ASUG SAP BusinessObjects
USER CONFERENCE

OCTOBER 9-12, 2011
ORLANDO, FLORIDA

BUSINESS INTELLIGENCE
FOR A PASSIONATE COMMUNITY

# Building a Data Warehouse: Data Modeling

David G. Rathbun

Werner Daehn, SAP

ASUG Real Experience. Real Advantage.

# The “Building a Data Warehouse” Series

- Our goal, within those three days, is to build an entire DWH
- Source is a Controlling System
- We want to analyze Actual bookings and Plan data

| | | |
|---|---|---|
| Monday | 9:30 AM - 10:30 PM | Data Modeling |
| Monday | 10:45 AM - 11:45 AM | Getting the data into the DWH database |
| Monday | 1:30 PM - 2:30 PM | Building a Universe |
| Tuesday | 9:30 AM - 10:30 AM | Intro to SAP Crystal Reports |
| Tuesday | 1:30 PM - 2:30 PM | Report Development in Web Intelligence |
| Wednesday | 9:15 AM - 10:15 AM | Data Quality is key for BI |
| Wednesday | 10:30 AM - 11:30 AM | Enhance the DWH with external data |

# About Dave

- Dedicated to BusinessObjects solutions since 1995
  - Consultant and trainer for fifteen years
  - Currently BI Solutions Architect for PepsiCo
  - Note: Content is my own and does not reflect my employer
- 16 consecutive years presenting at major BI conferences
  - United States, Europe, Australia
- Charter member of BOB
  - http://busobj.forumtopics.com
- I Blog! Dave's Adventures in Business Intelligence
  - http://www.dagira.com
- SAP Mentor for 2009 – 2012

# Key Points

- The entire goal is to reduce the number of joins for queries
- Star Schema Data Model is the most efficient
- Don’t be part of any religious war, rather know the ideas and when to use what
  - *“Religious wars are basically people killing each other over who has the better imaginary friend.” (Richard Jeni)*

# Why Data Modeling Is Important

- Source systems are built for fast single row data entry and updates
  - Normalized data models are optimized for entry / update
  - TPC benchmarks measure data capture
- BI tools need to navigate freely
  - Queries need to run efficiently
  - Normalized models do not easily support table scans with multiple joins in an efficient fashion
  - Database has to aggregate vast amounts of data quickly
- ETL tool has to copy the source data into the target data model efficiently

- Our data model has to support contradicting requirements

# Normalized Model Versus Star Schema

- Source uses a normalized schema
  - Each unique entity is a table of its own
  - Joins define relationships between tables
- Advantage
  - When updating one column only small tables are updated
  - When reallocating a row only the relationship column is updated
- Disadvantage
  - Highly normalized models are quite complex
  - Some application models have thousands of tables

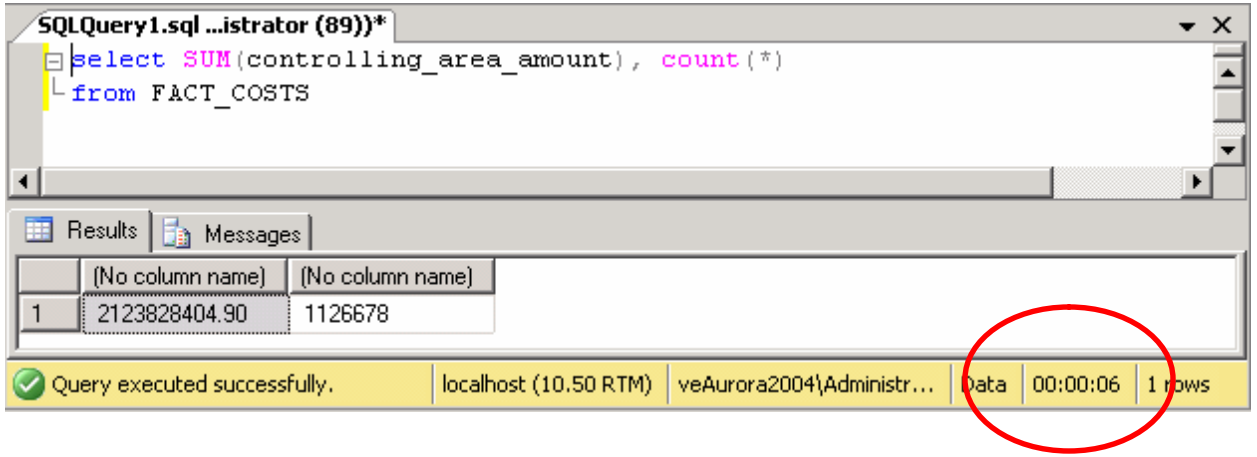
| Amount | CostElement |
|---|---|
| 100USD | 001010243 |

