Managing Reduction in Multidimensional Databases

Franck Ravat[1], Jiefu Song[1], Olivier Teste[2]

[1]IRIT – Université Toulouse I Capitole, 2 Rue du Doyen Gabriel Marty
F-31042 Toulouse Cedex 09
[2]IRIT – Université Toulouse II Jean Jaurès, 1 Place Georges Brassens
F-31703 Blagnac Cedex
[1,2]{ravat|song|teste}@irit.fr

Abstract. Dealing with large amount of data has always been a key focus of the Multidimensional Database (MDB) community, especially in the current era when data volume increases more and more rapidly. In this paper, we outline a conceptual modeling solution allowing reducing data in MDBs. A MDB after reduction is modeled with multiple states. Each state is valid during a period of time and aggregates data from a more recent state. We propose three alternatives of reduced MDB modeling at the logical level: (i) the *flat* modeling integrates all states into one single table, (ii) the *horizontal* modeling converts each state into a fact table and some dimension tables associated with a temporal interval and (iii) the *vertical* modeling breaks down a reduced MDB into separate tables, each table includes data from one or several states. We evaluate query execution efficiency in MDBs with and without data reduction. The result shows data reduction is an interesting solution, since it significantly decreases execution costs by 98.96% during our experimental assessments.

Keywords: Data Reduction, Relational Multidimensional Design, Experimental Assessments

1 Introduction

Multidimensional Databases (MDBs) are widely used in decision-support systems. A MDB organizes data according to analysis subjects (i.e. facts) and analysis axes (i.e. dimensions). A fact includes a set of numeric indicators (i.e. measures), while a dimension contains one or several granularities (i.e. levels). In today's highly competitive business context, data coming from both inside and outside a company are periodically added and then permanently stored in a MDB [2, 7]. The huge amount of data in a MDB slows down query execution, not to mention that decision-makers may easily get lost while facing all detailed data during analyses. Meanwhile, all data do not keep the same informative value over time. While detailed information is important for recent data, it may be of less interest for older data [10].

Reducing data can avoid an overly large MDB. It allows decreasing the amount of useless data and thus increasing query execution efficiency [11]. As detailed data lose their informative value over time, a data reduction solution should allow selectively

deleting useless data in a MDB. Moreover, it is necessary to aggregate data progressively, so that information is not lost after reduction but represented in a summarized form for comparative or trend analyses. This is achieved by eliminating a MDB's content deprecated for business analyses.

Our aim is to support effective and efficient decision-making by storing only data of high informative value over time in a MDB. In our previous work [1], we proposed a conceptual modeling solution for *MDBs with data reduction* (i.e. reduced MDBs). As modeling solutions at the logical level are seldom studied for MDBs whose schema changes over time, this paper focuses on the relational modeling of reduced MDBs. Some algorithms are proposed to automatically transform a conceptual reduced MDB into different relational forms. We carry out some experimental assessments to compare query execution efficiency in reduced and unreduced MDBs.

The paper is organized as follows: section 2 discusses the representative work related to data reduction; section 3 describes three relational modeling solutions and a schema design process for reduced MDBs; section 4 illustrates the benefits of reduced MDBs through some experimental assessments.

2 Related Work

Data reduction is a technique originally used in the data mining field [8]. In this context, data reduction aims at improving the accuracy of mining results by extracting significant and relevant features of sources. In the database field, data reduction is adapted to automatically delete expired data which are no longer of interest. We can cite the work [4] which enables data reduction by deleting content in materialized views. In the MDB field, related work focuses on reducing data in a fact. The authors of [10] describe a solution for the progressive aggregation and deletion of data in a fact. A set of criteria is proposed to summarize data according to higher granularities. The authors of [6] present a complete data reduction process. They study the conception, implementation, and influence of data reduction in a MDB's fact.

The above-mentioned work only allows reducing a fact. Our previous work [1] generalizes the reduction to a complete MDW. Consequently, both facts and dimensions can be reduced. Moreover, unlike some automatic reduction solutions, our proposed approach involves decision-makers in a reduction process. A designer determines with a decision-maker within which temporal interval a MDB schema is valid. As detailed information is often irrelevant to analyses over an old period, a MDB schema includes different contents over time. The further we look back in time, the fewer detailed data a MDB schema contains.

Specifically, a reduced MDB is composed of a set of *states* $\mathcal{E} = \{E_1; \ldots; E_n\}$. The state E_n is the latest state including the most complete schema, while the other states consist of a succession of reduced schemas over time. Each *state* E_i ($E_i \in \mathcal{E}$) corresponds to a star schema including a *fact* F_i and a set of *dimensions* $\mathcal{D}_i$ with necessarily a temporal dimension. The *state* E_i is stamped with a validation period $T_i = [t^i_{start}; t^i_{end}]$ defined on the temporal dimension. The *fact* F_i contains a set of *measures* $\mathcal{M}^{Fi} = \{m_1; \ldots; m_p\}$, while a *dimension* D_k ($D_k \in \mathcal{D}_i$) includes a set of *attributes* $\mathcal{A}^{Dk} = \{a_1; \ldots;$

a_q} organized in different *level*s. The set of instances of the measure m_s is denoted as *val*(m_s), while the set of instances of the attribute a_k is denoted as *dom*(a_k). We distinguish two types of attributes: a *parameter* p_x ($p_x \in \mathcal{A}^{Dk}$) is an attribute allowing identifying an unique *level* on the dimension D_k, while a *weak attribute* is a non-identifier attribute providing descriptive information to a *parameter*.

