Usage of in-memory column-based **SAP HANA** databases in enterprise information systems
SAP Labs **Czech Republic** in Brno

SAP ČR in Brno from **1995**, SAP Labs from **2016**

**S/4 HANA and Cloud development**

delivers innovative cross-platform web applications that are based on modern design principles and technologies.

**Globalization Services**

focus on developing country specific functions for SAP financial solutions

**Application Innovation Services**

supports all companies running SAP solutions with a strong focus on continuous innovation
Agenda

In-memory column store database

Examples

Additional features

SAP HANA Platform
In-memory column store database
Changes in Hardware
Performance bottleneck

<table>
<thead>
<tr>
<th>Type of Memory</th>
<th>Size</th>
<th>Latency (~)</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1 CPU Cache</td>
<td>64K</td>
<td>1 ns</td>
</tr>
<tr>
<td>L2 CPU Cache</td>
<td>256K</td>
<td>5 ns</td>
</tr>
<tr>
<td>L3 CPU Cache</td>
<td>8M</td>
<td>20 ns</td>
</tr>
<tr>
<td>Main Memory</td>
<td>GBs up to TBs</td>
<td>100 ns</td>
</tr>
<tr>
<td>Disk</td>
<td>TBs</td>
<td>&gt;1.000.000 ns</td>
</tr>
</tbody>
</table>
What is SAP HANA?
An Appliance of Hard- and Software

Hardware-Innovations

- Multi-core Architecture (8 x 10 core CPU / blade)
- Massive parallel scaling with many blades
- 64-bit address space – 2TB in current servers
- 100 GB/s data throughput
- Dramatic decline in price/performance

Software-Innovations

- Row and column store
- Compression
- Partitioning
- No aggregate tables
- Insert only on delta

Hardware-Innovations

Software-Innovations
Dictionary Encoding
Example

- 8 billion humans
- Each attribute is dictionary encoded

<table>
<thead>
<tr>
<th>recID</th>
<th>fname</th>
<th>Iname</th>
<th>gender</th>
<th>city</th>
<th>country</th>
<th>birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>39</td>
<td>John</td>
<td>Smith</td>
<td>m</td>
<td>Chicago</td>
<td>USA</td>
<td>12.03.1964</td>
</tr>
<tr>
<td>40</td>
<td>Mary</td>
<td>Brown</td>
<td>f</td>
<td>London</td>
<td>UK</td>
<td>12.05.1964</td>
</tr>
<tr>
<td>41</td>
<td>Jane</td>
<td>Doe</td>
<td>f</td>
<td>Palo Alto</td>
<td>USA</td>
<td>23.04.1976</td>
</tr>
<tr>
<td>42</td>
<td>John</td>
<td>Doe</td>
<td>m</td>
<td>Palo Alto</td>
<td>USA</td>
<td>17.06.1952</td>
</tr>
<tr>
<td>43</td>
<td>Peter</td>
<td>Schmidt</td>
<td>m</td>
<td>Potsdam</td>
<td>GER</td>
<td>11.11.1975</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

©2018 SAP SE or an SAP affiliate company. All rights reserved. | PUBLIC
Dictionary Encoding
Dictionary Encoding a Column

- A column is split into a dictionary and an attribute vector
- Dictionary stores all distinct values with implicit valueID
- Attribute vector stores valueIDs for all entries in the column
- Position is stored implicitly
- Enables offsetting with bit-encoded fixed-length data types

<table>
<thead>
<tr>
<th>recID</th>
<th>fname</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>39</td>
<td>John</td>
</tr>
<tr>
<td>40</td>
<td>Mary</td>
</tr>
<tr>
<td>41</td>
<td>Jane</td>
</tr>
<tr>
<td>42</td>
<td>John</td>
</tr>
<tr>
<td>43</td>
<td>Peter</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Dictionary for “fname”

<table>
<thead>
<tr>
<th>valueID</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>23</td>
<td>John</td>
</tr>
<tr>
<td>24</td>
<td>Mary</td>
</tr>
<tr>
<td>25</td>
<td>Jane</td>
</tr>
<tr>
<td>26</td>
<td>Peter</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Attribute Vector for “fname”

