Quality-driven Integration of Heterogeneous Information Systems

Felix Naumann*‡
Humboldt-Universität zu Berlin
naumann@dbis.informatik.hu-berlin.de

Ulf Leser‡
Technische Universität Berlin
leser@cs.tu-berlin.de

Johann Christoph Freytag
Humboldt-Universität zu Berlin
freytag@dbis.informatik.hu-berlin.de

Abstract

Integrated access to information that is spread over multiple, distributed, and heterogeneous sources is an important problem in many scientific and commercial domains. Typically there are many ways to obtain answers to a global query, using data from different sources in different combinations, but in general, it is prohibitively expensive to obtain all answers. While much work has been done on query processing and choosing plans under cost criteria, very little is known about the important problem of incorporating the information quality aspect into query planning.

In this paper we describe a framework for multidatabase query processing that fully includes the quality of information in many facets, such as completeness, timeliness, accuracy, etc. We seamlessly include information quality into a multidatabase query processor based on a view-rewriting mechanism. We model information quality at different levels: First, we perform a quality-driven source selection and continue only with the best sources. Second, we compute query-dependent information quality of the view definitions that describe the content of sources. Finally we determine the overall quality of plan alternatives by aggregating these information quality scores to find a set of high-quality query-answering plans.

1 Introduction

Integrated access to information that is spread over multiple, distributed and heterogeneous sources is an important problem in many scientific disciplines. For instance, a current list of molecular biology information systems (MBIS) enumerates more than 400 entries [Inf98] of publicly available data sources. These are frequently overlapping, replicated, or disjoint both in the type of data they store, as in the actual objects they contain: a particular gene of the human X chromosome is described in different facets in multiple sources: phenotype and related diseases in a source for human genes [McK94], location information in a source for X chromosome specific data [LWG+98] or in a source for the entire human genome [LCPL98],

*Contact author: Felix Naumann, Institut für Informatik – dbis, Humboldt-Universität zu Berlin, Unter den Linden 6, 10099 Berlin, Germany; Phone: +49 30 2093-3025

‡This research was supported by the German Research Society, Berlin-Brandenburg Graduate School in Distributed Information Systems (DFG grant no. GRK 316)

known mutations in mutation databases, sequences in the sequence databases [SMS+98] or from the sequencing institute directly etc. All these databases store data of varying quality. Considering for instance *accuracy*, one source might present only unprocessed, raw results while another source might offer confirmed results. Considering *timeliness*, one source provides results that are years old (but possibly still important) while another is continuously updated.

Providing an integrated, homogeneous and comprehensive access mechanism in such a framework is considered as one of the "most pressing problems" in genome research today [Rob95]. From the computer science point of view, it is tackled by techniques such as federated databases [SL90] or mediator-based data integration systems [Wie92]. In our work we focus on virtual, tightly integrated, structured integration systems for mainly two reasons:

- We believe that only tight federations, i.e., federations that have one single, non-redundant and homogeneous schema, offer a sufficient degree of abstraction and comfort for an average user. We do not consider the process of creating this integrated schema (see for instance [BLN86]), but proceed in a top-down fashion: assuming the existence of a global schema, we describe the content of data sources with respect to this schema in form of rules.

- We deploy a virtual integration process since data in molecular biology is produced very rapidly, and access to the most actual data is often crucial.

In this setting, it is the task of a *mediator* to automatically decompose, translate and distribute queries against the global schema into queries against physical data sources.

Unfortunately, there usually are many ways to answer a global query, using data from different sources in different combinations. For instance, there are numerous sources that hold mapping information of the human X chromosome [LLRC98]. Each of these sources is potentially interesting for a global query requiring such data, but executing all possible combinations is infeasible. The mediator, therefore, must choose between logically equivalent sources. Naturally, this selection should regard the quality of the data, comprising criteria such as timeliness, completeness, accuracy etc. Considering information quality (IQ) during query processing is very important in a scientific and especially statistical context for several reasons: First, there are simply too many relevant sources with varying IQ, so selecting the "best" is essential. The goodness of a plan depends primarily on the expected quality of the results and not on pure technical criteria such as response time. Poor IQ can have considerable social and economic impact [WS96]. For example, missing year 2000 compliance is just one example of many others. Second, scientific databases are very sensitive regarding timeliness and accuracy of data; results become outdated quickly, and the intrinsic fuzziness of many experimental techniques leads to data of varying quality, depending on the quality standards of a particular data source. Finally, the result of the integration process is directly influenced by data quality. A large company has reported, that up to 60% of the information integrated to their data warehouse was unusable due to the poor quality of the input data [Orr98].

In this paper we describe our approach to a tight integration of classical query planning, which tries to find correct combinations of source queries for a given global query, with the assessment and consideration of information quality for source selection. We extend an existing framework for query planning [Les98] which is based on rules that define the semantic relationship between *queries*. We found this level of granularity very helpful since many IQ criteria cannot be assigned to an entire source. For instance, one source might

be very up-to-date in the integration of X chromosome data and only occasionally include Y chromosome data. Timeliness of this source cannot be described with one value, but is naturally assigned to different queries against the source. Another example is a source which offers two different interfaces, one using a simple WWW interface, the other being a direct SQL channel. Logical query planning process must consider that the SQL interface can probably answer more complex queries than the WWW interface - but the two ways of access probably differ also in the average response time, the required amount of parsing, update frequency, etc.

Our method therefore distinguishes between source-specific, query-specific and attribute-specific quality criteria (see Section 3). We extend the existing planning algorithm with two steps: First we reduce the overall number of sources in a pre-processing phase. This is reasonable since it is often the case that certain sources are 'worse' than others in all criteria. We however ensure not to lose any source that is unique in some aspect, i.e., the only source storing a certain attribute value. Filtering sources is important since our planning algorithm has time-complexity which is worst-case exponential in the number of correspondences and hence roughly in the number of sources. Second, we rank all plans by evaluating query-specific and attribute-specific IQ scores following the join-structure of a plan. We eventually execute plans by decreasing quality until a stop criterion is reached: either the 10% best plans or until a total quality threshold is reached.

Related work. Database interoperability and data integration for molecular biology databases is addressed in a number of projects. For instance, OPM [CKM+98], P/FDM [KDG96] or bioKleisli [DOTW97] are multidatabase query languages that were developed especially for this purpose. In these loosely coupled systems the database administrator can define integrated views which simulate a global schema. However, our planning mechanism is strictly more powerful than views (see Section 2). The TAMBIS project [BBB+98] uses a formal ontology for the integration of semantically heterogeneous data sources, but until now little is known about their query planning and source selection methods.

Our logical planning method uses a local-as-view approach ([Ull97]) similar to the Information Manifold ([LRO96]) or InfoMaster ([DG97]). However, our notion of query correspondence assertions is an extension to the ideas of these projects in that it combines local-as-view with global-as-view modeling. In [Les98], we presented an improved algorithm for the query planning problem in this framework. However, the pure planning is not the focus of this paper, but the interleaving of logical planning with quality considerations.

