An Approach to Query Cost Modelling in Numeric Databases

Kalervo Järvelin
Department of Library and Information Science, University of Tampere, P.O. Box 607 SF-33101 Tampere, Finland

Numeric online databases (NDBs) have become essential in information retrieval (IR). NDBs differ from traditional bibliographic databases (BDBs) with respect to their content, structural complexity, data manipulation capabilities, and the complexity of the user interfaces and user charging schemes. Recent trends in user charging policy for all online IR have been based on charging either for information actually retrieved from the database or for the costs of processing user queries rather than for the connect-time. However, the viability of such charging schemes depends totally on the user's possibilities of estimating the charges in advance, during the query negotiation phase. Due to the complexity of the modern database management systems (DBMS), users cannot estimate such charges in advance without cost estimation tools. In this article, user charges based on query processing costs in NDBs are considered and the problem of estimating such charges in advance is analyzed. A systematic and general approach with many desirable properties is proposed to query cost estimation. The approach is based on the well-known relational data model (RDM) and the query optimization, cardinality estimation and file design techniques developed in this context. Tools based on the approach are necessary components of query interfaces to NDBs if a mode of charging not based on connect-time is used.

1. Introduction

Numeric databases (NDBs) are rapidly gaining in popularity in information retrieval (IR) and their use is growing faster than that of bibliographic databases (BDBs) [1–7]. NDBs differ in many respects from BDBs [8,9]. Firstly, NDBs contain *data* instead of references. Secondly, their *structure* is more complex than that of BDBs. Queries in NDBs often require *combining* data from several files the organizations of which vary, whereas the standard in BDBs is the inverted file. Thirdly, while *manipulation* of data is essential in NDBs, plain retrieval characterizes BDBs. Fourthly, the user *interfaces* to NDBs, especially the data manipulation languages, are more complex and powerful than those in BDBs. Fifthly, the *charging* schemes in NDBs are also more complex including connect-time, computer processing, storage use, graphics use, off-line printing, administrative overheads, subscription, and telecommunication fares [5].

Recent studies on charging for online IR propose charging the users rather for the information they retrieve or for its retrieval cost than for connect-time [10–14]. In this article these charging principles are referred to as *item-based* charging (i.e., charges are based on the number and type of the retrieved information items) and *processing-based* charging (i.e. charges are based on the cost of retrieving and processing these items).

The purpose of this article is to present an approach to advance estimation of processing-based user charges, brielfly termed *query cost modelling*. The costs of queries depend on their characteristics, the actual database contents and the file structures employed in the database and may vary over a broad range [9]. Query characteristics and database contents determine the volume of data retrieved, i.e., the *cardinalities* of result and intermediate files which depend on data distributions in the database [15–20]. File structures determine the possible access strategies for queries [21–25]. Together with the hardware they also determine access cost. The nonexpert database user cannot be assumed to know the actual distributions of data in a database nor the file structures, nor how such information is used for cardinality estimation or access strategy selection. Therefore automatic cost estimation tools are called for.

In order to be useful and convenient such tools for user charge estimation must meet a number of requirements:

- The tools should *cover* the estimation of user charges whether based on items retrieved, processing or e.g. data communication.
- The tools must also provide *readable and rich enough descriptions of query results* so that users may form a clear idea of the results and modify the query if necessary.
- To cope with often complex and unanticipated queries in NDBs the tools must be *general*. They must allow estimation of the user charges for any well-formed query.

Received October 9, 1985; accepted May 26, 1987.

- Because users formulate their queries, and decide on their execution, on the basis of the charge estimates, the estimates must be reasonably *accurate*.
- The tools must make charge estimation *convenient* for the database user; they must not assume extra knowledge, technical skills nor effort on behalf of the user.
- The tools must be *fast enough* to support interactive query formulation.
- The tools must be *compatible with the query systems* of DBMSs so that they can be made available where they are needed.
- The *costs of using tools* must be *insignificant* with respect to query charges.
- The tools must be *explicitly* and *precisely defined* in terms of *how* estimates are computed and *what* information is used.

A methodology for building user charge estimation tools that meet these requirements has been proposed in [8]. It consists of solutions to three subproblems of user charge estimation, namely cardinality estimation [15], item-based charge estimation [26] and processing-based charge estimation [27–29]. The methodology has been developed in the context of the relational data model (RDM) [30]. In the RDM, queries are expressed in relational algebra (RA) or query languages based on it (e.g., SQL [31,32] and the data are represented as *relations* (flat files) consisting of *tuples* (records).

The *query processing cost* can be defined and measured in many ways. A common measure is the *CPU time* used by the query. It is a relevant measure when the CPU is the critical resource in processing, i.e. when the queries are *CPU-bound* [33]. Query applications are typically *transfer-bound* (or I/O-bound [33]), i.e. due to the relatively simple processing, data transfer is the critical resource [16,23–25,34]. In this article, data transfer cost is used as the measure as only query applications are considered. It will be measured by *data transfer time*, i.e., the time (in milliseconds) required by the transfer of data blocks between the central and auxiliary memories [21,22,33]. The cost of maintaining the database is irrelevant from the query cost modelling point of view and is therefore not considered.

