Institut für Informationssysteme
Technische Universität Braunschweig

Data Warehousing & Mining Techniques

Wolf-Tilo Balke
Muhammad Usman
Institut für Informationssysteme
Technische Universität Braunschweig
http://www.ifis.cs.tu-bs.de

Summary

- Last Week: DW-Project
- ETL Process

- This Lecture: Real-Time DW

8. Real-Time DW

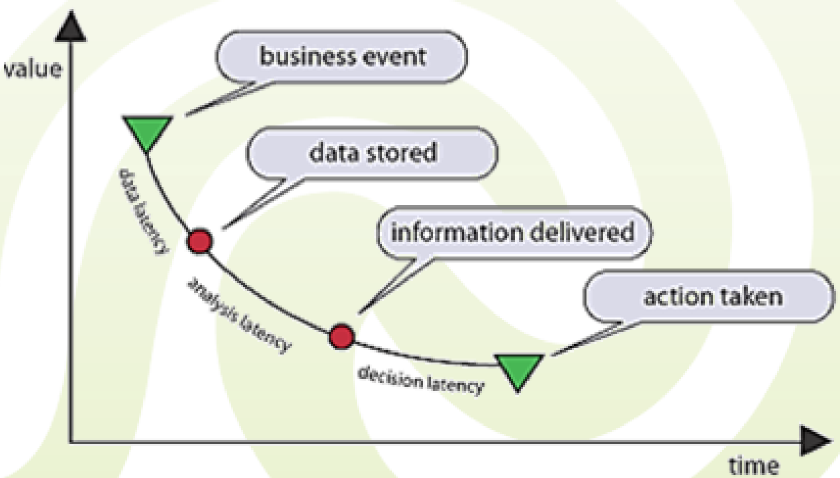
8. Real-Time Data Warehouses

8.1 Real-Time DW - Why?

- **Customer business scenario:** a utility company owns plants generating energy
- Existing DW supports planning by recommending:
  - The production capacity
  - The reserve capacity
  - When to buy supplemental energy, as needed

8.1 Real-Time DW - Why?

- Each day is pre-planned on **historical behavior**
  - Peak demand periods are somewhat predictable
- Good planning is important because:
  - Expensive to have unused capacity!
  - Cheaper to buy energy ahead!
- **Planning on last week's average is not enough**



8.1 Real-Time DW - Why?

- Getting more **in-time** accuracy enhances operational business
  - Compare **today's** plant output and customer consumption volumes to yesterday’s or last week’s average
  - Know when to purchase additional options or supplies
- Customer Target: have the actual data from the operational environment available for analytics within a **5 minute lag**
- **Real-time ≠ fast**
  - Real time DW has the capability to **enforce** time constraints

8.1 Real-Time DW - Why?

- The most difficult part for complying with the 5 minutes time constraint is the **ETL process**
  - ETL tools usually operate in batch mode on a certain schedule nightly, weekly or monthly
  - ETL typically involves **downtime** for the DW during the loading step (usually happens over night)
  - Data is extracted into flat files in the staging area **outside the DBMS**, where it is not available for querying
  - ETL may take **hours** to finish
- The ETL process needs to be re-designed to meet real-time constraints

8.2 Real-Time ETL

- Solutions enabling **real-time ETL**
  - Microbatch
  - Direct Trickle-feed
  - Trickle & Flip
  - External Real-time Data Cache (ERDC)

8.2 Microbatches

- **Microbatches** relies on the classical ETL-batch solution to provide **near-real-time ETL**
  – Reduce the time interval between consecutive loads (typical intervals are 3-4 hours)
  – Transformation tasks have to be fully automatized (no human intervention)
  – However, the time interval can't be reduced to minutes
  – The interval depends on:
    - The **data volume**
    - The **operational system and OLAP load** during the microbatch process

8.2 Direct Trickle-Feed

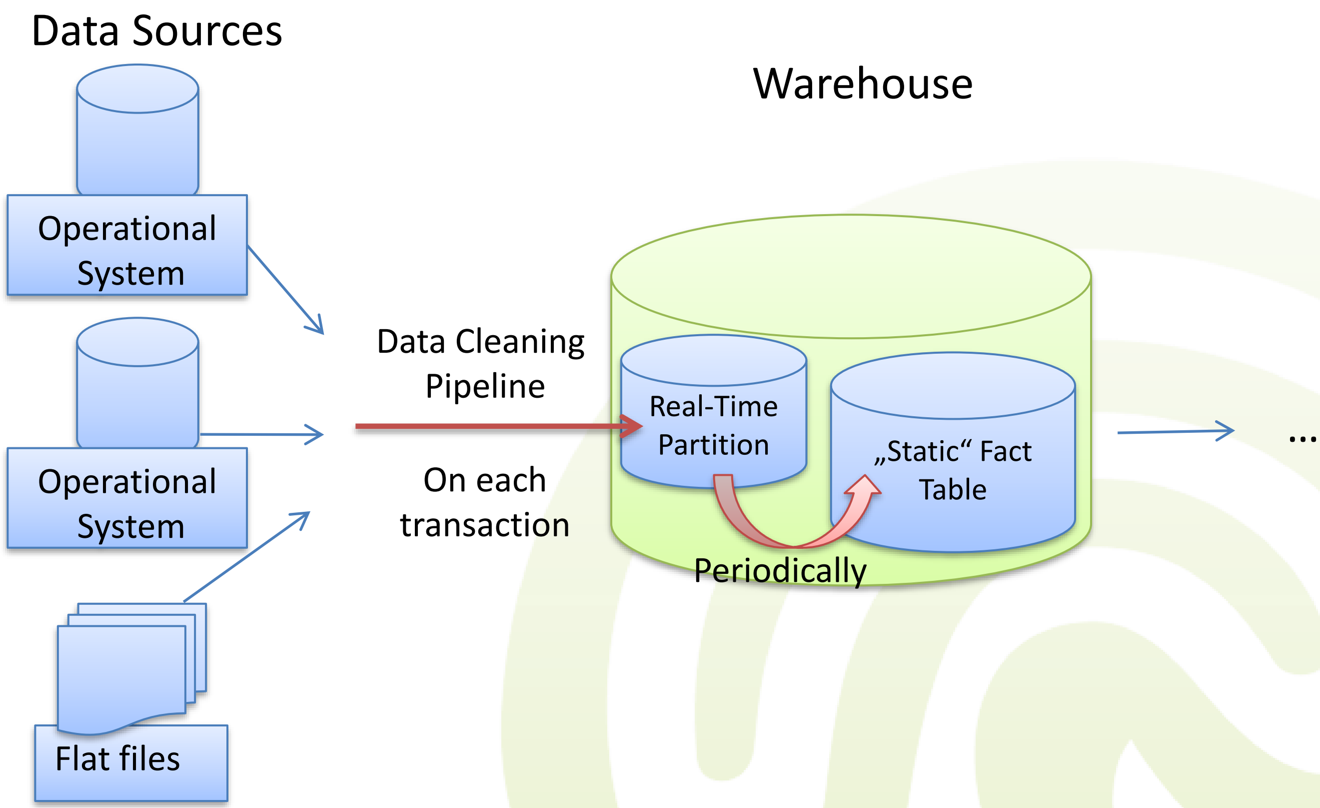
- **Direct Trickle-Feed** (also called continuous feed)
  - **Continuously** feed the DW with new data from the operational environment
    - Directly propagate changes on data from the OD store as inserts into the DW fact table
  - But **constantly updating** the same tables being queried by a **reporting or OLAP** tool can cause the DW's query **performance** to degrade – **query contention**
    - Under moderate to heavy usage from either OLAP queries or the incoming data, most RDMS will block the incoming data – the **Walmart scenario**

