INFORMATION QUALITY: HOW GOOD ARE OFF-THE-SHELF DBMS?

Felix Naumann
Humboldt-Universität zu Berlin, Germany
naumann@informatik.hu-berlin.de

Mary Roth
IBM Silicon Valley Lab, USA
torkroth@us.ibm.com

Abstract: Commercial database management systems (DBMS) have come a long way with respect to efficiency and more recently, with respect to quality and user friendliness. Not only do they provide an efficient means to store large amounts of data and intuitive query languages to access the data; popular DBMS also provide a whole suite of tools to assess, store, manage, clean, and retrieve data in a user-friendly way. Some of these feature address database experts, others are targeted at end-users with little or even no database knowledge. The recent developments in the field of autonomic computing drive the ease-of-use even further.
In this report we study how well a typical DBMS meets the goal of providing a high-quality data storage and retrieval facility. To this end, we draw on an established set of information quality criteria and assess how well an exemplary DBMS fares. While quality criteria are usually defined for a set of data, we extend, wherever possible, the definitions to the systems that manage this data.

Key Words: Databases, Information Quality Criteria, Assessment

1 THE QUALITY-IN-QUALITY-OUT PRINCIPLE

Arguably the most wide-spread architecture to store, manage, and retrieve structured data is the relational database management system (DBMS) architecture. Starting with System R [AB+79] of IBM, which evolved to the IBM DB2 database system, today there are many commercial systems storing Petabytes of data. Other prominent examples are Oracle database [Oracle], Microsoft's SQL Server [MSSQL], and MySQL [MySQL]. Research and development for DBMS follows two main directions: scalability and usability. With the growing demand to store more and more data, databases systems have scaled in the hardware they use and in the software managing the data. Additionally, administrators of databases and end-users of the data demand more and more functionality that either adds value to the DBMS or makes its use easier. In this paper, we analyze how well modern DBMS are able to meet user demands, or at least help database administrators (DBAs) meet user demands regarding their everyday work with the DBMS or applications built on top. Here, user demands are expressed as a set of information quality criteria taken from the empirical study of Wang and Strong [WS96].

Information quality is a measure to assess the value of data to perform the task at hand [WS96]. Other definitions mention fitness for use [TB98] or user satisfaction [DM92]. As DBMS are one of the most common means to generate, manage, and provide this data, it is worthwhile to examine how they influence the quality of the information they handle. This influence is both explicit within the core functionality of a DBMS and implicit through tools that help data providers, developers, managers, and consumers derive the most value from the data.

To examine DBMS with respect to the quality of information they are able to supply, we apply a large set of IQ criteria to DBMS as an entire system. Usually, IQ criteria are used to assess the quality of data and information, and not the quality of a system. Addressing this mismatch, we analyze not the DBMS itself, but its ability to provide high quality data. DBMS are not the sole source of high information quality, but they are designed to at least not diminish quality. While the well-known *garbage-in-garbage-out* principle holds for any system dealing with data, we postulate the *quality-in-quality-out* principle for modern, well-designed DBMS. For instance, if data is generated and inserted into a DBMS in a timely manner, a good DBMS will not unduly delay the accessibility of the data to users. Another example is the completeness of information: DBMS are developed to always return complete (and correct) answers to queries. Only if the stored base data is incomplete or incorrect will a DBMS answer with an inferior result. In this spirit we analyze several quality dimensions and provide details on if and how a typical DBMS meets IQ demands. In this paper we ignore the issue of software quality and assume a DBMS that correctly implements the SQL standard and its added functionality.

Structure of this paper. In the following section we introduce IBM's DB2 database system as a typical representative of commercial DBMS. Additionally, we present three different roles that DBMS users acquire, each with different needs towards information quality and each with different demands on a DBMS. In Section 3 we enumerate a comprehensive set of information quality criteria and discuss for each, if and how a DBMS used in different roles affects them. We conclude with a summary of our findings in Section 4.

2. DATABASE MANAGEMENT SYSTEMS AND THEIR USERS

A software product, such as a database management system (DBMS), is used by many different persons with different educational backgrounds, different IT needs, and, most importantly, with different roles inside an organization. These roles are the basis of our assessment of DBMS quality.

2.1 DB2 Universal Database – A Typical, Off-the-Shelf Database Management System

Figure 1 illustrates a layered DBMS architecture that has evolved over the past 30 years of relational database technology evolution and is typical of most DBMS products in the marketplace.Requests to the DBMS are formulated in a semantically rich, declarative language and submitted via a command line or programming interface. SQL (Structured Query Language, [SQL99]) has developed as the standard language for manipulating relational data, and interfaces such as JDBC and ODBC have evolved as the standard programming APIs by which SQL requests are submitted to the DBMS. Graphical tools, packaged and custom applications submit SQL requests through these programming interfaces. The maturity and wide adoption of these standards ensures that a wide variety of documentation is available to describe their use.

A query processor sits below the DBMS interface layer and executes requests to store and retrieve data as efficiently and robustly as possible. The query processor is made up of multiple components that work together to transform a declarative request for data into a series of executable steps that satisfy the request. The top level component is the query parser, which relies on a lexical analyzer and parser to interpret the request and populate a data graph that captures the semantics of the request. This data graph is passed to a query optimizer that analyzes the graph and explores several strategies for answering the request, eventually choosing the most efficient plan according to some criteria, such as to minimize response time, throughput, or resource utilization. Once a plan has been selected, the code generator gener-

ates code to implement the plan. The runtime execution engine executes the code, interacting with a storage manager to retrieve and combine the appropriate data to fulfill the original request.