Based on this conceptual reduced MDB modeling, our previous work [1] presents some multidimensional analyses operators allowing (i) choosing analysis subjects and axes (i.e. *Display*), (ii) aggregating data (i.e. *Drilldown* and *Rollup*), (iii) changing analysis axes (i.e. *Rotate*), and (iii) filtering analysis results (i.e. *Select*).

In this paper, we complete the reduced MDB modeling by studying modeling alternatives at the logical level. The efficiency of each alternative will be studied through some experimental assessments.

3 Relational Modeling of MDBs with Data Reduction

In this section, we describe the logical modeling of reduced MDBs. This modeling is based on three relational modeling alternatives. An algorithm is proposed for each alternative to automate the transformation from a conceptual reduced MDB into a relational reduced MDB.

3.1 Case Study

A MDB contains a fact, named *Sales*, which includes one measure named *Amount*. The measure can be calculated along three dimensions, namely *Products*, *Customers* and *Times*. The current MDB contains all sale data from 1990 to 2017. However, since most today's products and customers did not exist before, the MDB is reduced by creating three states as follows: (i) the latest state E_3 contains all detailed data within all dimensions from 2010 to 2017; (ii) the second state E_2 includes aggregated data starting from products' *Range*, customers' *Town* and sale date ID_{Time} between 2000 to 2010; (iii) the oldest state E_1 supports historical sales analyses by products' *Sector* and *Year* from 1900 to 2000. Figure 1 shows the reduced MDB's states according to the graphical notation proposed in [5].

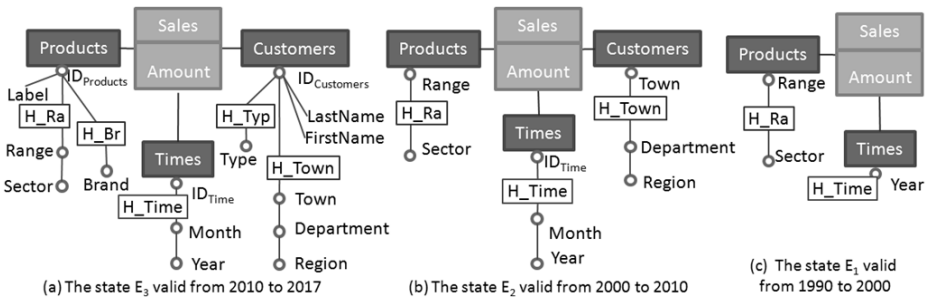


Fig. 1. Reduced MDB schema evolutions over time

3.2 *Flat* modeling of a reduced MDB

The first alternative is called *flat* modeling. It integrates all states into one single flat table. All attributes and all measures before data reduction constitute the columns of a flat table. We propose the following algorithm for the *flat* modeling.

Algorithm 1. *Flat Modeling*

Input: a reduced MDB composed of a set of states $\mathcal{E} = \{E_1; \ldots; E_n\}$.

Output: a flat table T_{Flat} = (SynKey, $\mathcal{A}_{Flat}$, $\mathcal{M}_{Flat}$), where SynKey is a synthetic key; $\mathcal{A}_{Flat}=\{a_1; ...; a_m\}$ is a set of attributes; $\mathcal{M}_{Flat}=\{m_1; \ldots; m_p\}$ is a set of measures.

Begin

1. *Find the latest state* E_n *of the reduced* MDB, $E_n \in \mathcal{E}$, $E_n=\{F_n; \mathcal{D}_n; T_n\}$;
2. *Create a flat table* T_{Flat}, *set* $\mathcal{A}_{Flat} \leftarrow \bigcup_{D_i \in \mathcal{D}_n} \mathcal{A}^{D_i}$, *set* $\mathcal{M}_{Flat} \leftarrow \mathcal{M}^{F_n}$;
3. For each $E_i \in \mathcal{E}$
4. For each $a_k \in \mathcal{A}_{Flat}$ do,
5. If $a_k \notin \bigcup_{D_j \in \mathcal{D}_i} \mathcal{A}^{D_j}$ then *set dom*$(a_k) \leftarrow \emptyset$;
6. End for
7. For each $m_s \in \mathcal{M}_{Flat}$ do,
8. If $m_s \notin \mathcal{M}^{F_i}$ then *set val*$(m_s) \leftarrow \emptyset$;
9. End for
10. *Insert into* T_{Flat} *with tuples* $(i_{a_1}, \ldots, i_{a_m}, i_{m_1}, \ldots, i_{m_p})$, *such as* $\forall i_{a_k} \in \{i_{a_1}; \ldots; i_{a_m}\}$, $i_{a_k} \in dom(a_k)$ *and* $\forall i_{m_s} \in \{i_{m_1}; \ldots; i_{m_p}\}$, $i_{m_s} \in val(m_s)$;
11. End for