8.2 Direct Trickle-Feed

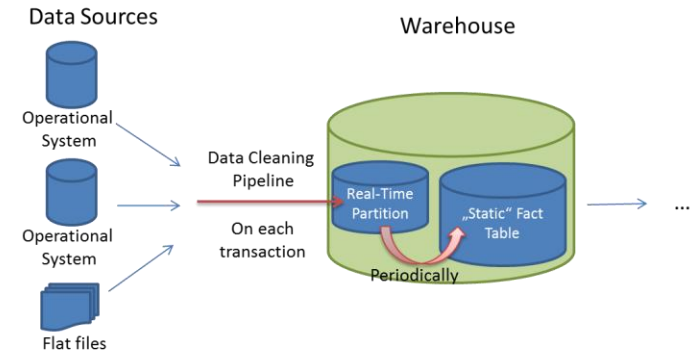
- Solution: **separate real-time fact partition**
- Idea is using a separate table subject to special rules for update and query
  - Contains all the activity that occurs since the last update of the "static" (updated each night) fact table
  - Linked as seamlessly as possible to the content of the "static" fact table
  - Indexed only lightly to account for incoming data
  - Support highly responsive queries

8.2 Direct Trickle-Feed

- Real-time partition architecture



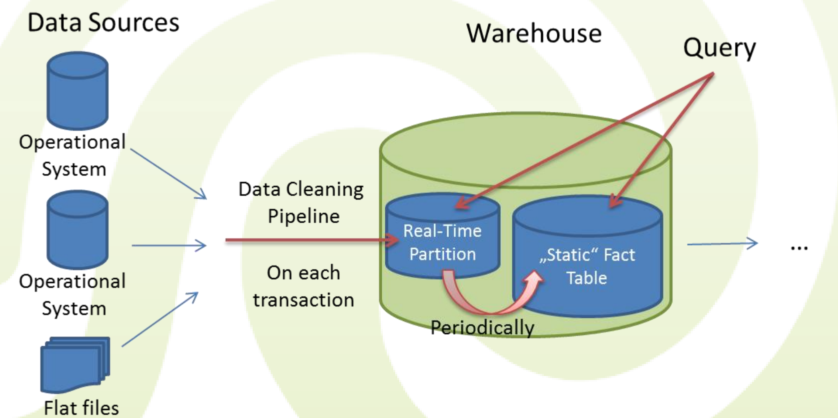
8.2 Real-Time Partition in Practice



- The Amazon DW:
  - An estimate of 55 GB sales transactions a year
    - For a day, this would mean about 150 MB of raw data
  - The "static" fact table just for the sales cube, for the last 3 years, would be about 150 GB
    - For fast OLAP querying it will be heavily indexed and supported by aggregates
  - The small 150 MB real-time partition can be pinned in memory for rapid loading and querying

8.2 Real-Time Partition in Practice

- The query: How did the sales for today evolve until now (Tuesday the 6th of December, at 10:30) compared to the last 3 Tuesdays?
  - The **query processor** detects that **fresh data** is required by the query
  - The query processor sends the same query to both the "static" fact table and to the real-time partition