There is much research showing the importance of information quality for businesses and users [WS96, Red98], many techniques have been proposed to improve and maintain quality of individual information sources [Wan98], However, none of the mentioned integration projects considers information quality in any way. With the exception of TAMBIS, users must always specify exactly which sources shall be used for the different relations or classes of their queries. Quality-driven source selection and plan selection is not possible in these systems.

Several research projects such as the GlOSS system [GGMT94] or the Information Manifold [FKL97] focussed on the problem of source selection for text based information systems. However, selection is typically confined to criteria used in information retrieval systems using word-counting measures or to traditional DBMS criteria such as response time.

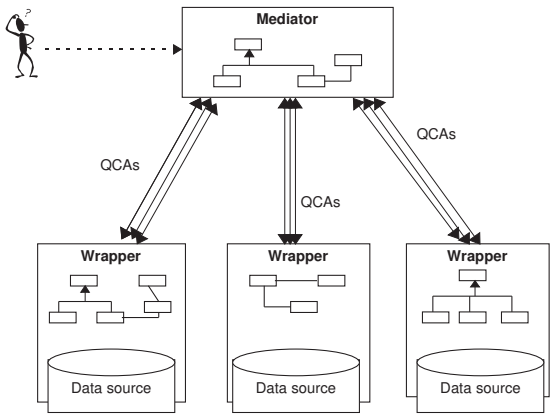

Figure 1: Integration architecture: Source contents described with set of QCAs

Structure of this paper. In this work we describe a framework for a quality-driven query processor. We shortly describe the logical query processing in Section 2. Section 3 formally introduces information quality as a set of properties together with methods to determine their values. In Section 4 we then show how IQ plays a decisive role in query processing and leads to better final results. We conclude in Section 5 and give a brief outlook to future work.

2 Logical Planning of Queries

Our framework is based on a declarative approach to query processing in heterogeneous environments. For space limitations we describe the framework only very briefly. Details can be found in [Les99, Les98].

Our system refines the generic wrapper - mediator architecture described in [Wie92] (see Figure 1). It uses the relational data model inside the mediator. We assume that each data source is wrapped by a source-specific module. The task of this *wrapper* is to offer a relational export schema and query interface, hiding the particular data model, access path, and interface technology. Wrappers are used by *mediators* which integrate the data from different wrappers.

Example. We will use the following example throughout the paper, demonstarting the flavor of our approach. The example contains various simplifications, such as object names as global keys or equivalence of attribute units, domain and structure. For space limitations we refrain from using real-life information systems but describe some fictitious sources with fictitious IQ scores. We describe a global mediator storing data about genes. Our global schema is given in Table 1. The mediator considers 5 different sources, shortly described in Table 2. Using these sources, we answer a user query for all genes of the X chromosome together with their related diseases, sequence, origin, and annotations (Section 4).

To answer queries and to select sources, a mediator must know the content of each source with respect to its own schema. This knowledge is encoded in sets of query correspondence assertions (QCAs), that are set-oriented equations between relations of a source and relations of the mediator's global schema. QCAs are specified by a human administrator and form the

gene(Gn,Di)	Gene relation storing gene names and related diseases.
sequence(Gn,Se,Or,An)	Sequence relation storing the nucleotide sequence of genes, together with origin and annotation.
EST(En,Ch,Po,P1,P2)	EST relation storing EST name, chromosome and position on the chromosome, together with two primer sequences.
genecluster(Gn,En)	Relates genes to EST clusters.

Table 1: Global schema of the mediator.

S_1	Source 1 stores sequences which it copies infrequently from other sites, sometimes introducing parsing errors.
S_2	Source 2 also copies sequence data from other sites, also infrequently updated, but uses more sites and is hence more complete.
S_3	Source 3 is the WWW server of a research institute which does its own sequencing. Sequence data is hence highly up-to-date, but only few annotations are provided. Furthermore, the server is frequently unavailable.
S_4	Source 4 is a renowned commercial provider of EST data. Provides two interfaces: *www* is a WWW server. Use is free, but connection is slow and only the chromosome location is retrievable. *direct* is a fast SQL connection through which clients can retrieve all attributes, but there is a charge per query. Primer sequences are available for 95% of their ESTs, positions on the chromosome for only 60%.
S_5	Source 5 is a directly accessible relational database which stores mapping and sequence data for genes. The schema is different than our global schema. The mediator uses two queries: one relates genes and sequences, the other relates genes to their EST clusters.

Table 2: Information sources with brief descriptions. For their interface relations as exposed by a wrapper see Table 3.

basis for the semantic reasoning process performed inside the mediator. They only capture the logical properties of sources; quality properties are defined in the next section. A QCA has the general form:

$$MQ \leftarrow Si.vj \leftarrow WQ$$

where the mediator query MQ denotes a set of relations forming a conjunctive query that specifies which data of the global schema are represented by the source. $S_i.v_j$ is an arbitrary view name, with the name of the source as prefix. The wrapper query WQ denotes the corresponding query against the export schema of the source. Both queries MQ and WQ must be conjunctive [AHV95], and the view must be *safe* in both directions, i.e., variables in the view must appear in both queries. A QCA asserts that the tuples obtained by executing WQ, projected to the variables in $S_i.v_j$, contain only intensionally correct value combinations for the appearances of the attributes of $S_i.v_j$ in MQ.

For instance, the following simple QCA describes the content of S_1:

$$\text{sequence(Gn,Se,Or,An)} \leftarrow \text{S1.v1(Gn,Se,Or,An)} \leftarrow \text{seq(Gn,Se,Or,An)}$$

Global relations can appear in many QCAs. The *extension* of a global relation is defined as the union over all wrapper relations that partly overlap. For instance, the global extension

of *sequence* would be the union over the extension of *seq* in sources S_1, S_2 and S_3. Describing S_5 requires two QCAs, one for each of the two queries used by the mediator:

gene(Gn,Di), sequence(Gn,Se,-,An) ← S5.v1(Gn,Di,Se,An)
← genes(GID,Gn,Di), genepart(GID,PID), part(PID,Se,An);
gene(Gn,-),genecluster(Gn,En), EST(En,Ch,-,-,-) ← S5.v2(Gn,En,Ch,P1,P2)
← clustering(Gn,En,CID), cluster(CID,Ch), primers(En,P1,P2);

See Table 3 for the list of QCAs that describe the content of the different sources in our example.

S_1:	QCA_1	sequence(Gn, Se, Or, An) ← S1.v1(Gn, Se, Or, An) ← seq(Gn, Se, Or, An)
S_2:	QCA_2	sequence(Gn, Se, Or, An) ← S2.v1(Gn, Se, Or, An) ← seq(Gn, Se, Or, An)
S_3:	QCA_3	sequence(Gn, Se, Or, An) ← S3.v1(Gn, Se, Or, An) ← seq(Gn, Se, Or, An)
S_4:	QCA_4	EST(En, Ch, -, -, -) ← S4.v1(En, Ch) ← www(En, Ch)
	QCA_5	EST(En, Ch, Po, P1, P2) ← S4.v2(En, Ch, Po, P1, P2) ← direct(En, Ch, Po, P1, P2)
S_5:	QCA_6	gene(Gn, Di), sequence(Gn, Se, -, An) ← S5.v1(Gn, Di, Se, An)
		← genes(GID, Gn, De), genepart(GID, PID, -), part(PID, Se, An)
	QCA_7	gene(Gn, -), genecluster(Gn, En), EST(En, Ch, -, -, -) ← S5.v2(Gn, En, Ch, P1, P2)
		← clustering(Gn, En, Cl), cluster(Cl, Ch), primers(En, P1, P2)

Table 3: QCAs describing the semantics of the seven possible wrapper queries.