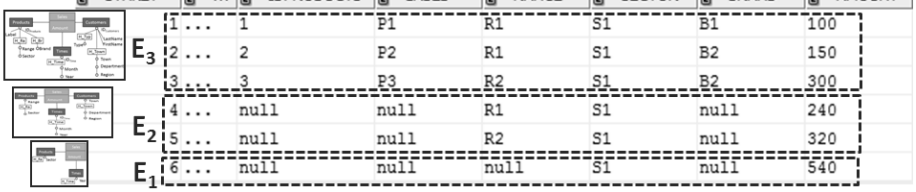
End

After creating the structure of a flat table (*cf.* lines 1 and 2), the algorithm extracts data from each state and loads the flat table. Specifically, if the attribute a_k (or the measure m_s) from the flat table does not exist in the state E_i, the algorithm assigns an empty set to its instances (*cf.* lines 3-9). Then, measure instances and related attribute instances from each state are loaded in the flat table (*cf.* line 10). The time span of a flat table corresponds to the union of all states' temporal intervals.

Example. We apply the algorithm 1 to the reduced MDB of our case study. The relational schema of the output flat table is as follows.

```
FLAT_SALES (SYNKEY, IDTIME, MONTH, YEAR, IDCUSTOMERS, LASTNAME,
            FIRSTNAME, TOWN, DEPARTMENT, REGION, TYPE, IDPRODUCTS,
            LABEL, RANGE, SECTOR, BRAND, AMOUNT)
```

time span [1990, 2017]



	SYNKEY	...	IDPRODUCTS	LABEL	RANGE	SECTOR	BRAND	AMOUNT
E_3	1	...	1	P1	R1	S1	B1	100
	2	...	2	P2	R1	S1	B2	150
	3	...	3	P3	R2	S1	B2	300
E_2	4	...	null	null	R1	S1	null	240
	5	...	null	null	R2	S1	null	320
E_1	6	...	null	null	null	S1	null	540

Fig. 2. A snapshot of instances organized according to the *flat* modeling

A snapshot[1] of instances in the flat table is shown in the figure 2. Instances from the latest state E_3 are directly loaded in the flat table (*cf.* lines 3 and 10), while the other two states E_2 and E_1 are loaded with *NULL* value as placeholder for the deleted attributes' instances (cf. lines 3-6 and 10).

3.3 *Horizontal* modeling of a reduced MDB

The second relational modeling alternative is named *horizontal*. Each state is implemented through a *fact table* and a set of *dimension tables*. The algorithm of the *horizontal* modeling is as follows.

Algorithm 2. *Horizontal Modeling*

Input: a reduced MDB composed of a set of states $\mathcal{E} = \{E_1; \ldots; E_n\}$.

Outputs:

- a set of fact tables $\mathcal{T}_{Fact}=\{T_{F_1}; \ldots; T_{F_n}\}$, such as $\forall T_{F_i} \in \mathcal{T}_{Fact}$, $T_{F_i}=(\underline{SynKey_i}, \mathcal{FKey}_i, \mathcal{M}_i)$ implements the fact F_i of the state E_i, where $SynKey_i$ is a synthetic primary key; $\mathcal{FKey}_i$ is a set of foreign keys; $\mathcal{M}_i$ is a set of measures in T_{F_i}.
- a set of dimension tables $\mathcal{T}_{Dim}=\{T_{D_1^{E_1}}; \ldots; T_{D_w^{E_n}}\}$, such as $\forall T_{D_j^{E_i}} \in \mathcal{T}_{Dim}$, $T_{D_j^{E_i}} = (\underline{Key_{T_{ji}}}, \mathcal{A}_{T_{ji}})$ implements the dimension D_j of the state E_i, where $Key_{T_{ji}}$ is a primary key of the dimension table; $\mathcal{A}_{T_{ji}}$ is a set of attributes.

Begin

1. For each state $E_i \in \mathcal{E}$, $E_i=\{F_i; \mathcal{D}_i; T_i\}$
2. *Create a fact table* $T_{F_i} = (\underline{SynKey_i}, \mathcal{FKey}_i, \mathcal{M}_i)$, *where* $\mathcal{FKey}_i \leftarrow \emptyset$, $\mathcal{M}_i \leftarrow \mathcal{M}^{F_i}$;
3. For each dimension $D_j \in \mathcal{D}_i$
4. *Find the parameter* p_1 *on the lowest granularity of* D_i ;
5. $\mathcal{FKey}_i \leftarrow \mathcal{FKey}_i \cup \{p_1\}$;
6. *Create a dimension table* $T_{D_j^{E_i}}=(\underline{Key_{T_{ji}}}, \mathcal{A}_{T_{ji}})$, *set* $Key_{T_{ji}} \leftarrow p_1$, $\mathcal{A}_{T_{ji}} \leftarrow \mathcal{A}^{D_i} \backslash \{p_1\}$;
7. *Insert attribute instances within* D_j *into* $T_{D_j^{E_i}}$;
8. End for
9. *Insert measure instances within* F_i *with related parameter instances into* T_{F_i}
10. End for

End

The *horizontal* modeling creates a fact table T_{F_i} for each state E_i. Each fact table includes all measures from the fact F_i and a set of foreign keys (*cf.* lines 1 and 2). Each foreign key consists of the parameter on the lowest granularity of a dimension from the state E_i (*cf.* line 5). Each dimension D_j is converted into a dimension table as follows: the parameter p_1 of the lowest granularity on D_j is used as a primary key, while other attributes on the dimension (i.e. $\mathcal{A}^{D_i} \backslash \{p_1\}$) are directly added in the dimension table (*cf.* lines 3-6). Consequently, the time span of a fact table and a dimension table corresponds to the temporal interval of the involved state.