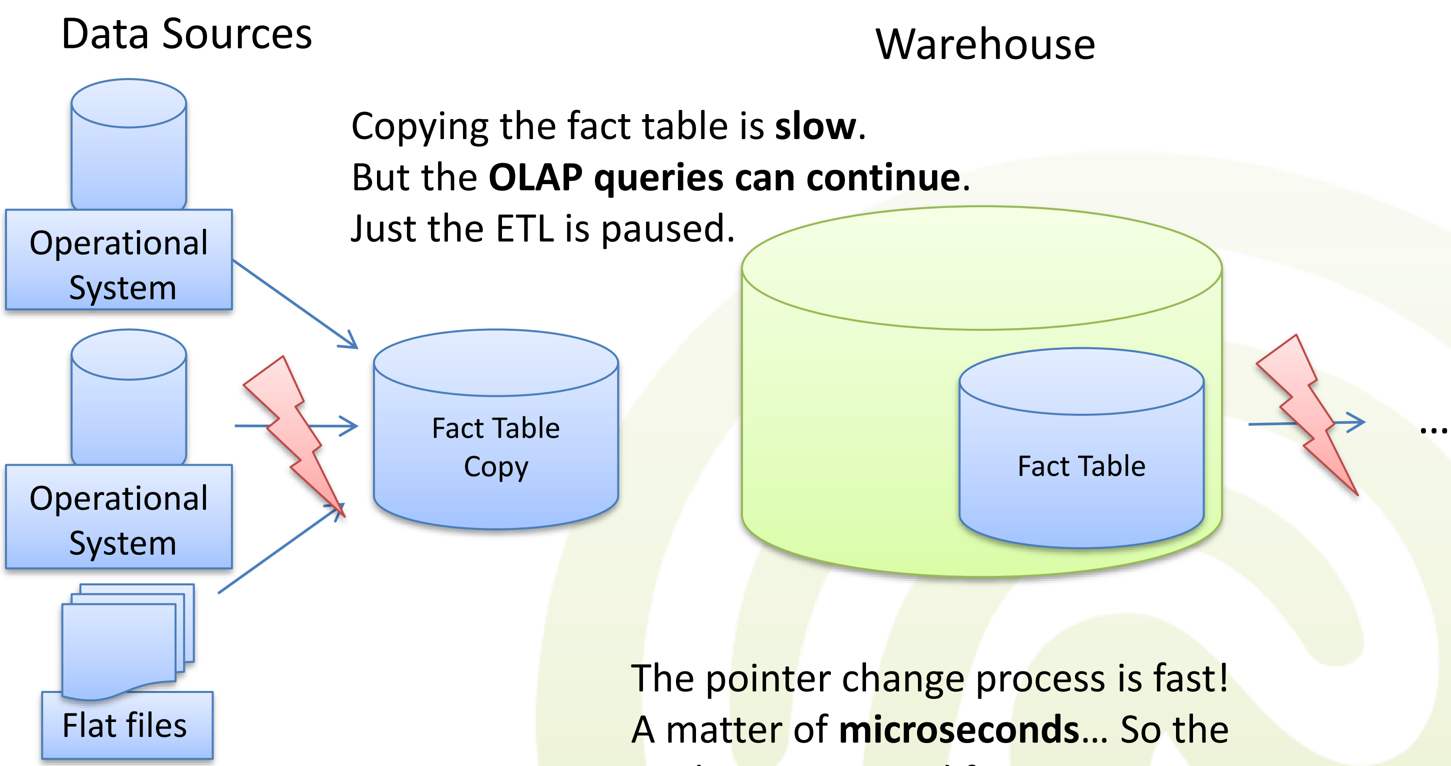
8.2 Real-Time Partition in Practice

- Data from the real-time partition is aggregated to the necessary granularity in concordance with the classification schema
- The system benefits from the **indexes** on the "static" fact table, and from the small **size** of the in-memory real-time partition
- As a downside, this approach may still suffer from **query contention**, if daily data volume is high

8.2 Real-Time ETL

- **Trickle & Flip**
  - Instead of continuously feeding the data into the DW, the data is fed into the staging tables in the **exact same format as the target fact tables**
    - This way the staging tables are copies of the fact tables, with up-to-date data
  - Periodically, the trickle is halted, the staging tables are copied, and renamed to the active fact table names (just a pointer switch operation)
    - The active fact table is deleted, and the process begins anew

- Architecture:



Copying the fact table is **slow**.
But the **OLAP queries can continue**.
Just the ETL is paused.

The pointer change process is fast! A matter of **microseconds**... So the analysis is stopped for just a very short time during the **flip.**

8.2 Trickle & Flip

- The main problem is **scalability**
  - The staging tables must contain the **complete fact table**
  - If the fact table is too large the copying process is slow
  - This method is only useful if the frequency of refreshs corresponds to the time it takes to perform the flip
- **No query contention** between complex OLAP queries and loading inserts
  - During flipping queries are paused

8.2 ERDC

- **External Real-Time Data Cache (ERDC)**
  - Leave the DW largely as-is
  - Store the incoming real-time data outside the traditional DW in order to completely avoid performance problems
  - The data cache can simply be a dedicated database server for loading storing and processing real-time data

8.2 ERDC

- Two approaches querying the real-time data
  - **Separate & isolate** in the real-time data cache
    - All queries involving real-time data are redirected towards the ERDC – similar procedure as the direct trickle-feed
  - **Just-in-time information merge** from external data cache (JIM)
    - The real-time data required to answer any particular query is seamlessly imaged to the regular DW on a temporary basis – similar to trickle & flip

8.2 Separate & Isolate

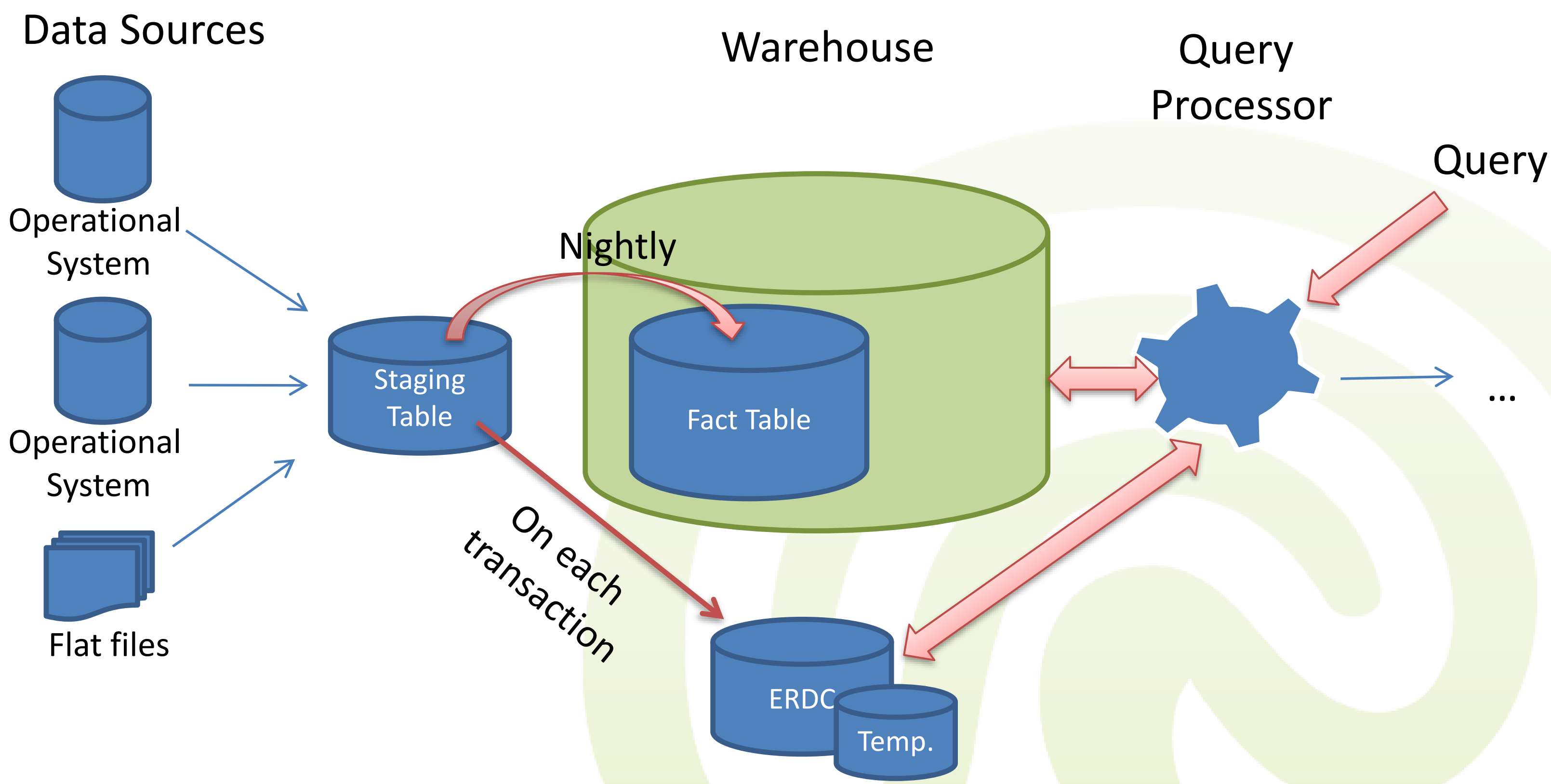
- All the real-time data loading and querying is routed to the external database
  - No scalability problem for the DW itself
  - The assumption is that the real-time daily data volume is small
  - Data is regularly (e.g., every night) loaded from the ERDC into the DW
- With the real-time data external to the DW it is not possible for a single report to **join real-time and historical** information