<table>
<thead>
<tr>
<th>position</th>
<th>valueID</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>39</td>
<td>23</td>
</tr>
<tr>
<td>40</td>
<td>24</td>
</tr>
<tr>
<td>41</td>
<td>25</td>
</tr>
<tr>
<td>42</td>
<td>23</td>
</tr>
<tr>
<td>43</td>
<td>26</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
# Dictionary Encoding

## Data Size Examples

<table>
<thead>
<tr>
<th>Column</th>
<th>Cardinality</th>
<th>Bits Needed</th>
<th>Item Size</th>
<th>Plain Size</th>
<th>Size with Dictionary (Dictionary + Column)</th>
<th>Compression Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>First names</td>
<td>5 millions</td>
<td>23 bit</td>
<td>50 Byte</td>
<td>400GB</td>
<td>250MB + 23GB</td>
<td>≈ 17</td>
</tr>
<tr>
<td>Last names</td>
<td>8 millions</td>
<td>23 bit</td>
<td>50 Byte</td>
<td>400GB</td>
<td>400MB + 23GB</td>
<td>≈ 17</td>
</tr>
<tr>
<td>Gender</td>
<td>2</td>
<td>1 bit</td>
<td>1 Byte</td>
<td>8GB</td>
<td>2b + 1GB</td>
<td>≈ 8</td>
</tr>
<tr>
<td>City</td>
<td>1 million</td>
<td>20 bit</td>
<td>50 Byte</td>
<td>400GB</td>
<td>50MB + 20GB</td>
<td>≈ 20</td>
</tr>
<tr>
<td>Country</td>
<td>200</td>
<td>8 bit</td>
<td>47 Byte</td>
<td>376GB</td>
<td>9.4kB + 8GB</td>
<td>≈ 47</td>
</tr>
<tr>
<td>Birthday</td>
<td>40000</td>
<td>16 bit</td>
<td>2 Byte</td>
<td>16GB</td>
<td>80kB + 16GB</td>
<td>≈ 1</td>
</tr>
<tr>
<td>Totals</td>
<td></td>
<td>200 Byte</td>
<td>≈ 1.6TB</td>
<td>≈ 92GB</td>
<td>≈ 17</td>
<td></td>
</tr>
</tbody>
</table>
Compression
Compression Techniques

- For attribute vector
  - Prefix encoding
  - Run length encoding
  - Cluster encoding
  - Sparse encoding
  - Indirect encoding
    - Sequence is partitioned into $N$ blocks of size $S$ (typically 1024)
    - If a block contains only a few distinct values an additional dictionary is used to encode the values in that block
    - Additionally: links to the new dictionaries + blocks that have a dictionary
Compression
Indirect Encoding

