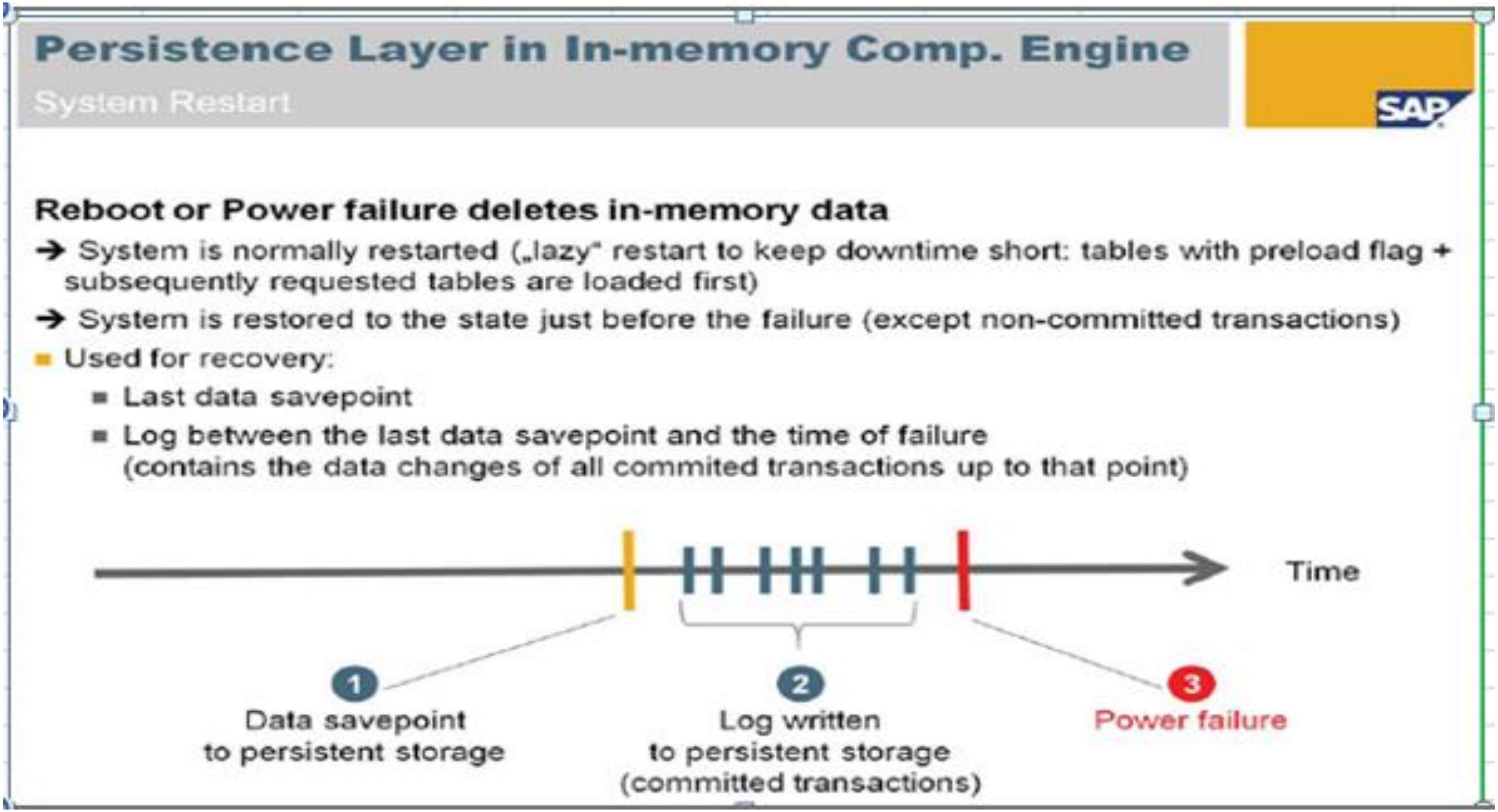
Architecture Overview

- **Persistence Layer:** The persistency layer handles page management and logging(redo and Undo logs) and permanently stores data in a disk storage. This storage has separate volumes for data and log.
- **Transaction Management:** In order to provide consistent views of the data at any given point in time (an ongoing transaction must only see that part of the data that was committed before that transaction was started).
- **Administration Manager:** The In-Memory Computing Studio has an administration component.
- Starting/stopping the In-Memory Computing Engine (upon start, the in-memory stores are reconstructed from the persistence layer).
- User administration including creating/deleting users and authorizations.
- Table administration, including creating indexes or some part of the configuration for data replication.
- Creating or replaying a backup.

Architecture Overview

- Replication Server and Load Controller: are the engine-side part of the Sybase replication manager.
- Data Modelling for data replications.
- Modeling can be done in several places (bottom-up description).
- If data services is used to create and fill the table, first modeling decisions can be made here.
- Data models can be created within the In Memory Computing Engine.
- Models are stored in form of views and associated metadata in the engine.
- The front-end tool to create these models in the In-Memory Computing Studio (Information Modeler within that tool).
- Depending on the front-end tool used to retrieve data from the In-Memory Computing Engine, further modeling decisions can be made in universes (SAP Business Objects Information Design Tool) or other semantic layers.

Architecture Overview



Persistence Layer in In-memory Comp. Engine

System Restart

SAP

Reboot or Power failure deletes in-memory data

→ System is normally restarted („lazy" restart to keep downtime short: tables with preload flag + subsequently requested tables are loaded first)

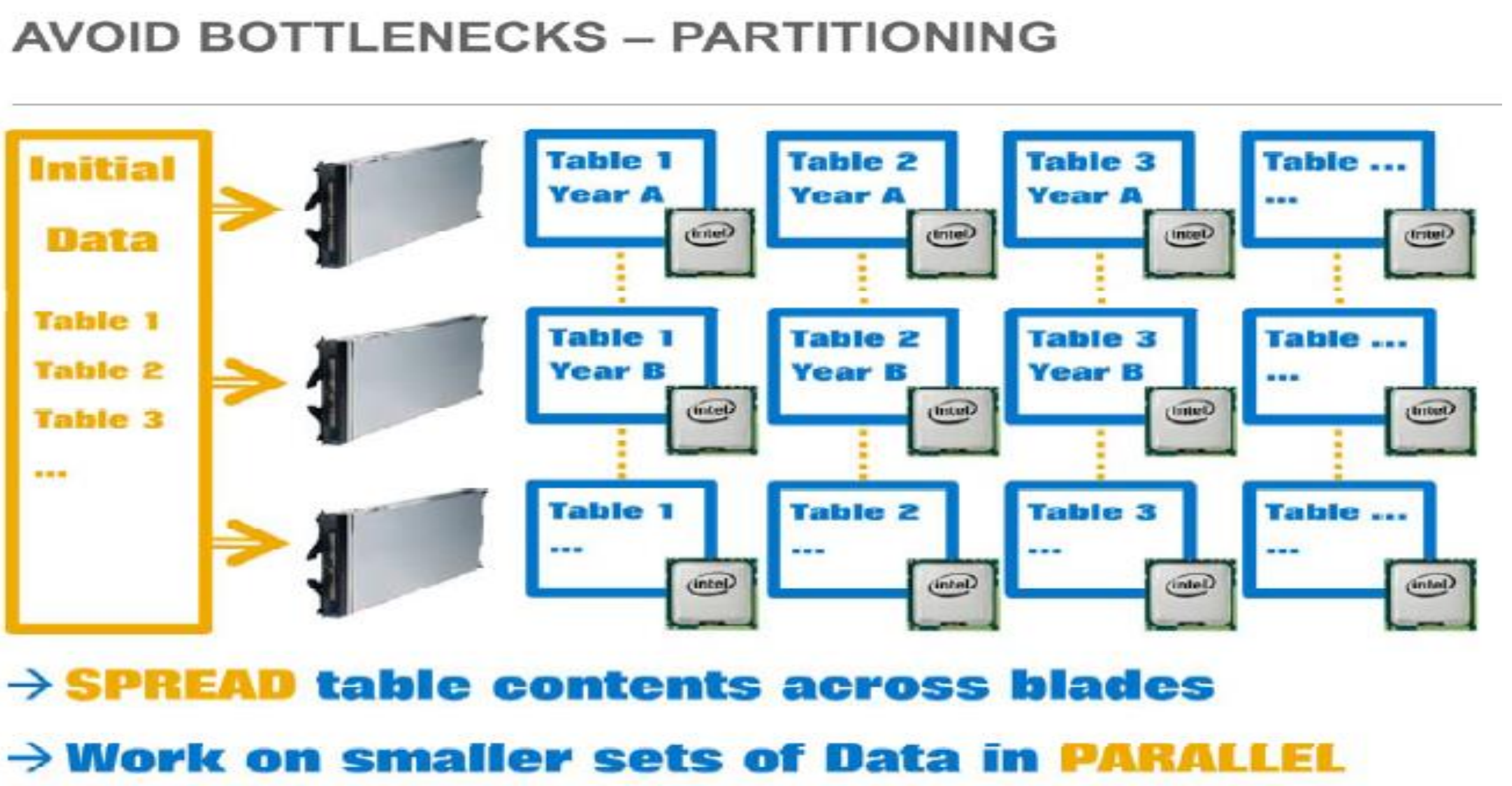
→ System is restored to the state just before the failure (except non-committed transactions)

- Used for recovery:
  - Last data savepoint
  - Log between the last data savepoint and the time of failure (contains the data changes of all commited transactions up to that point)

Time
1
Data savepoint to persistent storage
2
Log written to persistent storage (committed transactions)
3
Power failure

Copyright © Capgemini 2015. All Rights Reserved

In Memory Design



AVOID BOTTLENECKS – PARTITIONING

Initial

Data

Table 1

Table 2

Table 3

...

Table 1
Year A

Table 2
Year A

Table 3
Year A

Table ...
...

Table 1
Year B

Table 2
Year B

Table 3
Year B

Table ...
...

Table 1
...

Table 2
...

Table 3
...

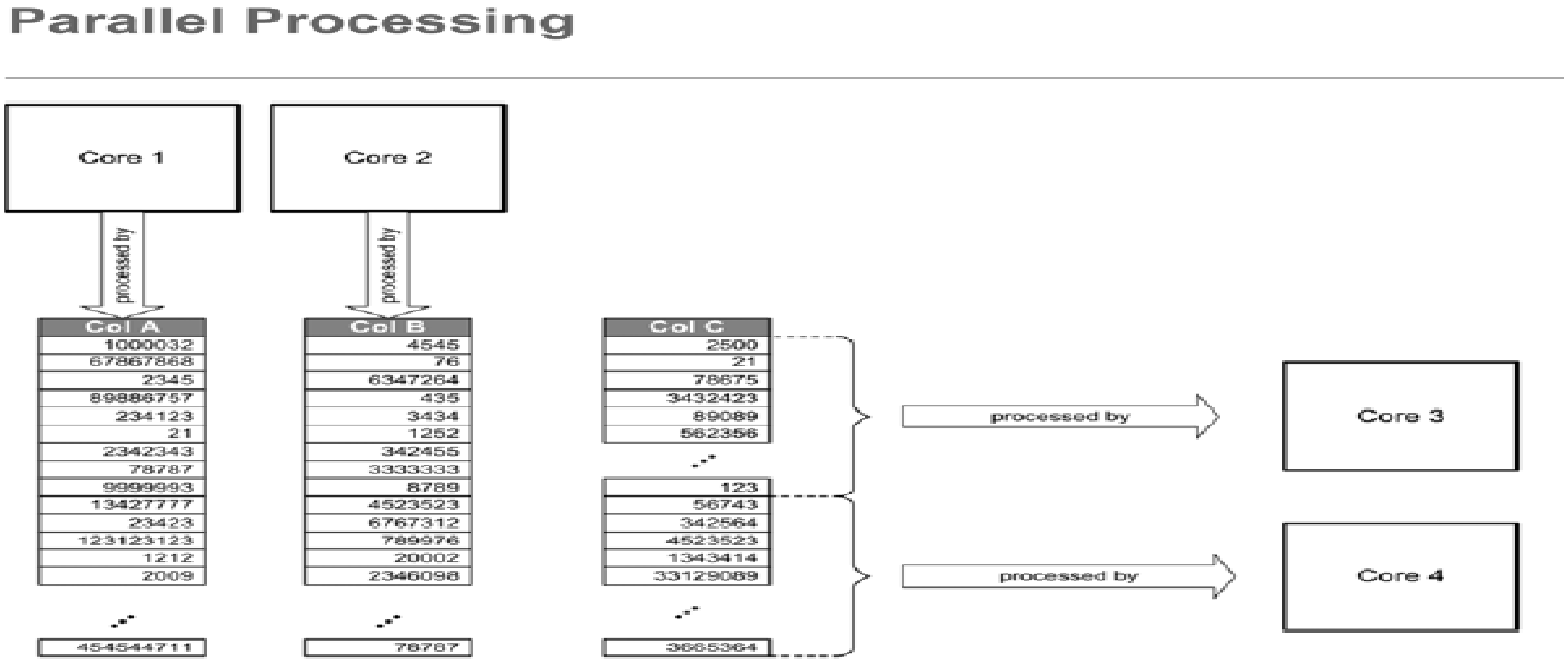
Table ...
...