The query processor handles not only requests to retrieve data, but it also handles requests to insert, update and delete data in the database. Such operations may be performed concurrently by multiple users. System services such as transaction management, concurrency control, logging, and recovery, maintain the *ACID* properties of a database that enable a DBMS to safely support concurrent query, insert, update, and delete operations by multiple users. ACID stands for Atomicity, Consistency, Isolation, and Durability (for details beyond the following brief descriptions of the properties see [Gra93]).

Atomicity refers to "all or nothing" behavior for database operations. Operations that must be logically performed together are grouped into *transactions*, and the DBMS provides transaction management services to ensure that either all or none of the operations of a transaction are performed in order to preserve a consistent view of data. For example, a bank transfer requires removing money from one account and adding it to another. The Atomicity property of the database ensures that if the operations are grouped together in a transaction, then a partial update – removing money from the first account without adding it to the second – will not occur.

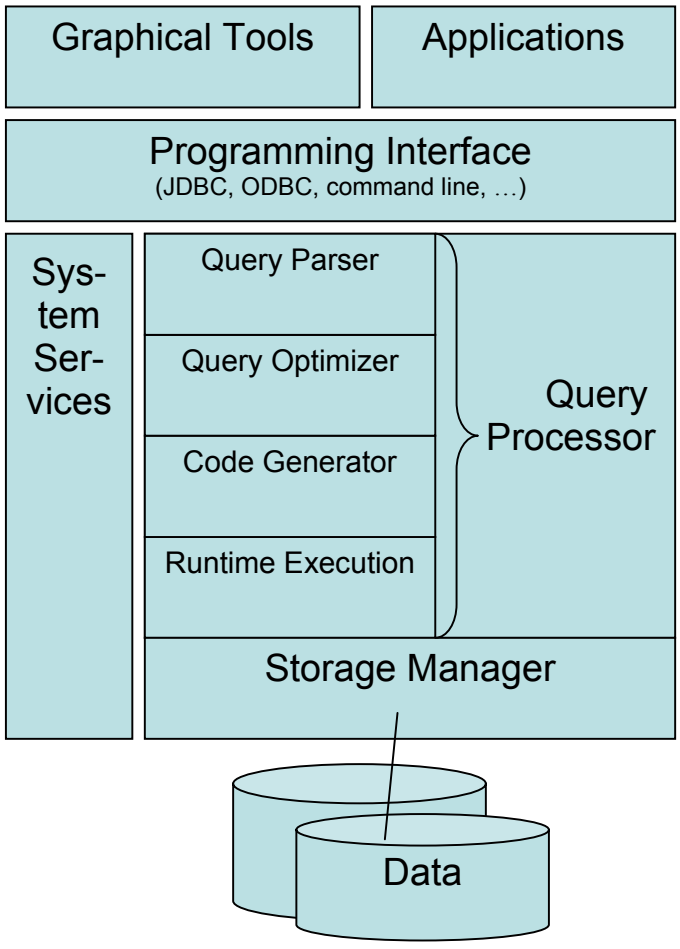

Figure 1: A generic DBMS architecture

Consistency refers to maintaining a consistent view of the data in the database both internally and with respect to the applications that access the data. Consistency is enforced by a DBMS by the services that provide atomicity and isolation. A committed transaction takes a database from one consistent state to another consistent state. For example, the two bank accounts of the previous example are in a consistent state before a transfer occurs. During the execution of the transaction, the database is temporarily inconsistent when the money is removed from one account but not yet added to the second account. However, after the transaction commits, the two bank accounts are again in a consistent state, with money successfully transferred from the first account to the second account.

Isolation ensures that concurrent users of a database will execute their transactions in isolation, and not interfere with each other's operations. For example, if Felix and Mary both initiate a transfer of funds for the same set of bank accounts, the DBMS's locking and concurrency services ensure that either Felix's transaction will execute to completion before Mary's, or vice versa. Thus, Mary will not see Felix's partial updates (e.g., the value of the account being transferred to), nor will Felix see Mary's partial updates.

Durability refers to the property that any transaction committed to the database will not be lost. Durability is maintained by the DBMS through logging and database recovery services. Logging records the state of database transactions, including whether they were committed, and database recovery restores a database to a consistent state in the event of a hardware or software failure. For example, when the transaction transferring money from one account to another is committed, the DBMS records enough information about the transaction, even in the event of a hardware failure.

Many of the challenges in information quality arise when integrating data from many, autonomous sources; duplicates appear when a real-world object is represented in more than one source, information is "tainted" when integrated with data from an untrustworthy source, completeness can be increased through integration. Simultaneously, much DBMS vendors have paid great effort to introduce integrating capabilities into their base DBMS product. These capabilities appear as data warehousing environments, as federated database systems, such as IBM's Information Integrator, or as grid-enabled DBMS, such as Oracle's 10g database. In fact, many of the integrating technologies have become fundamental components of the DBMS. Thus, in the following discussions we include DBMS integration technology whenever applicable to the IQ criterion at hand.