Example: fname column, table sorted by country

```
<table>
<thead>
<tr>
<th>Dictionary encoded attribute vector</th>
<th>block size = 1024</th>
<th>≈ 21.4GB</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 126 576 55 126 2 2 ... 55 881 212 3 19 461 792 45 ... 13 ...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8 bit</td>
<td>23 bit</td>
<td>23 bit</td>
</tr>
<tr>
<td>0 2 3 1 2 0 0 ... 1 881 212 3 19 461 792 45 ... 13 ...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Block 2 is not compressed

Assumption: each set of 1024 people of the same country contains on average 200 different names

Dictionaries: \((200 \times 23 \text{ bit} + 64 \text{ bit}) \times \#\text{blocks}\) \(4.2 \text{ GB}\)

Address of dictionary \((7.8 \text{ mio})\)

Compressed vector: \(8 \text{ billion} \times 8 \text{ bit}\) \(7.6 \text{ GB} \approx 11.8 \text{ GB}\)
Tuple Reconstruction
Row store

- All attributes are stored consecutively
- 200 byte $\rightarrow$ 4 cache accesses $\rightarrow$ 64 byte $\rightarrow$ 256 byte
- Read with 4MB/ms/core
- $\rightarrow$ $\approx 0.064 \mu s$ with 1 core

Table: world_population

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Country</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 8 x 10^9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data loaded and used
Data loaded but not used
Tuple Reconstruction
Column Store

- 1 cache access for each attribute
- 6 cache accesses à 64 byte \(\rightarrow 384\) byte
- Read with 4MB/ms/core
- \(\rightarrow \approx 0.096\) μs with 1 core

Table: world_population

- First Name
- Last Name
- Gender
- Country
- City
- Birthday

Data loaded and used
Data loaded but not used
Scan Performance
Row Store – Full Table Scan

- Table size 8 billion tuples × 200 bytes per tuple
  ≈ 1.6 TB
- Scan through all rows with 4 MB/ms/core
  → 400 s with 1 core
Scan Performance
Row Store – Stride Access “Gender”

Table: world_population

<table>
<thead>
<tr>
<th>First Name</th>
<th>Last Name</th>
<th>Gender</th>
<th>Country</th>
<th>City</th>
<th>Birthday</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row 8 x 10^9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- 8 billion cache accesses à 64 byte ≈ 512 GB
- Read with 4 MB/ms/core → 128 s with 1 core
Scan Performance
Column Store – Full Column Scan „Gender“

Table: world_population

- First Name
- Last Name
- Gender
- Country
- City
- Birthday

- Size of attribute vector “Gender”:
  8 billion tuples × 1 bit per tuple
  ≈ 1 GB

- Scan through column with 4MB/ms/core
  \( \rightarrow 0.25 \text{s with 1 core} \)
Database Operations

INSERT – example (With New Dictionary Entry)

**INSERT INTO world_population VALUES** (Karen, Schulze, f, GER, Rostock, 06-20-2014)

1. Look-up on dictionary → no entry found
2. Append new value to dictionary
3. Sort Dictionary
4. Change valueIDs in attribute vector
5. Append new valueID to attribute vector
DELETE FROM world_population WHERE fname = “Jane” and lname = “Doe”
Database Operations

UPDATE

**UPDATE** world_population **SET** city = „Bamberg“
**WHERE** fname = “Hanna” **AND** lname = “Schulze”

Combination of DELETE and INSERT operation

1. Look-up „Bamberg“ in dictionary → entry not found
2. Append new value „Bamberg“ to dictionary
3. Reorganize dictionary
4. Replace old values with new values in attribute vector (expensive)
Examples
Performance measurement
Examples

- System QM0 – 48 TB / 1100 CPUs

<table>
<thead>
<tr>
<th>Table</th>
<th>Store</th>
<th>Rows</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDOCA_C</td>
<td>Column</td>
<td>110 million</td>
<td>5 GB</td>
<td>1,8 s</td>
</tr>
<tr>
<td>ACDOCA_R</td>
<td>Row</td>
<td>110 million</td>
<td>240 GB</td>
<td>22,5 s</td>
</tr>
<tr>
<td>ACDOCA_sm</td>
<td>Column</td>
<td>5 million</td>
<td>140 MB</td>
<td>0,3 s</td>
</tr>
<tr>
<td>ACDOCA</td>
<td>Column</td>
<td><strong>19,5 billion</strong></td>
<td>1,3 TB</td>
<td>139 s</td>
</tr>
<tr>
<td>CDHR</td>
<td>Column</td>
<td>31 million</td>
<td>1,3 GB</td>
<td>12,4 s</td>
</tr>
<tr>
<td>CDPOS</td>
<td>Column</td>
<td>730 million</td>
<td>44 GB</td>
<td></td>
</tr>
</tbody>
</table>

- System HANA Express edition (VM) – 16 GB / 4 CPUs

<table>
<thead>
<tr>
<th>Table</th>
<th>Store</th>
<th>Rows</th>
<th>Size</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACDOCA_sm</td>
<td>Column</td>
<td>5 million</td>
<td>140 MB</td>
<td>0,9 s</td>
</tr>
</tbody>
</table>
Application improvements
CO-PA Accelerator: Result provided by a Customer

- COPA Accelerator implemented within 8 weeks (including test & production landscape).
- No manual modeling or creation of analytical views in HANA needed.
- Only replication of transactional CO-PA data needed.

- Some figures about data volume:
  - Total records in HANA: 550 Mil
  - Total volume in HANA: 30 GB
  - Total volume in ERP DB2:
    - 580 GB uncompressed,
    - 140 GB compressed (on disk)
- Number of posted records/day:
  - 100.000 - 200.000
- Initial replication took ~24 hours
Application improvements
CO-PA Accelerator: Top Down Accelerating Period-End Closing

Total runtime reduced by 2-50% maximum (case-dependent)

Total runtime reduced up to 80%
# Application improvements

**CO-PA Accelerator: KE28 – Validation with Productive Customer-Data**

<table>
<thead>
<tr>
<th>CO-PA Data</th>
<th>KE28 w/o SAP HANA</th>
<th>HANA-optimized KE28</th>
<th>Acceleration in Factors</th>
<th>Acceleration in %</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Top-Down-Distribution Level 1</td>
<td>5.880 sec</td>
<td>184 sec</td>
<td>32</td>
<td>97 %</td>
</tr>
<tr>
<td>• 6 Variants with Postings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 10 Variants without Postings</td>
<td>7.550 sec</td>
<td>194 sec</td>
<td>39</td>
<td>97 %</td>
</tr>
<tr>
<td>Top-Down-Distribution Level 2</td>
<td>25.096 sec</td>
<td>13.282 sec</td>
<td>2</td>
<td>50 %</td>
</tr>
<tr>
<td>• 13 Variants with Postings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 181 Varianten without Postings</td>
<td>64.557 sec</td>
<td>1.782 sec</td>
<td>36</td>
<td>97 %</td>
</tr>
<tr>
<td>Total Runtime</td>
<td>28,6 h</td>
<td>4,3 h</td>
<td>7</td>
<td>86 %</td>
</tr>
</tbody>
</table>

- Existing KE28 Variants will be accelerated with no changes to customizing or job-scheduling up to factor 40
- Significant unload of primary DB during period-end closing activities
Additional features
SAP HANA holds the bulk of its data in memory for maximum performance, but still uses persistent storage to provide a fallback in case of failure.

- During normal operation, data is automatically saved from memory to disk at regular savepoints. Additionally, all data changes are captured in redo log entries. A redo log entry is written to disk after each committed database transaction.
- Support for multitenant database containers
- Apply to all of SAP HANA, both the hot and the warm store
- Backint enables 3rd party tool vendors to directly connect backup agents
Multitenant and scale

Multitenant Data Base

- A single SAP HANA can contain several isolated databases
- The tenant databases share computing resources (RAM, CPU), SW installation, system administration (start/stop system).
- The tenant database has their own metadata, data, and users.

Scale the system

- More data -> more RAM -> more CPUs
- Scale up: one system up 20 CPU and 20 TB RAM.
- Scale out: combining multiple independent nodes into one system (supporting multitenant).
Dynamic Tiering

Data temperatures

Data tiering is the assignment of data to various tiers/storage media based upon data type, performance requirements, frequency of access.

- **Hot Store**
  - This tier is used to store mission-critical data for real-time processing and real-time analytics.
  - Data is retained “in-memory”.

- **Warm Store**
  - This tier is used to store data with reduced performance SLAs, which is less frequently accessed.
  - Data is stored on a lower cost storage tier, managed as a unified part of the SAP HANA database.

- **Cold Store**
  - This tier is used to store voluminous data for sporadic or very limited access.
  - Data is stored on disk, in columnar structures on SAP IQ or in Hadoop HDFS.
SAP HANA Platform
SAP HANA: The business data platform for the intelligent enterprise

SAP HANA PLATFORM

APPLICATION DEVELOPMENT
- Web Server
- JavaScript
- Fiori UX
- Graphic Modeler
- Application Lifecycle Management

ADVANCED ANALYTICAL PROCESSING
- Spatial
- Graph
- Predictive
- Search
- Text Analytics
- Streaming Analytics
- Series Data
- Business Functions

DATA INTEGRATION & QUALITY
- Data Virtualization
- ELT & Replication
- Data Quality
- Hadoop & Spark Integration
- Remote Data Sync

DATABASE MANAGEMENT
- Columnar
- Multi-Core & Parallelization
- Advanced Compression
- Multi-tenancy
- Multi-Tier Storage
- Data Modeling
- Openness
- Admin & Security
- High Availability & Disaster Recovery
Graph
Definition

- The property graph model provides directed, attributed multi-relation graphs
- Use cases
  - Social network, company/organizations, production and supply chains, citation networks, authorization and role concepts, knowledge graphs...
- SAP HANA supports data graph processing directly in the server.
- Built-in functions like shortest path, get neighborhood, topological analysis of complete graph
- Support for pattern matching using openCypher
- GraphScript to develop custom graph algorithms
Graph

Code example

A GRAPH WORKSPACE exposes the data to the graph engine