Query cost estimation in databases has been analyzed in the literature on file and database design (either for the RDM or in general, e.g., in [21,22,33,35,36]) and RA query optimization [16,23–25,32,34,37]. Informing database users about query costs has so far not been considered because this literature focuses on business database applications [9]. Therefore the first two of the requirements above (coverage, richness) are not satisfied by file design and query optimization techniques. The file design techniques provide accurate means of single-file search cost estimation but are not suitable as such for multi-file query cost modelling. The requirements on user convenience, speed and cheapness of use are often met by the query optimization techniques but generally not by file design techniques. It is therefore necessary to synthesize the techniques of file design and query optimization.

In summary, the approach to query cost modelling is based on the following assumptions:

- query processing is *transfer-bound;*
- the database maintenance cost per query is zero;
- the database files are *flat* (i.e., nonhierarchical) files and stored on *disks;*
- the records in the files are *uniformly distributed* over the values of their attributes and the attribute values are *independent* and *uniformly distributed* over their ranges (these are standard assumptions in the literature, e.g., [16,20,23,38])
- the queries have *no correlation* (this is also a standard assumption, e.g., [39], and more reasonable in NDBs than in business databases);
- the file system of the DBMS reserves two buffer slots (size one block) for each file and its overflow area (if any) and each of its indices (one permanently for the index root and the other for the rest of the index), as in [22]; this assumption has an effect on the estimated costs but is not a limitation of the approach.

2. The Components of the Query Cost Model

File Descriptions

Three issues must be considered in the development of query cost estimation methods: the kind of information on database use which is necessary for user charge estimation, its representation, and its use. The *types of information* can be categorized into information on user queries, databases and the hw/sw environment [9]. In this article, the description of the database contents and structure will consist of *file descriptions*. The components of file descriptions are file names (**FN**), file cardinalities (**CARD**), descriptions of the attributes of the file (**AD**), descriptions of the functional dependencies among the attributes (**FD**), and descriptions of the file organization (**OD**). Moreover, because RA queries are often multi-phased, the estimated processing costs (**COST**) of subqueries are represented in the file descriptions. An appropriate means of representing the file descriptions is the *n-tuple representation*, which has become popular for solving problems which require precise, explicit and many-sided specifications (e.g. [40–42]). The file descriptions are structured as *six-tuples*. Consider, as an example, the sample file PRODUCTS which is partially depicted in Figure 1.

PRODUCTS (PRODUCT-NO,	TRADEMARK,	TYPE,	MANUF-NO,	QSALES	)
1512	GILLETTE	19500	260011	502000	
1586	GILLETTE	19190	260011	107000	
376203	BRAUN	19190	7005	408000	
95051	BLUE STRATOS	19500	530286	200000	
556556	TABAC	19500	1050	900000	
•					
•					
•					

FIG. 1. The sample file PRODUCTS

The formal file description for the PRODUCTS file is:

P = (PRODUCTS, 250000, **{**(PRODUCT-NO, 7, 250000, int, 1000, 9999999), (TRADEMARK, 20, 200000, char, λ, λ), (TYPE, 5, 17000, int, 100, 50000), (MANUF-NO, 6, 12000, int, 1000, 999999), (QSALES, 7, 20000, int, 0, 9999999)**}**, **{**{PRODUCT-NO} ⟶ {PRODUCT-NO, TRADEMARK, TYPE, MANUF-NO, QSALES}, {TRADEMARK} ⟶ {MANUF-NO}, . . .**}**, **(**indexed-file, (45, λ, {(TRADEMARK, 7813, 3, 200000), (TYPE, 2715, 2, 17000), (PRODUCT-NO, 4483, 3, 250000)})**)**, 0)

The six main components of this file description are:

- **FN**, the *file name*, e.g., PRODUCTS,
- **CARD**, the file *cardinality*, e.g., 250000,
- **AD**, the *attribute description set* between the first pair of bold-face braces,
- **FD**, the *functional dependency set* between the second pair of bold-face braces,
- **OD**, the *organization description* between the following bold-face parentheses,
- **COST**, the processing *cost* parameter, which here has the value 0.

The *attribute description set* describes the attributes of a file. It is a set of attribute descriptions, each of which is a six-tuple with the components:

- the *attribute name*, e.g., PRODUCT-NO,
- the *length* of the field used to store the attribute values (in bytes), e.g., 7,
- the *attribute selectivity* (i.e., the number of different values), e.g., 250000,
- the *domain*, e.g., int (for integers),
- the *lower bound* value, e.g., 1000, and
- the *upper bound* value, e.g., 9999999.

The attribute description for the attribute PRODUCT-NO, for example, is (PRODUCT-NO, 7, 250000, int, 1000, 9999999). That is, product numbers are represented by seven digits, there are 250 000 different product numbers, all integers, and between the values 1 000 and 9 999 999. In the attribute descriptions, the symbol λ denotes unnecessary components, char character strings and int integers as domains. The components lower and upper bound are unnecessary for character string attributes.