Users that interact with a DBMS directly typically rely on two types of graphical tools. Such tools are either packaged with the database itself or produced by third party vendors. Administrative tools, such as DB2's Control Center or Oracle's Enterprise Manager are used to configure and maintain the database, including such tasks as defining database layout, governing user access to the data, monitoring performance, and scheduling backups to ensure the data is recoverable in the event of a failure. Application development tools, such as DB2 Development Center, Computer Associates Erwin Data Modeler, or Rational Rose, are used to create database artifacts that are used by applications, including such tasks as developing models of the data required for an application, or generating functions and stored procedures that encapsulate much of the data access logic for the application. In conclusion, large database vendors offer same or similar base functionality, which is the focus of our research. We do mention DB2's additional functionality that is particular to data quality improvement.

2.2 Roles of DBMS customers

An exhaustive analysis of DBMS use may identify a large population of DBMS users, each with distinct purposes and roles. Rather than address all of these roles, we focus our attention on three important DBMS customers, shown in Figure 2, each with different needs with respect to information quality and each with different means to improve IQ.

- **Administrator**: Database administrators are trained DBMS experts who install and maintain DBMS and accompanying tools. Their role is to manage the integrity, security, and overall performance of a DBMS system, including such tasks as scheduling backups and database reorganization activities, optimizing storage layout, tuning performance in response to application requirements, and managing user access to the data. Administrators often use the administrative graphical tools and command line interface shown in Figure 1 to perform their tasks.

- **Developer**: Application developers create software to support a particular business need. An application developer works with a domain expert to define the application requirements, architect a solution, and implement application code to meet those requirements. The application is typically composed of distinct software components that encapsulate the logic for a particular aspect of the solution, and application developers specialize in component areas. For example, an application that produces a daily report of business transactions might be composed of a component that runs

queries against the database to retrieve the day's transaction information, a component that combines and summarizes the data according to specific business rules, and a web-based interface that formats the information in an easy-to-read format. Application developers typically use application development tools and the programming interface illustrated in Figure 1 to implement their applications.

- **Business Manager**: A business manager relies on applications developed by application developers in order to make robust business decisions. The health and robustness of the database and the applications built on top are critical to the line of business; success and profitability often depend on the reliability and quality information produced by the applications. If a DBMS that stores order information is unavailable for several hours, business transactions cannot occur, and the business loses revenue. If data is corrupted by the DBMS or if information is computed incorrectly, the business may be subject to fines or penalties.

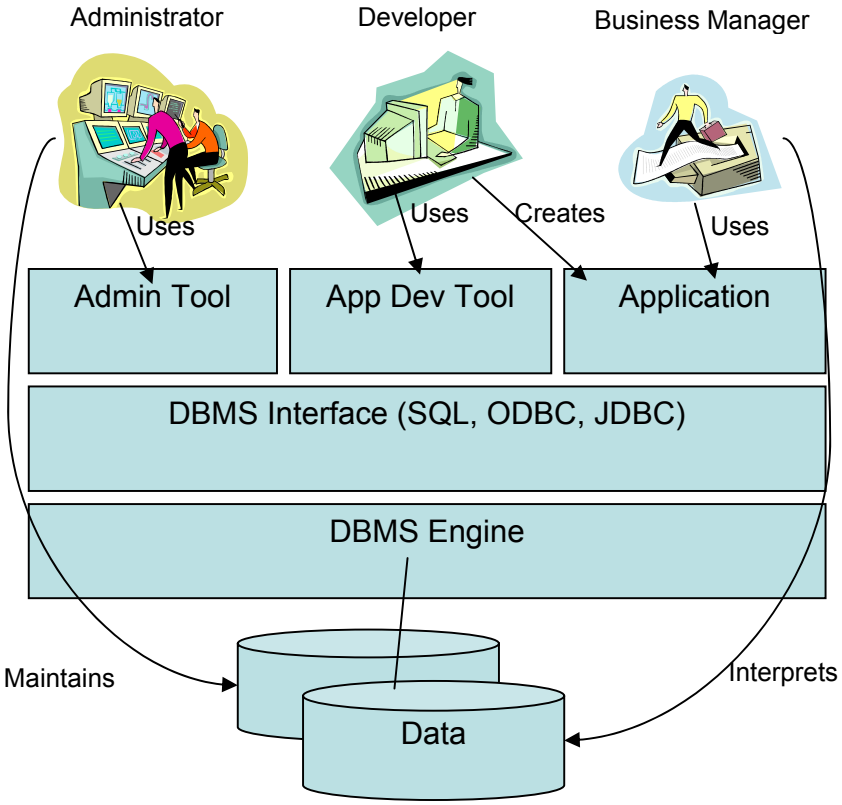

Figure 2: Three Roles for DBMS Usage

3. The Quality of Database Management Systems

We analyze a set of information quality criteria from the perspective of the three distinct customer roles described above. There have been many attempts to collect and classify IQ criteria, most notably the empirical study of Wang and Strong [WS96], in which the authors aggregated survey results to find a final list of fifteen IQ criteria. Here, the IQ criteria are ordered according to that classification, but we extended the set at appropriate places by several criteria that are of particular importance to DBMS users (and at which DBMS excel). For convenience, we quote the definitions of [WS96].

