Lenz, Hans-Joachim; Günther, Oliver; Boyens, Claus

Working Paper

Statistical Databases

Papers, No. 2004,28

Provided in Cooperation with:
CASE - Center for Applied Statistics and Economics, Humboldt University Berlin

Suggested Citation: Lenz, Hans-Joachim; Günther, Oliver; Boyens, Claus (2004) : Statistical Databases, Papers, No. 2004,28, Humboldt-Universität zu Berlin, Center for Applied Statistics and Economics (CASE), Berlin

This Version is available at:
https://hdl.handle.net/10419/22201

Statistical Databases

Claus Boyens[1], Oliver Günther[1] and Hans-J. Lenz[2]

[1]Humboldt-Universität zu Berlin, Institut für Wirtschaftsinformatik,
Wirtschaftswissenschaftliche Fakultät, Spandauer Str. 1, 10178 Berlin, Germany

[2]Freie Universität Berlin, Institut für Produktion, Wirtschaftsinformatik und Operations Research, Fachbereich Wirtschaftswissenschaft, Garystr. 21, 14195 Berlin, Germany

Keywords: Relational database systems, statistical databases, data warehouses, metadata,

1 Introduction

Most data collected in statistics and science is still stored in simple *flat files*, usually data matrices with rows identified by a case identifier (case_id), columns corresponding to attributes (variables), and numerical data types for the elements of each matrix due to coding of all attributes involved. Each row (tuple) carries the (coded) values of the attributes, besides the case_id. Due to a suitable encoding that maps a natural domain to numerical ones, all matrix entries have a numeric data type. The scales of the attributes may of course be quite different.

A simple example is given by census data stored at statistical offices in files according to a schema like

```
census_questionnaire (case_id, age-group, gender,
       profession,...).
```

While science gains their data from experiments, statistical agencies collect their data still mostly off-line from surveys, reports or census. Industry and services get their data on-line from their business process management, i.e., from their logistical, production and administrative transactions. A typical example is sales data, which may be represented by a schema like

```
sales (transaction_id, customer_id, date,
       product_name, quantity, price, amount).
```

Such data is called *microdata*, since it is kept in its original form and is not divisible but atomic. In the business area such data is labeled as *on-line transaction data* because it is subject to frequent updates and is the basis for continuous business transactions. The use of a simple file system to store

microdata is rarely a good choice because of a lack of safety and integrity, and retrieval problems. Such data should rather be stored as tables of a relational database. A *database management system* (DBMS) asserts safety, integrity and retrieval flexibility. For instance, a query like "Find prices and amount of all sales since year 2001 where customer 007 with product 4711 is involved" can be simply stated in *structured query language* (SQL) as

```
SELECT price, amount FROM sales
WHERE year >= 2001
AND customer_id = 007
AND product_name = 4711;.
```

It is interesting to note that SQL provides for a set of query operators that is relationally complete. One may thus process any reasonable query as long as it does not involve “transitive closure”, i.e. a potentially infinite loop based on some logical inference (such as a part-of hierarchy).

Macrodata is derived from microdata by applying statistical functions, aggregation and grouping, and consequently has a larger granularity. For example, a business analyst might be interested in a *three-way table* (*data cube*) of total sales classified by `month` and `year`, `customer_id` and `product_name`. Such a retrieval can be achieved on sales by the command:

```
SELECT SUM(sales), date.month, date.year,
       customer_id, product_name
FROM sales Group BY date.month, date.year,
       customer_id, product_name;.
```

This type of activities is coined *on-line analytical operations* (OLAP), which expresses clearly its objective, i.e. a statistical analysis of data for planning, decision support, and controlling.

As we shall see later there does not exist a clear boundary between retrieval and statistical modeling. However, a statistical function like sum (or average) must be selected for a specific query, which does imply modeling. Consequently, there will not exist a closed set of operators on such multi-way tables. Moreover, there are two further problems involved. First of all, which data structure of such kind of data is efficient, and secondly, what kind of background information is needed, to assist the management and the interpretation of real data? This leads to discuss *metadata* as data about real data and functions. Modern database management systems encapsulate metadata in a *repository* (integrated metadata database).

In the following we are first concerned with some fundamentals of data management. Then we turn to the architecture of a statistical database or a data warehouse (DW) and some concepts related to it. We pay special attention to conceptual data structures and related operators involved, the summarizability problem, and hierarchical attributes. We discuss metadata, access methods and

"extraction, transformation and loading" (ETL). We close with metadata and extensible markup language (XML), and privacy.

2 Fundamentals of Data Management

We start our discussion with file systems, have a look at database systems (DBSs) useful to store transaction or microdata, and finally turn to DWs which host macrodata either in a real (materialized) or virtual form.

2.1 File systems

Data is classically stored in *files*. Files can be viewed as a conceptually related set R of records, which are represented by a given record type, see Wirth (1986), and an access mode (direct or sequential). If the records have a numeric type for each of its fields and the mode is sequential, then a data matrix can be stored in a sequential file. A collection of such files is called a file system (FS), if there exist logical relations between the files $f \in FS$, a set of constraints on FS and application software. Typical applications in statistics are simple surveys like price surveys, where in most cases only one file is needed. A more complex file system is compulsory if, for instance, stratified or panel sampling designs are considered, where various sampling periods, areas, objects and units (carriers of interest) are involved. Moreover, relational data mining, as described by Dzeroski and Lavrac (2001) and Wrobel (2001), is devoted to such data structures.

File systems are appropriate if only single user-access and weakly logically connected files with simple constraints are effective. Note that application programs must be specially tailored to execute queries, and to achieve data safety and security. This implies data dependence between the software and the files referenced, which reduces the program's flexibility with respect to structural changes of the data structure. These pitfalls can be overcome by DBSs.

2.2 Relational Database Systems (RDBS)

Multi-user access, complex data structures and logical restrictions ask for a *relational database system* (RDBS). It consists of a set T of relations (flat tables) together with a set S of corresponding schemas and a set C of constraints, a database management system and application software. A database schema describes the attributes (variables) of a specific table, its data types and roles. To avoid redundancy and anomalies during insert, delete or update transactions, those tables should be transformed into a "normal form", see Elmasri and Navathe (1999). As an example, we take a Census. When we look at the RDBS ´Census´

from a conceptual point of view, there are four table schemas involved: Census-questionnaire, household, dwelling, and employment. We shall consider only the first two in some detail, and select only some few attributes for the sake of brevity. The first schema is

```
census_questionnaire(case_id, age-group, gender,
       profession, ...).
```

Its first three attributes are numeric and the fourth one is of type 'string'. The attribute `case_id` acts as a primary key, i.e., the remaining attributes are functionally dependent on it. Because a key attribute uniquely identifies any tuple (record) of the corresponding table (set of tuples), there is one constraint among others saying that duplicates in a given table are not allowed. In order to mention just one further constraint, the domain of the identifier `case_id` may be restricted to the set of positive integers.

The next schema is

```
household (household_id, case-id, role,...).
```

The first two attributes have a numeric domain, while `role` is of type 'string' with the value set {"member", "owner"}. Of course, we have again the constraint that duplicates are not allowed, but we need at least one further restriction to ensure reference integrity, i.e., whenever there exist entries of people grouped together in a household, each of their corresponding records in `census_questionnaire` has to exist.