```sql
CREATE GRAPH WORKSPACE [SCHEMA].[NAME]
  EDGE TABLE [SCHEMA].[EDGE TABLE/VIEW]
    SOURCE COLUMN source
    TARGET COLUMN target
    KEY COLUMN id
  VERTEX TABLE [SCHEMA].[NODE TABLE/VIEW]
    KEY COLUMN id;
```

<table>
<thead>
<tr>
<th>ID</th>
<th>TYPE</th>
<th>NAME</th>
<th>YEAR</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUT-6841</td>
<td>Author</td>
<td>Richardson, Fred</td>
<td></td>
</tr>
<tr>
<td>H94-1009</td>
<td>Paper</td>
<td>The Hub and Spoke Paradigm for CSR Evaluation</td>
<td>1994</td>
</tr>
<tr>
<td>ORG-523</td>
<td>Organization</td>
<td>Boston University</td>
<td></td>
</tr>
<tr>
<td>H92-1076</td>
<td>Paper</td>
<td>Spontaneous Speech Collection for the CSR Corpus</td>
<td>1992</td>
</tr>
</tbody>
</table>

- MATCH (A)-[e1]->(P1), (P1)-[e2]->(P2)
  WHERE A.NAME = ‘Fred’
  AND P1.TYPE = ‘Paper’
  AND e2.TYPE = ‘citation’
  RETURN P2.TITLE AS TITLE
**Graph**

Customer example

Customer collects and analyze data about companies, people and their connections.

**Graph size**
- 24 Mio Nodes (organizations, persons)
- 125 Mio. Edges (owns, knows, etc.)

**UBO (ultimate beneficial owner) description**
- All persons owning 25% or more of a company are UBO
- All persons “controlling” a company which owns 25% or more are UBO

**UBO implementation with HANA Graph**
- ~15 lines of code
- 5 minutes to identify all UBOs for millions of companies
Spatial data
Types and functions

SAP HANA provides native spatial data processing

- Natively store 2D, 3D and 4D vector data types (x, y, z, m)
- Over 80 native SQL based geospatial functions
- Open standards (OGC, 1999 SQL/MM)

Vector data