[1] For the sake of simplicity, all snapshots in this section include only the dimension *Products*.

Example. According to the algorithm 2, the reduced MDB of the case study is implemented through 3 fact tables and 8 dimension tables.

```
E3_TIMES     (IDTIME, MONTH, YEAR)
E3_CUSTOMERS (IDCUSTOMERS, LASTNAME, FIRSTNAME, TOWN, DEPARTMENT,
              REGION, TYPE)
E3_PRODUCTS  (IDPRODUCTS, LABEL, RANGE, SECTOR, BRAND)
E3_SALES     (SYNKEY, IDTIME#, IDCUSTOMERS#, IDPRODUCTS#, AMOUNT)
E2_TIMES     (IDTIME, MONTH, YEAR)
E2_CUSTOMERS (TOWN, DEPARTMENT, REGION)
E2_PRODUCTS  (RANGE, SECTOR)
E2_SALES     (SYNKEY, IDTIME#, TOWN#, RANGE#, AMOUNT)
E1_TIMES     (YEAR)
E1_PRODUCTS  (SECTOR)
E1_SALES     (SYNKEY, YEAR#, SECTOR#, AMOUNT)
```

Figure 3 displays a snapshot of instances in the reduced MDB implemented according to the *horizontal* modeling.

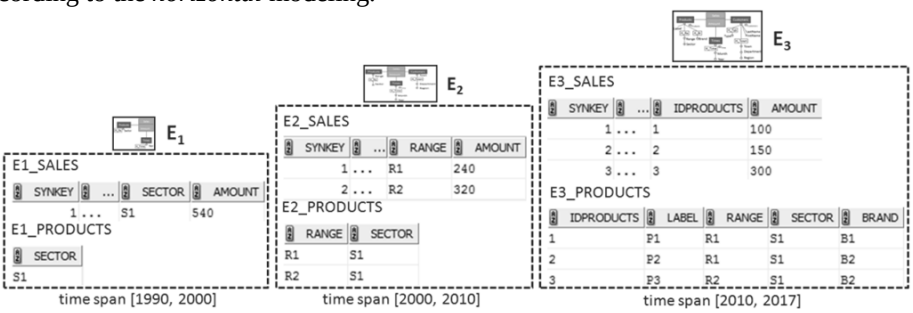


Fig. 3. A snapshot of instances organized according to the *horizontal* modeling

3.4 *Vertical* modeling of a reduced MDB

The third alternative is named *vertical* modeling. It gathers common components among states into separate tables called vertical tables. Each vertical table includes measures and attributes shared by some states. We propose the following algorithm for the *vertical* modeling.

Algorithm 3. *Vertical Modeling*

Input: a reduced MDB composed of a set of states $\mathcal{E} = \{E_1, \ldots, E_n\}$.

Output: a set of vertical tables $\mathcal{T}_V=\{T_{V_1}, \ldots, T_{V_n}\}$, such as $\forall T_{V_i} \in \mathcal{T}_V$, $T_{V_i}=\{\underline{SynKey_i}, \mathcal{A}_i, \mathcal{M}_i\}$ is a vertical table for a subset of states, where $SynKey_i$ is a synthetic key; $\mathcal{A}_i$ is a set of attributes; $\mathcal{M}_i$ is a set of measures.

Begin

1. For each i from 1 to n ($n=|\mathcal{E}|$)
2. Create a vertical table $T_{V_i}=\{\underline{SynKey_i}, \mathcal{A}_i, \mathcal{M}_i\}$, where
   - $\mathcal{A}_i \leftarrow \bigcup_{D_k \in \mathcal{D}_i} \mathcal{A}^{D_k}$ such as $\mathcal{D}_i$ is the set of dimensions from the state E_i;
   - $\mathcal{M}_i \leftarrow \mathcal{M}^{F_i}$ such as F_i is the fact from the state E_i;

3. For each $E_x \in \mathcal{E}$
4. *Insert into* T_{V_i} *instances of attributes* $\mathcal{A}_i$;
5. *Insert into* T_{V_i} *aggregated values of measures* $\mathcal{M}_i$ *from* E_x *according to* $\mathcal{A}_i$;
6. End For
7. $\mathcal{E} \leftarrow \mathcal{E} \setminus \{E_i\}$;
8. End for

End

According to the definition of the data reduction, attributes and measures from an old state E_i must exist in a more recent state E_j ($i<j$). Therefore, to gathers common components in a subset of states $\{E_i, \ldots, E_n\}$ ($1 \leq i \leq n$), the ith vertical table T_{V_i} groups together attributes and measures from the ith state (*cf.* lines 1 and 2). Then, for each state E_x in $\{E_i, \ldots, E_n\}$, instances of each attribute in $\mathcal{A}_i$ are retrieved from the state E_x and then loaded in T_{V_i}. Based on the attribute instances, values of each measure in $\mathcal{M}_i$ from E_x are aggregated and then inserted into T_{V_i} (*cf.* lines 3-6). Consequently, each vertical table T_{V_i} covers a time span from the state E_i to the latest state E_n.