Last but not least, we reconsider our sales example from the introduction. The schema is

```
sales (transaction_id, customer_id, date, product_name,
      quantity, price, amount)
```

The primary key is `transaction_id`, which implies that only one product can be part of any transaction. Evidently, this scheme is not normalized, because `price` depends on `product_name` besides of `transaction_id`, and `amount = quantity * price`. The relation itself is of degree (number of attributes) seven. The six attributes `customer_id`, `date`, `...`, `amount` span a six-dimensional data space, where each tuple has six data elements, and is identified by its corresponding `transaction_id`. We represent four tuples in Table 1 to illustrate the difference between a schema and its corresponding relation (table). We use abbreviations in the header of the table `sales`.

Table 1: The relational table `sales` of degree 7 and cardinality 4

Transaction_id	customer_id	date	Product_name	quantity	price	amount
015	A	4 Jan 97	Tennis Shoes	200	95	19000.00
018	A	4 Jan 97	Tennis Balls	300	1.50	450.00
004	A	3 Jan 97	Tennis Nets	350	27	9450.00
009	C	3 Jan 97	Tennis Shoes	100	95	9500.00
…	…	…	…	…	...	...

The need of various users for different data can be satisfied by the concept of virtual relations (views), which can be created on top of an existing DBS.

Note that the term "table" used in a relational database to store such information is quite different from the tables statisticians use for the same purpose. Table 2 shows the representation of the same information in a different table structure that allows the natural computation of aggregates along rows and columns ("margin sums" etc.). Note that this table structure cannot be mapped directly into a relational database context due to the margins (Total or ALL), see Gray et al. (1996).

Table 2: `sales` data in the form of the three-way statistical table `total_sales`

3 Jan 1997	Tennis Shoes	Tennis Balls	Tennis Nets
Customer A	0	0	350
Customer B	0	0	0
Customer C	100	0	0
Total	**100**	**0**	**350**

4 Jan 1997	Tennis Shoes	Tennis Balls	Tennis Nets
Customer A	300	400	450
Customer B	1100	1100	800
Customer C	600	1600	350
Total	**2350**	**3400**	**1900**

Let us close this example with a discussion of the background information needed. We mentioned above metadata like schema names, attribute names, data types, roles (key vs. non key) of attributes, constraints etc. All this can be considered as technical metadata. Moreover, we need further metadata of a semantic and statistical type. Take for instance the attributes `quantity`, `price` and `amount`. What is their definition? As far as amount is concerned we have "`amount = quantity * price`". Furthermore, we need the corresponding *measurement units* which may be units, €/unit and €. As far as data collection at Statistical Offices is concerned, we may need information about the *periodicity* of data surveys like 'annual', 'quarterly' or 'monthly'. With respect to data analysis we may be interested in the *measurement scale*. While `product_name` has a nominal scale allowing only operations like 'equal' and 'not equal', the attributes `quantity`, `price` and `amount` have a metric scale allowing for all basic numerical operations. There exist further ambiguities. For example, the *generation mode* of the attribute `sales` may have the categories 'real', 'simulated' or 'forecasted'. There may exist further vagueness about `sales` of category 'real' unless its *update state* is set to 'final', and not to 'provisional'.

2.3 Data Warehouse Systems (DWS)

A *data warehouse system* (DWS) consists of a (replicated) micro database, a set of materialized or virtual multi-way tables (*data cubes*) needed to represent macro (pre-aggregated and grouped) data, a *data warehouse management system* (DWMS), and a repository, which stores all required technical, statistical and semantic metadata.

As an example of a data cube, we remind the reader of the three-way table presented above:

```
total_sales (date.month, date.year, customer_id,
             product_name, sum(sales)).
```

This table is represented in a relational form, where `date`, `customer_id` and `product_name` are concatenated as a primary key. These attributes are called *dimensions*. Evidently, the non-key attribute `sum(sales)` is fully dependent upon this key, i.e. given the values of `date.month`, `date.year`, `customer_id`, `product_name` there exist one and only one value of `sum(sales)` if missing values (null values) are excluded.

Views are useful again and can be provided by joining cubes or sub-cubes in combination with table projections to lower dimensions. It is worthwhile considering separately the attributes `sum(sales)`, `date` and `product_name`. The first attribute is sometimes called summary attribute and

is composed of the statistical `sum` applied to the attribute `sales`, see Shoshani (1997). This operation is feasible because the function `sum` and the attribute `sales` have an identical data type, i.e., a metric type. Moreover, the attribute `sales` is of attribute type `flow`, but not `stock`. While summarizing over flows (rates) is reasonable, such an operation over stocks like 'number of customers' is nonsense. Evidently, such and further integrity constraints must be effective for a DWS, in order to protect the naive user from nonsense queries. This is extremely important for data warehousing, because contrary to database queries, in DWS the application of statistical functions is an inherent part of any query.

Furthermore, there exists a specific problem related to `date`. This attribute can be decomposed into `month` and `year` but these components are functionally dependent, i.e., for a given month of a calendar year the year is fixed. We thus have (`month, year`) -> `year` as a functional dependency. Therefore only one dimension called `date` is used for the two attributes `month` and `year` in the data cube above. There may be further temporal levels like hour, day, month, quarter and year. Such hierarchical attributes are called taxonomies and need special attention, see Lehner et al. (1998). It is quite remarkable that all dimensions can be allocated to three principal groups: time, location and subject. This is called the 3D-principle, see Lenz (1994).

Let us have a further look at *taxonomie*s that are unbalanced and asymmetric. This may happen in case of a product or regional hierarchy. In our running example the subgroups tennis shoes and balls may be grouped together as product_group1, while tennis nets build-up product_group2, but are free of sub-grouping. Both groups 1 and 2 build the root group product_all. As subgroups exist only for shoes and balls, subgroups are no longer functionally dependent on `product_name`, but only weakly functionally dependent, see Lehner et al. (1998), Fig. 1. This implies that queries, which involve sub-grouping over products, are not feasible and must be refused. Further pitfalls of operations on a data-cube are given in Lenz and Shoshani (1997) and Lenz and Thalheim (2001).

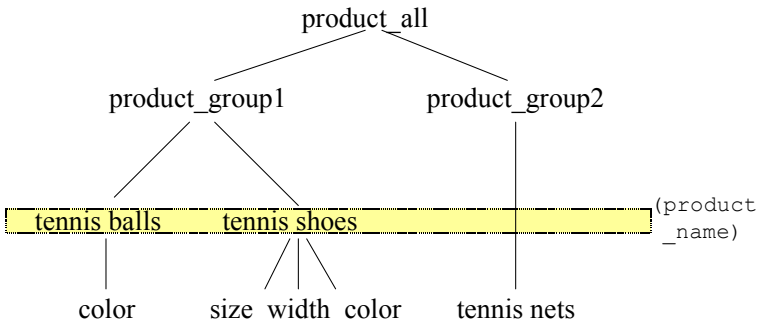

Fig. 1: The product taxonomy with a weak functional dependency

This discussion shows that real data without metadata is more or less useless especially for on-line analytical processing (OLAP). A repository with metadata has become a prerequisite of any DBS engineering and sound data analysis.