| CostElement | CtrlArea |
|---|---|
| 001010243 | 1000 |

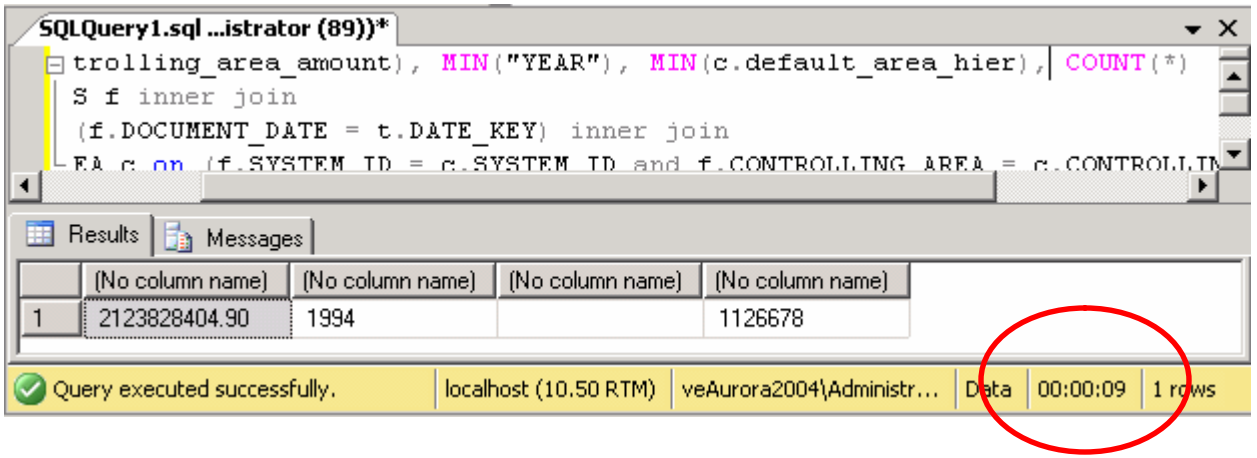
| CtrlArea | Name |
|---|---|
| 1000 | ASUG |

# Performance Impact of Joins

- A Sum() operation forces a full table scan of the large table
  - 6 seconds

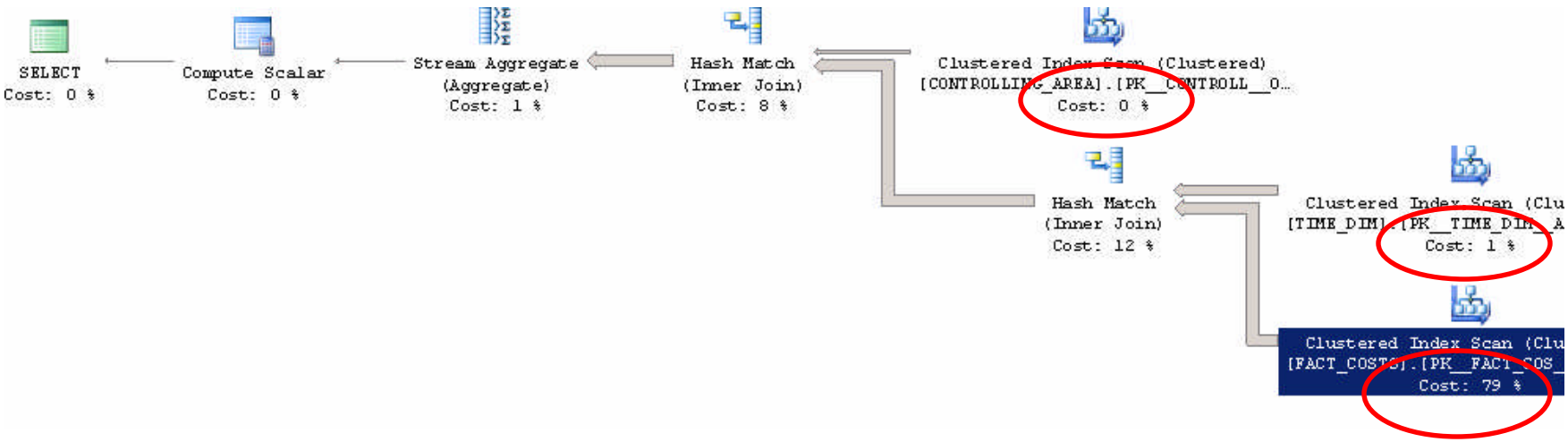



- Same result from normalized schema (two small tables joined)
  - 9 seconds



# Performance Impact of Joins

- The execution plan show the obvious
  - Fact reading is 79%
  - Reading the two tiny tables is1%
  - The two joins cost 20%



# How Do We Reduce The Number of Joins?

- Sum(amount) for the CtrlArea Name = ‘ASUG’

| Amount | CostElement |
|---|---|
| 100USD | 001010243 |

| CostElement | CtrlArea |
|---|---|
| 001010243 | 1000 |

| CtrlArea | Name |
|---|---|
| 1000 | ASUG |

- Option 1 – all into one table
  - Result: a 500MB Database grows to 4TB!