For a given user query against the mediator schema, the mediator tries to find combinations of QCAs (*plans*) that are *semantically contained* [ASU79] in the user query. We call such a plan *correct*. In Section 4.2 we will briefly describe an algorithm which finds all correct plans. Note that results obtained by executing different plans (there can be more than one) will be different if they address different sources. The *complete* answer to a user query with respect to the given QCAs is the union over the answers of all correct plans.

However, there can be prohibitively many correct plans. Consider a query asking for the sequence of a specific gene. The mediator detects that S_5 can be used for the *gene* part of the query and S_1, S_2 and S_3 for the *sequence*-part. This already sums up to three different plans. Suppose there are two more sources storing genes, the number of correct plans would increase to nine. However, if the user is, for instance, particularly interested in complete annotation, plans using S_3 are not very promising; if highly up-to-date data is required, S_1 could probably be ignored. In the following sections we will show how we integrate logical planning and information quality.

Note that our query planning is fundamentally different from classical query optimization for relational DBMS: our planning finds plans that are correct and that possibly generate different results, while optimization considers different plans to execute a given query, producing the same result.

3 Information Quality and Integrated Scientific Databases

There is no common or agreed definition or measure for information quality or the quality of an information source, apart from such general notions as "fitness for use" [Jur74, TB98]. In this section we define information quality (IQ) as a set of quality criteria. An information

source or a query plan achieves certain scores in each of these criteria. We aggregate the scores to determine a total IQ score using them to rank the sources and plans. Based on this ranking we execute only the best plans with the best sources disregarding the rest. Several questions arise: Which IQ criteria define information quality, how can we measure quality, and how do we evaluate the measured scores?

To answer the first question, Wang and Strong have empirically identified fifteen IQ criteria regarded by data consumers as the most important. They classified these into "intrinsic quality", "accessibility, "contextual quality", and "representational quality" as shown in Table 4 [WS96]. Their framework has already been used effectively in industry and government. In the following chapter we adapt this set of criteria to the scope of molecular biology information systems.

IQ Category	**IQ Dimensions**
Intrinsic IQ	Accuracy, Objectivity, Believability, Reputation
Accessibility IQ	Access, Security
Contextual IQ	Relevancy, Value-Added, Timeliness, Completeness, Amount of data
Representational IQ	Interpretability, Understandability, Concise representation, Consistent representation

Table 4: IQ Categories and Dimension by Wang and Strong

3.1 IQ Criteria for the Integration of Information Sources

As already mentioned in the introduction, it is not always sufficient to assign quality measurements to entire sources. Since our planning process already uses queries as the basic level of correspondence, it is natural to assign IQ scores to QCAs. Furthermore, assessments can even apply only to certain attributes. For instance, we described S_3 as having relatively few annotations attached to the sequences they store. In such cases, we need to define specifically that the completeness of the *annotation* attribute in this QCA is not very high. We therefore distinguish three classes of quality assignment, which are each treated differently:

- *Source-specific criteria* determine the overall quality of an information source. Criteria of this category apply to all information of the source, independently from how it is obtained. Criteria scores of this class stay unchanged as long as the source itself does not dramatically change.

- *QCA-specific criteria* determine quality aspects of specific queries that are computable by a source. Using this finer granularity, we can e.g. model different prices for different queries. For instance, S_4 charges a fee for queries as defined through QCA_5 but not for queries using QCA_4.

- *User query-specific criteria* assess the quality of an information source in terms of answering a specific user query. Hence the scores for these criteria can only be determined at "query time". For example, we calculate the appropriateness of a QCA for a query and punish such QCAs that deliver unnecessary attributes. We also allow to discern between the importance of single attributes of a user query (see completeness).

For the domain of Molecular Biology Information Systems we slightly modified the original set of IQ criteria by Wang and Strong considering the specific needs of biologists using the systems and the properties of existing information systems. Criteria that are not applicable to our area of discourse or data integration model are omitted. Also, we have added the two criteria which play a particularly important role for MBIS, namely reliability and price, and have split the accessibility criterion. Table 5 summarizes and categorizes the criteria used. As usual, we assume independence of the criteria. In the section that follows we give a brief explanation of each criterion.

Dependency	**Criterion**	**Brief explanation** (for details, see below)
Source-specific	Ease of understanding	User ranking
	Reputation	User ranking
	Reliability	Ranking of experimental method
	Timeliness	Average age of the data
QCA-specific	Availability	Percentage of time the source is accessible
	Price	Monetary price of a query
	Represent. Consistency	Wrapper workload
	Response Time	Average waiting time for response
	Accuracy	Percentage of objects without errors
	Relevancy	Percentage of real world objects represented
User query-specific	Completeness	Fullness of the relation
	Amount	Number of unwanted attributes

Table 5: Classification of Quality Criteria for MBISs

Please note that for such a classification it is not always clear which criterion fits in which class of Table 5. For instance, if sources charge the same amount of money for each query, the price criterion should be only source-specific. If, on the other hand, a source provides data produced by different experimental methods, the reliability criterion should be QCA-specific. Depending on the application domain and the structure of the available sources, the classification may vary. Moving a criterion down the list, e.g., from source-specific to QCA-specific, might increase the number of scores that have to be obtained a-priori; moving it up loses the possibility to express fine-grained differences. Finally, if the information of a source can be partitioned into sets with heavily diverging IQ scores, the QCAs of this source can be split according to this partitioning and each new QCA will be given individual IQ scores.

3.2 Defining and Measuring IQ Criteria

In the following paragraphs we translate the written definition of each criterion from Wang and Strongt [WS96] to a numerically measurable score or function. We are aware of the difficulties of numerically expressing certain criteria, but since the absolute IQ scores are not important, but rather their relative values, we believe that our approach is reasonable. Another problem that all projects addressing IQ are facing, is the question of objectivity of scores. Some of the criteria below cannot be measured, but are highly subjective, such as

source reputation. We suggest user profiles, i.e., sets of IQ scores for all subjective criteria that are set-up once by each user and then used for all its future queries.

Ease of understanding is the "extent to which data are clear without ambiguity and easily comprehended."(all citations from [WS96]) Since the actual information gained from a source will be in relational form, Ease of understanding in terms of the data will be equal for all sources. However when data is enhanced with other information of the source, such as experimental method etc., or if a link to the source is added to the response, the understandability of the source itself will be an issue.

The score for this criterion is a grade from 1 (not understandable) to 10 (very easily understandable). It could possibly be determined with the help of a questionnaire.