The *functional dependency set* (FD-set) represents the functional dependencies [30] as **A** → **B**, where **A** and **B** are sets of attribute names. The attributes named in **A** functionally determine the attributes named in **B**. For example, {PRODUCT-NO} → {PRODUCT-NO, TRADEMARK, TYPE, MANUF-NO, QSALES} represents the functional dependency of the product data on the PRODUCT-NO values. The FD set is a set of such FD representations. Three dots '. . .' in an FD set are used to indicate that the FD set is not given in toto: further FDs that can be derived on the basis of those given (see e.g., [30]).

The organization description **OD** consists of two main components, the first one giving the file organization type (e.g., indexed-file) and the second containing the parameters necessary for describing files of that particular type. In the case of an indexed file, these components are the *record length* (e.g., 45, sum of attribute field lengths), the *sort key* (an attribute name, e.g., λ for no sort order), the *index description set* giving details of the indices to the file. Each index description contains the *name* of the indexed attribute (e.g., TYPE), the *number of leaf blocks* in the index (e.g., 2715), the *height* of the index (e.g., 2), and the *number of accession lists* in the index (e.g., 17000). These are key parameters governing the behavior of tree-structured, exhaustive, multi-level and single-attribute indices which are typical in modern DBMSs [21,22,34–36]. In the sample file description **P**, the index description set {(TRADEMARK, 7813, 3, 200000), (TYPE, 2715, 2, 17000), (PRODUCT-NO, 4483, 3, 250000)} indicates that there are three indices on the attributes TRADEMARK, TYPE and PRODUCT-NO.

Computer System Description

The parameters describing relevant features of the hw/sw environment for query cost modelling can be classified into *block-*, *buffer and sort-* and *disk unit*-related parameters [22,24,25,28,33–35]:

- *Block-related parameters:*
  - -**B**, the data block length (in bytes)
  - -**B′**, the index block length (in bytes)
  - -**llf**, the index leaf block load factor
  - -**nlf**, the index node block load factor
  - -**pl**, the record or block identifier length (in bytes)
- *Buffer-and sort-related parameters:*
  - -**z**, the number of queues used in the sort subroutines
  - -**f**, the number of central memory block slots for data buffering
- *Disk unit-related parameters:*
  - -**s**, the average seek time of the disk unit (in milliseconds)
  - -**r**, the theoretical data transfer rate (in bytes per millisecond)
  - -**nr**, the net data transfer rate for bulk transfer (in bytes per millisecond)
  - -**l**, the track length of the disk unit (in bytes)
  - -**t**, the number of tracks per cylinder in the disk unit
  - -**c**, the number of cylinders per disk unit
  - -**d**, the average rotational delay (in milliseconds)

These parameters are given by the storage parameters (**sp**) which also form an n-tuple, in this case a *fourteen-tuple* (**B**, **B′**, **llf**, **nlf**, **pl**, **z**, **f**, **s**, **r**, **nr**, **l**, **t**, **c**, **d**). The values of these parameters must always be measured from the actual hw/sw environment of the DBMS. The following sample storage parameters are assumed in the examples: *sp′* = (2048, 1024, 0.75, 0.75, 8, 2, 4, 30, 512, 256, 20000, 20, 400, 8). Block access times, for example, can now be computed on the basis of these parameters. The data block access time in random block accesses is **s** + **d** + **B**/**r**, which gives 42 ms on *sp′*. Functions for computing such matters are defined formally in [28].

Distributed database systems may have several different hw/sw environments. These are taken into account by describing their storage parameters separately.

Query Description

Assume that the sample database also contains a file on market information named MARKETS as depicted in Figure 2. Assume that this file has some 50 million records for 250 000 different products, 80 different countries and five different years (1980–1984) and that the file has two indices, one the attribute PRODUCT# and the other on COUNTRY. The file description of the MARKETS file, denoted by **M**, gives this information. (The file description **M** is not given here).

Consider the sample query **Q1**: "Give the product numbers, trademarks, and market shares of after-shave lotions in TAHITI in 1984." (Assumption: the product TYPE of after-shave lotions is 19500). This query is expressed in SQL [31,32] as follows:

```
select   PRODUCT-NO, TRADEMARK,
         MARKET-SHARE
from     PRODUCTS, MARKETS
where    PRODUCTS.TYPE = 19500 and
         MARKETS.COUNTRY = TAHITI and
         MARKETS.YEAR = 1984 and
         PRODUCTS.PRODUCT-NO
          = MARKETS.PRODUCT#
```

This query involves two restrictions, one on PRODUCTS and one on MARKETS, a join of the restriction results on PRODUCT-NO = PRODUCT# and finally a projection on the three requested attributes (Figure 3(a)). It is assumed in the present article that each single RA operation can be implemented by as many *implementation procedures* as there are access strategy alternatives in its execution [24,37]. One possible, while not necessarily the most efficient, implementation strategy for the sample query involves accessing PRODUCTS via the index on TYPE and MARKETS by file scan. These are accomplished by *generalized restriction procedures* [23,24,32]. After the restrictions, their results should be sorted on PRODUCT-NO and PRODUCT#, respectively, to facilitate their joining by a *merge-join* procedure [24,25,30,34,43]. Sorting is done by *sort procedures*. The join result is input to a projection procedure which in this case only needs a single pass through the join result because the key of the operand (PRODUCT-NO) is preserved [24,25,34]. The projection can thus be implemented by an efficient *key-preserving projection procedure* [24].