3 Architectures, Concepts and Operators

We first consider the architecture of micro or operational data used for *online transaction processing* (OLTP), and then illustrate the different architecture of macro or analytical data used for decision support and its relation to operational data, see OLAP. We note that the key features of a DBS for OLTP data are: transaction-oriented, measurement- or record-based, real time processing of inserts, deletes and updates of records. In contrary, a DWS for OLAP data is characterized by the features: subject-oriented, integrated and aggregated, calendar or fiscal period related, and non-volatile, see Inmon (1992).

3.1 Architecture of a database system for OLTP

The architecture of DBS can be represented by the quintuple (data sources, application server, DB server with a DBMS, application server, DB and repository); see also Fig. 2. As mentioned above, business processes act as data sources in commercial systems, while at statistical offices data is supplied by surveys, periodic reports or a census. Similarly, in science the data is generated by observations or measurements collected by field or simulation experiments. We represent the architecture in Fig.2.

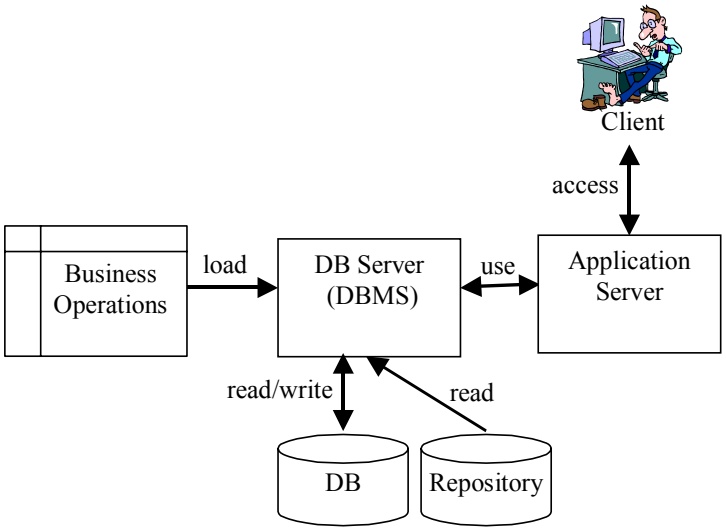

Fig. 2: Architecture of a DBS used to manage and query operational data

As an example from business we consider a company, which manages wages and salaries of its employees. The data is generated by bookkeeping, the DBMS administers the real and metadata, processes queries, and controls transactions. The application server is responsible for running the software for wage and salary computation, while the client is used as a presentation layer.

3.2 Architecture of a data warehouse

The main components of the architecture of any OLAP application are heterogeneous data sources S like internal or external databases or files, an OLAP server with DWMS, DW, Repository and Data Marts, and OLAP clients. The DWMS is responsible for load management, query management and warehouse management.

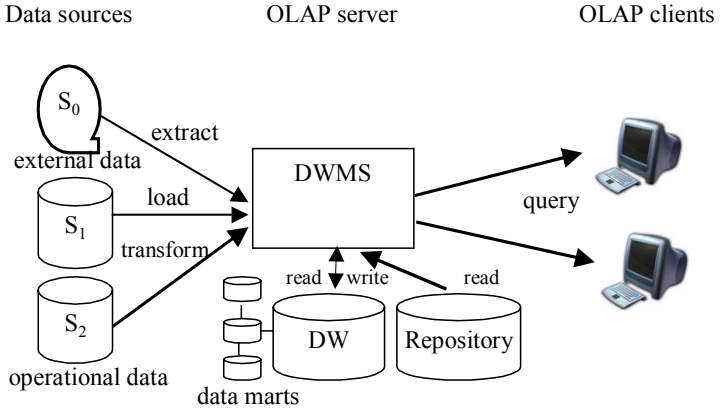

Fig. 3: DW Architecture

The DW (see Fig. 3) incorporates data replications, archived data and aggregated data stored as data cubes. The departmental view on the whole data is given by subsets of the data cube, called data marts.

As can be seen from Fig. 3, analytical processing is concerned with data from various data sources, i.e., external or internal (operational) data. These sources are integrated by ETL in data marts in an unified manner. The data marts can be viewed as collections of data cubes.

There exist two types of OLAP clients:

(i) stand-alone applications like spreadsheets with a DW interface, and

(ii) Web clients that use an Internet browser and often applets.

3.3 Concepts (ROLAP, MOLAP, HOLAP, cube operators)

As we have seen above, the schema of a data cube consists of a cube identifier (name), a list of identifying attributes called dimensions and a statistical function like min, max, count (frequency), sum, avg (arithmetic mean) applied to a summary attribute. Furthermore, the data types of the attributes and integrity constraints must be given. As an example we take from above the data cube "sales cross-classified by (month, year), customer and product":

```
total_sales(date.month, date.year, customer_id,
            product_name, sum(sales)).
```

Evidently, the dimensions span a three-dimensional space on which the statistical function `sum (sales)` is defined. The corresponding data types are date (mm.yyyy), integer, string and decimal.

3.3.1 Relational OLAP (ROLAP)

In the following we turn to the conceptual mapping of a data cube into a relational database schema. This approach is called *ROLAP* for Relational OLAP, see Raden (1996). There exist two schemas, star and snowflake schemas. As illustrated in Fig. 4, the star schema uses two different types of schemas, which refer to two types of corresponding tables:

1. a *fact table* with a primary key reference to each dimension and the facts which are composed of at least a statistical function and a summary attribute.
2. a *dimension table* for each dimension with a primary key and a level indicator for each entry of a hierarchical attribute.

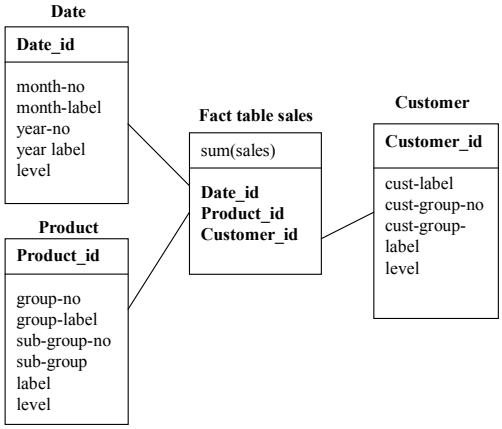

Fig. 4: Star schema of a three-dimensional data cube (one fact table, three dimension tables; the product hierarchy is assumed to have two levels)

The star schema models all kind of hierarchical attributes including parallel hierarchies, see Lehner et al. (1997). The schema is not normalized as becomes obvious, for example, from the dimension table `Date`. The attributes `month` and `year` are nested, which implies some redundancy. For small or medium-sized data volumes, such schemas have a sufficient performance because join operations are only necessary between the fact table and the related dimension tables.