Reputation is the "extent to which data are trusted or highly regarded in terms of their source." Reputation is, of course, a very personal criteria. However, we observed that most biologists actually prefer certain sources over others for reasons that are not always clear. For instance, people tend to trust data from their own institute more than external data, as they also tend to prefer well-known sources. Reputation is of course highly subjective, but we believe that the mediator must take into account the user's preferences. Again, reputation is a grade from 1 (bad reputation) to 10 (very good reputation).

Reliability is used to assess the methods that are used inside the source to generate or to analyze data. Reliability is also a grade between 1 and 10. We use this newly introduced criterion to discern between results from sources using different experimental methods. For instance, YAC mapping data has to cope with the high rate of chimerism in many YAC libraries, which sometimes makes results suspicious. Since many MBIS integrate data from other sources, we use reliability also to assess the diligence of this process. For instance, a low reliability score of such a 'warehouse' MBIS indicates frequent discrepancies to the original data.

Timeliness is the "extent to which the age of the data is appropriate for the task at hand". In a fast growing area such as molecular biology it is reasonable to use the update-frequency of data source rather than the average age of the data as criterion. To determine the timeliness-score we rely on update information provided by the information source.

Availability of an information source is the probability that a feasible query is answered in a given time range. Availability is a technical measure concerning hardware and software of the source and the network connection between the mediator and the source. For simplicity, we assume that a source either delivers its complete response or no response at all. A partial response, which may occur if the system breaks down during transmission is counted as no response.

Availability can be measured with the help of statistics derived from previous (calibration-) queries to the information source. Knowledge of the technical equipment and software of the information source can help determine availability. It is given as the percentage of time that the source is accessible.

Price is a new criterion, which we added due to the growing importance of commercial data providers in many application domains. It is the amount of (real) money a user has to pay for a query and is given by the provider. We observed that commercial data providers either charge on a subscription basis for their data set or on a pay-by-query basis. We assume

pay-by-query for our model; if sources have a subscription fee, the score of the price criterion has to be estimated. The price per query is measured in US dollar.

Representational Consistency is the "extent to which data are always presented in the same format and are compatible with previous data." Since we review multiple sources, we extend this definition to not only compare compatibility with previous data but also with data of other sources. We assume wrappers to deliver a relational export schema which is always consistent with the global schema against which we query.

Representational consistency is thus a criterion to measure the work of the wrapper necessary to parse files, transform units and scales or translate identifiers into canonical object names. We measure representational consistency as the average time consumption for this task.

Response Time is the amount of time between submission of the mediator query MQ as defined by the QCA and receiving the result. It is measured in seconds. QCA-response times can be identified using calibration techniques. Note, that we do not perform traditional cost-based optimization. Response time is one quality criterion among many others.

Accuracy is the "extent to which data are correct, reliable and certified free of error." For our context this is the percentage of data without *data errors* such as non-unique keys or out of range values. Such errors are usually produced during data input. Pierce gives a survey of counting methods for data errors [Pie98]. We suggest a sampling based method, where the overall number of errors is estimated using only a small sample. Accuracy is not to be confused with intrinsic error rates of the experimental method which we capture in Reliability.

Relevancy is the "extent to which data are applicable and helpful for the task at hand." Usually this criterion is regarded as highly user-dependent, since only the user can determine whether something is relevant or not. This is true for information retrieval type queries, when no schema is available. Since we use a relational schema against which user queries are posed, the Relevancy score determines the percentage of real world objects the information source has stored. Thus, it measures *horizontal fitness* of a source (see Figure 2).

For comparability and stability, we want to measure relevancy as a percentage, however, using this unit comes with some difficulties. For example, consider vertical fitness regarding human genes. One could try to use the absolute number of human genes, but this number is not yet known. Using the current number of known genes would require a frequent adaptation of scores. We therefore do not prescribe what number shall be used as the maximum value and only require that the same value is used for all QCAs of a certain relation.

Completeness is the "extent to which data are of sufficient breadth, depth, and scope for the task at hand". We observed that not all attributes of a user query are equally important for a user. It is quite common to pose queries such as "Give me all genes on Xq23 and their exact locations - and, if possible, also related phenotypes". A source that has more data for the first two attributes is preferable to a source with less data for gene positions, but more attached phenotypes.

Completeness is an attribute-level criterion directly depending on a user query, it cannot be assessed in advance. Users specify a weighting together with the query by scoring each attribute of the desired result with a score between 1 and 100 (very important). Furthermore, we require that each key (that is used in a join) gets a score of 100 to be compatible with our logical planner.

Later on, we shall use this user query weighting together with the attribute-specific completeness measure *in each QCA* to calculate completeness scores for each attribute of a QCA with respect to the user query. Hence, Completeness also measures the *horizontal fitness* of a QCA (see Figure 2).

Amount is defined as the "extent to which the quantity or volume of available data is appropriate". While a large number of objects in a source is desirable (see relevancy), a too large number of attributes is undesirable and inappropriate. Consider again the query for genes on Xq23 and their locations. A source which automatically delivers a large amount of additional attributes, creates an unnecessary large response which must be dealt with by the mediator.

We measure Amount of a QCA as the number of unnecessary attributes it contains, i.e., attributes not selected in the user query. Thus, amount measures *vertical fitness* of a source, as explained in Figure 2.

Gn	**Se**	**An**
100 %	80 %	50 %
DMD	CGAT...	musc. dyst.
ETX1	`null`	`null`
F8A	GACT...	`null`
ZFX	GATT...	zink-fing.

(a) horizontally unfit – few genes, few annotations; vertically fit – exact attribute match

Gn	**Se**	**Date**	**An**	**Pos.**
100 %	80 %	80 %	80 %	100 %
DMD	CGAT...	1/1/98	musc. dystr.	Xp21
ETX1	`null`	10/11/97	`null`	Xp21
F8A	GACT...	5/5/98	Fact. 8-assoc.	Xq28
LYP	ATTG...	`null`	lymph. synd.	Xq25
SYNI	TAGC...	12/3/97	synapsin I	Xp11
ZFX	GATT...	14/1/98	zink-finger	Xp22

(b) horizontally fit – many genes, good annotation; vertically unfit – too many attributes

Figure 2: Two Sources for $UQ = (Gn, Se, An)$

Wang & Strong-criteria omitted. Some criteria can be omitted, mainly due to our use of the relational model. We assume that the global relational schema models all desired data, thus all correct answers to the user query are relevant. The same is true for the Value Added criterion: The value of the response can be derived from the schema before queries are placed. Concise Representation can again be omitted as a criterion, since we assume a relational global schema and the response is as concise as the user states this in the user query. Interpretability is the "extent to which data are in appropriate language and units and the data definitions are clear." In our relational model it is the responsibility of the wrapper to resolve any discrepancies in format, units etc. Thus, this criterion is omitted. Objectivity was omitted since we assume that all experiments resulting in database entries are performed unbiased. Access Security is a criterion concerning the security of the users data. However, we solely deal with read-only type information systems, where no user information is stored. Believability is merged with the Reputation-criterion to minimize user-interaction with the system.