3.1 Intrinsic IQ Criteria

Believability
"Believability is the extent to which data are accepted or regarded as true, real, and credible" [WS96]. In a sense, believability is the expected accuracy. Due to the ACID properties described earlier, DBMS faithfully report data as it was entered into the database. A main source for doubts in the accuracy of data is the difficult formulation of complex SQL queries. Thus, managers must rely on the ability of developers to correctly write and document queries that have the desired semantics. Research has developed methods to automatically detect dubious queries and subqueries and issue warnings to the developer [Bra02]. For instance, some queries always produce empty results because they contain mutually exclusive filtering predicates. Such techniques have not yet found their way into commercial DBMS.

However, given high quality data and a correct query, i.e., a query that reflects the semantics of the application, the produced query result is guaranteed to be correct. Through the very good track record of DBMS, developers and managers worldwide nowadays entrust DBMS with even their most business-critical data, proving a strong belief that DBMS will correctly produce and transform the data that is stored within.

Accuracy
"Accuracy is the extent to which data are correct, reliable, and certified free of error" [WS96]. It is typically determined as the quotient of the number of correct values in a database and the overall number of values in the database.

With respect to the data as it is accessible, a DBMS always provides accurate answers to a query (in terms of DBMS, this property is called *correctness*). This IQ criterion best exemplifies the quality-in-quality-out principle of DBMSs. Correctness, and thus accuracy, is ensured by the use of checksums, locks and other methods of transaction processing. In particular, DBMS provide durability (the D in ACID), so that no data is lost. Since the main goal of a DBMS is to store data faithfully, vendors take great care to ensure the accurate insertion of data and the precise delivery of that data in response to queries.

General DBMS have no means to automatically correct inaccurate data. They do however provide some means to detect inaccurate data: Through the usage of pre-defined and user-defined types (UDTs), a DBMS verifies if a data value conforms to the syntactic definition of its type. Furthermore, administrators can implement triggers that perform certain actions and issue warnings when a possibly incorrect value, such as an outlier, is inserted into the database.

Objectivity
"Objectivity is the extent to which data are unbiased, unprejudiced, and impartial" [WS96]. It is trivial to show that DBMS have perfect objectivity: The durability property of DBMS ensures that data entered into a database is not changed. Since DBMS are fully automatic machines and software, they are not prejudiced and are always impartial.

Reputation
"Reputation is the extent to which data are trusted or highly regarded in terms of their source or content" [WS96]. Arguments similar to those for the Objectivity criterion also hold with respect to the reputation a DBMS conveys. In fact, there are occasions when the reputation of the DBMS increases the reputation of the data stored within. Major DBMS vendors, such as IBM, Oracle, and Sybase, have successfully mar-

keted mature DBMS products for many years and have thus gained a high reputation. IT managers have a tendency to trust data coming from a reputed DBMS more than from other less-renown IT products.

3.2 Contextual IQ Criteria

Value-Added
"Value-Added is the extent to which data are beneficial and provides advantages from their use" [WS96]. Apart from simply storing data and retrieving data upon request, modern DBMS have many capabilities to add value to data. In [WG97] the authors provide a list of features that DBMS (and in particular federated DBMS) may provide to add value to the base data. These features include simple aggregation of data, which most every DBMS is able to provide, search capabilities (such as those provided by DB2 Net Extender) to allow key-word searching, triggers to automatically react to certain events, etc. Many of the features listed there are directly provided by DBMS. For there rest, DBMS have standard interfaces so that third-party vendors can implement appropriate business intelligence applications on top of the DBMS. IBM's Information Integrator provides the ability to integrate multiple information sources within a single federated system, thereby directly providing value through integration, and indirectly through various capabilities for federated systems.

Relevancy
"Relevancy (or relevance) is the extent to which data are applicable and helpful for the task at hand" [WS96]. DBMS can improve relevancy of data in with two techniques: the ability to declaratively select appropriate data, and the ability to access additional, remote data.

Using the declarative SQL language, developers can implement pre-defined views that select and filter data according to the application and according to the needs of the business manager consuming the data. SQL views can limit the set of columns of a data set exported and can limit the rows of the data set according to some filter predicates.

Using technology to federate multiple, heterogeneous and remote data sources, federated DBMS, such as IBM's Information Integrator, are able to greatly increase the amount of data accessible to the developer. Thus, developers are given more freedom to include relevant sources and improve relevancy in the final, integrated result.

Timeliness
"Timeliness is the extent to which the age of the data is appropriate for the task at hand" [WS96]. Commercial DBMS provide several means to optimize the timeliness of data. At the database core level, DBMS ensure timeliness by making updates to the data visible immediately after the updating transaction is completed. High timeliness is jeopardized when same data is stored at multiple locations and then updated at only one location. Only after such updates are propagated to the other copies is the data once again up-to-date. Unfortunately, DBMS frequently duplicate data to ensure high response times. For instance, data accessed by a remote client is often cached at the client side. Another example is the deliberate replication of data at multiple clients. Finally, materialized views store pre-calculated query results from base data. In all cases, updates to the base data are not immediately reflected in the copies.

To improve timeliness, DBMS have implemented different schemes of update propagation, such as those available with DB2 Information Integrator Replication Server. Such replication offerings often support flexible latency characteristics for update propagation (continuous, event-driven, or scheduled), as well as transaction-consistent and table-at-a-time replication. Finally, there has been much applied research on the so-called view-update problem [LSPC00].

