Multi Level Data Navigation with MicroStrategy

EMPOWER BI

www.empowerbi.com

EMPOWER BI
knowledge that works

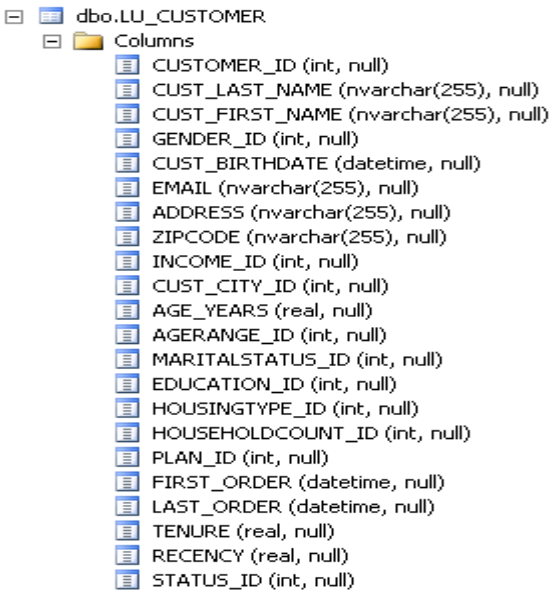
Introduction

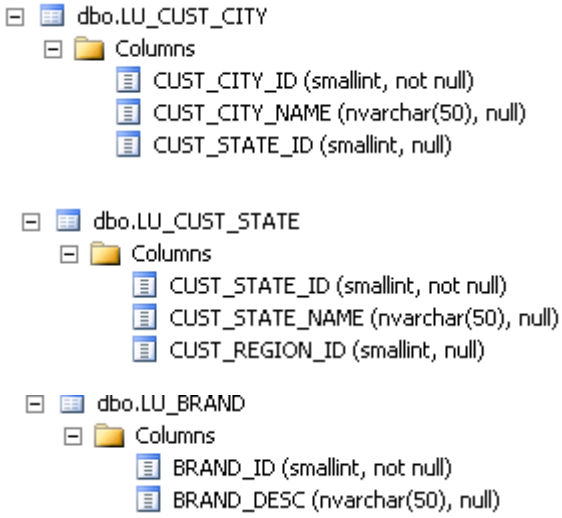
- What is multi-level data (MLD)?
- Why is MLD important?
- How is MLD best implemented?
- How do you create a schema to navigate it?
- Fully worked example
- A parting word
- Summary

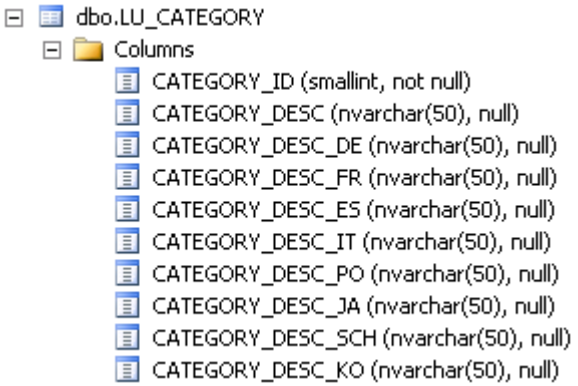
EMPOWER BI
knowledge that works

What is Multi-Level Data?

- The vast majority of queries use summary data
- Microstrategy has implemented Intelligent Cubes to support this behaviour
- Microstrategy has implemented multi-level lookup tables for many years



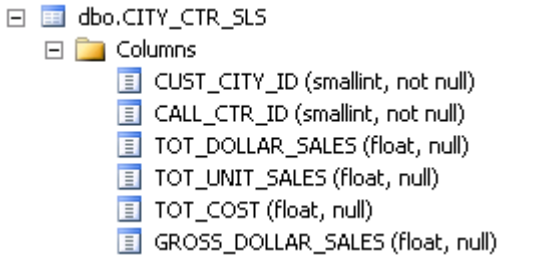


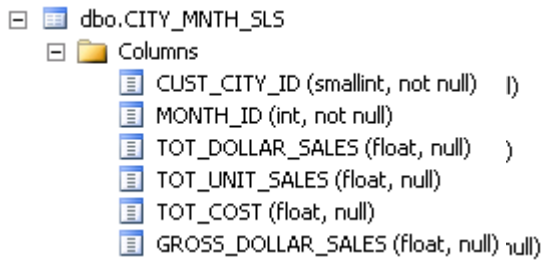


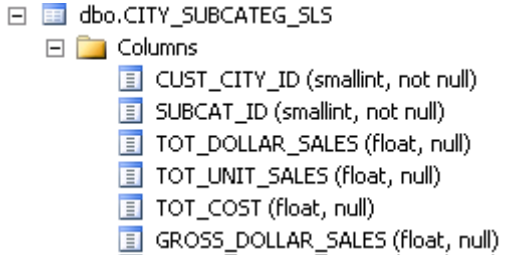
EMPOWER BI
knowledge that works

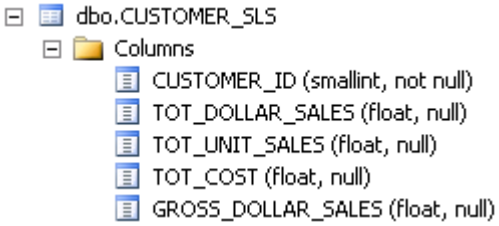
What is Multi-Level Data?