| Amount | CostElement | CtrlArea | Name |
|---|---|---|---|
| 100USD | 001010243 | 1000 | ASUG |

- Option 2 – a Star Schema

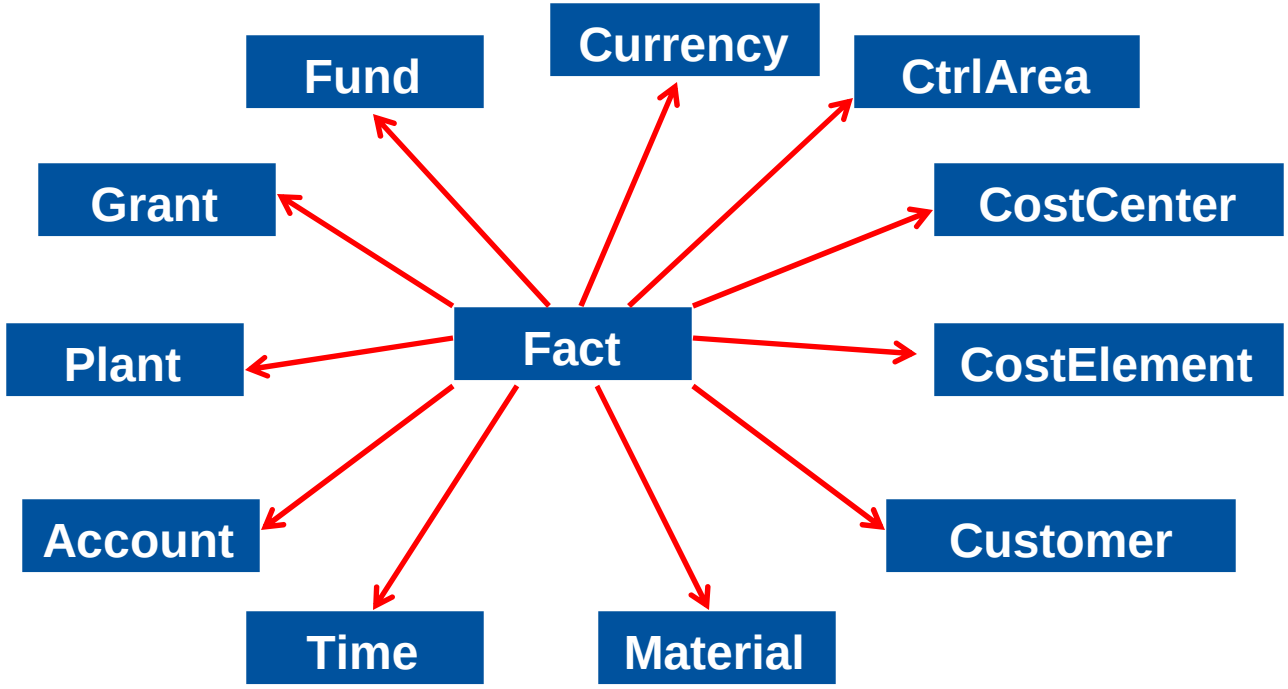
| Amount | CostElement | CtrlArea |
|---|---|---|
| 100USD | 001010243 | 1000 |

| CtrlArea | Name |
|---|---|
| 1000 | ASUG |

| CostElement | CtrlArea |
|---|---|
| 001010243 | 1000 |

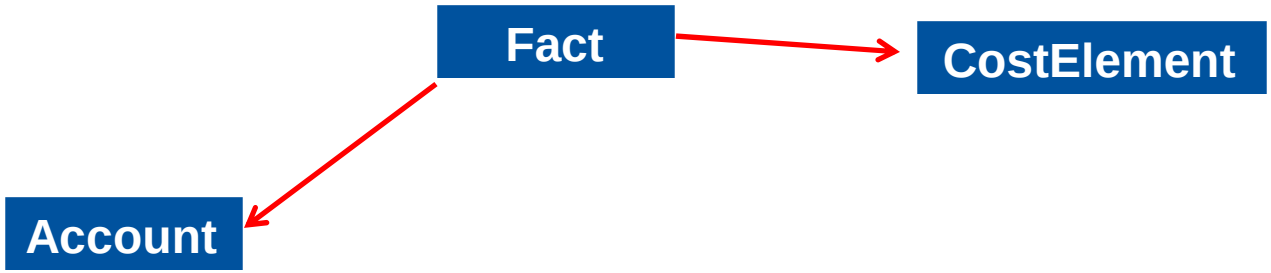
# What Is A Star Schema?