4 Finding the Best Sources and Plans

Creating good execution plans for a user query in any DBMS involves a search space of all plans and a valuation model to compare plans with one another. We propose a three-phase approach to quality-driven information integration in multidatabases: In the first phase we reduce computational cost of the second phase by filtering out low quality sources based on the source-specific criteria, and by continuing with only the best sources. The second phase uses the QCAs of the remaining sources to generate all correct plans, thus establishing the search space for the last phase. In that phase we explore the entire search space using the remaining criteria and choose the best plans for execution. For simplicity we do not apply a search strategy to combine Phases 2 and 3, rather we materialize and examine the entire search space. Input to our model is the set of IQ scores for the criteria as described in Section 3.2. Some of these scores are predetermined while others are functions of the user query and user preferences. The result is an IQ vector with numerical scores for each criterion which will be used to rank sources and plans.

In the following sections we describe each step in detail by using an example. We use the global schema as described in Table 1 and the sources as described in Table 2. The content of each source with respect to the global schema is described by seven QCAs presented in Table 3.

	S_1	S_2	S_3	S_4		S_5	
	QCA_1	QCA_2	QCA_3	QCA_4	QCA_5	QCA_6	QCA_7
EoU(grade)	5	7	7	8		6	
Rep.(grade)	5	5	7	8		7	
Reli.(grade)	2	6	4	6		6	
Tim.(days)	30	30	2	1		7	
Av.(%)	99	99	60	80	99	95	95
Pr.(US$)	0	0	0	0	1	0	0
R.C.(sec)	1	1	.5	.7	.2	.7	.7
R.T.(sec)	.2	.2	.2	3	.1	1	1
Ac.(%)	99.9	99.9	99.8	99.95	99.95	99.95	99.95
Relev.(%)	60	80	90	80	80	60	60

Table 6: IQ Scores s_{ij} of 5 MBISs used in 7 QCAs. Scores are partly inferred from the informal description in Table 2. Note that completeness and amount are not contained since they depend on the specific user query.

To find a ranking of the sources in Phase 1 or a ranking of plans in Phase 3 based on multiple criteria one faces two problems: (i) The range and units of the IQ scores of the criteria vary, making it necessary to scale the scores. (ii) The importance of the criteria may vary making it necessary to find a user-specific weighting of the criteria. Several *multiple attribute decision making* methods have been proposed to solve these problems [Nau98]. To find the best sources in the first phase, we use the "Data Envelopment Analysis" method; to rank the execution plans in the third phase we apply the "Simple Additive Weighting" method.

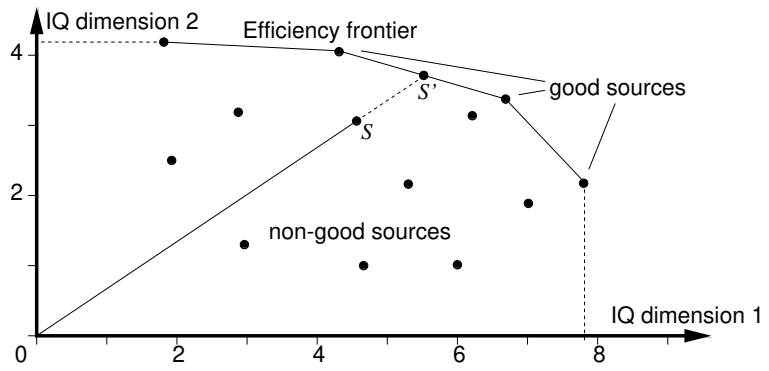

Figure 3: Classifying Sources with Data Envelopment Analysis

4.1 Phase 1: Source Selection

Our logical planning algorithm can potentially generate an exponential number of plans in the length of the user query and the number of QCAs. Furthermore, finding such plans is also exponential in the number of QCAs [LMSS95]. Therefore, we thrive to decrease this number before we start planning. For this purpose, we use the source-specific quality criteria, thus "weeding out" sources that are qualitatively not as good as others. Our goal is to find a certain number or percentage of best sources independently of any user-specific weighting. The mediator performs Phase 1 only once after start-up and does not repeat it until an information source dramatically changes in a source-specific criterion, or until a new information source is added to the system.

To evaluate such a large amount of sources in a *general*, user-independent way, in [NFS98] we have proposed Data Envelopment Analysis (DEA) developed by Charnes et al. as a general method to classify a population of observations [CCR78]. The DEA method avoids the mentioned decision making problems of scaling and user weighting by defining an efficiency frontier as the convex hull of the unscaled and unweighted vector space of IQ dimensions. Figure 3 shows this vector space for two arbitrary IQ dimensions. Those sources on the hull are defined as "good", those below as "non-good". Consider the non-good source S in Figure 3. Assuming constant returns to scale, the virtual but realistic source S' is constructed as a convex combination of the two neighboring sources on the efficiency frontier. Clearly source S' is better than source S, thus S is non-good.

To determine whether a source is on the frontier or below, we solve the following linear program (LP) once for each information source S_{j_0} with IQ scores s_{ij}:

$$\begin{array}{lrll} \text{maximize} & IQ(S_{j_0}) := & \sum_i w_i \cdot s_{ij_0} & \\ \text{subject to} & IQ(S_j) \;\; = & \sum_i w_i \cdot s_{ij} & \leq 1 \;\; \text{for all sources } j = 1, \ldots, n \\ & & w_i & \geq \varepsilon > 0 \;\; \text{for } i = 1, \ldots, 4 \end{array}$$

The result of each LP is the optimal quality score $IQ(S_{j_0})$ of the examined source. The score for each source is either 1 (on the frontier / good) or below 1 (below the frontier / non-good). By fine-tuning the ε-parameter we can vary the number of good sources to the desired percentage.

For further planning, we want to completely disregard non-good sources but at the same time not reduce the global schema. There is a danger of removing a source that has a low IQ but is the *only* source providing a certain attribute of the global schema. Thus, the DEA

process is repeated several times to avoid suppressing a source which exclusively provides a specific attribute: For each attribute of the global schema we determine those sources that provide this attribute and the set of good sources among them. In the following two phases we only consider sources in the union of these sets. In this way, the global schema remains intact.

Example. Due to the small number of sources in our example this phase only excludes source S_1 which is dominated by S_2. That is, S_2 is equal to or better than S_1 in all criteria.

4.2 Phase 2: Plan Creation

The goal of this phase is to find all combinations of QCAs that together obtain semantically correct answers to a given user query. Every QCA defines a view on the mediator schema, and those views, respectively their corresponding wrapper query WQ, are the only queries that are directly executable. We must find combinations of such views that generate correct tuples. This is equivalent to the problem of answering a query against a relational schema using only a set of views on the same schema. Levy et al. show that this problem is decidable for conjunctive queries and conjunctive view definitions [LMSS95]. They show that one has to test query containment for potentially all combinations of views up to a certain length. Chandra and Merlin showed that this test is already NP-complete [CM77]. In [Les98] we described an improved algorithm that exploits the fact that in our framework one of the two queries for the containment test is fixed (the user query), while the other is combined out of pre-defined building blocks (the set of QCAs). For space limitations, we here use a simpler algorithm similar to the one in [LRO96].