Queries are represented as RA expressions. The implementation strategies, selected by the optimizer, as sequences of implementation procedures are represented analogously as tree-structure expressions consisting of implementation procedure calls [8].

The result of this simple sample query is partially depicted in Figure 3(b). Before this result is evaluated, i.e., before the query (the procedure sequence) is executed, the query cost model is utilized. Thereby the database user receives a description of the result and a cost estimate before query execution. The user has a chance to judge the result against its costs before execution.

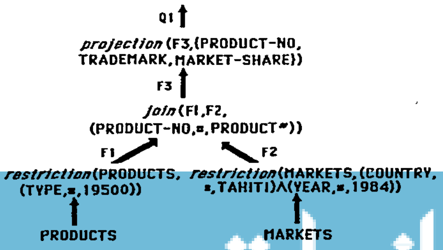
Cost Estimation Models

The *use* of the information necessary for query cost modelling is defined by *estimation models* for each implementation procedure. These estimation models estimate the cardinalities and other characteristics of the results of procedures, including the processing-based charges. Each model consists of precisely defined mathematical functions which construct the components of the result file description on the basis of its operand file description(s), the storage parameters and the characteristics and parameters of the procedure. The result file description is structurally equivalent to the operand file description(s). The structure of the estimation models is explained in [8,9,15,26].

Basically, the query cost model operates as follows. When the database user has given his tentative query in the query language of the DBMS, it is first transformed into the equivalent RA expression and optimized by the query optimizer [23–25,34,37], whereafter the corresponding query cost model expression is constructed. Optimization is necessary in the first phase because it fixes the access

MARKETS (PRODUCT#,	COUNTRY,	YEAR,	MARKET-SHARE)
1512	TAHITI	1984	0,5
1512	CONGO	1984	0,1
1512	CONGO	1983	0,12
95051	TAHITI	1984	0,02
556556	BURMA	1983	0,75
556556	TAHITI	1984	0,42
⋮			

FIG. 2. The sample file MARKETS



Q1
projection(F3,(PRODUCT-NO, TRADEMARK,MARKET-SHARE))
F3
join(F1,F2, (PRODUCT-NO,=,PRODUCT#))
F1
F2
restriction(PRODUCTS, (TYPE,=,19500))
restriction(MARKETS,(COUNTRY, =,TAHITI)∧(YEAR,=,1984))
PRODUCTS
MARKETS

FIG. 3(a). The sample query as a tree structure.

Q1 (PRODUCT-NO,	TRADEMARK,	MARKET-SHARE)
1512	GILLETTE	0,5
95051	BLUE STRATOS	0,02
556556	TABAC	0,42
⋮		

FIG. 3(b). A part of the sample query result strategy of the query and therefore also determines query cost. In the cost model expression, each RA implementation procedure is represented by its estimation model, its parameters by the same parameters in the model and its operand files by their descriptions. Each estimation model constructs the file description of the procedure's result. This is then used in the estimation model of any subsequent procedure.

3. Cost Estimation

Costs of Restrictions

The estimation model for generalized restriction procedures consists of five functions for constructing the components **CARD**, **AD**, **FD**, **OD**, and **COST** of the file descriptions. The functions for the first three components have been defined by Järvelin in [15,44]. They allow complex Boolean predicates in the result cardinality estimation and also handle multiple conditions on one attribute correctly (e.g., TYPE $\geq$ 19000 and TYPE $\leq$ 20000). Their results for the sample query **Q1** are summarized below. The construction of the **OD** and **COST** components is then considered in more detail.

The restriction on the PRODUCTS file in the sample query **Q1** gives records with TYPE value equal to 19500. Assume that the result file has the following new properties when compared to the operand (their evaluation is presented in detail in [15]):

- its cardinality is 15
- the attribute TYPE: its selectivity is 1; its range is from 19500 to 19500
- other attributes: their selectivities are set at 15
- the attribute TYPE becomes functionally dependent on all the other attributes.

The restriction on the MARKETS file produces records concerning TAHITI and the year 1984. Assume that the result has the following new properties when compared to the operand (see [15] for details):

- its cardinality is 125000
- the attribute PRODUCT#: its selectivity is 125000
- the attribute COUNTRY: its selectivity is 1
- the attribute YEAR: its selectivity is 1; its range is from 1984 to 1984
- the attributes COUNTRY and YEAR become functionally dependent on all the other attributes.