In order to normalize tables by level attributes, the snowflake schema was introduced. Instead of modelling each dimension by one table, a table is created for each *level* of a hierarchical attribute. The schemas involved are related by identifiers, which play the role either of a primary or a foreign key. In Fig. 5 we display only the normalized dimension tables `Month` and `Year` and the fact table `Sales`. The identifiers are `month-no` in the fact table and dimension table `Month` and `year-no` in the dimension table `Year`.

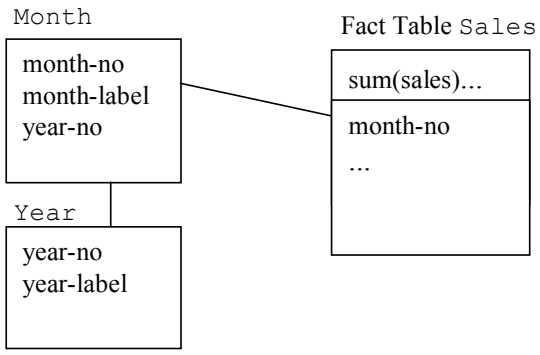

Fig. 5: Data cube `Sales` represented (fractionally) as a snowflake schema

It can be shown that the normalization is lossless by applying an inner join to the tables of a snowflake schema.

3.3.2 Other Storage Modes (MOLAP, HOLAP)

The above conceptual model of a star or snowflake schema may lead to the wrong conclusion that data cubes are exclusively represented by a relational data model approach. There exist further storage modes, which are in use.

The main advantage of ROLAP lies in the reliability, security and ease of loading of the DW based on *Relational DBMS* (RDBMS) technology. As was mentioned above, this is achieved due to the mapping of facts into a normalized relation and dimension into a mostly non-normalized relation of a relational database. As the set of statistical functions in SQL is too restrictive, some of the functionality of OLAP must be added to the application server. An example is to find the *top-ten* among all products sold in a given period.

Multi-dimensional OLAP (MOLAP) makes use of specially tailored data structures like arrays and associated dimension lists or bitmaps. The operational data is extracted and stored as aggregates in those structures. The performance is acceptable for up to medium-sized data sets (< 1Gbyte). There exists a multi-dimensional query language called MDX (Multidimensional Expressions), see Microsoft (1998). "XML for Analysis" defines a standardized programming interface for an OLAP server, see `http://www.xmla.org`. An OLAP client encodes a query of a data cube and inserts it into a XML document, which specifies the method "execute" and the accompanying parameters according to the "Simple Object Access Protocol" (SOAP). This document is transmitted over the Internet based on the "Hypertext Transfer Protocol" (HTTP). After decoding the OLAP server executes the query, and sends the data back in a XML document to the client according to SOAP. For further details see Messerschmidt and Schweinsberg (2003). MOLAP has the disadvantage of "miss hits" if a data cube cannot be stored fully in-core and an access to a second storage device is necessary. Moreover, array compression or sparse array handling is needed because mostly the data cube or, equivalently, the arrays are sparse.

Hybrid OLAP (HOLAP) tries to combine the advantages of relational and multi-dimensional database technology. The relational model is used to store replicated and low-level aggregates, while the multi-dimensional model is responsible for high-level aggregates.

3.3.3 Data Cube Operators

Data cubes are used for analytical purposes and not for (simple) transaction processing. Therefore there does not exist a clear boundary between data extraction or retrieval and data analysis. Therefore there does not exist a minimal, closed and complete set of OLAP operators. The mostly built-in operators on data cubes in commercial DWs are the following, see Shoshani (1997) and Jarke et al. (2000).

Slicing $\sigma_c(T)$ is to select data from a cube T according to a fixed condition c. This operation is called in Statistics conditioning if only frequencies (counts) applied to multi-way tables are considered. For example, we can retrieve data from `total_sales` according to $\sigma_{\texttt{product\_id, customer\_id, month, year==97}}(\texttt{total\_sales})$.

Dicing $\pi_c(T)$ is table projection on T by selecting a sub-cube T' of some lower dimension c than the original cube T has. This operation is equivalent to marginalization in Statistics, i.e. projection of a data space into a lower dimension. For instance $\pi_{\texttt{date, customer\_no}}(\texttt{total\_sales})$ retrieves a sub-cube of total sales cross-classified by `date` and `customer`.

Table aggregation (roll-up) and *disaggregation* (drill-down) are operations on data cubes if at least on dimension is hierarchical. For example $\rho_{\texttt{year, customer\_no,}}$

$_{product\_no}$(total_sales) is a query for less fine-grained data, i.e. for years and summarizing over all months per year. This specific operation is called temporal aggregation. We observe that such an operation is not allowable if a type conflict happens with respect to the summary attribute. This is the case if the attribute 'sales' is substituted by 'no of employees', see Lenz and Shoshani (1997).

Drill-across $\delta_{level,\ node,\ attribute}(T)$ is a navigation on the same level through the various subtrees of a hierarchical attribute starting at a given node. For example, retrieving products from level 1 (product_group) with start at product_group1 (shoes and balls) of the taxonomy "Product" delivers data about tennis nets.

In order to compute ratios, products etc. of data cubes the *join operator* $\gamma_{\otimes}(T_1,T_2)$ is needed. For instance, as sales = turnover * price we have sales:=γ_{*}(turnover, price).

We note that there exist further operators like pivot (rotation of a cube), see Jarke et al. (2000), or cube, which was introduced by Gray et al. (1996). It delivers the margins *ALL* for any subset of dimensions.

3.4 Summarizability and Normal Forms

The main objective of *summarizability* is to guarantee correct results of the cube operation roll-up and the utilization of statistical (aggregation) functions like min, max, avg, sum and count under all circumstances, see Lenz and Shoshani (1997). The corresponding integrity constraints are non-overlapping levels of dimensions, completeness and type compatibility. The first condition assures that each node of a taxonomy has at most one preceding node except for the root node. The second one ascertains that any node on a low level granularity corresponds to at least one node of a higher granularity. Type compatibility guarantees that the application of any statistical function on a summary attribute is sound. In a preceding section we mentioned the unfeasibility of aggregation of stocks over time. Another example is the misuse of the sum operator applied to code numbers of professions.

As Lehner et al. (1998) pointed out, the integrity constraint of completeness may turn out to be too restrictive. This happens if there exist structural missing values (null values) in taxonomies. For example, the German state Bavaria is divided into regions called "Kreise". Berlin is a city as well as an autonomous German state. It is not divided into regions, but into suburbs called "Bezirke". In such cases a context sensitive summarizability constraint is appropriate. The authors consequently proposed three multi-dimensional normal forms for fact tables. Lechtenbörger and Vossen (2001) improved the design of these normal forms.

3.5 Comparison of Terminologies

To sum up this chapter, Tables 3 and 4 compare the terminology of statistical databases and OLAP, see Shoshani (1997).

Table 3: Comparison of Concepts

Statistical Databases	**OLAP**
Categorical attribute	Dimension
Structural attribute	Dimension hierarchy
Category	Dimension value
Summary attribute	Fact
Statistical object, multidimensional table	Data cube
Cross product	Multidimensionality

Table 4: Comparison of Operators

Statistical Databases	**OLAP**
table projection	Dice
table selection	Slice
table aggregation	Roll-up
table disaggregation	Drill-down
table join	term missing
term missing	Drill across
viewing	pivoting

4 Access Methods

4.1 Views (Virtual Tables)

Statistical databases are often accessed by different users with different intentions and different access rights. As already indicated in section 2.2, these different requirements can be accounted for by using *views*. These views are derived *virtual tables*, which are computed from the (actually stored) base tables, see Elmasri and Navathe (1999). There are two main purposes for the use of views.