→ SPREAD table contents across blades

→ Work on smaller sets of Data in PARALLEL

Parallel Processing

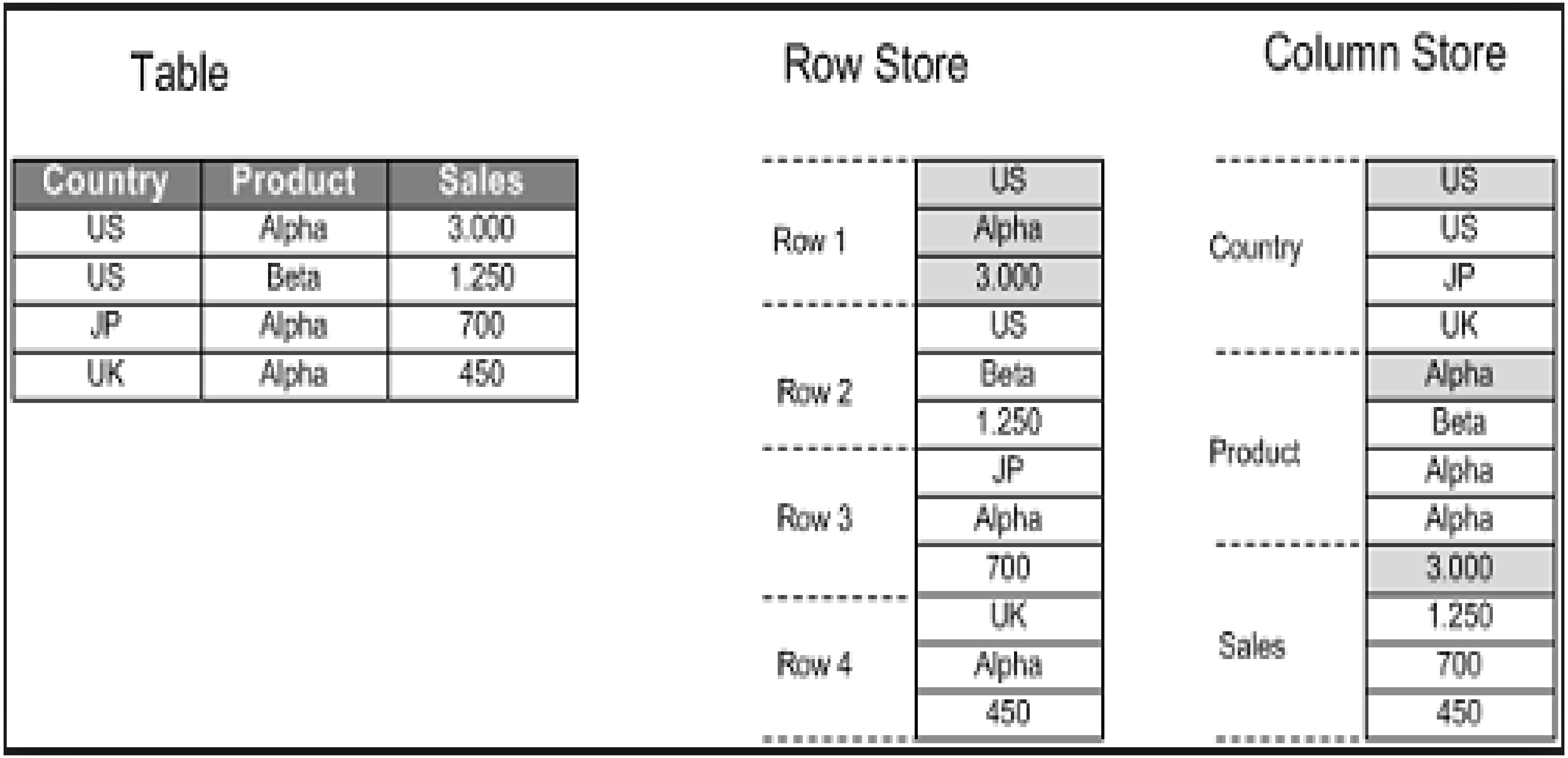
SAP HANA was designed to perform its basic calculations, such as analytic joins, scans and aggregations in parallel. Often it uses hundreds of cores at the same time, fully utilizing the available computing resources of distributed systems.



Col A	Col B	Col C
1000032	4545	2500
67867868	76	21
2345	6347264	78675
89886757	435	3432423
234123	3434	89089
21	1252	562356
2342343	342455	...
78787	3333333	
9999993	8789	123
13427777	4523523	56743
23423	6767312	342564
123123123	789976	4523523
1212	20002	1343414
2009	2346098	331290089
...	...	...
454544711	78787	3665364

Row Store and Column Store

A database table is conceptually a two-dimensional data structure organized in rows and columns. Computer memory, in contrast, is organized as a linear structure. A table can be represented in row-order or column-order.



Table

Country	Product	Sales
US	Alpha	3.000
US	Beta	1.250
JP	Alpha	700
UK	Alpha	450

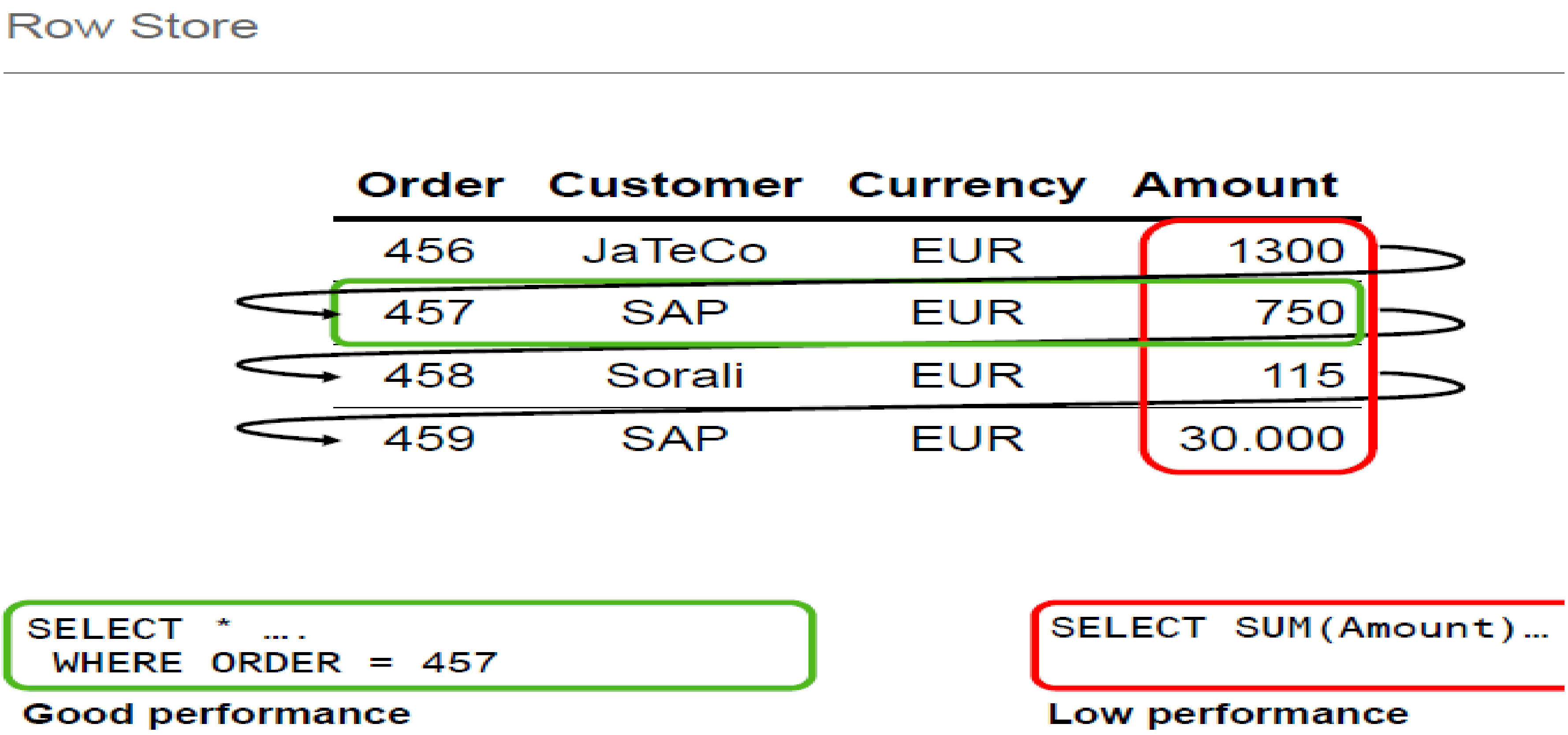
Row Store

Row 1	US
	Alpha
	3.000
Row 2	US
	Beta
	1.250
Row 3	JP
	Alpha
	700
Row 4	UK
	Alpha
	450

Column Store

Country	US
	US
	JP
	UK
Product	Alpha
	Beta
	Alpha
	Alpha
Sales	3.000
	1.250
	700
	450

Row Store



Row Store

Order	Customer	Currency	Amount
456	JaTeCo	EUR	1300
457	SAP	EUR	750
458	Sorali	EUR	115
459	SAP	EUR	30.000

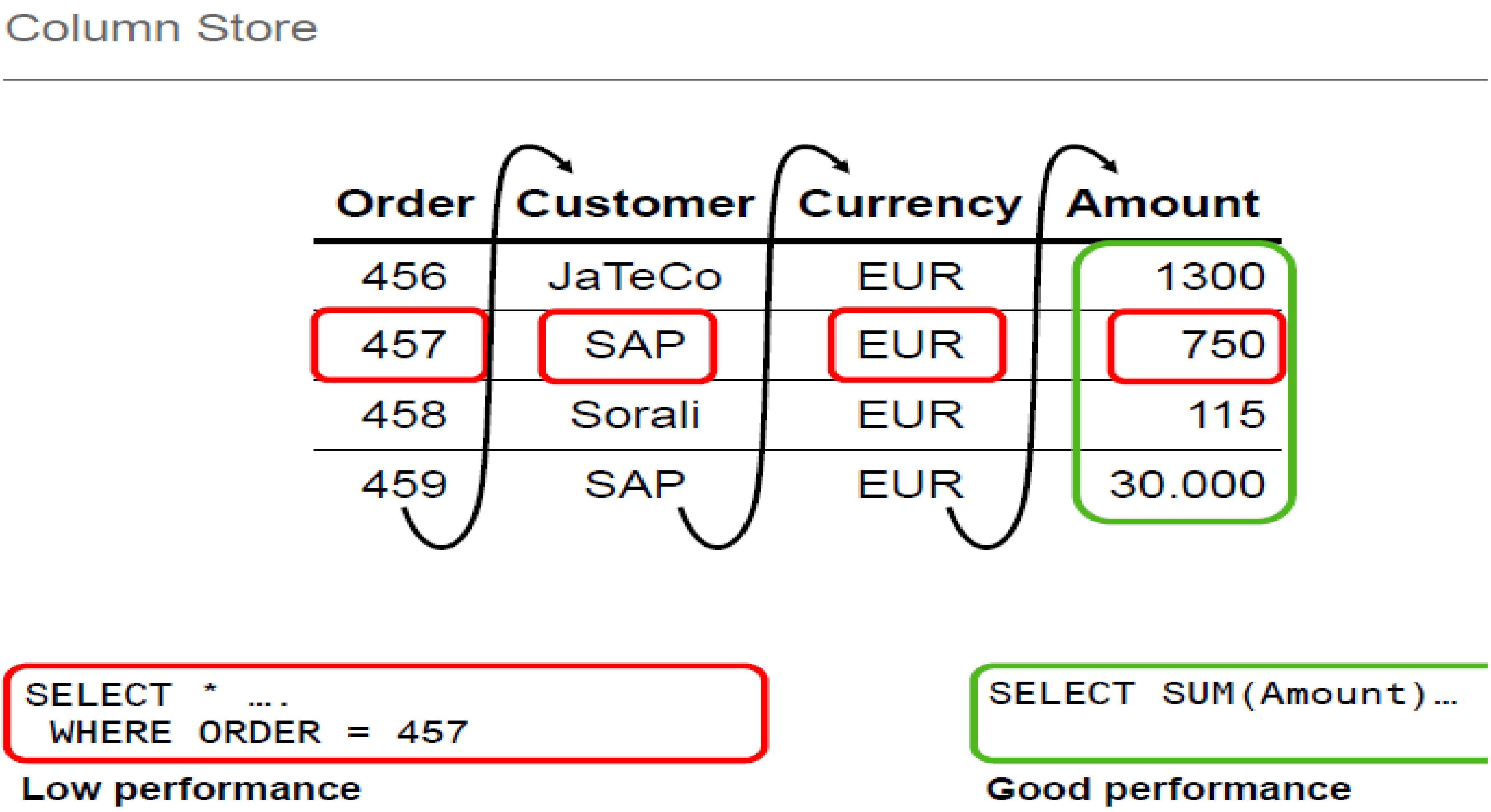
```
SELECT  *  ….
 WHERE  ORDER  =  457
```