Completeness
"Completeness is the extent to which data are of sufficient breadth, depth, and scope for the task at hand" [WS96]. Usually, completeness can be measured as the number of records in a database, assuming that larger numbers are of higher utility (breadth). In [NFL04], we introduce a completeness definition that also accounts for the number of attributes of records and the number of (useless) null-values in database. The arguments to support our claim that completeness is well-accounted for by DBMS are similar to the ones for the accuracy criterion. In a sense, completeness is built-in in databases and with respect to the data as it is accessible, a DBMS always provides complete answers to a query.

Completeness is of great importance and particular difficulty when integrating multiple data sources. All modern DBMS provide flexible means to access remote data sources, either by supporting Extract-Transform-Load (ETL) processes to fill data warehouses (materialized integration) or by providing federated access to remote sources at query time (virtual integration). Many ETL vendors, such as Ascential Datastage or Oracle Warehouse Builder supply advanced ETL tools that unburden administrators from programming individual transformations and instead allow them to concentrate on the high-level definition of entire transformation processes. For virtual integration, extensive tooling exists to define so-called wrappers, which automatically translate and distribute queries to appropriate remote sources. The ability of wrappers to access sources of many different types and flavors (XML, Excel, flat files, other DBMS, Web Services, etc.) help administrators improve completeness by wrapping all sources relevant to a particular domain (see for instance the DB2 XML Wrapper [JS03] and a tool to easily create one [JMN03]). For example, consider a typical insurance claim, which may include account information stored in a relational database, photographs stored in a content management system, and a police report received as an XML document. Virtual integration enables the insurance application to view all of this data as though it were stored and queried through a single interface.

Appropriate amount of data
"Appropriate amount of data is the extent to which the quantity or volume of available data is appropriate" [WS96]. While selecting the set and amount of data that is *just right* for the task at hand is a difficult problem, DBMS help developers achieve this goal in two ways.

First, DBMS provide means to explicitly limit the amount of data returned through SQL extensions such as "`FETCH FIRST N ROWS ONLY`" in DB2 (for Microsoft SQL Server: "`SELECT TOP N ... FROM ...`"; for Oracle: "`OPTIMIZER_MODE = FIRST_ROWS_N`"). With this command the DBMS limits the number of rows returned to the data consumer. The technique improves latency (see below), addresses problems of limited bandwidth, and avoids users being overwhelmed by too much data.

Second, commercial DBMS place virtually no limit on the amount of data they can manage. [WC03] lists some of the largest known database instances by different criteria, such as number of rows, size in Byte, etc. According to that survey, the largest databases store close to 100 Terabyte of data and up to ½ trillion rows. In summary, large DBMS support appropriate amounts of data by providing capabilities to reduce the amount as much as desired and simultaneously allow the amount to be as large as necessary.

Latency and Response Time (new criteria)
Latency is the amount of time from issuing the query until the first data item reaches the user. Latency is particularly important for end-users such as business managers. Instead of waiting for the DBMS to calculate the entire result of a query or to produce all pages of a report, modern DBMS have the ability to "fast-track" first results. Thus, users can browse the first part of a result. Further parts are either calculated on demand (saving execution cost if they are never needed) or calculated in the background, i.e., while displaying first results.

The most important technique to ensure good (low) latency is that of pipelined query execution. Data is sent through an execution pipeline, so that calculated results arrive at the user's application even while base-data is still being read [Gra93,PMCLS90]. All modern DBMS are able to recognize a potential for pipelines for a given query and are able to execute the query in pipelined mode. Developers of applications are able to use certain functions, so that the applications also profit from low latency of the DBMS. For instance the getNext() function explicitly produces only the next subset of answers. Administrators can use DB2's SQL extension "FETCH FIRST N ROWS ONLY" to achieve low latency and also decrease query execution cost.

While latency measures the time until a first part of a response reaches the data consumer, response time measures the delay in seconds between submission of a query by the user and reception of the *complete response* from the DBMS. Response time is the main and often sole criterion for traditional database optimizers. In consequence there has been an enormous amount of academic and industrial research to improve response time of queries even over very large databases, much of which has been implemented in commercial DBMS. For lack of space, we refer the reader to surveys for query optimization for centralized databases [Gra93] and for distributed databases [Kos00]. While many of these optimization techniques must be installed and maintained by hand, recent work on autonomic computing lifts the burden of DBMS administrators. Automatic advisors, such as the Design Advisor of IBM DB2 Stinger, recommend optimal settings for many DBMS parameters, for instance regarding the use of indices and the use of materialized views [Z+04].

3.3 Representational IQ Criteria

Interpretability
"Interpretability is the extent to which data are in appropriate language and units and the data definitions are clear" [WS96]. Interpretability is highly dependent on the interface that users have to the data. DBMS interfaces have two roles. One role is to *interpret* given data using metadata, such as the schema, constraints, etc. The other role of interfaces is to allow users *interaction* with the data, such as creating new data, reformatting data, and aggregating data.

Interpretation of data largely depends on the underlying schema, which can be well or poorly designed. DBMS support good schema design through visual tools, such as Computer Associate's ERWin Data Modeler. Also, DBMS provide multiple language support to improve interpretability. The main tools around DB2 are available in 15 different languages. Recent developments have given administrators and programmers more flexibility in schema design: Table and attribute names are no longer restricted to eight bytes, and user defined types (UDTs), user-defined functions (UDFs), and table functions allow development of specialized data items.