8.2 Just-in-Time Merge

- An extension of the Separate & Isolate method
- Workflow at query time
  - Queries are pre-processed to determine which **real-time data** parts are **required** (attributes & rows)
  - A snapshot of these required parts is taken and loaded as temporary tables in the DW
  - Once the tables are loaded in the DW, the **query is re-written** to also include the data from the temporary tables

8.2 Just-in-Time Merge

- Architecture:



8.2 Just-in-Time Merge

- Advantages of JIM
  - Less scalability problems as the real-time data is brought into the DW only on request
  - **Query contention** is not a problem as the data in the temporary tables are snapshots and **do not change while queried**

- Problem: the query processor is very complex

8.2 Real-Time ETL

- **Choose** one of the real-time ETL enabling methods based on your needs!
  - Microbatch offers the easy/cheap way out, when near real-time ETL satisfies the requirements (3-4 hours)
  - Direct Trickle-feed with real-time partitions offers real-time performance
    - It works fine for normal daily data loads
    - But may suffer from **query contention** for complex OLAP queries and large number of users

8.2 Real-Time ETL

- Trickle & Flip offers near real-time performance
  - The size of the delay is dictated by the size of the fact table – more precisely by the overhead of making a copy of the fact table
- External Real-time Data Cache (ERDC) with Just-in-Time Merge offers true real-time performance while allowing for
  - Rapidly changing data (10-1000 transactions/sec from the ODS)
  - Concurrent access for hundreds of users
  - Conjunction of real-time and historical data

- ERDC with Just-in-Time Merging sounds great…but **it's not supported by any product yet!**

8.3 OLAP on Changing Data

- OLAP queries were designed to operate on top of **unchanging, static, historical data**
  - No precautions are taken to ensure **consistency**
    - Data changes concurrent to execution may negatively influence the results of OLAP queries
  - This leads to **inconsistent results** and **confusing reports**

- For a *sales by product category* report an OLAP tool issues the following multi-pass SQL statement

```
0:00 create table TEMP1 (
        Category_Id LONG, DOLLARSALES DOUBLE)

0:01 insert into TEMP1
      select     a11.[Category_Id] AS Category_Id,
        sum(a11.[Tot_Dollar_Sales]) as DOLLARSALES
      from       [YR_CATEGORY_SLS] a11
      group by   a11.[Category_Id]

0:05 create table TEMP2 ( ALLPRODUCTSD DOUBLE)

0:06 insert into TEMP2
      select sum(a11.[Tot_Dollar_Sales]) as
      ALLPRODUCTSD
      from       [YR_CATEGORY_SLS] a11

0:08 select distinct pa1.[Category_Id] AS Category_Id,
        a11.[Category_Desc] AS Category_Desc,
        pa1.[DOLLARSALES] as DOLLARSALES,
        (pa1.[DOLLARSALES] / pa2.[ALLPRODUCTSD])
          as DOLLARSALESC
      from       [TEMP1] pa1,
                 [TEMP2] pa2,
                 [LU_CATEGORY] a11
      where      pa1.[Category_Id] =
      a11.[Category_Id]

0:09 drop table TEMP1
0:10 drop table TEMP2
```

Category	Metrics Dollar Sales	Dollar Sales Contribution to all Products Abs.
Electronics	$39,915.00	19.7%
Food	$10,938.00	5.4%
Gifts	$36,362.00	18.0%
Health&Beauty	$11,707.00	5.8%
Household	$88,774.00	43.8%
Kid's Corner	$5,502.00	2.7%
Travel	$9,289.00	4.6%
Total	$202,487.00	100.0%

8.3 Example: OLAP on Changing Data

- Lets assume few larger sales are inserted in the DW between seconds 1 and 6
  - The sales by category from TEMP1 won't match the sum from TEMP2
  - The report will then show inconsistent values

Category	Metrics Dollar Sales	Dollar Sales Contribution to all Products Abs.
Electronics	$39,915.00	19.2%
Food	$10,938.00	5.3%
Gifts	$36,362.00	17.5%
Health&Beauty	$11,707.00	5.6%
Household	$88,774.00	42.8%
Kid's Corner	$5,502.00	2.7%
Travel	$9,289.00	4.5%
Total	$207,487.00	97.6%

```
0:00 create table TEMP1 (
       Category_Id LONG, DOLLARSALES DOUBLE)

0:01 insert into TEMP1
     select    a11.[Category_Id] AS Category_Id,
       sum(a11.[Tot_Dollar_Sales]) as DOLLARSALES
     from      [YR_CATEGORY_SLS] a11
     group by  a11.[Category_Id]

0:05 create table TEMP2 ( ALLPRODUCTSD DOUBLE)

0:06 insert into TEMP2
     select sum(a11.[Tot_Dollar_Sales]) as
     ALLPRODUCTSD
     from      [YR_CATEGORY_SLS] a11

0:08 select distinct pa1.[Category_Id] AS Category_Id,
       a11.[Category_Desc] AS Category_Desc,
       pa1.[DOLLARSALES] as DOLLARSALES,
       (pa1.[DOLLARSALES] / pa2.[ALLPRODUCTSD])
         as DOLLARSALESC
     from      [TEMP1] pa1,
               [TEMP2] pa2,
               [LU_CATEGORY] a11
     where     pa1.[Category_Id] =
     a11.[Category_Id]

0:09 drop table TEMP1
0:10 drop table TEMP2
```

Sales of $5000 for different products

8.3 OLAP on Changing Data

- Is such a scenario possible?
  - OLAP queries may have response times ranging from seconds up to hours!
  - It is not uncommon for OLAP reports to consist of 10-50 passes!
- Such a scenario is not only possible but **most probable** and a **problem for real-time DW**
  - True real-time solutions like the Direct Trickle-feed with Real-Time Partition are affected by this problem

8.3 OLAP on Changing Data

- Real-time ETL solutions not affected by the inconsistency problem
  - Near real-time solutions like Microbatch and Trickle & Flip
    - Both work on "fresh" **copies** of data which are **not refreshed during OLAP queries**
  - The real-time solution ERDC Just-in-Time solution is also not affected
    - Because data from the external cache is **shadowed on demand** in the DW

8.4 Back to Basics

Detour

- OLTP and OLAP: How different are they?
  - Both involve a large number of selects
  - More range lookups for the OLAP and fewer inserts updates and deletes
  - But the OLAP queries span over **larger amounts of data** and **take longer** to resolve

8.4 Hybrid Solutions

Detour

- The approaches we have discussed until now are all **workarounds** for the **locking** constraints combined with **poor hardware performance**
  - OLAP queries take too long to execute
  - OLTP is locked out while performing OLAP
- Simple concept: Shadow the OLTP data in real-time, for OLAP to run on those shadows
  - Avoid **inconsistencies**
  - Limited by **yesterday's hardware**
- **Can we build a hybrid OLTP & OLAP solution with better hardware?**