Good performance

```
SELECT  SUM(Amount)…
```

Low performance

Column Store



Column Store

Order	Customer	Currency	Amount
456	JaTeCo	EUR	1300
457	SAP	EUR	750
458	Sorali	EUR	115
459	SAP	EUR	30.000

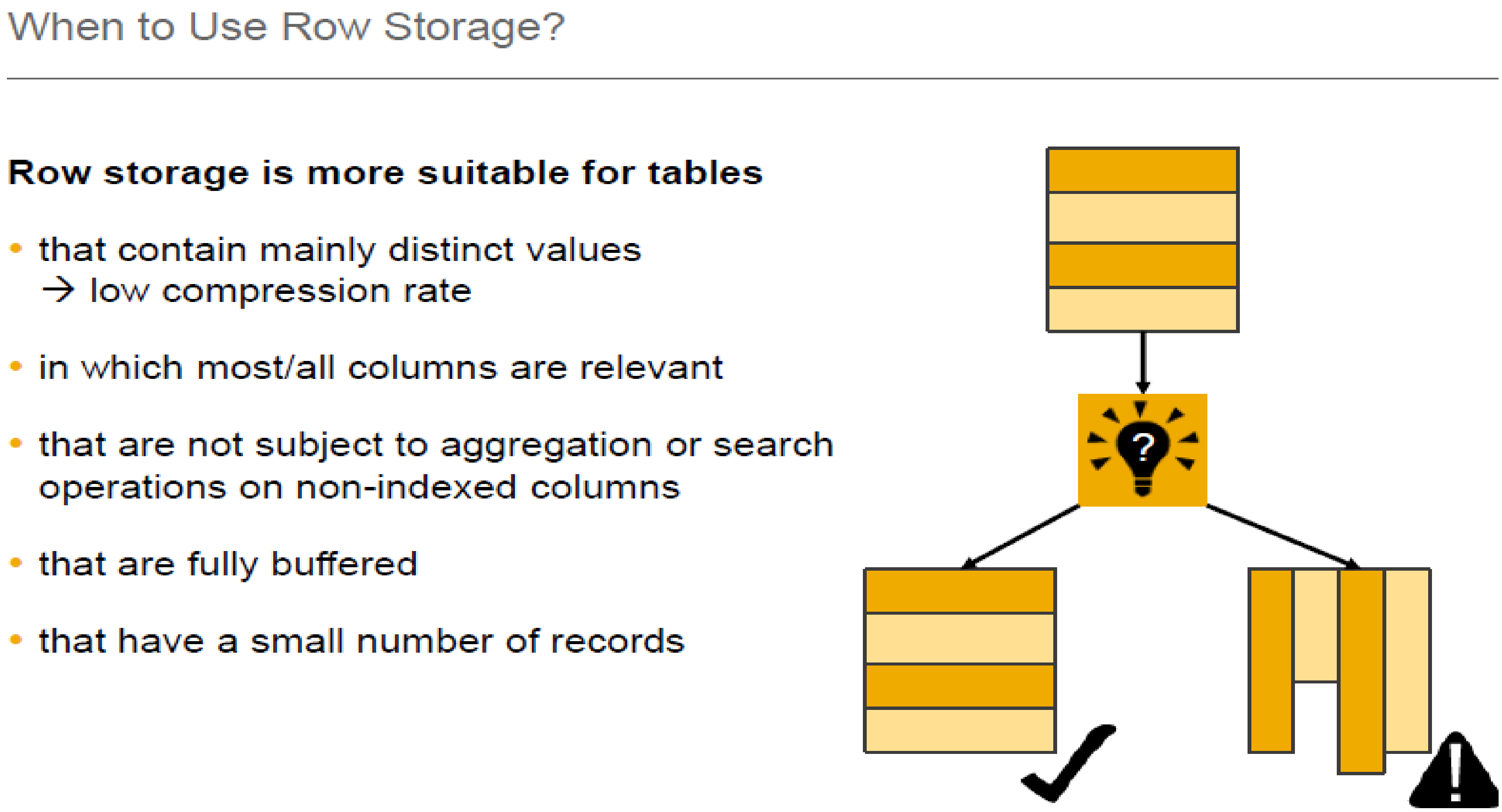
```
SELECT * ….
 WHERE ORDER = 457
```

Low performance

```
SELECT SUM(Amount)…
```

Good performance

When to use row storage

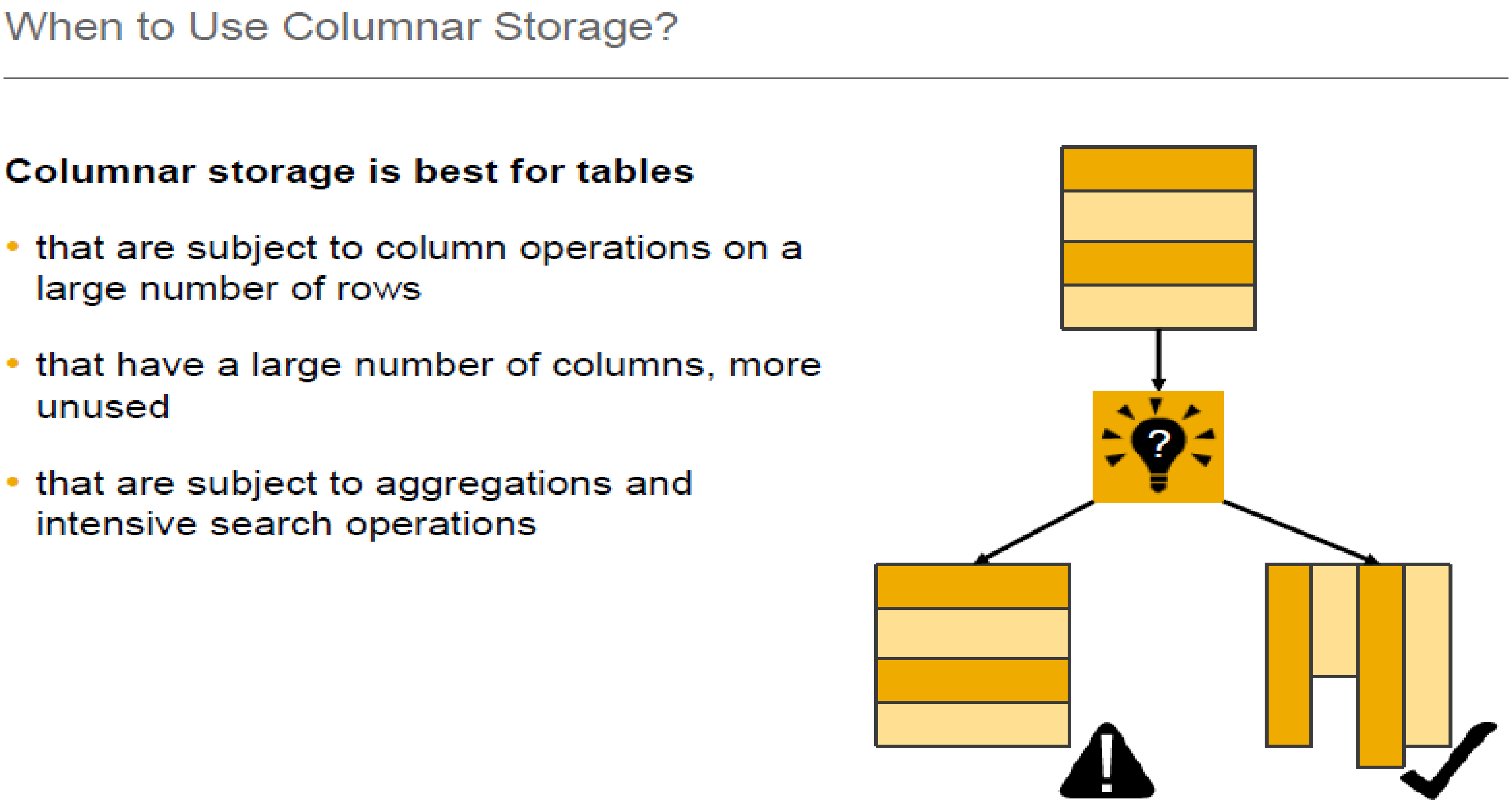


When to Use Row Storage?

Row storage is more suitable for tables

- that contain mainly distinct values
  → low compression rate
- in which most/all columns are relevant
- that are not subject to aggregation or search operations on non-indexed columns
- that are fully buffered
- that have a small number of records

When to use columnar storage

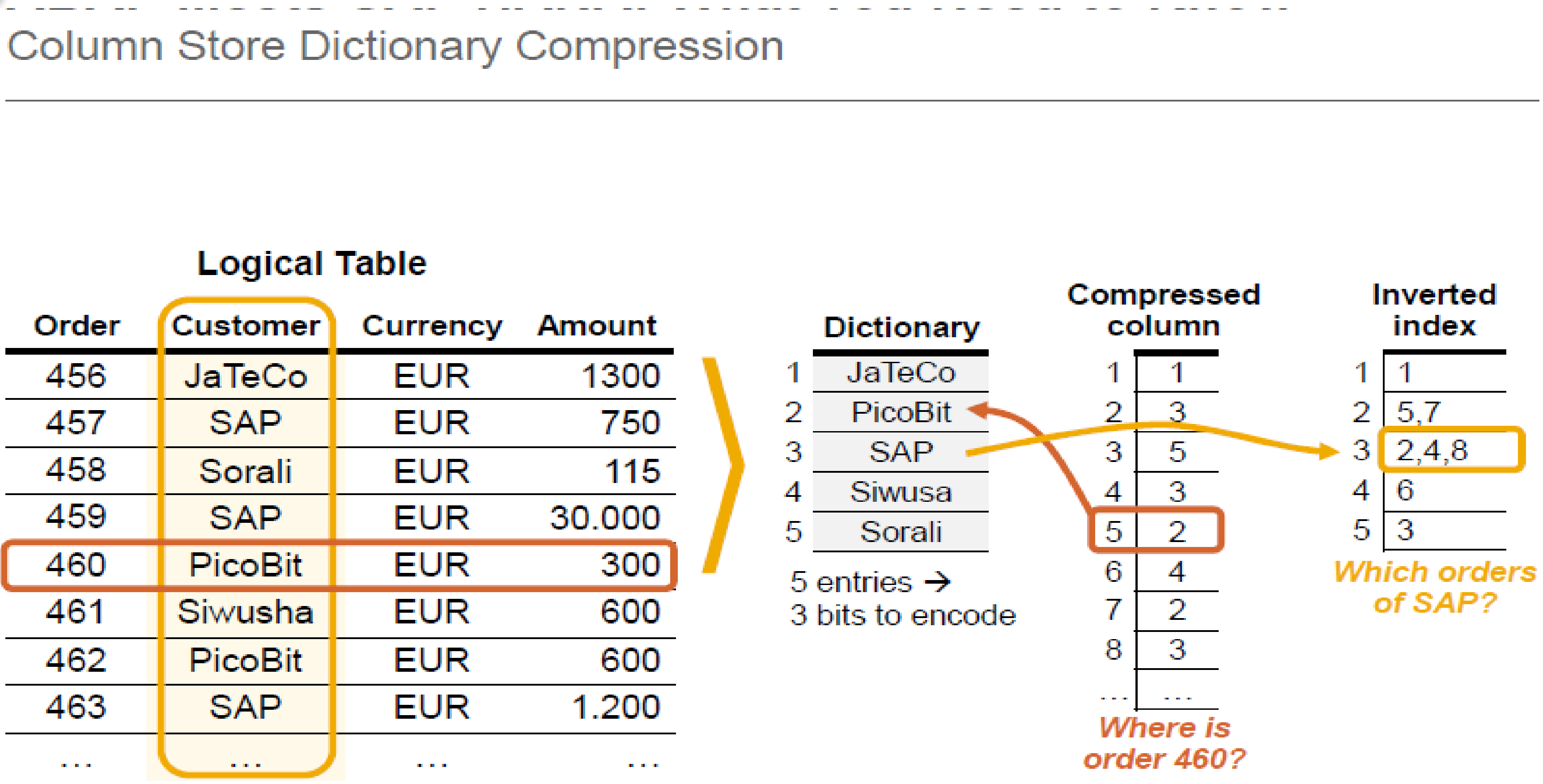


When to Use Columnar Storage?