Example. After applying the algorithm 3 to our case study, we obtain the following three vertical tables.

```
VTABLE1  (SYNKEY, YEAR, SECTOR, AMOUNT)
VTABLE2  (SYNKEY, IDTIME, MONTH, YEAR, TOWN, DEPARTMENT, REGION,
          RANGE, SECTOR, AMOUNT)
VTABLE3  (SYNKEY, IDTIME, MONTH, YEAR, IDCUSTOMERS, LASTNAME,
          FIRSTNAME, TOWN, DEPARTMENT, REGION, TYPE, IDPRODUCTS,
          LABEL, RANGE, SECTOR, BRAND, AMOUNT)
```

The snapshot presented in figure 4 indicates a state of reduced MDB is implemented through one or several vertical tables. For instance, data from the latest state E_3 are found within all vertical tables: (i) *VTABLE3* includes the sale *amount* from 2010 to 2017 by *IDProducts*; (ii) *VTABLE2* aggregates the *amount* from the state E_3 according to products' *range*; (iii) *VTABLE3* further aggregates the *amount* from the state E_3 according to product's *sector*.

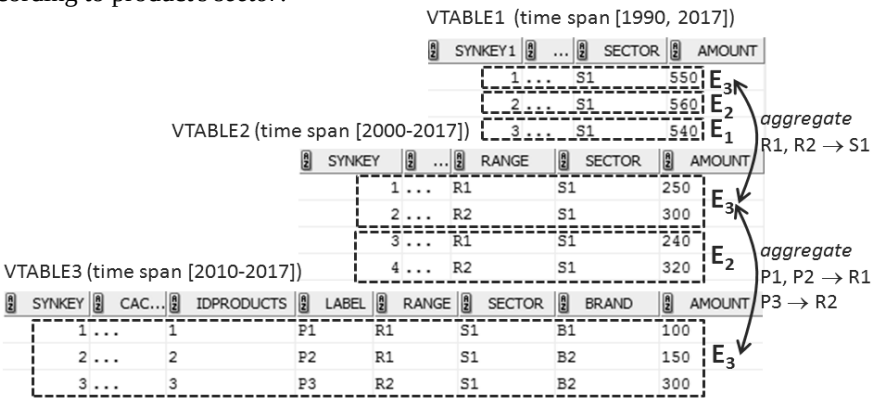


Fig. 4. A snapshot of instances organized according to the *vertical* modeling

3.5 Comparison among relational modeling alternatives

A conceptual reduced MDB can be transformed into various relational schemas. Extracting the same data requires applying different queries to different relational schemas with different data redundancy ratios.

The *flat* modeling consists of a simplistic way which converts the whole reduced MDB into one relation. It frees queries from joins, regardless of the number of involved dimensions. However, the *flat* modeling causes high data redundancy: attribute instances are repetitively stored in the relation with related measure instances.

The *horizontal* modeling is a more complex method which converts measures and attributes from one state into independent relations. It minimizes data redundancy by associating attribute instances with related measure instances through *primary key - foreign key* relationships. However, the *horizontal* modeling requires joins in queries involving dimension tables.

The *vertical* modeling converts measures and attributes shared by several states into separate relations. This modeling has multiple advantages. On one hand, queries involving several dimensions do not have to include joins. On the other hand, data redundancy is reduced to attribute instances within some high levels on dimensions.

To accurately and quantitatively study the influences of different relational modeling alternatives on query execution efficiency, the remainder of this paper focuses on some experimental assessments.

4 Experimental assessments

In this section, we carry out some experimental assessments by executing queries in reduced and unreduced MDBs populated with data according to different volumes.

4.1 Protocol

The objective of our experimental assessments is twofold: (i) studying if all relational modeling alternatives for reduced MDBs help improving query execution efficiency and (ii) identifying the most efficient relational modeling of reduced MDBs. Existing multidimensional data benchmarks (e.g. TPC-DS[2] and SSB[9]) are designed to measure a system's performance [3]. They do not allow testing the effect of different reduced modeling solutions, since the included MDB is composed of only one state.

Facing this issue, we have to generate our own test data during the experimental assessments. The MDB of our case study is used and populated with synthetic data. Three reduced MDB implementations, namely *flat*, *horizontal* and *vertical*, are built according to the relational modeling alternatives (*cf.* section 3). Two unreduced MDBs are used as baseline to assess the impact of data reduction: (i) the *unreduced flat* MDB integrates all attributes and measures before reduction into one table and (ii) the *unreduced horizontal* MDB includes one fact table and three dimension tables

[2] http://www.tpc.org/tpcds/

without reduction. The number of tuples as well as redundancy ratio of attribute instances according to MDB implementation and scale factor is shown in table 1.