Query systems typically *pipeline* the intermediate results of queries if possible [24,25,34]. Therefore the regular organization of the intermediate files is that of pipelines. Two parameters are sufficient to describe the structure of pipelines for query cost modelling: the *record length*, which is derived on the basis of the attribute field lengths, and the name of the *sort key*, if any, of the records. Restriction results are always pipelined. No difficulty is involved in constructing the parameters by the function *r-od*. In the sample query **Q1** the results from the restrictions on PRODUCTS and MARKETS have record lengths 45 and 41, respectively (the same as in the operands). Neither result is sorted because the operands are not sorted, either.

Cost estimation functions for restrictions must be capable of estimating the cost of any access strategy that may be utilized for the operation. In practice, optimizers decide between file scan and index utilization on a very rough basis [23–25,31,32,34,37]. Having the optimizer's decision, the cost of the access strategy can be estimated in detail. In the present approach, the file descriptions and storage parameters convey necessary, detailed parameters for cost estimation. Järvelin has defined formally and generally a function that estimates the cost of evaluating *general* Boolean predicates (excluding negations) in any of six flat file types (including sorted and indexed files) [28]. This function is based on standard file design formulae, in [21,22,33], but extends and generalizes them in an essential way. Due to its modular structure it can easily be customized to the usual query environments. The cost estimation function for restrictions *r-cost* is built on this function [27].

In the sampe query, the PRODUCTS file is accessed via the index on TYPE to obtain the records on after-shave lotions. In the file description **P**, the description of this index is (TYPE, 2715, 2, 17000). This means that, in order to retrieve the record identifiers in the accession list for the TYPE value 19500, the index height (2 levels) must be traversed. This takes two index block accesses (40 ms each). The retrieval of the estimated 15 target records identified by the accession list requires 15 data block accesses (42 ms each). The total cost of this operation is therefore $2 \times 40 + 15 \times 42 = 710$ ms. A scan of the MARKETS file takes a much longer time. The file was assumed to contain 50 million records with record length 41. Because the data block length is 2048 bytes, this file requires 1020409 blocks and its scan cost is about 8.16×10^6 ms (8 ms per block). These estimates are derived automatically.

The estimation model for generalized restriction procedures is a function (called *r-description*) which constructs the file descriptions of the restriction results by organizing the file name Λ and the results by the functions for the components **CARD**, **AD**, **FD**, **OD**, and **COST** as a six-tuple [27]. The file name component of the result file description is denoted by Λ (for a missing name), because naming the intermediate or result file descriptions is irrelevant from the cost modelling viewpoint.

For example, the file description for the result of the restriction on PRODUCTS will be (Λ, 15, {(PRODUCT-NO, 7, 15, int, 1000, 9999999), (TRADEMARK, 20, 15, char, λ, λ), (TYPE, 5, 1, int, 19500, 19500), (MANUF-NO, 6, 15, int, 1000, 999999), (QSALES, 7, 15, int, 0, 9999999)}, {{PRODUCT-NO}$\rightarrow${PRODUCT-NO, TRADEMARK, TYPE, MANUF-NO, QSALES}, {TRADEMARK}$\rightarrow${MANUF-NO}, {PRODUCT-NO} $\rightarrow$ {TYPE}, {TRADEMARK} $\rightarrow$ {TYPE}, {MANUF-NO} $\rightarrow$ {TYPE}, {QSALES} $\rightarrow$ {TYPE}, . . .}, (pipe, (45, λ)), 710). Details are given in [8].

Costs of Sorting

The following procedures in the sample query are sort procedures. Sorting affects only the components **OD** and **COST** in the file descriptions. The result of the sort procedure is a sorted pipe file. The cost of sorting must be added to the **COST** component of the operand in order to reflect all the costs associated with the production of the sort result. Assuming the common *z-way-sort-merge* method [34,45], the function by Blasgen and Eswaran [34] has been modified so as to give the cost in milliseconds [28]. The costs of the sort procedures in the sample query are zero milliseconds for the small intermediate file originating from the PRODUCTS file, and 1.18×10^6 ms for the large intermediate file originating from the MARKETS file. The total costs associated with these files are now 71 ms and 9.34×10^6 ms, respectively. The estimation model for the sort procedures [27] constructs the file descriptions of the sort results. The **OD** components give the new sort keys and the **COST** components the new costs. Otherwise the result file descriptions remain the same as the operand file descriptions.

Costs of Joins

The estimation model for merging equi-join procedures also consists of five functions for constructing the components **CARD**, **AD**, **FD**, **OD**, and **COST**. The functions for the **CARD**, **AD**, **FD**-components have been defined in [15,44]. They take the join attribute ranges and selectivities, which need not be equal, into account. Their results for the sample query **Q1** are summarized below. The functions for the **OD** and **COST**-components are then considered in more detail.

The join on the two intermediate files in the sample query **Q1** gives records with equal values of PRODUCT-NO and PRODUCT#. Assume that the result has the following new properties when compared to the operands (see [15] for details):

- its cardinality is 15
- the attribute PRODUCT#: its selectivity is 15
- the attribute MARKET-SHARE: its selectivity is 15
- the attributes PRODUCT-NO and PRODUCT# become functionally dependent on each other.