Columnar storage is best for tables

- that are subject to column operations on a large number of rows
- that have a large number of columns, more unused
- that are subject to aggregations and intensive search operations

Dictionary Compression



Column Store Dictionary Compression

Logical Table

Order	Customer	Currency	Amount
456	JaTeCo	EUR	1300
457	SAP	EUR	750
458	Sorali	EUR	115
459	SAP	EUR	30.000
460	PicoBit	EUR	300
461	Siwusha	EUR	600
462	PicoBit	EUR	600
463	SAP	EUR	1.200
...	...	...	...

Dictionary

1	JaTeCo
2	PicoBit
3	SAP
4	Siwusa
5	Sorali

5 entries →
3 bits to encode

Compressed column

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

Where is order 460?

Inverted index

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

Which orders of SAP?

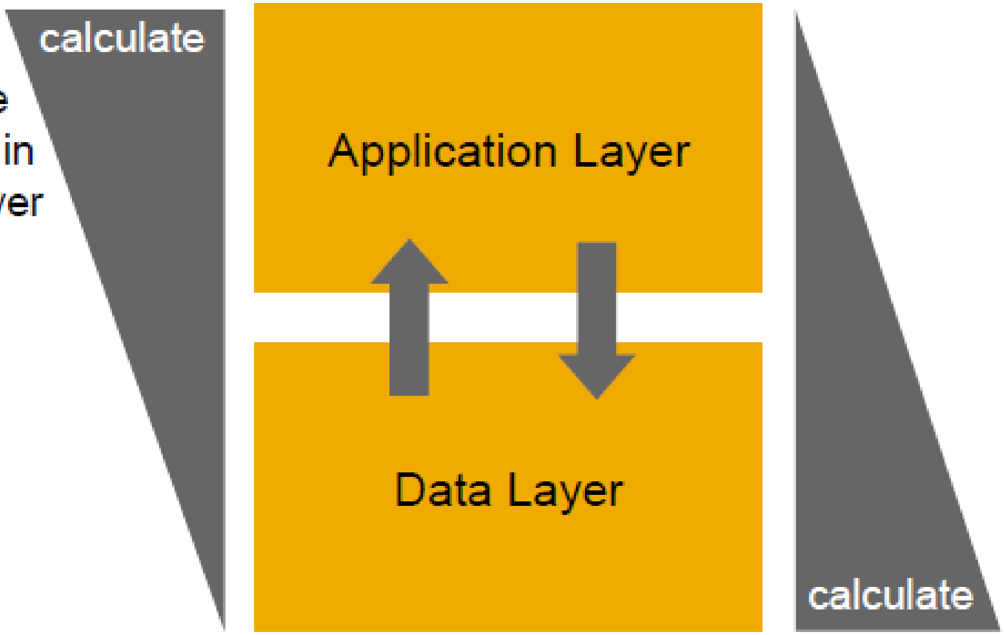
SAP HANA Coding Pattern

In-memory computing imperative:

- Avoid (unnecessary) movement of large data volume
- Perform data-intensive calculations in the database

Today:

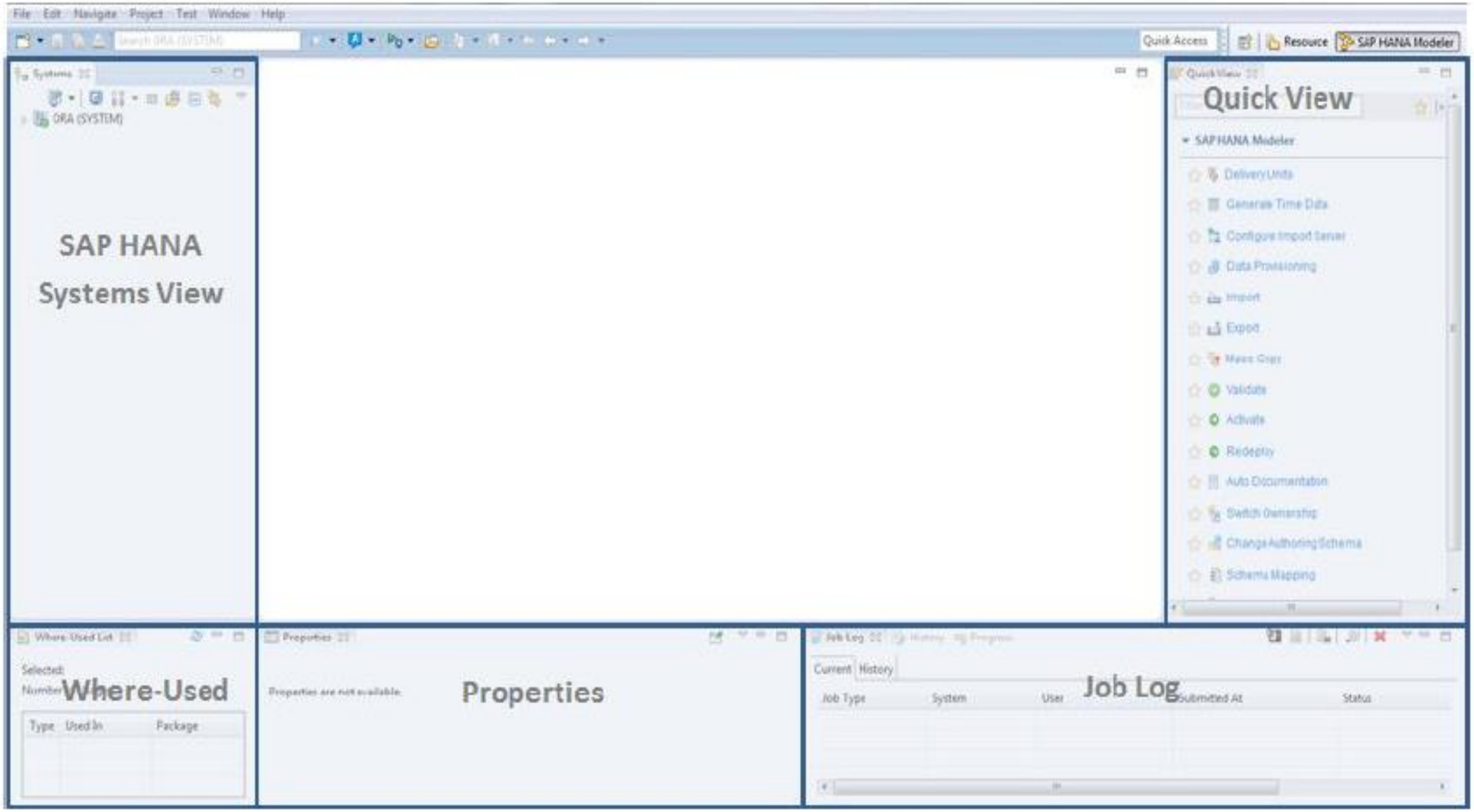
Data-intensive computations in application layer



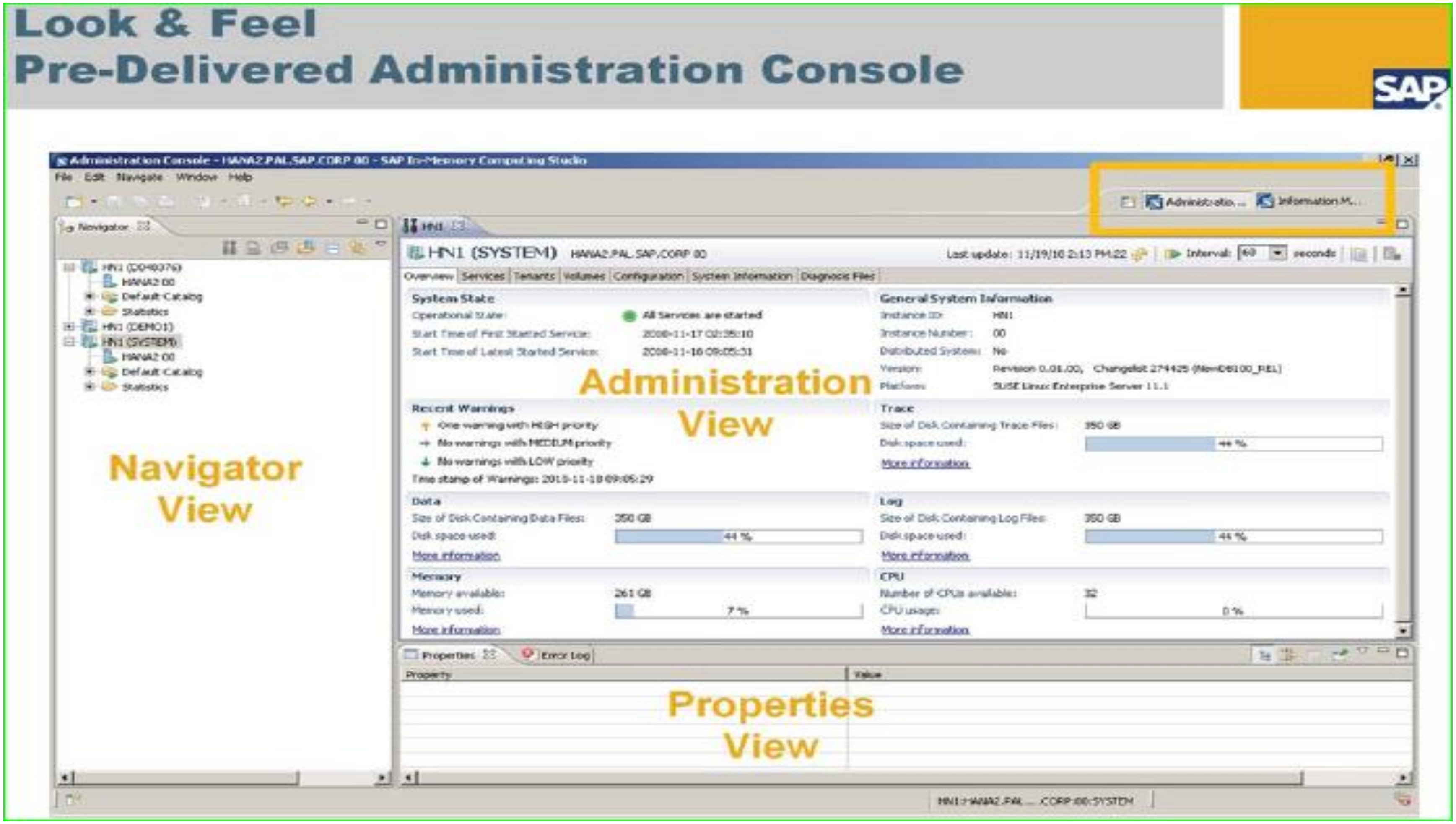
SAP HANA:

Delegate data-intensive operations to data layer

Introduction to SAP HANA Studio



Administration Console



Look & Feel

Pre-Delivered Administration Console

Administration Console - HANA2.PAL.SAP.CORP 00 - SAP In-Memory Computing Studio

File Edit Navigate Window Help

Navigator

- HN1 (D048376)
  - HANA2 00
  - Default Catalog
  - Statistics
- HN1 (DEMO1)
- HN1 (SYSTEM)
  - HANA2 00
  - Default Catalog
  - Statistics

HN1 (SYSTEM) HANA2.PAL.SAP.CORP 00

Last update: 11/19/10 2:13 PM:22 Interval: 60 seconds

Overview | Services | Tenants | Volumes | Configuration | System Information | Diagnosis Files

System State

Operational State: All Services are started

Start Time of First Started Service: 2008-11-17 02:35:10

Start Time of Latest Started Service: 2008-11-18 09:05:31

General System Information

Instance ID: HN1

Instance Number: 00

Distributed System: No

Version: Revision 0.01.00, Changelist 274425 (NewDB100_REL)