Interfaces for *interaction* with the data in a DBMS range from highly technical interfaces (SQL query language) to somewhat technical interfaces (query- and view-building tools, such as DB2 SQL Assist) to non-technical interfaces for business managers (report generators).

Understandability
"Understandability (or ease of understanding) is the extent to which data are clear without ambiguity and easily comprehended" [WS96]. Thus, understandability measures how well a DBMS presents its data, so that the user is able to comprehend its semantic value. Standard DBMS query interfaces represent data in

simple tables, using column names stored in the system catalog. Until several years ago, column names were restricted to eight characters in length, severely limiting the expressivity and thus understandability of the data. This has now changed and developers can give tables and columns more verbose names. Also, many third-party applications specialize in visualizing complex data.

One of the main concerns of DBMS administrators is to maintain efficient query response times to all queries. In particular, administrators are concerned with optimization of queries, i.e., finding an efficient execution strategy. DB2 provides an advanced visualization tool called Visual Explain, which graphically displays a query execution plan along with table sizes, join algorithms, filter selectivity and result size. Such visualization improves understanding of the data and the DBMS view of the data.

Visualization of query result data itself is not the focus of the DBMS, but rather of applications built on top. In consequence, business managers are rarely exposed to a simple tabular display of data directly from the DBMS, but instead use sophisticated tools that are able to additionally visualize data in other models, such as bar charts that more accurately reflects the business domain.

Representational consistency
"Representational consistency is the extent to which data are always represented in the same format and are compatible with previous data" [WS96]. Representational consistency from an administrator's point of view is accomplished through DBMS support of standard data exchange protocols, such as JDBC and ODBC, and standard data exchange formats, such `CSV` (comma separated lists) and `XML` (extensible markup language). These standards, which are supported by all DBMS vendors, allow fairly easy data exchange at the byte-level among different data sources.

Achieving representational consistency in terms of the structure of the data as reflected in a schema is a far more difficult task. Two main research areas have dealt with the integration of heterogeneously modeled data: schema integration (see for instance [BLN86]) and schema mapping (see for instance [H+02]). While schema integration attempts to generate a new integrated schema from different representations, schema mapping simply represents correspondences among heterogeneous schemas. The correspondences are then interpreted as data transformations that help achieve representational consistency among multiple data sources. Unfortunately, both approaches are not yet well supported in commercial DBMS. However, several prototypical tools with graphical interfaces for schema mapping are under development by vendors, such as Clio at IBM Research [H+02] and Rondo at Microsoft Research [MRB03].

To conclude, achieving representational consistency is an extremely difficult task not yet well supported by DBMS and research towards semi-automatically solving it is only now underway.

Representational conciseness
"Representational conciseness is the extent to which data are compactly represented without being overwhelming" [WS96]. Data representation in most modern DBMS follows the relational model, which structures data into relations (tables) storing tuples (records) each having a set of attributes (stored in columns) [Co70]. For many applications this is an intuitive and compact method to model data. In particular, through the process of normalizing tables, the representation can be rendered as compact as possible.

On the other hand, many applications have a more hierarchical view of data, so that an XML data model might be preferred. Most modern DBMS have some capability of storing and retrieving XML data, or even producing XML data from a set of tables through the new SQL/XML standard [SQLX].

As already argued for the Relevancy criterion, DBMS provide many means for developers to reduce the amount of data to some relevant and concise subset as needed for any particular application. Reduction is achieved by selecting relevant columns (called projection), filtering relevant rows (called selection), and summarizing data (aggregation).

3.4 Accessibility IQ Criteria

Accessibility/Availability
"Accessibility is the extent to which data are available or easily and quickly retrievable" [WS96]. Availability of a DBMS is the probability that a feasible query is correctly answered in a given time range. Availability is a technical measure concerning both hardware, software, and their compatibility. Availability and accessibility are crucial features for large scale, mission critical applications. For example, a DBMS server failure in the middle of a time-sensitive stock transaction or bank transfer could have an immediate effect on the core business.

DBMS have invested heavily in technology to achieve high availability. Centralized DBMS running on a single server have advanced in recent years to parallel DBMS and distributed DBMS running on multiple servers possibly distributed worldwide. Distribution opens the possibility of two means to achieve high availability: clustering and replication [WM03]. Both clustering and replication exploit distribution to minimize the risk that DBMS software or hardware introduces as a single point of failure for applications that depend on the DBMS. DBMS clusters exploit features in the operating system on which the DBMS is deployed to automatically react to software or hardware failures. In the event of a failure, the DBMS cluster automatically engages a backup system to take over for the failed component. DBMS cluster technology achieves high availability for an entire DBMS system. It requires specific software and hardware to implement, and places restrictions on the distance and hardware over which it can be deployed. DBMS replication is an alternative technology for high availability. Replication allows an administrator to control the subset of data to be managed for availability, including what subset of the data to replicate, where to place replicas (including across platforms and over long distances), how to keep them up-to-date, and which replica to use to answer a particular query [BG+04]. DB2 Information Integrator Masala Replication Server, for example, supports multiple source/target topologies for replication, including 1-to-many, many-to-1, and many-to-many.