1. It makes the use of the DBS or DW more convenient for the user by providing only customized parts of the whole data cube.
2. It enforces security constraints by restricting operations on the base tables and by granting users access to their specific views only.

The following SQL statement creates a view for the manager of the product "Tennis Nets" from our example in Table 1. It only permits to look up the revenues for "Tennis Nets" while for all other products, viewing the sales and modifying the corresponding base tables is not possible.

```
CREATE VIEW tennis_nets_manager AS
   SELECT date.month, date.year, customer_id,
   sum(sales)
   FROM total_sales WHERE product_name = "Tennis
   Nets";
```

Views can never contain information that is not present in the base tables as the DBS translates all view queries into equivalent queries that refer only to base tables.

Base tables of a DW may contain millions of tuples. Scanning these tables can be time-consuming and may slow down the interaction between the decision support system and the user significantly. One strategy to speed up the access to aggregated data is to pre-compute a range of probable queries and to store the results in *materialized* views, see Gupta et al. (1997). The access to these materialized views is then much faster than computing data on demand. Yet there are drawbacks to this strategy. The pre-computed data need space, the prediction of the users' queries is difficult, and each change in the base table requires an update of the materialized view also. This is known as the *view maintenance problem*, see Huyn (1997).

4.2 Tree-based Indexing

The tables of a DW can physically be accessed either by a sequential scan or by random access. With today's hard disks, a sequential scan is 10 to 20 times faster than random access, see Jürgens (2002). That means if more than approximately 5% to 10% of the data has to be accessed in a table, it is faster to scan the entire table than addressing specific tuples via random access. In order to avoid full table scans, the number of tuples involved in the result computation has to be reduced. This can be achieved via *index structures,* which permit a fast look-up of specific tuples.

The best-known index structure for one-dimensional data (i.e. data with just one key such as `product_id`) is the *B-tree*, see Bayer and McCreight (1972), Comer (1979). Pointers to the data items are stored in the leaf nodes of a balanced tree. The B-tree is a very general and flexible index structure, yet in some specific cases it may be outperformed by different kinds of hashing, see Gaede and Günther (1998).

The *universal B-tree* (*UB-tree*, see Bayer, 1997) is an extension of the B-tree for indexing multidimensional data such as `total_sales (date.month, date.year, customer_id, product_name, sum(sales))`. The approach partitions the multidimensional data space into squares each of which is captured by a space-filling Z-curve, see Fig. 6. For each record, the Z-address of the square, which contains the key values is computed. These Z-addresses are one-dimensional and serve as the new primary keys for the records, which can then be indexed with a standard B-tree.

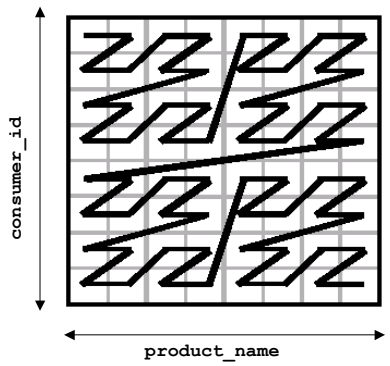

Fig. 6: The UB-Tree: Partition and capture of multidimensional space with the Z-curve

Another approach for indexing multidimensional data is the *R-tree*, see Guttman (1984). It uses rectangles to represent multidimensional intervals. The leaf rectangles correspond to entries in the database. The parent nodes contain all child nodes and the minimal bounding rectangle. The root rectangle covers the entire

query space. An example of how to store sales indexes in an R-tree when `product_name` and `customer_id` build the concatenated primary key is shown in Fig. 7. The minimal bounding rectangle of the dashed-line rectangles A, B and C constitutes the entire search space.

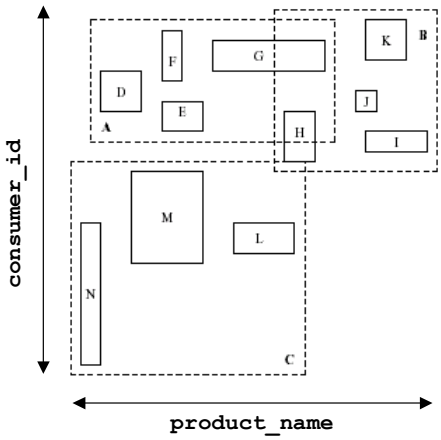

Fig. 7: An exemplary R-Tree

Refinements are the R^+-tree of Sellis et al. (1985), the R*-tree of Beckmann et al. (1990) and a slightly improved version called R^*_a-tree of Jürgens (2002).

4.3 Bitmap Index Structures

An important alternative to tree index structures is *bitmap indexing*. For each value of an attribute, a bitmap vector indicates whether or not it is assumed in the records of the table, see Chan and Ioannidis (1998), O'Neil and Quass (1997), Wu and Buchmann (1998). Table one shows a bitmap index for the attribute `product_name` corresponding to the example presented in Table 5.

Table 5: Bitmap index for the attribute `product_name`

transaction_id	Tennis Balls	Tennis Nets	Tennis Shoes
015	0	0	1
018	1	0	0
004	0	1	0
009	0	0	1

The bitmap vector for the attribute value "Tennis Balls" is $(0, 1, 0, 0)^T$. Such a set of bitmap vectors is created for all dimensions. In our `total_sales` example,

bitmap indexes have to be created further for `(date.month, date.year)` and `customer_id`.

The size of the bitmap index depends on the number of tuples and on the cardinality of the attribute domain. As the required operations on bitmaps are simple they are very fast. Thus loading blocks from disc and performing the basic Boolean operations is efficient, especially if the number of dimension is high, see Jürgens (2002). As bitmaps are often sparse, they are well suited for compression techniques. This is the reason why many commercial DBSs are implemented using bitmaps. However, standard bitmaps indexes become space consuming for high attribute's domain cardinality, and they are not very efficient for (low dimensional) range queries, which are typical for DW systems.

Several approaches have been proposed to overcome these drawbacks like the multi-component equality encoded bitmap index, see Chan and Ioannidis (1998). The basic idea is to compress bitmap indexes by encoding all values into a smaller number system by applying modular multiplication. This significantly reduces the space requirements for attributes of high cardinality.

To summarize, bitmaps are more suited for high-dimensional queries with low attribute cardinality. Tree index structures are better for low-dimensional range queries with attributes of high cardinality.

5 Extraction, Transformation and Loading (ETL)

ETL is a shorthand notation for a workflow of the initial popularization or a follow-up update of a DW, a data mart or an OLAP application. In the first step data must be extracted from the various data sources and temporarily stored in a so-called staging area of a DWS. Transformation means to modify data, schema and data quality according to requirement specifications of the DWS. Loading is the integration of replicated and aggregated data in the DW. As the data volume may be huge, incremental loading within pre-selected time slots by means of a bulk loader is appropriate.