Platform: SUSE Linux Enterprise Server 11.1

Recent Warnings

One warning with HIGH priority

No warnings with MEDIUM priority

No warnings with LOW priority

Time stamp of Warnings: 2010-11-18 09:05:29

Trace

Size of Disk Containing Trace Files: 350 GB

Disk space used: 44 %

More information

Data

Size of Disk Containing Data Files: 350 GB

Disk space used: 44 %

More information

Log

Size of Disk Containing Log Files: 350 GB

Disk space used: 44 %

More information

Memory

Memory available: 261 GB

Memory used: 7 %

More information

CPU

Number of CPUs available: 32

CPU usage: 0 %

More information

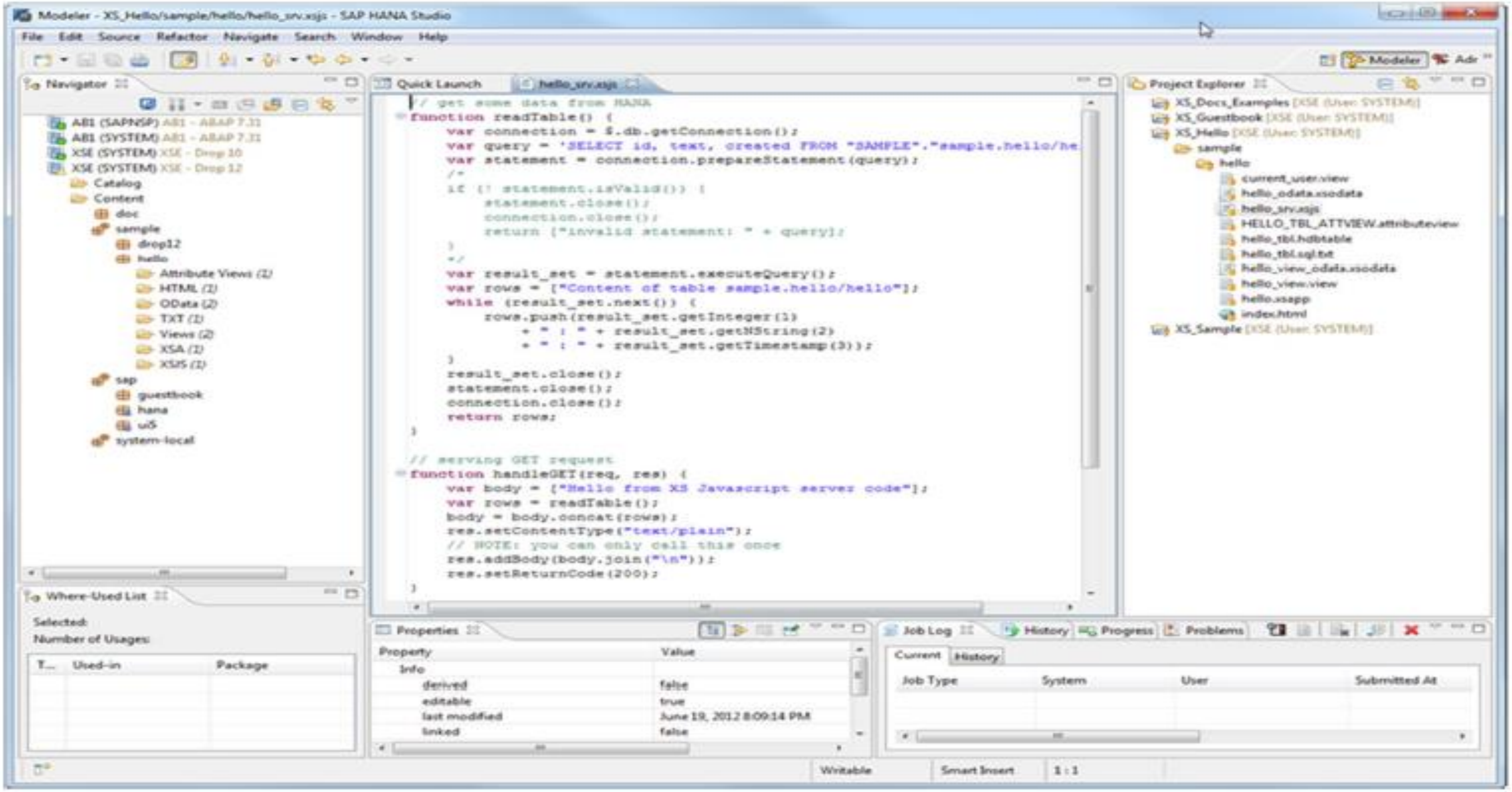
Properties | Error Log

Property	Value

HN1:HANA2.PAL....CORP 00:SYSTEM

Introduction to SAP HANA Studio

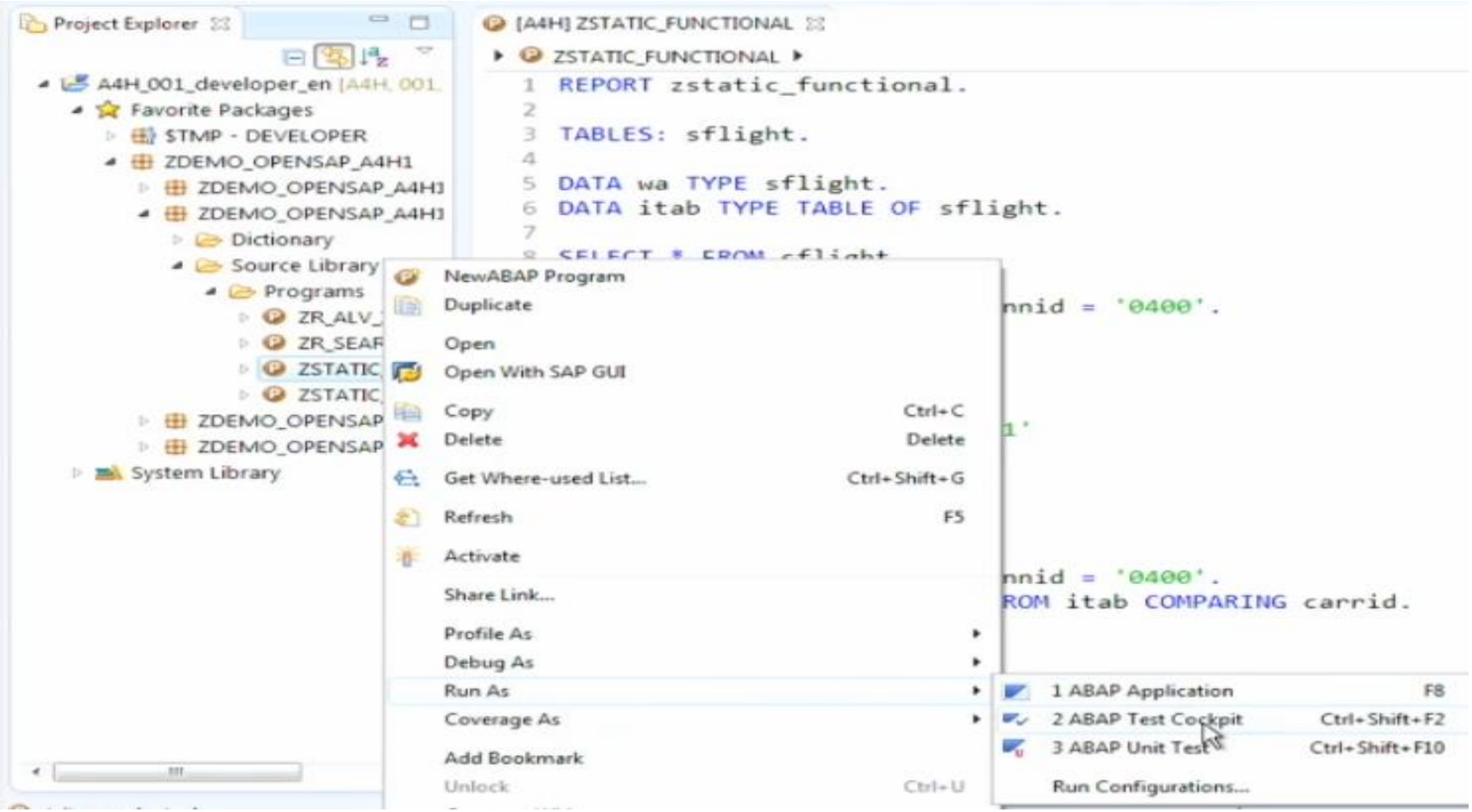
SAP HANA studio: Editing Tools



```
// get some data from HANA
function readTable() {
    var connection = $.db.getConnection();
    var query = 'SELECT id, text, created FROM "SAMPLE"."sample.hello/he
    var statement = connection.prepareStatement(query);
    /*
    if (! statement.isValid()) {
        statement.close();
        connection.close();
        return ["invalid statement: " + query];
    }
    */
    var result_set = statement.executeQuery();
    var rows = ["Content of table sample.hello/hello"];
    while (result_set.next()) {
        rows.push(result_set.getInteger(1)
            + " : " + result_set.getNString(2)
            + " : " + result_set.getTimestamp(3));
    }
    result_set.close();
    statement.close();
    connection.close();
    return rows;
}

// serving GET request
function handleGET(req, res) {
    var body = ["Hello from XS Javascript server code"];
    var rows = readTable();
    body = body.concat(rows);
    res.setContentType("text/plain");
    // NOTE: you can only call this once
    res.addBody(body.join("\n"));
    res.setReturnCode(200);
}
```

Property	Value
Info	
derived	false
editable	true
last modified	June 19, 2012 8:09:14 PM
linked	false

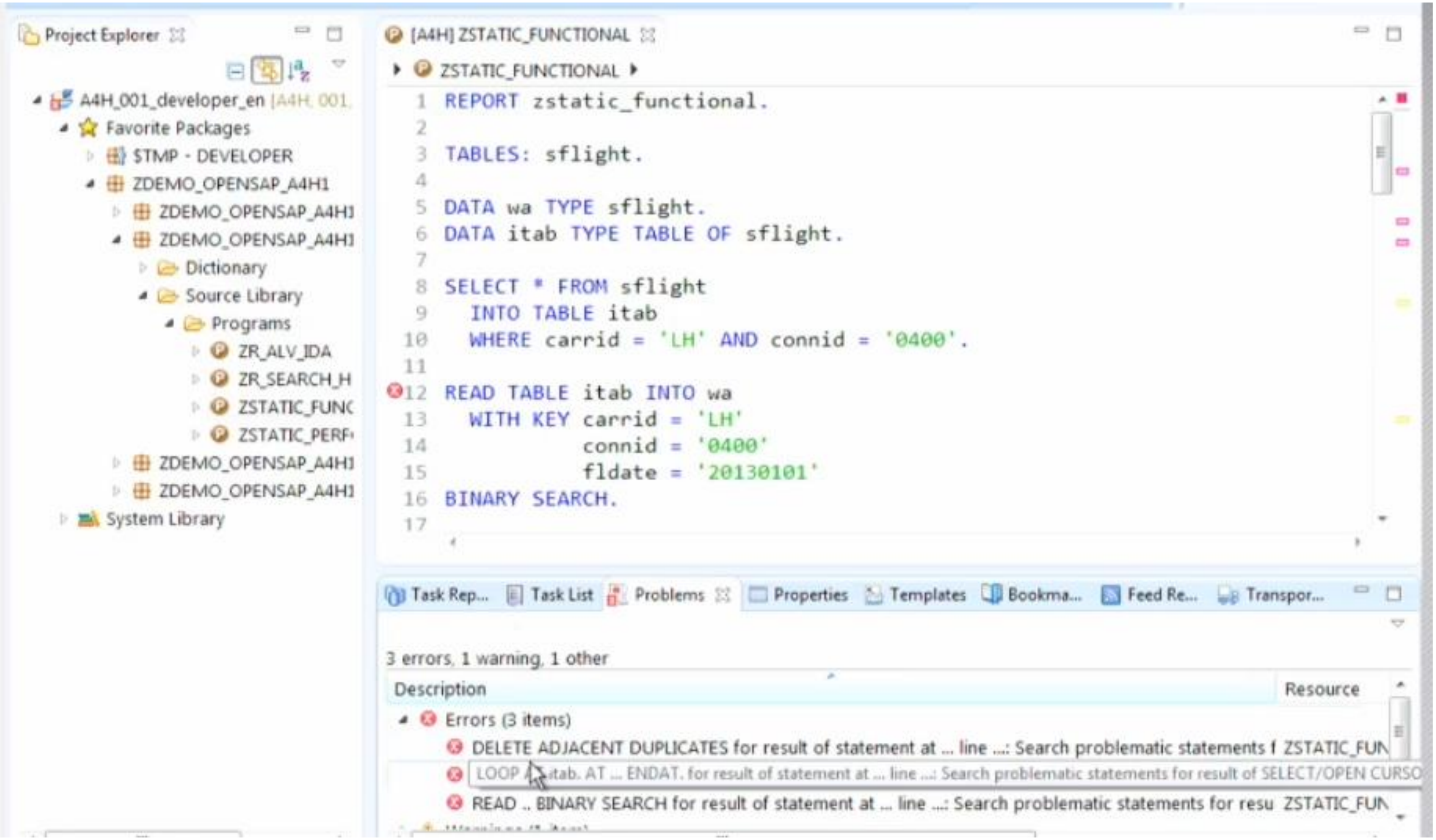
ABAP Code on HANA Studio



```
REPORT zstatic_functional.

TABLES: sflight.

DATA wa TYPE sflight.
DATA itab TYPE TABLE OF sflight.

SELECT * FROM sflight
nnid = '0400'.
1'
nnid = '0400'.
ROM itab COMPARING carrid.
```