Example. Imagine a user query (UQ) asking for all genes of the X chromosome together with their related diseases, sequence, origin, and annotations. We answer this query by joining the *gene* relation with *sequence* for the attributes origin and annotation. We must also join *gene* to *EST* to ensure the chromosome-condition:

$$UQ(Gn(100), Di(100), Se(100), Or(30), An(70)) \leftarrow$$
$$\text{gene}(Gn, Di), \text{sequence}(Gn, Se, Or, An), \text{genecluster}(Gn, En), \text{EST}(En, Ch, \text{-}, \text{-}, \text{-}), Ch =' X';$$

The user weightings for each attribute are used to compute the completeness score later on. First, for each relation of UQ we determine the set of QCAs (bucket) which contain them in their mediator query MQ (see Table 3). We must also check if the QCAs *export* all necessary attributes, i.e., those that are required in UQ:

$$\begin{array}{llll} \text{bucket}(gene) & = \{QCA_6\} & \text{bucket}(genecluster) & = \{QCA_7\} \\ \text{bucket}(sequence) & = \{QCA_2, QCA_3\} & \text{bucket}(EST) & = \{QCA_4, QCA_5, QCA_7\} \end{array}$$

QCA_7 does not occur in bucket(*gene*) because it does not export the required Di attribute; the same holds for QCA_6 in bucket(*sequence*) and attribute Or. QCA_1 does not appear in bucket(*sequence*) since S_1 was deleted from the set of sources in Phase 1.

In a second step we enumerate the cartesian product of all buckets and check for each combination, (1) if it has no contradictions, for instance contradicting conditions, (2) if it is semantically contained in UQ, and (3) whether it can be minimized, i.e., whether certain QCAs are redundant. In our case, all combinations are contained and have no contradictions.

Furthermore, all double occurrences of QCAs are redundant. We end up with the following plans, each of them possibly producing a different set of correct tuples for UQ. In Phase 3 we predict the IQ of these results and choose the best plans for execution.

$$P_1 = QCA_6 \bowtie QCA_2 \bowtie QCA_7 \bowtie QCA_4 \qquad P_4 = QCA_6 \bowtie QCA_3 \bowtie QCA_7 \bowtie QCA_4$$
$$P_2 = QCA_6 \bowtie QCA_2 \bowtie QCA_7 \bowtie QCA_5 \qquad P_5 = QCA_6 \bowtie QCA_3 \bowtie QCA_7 \bowtie QCA_5$$
$$P_3 = QCA_6 \bowtie QCA_2 \bowtie QCA_7 \qquad P_6 = QCA_6 \bowtie QCA_3 \bowtie QCA_7$$

A plan is executed by computing the WQs that correspond to the MQs which form the plan, propagating variables bindings from QCA to QCA as usual. If a WQ is executed, the resulting tuples are temporarily stored in an instance of the mediator schema. For each tuple of the result one tuple is generated for each occurrence of each relation of MQ. Variables in MQ that appear in the view head are bound to the corresponding values in the result tuple; variables that do not appear in the view head are either filled with `null` or with relationship-preserving key values that are generated automatically. These keys can never be contained in the final answer: if the corresponding attributes are requested by a user query, plans that do not export them, i.e., do not contain them in their $S_i.v$, would not be semantically contained by the definition of containment. Attributes that have a '-' in their position in MQ are also filled with `null`.

4.3 Phase 3: Plan Selection

Given the set of correct plans, the goal of this phase is to qualitatively rank the plans of the planning phase and ultimatively to restrict plan execution to some best percent of all plans, or alternatively, to as many plans as possible or necessary to meet certain cost- or quality-constraints.

Following the DBMS approach of cost models for query execution plans with a tree-structure, we define a quality model for the tree-structured plans created in Phase 2. Leaves represent QCAs that deliver the base data. Those data are subsequently processed within the inner nodes of the tree, which represent inner join operators performed by the mediator.

Plan selection proceeds in three steps: First the IQ scores of the QCAs are determined (3a). The quality model then aggregates these scores along tree paths to gain an overall quality score at the root of the tree (3b). Finally, this score is used to rank the plans (3c).

Phase 3.a: QCA Quality. An IQ vector of length 8 is attached to each QCA, i.e., one dimension for each non-source-specific criterion of Table 5. The criterion category specifies how the scores for the individual criteria are determined:

- *QCA-specific* criteria have fixed scores for each QCA. Thus they are determined only once or whenever the the corresponding source undergoes major changes.
- The scores of the *user query-specific* criteria on the other hand are recalculated for each user-query UQ, depending on the set and weighting of attributes specified. Amount is simply the number of unnecessary attributes, i.e., attributes returned by the QCA but not specified in the user query. Completeness of a QCA is calculated at attribute level: The completeness score of each attribute of the QCA is weighted with the importance specified in the user query $(1 - 100)$. The resulting scores are summed up and divided by the sum of the user query weights. The result is a weighted average completeness of the attributes in a QCA.

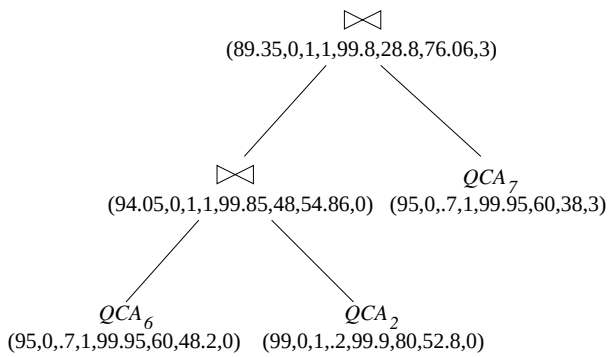

Figure 4: Merging IQ vectors in join nodes in plan P_3

The general IQ vector for QCAs is (abbreviated):

$$\begin{aligned} IQ(QCA_i) &:= (s_{i5}, \ldots, s_{i11}) \\ &= (Av, Pr, RC, RT, Ac, Rel, Com(UQ), Am(UQ)) \end{aligned}$$

Example. We determine the following IQ vectors for the QCAs participating in plans P_1 through P_6. The first six scores are taken from Table 6, the remaining two scores are calculated using the user query UQ.

$$\begin{aligned} IQ(QCA_2) &= (99, 0, 1, .2, 99.9, 80, 52.8, 0) & IQ(QCA_5) &= (99, 1, .2, .1, 99.95, 80, 20, 4) \\ IQ(QCA_3) &= (60, 0, .5, .2, 99.8, 90, 49, 0) & IQ(QCA_6) &= (95, 0, .7, 1, 99.95, 60, 48.2, 0) \\ IQ(QCA_4) &= (80, 0, .7, 3, 99.95, 80, 20, 1) & IQ(QCA_7) &= (95, 0, .7, 1, 99.95, 60, 38, 3) \end{aligned}$$

Up to this point, each leaf node of each plan-tree is assigned an IQ vector. However, we have no total IQ vector for the plans. These scores are obtained through the plan quality model.