Also the results of merging equi-joins are pipelined. Therefore the function for organization description simply notifies the file organization type (pipe-file), the record length and the sort-key. In the sample query **Q1** the two latter are 86 (i.e., the sum of the operand record lengths) and PRODUCT-NO. The **OD** component is thus (pipe-file, (86, PRODUCT-NO)).

The merging equi-join procedure is efficient if the join attribute selectivities are good: in such a case one pass through the operand files if sufficient [43]. In the worst case, i.e., with poor join attribute selectivities, the cost of the merge-join procedure is considerably higher than the optimal [29]. Järvelin provides a cost function for the procedure which takes the effect of the join attribute selectivities into account [29]. The **COST** component in the result file description for this procedure is always the sum of the **COST** components of the operand file descriptions and the operand merge cost [27].

In the join of the sample query **Q1**, the join attribute selectivities are good. Because the operands are pipe files, their merging is performed in the central memory and at no extra cost. In the sample case, therefore, the **COST** component of the result file description is about 9.34×10^6 ms, i.e., about 2 h 40 min, most of which is due to the high scan cost of the MARKETS file.

The cost estimation model for the merging equi-join, the function *m-j-description* [27], organizes the derived components **CARD**, **AD**, **FD**, **OD**, and **COST** and the file name Λ as a six-tuple to produce the result file description. The formal file description for the sample join is bypassed.

Costs of Projection

The estimation model for key-preserving projection procedures consists of functions for constructing the components **CARD**, **AD**, **FD**, **OD**, and **COST** of the result file description. The functions for the components **CARD**, **AD**, and **FD** have been defined in [15,44]. They contribute the use of functional dependencies as a novel means in cardinality estimation. Their results for the sample query are summarized first. The remaining functions are then considered.

The projection on the join result in the sample query **Q1** produces records containing values for the requested attributes PRODUCT-NO, TRADEMARK, and MARKET-SHARE. Assume that the result has the following new properties: its cardinality is 15, the attribute descriptions are the same as in the operand, and the attribute PRODUCT-NO is the key of the result (see [15] for details).

The projection results are also pipelined whenever possible. The function for constructing the organization description for the result recognizes the operand sort order, if any, and computes the record length as the sum of the attribute field lengths. In the sample query **Q1** the **OD** component for the projection result is (pipe-file, (32, PRODUCT-NO)).

Cost estimation for key-preserving projection procedures must take into account the file structures the operands may have. The operand files are always scanned, possibly utilizing the operand sort order, if any. Järvelin has analyzed the scan costs of files with six file structures and provides a general formula for their estimation [28]. The cost estimation function for projections *k-p-cost* is built on this function [27]. It sums the operand file scan cost and the operand's **COST** component.

Although the projection procedure in the sample query **Q1** scans the operand, no costs are incurred because the operand is pipelined. Therefore the **COST** component of the result file description of the sample query obtains the same value as in the operand file description, i.e. about 9.34×10^6 ms (about 2 h 40 min).

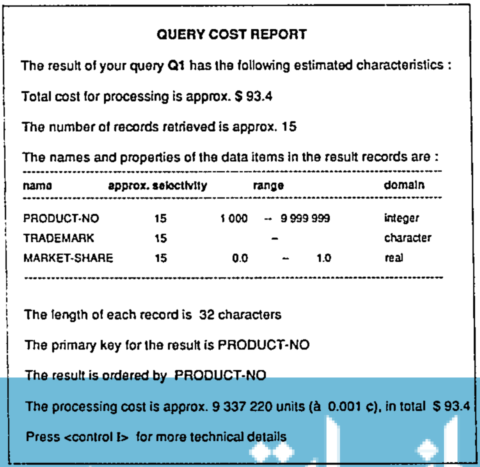
The cost estimation model, the function *k-p-description* [27], for the key-preserving projection procedures constructs the result file descriptions by organizing the file name Λ and the components **CARD**, **AD**, **FD**, **OD**, and **COST** as a six-tuple. The presentation of the formal result file description is bypassed.

4. Cost Reporting

The Query Cost Report

The user charge estimation methodology, and the tools based on it, produce as their results formal file descriptions. As such they are quite unreadable for nonexperts. However, the file descriptions can be interpreted in concrete and easily intelligible terms as *query cost reports*, which summarize essential features of query results. The sample report in Figure 4 is based on the cost estimation result of the sample query **Q1**. The data in the report are derived automatically on the basis of the query (expressed in the SQL), the sample file descriptions **P** and **M** and the storage parameters *sp'* [8,29]. The coefficient for scaling the processing-based charge estimate (in ms) into dollars is imaginary (any economically justifiable coefficient can be used). The technical details option would report the field lengths of the records, their functional dependencies and the file organization description in an analogous way.