```
"POINT".ST_Within("RECTANGLE") = 1
```
Spatial data
Code example

- CREATE COLUMN TABLE shapes (  
  id BIGINT,  
  description NVARCHAR(100),  
  shape ST_GEOMETRY(4326) );

- INSERT INTO shapes VALUES (1, 'a', new ST_Point('POINT(1.6, 2.0)', 4326));
- INSERT INTO shapes VALUES (3, 'c', new ST_Polygon('Polygon((0 0, 1 0, 1 1,0 1,0 0))', 4326));

- SELECT id, description, shape.ST_AsSVG(), shape.ST_Area()  
  FROM shape  
  WHERE shape.ST_Intersects( new ST_POLYGON(  
    'POLYGON((0 0, 3 0, 3 3, 0 3, 0 0))' ) ) = 1  
  AND shape.ST_GeometryType() = 'ST_Polygon';
Airlines need real-time insights into flight operations of several thousand flights per day and be situational-aware of meteorological conditions which can result in cancellations or delay of flights. Airlines need the ability to manage airline traffic in real-time with a global view and provide decision-support to flight dispatchers and pilots to find alternative trajectories while minimizing costs.

10x faster
SAP HANA, express edition

SAP HANA, express edition is a database and application development platform. You can run it for free (up to 32GB of RAM) on your laptop and start building new apps.
Resources
Resources

- SAP HANA Academy Videos: https://www.youtube.com/user/saphanaacademy
Thank you.

Contact information:

Radim Benek
Development Expert, SAP Labs Czech Republic
SAP CR, spol. s r.o.
Holandská 6/8
639 00 Brno
radim.benek@sap.com