- Everything is connected to the fact table
- Dimension tables contain all attributes including text
- Dependencies between attributes are not important anymore
- Their relationships to the central measure is



# What Is A Star Schema?

- There is not the perfect data model, it depends on the queries
  - Are Account attributes queried often but Cost Element is not?
    - Then two separate dimensions are perfect
  - Are Account + Cost Element attributes always queried together?
    - Then both could be merged into one dimension
- Design decisions can be difficult to revisit
- Gathering effective report requirements up front is a must



# What About HANA? Column Store Databases?

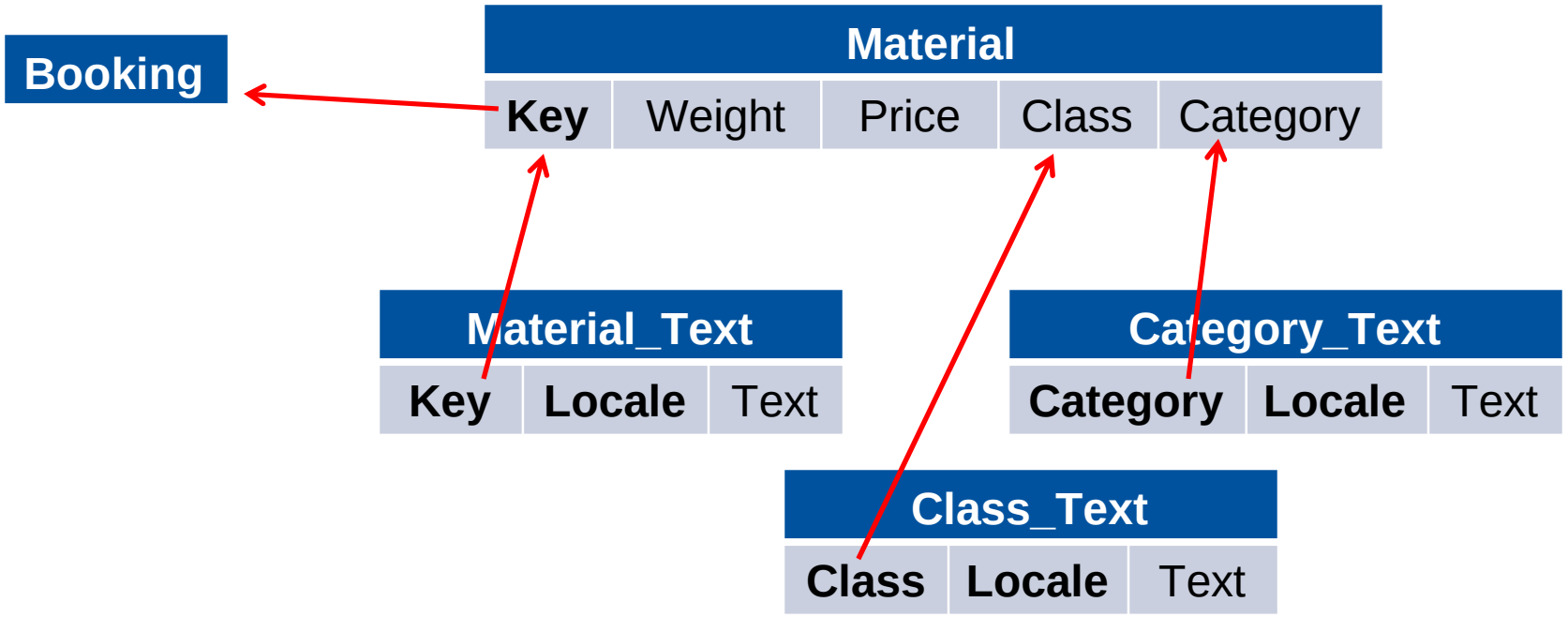
- Column store databases can have zero joins
  - I/O is reduced but size of data storage increases
- Size concern can be addressed with compression
  - Size is reduced but CPU use increases
- In-memory processing dramatically improves performance
  - I/O has less overhead in memory when compared to disk
  - Consider solid-state hard drives as well

# Issues We Had To Solve

- Text Attributes in multiple languages
- Time dependencies
- Historically correct queries
- Hierarchies
- Currency conversion
- Multiple facts