Table 1. Scale factors and number of tuples with attribute instance redundancy ratio.

Relational Modeling	number of tuples and redundancy ratio of attribute instances			
	SF1	*SF2*	*SF3*	*SF4*
unreduced flat	10^6(99.0%)	2.5×10^7(99.6%)	10^8(99.9%)	4×10^8(99.9%)
unreduced horizontal	10^6 (≈0%)	2.5×10^7 (≈0%)	10^8(≈0%)	4×10^8(≈0%)
flat	3.2×10^5(97.8%)	8×10^6(99.9%)	3.2×10^7(99.9%)	1.27×10^8(99.9%)
horizontal	3.2×10^5 (≈0%)	8×10^6 (≈0%)	3.2×10^7 (≈0%)	1.27×10^8(≈0%)
vertical	3.4×10^5(91.6%)	8.4×10^6 (92.4%)	3.4×10^7 (92.5%)	1.34×10^8(92.5%)

During the experimental assessment, we consider only queries producing full answers in MDBs before and after data reduction. Meanwhile, different queries should involve different dimensions in different states during querying. Table 2 shows our proposed 12 queries. Specifically, queries Q_1-Q_3 involve one dimension in one state; queries Q_4-Q_7 involve multiple dimensions in one state; queries Q_8 and Q_9 involve different dimensions in two states; Q_{10}-Q_{12} involve different dimensions in all states.

Table 2. 12 queries involving different dimensions and time spans.

Query	No. of Dimensions	Time span and state
Q_1: Annual sale amount for the last three years	1	[2014, 2017] E_3
Q_2: Annual sale amount in 2008	1	[2008, 2008] E_2
Q_3: Annual sale amount before 2000	1	[1990, 2000] E_1
Q_4: Sale amount by customer from 2010 to 2012	2	[2010, 2012] E_3
Q_5: Monthly sale amount by town from 2000 to 2005	2	[2000, 2005] E_2
Q_6: Monthly sale amount by town and sector in 2012	3	[2012, 2012] E_3
Q_7: Annual sale amount by town and sector from 2000 to 2005	3	[2000, 2005] E_2
Q_8: Monthly sale amount since 2000	1	[2000, 2017] E_2, E_3
Q_9: Annual sale amount per town from 2002 to 2012	2	[2002, 2012] E_2, E_3
Q_{10}: Annual sale amount	1	[1990, 2017] E_1, E_2, E_3
Q_{11}: Annual sale amount per sector	2	[1990, 2017] E_1, E_2, E_3
Q_{12}: Total sale amount	n/a	[1990, 2017] E_1, E_2, E_3

For each query, we record the *execution costs* provided by the *Explain Plan* command of the Oracle 12c DBMS without any optimization techniques (e.g. index and table partitioning). The hardware configuration is as follows: 2×CPU@2.33GHz with 2 cores, 128GB RAM and 1TB SSD Disk in RAID6.

4.2 Observations and discussions

In this section, we study the query execution costs in reduced and unreduced MDBs of different scale factors.

Observation. From figure 5, we can see the same trend is found in MDBs of different scale factors. The lowest execution costs of the twelve queries come from different implementations of reduced MDB. Specifically, for queries covering a time span within the temporal interval of one state, regardless of the scale factor and the number of dimensions included, (i) the lowest execution costs of Q_1, Q_4 and Q_6 (with-

in the temporal interval of E_3) are found within the *vertical* MDB; (ii) the lowest execution costs of Q_2, Q_5 and Q_7 (within the temporal interval of E_2) are produced by the *horizontal* MDB; (iii) both the *vertical* and the *horizontal* MDBs are cost-efficient for Q3 (within the temporal interval of E_1). All queries involving multiple states are more efficiently computed within the *vertical* MDB (from Q8 to Q12), regardless of the MDB volume and the number of states as well as dimensions involved.