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

ABAP Code and Errors on HANA Studio

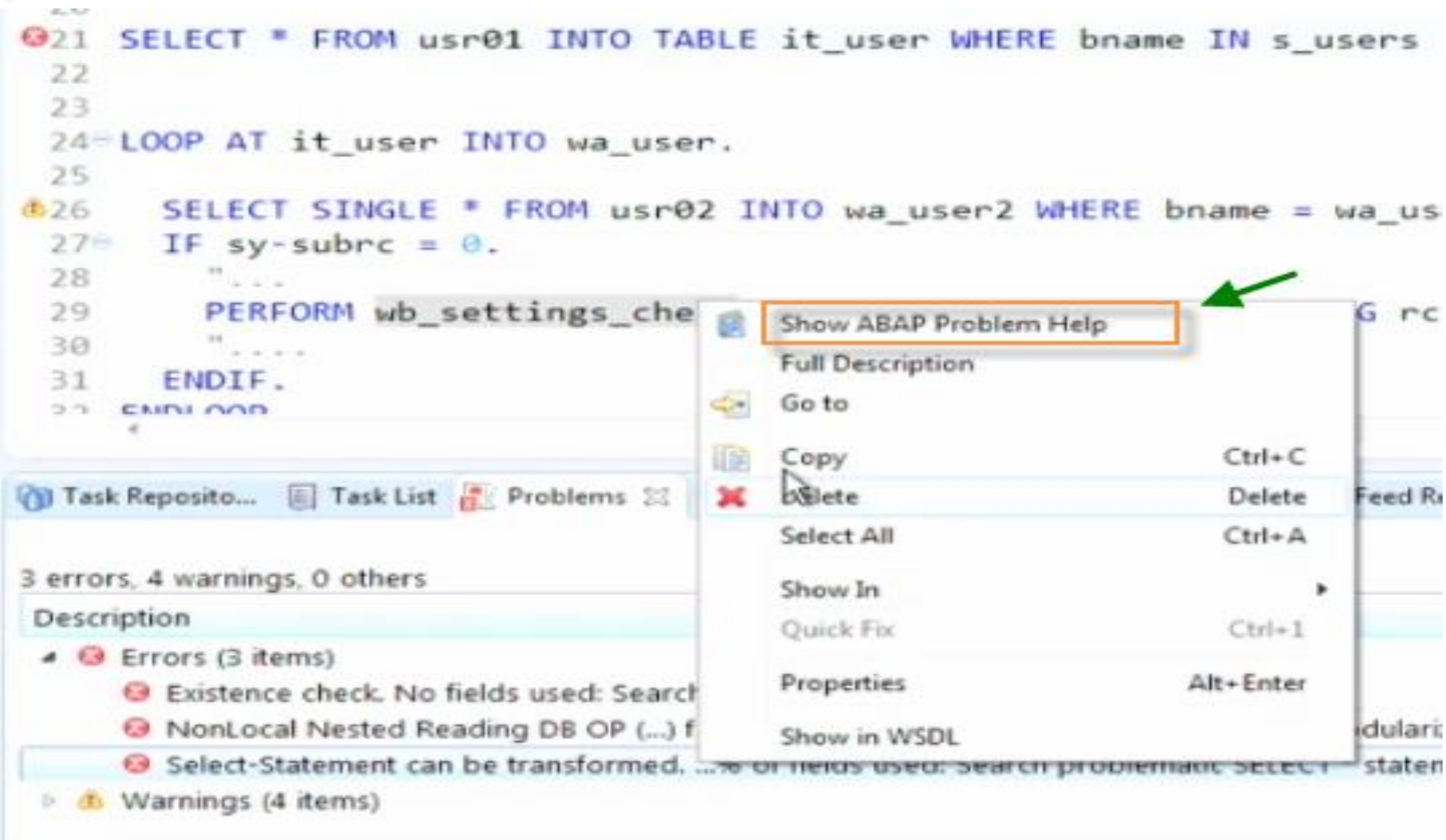


```
REPORT zstatic_functional.

TABLES: sflight.

DATA wa TYPE sflight.
DATA itab TYPE TABLE OF sflight.

SELECT * FROM sflight
  INTO TABLE itab
  WHERE carrid = 'LH' AND connid = '0400'.

READ TABLE itab INTO wa
  WITH KEY carrid = 'LH'
           connid = '0400'
           fldate = '20130101'
BINARY SEARCH.
```

3 errors, 1 warning, 1 other

Description	Resource
Errors (3 items)	
DELETE ADJACENT DUPLICATES for result of statement at ... line ...: Search problematic statements f	ZSTATIC_FUN
LOOP A itab. AT ... ENDAT. for result of statement at ... line ...: Search problematic statements for result of SELECT/OPEN CURSO	
READ .. BINARY SEARCH for result of statement at ... line ...: Search problematic statements for resu	ZSTATIC_FUN

ABAP Code and Help on HANA Studio



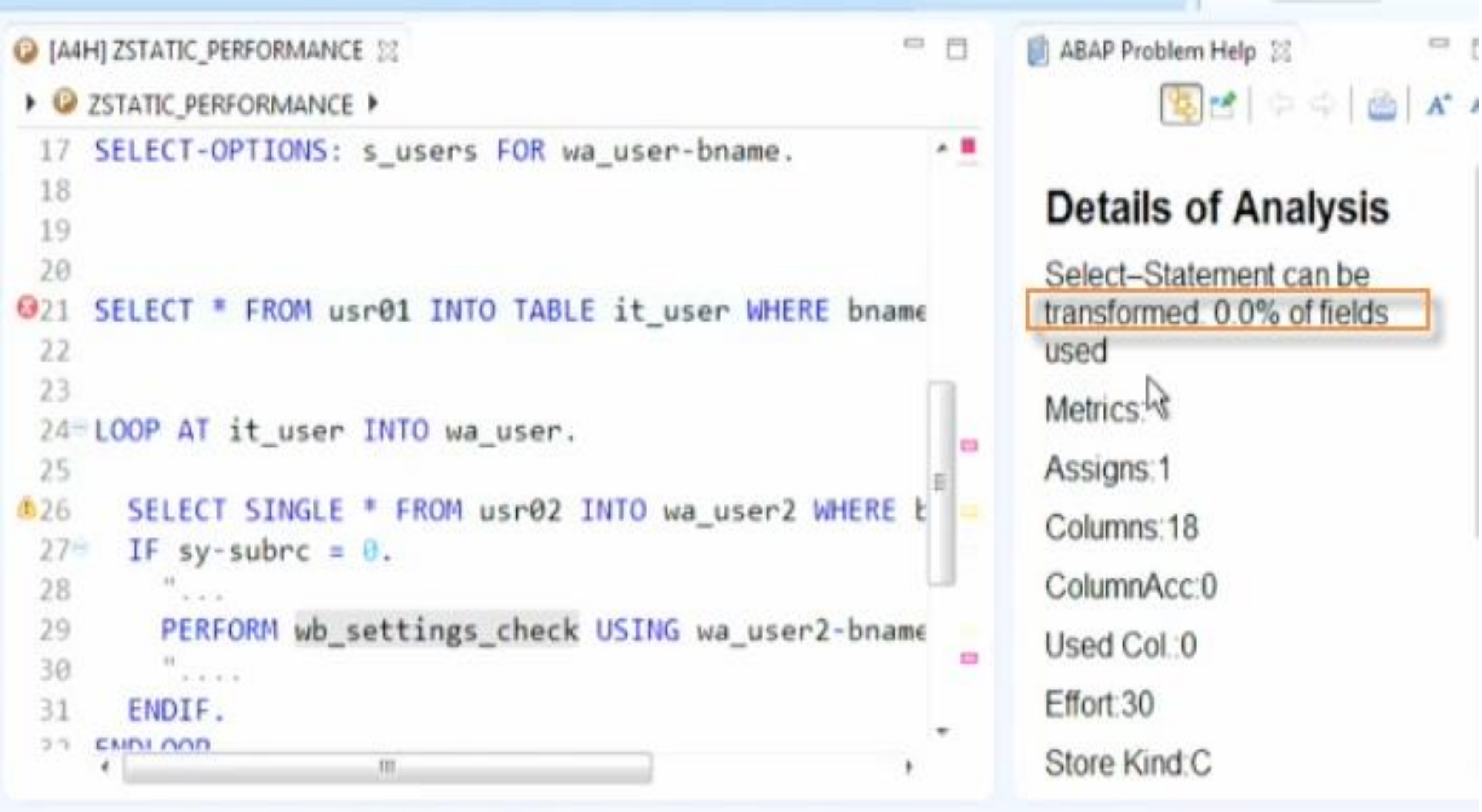
```
SELECT * FROM usr01 INTO TABLE it_user WHERE bname IN s_users


LOOP AT it_user INTO wa_user.

  SELECT SINGLE * FROM usr02 INTO wa_user2 WHERE bname = wa_us
  IF sy-subrc = 0.
    "...
    PERFORM wb_settings_che                              G rc
    "....
  ENDIF.
ENDLOOP
```

3 errors, 4 warnings, 0 others

Description

- Errors (3 items)
  - Existence check. No fields used: Searcl
  - NonLocal Nested Reading DB OP (...) f
  - Select-Statement can be transformed. ...% of fields used: Search problematic SELECT statem
- Warnings (4 items)

ABAP Analysis on HANA Studio



```
SELECT-OPTIONS: s_users FOR wa_user-bname.



SELECT * FROM usr01 INTO TABLE it_user WHERE bname


LOOP AT it_user INTO wa_user.

  SELECT SINGLE * FROM usr02 INTO wa_user2 WHERE b
  IF sy-subrc = 0.
    "...
    PERFORM wb_settings_check USING wa_user2-bname
    "....
  ENDIF.
ENDLOOP
```

Details of Analysis

Select–Statement can be transformed. 0.0% of fields used

Metrics:

Assigns:1

Columns:18

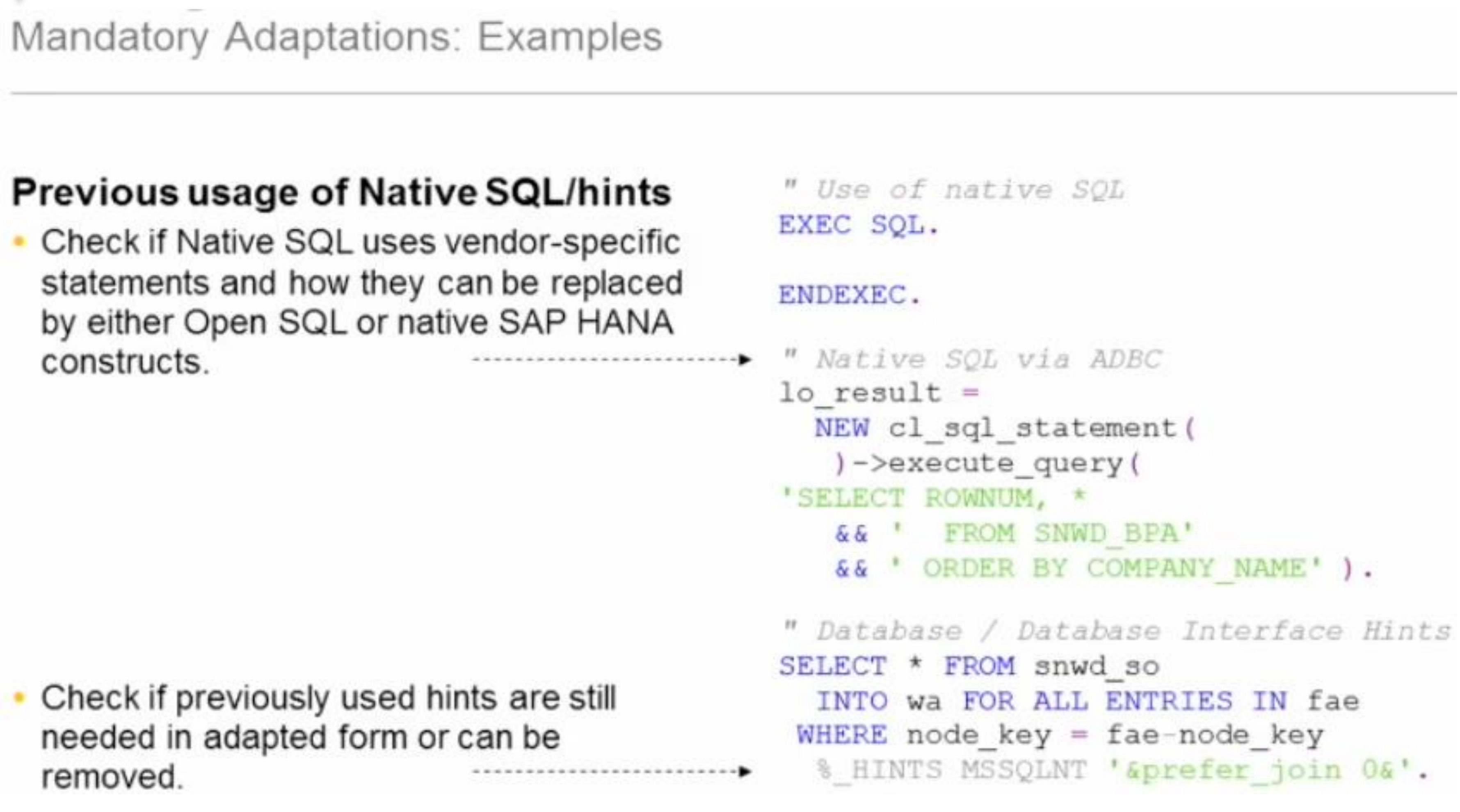
ColumnAcc:0

Used Col.:0

Effort:30

Store Kind:C

SAP HANA Mandatory Adaptations



Mandatory Adaptations: Examples

Previous usage of Native SQL/hints

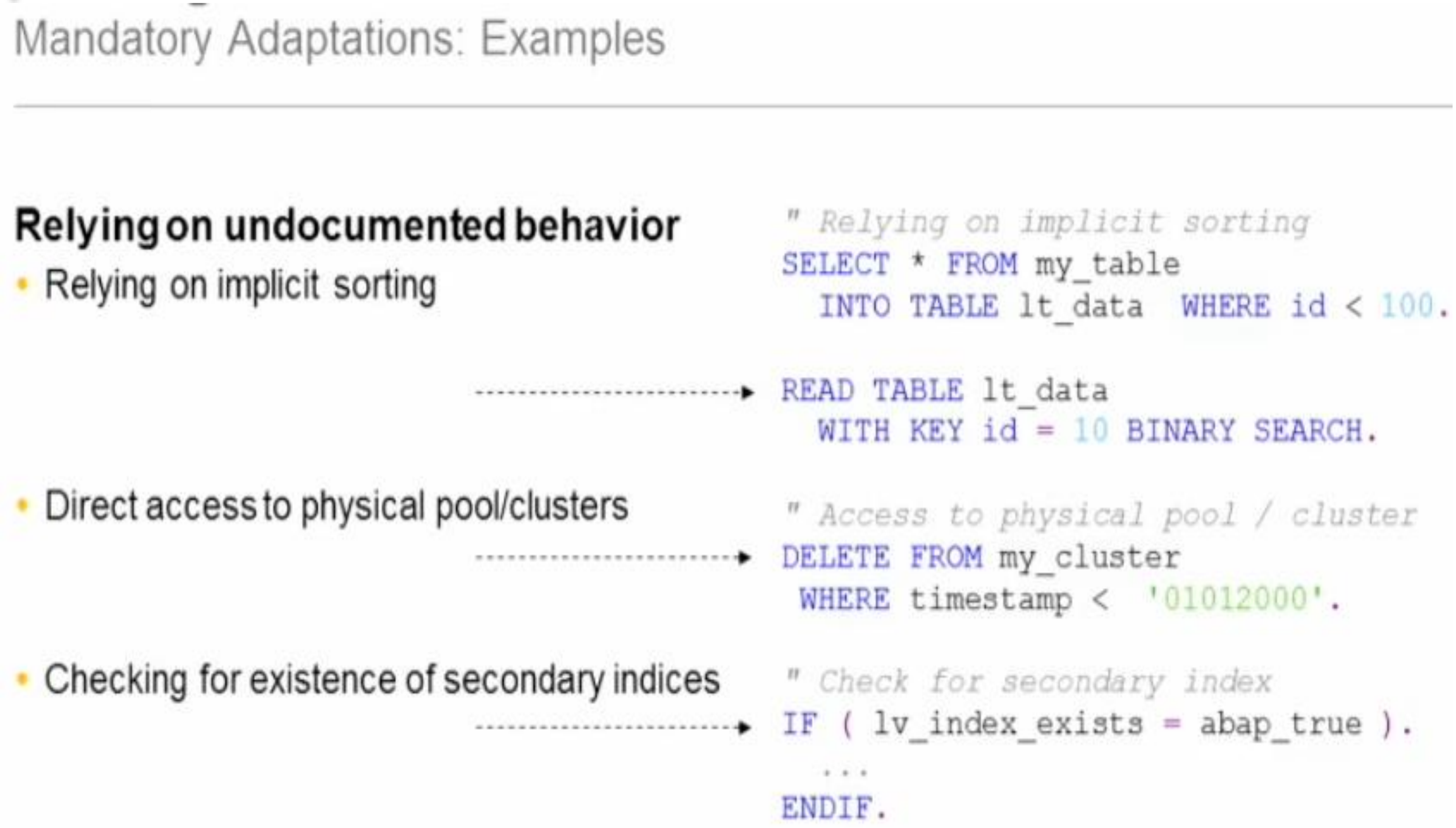
- Check if Native SQL uses vendor-specific statements and how they can be replaced by either Open SQL or native SAP HANA constructs.

```
" Use of native SQL
EXEC SQL.

ENDEXEC.

" Native SQL via ADBC
lo_result =
  NEW cl_sql_statement(
   )->execute_query(
'SELECT ROWNUM, *
   && '   FROM SNWD_BPA'
   && ' ORDER BY COMPANY_NAME' ).
```

- Check if previously used hints are still needed in adapted form or can be removed.

```
" Database / Database Interface Hints
SELECT * FROM snwd_so
  INTO wa FOR ALL ENTRIES IN fae
 WHERE node_key = fae-node_key
  %_HINTS MSSQLNT '&prefer_join 0&'.
```

SAP HANA Mandatory Adaptations



Mandatory Adaptations: Examples

Relying on undocumented behavior

- Relying on implicit sorting

```
" Relying on implicit sorting
SELECT * FROM my_table
  INTO TABLE lt_data  WHERE id < 100.

READ TABLE lt_data
  WITH KEY id = 10 BINARY SEARCH.
```

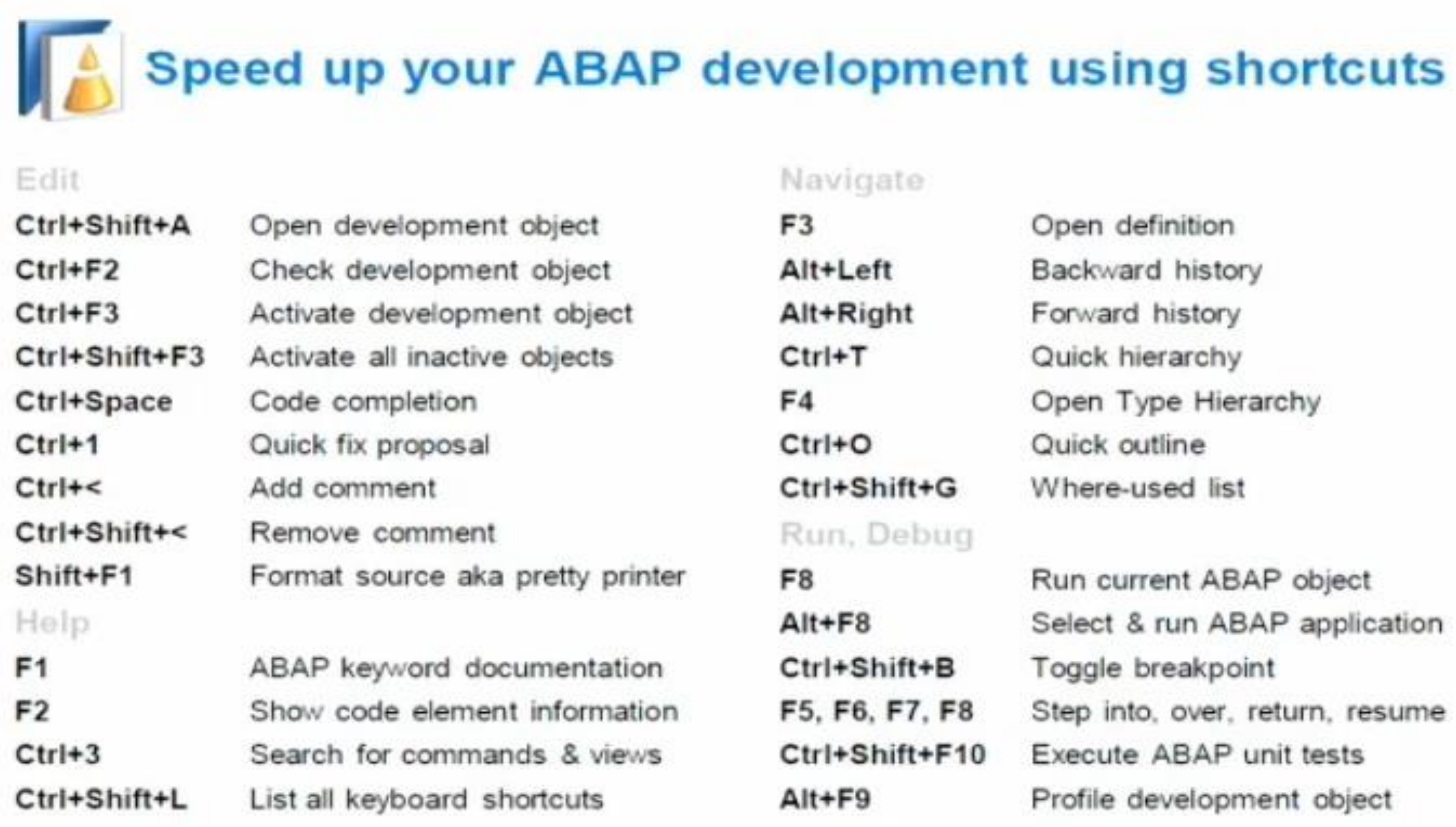
- Direct access to physical pool/clusters

```
" Access to physical pool / cluster
DELETE FROM my_cluster
 WHERE timestamp <  '01012000'.
```

- Checking for existence of secondary indices

```
" Check for secondary index
IF ( lv_index_exists = abap_true ).
  ...
ENDIF.
```

ABAP Development Shortcuts



Speed up your ABAP development using shortcuts

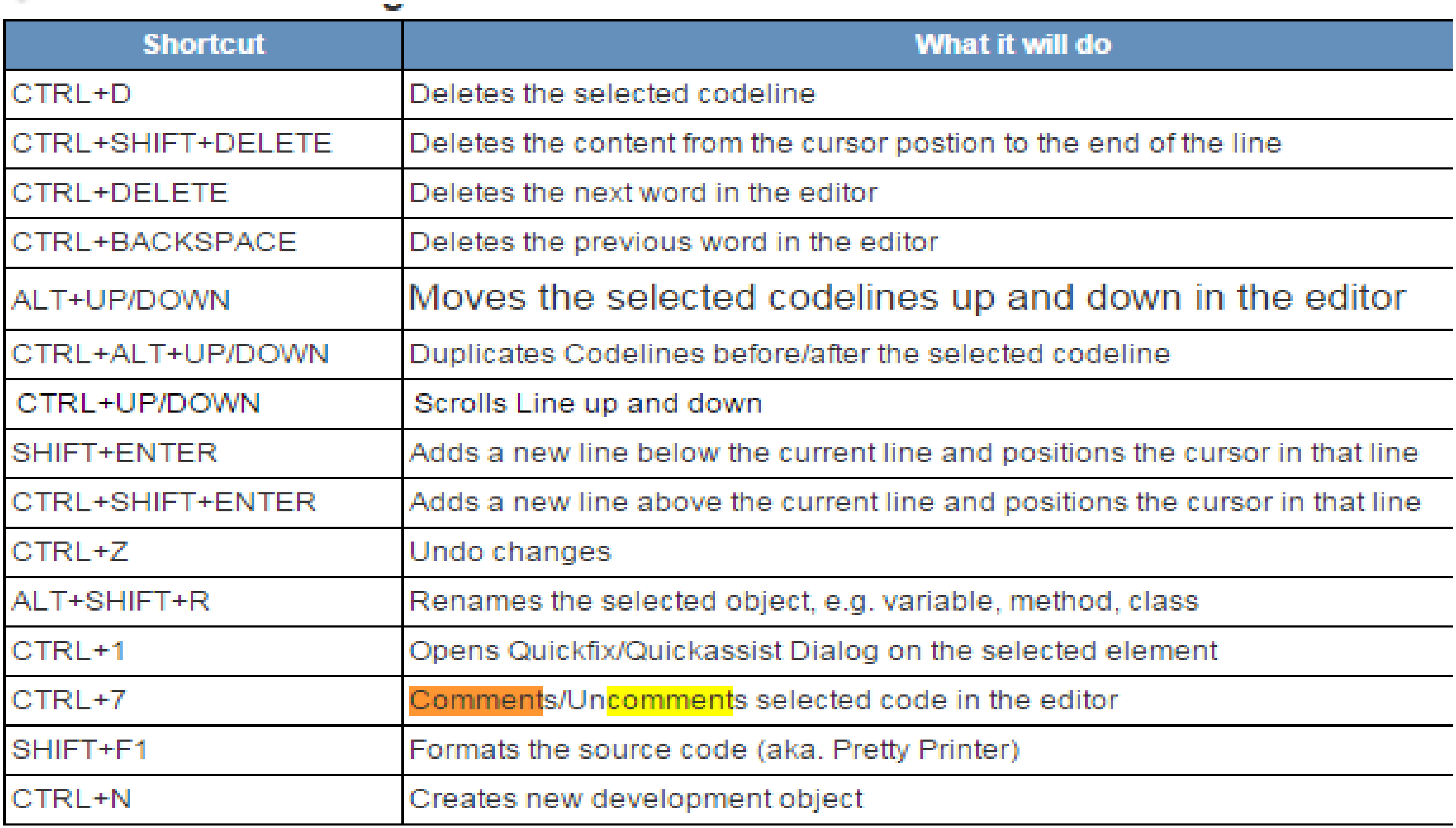
Edit	
Ctrl+Shift+A	Open development object
Ctrl+F2	Check development object
Ctrl+F3	Activate development object
Ctrl+Shift+F3	Activate all inactive objects
Ctrl+Space	Code completion
Ctrl+1	Quick fix proposal
Ctrl+<	Add comment
Ctrl+Shift+<	Remove comment
Shift+F1	Format source aka pretty printer