5.1.1 Extraction

Extraction can be triggered by events linked to time and state of a DBS in operation or can be executed under human control. Mostly extraction is deferred according to an extraction schedule supplied by monitoring of the DWS. However, changes of data in the source system are tracked in real time, if the actuality of data is mandatory for some decision makers, see Kimball (1996).

As the data sources are generally heterogeneous, the efforts to wrap single data sources can be enormous. Therefore software companies defined standard

interfaces, which are supported by almost all DBMS and ETL tools. For example, the OLE database provider for ODBC, see Microsoft (1998, 2003), Oracle (2003) and IBM (2003).

5.1.2 Transformation

Transformations are needed to resolve conflicts of schema and data integration and to improve data quality, see **Chapter \ref{III.9}**.

We first turn to the first type of conflicts. Spaccapietra et al. (1992) consider four classes of conflicts of schema integration, which are to be resolved in each case.

(i) *Semantic conflicts* exist, if two source schemas refer to the same object, but the corresponding set of attributes is not identical, i.e. the class extensions are different. As an example take two customer files. One record structure includes the attribute name `gender`, while it is missing in the other one.

(ii) A second kind of conflict of integration happens if *synonyms*, *homonyms*, different *data types*, *domains* or *measurements units* exist. For instance, think of the synonym part / article, a homonym like water / money pool, string /date as a domain, and Euro / USD. The ambiguity of our natural language becomes clear when one thinks of the meaning of "name" – family name, nickname, former family name, artist name, friar name,...

(iii) *Schema heterogeneity conflicts* appear if the source schemas differ from the target schema of the DW. For example, sales and departments can be modeled as two relations `Sales` and `Department` of a relational data model or as a nested relation `Department \ Sales` as part of an object oriented model. Another kind of conflict corresponds to the mapping of local source keys to global surrogates, see Bauer and Günzel (2001). This problem gets tightened if entity identification is necessary in order decide whether a pair of records from two data sources refer to same entity or not. Fellegi and Sunter (1969) were the first to solve this problem by the record-linkage technique, which is now considered as a special classification method, see Neiling (2003).

(iv) *Structural conflicts* are present if the representation of an object is different in two schemas. There may exist only one customer schema with the attribute `gender` in order to discriminate between "males" and "females". Alternatively, there may be two schemas in use, one linked to "females", the other one to "males".

The second type of conflicts, i.e., conflicts of data integration, happens, if false or differently represented data are to be integrated. False data are generated by erroneous or obsolete entries. Differences in representation are caused by non-

identical coding like male/female versus 0/1 or by different sizes of rounding-off errors.

6 Metadata and XML

McCarthy (1982) described *metadata* as data about data. However, the technical progress of OLTP and OLAP DBSs, workflow techniques and information dissemination has made it necessary, to use a more general definition of metadata.

Metadata is now interpreted as any kind of integrated data used for the design, implementation and usage of an information system. This implies that metadata not only describes real data, but functions or methods, data suppliers or sources and data receivers or sinks, too. It does not only give background information about the technology of a DBS or DWS, but about its semantic, structure, statistics and functionality. Especially, the semantic metadata enable the common user to retrieve definitions of an attribute, to select and filter values of meta attributes, and to navigate through taxonomies.

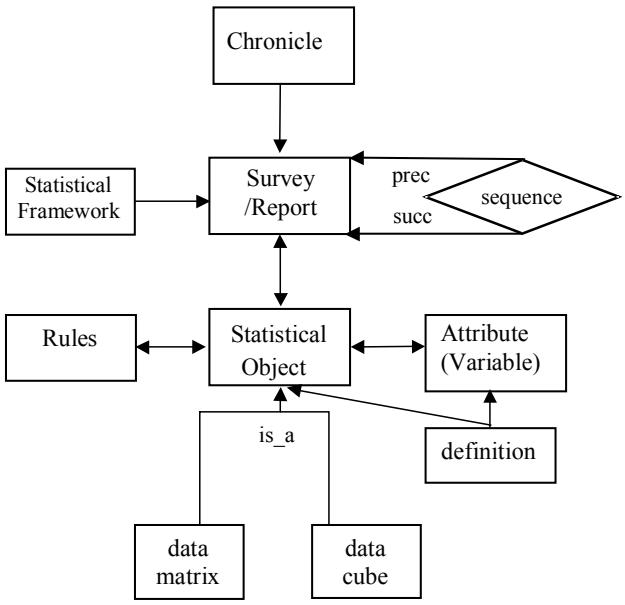

Fig. 8: Statistical view of metadata

In Fig. 8 we present a view of a conceptually designed metadata. Its core is given by a statistical object, which is either a specialisation of a data matrix or a data

cube. It is uniquely described by a definition, and is related in a many to many way to validation and processing rules, surveys or reports and attributes. As we present only a view, no further refinement is given with respect to attributes like roles (measure, key, property), scales (nominal, ordinal, cardinal), ontologies or even domains (natural, coded) etc. Each statistical object is linked to at least one survey or report. Surveys or reports can be sequenced according to preceding or succeeding ones, are related to a statistical framework ("statistical documentation") giving details about sampling scheme and frame, population and statistical methods, and are associated to a chronicle as calendar of events. Furthermore references to the specific literature and law are included. The corresponding substructure is not displayed in Fig. 8. For further information about the metadata structure from the user's point of view, see Lenz (1994).

As metadata is stored and can be retrieved similar to real data, it is captured in a *repository* and is managed by a metadata manager. A repository can be accessed by users, administrators and software engineers according to their privileges and read-write rights.

Such repositories are offered from various vendors. Microsoft (2001) labelled its repository as "metadata services", and it is integrated in its SQL server. Alliances were founded to harmonize the metadata models and to standardize the exchange formats. Leading examples are the Open Information Model of the Metadata Coalition (MDC), see `http://www.mdcinfo.com`, and the Common Warehouse Metamodel (CWM), which was developed by the Object Management Group (OMG), see `http://www.omg.org`. Since the year 2000 both groups were fused and try to merge their models. Due to the increasing importance of XML and XML databases, import and export format of metadata based on XML is becoming an industrial standard. This happened to OLAP client-server architectures, see "XML for Analysis" as referred in section 3.3.2.

7 Privacy and Security

7.1 Preventing Disclosure of Confidential Information

The statistical databases that are built by government agencies and non-profit organizations often contain confidential information such as income, credit ratings, type of disease or test scores of individuals. In corporate DWs, some strategic figures that are not related to individuals like sales for recently launched products may also be confidential. Whenever sensitive data is exchanged, it must be transmitted over a secure channel like the *Secure Socket Layer* (SSL), see Netscape (1996) in order to prevent unauthorized use of the system. For the

purposes of this chapter, we assume that adequate measures for security and access control are in place, see Stallings (1999).

However, even if the information in the statistical database safely reaches the correct addressee, the system has to ensure that the released information does not compromise the privacy of individuals or other confidential information. Privacy breaches do not only occur as obvious disclosures of individual values in single queries. Often, the combination of multiple non-confidential query results may allow for the *inference* of new confidential facts that were formerly unknown.