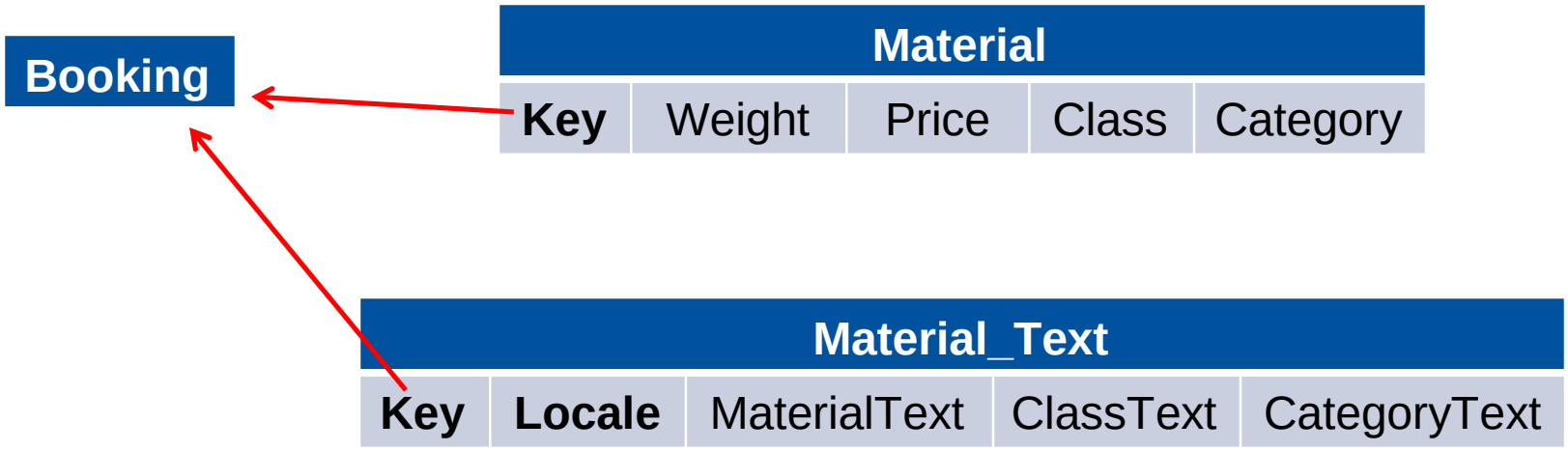
# Text Attributes In Multiple Languages

- With one language the solution is simple
  - Translated text is an attribute added to the dimension directly
  - One row exists for each language (locale)
- The impact of this design is multiple extra joins



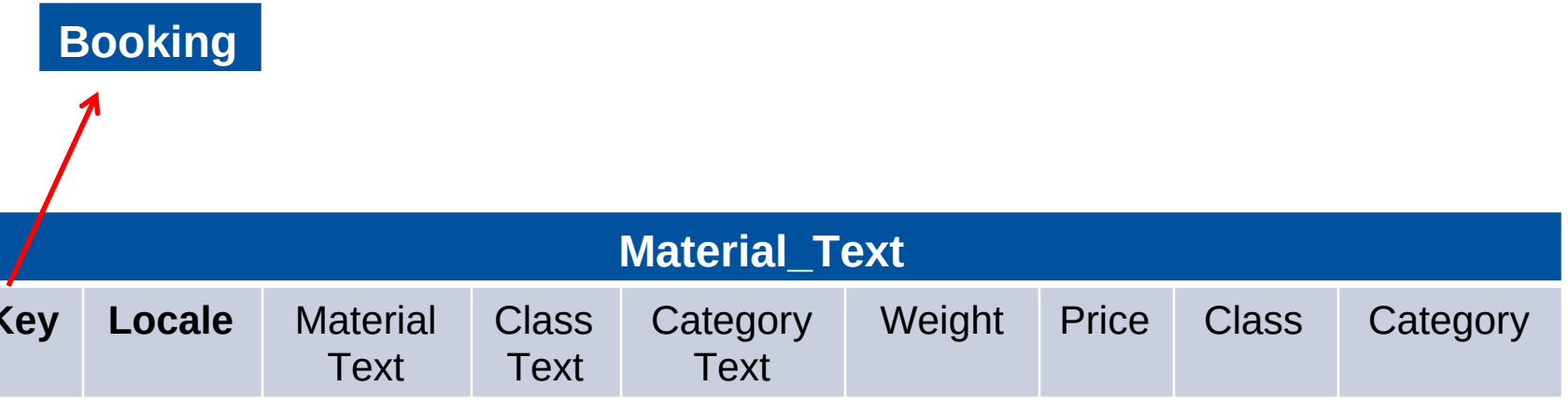
# Text Attributes In Multiple Languages

- Additional material text dimension
  - One join for all text columns at once
  - Could be a large dimension
  - Often attributes and text are queried together
  - Requires extra joins



# Text Attributes In Multiple Languages

- The Material dimension is Locale aware
  - One join for Text and Attributes
  - Will be a large dimension table
  - Use with care, e.g. only a few attributes exist but lot of text