In addition to high availability features, DBMS provide a number of features to support user accessibility to the data managed by the DBMS. As shown in Figure 1, DBMS provide a number of different interfaces from visual to programming APIs to access data. Most standard DBMS APIs such as ODBC are fairly mature; however, DBMS vendors tend to keep up with current trends and technologies. For example, most DBMS vendors have developed tools to allow administration and user access to DBMS via a Web interface in a browser. Examples include IBM's DB2 Web Query Tool, Oracle's Application Server Portal, and Microsoft's SQL Server Web Data Administrator. In addition to flexibility with regard to interfaces, DBMS also provide features that allow administrators to customize user accessibility to data. Examples of such features include user authentication, object-level privileges, database views, and tools like DB2 Query Patroller that regulate database resource use by user-type.

To summarize, DBMS provide a number of built-in availability and accessibility features to provide reliable access to data at the time it is needed by the applications and users that depend on it.

Access Security
"Security is the extent to which access to data can be restricted and hence kept secure" [WS96]. Security covers technical aspects, such as cryptography, secure login etc., but also includes the possibility of anonymization of the user and authentication of the DBMS by a trusted organization. Modern DBMS feature several abilities to ensure high security. First, the SQL standard specifies commands to GRANT and REVOKE detailed privileges to individual data objects, such as tables, views, functions, etc. Second, DBMS implement sophisticated authentication policies at servers and clients. Finally, DBMS vendors usually provide entire security tool suites ranging from security auditing and monitoring, intrusion detection, to privacy and security management tools.

Customer support & Documentation (new criterion)
Customer support is the amount and usefulness of human help provided to users either electronically or personally. Documentation is the amount and usefulness of (electronic) documents describing the DBMS and its functionalities and guiding customers during system deployment and application development. Business managers are usually not confronted with the DBMS themselves, but rather with applications on top, for which there exist separate support systems and documentation.

Administrators and developers: Large installations of DBMS are usually accompanied by agreements on technical support. The degree of support is available at different levels and costs. In addition to this explicit and contractually agreed upon assistance, all vendors supply a multitude of resources to guide users. Questions to dedicated newsgroups are answered by experts; context-sensitive help-systems are provided within the DBMS and online; detailed documentation (IBM's Redbooks), tutorials, technical articles, white papers are available for download; user conferences, such as the International DB2 User Group conference (IDUG), are held regularly throughout the world; and reference implementations to common tasks are available for download, etc. In summary, customer support and documentation can be considered good, in particular for DBMS of large vendors.

4. SUMMARY AND CONCLUSIONS

In conclusion, DBMS are able to uphold the quality-in-quality-out principle postulated in this paper. Table 1 on the following page summarizes our results by listing for each criterion the most important techniques to uphold a quality level or to improve the information quality in that dimension.

In several cases, DBMS are able to improve on the quality of the data within. Most notably, DBMS add value to data by providing many advanced value-adding services, such as aggregation, integration, etc. as mentioned in [WG97]. The powerful SQL query language supported by all DBMS and the vendor-specific extensions allow developers to manipulate data in many ways to enhance its quality: Selection and projections allows users to limit data to just the right subset improving relevancy, appropriate amount and response time. Tools at the metadata level, such as schema design and integration tools, improve representational consistency and conciseness, understandability and interpretability. Through their support of the ACID properties, algorithms at the database core ensure that data is not lost or tampered with, upholding criteria such as accuracy, completeness, and believability. The ability of DBMS to integrate data from multiple heterogeneous sources also improves completeness and relevancy of data. Finally, many usability tools to maintain and use DBMS improve security, support, and accessibility.

The only detriment to information quality imposed by the use of DBMS is the potentially limiting expressiveness of the relational model. DBMS data is modeled in relations using a set of attributes, while often a hierarchical or other model is the more natural way of expressing the structure of data. With the recent

move of major DBMS towards support of the XML data model and the XQuery language, this disadvantage is already fading.

To summarize, it should come as no surprise that relational databases are able to uphold the *quality-in-quality-out* criteria described in this paper. DBMS technology has matured over twenty years of development, and confidence in the reliability and robustness of the DBMS model is reflected in countless mission-critical applications deployed on DBMS systems today.

IQ Criterion	DBMS capabilities	Summary
Believability	ACID properties	%
Accuracy	ACID, UDTs, Triggers, checksums, outlier detection	%
Objectivity	Durability, "unbiased" machine	%
Reputation	Maturity of DBMS, reputation of DBMS	+
Value Added	Aggregation, federation, integration, search, etc.	+
Relevancy	Powerful query language, access remote data	+
Timeliness	Update propagation for materialized views and replication	%
Completeness	ACID, ETL, Wrappers for heterogeneous data	+
Appr. Amount	First N Rows, scalability to many Terabyte	%
Latency / Response Time	Pipelining, First N Rows, advanced query optimization, autonomic computing	+
Interpretability	Visual schema design, language support, UDTS, UDFs, query builders	+
Understandability	Metadata restrictions (8 char), query plan explanation for administrators, visualization tools on top of DBMS	–
Repr. Consistency	JDBC, XML, schema integration tools, schema mapping, relational data model	+/–
Repr. Conciseness	Relational data model, normalization, SQL/XML, query language	+
Accessibility	Distribution, replication, web interfaces, portals	+
Security	SQL privilege management, authentication, security auditing	+
Support / Documentation	Phone and email support, tutorials, manuals, conferences, newsgroups	+

Table 1: Summarizing effects of DBMS on IQ (– detrimental, % no effect, + beneficial)

REFERENCES

[AB+79] M.M. Astrahan, M.W. Blasgen, D.D. Chamberlin, J. Gray, W.F. King III, B.G. Lindsay, R.A. Lorie, J.W. Mehl, T.G. Price, G.R. Putzolu, M. Schkolnick, P.G. Selinger, D.R. Slutz, H.R. Strong, P. Tiberio, I.L. Traiger, B.W. Wade, R.A. Yost: *System R: A Relational Data Base Management System*. In IEEE Computer 12(5): 42-48, 1979.