Help	
F1	ABAP keyword documentation
F2	Show code element information
Ctrl+3	Search for commands & views
Ctrl+Shift+L	List all keyboard shortcuts

Navigate	
F3	Open definition
Alt+Left	Backward history
Alt+Right	Forward history
Ctrl+T	Quick hierarchy
F4	Open Type Hierarchy
Ctrl+O	Quick outline
Ctrl+Shift+G	Where-used list

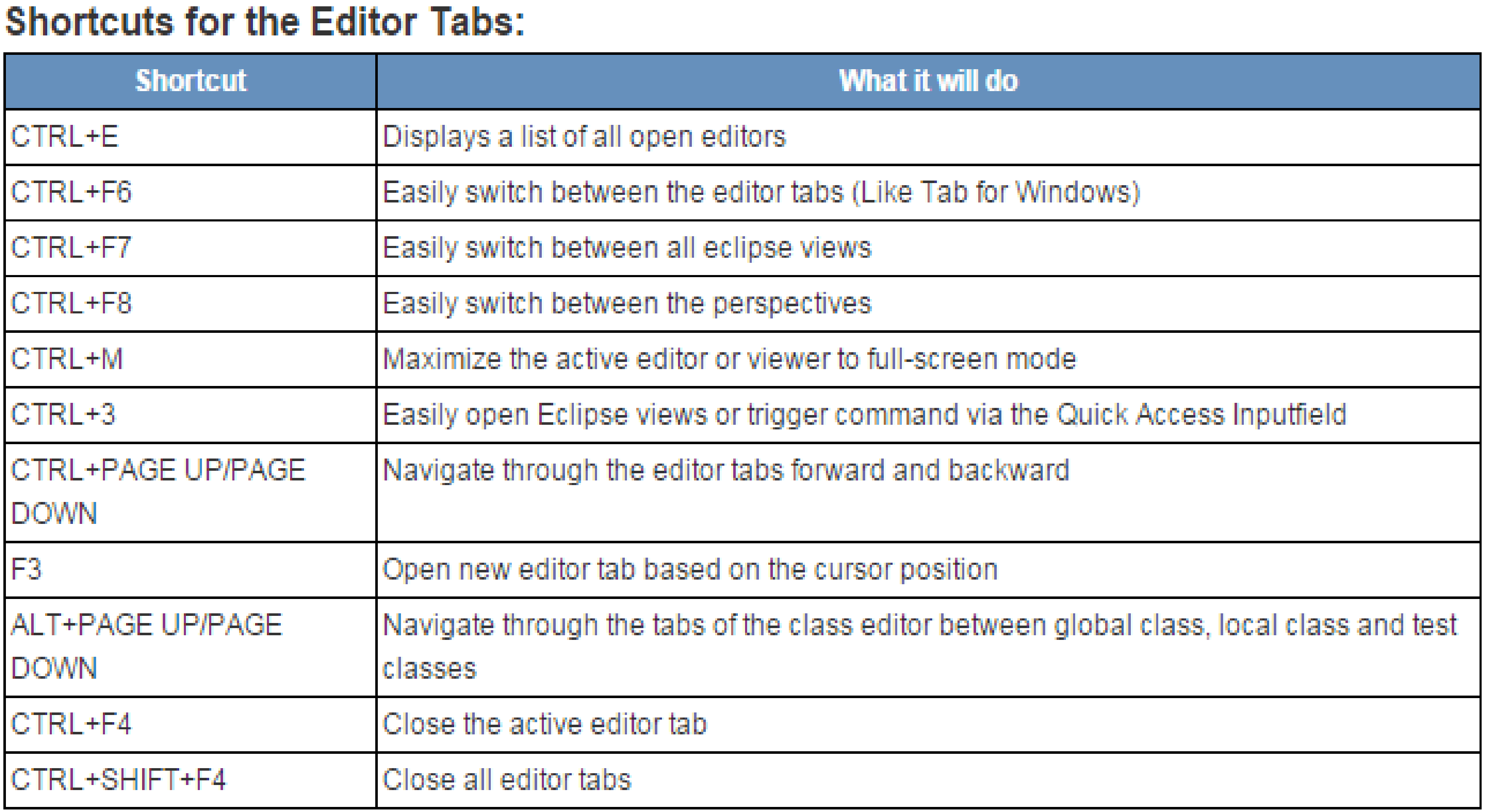
Run, Debug	
F8	Run current ABAP object
Alt+F8	Select & run ABAP application
Ctrl+Shift+B	Toggle breakpoint
F5, F6, F7, F8	Step into, over, return, resume
Ctrl+Shift+F10	Execute ABAP unit tests
Alt+F9	Profile development object

ABAP Development Shortcuts



Shortcut	What it will do
CTRL+D	Deletes the selected codeline
CTRL+SHIFT+DELETE	Deletes the content from the cursor postion to the end of the line
CTRL+DELETE	Deletes the next word in the editor
CTRL+BACKSPACE	Deletes the previous word in the editor
ALT+UP/DOWN	Moves the selected codelines up and down in the editor
CTRL+ALT+UP/DOWN	Duplicates Codelines before/after the selected codeline
CTRL+UP/DOWN	Scrolls Line up and down
SHIFT+ENTER	Adds a new line below the current line and positions the cursor in that line
CTRL+SHIFT+ENTER	Adds a new line above the current line and positions the cursor in that line
CTRL+Z	Undo changes
ALT+SHIFT+R	Renames the selected object, e.g. variable, method, class
CTRL+1	Opens Quickfix/Quickassist Dialog on the selected element
CTRL+7	Comments/Uncomments selected code in the editor
SHIFT+F1	Formats the source code (aka. Pretty Printer)
CTRL+N	Creates new development object

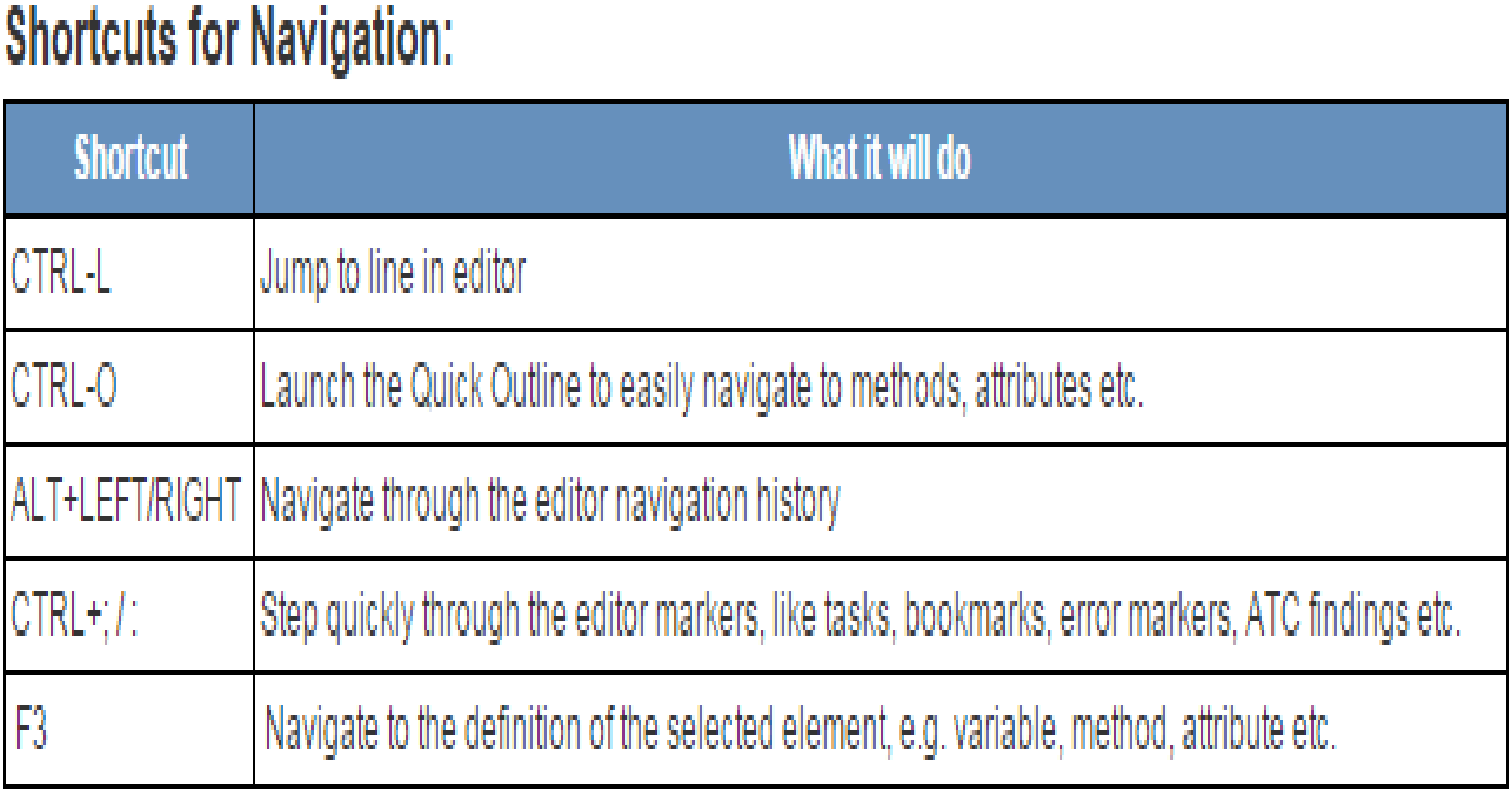
ABAP Development Shortcuts



Shortcuts for the Editor Tabs:

Shortcut	What it will do
CTRL+E	Displays a list of all open editors
CTRL+F6	Easily switch between the editor tabs (Like Tab for Windows)
CTRL+F7	Easily switch between all eclipse views
CTRL+F8	Easily switch between the perspectives
CTRL+M	Maximize the active editor or viewer to full-screen mode
CTRL+3	Easily open Eclipse views or trigger command via the Quick Access Inputfield
CTRL+PAGE UP/PAGE DOWN	Navigate through the editor tabs forward and backward
F3	Open new editor tab based on the cursor position
ALT+PAGE UP/PAGE DOWN	Navigate through the tabs of the class editor between global class, local class and test classes
CTRL+F4	Close the active editor tab
CTRL+SHIFT+F4	Close all editor tabs

ABAP Development Shortcuts



Shortcuts for Navigation:

Shortcut	What it will do
CTRL-L	Jump to line in editor
CTRL-O	Launch the Quick Outline to easily navigate to methods, attributes etc.
ALT+LEFT/RIGHT	Navigate through the editor navigation history
CTRL+; / :	Step quickly through the editor markers, like tasks, bookmarks, error markers, ATC findings etc.
F3	Navigate to the definition of the selected element, e.g. variable, method, attribute etc.

Summary

- By end of this course, participants know
  - Understand SAP HANA
  - To work with ABAP ON HANA
  - Understand SAP HANA System Architecture
  - To know SAP IN-Memory Strategy and Technology
  - To work with Row store and Column Store tables

Capgemini
CONSULTING.TECHNOLOGY.OUTSOURCING

Thank you

This presentation is for internal use only and not intended for external distribution. Its contents must be held in confidence.