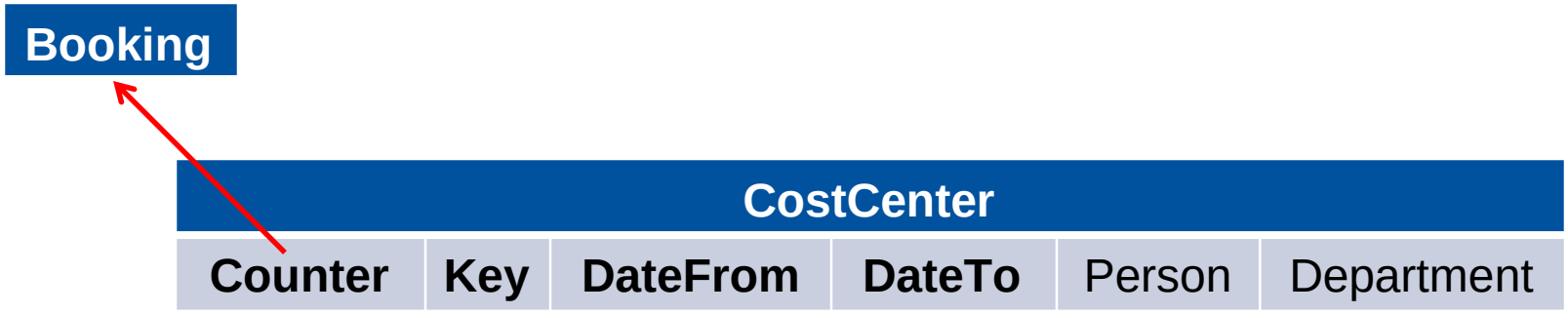

- Why translate in the database instead of the universe?
  - Translation information is available for all access paths

# Time Dependencies

- The source has a DateFrom field part of the primary key
- This can't be joined, at least a DateTo we need as well
  - `join on (fact.key = cc.key and fact.bookingdate between cc.datefrom and cc.dateto)`
- Later corrections possible without modifying the fact
- Later changes possible hence query results can change
- Small table size
- Database optimizations difficult to do

# Time Dependencies

- We add a surrogate key to the dimension
- When loading the fact we store the matching surrogate key
  - `join on (fact.cc_counter = cc.counter)`
- Query results remain stable
- Easy to optimize the database, e.g. aggregates
- Table remains small



# Historically Correct Queries

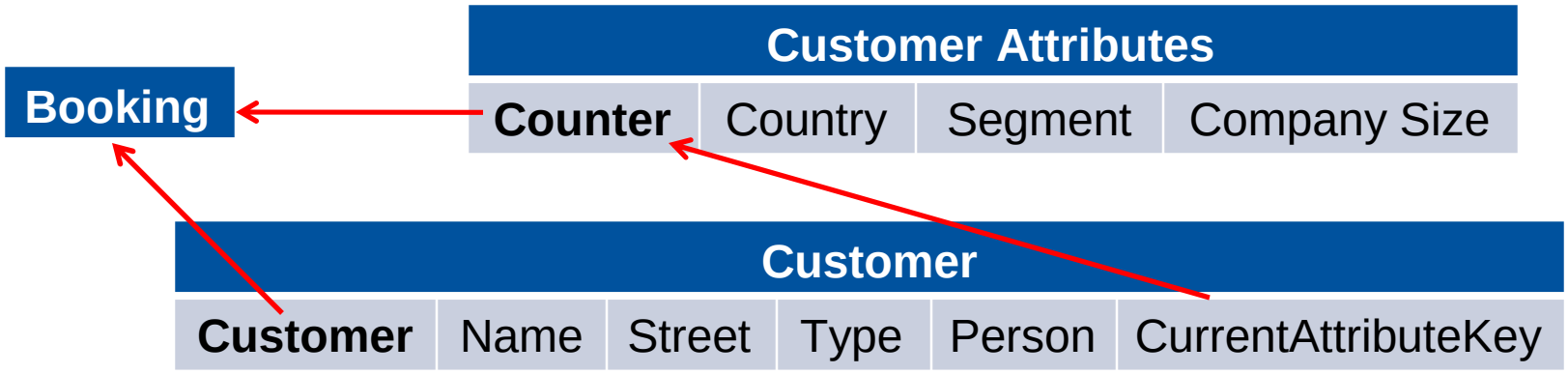
- One goal might be to keep the queries stable
- If no booking got added, the query shall return the same data
  - ```
    Select sum(costs) from fact_costs inner join
    customer c on (fact.customer=c.customer)
    where c.country = 'US'
    ```
- What if one customer record got updated from US to CA?
- All costs will show up in the new country
- Might be the desired result…
- …Might not
    ```

# Historically Correct Queries

- Slow Changing Dimension Type 2
  - We never update the customer, we insert it with a new counter
  - Fact contains the counter, not the customer number as FK
- Move the important columns into the fact
  - When can update the customer dimension – it contains the current values
  - The historical correct country is in the fact table
- Build a second derived dimension with customer attributes
  - Customer table gets updated
  - The current attribute combination is added to a table
  - The fact contains both dimension tables primary keys
  - Like in…

# Historically Correct Queries

- Derived Dimension of Customer
  - Small tables
  - Especially fast when Customer Attributes are queried
  - Can query booking historically correct and with current attributes
  - An nice technique for large dimension tables!



# Hierarchies

- The perfect solution is a Parent-Child-Table as it supports
  - Unbalanced Hierarchies where different nodes have different depths
  - Unlimited depth instead of an upper limit
  - Small and efficient storage
- But it is hard to query
  - “Node” contains cost centers and group ids
- The tool has to support it
  - Crystal does
  - Web Intelligence does not