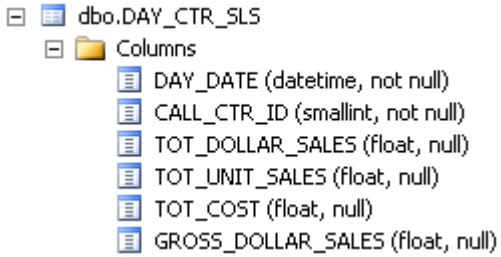
- You can see them in the tutorial database…
- These are example summary level fact tables

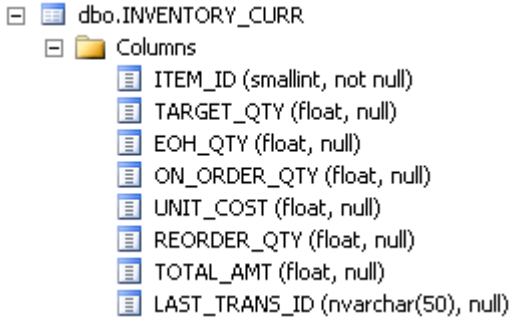












- The following diagrams show you simple diagrams of how these things evolved

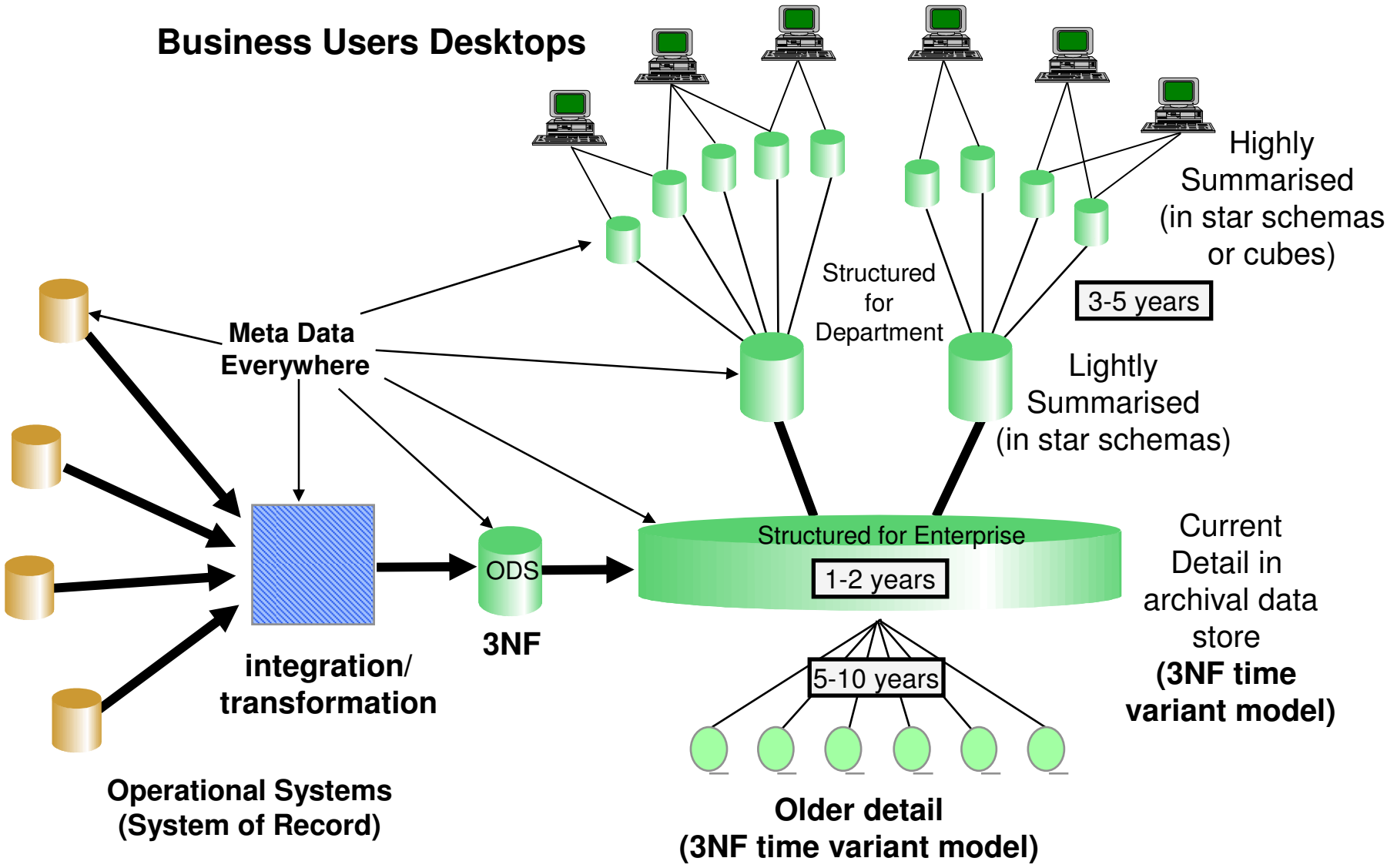
EMPOWER BI
knowledge that works

Evolution of BI Models

- Pre 1995 no companies mixed 3NF and Dimensional
  - Companies did one or the other…not both
- In late 1995 two pioneers ‘tried out’ mixing the two modeling techniques in one design. It worked well.