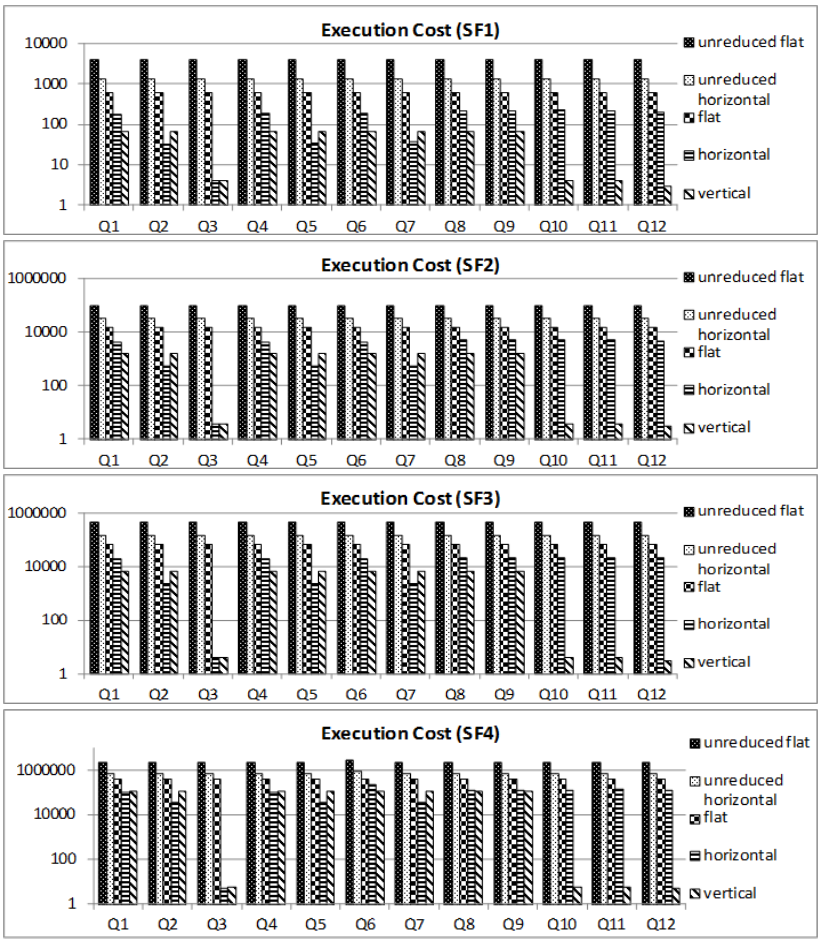


Fig. 5. Query execution costs in reduced and unreduced MDBs of different scale factors

Discussion. Based on the above observations, we can conclude that regardless of the scale factor, reduced MDBs always produce lower execution costs than unreduced MDBs. The execution costs in reduced MDBs (i) are not significantly influenced by the number of dimensions involved in a query but (ii) highly depend on the time span involved in a query. The more a query and a reduced MDB implementation share in terms of time span, the lower the execution costs become. When a query only in-

volves old states, the influence of time span is weakened by the small volume of data within the *horizontal* and the *vertical* MDBs.

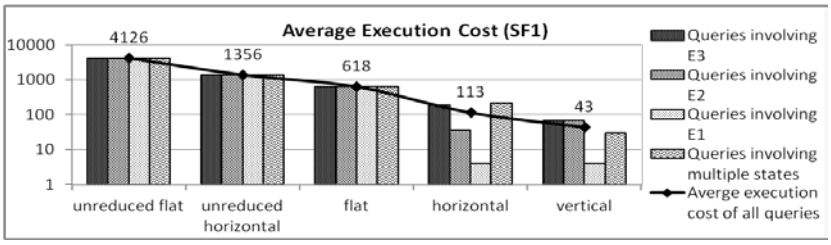


Fig. 6. Average execution costs by query type and MDB implementation of *SF1*

Observation. Figure 6 shows the average query execution costs in MDBs of the scale factor *SF1* according to query type. No matter how many states are involved in queries, the average execution costs in reduced MDBs are always lower than in unreduced MDBs. The highest average execution costs are found within the *unreduced flat* MDB, while the lowest one is obtained by the *vertical* MDB. Comparing with unreduced MDBs, reduced MDBs significantly decrease the execution costs: from 54.4.% to 98.96%.

As we can see from figure 7, the same trend is found in MDBs of larger scale factors. From the *unreduced flat* MDB to the *vertical* MDB, the average execution costs decrease significantly: over 100 times (*cf.* the vertical axis on the left in figure 7). Moreover, the differences between the average execution costs in unreduced and reduced MDBs keep increasing as the data volume grows; i.e. from *SF1* to *SF4*, the gap has widened about 513 times (*cf.* the vertical axis on the right in figure 7).

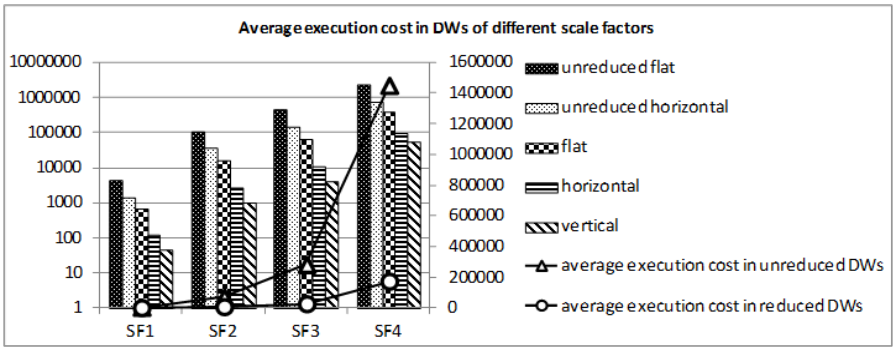


Fig. 7. Average execution costs of all queries in MDBs of different scale factors

In reduced MDBs, the decrease in execution costs is directly reflected in the gain in query runtime. Figure 8 shows the average runtime in MDBs of the scale factor 4 according to query type.

Discussion. All reduced MDBs allow significantly saving the query execution costs, regardless of the scale factor and the query type. More importantly, the results of our experimental assessments show the scalability of our proposal: the larger the

MDB is, the more significant the decrease in execution costs becomes after data reduction. The most efficient relational modeling is the *vertical* MDB. It groups measure instances and related attribute instances from one state together and implements them in one table. Consequently, data redundancy is reduced, while queries involving multiple dimensions are freed from joins in a *vertical* reduced MDB.