```
select id, parent_id, node, node_name from cost_center_hier_p_c
where controlling_area = '1000' and LOCALE = 'en_US'
order by 1
```

Results | Messages

| | id | parent_id | node | node_name |
|---|---|---|---|---|
| 31 | 30 | 2 | 1000H1300 | Marketing and Sales |
| 32 | 31 | 30 | 1000H1310 | Sales |
| 33 | 32 | 31 | 0000003100 | Motorcycle Sales |
| 34 | 33 | 31 | 0000003105 | Automotive Sales |
| 35 | 34 | 31 | 0000003110 | Pump Sales |
| 36 | 35 | 31 | 0000003120 | Sales Paint/Solvents |
| 37 | 36 | 31 | 0000003125 | Sales Pharma/Cosmet. |
| 38 | 37 | 31 | 0000003130 | Light Bulb Sales |
| 39 | 38 | 31 | 0000003135 | Sales Foodstuffs |
| 40 | 39 | 31 | 0000003140 | High-Tech Sales |
| 41 | 40 | 31 | 0000003150 | Elevator Sales |
| 42 | 41 | 31 | SEM2151000 | sem2151000 |
| 43 | 42 | 30 | 1000H1320 | Marketing |
| 44 | 43 | 42 | 0000003200 | Marketing |

# Hierarchies

- I have previously covered hierarchy models in great detail in a previous session in 2009
  - “Universe Models for Recursive Data”
  - http://www.dagira.com
- Covers several hierarchical strategies
  - Flattened via aliases or snowflake tables
  - Ancestor descendant
  - Depth first tree traversal

# Hierarchies

- L1 is the topmost cost center group, L2 the second,…
- Somewhere at L4, L5, L6,… is the cost center itself
- Deeper levels repeat the cost center again
- Drill down from cost center to another cost center