- Since 1996 these two pioneers recommended to clients the most complete solution was a combination of both
  - It is also the most expensive approach
  - Not everyone would buy into this approach
- Diagram on next page shows ‘Best Practice Since 1996’…

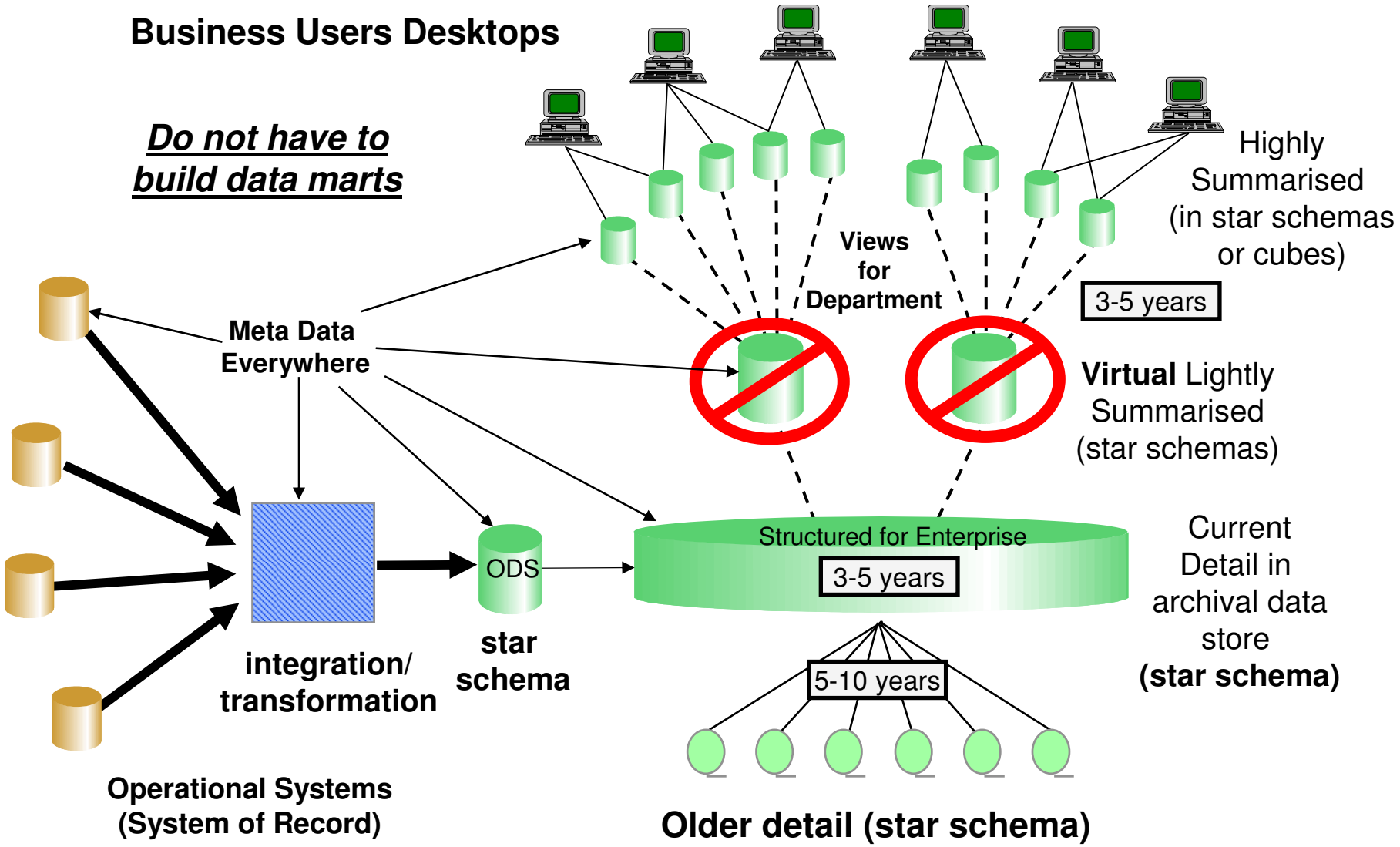
EMPOWER BI
knowledge that works

Best Practice Since 1996



EMPOWER BI
knowledge that works

New BIDA Proposition



EMPOWER BI
knowledge that works

Why Is MLD Important?