We give an example. From Table 1, we take the total sales for "Tennis Shoes" (28,500), "Tennis Balls" (450), "Tennis Nets" (9450) and a fourth, new product ("Tennis Socks", 500). We assume that sum queries for groups of products are allowed but that single, product-specific sales values are confidential. After querying the sum for balls and shoes (28,950) and for balls and socks (950), the user can infer an interval of [28,000; 28,950] for the sales of shoes, as sales cannot be negative. The length of the interval, which is the maximum error of the user's estimation of the confidential shoe sales is only 3.3% of the actual value. This particular case of disclosure is called *interval inference*, see Li et al. (2002). Other types of inference include *exact inference* (concluding the exact value of 28,500 for shoes sales) and *statistical inference* (inferring estimates like mean

$$\bar{x}_{\text{Tennis Shoes}} = 30{,}000 \text{ and standard deviation } s_{\text{Tennis Shoes}} = 5{,}000).$$

If a researcher is granted ad-hoc access to a statistical database, there are basically two different approaches to protect information that is private and confidential from being revealed by a malevolent snooper, see Adam and Wortmann (1989), Agrawal and Srikant (2000), Fig. 9. In the first approach, the kind and number of queries that a researcher poses to the statistical database is restricted (*query restriction)*. In the second approach, the entire database is subject to a manipulation that protects single values but preserves the statistical properties which are of interest to the user. Then the perturbed database can be accessed by a researcher without restrictions (*data perturbation*). In the following, we give an overview of disclosure protection techniques of this kind.

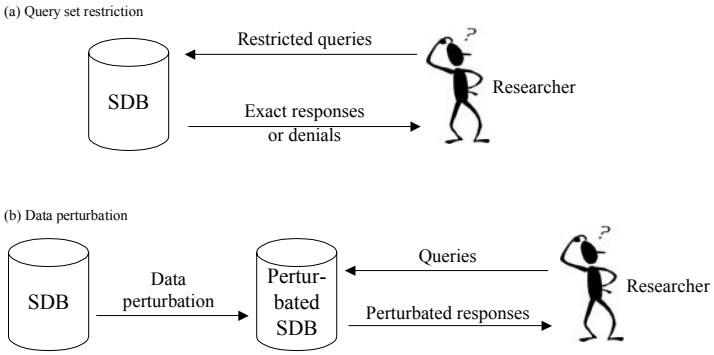

Fig. 9: (a) Query set restriction and (b) data perturbation
Source: Adam and Wortmann (1989)

7.2 Query set restriction

With this approach a query is either denied or responded with an exact answer as the upper sketch in Fig. 9 indicates.

Query set size control, Fellegi (1972) works by setting lower and upper bounds for the size of the query answer set based on the properties of the database and on the preferences fixed by the database administrator. If the number of returned records did not lie within these bounds, the information request would have to be rejected and the query answer is denied. As queries that are issued sequentially by one user often have a large numbers of entities in common, an improvement is the restriction of these entities to a maximum number, see Dobkin et al. (1979). Although popular, this method is not robust enough as a stand-alone solution, see Denning (1982).

Auditing involves keeping up-to-date logs of all queries made by each user and constantly checking for possible disclosures whenever a new query is issued. One major drawback of this method is that it requires huge amounts of storage and CPU time to keep these logs updated. A well-known implementation of such an audit system is *Audit Expert* by Chin and Özsoyoglu (1982). It uses binary matrices, see bitmap indexes in section 4.3, to indicate whether or not a record was involved in a query.

Cell suppression, see Cox (1980) is an important method for categorical databases when information is published in tabular form. Especially Census Bureaus often make use of tabular data and publish counts of individuals based on different categories. One of the main privacy objectives is to avoid answers of small size. For example, if a snooper knows somebody's residence, age and employer, he can issue a query for `(ZIP=10178, Age= 57, Employer= 'ABC')`. If the

answer is one entity, the snooper could go on and query for `(ZIP= 10178, Age= 57, Employer= 'ABC', Diagnosis= 'Depression')`. If the answer is one again, the database is compromised and the person with the diagnosis identified. The cells should have to be suppressed. A common criterion to decide whether or not to suppress a cell is the *N-k rule* where a cell is suppressed if the top N respondents contribute at least k% of the cell total. N and k are parameters that are fixed by the database administrator, i.e. the Census Bureau. In the exemplary case of N= 2 and k= 10%, a cell which indicates aggregated income ($10M) of 100 individuals would have to be suppressed if the top two earners' aggregate income exceeded $1M.

7.3 Data Perturbation

In the query restriction approach, either exact data is delivered from the original database or the query is denied. As depicted in the lower sketch of Fig. 9, an alternative is to perturb the original values such that confidential, individual data become useless for a snooper while the statistical properties of the attribute are preserved. The manipulated data is stored in a second database and is then freely accessible for the users.

If in Table 1, we permute the sales of tennis balls, tennis nets and tennis shoes, individual sales data is not correct anymore. But the arithmetic average and the standard deviation of the attribute `sales` stay the same. This procedure is called *data swapping* , see Denning (1982).

Noise addition for numerical attributes, see Traub et al. (1984), means adding a disturbing term to each value: $Y_k = X_k + e_k$, where X_k is the original value and e_k adheres to a given probability distribution with mean zero. As for every value X_k value, the perturbation e_k is fixed, conducting multiple queries does not refine the snooper's search for confidential single values.

A hybrid approach are *random-sample queries,* Denning (1982), where a sample is drawn from the query set in such a way that each entity of the complete set is included in the sample with probability P. If, for example, the sample of a COUNT query has n entities, then the size of the not perturbed query set can be estimated as n/P. If P is large, there should be a set-size restriction to avoid small query sets where all entities are included.

7.4 Disclosure Risk vs. Data Utility

All methods presented in the preceding sections aim at lowering the disclosure risk for data that is private and confidential. But at the same time, each of these methods reduces, in some way, the utility of the data for the legitimate data user. Duncan and Keller-McNulty (2001) present a formal framework to measure this

trade-off between disclosure risk and data utility, the *Risk-Utility* (R-U) map. There are numberless measures for disclosure risk, see Domingo-Ferrer et al. (2002) for an excellent overview. We already gave an intuitive measure for interval inference. The sales for tennis shoes were predicted with an error of only 3.3%, see section 7.1.

However, it is far more difficult to measure data utility because it strongly depends on the varying preferences of the data user. Especially for this reason, classifying statistical disclosure control methods as presented here on an absolute scale is almost an impossible task.

References

Adam, N. and Wortmann, J. (1989). Security-Control Methods for Statistical Databases: A Comparative Study. *ACM Computing Surveys*, 21(4): 515-556.

Agrawal, R. and Srikant, R.(2000). Privacy-preserving Data Mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp.439-450.

Bauer, A. and Günzel, H. (eds) (2001). *Data Warehouse Systeme*. dpunkt.verlag, Heidelberg.

Bayer, R. (1997). The universal B-Tree for multidimensional Indexing: General Concepts. In *World-Wide Computing and Its Applications '97 (WWCA '97)*. Tsukuba, Japan, 10-11, Lecture Notes on Computer Science, Springer Verlag.