8.4 Where is the Bottleneck? Detour

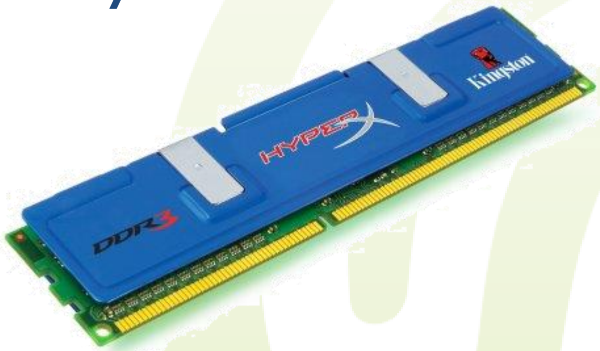
- **The bottleneck:** Slowest components in the computer are usually the **hard drives**
  - Most of the time, the CPU waits for data to be read from the HDD into the random access memory (RAM)

- Windows boots up to 3 times faster when installed on an SSD drive than on a HDD

8.4 Storage Media

Detour

- Data is stored on a **storage media**. Media highly differ in terms of
  - Random **Access** Speed
  - Random/ Sequential **Read/Write** speed
  - **Capacity**
  - **Cost** per Capacity



8.4 Storage Media

Detour

- **Random Access Time**: Average time to access a random piece of data at a known media position
  - Usually measured in ms or ns
  - Within some media, access time can vary depending on the position (e.g. hard disks)
- **Transfer Rate**: Average amount of consecutive data which can be transferred per time unit
  - Usually measured in KB/sec, MB/sec, GB/sec,…

8.4 Storage Media

Detour

- **Volatile:** Memory needs constant power to keep data
  - **Dynamic**: Dynamic volatile memory needs to be "refreshed" regularly to keep data
  - **Static**: No refresh necessary
- Access Modes
  - **Random Access:** Any piece of data can be accessed in approximately the same time
  - **Sequential Access:** Data can only be accessed in sequential order

8.4 Storage Media

Detour

- Media characteristics result in a **storage hierarchy**
- DBMS optimize data distribution among the storage levels
  - **Primary Storage**: Fast, limited capacity, high price, usually volatile electronic storage
    - Frequently used data / current work data
  - **Secondary Storage**: Slower, large capacity, lower price
    - Main stored data
  - **Tertiary Storage:** Even slower, huge capacity, even lower price, usually offline
    - Backup and long term storage of not frequently used data

8.4 Storage Media

Detour

Type	Media	Size	Random Acc. Speed	Transfer Speed	Characteristics
Pri	L1-Processor Cache (Intel QX9000)	32 KiB	0.0008 ms	6200 MB/sec	Vol, Stat, RA,OL
Pri	DDR3-Ram (Corsair 1600C7DHX)	16 GiB	0.004 ms	8000 MB/sec	Vol, Dyn, Ra, OL
Sec	Harddrive SSD (OCZ Vertex2)	22 TB	< 1 ms	285 MB/sec	Stat, RA, OL
Sec	Harddrive Magnetic (Seagate ST32000641AS)	3 TB	8.5 ms	138 MB/sec	Stat, RA, OL
Ter	DVD+R (Verbatim DVD+R)	PB	98 ms	11 MB/sec	Stat, RA, OF, WORM
Ter	LTO Streamer (Freecom LTO-920i)	PB	58 sec	120 MB/sec	Stat, SA, OF

Pri= Primary, Sec=Secondary, Ter=Tertiary
Vol=Volatile, Stat=Static, Dyn=Dynamic, RA=Random Access, SA=Sequential Access
OL=Online, OF=Offline, WORM=Write Once Read Many

8.4 Storage Media

- Alternative to hard-drives: **SSD**
  - Use microchips which retain data in non-volatile **memory** and contain **no moving parts**
- Use the same interface as hard disk drives
  - Easily replacing in most applications possible
- Key components
  - Memory
  - Controller (embedded processor)

8.4 SSDs

Detour

- The memory:
  - Retains memory even without power
  - Slower than DRAM solutions
  - **Wears down!**
- The controller:
  - **Wear levelling**, **bad block mapping**, read and write caching, encryption.

8.4 SSDs

Detour

- Advantages
  - Low access time and latency
  - No moving parts → shock resistant
  - Silent
  - Lighter and more energy-efficient than HDDs
- Disadvantages
  - Divided into blocks; if one byte is changed the whole block has to be rewritten (write amplification)
  - 10 % of the storage capacity are allocated (spare area)
  - **Limited ability of being rewritten** (between 3000 and 100,000 cycles per cell)
    - Wear leveling algorithms assure that write operations are equally distributed to the cells

8.4 SSD based DWs

Detour

- Hybrid solutions (OLTP & OLAP) with SSDs
  - Advantages:
    - Up to 10x faster than HDDs
    - Stores persistent data
  - Disadvantages:
    - SSDs wear down: limited ability of being **rewritten**
      - Not really suitable for OLTP
    - Not fast enough

- Dynamic Random Access Memory(DRAM)
  - Based on the volatile random access memory
    - Sometimes use **internal battery or external power** device to ensure data persistence
  - Ultrafast data access (< 10 microseconds)
  - Rather expensive

8.4 DRAM based DWs

Detour

- Hybrid solutions (OLTP & OLAP) with **DRAM**
  - **Very high data access speed**
    - Approx. 2000x faster than HDDs and 200x faster than SSDs
  - Once the power supply is gone, the data is lost
    - Use aditional bateries and external SSDs to make data persistent
- High cost: Multi-CPU server with **several terabytes of RAM** cost 60 000 euros - TU München – the HyPer system

8.4 HyPer

- Hybrid OLTP & OLAP main memory database system [A. Kemper and T. Neumann, ICDE 2011]

- System goals:
  – Process OLTP transactions at rate of tens of thousands per second, and at the same time
  – Process OLAP queries on up-to-date snapshots of the transactional data

8.4 HyPer: OLTP Processing

- Hyper is a main memory database that runs OLTP queries **sequentially**
  - Since all the data is already in the primary memory, the CPU doesn't have to wait for IO operations from the HDD
  - An *order entry* or a *payment processing* takes only around **10 microseconds**
  - **Tens of thousands/second** of such OLTP transactions can be sustained with such a system

8.4 HyPer: OLAP Processing

- Even with such fast hardware, allowing for OLAP to be injected in the OLTP workload would clog the system
  - OLAP queries finish in about 30 milliseconds on such a system
  - Locking the system for 30 milliseconds it blocks – in this time interval **3 000 OLTP queries** could have been executed
- Central idea: **run OLAP on OLTP snapshots**