- #1 reason is speed of query
- If you summarise 1000x the query will be 1000x faster…approximately
- Summarisation is the #1 tool for fast queries
- Everyone does it no matter some deny it
- Particularly necessary for dashboards and interactive query processing such as drill down
- MSTR has done this for many years

EMPOWER BI
knowledge that works

How Is MLD Best Implemented?

- Ralph Kimball of “Star Schema” fame did it best
- These skills were “lost” in the 90s
- It is done by using a single underlying table with a “Level” column and views places over the top
- Having one table per summary costs a lot of maintenance time for ETL programmers
- 100% replicates Microstrategy recommendations
- So lets look at an example
- Our friend the day table…

The Multi Level Day Table

```
CREATE TABLE [dbo].[TD0005](
[pk_TD0005] [int] NOT NULL DEFAULT ((0)),
[level_col] [varchar](10) NOT NULL DEFAULT ('unknown'),
[dim_char_ky_fld] [varchar](255) NOT NULL DEFAULT ('unknown'),
[day_date] [datetime] NULL,
[day_name] [varchar](9) NULL,
[day_name_sdesc] [varchar](3) NULL,
[day_in_week] [int] NULL,
[day_in_month] [int] NULL,
[day_in_year] [int] NULL,
[week_in_month] [int] NULL,
[week_in_year] [int] NULL,
[month_name] [varchar](15) NULL,
[month_name_sdesc] [varchar](3) NULL,
[month_in_year] [int] NULL,
[calendar_qtr] [int] NULL,
[calendar_qtr_str] [varchar](1) NULL,
[month_in_qtr] [int] NULL,
[week_in_qtr] [int] NULL,
[day_in_qtr] [int] NULL,
[financial_qtr] [int] NULL,
[financial_qtr_str] [varchar](1) NULL,
[financial_year] [int] NULL,
[financial_year_name] [varchar](255) NULL,
[month_in_fncl_qtr] [int] NULL,
[week_in_fncl_qtr] [int] NULL,
[day_in_fncl_qtr] [int] NULL,
[year_name] [varchar](4) NULL,
[year_num] [int] NULL,
[season_name] [varchar](15) NULL,
[season_name_sdesc] [varchar](3) NULL,
[num_days_since_1970] [int] NULL,
[num_weeks_since_1970] [int] NULL,
[num_months_since_1970] [int] NULL,
[hldy_ind] [varchar](1) NULL,
[xmas_hldy_ind] [varchar](1) NULL,
[easter_hldy_ind] [varchar](1) NULL,
[last_day_in_month_flag] [varchar](1) NULL,
[same_weekday_year_ago] [datetime] NULL,
[week_begin_date] [datetime] NULL,
[report_period_01_flag] [varchar](1) NOT NULL DEFAULT ('N'),
[report_period_02_flag] [varchar](1) NOT NULL DEFAULT ('N'),
[report_period_03_flag] [varchar](1) NOT NULL DEFAULT ('N'),
[report_period_04_flag] [varchar](1) NOT NULL DEFAULT ('N'),
[report_period_05_flag] [varchar](1) NOT NULL DEFAULT ('N'),
[week_day_ind] [char](1) NULL DEFAULT ('N'),
[year_month_num] [int] NULL DEFAULT ((0))
```

```
[td0005_key_ag1] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag1]  DEFAULT ((0)),
[td0005_key_ag2] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag2]  DEFAULT ((0)),
[td0005_key_ag3] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag3]  DEFAULT ((0)),
[td0005_key_ag4] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag4]  DEFAULT ((0)),
[td0005_key_ag5] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag5]  DEFAULT ((0)),
[td0005_key_ag6] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag6]  DEFAULT ((0)),
[td0005_key_ag7] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag7]  DEFAULT ((0)),
[td0005_key_ag8] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag8]  DEFAULT ((0)),
[td0005_key_ag9] [int] NOT NULL CONSTRAINT [DF_TD0005_td0005_ag9]  DEFAULT ((0)),
```

- Note the primary key
- Level column set to “detail” and level1-9
- Dim_char_ky_fld set to character string key
- Various fields describing the date
- These fields can be created in Excel and loaded
- Notice 9 aggregate keys

The Multi Level Day Table

```
SELECT TOP 1000 [pk_TD0005]
      ,[level_col]
      ,[dim_char_ky_fld]
      ,[day_date]
      ,[td0005_key_ag1]
      ,[td0005_key_ag2]
      ,[td0005_key_ag3]
      ,[td0005_key_ag8]
      ,[td0005_key_ag9]
  FROM [EBIHS_C006_DWH].[dbo].[TD0005]
  order by 1
```

- Note the primary key is unique and sequential
- Level column set to “detail” and level1-8 + “Total”
- Levels are Day, Week, Months, Quarter, Year, Total
- Dim_char_ky_fld set to character string key
- Day date included to show the date
- Other fields removed from slide
- Notice aggregate keys for all higher levels on rows
- Detail rows have all levels keys above them set