Phase 3.b: Plan Quality. Corresponding to the idea of *cost models* for DBMSs, we propose a *quality model* to calculate the total IQ score of a plan. Since we only consider join-operators, a plan is a binary tree with QCAs as leaves and join operators as inner nodes. The IQ vector for an inner node is recursively calculated according to Equation (1) as a combination of the IQ vectors of its left and right child nodes l and r.

$$IQ(l \bowtie r) = IQ(l) \bowtie IQ(r) := (s_{l5} \bowtie s_{r5}, \ldots, s_{l12} \bowtie s_{r12}) \quad (1)$$

The $\bowtie$-operator (or "merge function") is resolved according to Table 7. Figure 4 shows the plan tree for $P_3 = QCA_6 \bowtie QCA_2 \bowtie QCA_7$ with its aggregated IQ vectors.

Since all merge functions in Table 7 are both commutative and associative, a change of the join execution order within a plan has no effect on its IQ score. This is desirable, since the quality perceived by the user is that of the query result and not the quality of how this result is obtained (the plan). Thus, unlike planning in traditional DBMSs, we do not have to consider the execution order of the joins within the plan. Also, we do not consider the execution time of joins performed by the mediator since we assume, that execution time is dominated by the response times of the sources.

Criterion	Merge function "$\circ$"	Brief explanation
Availability	$s_{l5} \cdot s_{r5}$	Probability that both sites are accessible.
Price	$s_{l6} + s_{r6}$	Both queries must be payed.
Repr. Consistency	$\max[s_{l7}, s_{r7}]$	Wrapper integrates sources in parallel.
Response Time	$\max[s_{l8}, s_{r8}]$	Both children are processed in parallel.
Accuracy	$s_{l9} \cdot s_{r9}$	Probability that left and right side do not contain an error.
Relevancy	$s_{l10} \cdot s_{r10}$	Probability for join match.
Completeness	$s_{l11} + s_{r11} - s_{l11} \cdot s_{r11}$	Probability that either left or right side has non-`null` value (operations at attribute level).
Amount	$s_{l12} + s_{r12}$	All unnecessary attributes must be dealt with.

Table 7: Merge functions for Quality Criteria

Example. The six plans have IQ vectors

$IQ(P_1) = (71.00, 0, 1, 3, 99.75, 23.04, 78.06, 4)$ $IQ(P_4) = (43.32, 0, .7, 3, 99.65, 25.92, 75.94, 4)$
$IQ(P_2) = (88.45, 1, 1, 1, 99.75, 23.04, 78.06, 7)$ $IQ(P_5) = (53.61, 1, .7, 1, 99.65, 25.92, 75.94, 7)$
$IQ(P_3) = (89.35, 0, 1, 1, 99.80, 28.80, 76.06, 3)$ $IQ(P_6) = (54.50, 0, .7, 1, 99.70, 32.40, 73.94, 3)$

Up to this point, the scores are neither scaled nor weighted.

Phase 3.c: Plan Ranking. The previous phase delivers a set of plans with an IQ vector for each plan. The scores of the vectors must be scaled, weighted, and compared to find a total IQ score for each plan and thus define a ranking of the plans. To this end, we use the Simple Additive Weighting (SAW) method. It is one of the simplest but nevertheless well perceived decision making methods, in that its ranking results are usually very close to results of more sophisticated methods [HY81]. The method is comprised of three basic steps: Scale the scores to make them comparable, apply the user weighting and sum up the scores for each source.

The IQ scores of the criteria availability, accuracy, relevancy, and completeness are scaled according to Equation (2), where $s_j^{\min}$ and $s_j^{\max}$ are the minimum and maximum score respectively in criterion j (see below). The criteria price, representational consistency, response time, and amount are negative criteria, i.e., the higher the score, the worse the quality. Thus, they are scaled according to Equation (3). With these scaling functions all scores are in $[0, 1]$, the best score of any criterion obtains the value 1, and the worst score of any criterion obtains the value 0. This property assures comparability of scores across different criteria and in different ranges.

For the weighting step SAW requires a weight-vector $W = (w_1, \ldots, w_m)$ specified by the user such that $\sum_{j=1}^{m} w_j$ equals 1. The weight-vector reflects the importance of the individual criteria. It can either be defined by the user, or a default vector can be used. For a plan P_i the overall quality score $IQ(P_i)$ is calculated as the weighted sum of Equation (4).

$$v_{ij} := \frac{s_{ij} - s_j^{\min}}{s_j^{\max} - s_j^{\min}} \quad (2) \qquad v_{ij} := \frac{s_j^{\max} - s_{ij}}{s_j^{\max} - s_j^{\min}} \quad (3) \qquad IQ(P_i) := \sum_{j=1}^{m} w_j \cdot v_{ij} \quad (4)$$

The final IQ score of the plan is in $[0, 1]$ and gives the ranking position of the plan. After the IQ scores for all plans have been calculated, we choose and execute the best plans.

Example. With the indifferent weighting vector $W = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ where all criteria have equal importance, the following IQ scores are obtained (in ranking order):

$$IQ(P_3) = .7663 \quad IQ(P_6) = .697 \quad IQ(P_1) = .5023$$
$$IQ(P_2) = .4559 \quad IQ(P_4) = .4429 \quad IQ(P_5) = .3771$$

A user preferring quick response time at any price, might specify the weighting $W = (\frac{1}{8}, \frac{0}{8}, \frac{1}{8}, \frac{2}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8})$ and obtain a ranking of the plans in the order P_3, P_6, P_2, P_5, P_1, P_4. P_1 is ranked lower than before because it includes QCA_4 which has a very high response time. Despite its high price, plan P_2 is ranked higher than before since it has a low response time. A user preferring high relevancy even if accuracy is low might specify the weighting $W = (\frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{1}{8}, \frac{0}{8}, \frac{2}{8}, \frac{1}{8}, \frac{1}{8})$ and obtain a ranking of the plans in the order P_6, P_3, P_4, P_1, P_5, P_2. In this weighting, P_3 is ranked highest, since it includes QCA_3 which has a high relevancy.

With the exception of the completeness criterion, all merge functions decrease the aggregated IQ scores with each additional QCA. Thus, there is a natural tendency favoring short plans, i.e., plans consisting of few QCAs. Not only does this reflect the influence of the criteria, it also conforms to intuition: Biologists will probably not be happy to accept results where the four attributes of the query are generated in four different sources.

5 Conclusion and Outlook

We have proposed a novel method to the well known and important, yet frequently ignored problem of considering information quality in logical information integration. This problem has not, to our best knowledge, been adequately addressed before. Our results offer a solution to the notorious problem of information overload, based on a filtering of important information based on a rich set of quality criteria. With the help of these criteria quality-driven information integration identifies high quality plans which produce high quality results. Clearly, the selection of quality criteria is a subjective task. However, our method is by no way restricted to the criteria we used in this paper.

We have described merge functions for each criterion which calculate the quality of the information in a join result. Due to the associativity of these merge-functions, we determine the quality of the result independently of how this result is created. This freedom will allow us to include easily binding patterns in the user query and the QCAs, which can potentially dictate a specific join order. Furthermore, a traditional post-optimization can be performed to find the best join order of the chosen plans without influencing their quality score.

Future work will also include a tighter cooperation of the plan creation phase and plan selection phase: Information quality scores can be used in a branch & bound fashion to dramatically improve planning time. Lifting our current model to a higher level, we plan not only to calculate the quality of plan results, but also the quality of the union of several plans to find the best combination of plans to execute. We belive that the same principles of calculating and merging quality scores apply.