Bayer, R. and McCreight, E. (1972). Organization and maintenance of large ordered indexes. *Acta Informatica*, 1(3): 173-189.

Beckmann, N., Kriegel, H.-P., Schneider, R. and Seeger, B. (1990). The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 323-331, New York.

Chan, C.-Y. and Ionanidis, Y. E. (1998). Bitmap index design and evaluation. In *Proceedings of the ACM SIGMOD International Conference on Management of Data,* pp 355-366.

Chin, F. Y. and Özsoyoglu, G. (1982). Auditing and inference control in statistical databases. *IEEE Transactions on Software Engineering*, 8(6): 574-582.

Comer, D. (1979). The ubiquitous B-Tree. *ACM Computing Surveys*, 11(2): 121-138.

Cox, L. H. (1980). Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75: 377-385.

Denning, D. E. (1982). *Cryptography and Data Security*. Addison-Wesley.

Dobkin, D., Jones, A. K. and Lipton, R. J. (1979). Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4(1): 97-106.

Domingo-Ferrer, J., Oganian, A. and Torra, V. (2002). Information-Theoretic Disclosure Risk Measures in Statistical Disclosure Control of Tabular Data. In *Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM '02)*.

Duncan, G. and Keller-McNulty, S. (2001). Disclosure risk vs. data utility: The R-U confidentiality map. *Technical Report. Statistical Sciences Group*. Los Alamos National Laboratory.

Dzeroski, S. and Lavrac, N. (eds) (2001). *Relational Data Mining*, Springer Verlag, Heidelberg.

Elmasri, R. and Navathe, S. B. (1999). *Fundamentals of Database Systems*, 3rd Edition, Addison Wesley.

Fellegi, I. P. (1972). On the question of statistical confidentiality. *Journal of the American Statistical Association*, 67(337): 7-18.

Fellegi, I. P. and Sunter, A.B. (1969). A Theory of Record Linkage. *Journal of the American Statistical Association*, 40, 1183-1210.

Gaede, V. and Günther, O. (1998). Multidimensional Access Methods. *ACM Computing Surveys*, 30(2): 170-231.

Gray, J., Bosworth, A., Layman, A. and Pirahesh, H. (1996). Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Total. In *Proceedings of the 12th International Conference on Data Engineering (ICDE'96)*, New Orleans, USA, IEEE Computer Society, pp.29-53.

Gupta, H., Harinarayan, V., Rajaraman, A. J. and Ullman, D. (1997). Index selection for OLAP. In *Proceedings of the Thirteenth International Conference on Data Engineering (ICDE '97)*, Birmingham U.K., IEEE Computer Society.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 47-57.

Huyn, N. (1997). Multiple-view self-maintenance in data warehousing environments. In *Proceedings of the 23rd Conference on Very Large Databases (VLDB)*, pp. 26-35.

IBM (2003). DB2 OLAP Server.
[`http://www-3.ibm.com/software/data/db2/db2olap/library.html`]

Inmon, W. H. (1992). *Building the Data Warehouse*, Wiley, New York.

Jarke, M. Lenzerini, M. Vassiliou, Y. and Vassiliadis, P. (2003). *Fundamentals of Data Warehouses*, 2nd ed., Springer, Berlin et al.

Jürgens, M. (2002). *Index Structures for Data Warehouses.* Springer Lecture Notes in Computer Science, Berlin/Heidelberg/New York.

Kimball, R. (1996). *The Data Warehouse Toolkit*. Wiley, New York.

Lechtenbörger, J. and Vossen, G. (2001). Quality-oriented Data Warehouse Schema Design. In *information technology*, vol. 45, 190-195.

Lehner, W., Albrecht, J. and Wedekind, H. (1998). Normal Forms for Multidimensional Databases. In *Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM'98)*, Capri, Italia, 63-72.

Lenz, H.-J. (1994). A rigorous treatment of Microdata, Macrodata and Metadata. In Dutter, R. (ed.), *Proceedings in Computational Statistics*, Physica, Heidelberg.

Lenz, H.-J. and Shoshani, A. (1997). Summarizability in OLAP and Statistical Databases, In *Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM '97)*, Washington, USA.

Lenz, H.-J. and Thalheim, B. (2001). OLAP Databases and Aggregation Functions. In *Proceedings of the 14th International Conference on Scientific and Statistical Data Management (SSDBM '01)*, Washington, USA.

Li, Y., Wang, L., Wang, X. and Jajodia, S. (2002). Auditing interval-based inference. In *Proceedings of the 14th Conference on Advanced Information Systems Engineering (CAiSE'02)*, Toronto, Canada.

McCarthy, J. (1982). Metadata Management for Large Statistical Databases, In *Proceedings of the 8th International Conference on Very Large Data Bases*, Mexico City, Mexico.

Messerschmidt, H. and Schweinsberg, K. (2003). *OLAP mit dem SQL-Server*, dpunkt. verlag, Heidelberg.

Microsoft (1998). OLE DB for OLAP Programmer's Reference.

Microsoft (2003). Microsoft SQL Server: Data Transformation Services (DTS). [http://www.microsoft.com/sql/evaluation/features/datatran.asp]

O'Neil, P. and Quass, D. (1997). Improved query performance with variant indexes. *SIGMOD record*, 26(2): 38-49.

Oracle (2003). Oracle Warehouse Builder - Product Information. [http://otn.oracle.com/products/warehouse/index.html]

Neiling, M. (2003). *Identifizierung von Realwelt-Objekten in multiplen Datenbanken*. PhD dissertation, Univ. of Cottbus, Germany.

Netscape (1996). Secure Socket Layer 3.0 Specification [http://wp.netscape.com/eng/ssl3/]

Raden, N. (1996). Star Schema 101. Archer Decision Sciences, Santa Barbara, [http://members.aol.com/nraden/str101_e.htm] (2000.12.12)

Shoshani, A. (1997). OLAP and Statistical Databases: Similarities and Differences. In *Proceedings of the 16th ACM SIGACT-SIGMOD-SIGART Symposium on Principles and Database Systems (PODS'97)*, Tucson, USA, 185-196.

Spaccapietra, S., Parent, C. and Dupont, Y. (1992). Model independent assertions for integration of heterogeneous schemas. *VLDB Journal*, 1(1), 81-126.

Stallings, W. (1999). *Cryptography and Network Security, Principles and Practice*. Addison Wesley.

Traub, J. F., Yemini, Y. and Wozniakowski H. (1984). The statistical security of a statistical database. *ACM Transactions on Database Systems,* 9(4): 672-679.

Wirth, N. (1986). *Algorithms and data structures*, Englewood Cliffs, Prentice Hall.

Wrobel, S. (2001). Inductive Logic Programming for Knowledge Discovery in Databases. In Dzeroski, S. and Lavrac, N. (eds), *Ralational Data Mining*, Springer Verlag, Heidelberg.

Wu, M.-C. and Buchmann, A. P. (1998). Encoded bitmap indexing for data warehouses. In *Proceedings of the 14th International Conference on Data Engineering (ICDE)*, 220-230.