PK	level_col	char_ky_fld	day_date	ag1	ag2	ag3	ag8	g9
2	total	directive_total	1990-01-01 00:00:00.000	0	0	0	0	0
3	level8	1990	1990-01-01 00:00:00.000	0	0	0	0	2
4	level3	19901	1990-01-01 00:00:00.000	0	0	0	3	2
5	level2	19901	1990-01-01 00:00:00.000	0	0	4	3	2
6	level1	19901	1990-01-01 00:00:00.000	0	5	4	3	2
7	detail	1990-01-01	1990-01-01 00:00:00.000	6	5	4	3	2
8	detail	1990-01-02	1990-01-02 00:00:00.000	6	5	4	3	2
9	detail	1990-01-03	1990-01-03 00:00:00.000	6	5	4	3	2
10	detail	1990-01-04	1990-01-04 00:00:00.000	6	5	4	3	2
11	detail	1990-01-05	1990-01-05 00:00:00.000	6	5	4	3	2
12	detail	1990-01-06	1990-01-06 00:00:00.000	6	5	4	3	2
13	detail	1990-01-07	1990-01-07 00:00:00.000	6	5	4	3	2
14	level1	19902	1990-01-08 00:00:00.000	0	5	4	3	2
15	detail	1990-01-08	1990-01-08 00:00:00.000	14	5	4	3	2

The Day View

```
create view [dbo].[vm_day] as select
TD0005.pk_TD0005            pk_vm_day
,TD0005.day_date            day_date
,TD0005.day_name            day_name
...
,TD0005.level_col           level_col
,TD0005.dim_char_ky_fld     dim_char_ky_fld
,TD0005.td0005_key_ag1      dk_vm_week
,TD0005.td0005_key_ag2      dk_vm_month
,TD0005.td0005_key_ag3      dk_vm_quarter
,TD0005.td0005_key_ag8      dk_vm_year
,TD0005.td0005_key_ag9      dk_vm_time_total
from dbo.TD0005
where ( TD0005.level_col  = 'detail' ) or  TD0005.pk_TD0005 = 0
```

- MSTR sees vm_day as identical to a “table”
- Notice the “id” columns for higher level keys are visible on the view so MSTR can use them in hierarchies
- On the next slides we will show week, month, quarter, year views

PK	day_date	level_col	char_ky	week	month	quarter	year	total
7	1990-01-01 00:00:00.000	detail	1990-01-01	6	5	4	3	2
8	1990-01-02 00:00:00.000	detail	1990-01-02	6	5	4	3	2
9	1990-01-03 00:00:00.000	detail	1990-01-03	6	5	4	3	2
10	1990-01-04 00:00:00.000	detail	1990-01-04	6	5	4	3	2
11	1990-01-05 00:00:00.000	detail	1990-01-05	6	5	4	3	2
12	1990-01-06 00:00:00.000	detail	1990-01-06	6	5	4	3	2
13	1990-01-07 00:00:00.000	detail	1990-01-07	6	5	4	3	2
15	1990-01-08 00:00:00.000	detail	1990-01-08	14	5	4	3	2
16	1990-01-09 00:00:00.000	detail	1990-01-09	14	5	4	3	2

The Multi Level Day Table

Week

PK	week_begin_date	dk_vm_month	dk_vm_quarter	dk_vm_year	dk_vm_time_total
6	1990-01-01 00:00:00.000	5	4	3	2
14	1990-01-08 00:00:00.000	5	4	3	2
22	1990-01-15 00:00:00.000	5	4	3	2
30	1990-01-22 00:00:00.000	5	4	3	2
38	1990-01-29 00:00:00.000	5	4	3	2
47	1990-02-05 00:00:00.000	42	4	3	2
55	1990-02-12 00:00:00.000	42	4	3	2
63	1990-02-19 00:00:00.000	42	4	3	2
71	1990-02-26 00:00:00.000	42	4	3	2

Month

PK	first_day_of_month	month_name	dk_vm_quarter	dk_vm_year	dk_vm_time_total
5	1990-01-01 00:00:00.000	January	4	3	2
42	1990-02-01 00:00:00.000	February	4	3	2
75	1990-03-01 00:00:00.000	March	4	3	2
112	1990-04-01 00:00:00.000	April	111	3	2
148	1990-05-01 00:00:00.000	May	111	3	2
184	1990-06-01 00:00:00.000	June	111	3	2