8.4 HyPer: OLAP Processing

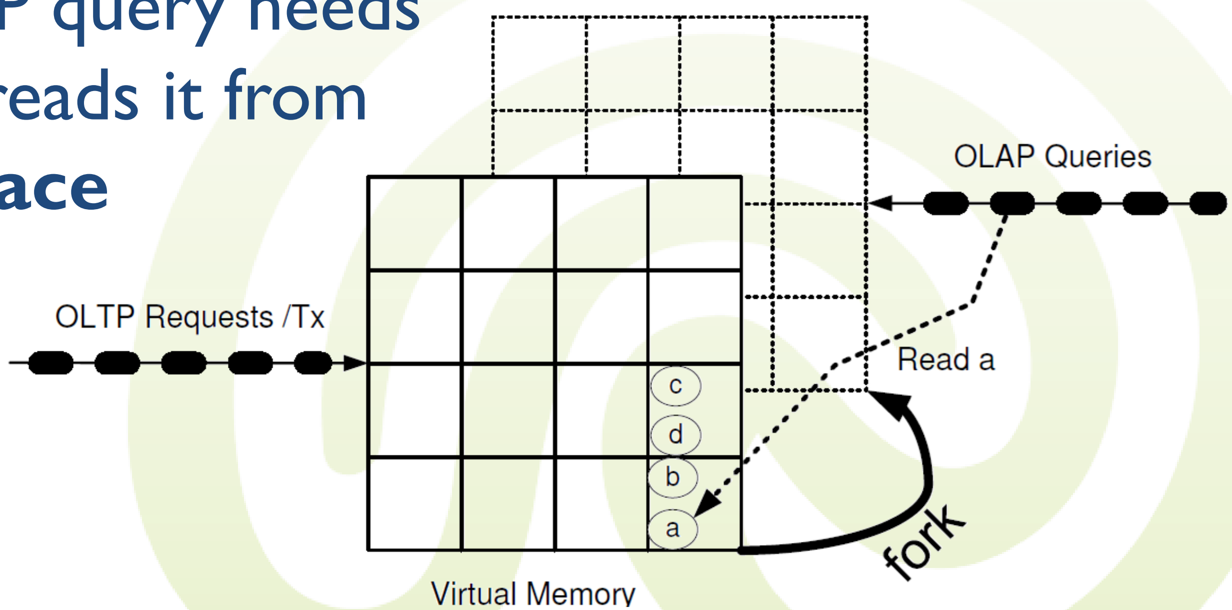
- **Snapshot management:** Create a virtual memory snapshot of the OLTP
  - Can be done with OS functionality e.g. *fork* system call for Unix based systems
  - **The fork system call**
    - Copies the complete process state to create a new process
    - The old process is called the **parent** and the new process the **child**
    - Immediately after fork the parent and the child are **identical**

8.4 The Fork System Call

- Initially the child's data pages are the same as the parent' pages
  - Except they are made read-only for the child
  - Uses a technique called **copy-on-write**
    - Only data that has been **changed by either parent or child** is physically copied in the child's memory pages
    - The rest of the data is **shadowed through pointers**
    - This way only a fraction of the data is actually transferred!

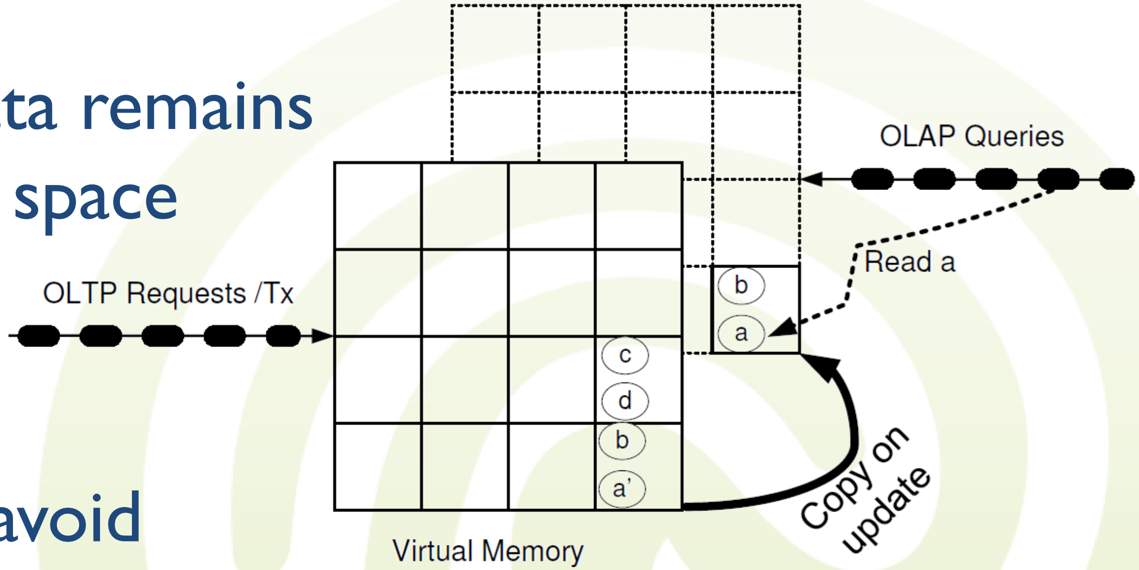
8.4 OLAP Snapshot Management

- The virtual memory snapshot of the OLTP memory pages created by a fork call are used for OLAP
  - The fork process doesn't copy anything!
  - When an OLAP query needs data page *a*, it reads it from the **parent space**



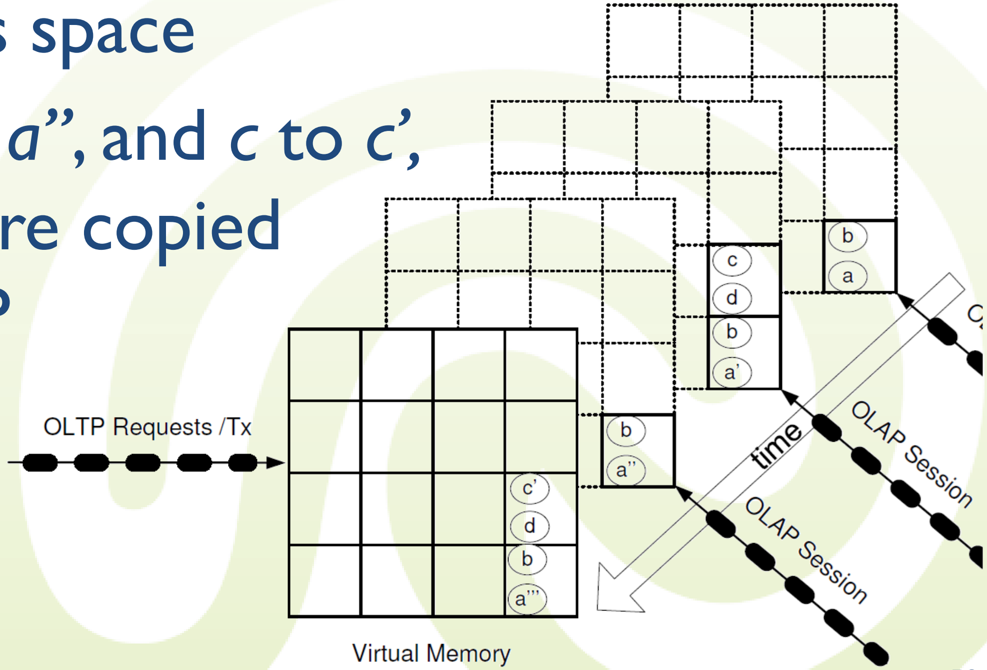
8.4 OLAP Snapshot Management

- As soon as *a* is **updated** by a transaction from OLTP to *a'*, the memory page containing the old data values *a* and *b* is **copied to the child** (OLAP) space
  - The rest of the data remains only in the parent space
  - This way we also avoid the problem of **OLAP on changing data**