Query cost reports provide database users with rich and understandable descriptions of query results and their costs. The reports contain essential items of information for database users, such as the query processing cost, the number of records retrieved, and the selectivities, ranges and domains of the retrieved attributes. Users therefore know *how much* they have to pay for *what* information. On the basis of the attribute descriptions they can judge rationally, *what information might be discarded* as unnecessary, or *what additional information should be requested*.



```
QUERY COST REPORT

The result of your query Q1 has the following estimated characteristics :

Total cost for processing is approx. $ 93.4

The number of records retrieved is approx. 15

The names and properties of the data items in the result records are :
---------------------------------------------------------------------------
name            approx. selectivity       range                 domain
---------------------------------------------------------------------------
PRODUCT-NO             15          1 000   --  9 999 999       integer
TRADEMARK              15                  -                   character
MARKET-SHARE           15            0.0   --        1.0       real
---------------------------------------------------------------------------

The length of each record is  32 characters

The primary key for the result is PRODUCT-NO

The result is ordered by  PRODUCT-NO

The processing cost is approx. 9 337 220 units (à  0.001 ¢), in total  $ 93.4

Press <control I>  for more technical details
```

FIG. 4. The query cost report for the sample query **Q1**

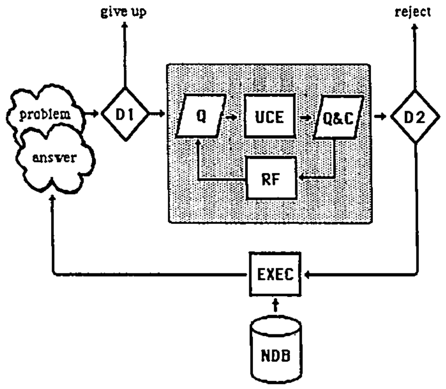
Moreover, the provision of the reports does not require extra effort on behalf of the user: all one has to give is the query expression. Query cost reports should therefore prove useful in query negotiation.

Use of Query Cost Reports

The position and use of user charge estimation tools and query cost reports is as in Figure 5. Having an information need, the user decides whether database use might be worthwhile or whether he should give up the effort. If some NDB seems promising, the user must formulate an appropriate query. For each tentative query formulation, one receives a query cost report which can be used for broadening or narrowing the query. If the expected results turn out to be too expensive and further modification of the query seems fruitful, the user may choose to do so. If not, the whole query may be rejected. If the query seems worthwhile, it can be executed. [9]

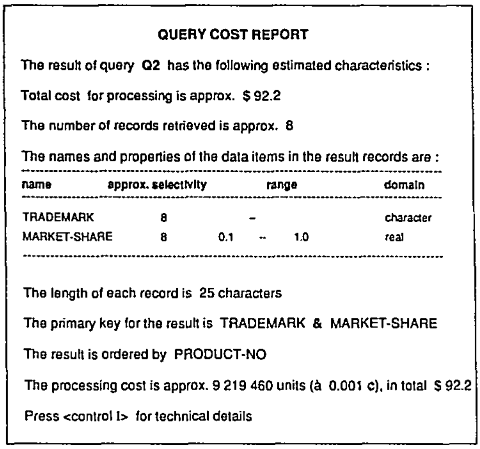
Assume, for example, that the user considers the sample query as too expensive and notices that in fact the user is interested in products with PRODUCT-NO value exceeding 5000000 and that the PRODUCT-NO information is actually unnecessary; moreover, only products with MARKET-SHAREs above 10% are of interest. The user may well expect that by limiting the scope of the query in such a way might reduce the costs. The modified query (**Q2**) is expressed as follows:

```
select   TRADEMARK, MARKET-SHARE
from     PRODUCTS, MARKETS
where    PRODUCTS.TYPE = 19500 and
         PRODUCTS.PRODUCT-NO ≥ 5 000 000 and
         MARKETS.COUNTRY = TAHITI and
```



LEGEND:			
D1	decision to seek information	Q	query candidate
UCE	user charge estimation	Q&C	query candidate and cost report
RF	query reformulation		
D2	decision to execute the query	NDB	the numeric database
EXEC	query execution		

FIG. 5. User charge estimation tools and query cost reports in the use of NDBs [9].



QUERY COST REPORT

The result of query Q2 has the following estimated characteristics :

Total cost for processing is approx. $ 92.2

The number of records retrieved is approx. 8

The names and properties of the data items in the result records are :

name	approx. selectivity	range	domain
TRADEMARK	8	-	character
MARKET-SHARE	8	0.1 -- 1.0	real

The length of each record is 25 characters

The primary key for the result is TRADEMARK & MARKET-SHARE

The result is ordered by PRODUCT-NO

The processing cost is approx. 9 219 460 units (à 0.001 c), in total $ 92.2

Press <control I> for technical details

FIG. 6. The query cost report for the sample query **Q2**

MARKETS.YEAR = 1984 **and**
MARKETS.MARKET-SHARE ≥ 0.1 **and**
PRODUCTS.PRODUCT-NO
= MARKETS.PRODUCT#

The query cost report for the modified query is illustrated in Figure 6. It indicates that the processing costs have remained essentially the same. This is due to the optimizer's decision to scan the MARKETS file, which corresponds to nearly 90% of the query costs and which is here assumed not to be affected by the modification. The user might now prefer the former query formulation, which provides much more information at only slightly increased cost.