Quarter

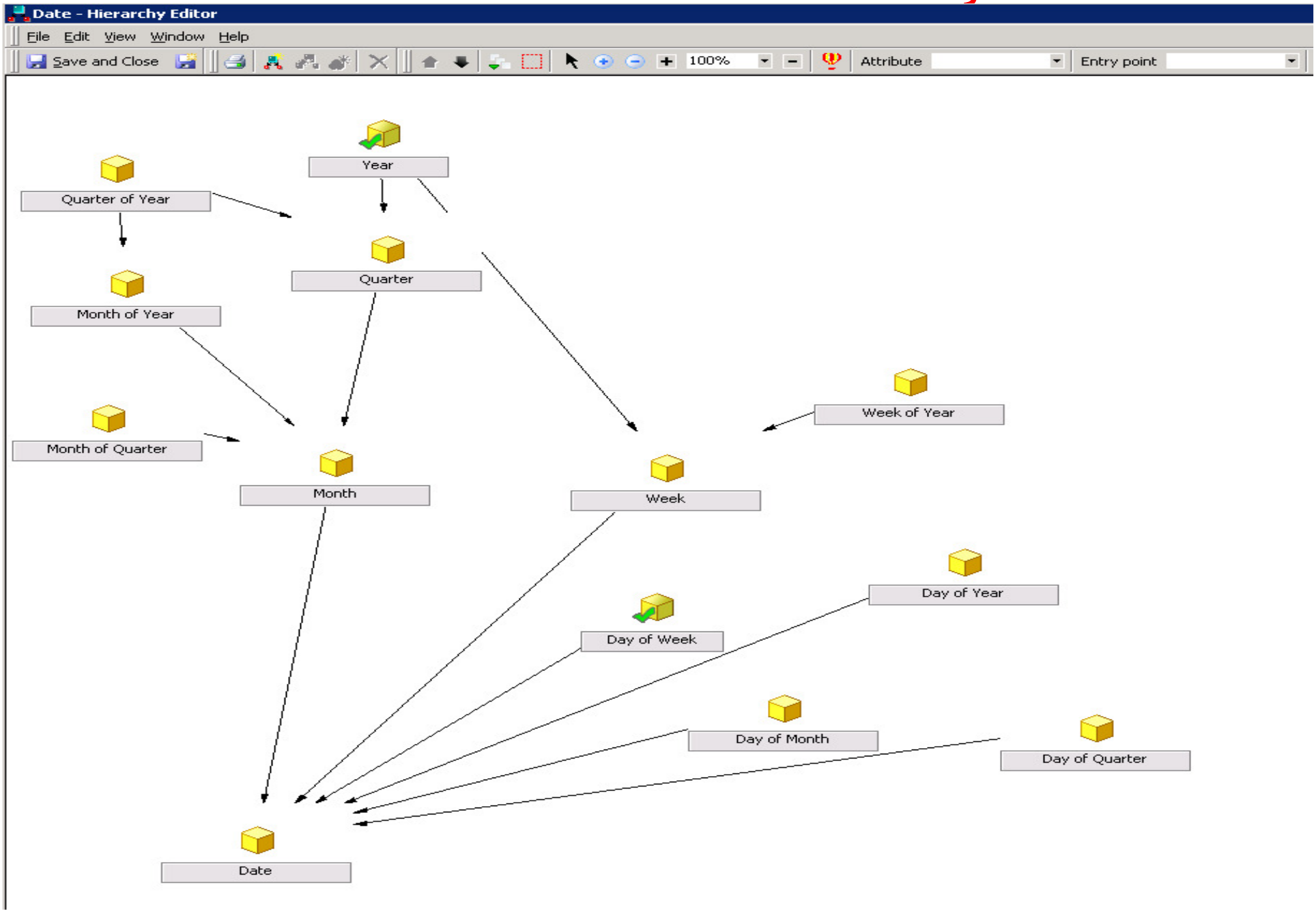
PK	first_day_of_quarter	dk_vm_year	dk_vm_time_total
4	1990-01-01 00:00:00.000	3	2
111	1990-04-01 00:00:00.000	3	2
219	1990-07-01 00:00:00.000	3	2
328	1990-10-01 00:00:00.000	3	2
439	1991-01-01 00:00:00.000	438	2

Year

PK	first_day_of_year	year_name	dk_vm_time_total
3	1990-01-01 00:00:00.000	1990	2
438	1991-01-01 00:00:00.000	1991	2
873	1992-01-01 00:00:00.000	1992	2