References

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases.* Addison-Wesley Publishing Company, Reading, Massachusetts, 1995.

[ASU79] Alfred V. Aho, Yehoshua Sagiv, and Jeffrey D. Ullman. Efficient optimisation of a class of relational expressions. *ACM Transactions on Database Systems*, 4(4):435–454, 1979.

[BBB+98] Patricia G. Baker, Andy Brass, Sean Bechhofer, Carole Goble, Norman Paton, and Robert Stevens. Tambis: Transparent access to multiple bioinformatics information sources. In *Proc. of the 6th Int. Conf. on Intelligent Systems for Molecular Biology*, pages 25–34, Montreal, Canada, 1998. AAAI Press, Menlo Park.

[BLN86] C. Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[CCR78] A. Charnes, W.W. Cooper, and E. Rhodes. Measuring the efficiency of decision making units. *European Journal of Operational Research*, 2:429–444, 1978.

[CKM+98] I-Min A. Chen, Anthony S. Kosky, Victor M. Markowitz, Christoph Sensen, Ernest Szeto, and Thodoros Topaloglou. Advanced query mechanisms for biological databases. In *6th Int. Conf. on Intelligent Systems for Molecular Biology*, pages 43–51, Montreal, Canada, 1998. AAAI Press, Menlo Park.

[CM77] Ashok K. Chandra and Philip M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *9th ACM Symposium on Theory of Computing*, pages 77–90, 1977.

[DG97] Oliver M. Duschka and Michael R. Genesereth. Answering recursive queries using views. In *16th ACM Symposium on Principles of Database Systems*, Tuscon, Arizona, 1997.

[DOTW97] Susan Davidson, G. Christian Overton, Val Tannen, and Limsoon Wong. Biokleisli: a digital library for biomedical researchers. *Int. Journal on Digital Libraries*, 1:36–53, 1997.

[FKL97] Daniela Florescu, Daphne Koller, and Alon Levy. Using probabilistic information in data integration. In *Proc. of the 23rd VLDB Conference*, Athens, Greece, 1997.

[GGMT94] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. The effectiveness of GlOSS for the text database recovery problem. In *Proc. of the ACM SIGMOD Conference*, 1994.

[HY81] Ching-Lai Hwang and Kwangsun Yoon. *Multiple Attribute Decision Making.* Number 186 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin/Heidelberg/New York, 1981.

[Inf98] InfoBiogen. Dbcat - the public catalog of databases. WWW Page http://www.infobiogen.fr/services/dbcat, Infobiogen, France, 1998.

[Jur74] J.M. Juran, editor. *Quality Control Handbook.* McGraw-Hill, New York, 3rd edition, 1974.

[KDG96] Graham J. L. Kemp, Joel Dupont, and Peter M. D. Gray. Using the functional data model to integrate distributed biological data sources. In *8th Int. Conf. on Scientific and Statistical Database Management*, pages 176 – 185, Stockholm, Sweden, 1996.

[LCPL98] Stan Letovsky, R.W. Cottingham, Christopher J. Porter, and Peter W.D. Li. Gdb: the human genome database. *Nucleic Acids Research*, 26(1):94–99, 1998.

[Les98] Ulf Leser. Combining heterogeneous data sources through query correspondence assertions. In *1st Workshop on Web Information and Data Management, in conjunction with CIKM'98*, pages 29–32, Washington, D.C., 1998.

[Les99] Ulf Leser. Designing a global information resource for molecular biology. In *8th GI Fachtagung: Datenbanksysteme in Buero, Technik und Wissenschaft*, Freiburg, Germany, 1999. to appear.

[LLRC98] Ulf Leser, Hans Lehrach, and Hugues Roest Crollius. Issues in developing integrated genomic databases and application to the human X chromosome. *Bioinformatics*, 14(7):583–690, 1998.

[LMSS95] Alon Y. Levy, Alberto O. Mendelzon, Yehoshua Sagiv, and Divesh Srivastava. Answering queries using views. In *14th ACM Symposium on Principles of Database Systems*, pages 95–104, San Jose, CA, 1995.

[LRO96] Alon Y. Levy, Anand Rajaraman, and Joann J. Ordille. Querying heterogeneous information sources using source descriptions. In *22nd Conference on Very Large Databases*, pages 251–262, Bombay, India, 1996.

[LWG+98] Ulf Leser, Robert Wagner, Andrei Grigoriev, Hans Lehrach, and Hugues Roest Crollius. IXDB, an X chromosome integrated database. *Nucleic Acids Research*, 26(1):108–111, 1998.

[McK94] V.A. McKusick. *Mendelian Inheritance in Man. Catalogs of Human Genes and Genetic Disorders.* Johns Hopkins University Press, Baltimore, 11th edition, 1994.

[Nau98] Felix Naumann. Data fusion and data quality. In *Proc. of the New Techniques & Technologies for Statistics Seminar (NTTS)*, Sorrento, Italy, 1998.

[NFS98] Felix Naumann, Johann Christoph Freytag, and Myra Spiliopoulou. Quality-driven source selection using Data Envelopment Analysis. In *Proc. of the 3rd Conference on Information Quality (IQ)*, Cambridge, MA, 1998.

[Orr98] Ken Orr. Data quality and systems theory. *Communications of the ACM*, February 1998.

[Pie98] Elizabeth M. Pierce. Enumerating data errors - a survey of the counting literature. In *Proc. of the 3rd Conference on Information Quality*, Cambridge, MA, 1998.

[Red98] Thomas C. Redman. The impact of poor data quality in the typical enterprise. *Communications of the ACM*, 41(2):79–82, 1998.

[Rob95] Robert J. Robbins. Information infrastructure for the human genome project. *IEEE Engineering in Medicine and Biology*, 14(6):746–759, 1995.

[SL90] Amit Sheth and James A. Larson. Federated database systems for managing distributed, heterogeneous and autonomous databases. *ACM Computing Survey*, 22(3):183–236, 1990.

[SMS+98] Guenter Stoesser, Mary Ann Moseley, Joanne Sleep, Michael McGowran, Maria Garcia-Pastor, and Peter Sterk. The embl nucleotide sequence database. *Nucleic Acids Research*, 26(1):8–15, 1998.

[TB98] Giri Kumar Tayi and Donald P. Ballou. Examining data quality. *Communications of the ACM*, 41(2):54–57, 1998.

[Ull97] Jeffrey D. Ullman. Information integration using logical views. In *6th Int. Conference on Database Theory; LNCS 1186*, pages 19–40, Delphi, Greece, 1997. LNCS 1186, Springer.

[Wan98] Richard Y. Wang. A product perspective on Total Data Quality Management. *Communications of the ACM*, 41(2):58–65, 1998.

[Wie92] Gio Wiederhold. Mediators in the architecture of future information systems. *IEEE Computer*, 25(3):38–49, 1992.

[WS96] Richard Y. Wang and Diane M. Strong. Beyond accuracy: What data quality means to data consumers. *Journal on Management of Information Systems*, 12, 4:5–34, 1996.