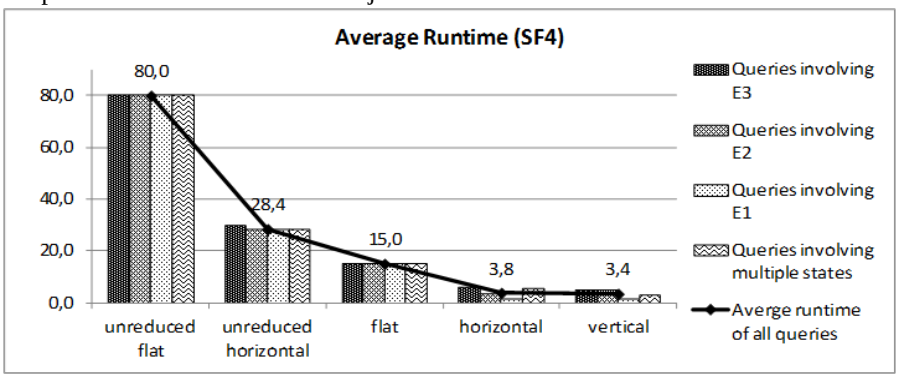


Fig. 8. Average runtime of queries involving different states in the largest MDBs

5 Conclusion

Our aim is to support effective and efficient decision-making by storing only data of high informative value over time in a MDB. In this paper, we outline a conceptual modeling solution allowing reducing both facts and dimensions in MDBs. A reduced MDB is modeled with multiple states. Each state is valid for a period of time.

Three relational modeling alternatives are proposed for reduced MDBs. The *flat* modeling integrates all measures and attributes from all states into one single flat table. The *horizontal* modeling converts each state into a fact table and a set of dimension tables. The *vertical* modeling gathers common measures and attributes shared by states into vertical tables. Different relational modeling alternatives (i) require different numbers of joins in analysis queries and (ii) bring in different degrees of information redundancy.

We carry out some experimental assessments to evaluate query execution efficiency in reduced and unreduced MDBs. The result shows the data reduction is a scalable solution: the larger the MDB is, the more significant the improvement in query execution efficiency becomes after the data reduction. During our experimental assessments, the improvement in terms of query execution costs ranges from 54.4.% to 98.96%. The most significant decrease in query execution costs is found in the *vertical* MDB, which makes it the most efficient relational modeling of reduced MDBs.

In the future, we intend to study the performance of our proposed relational modeling alternatives in other types of DBMS. As more and more NoSQL systems nowadays are adopted to deal with large amount of data, it would be necessary to study new data reduction strategies in the context of NoSQL. One of our ongoing work focuses on reducing data in *graph* databases and *triple store* (RDF) databases.

6 References

1. Atigui F, Ravat F, Song J, Teste O, Zurfluh G (2015) Facilitate Effective Decision-Making by Warehousing Reduced Data: Is It Feasible? Int J Decis Support Syst Technol 7:36–64.
2. Berkani N, Bellatreche L, Benatallah B (2016) A Value-Added Approach to Design BI Applications. In: Madria S, Hara T (eds) Big Data Anal. Knowl. Discov. pp 361–375
3. Darmont J, Bentayeb F, Boussaid O (2007) Benchmarking data warehouses. Int J Bus Intell Data Min 2:79–104. doi: 10.1504/IJBIDM.2007.012947
4. Garcia-Molina H, Labio W, Yang J (1998) Expiring data in a warehouse. In: 24rd Int. Conf. Very Large Data Bases. Morgan Kaufmann Publishers Inc., New York, pp 500–511
5. Golfarelli M, Maio D, Rizzi S (1998) Conceptual design of data warehouses from E/R schemes. In: Thirty-First Annu. Hawaii Int. Conf. Syst. Sci. IEEE Computer Society, Kohala Coast, HI, pp 334–343
6. Iftikhar N, Pedersen TB (2011) A rule-based tool for gradual granular data aggregation. In: Int. Workshop Data Warehous. OLAP. ACM Press, Glasgow, United Kingdom, pp 1–8
7. Nebot V, Berlanga R, Pérez JM, Aramburu MJ, Pedersen TB (2009) Multidimensional Integrated Ontologies: A Framework for Designing Semantic Data Warehouses. In: J. Data Semant. XIII. Springer Berlin Heidelberg, pp 1–36
8. Okun O, Priisalu H (2007) Unsupervised data reduction. Signal Process 87:2260–2267. doi: 10.1016/j.sigpro.2007.02.006
9. O'Neil P, O'Neil E, Chen X, Revilak S (2009) The Star Schema Benchmark and Augmented Fact Table Indexing. In: Perform. Eval. Benchmarking. Springer Berlin Heidelberg, Berlin, Heidelberg, pp 237–252
10. Skyt J, Jensen CS, Pederson TB (2002) Specification-based data reduction in dimensional data warehouses. IEEE Comput. Soc, p 278
11. Udo IJ, Afolabi B (2011) Hybrid Data Reduction Technique for Classification of Transaction Data. J Comput Sci Eng 6:12–16.