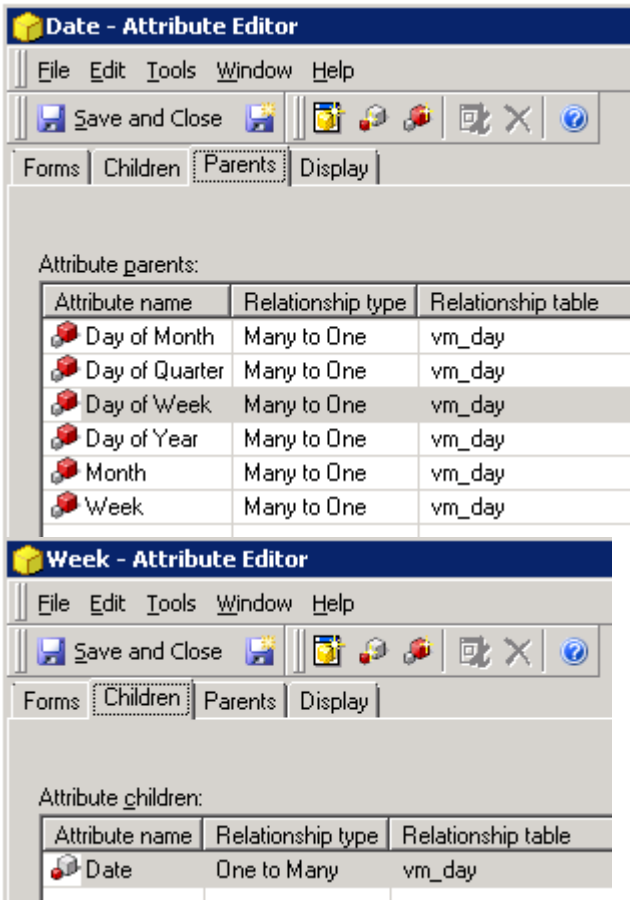
EMPOWER BI
knowledge that works

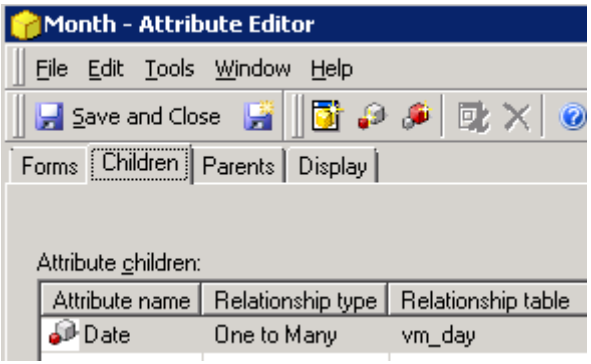
MSTR Date Hierarchy

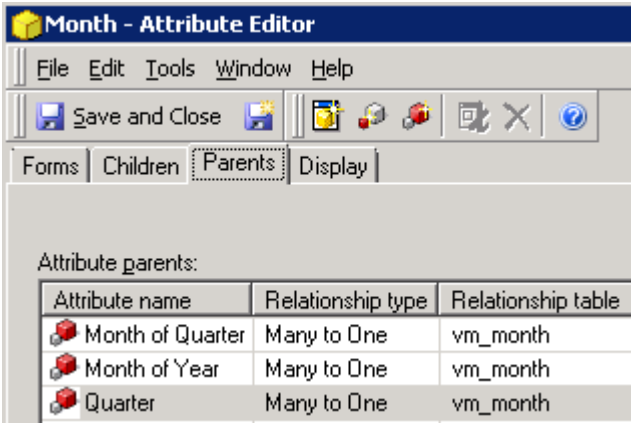


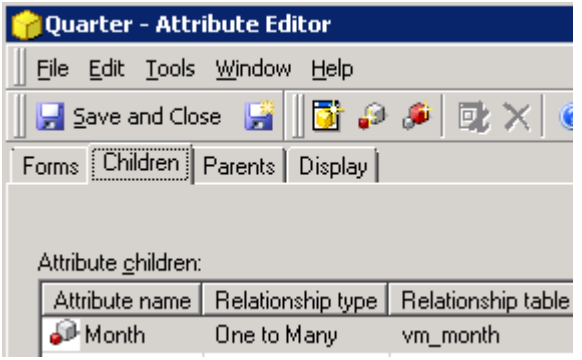
EMPOWER BI
knowledge that works

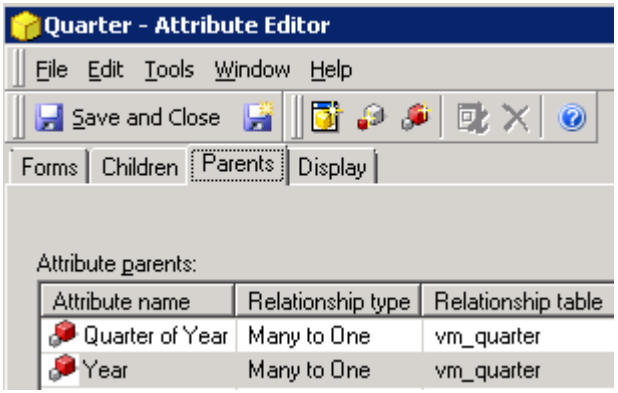
The MSTR Date Hierarchy

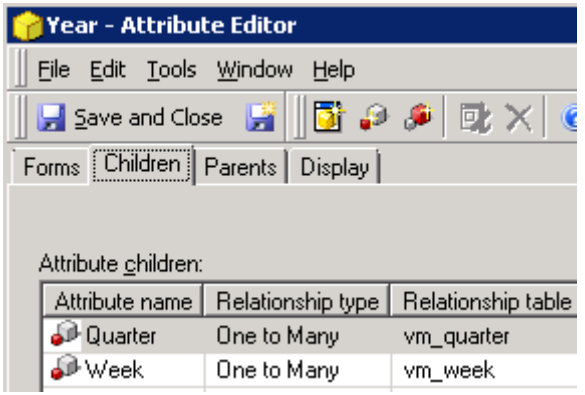












- Notice the hierarchies set up just as normal
- They are all using views, not tables