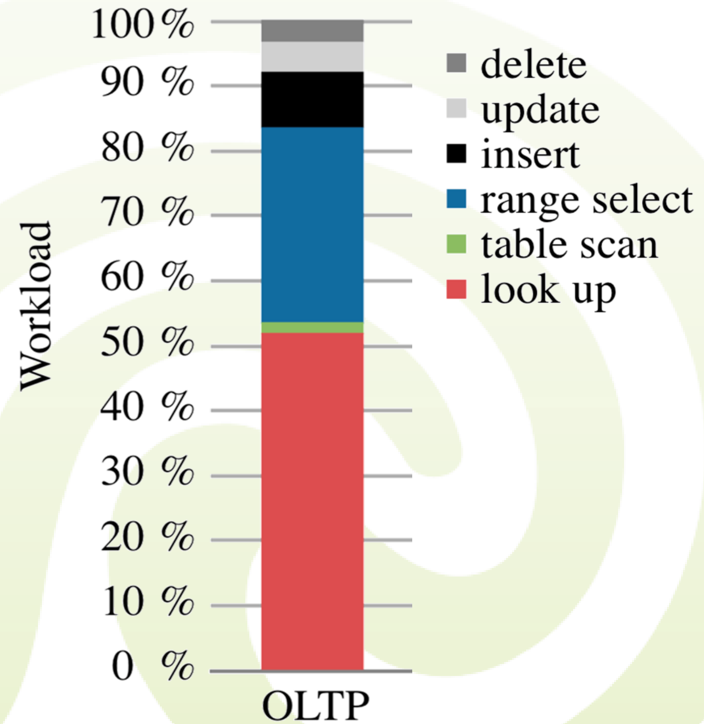
8.4 HyPer: OLAP Processing

- Multiple OLAP queries may make use of multi-core hardware and speed up querying even more
  - Each OLAP query runs as a new fork process
  - Once *a* is updated to *a'*, the page of *a* is copied into the first OLAP query's space
  - Once *a'* is updated to *a''*, and *c* to *c'*, the pages of *a'* and *c* are copied into the second OLAP query's space
  - Etc.



8.4 Multi-Thread OLTP Processing

- Multi-core architecture can also speed up OLTP
  - **Reads** can easily be parallelized
  - For inserts, updates and deletes, locks are needed
  - Still most of the workload is represented by **lookups**, table scans and range selects



8.4 Real-Time DW Products

Detour

- SAP HANA: real-time business intelligence with **in-memory** technology
  - Layer for transactional business applications, planning, data warehousing, and BI tools
  - All enterprise applications OLTP & OLAP share one **column store** database architecture

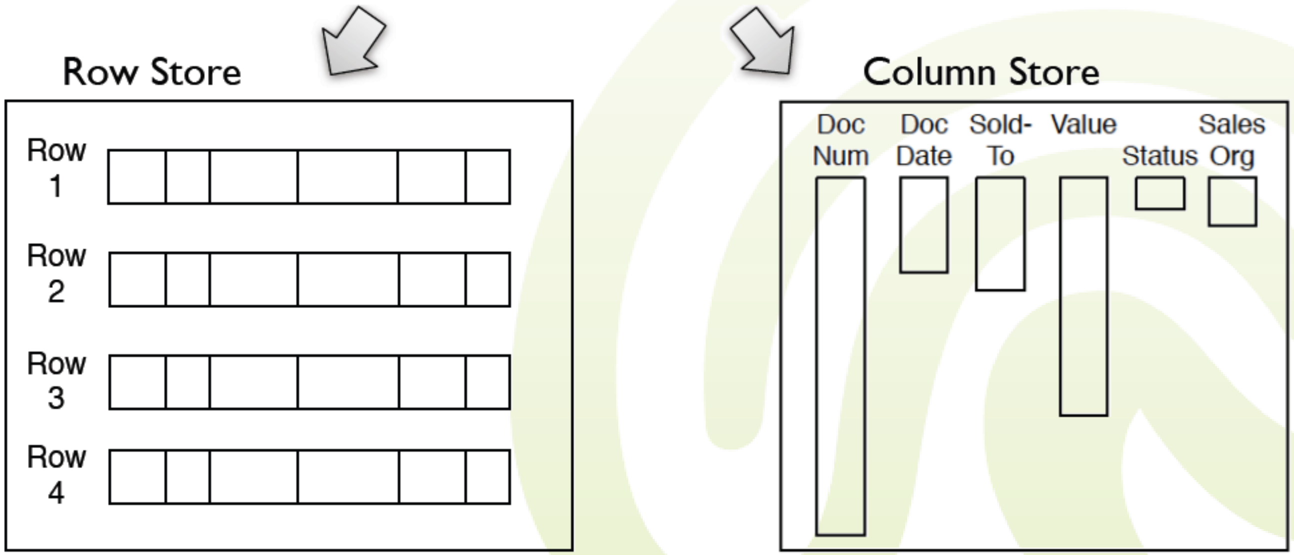
Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

- Column store?

Row vs. Column Store

(Compressed)

Document Number	Document Date	Sold-To Party	Order Value	Status	Sales Organization	...
95769214	2009-10-01	584	10.24	CLOSED	Germany Frankfurt	...
95769215	2009-10-01	1215	124.35	CLOSED	Germany Berlin	...
95779216	2009-10-21	584	47.11	OPEN	Germany Berlin	...
95779217	2009-10-21	454	21.20	OPEN	Germany Frankfurt	...



Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

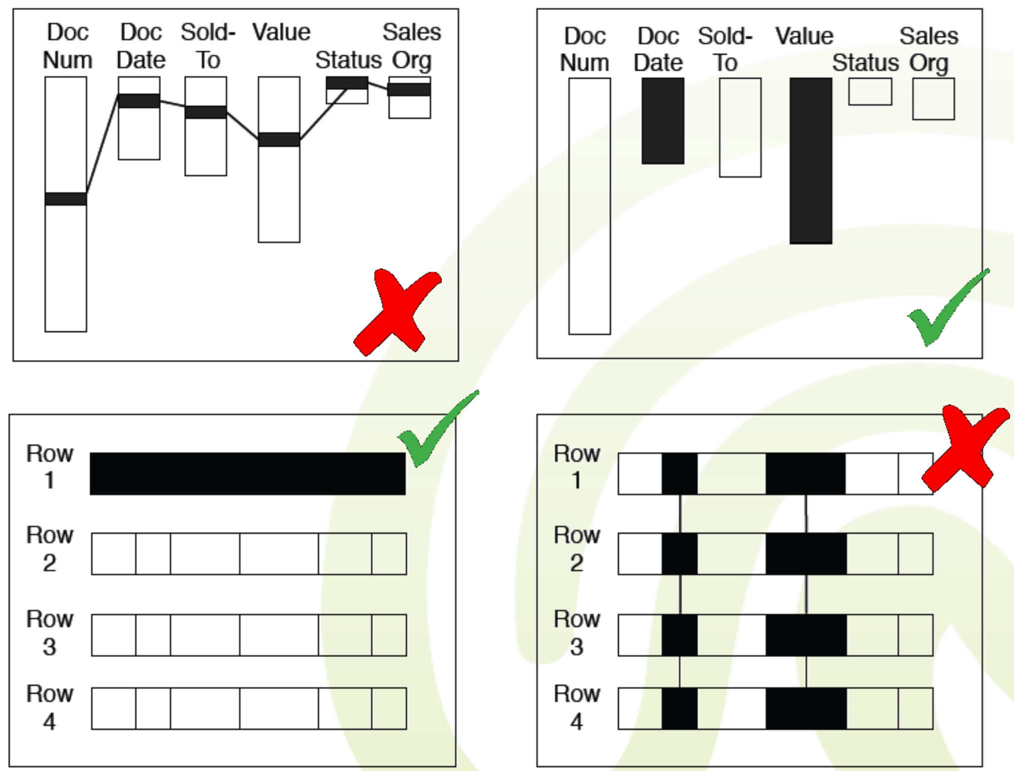
OLTP vs. OLAP Queries

SELECT *
FROM Sales Orders
WHERE Document Number = '95779216'

SELECT SUM(Order Value)
FROM Sales Orders
WHERE Document Date > 2009-01-20

Column Store

Row Store

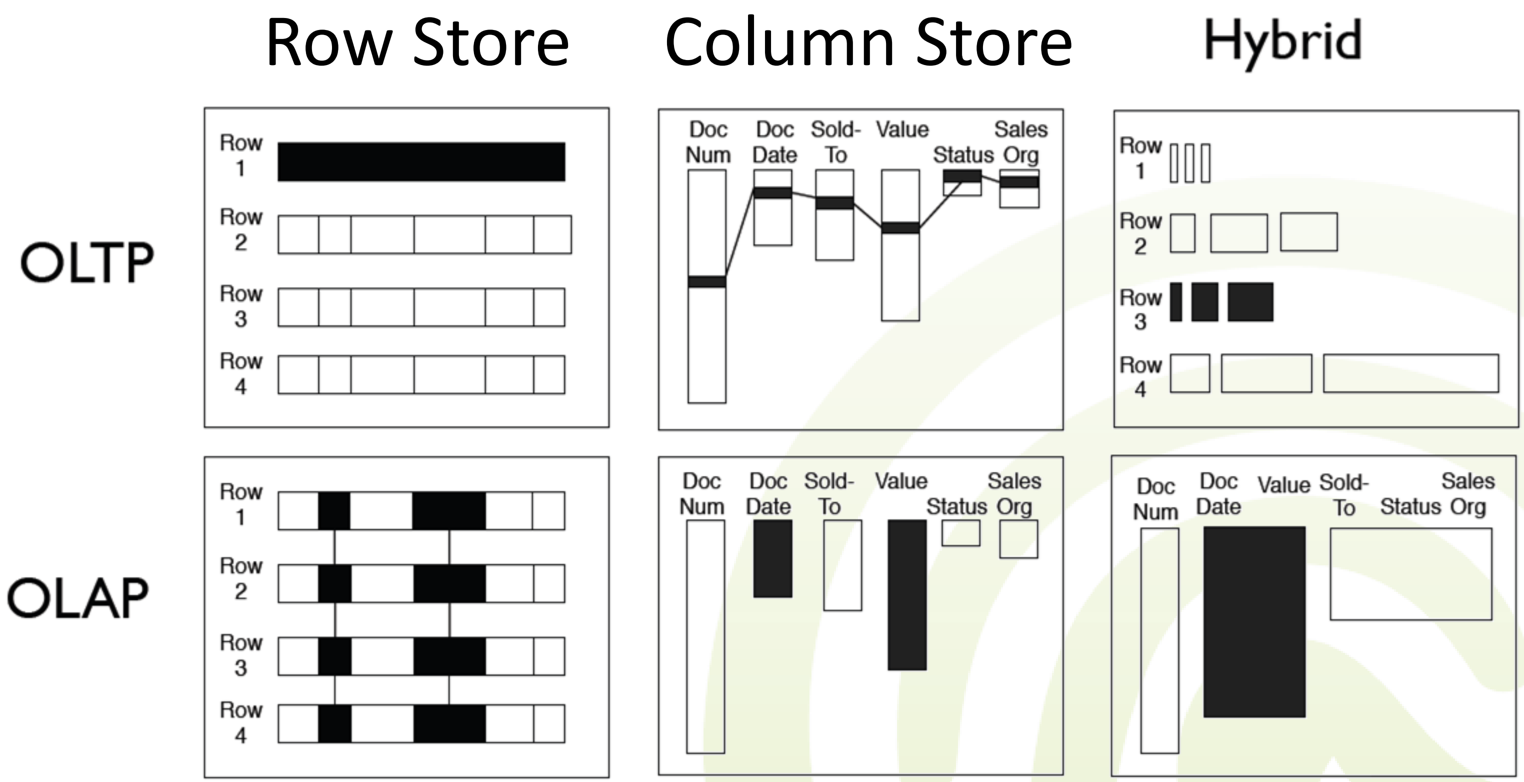


HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

8.4 SAP HANA

Detour

- HANA: hybrid storage



Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Summary

- How to enable real-time ETL
  - Microbatch and Trickle & Flip for near-time ETL
  - Direct Trickle-Feed with real-time partition – some query contention on the real-time partition
  - External Real-time Data Cache (ERDC) with Just-in-Time data is a pretty complex system
- OLAP and the Changing Data
  - Consistency problems
- OLTP & OLAP - Hybrid Solutions
  - HyPer and SAP HANA

Next lecture

- Business Intelligence (BI)
  – Principles of Data Mining
  – Association Rule Mining