[BG+04] S. Bourbonnais, V. Gogate, L. Haas, R. Horman, S. Malaika, I. Narang, V.S. Raman: An Information Infrastructure for the Grid Providing Data Virtualization and Distribution. To appear in IBM Systems Journal, Volume 43, Number 4 (November 2004).

[BLN86] C. Batini, M. Lenzerini, S. B. Navathe: *A comparative analysis of methodologies for database schema integration*. In ACM Computing Surveys 18(4): 323-364, 1986.

[Bra02] S. Brass: *Semantic Errors in SQL*. In Proc. of the GI Workshop Grundlagen von Datenbanken 2002, Fischland, Germany.

[Co70] E. F. Codd: *A Relational Model of Data for Large Shared Data Banks*. In Communications of the ACM 13(6): 377-387(1970)
[DM92] W.H. Delone, E.R. McLean: *Information systems success: the quest for the dependent variable*, In Information Systems Research 3(1), pages 60-95, 1992.
[Gra93] G. Graefe: *Query Evaluation Techniques for Large Databases*. In ACM Computing Surveys 25(2): 73-170, 1993.
[H+02] M.A. Hernández, L. Popa, Y. Velegrakis, R.J. Miller, F. Naumann, H. Ho: *Mapping XML and Relational Schemas with Clio*. In Proceedings of the International Conference on Data Engineering ICDE 2002: 498-499
[JMN03] V. Josifovski, S. Massmann, F. Naumann: Super-Fast XML Wrapper Generation in DB2: A Demonstration. In Proceedings of the International Conference on Data Engineering ICDE 2003: 756-758.
[JS03] V. Josifovski, P.M. Schwarz: Querying XML data sources in DB2: the XML Wrapper. In Proceedings of the International Conference on Data Engineering ICDE 2003: 809-820.
[Kos00] D. Kossmann: *The State of the art in distributed query processing*. In ACM Computing Surveys 32(4): 422-469, 2000.
[LSPC00] W. Lehner, R. Sidle, H. Pirahesh, R. Cochrane: *Maintenance of Automatic Summary Tables*. In Proceeding of the SIGMOD Conference 2000: 512-513
[MRB03] S. Melnik, E. Rahm, P.A. Bernstein: *Rondo: A Programming Platform for Generic Model Management*. In Proceedings of the SIGMOD Conference 2003: 193-204
[MSSQL] Microsoft SQL Server, www.microsoft.com/sql/
[MySQL] MySQL open source database, www.mysql.com/
[NFL04] F. Naumann, J.C. Freytag, U. Leser: *Completeness of Information Sources*, In Information Systems, Elsevier, in press, 2004.
[Oracle] Oracle Database, www.oracle.com/ip/deploy/database/oracle9i/
[PMCLS90] H. Pirahesh, C. Mohan, J.M. Cheng, T.S. Liu, P.G. Selinger: *Parallelism in Relational Data Base Systems: Architectural Issues and Design Approaches*. In Proceedings of the International Symposium on Databases in Parallel and Distributed Systems DPDS 1990: 4-29
[SQL99] International Standards Organization (ISO), Information Technology---*Database Language SQL,* Standard No. ISO/IEC 9075:1999.
[SQLX] The SQL/XML standard. http://www.sqlx.org/
[T98] G.K. Tayi, D.P. Ballou: *Examining Data Quality*, In Communications of the ACM 41(2): 54-57, 1998.
[WC03] Winter Corporation: *The TopTen Program,* http://www.wintercorp.com /VLDB/2003_TopTen_Survey/ TopTenProgram.html, 2003.
[WG97] G. Wiederhold, M.R. Genesereth: *The Conceptual Basis for Mediation Services*. In IEEE Expert 12(5): 38-47, 1997.
[WM03] M. Wright: *An Overview of High Availability and Disaster Recovery Features for DB2 UDB*. In IBM Developerworks, April 2003: http://www.ibm.com/developerworks/db2/library/techarticle/0304wright/ 0304wright.html
[WS96] R.Y. Wang, D.M. Strong: Beyond accuracy: What data quality means to data consumers. In Journal on Management of Information Systems, 12(4):5-34, 1996.
[Z+04] D.C. Zilio, C. Zuzarte, S. Lightstone, W. Ma, G.M. Lohman, R. Cochrane, H. Pirahesh, L. Colby, J. Gryz, E. Alton, D. Liang, G. Valentin: *Recommending Materialized Views and Indexes with IBM DB2 Design Advisor*. In Proceedings of the International Conference on Autonomic Computing, 180-188, 2004.

TRADEMARKS