EMPOWER BI
knowledge that works

The Multi Level Invoice Line Table

- So now we are ready to create multi-level facts
- These are accounts payable invoice lines, just for example
- We have day, week, month, quarter and year
  - vf_ap_invoice_line
  - vf_ap_invoice_line_week
  - vf_ap_invoice_line_month
  - vf_ap_invoice_line_quarter
  - vf_ap_invoice_line_year
- The day level is “detail” and has its own table
- The higher levels all cohabit the same table with a summary number in the view…see over

The Multi Level Invoice Line Table

- View snippets are as follows…

```
create view [dbo].[vf_ap_invoice_line] as select
TF0101.pk_TF0101  pk_vf_ap_invoice_line
.....
from dbo.TF0101
where TF0101.table_number = 508
```

```
create view [dbo].[vf_ap_invoice_line_week] as select
...

from z01_vf_ap_invoice_line_01_summary
where pk_aggregate_number = 201
```

```
create view [dbo].[vf_ap_invoice_line_month]
as select
...

from z01_vf_ap_invoice_line_01_summary
where pk_aggregate_number = 202
```

```
create view [dbo].[vf_ap_invoice_line_quarter]
as select
...

from z01_vf_ap_invoice_line_01_summary
where pk_aggregate_number = 203
```

```
create view [dbo].[vf_ap_invoice_line_year]
as select
...

from z01_vf_ap_invoice_line_01_summary
where  pk_aggregate_number = 204
```

- So now we will show you a short video that captures the SQL from MSTR and shows you that the summary levels are accessed properly
- We will also include the SQL on the following slides
- We just want to prove MSTR navigates the summaries perfectly!

EMPOWER BI
knowledge that works

Video Demonstration

The Multi Level Invoice Line Table

- SQL snippets are as follows…

Day

```
select      a11.dk_vm_invoice_date  pk_vm_day,
            max(CONVERT(DATETIME,
            CONVERT(VARCHAR(10), a12.day_date, 101)))  day_date,
            sum(a11.amount_col)  WJXBFS1
from        vf_ap_invoice_line          a11
            join        vm_day          a12
            on          (a11.dk_vm_invoice_date = a12.pk_vm_day)
group by    a11.dk_vm_invoice_date
```

Week

```
select      a11.dk_vm_invoice_week  pk_vm_week,
            max(a12.week_in_year)  week_in_year,
            sum(a11.amount_col)  WJXBFS1
from        vf_ap_invoice_line_week     a11
            join        vm_week         a12
            on          (a11.dk_vm_invoice_week =
                        a12.pk_vm_week)
group by    a11.dk_vm_invoice_week
```

Month

```
select      a11.dk_vm_invoice_month  pk_vm_month,
            max(a12.month_name_sdesc)  month_name_sdesc,
            sum(a11.amount_col)  WJXBFS1
from        vf_ap_invoice_line_month    a11
            join        vm_month        a12
            on          (a11.dk_vm_invoice_month =
                        a12.pk_vm_month)
group by    a11.dk_vm_invoice_month
```

Quarter

```
select      a11.dk_vm_invoice_quarter  pk_vm_quarter,
            max(a12.calendar_qtr)  calendar_qtr,
            sum(a11.amount_col)  WJXBFS1
from        vf_ap_invoice_line_quarter  a11
            join        vm_quarter      a12
            on          (a11.dk_vm_invoice_quarter =
                        a12.pk_vm_quarter)
group by    a11.dk_vm_invoice_quarter
```

Year

```
select      a11.dk_vm_invoice_year  pk_vm_year,
            max(a12.year_num)  year_num,
            sum(a11.amount_col)  WJXBFS1
from        vf_ap_invoice_line_year     a11
            join        vm_year         a12
              on        (a11.dk_vm_invoice_year = a12.pk_vm_year)
group by    a11.dk_vm_invoice_year
```

- Notice how the from and join clauses change to get data from the correct level of the day table and correct level of the ap invoice line fact tables.
- MSTR navigates summaries perfectly
- EBI Builds summaries perfectly

EMPOWER BI
knowledge that works

A Parting Word

- Multi Level Summaries are the #1 tool to create fast queries inside the database
- Using separate tables requires separate ETL
- With our Big Data Accelerator product we can create multi-level summaries simply and easily
- We use special multi-level aware ETL to populate the multi-level summaries with no extra work
- We look forward to helping you create your summaries faster, more easily and, most importantly, less expensively

EMPOWER BI
knowledge that works

Summary

- What is multi-level data (MLD)?
- Why is MLD important?
- How is MLD best implemented?
- How do you create a schema to navigate it?
- Fully worked example
- A parting word
- Summary

EMPOWER BI
knowledge that works

Thank You for Your Time!