```
select LEAF_LEVEL, COST_CENTER, C_L3_NODE_NAME, C_L4_NODE_NAME, C_L5_NODE
from cost_center_hier_flat
where controlling_area = '1000' and LOCALE = 'en_US'
```

Results | Messages

| | COST_CENTER | C_L3_NODE_NAME | C_L4_NODE_NAME | C_L5_NODE_NAME | C_L6_NODE_NAME |
|---|---|---|---|---|---|
| 18 | 0000003100 | Company 1000 - Germa | Marketing and Sales | Sales | Motorcycle Sales |
| 19 | 0000003105 | Company 1000 - Germa | Marketing and Sales | Sales | Automotive Sales |
| 20 | 0000003110 | Company 1000 - Germa | Marketing and Sales | Sales | Pump Sales |
| 21 | 0000003120 | Company 1000 - Germa | Marketing and Sales | Sales | Sales Paint/Solvents |
| 22 | 0000003125 | Company 1000 - Germa | Marketing and Sales | Sales | Sales Pharma/Cosmet. |
| 23 | 0000003130 | Company 1000 - Germa | Marketing and Sales | Sales | Light Bulb Sales |
| 24 | 0000003135 | Company 1000 - Germa | Marketing and Sales | Sales | Sales Foodstuffs |
| 25 | 0000003140 | Company 1000 - Germa | Marketing and Sales | Sales | High-Tech Sales |
| 26 | 0000003150 | Company 1000 - Germa | Marketing and Sales | Sales | Elevator Sales |
| 27 | 0000003200 | Company 1000 - Germa | Marketing and Sales | Marketing | Marketing |
| 28 | 0000004100 | Company 1000 - Germa | Technical Area | Services | Technical Service I |
| 29 | 0000004110 | Company 1000 - Germa | Technical Area | Services | Technical Facilities |
| 30 | 0000004120 | Company 1000 - Germa | Technical Area | Services | IT Service |

# Currency Conversion

- It is a simple task: Amount, date and rate for each day is known
- Pick the proper rate and multiply the amount
- A few questions however
  - What date? Today? Day of the booking?
  - Today's rate we will know tomorrow
  - What rate? Buy? Sell? Monthly average?
  - Yesterday a booking for 100USD = 86EUR was made, today it got canceled with -100USD = -84EUR???
- Depending on the answer different techniques are used

# Currency Conversion

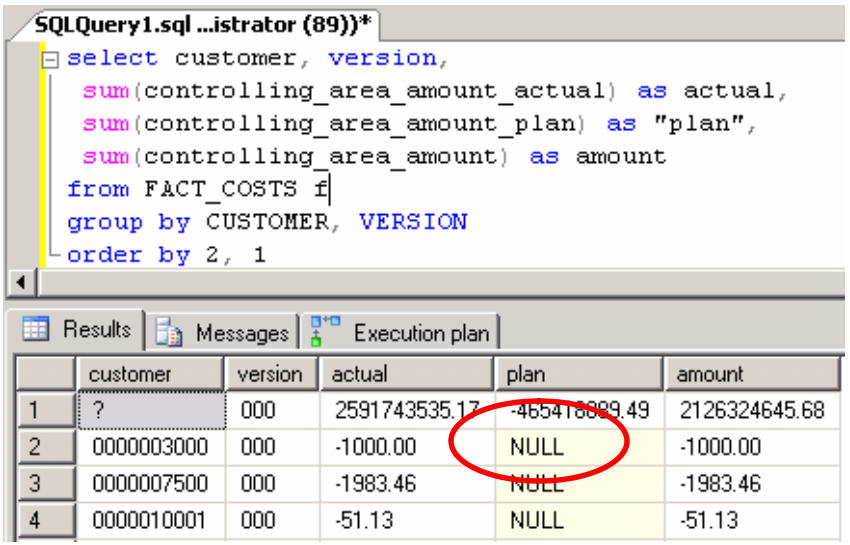
- What date?
  - Sales Person: “I made a deal about 100EUR, that’s 143USD.”
    - Hence we use the booking date for the conversion
  - Finance Person: “Thanks to the deal our bank account in Europe has a balance of 100EUR.”
    - Hence its value is different today
- What rate?
  - When we planned the revenues we assumed an average rate, not such a difference in conversion rates.
    - We have to convert using the year average planning rate
- Will new rates be effective for the past?
  - No, then we convert when the booking is loaded
  - Yes, then we store the original amount

# Multiple Facts

- We have two sources
  - Actuals per Material, Customer, CostCenter, Day,…
  - Plan per CostCenter, Month, Plan-Version,…
- Most queries will compare the two
  - Sum the Actuals per year
  - Sum the Plan data per year
  - Compare with Actuals – Plan calculation
- Multiple facts require multiple queries
  - Contexts can be used to split queries automatically
  - Report writers can create multiple data providers
  - … Or both measures can be placed into a single fact table

# Multiple Facts

- Fact table includes measures in addition to amounts
  - Plan column containing that value in case of a plan number
  - Actual column containing the value for real actuals
  - Only for version '000'
- All reports are still possible using the amount column
- But standard actual versus plan reports simply sum these two columns
- Filter on columns not planned on will show no plan value
  e.g. customer=3000 has actual but no plan amount



# Do Not Sacrifice Performance For Correctness!

- Primary goal should always be correctness
- Secondary goals are
  - Performance (queries or load processes)
  - Ease of use
- Some databases function better with different models
  - Teradata likes additional joins especially when conditions are added so a snowflake model might be better
  - Different appliances might like a column store better
- Compact models (star versus snowflake) can be easier to understand

# Reporting Is The End Of The Food Chain

- Data is initially obtained
  - Most likely a highly normalized model captures the information
- Data is extracted, transformed, and loaded into a warehouse
  - Transformation operations merge data from separate tables into a dimensional model
  - Fact tables are built containing reporting measures
- A universe is built to provide a query platform for the user
  - A common semantic layer is a huge benefit to a project
- After all of these steps a report, dashboard, or analytic can be built

# Push Back When Possible

- Calculations should be done as far away as possible
  - Do you always need to track “same period” orders? Put it into the data model and populate via ETL
  - Far better than repeatedly performing the same calculation in a query

  - Before


| From Date Key | Fiscal Period |
|---|---|
| 1000 | P01 |

| To Date Key | Fiscal Period |
|---|---|
| 1001 | P02 |

After

| Booking | SamePrd |
|---|---|
| 100 | 0 |
| 100 | 1 |

# Learning Points

- To build a good Data Model we have to know the typical queries, database capabilities, tool limitations and the source
- Therefore it is a team effort one person is leading
- When creating the model, quite a few decisions are made, possibly unconsciously

# The “Building a Data Warehouse” Series

- Our goal, within those three days, is to build an entire DWH
- Source is a Controlling System
- We want to analyze Actual bookings and Plan data

| | |
|---|---|
| Monday 9:30 AM - 10:30 PM | Data Modeling |
| Monday 10:45 AM - 11:45 AM | Getting the data into the DWH database |
| Monday 1:30 PM - 2:30 PM | Building a Universe |
| Tuesday 9:30 AM - 10:30 AM | Intro to SAP Crystal Reports |
| Tuesday 1:30 PM - 2:30 PM | Report Development in Web Intelligence |
| Wednesday 9:15 AM - 10:15 AM | Data Quality is key for BI |
| Wednesday 10:30 AM - 11:30 AM | Enhance the DWH with external data |

# Thank you for participating.

Please remember to complete and return your evaluation form following this session.

For ongoing education on this area of focus, visit the Year-Round Community page at www.asug.com/yrc

**SESSION CODE: 7001**