5. Discussion

It has been argued that modern charging bases, although economically justifiable, have the consequence that database users are unable to estimate their charges, which may vary over a considerable range. It is therefore necessary to develop tools for user charge estimation. The benefits of processing-based charging in NDBs cannot be achieved without such means.

An approach for estimating processing-based user charges, briefly query cost modelling, has been proposed in this article. A set of requirements to be met by such an approach was given. The status of the proposed methodology in the light of these requirements is now considered.

Järvelin has shown that the proposed approach is suitable not only for query cost modelling but also for query cardinality estimation and item-based charge estimation [15,26]. The modularity of the approach makes it hospitable to further extensions. For example, by incorporating communication network descriptions the approach can be extended to distributed database environments where data communication costs are also estimated. Therefore the proposed approach covers the user charge estimation on varying charging bases.

Query cost reports, generated on the basis of the results provided by the approach, show clearly that the results of cost estimation can be reported to the user in understandable terms. Essential information for user judgement during query negotiation is provided. Moreover, thorough technical descriptions are available upon request. Tools based on the proposed approach are inherently convenient in the use because the approach requires, on the part of the users, only their query formulations in the query language.

The file descriptions are more thorough than their counterparts in previous approaches. They bring together and present in one framework and in a systematic way essential parameters for query cost modelling from file design, query optimization and cardinality estimation studies. Absence of any component would lead to limitations and/or growing inaccuracy of the methods. To give some examples:

- Yao gives, for *query optimization* purposes, index names and their selectivities but not index width or height, nor the value ranges of their base attributes; the selectivities and ranges of unindexed attributes are not described at all, nor the FDs [25]. The equivalents of the **AD** and **FD** components are not used by Blasgen & Eswaran [34] and Hall [37]. Astrahan & Chamberlin [32], Hall [37], and Smith & Chang [24] give minimal or no quantitative data on database files. Cardinality estimation, index access cost estimation etc. are therefore inaccurate in such approaches. This can be tolerated in query optimization for relatively simple queries in business databases but leads to very inaccurate cost estimates.
- Average values for several parameters are often used in *file design*, e.g., average record length, the number of separate attributes and their average selectivity [39,46]. By neglecting individual attribute selectivities and ranges correct cardinality estimates cannot be computed. Detailed formulae for estimating access times for various file structures are often provided [21,22,33,35,47]. However, almost all cardinality estimation-related parameter values are premises (givens) in the formulae and therefore cannot be estimated within the approaches. As a consequence such approaches are insufficient for query cost modelling without the extensions proposed in this article.
- The approaches to *cardinality estimation*, per se, neglect file organization description. For example, Bernstein et al. [16], Christodoulakis [18,19], Merret & Otoo [48], Piatetsky-Shapiro & Connell [17] and Yu & Lin [20], use rough equivalents of **CARD**, and **AD**, but no equivalents of the components **FD**, **OD**, or **COST**. Query cost analysis on such a basis is not possible. The present approach provides contributions even in cardinality estimation [15].
- The **FD** component is a major extension to the previous approaches. Thorough representation of FDs is necessary for reliable estimation of projection cardinalities [15,30,49] and for access strategy selection for projections [24,25]. However, they have so far been

used only in a very limited sense. Järvelin has analyzed the behavior of complete FD sets in RA operations [42,50]. In this article these results are applied to query cost modelling.

This richness of description supports the generality, usability, accuracy, and flexibility of the proposed approach. Although no large-scale experiment with the approach has been conducted, the proposed approach improves the accuracy of cost estimates because (1) it is based on systematic coordination and application of standard techniques in three relevant areas in the literature and (2) several improvements e.g. in cardinality estimation have been pointed out [15].

The representation of file descriptions as n-tuples supports precise, formal definition of the whole approach. It guarantees general treatment of all cost estimation cases instead of a combination of sample cases. This is not achieved by table-based representations [16,20] nor by assuming implicit information (e.g., that the values of essential parameters are known [21,22,33,35,47]).

All the functions used in the estimation models have been defined *precisely* [15,27–29,44]. This means that their coding into query cost modelling software is fairly straightforward and that such software can be incorporated into database query systems. Query cost modelling tools certainly consume some of the resources of the DBMSs. This is most often proportional to the resource consumption of the queries being evaluated. A further issue to be considered in cost modelling tool development is the user charge for using the tool. This could be a fixed charge or it could be based on the tool connect-time. However, a charge for using the tool is obviously acceptable when weighed against the uncertainty it removes.

The proposed approach is applicable in the framework of the relational data model. In this framework it specifies, exactly and in detail, how the cost estimation tools should be constructed. For other data models and DBMS architectures it provides useful and well-founded suggestions concerning the types of representations and methods required for query cost modelling and other problems related to user charge estimation. Although file descriptions and functions for constructing them must be modified to fit other data models, the architecture of the approach remains appropriate.

The present approach is limited to query applications. It does not support cost estimation for the user of e.g. statistical or econometric models which often are CPU-bound and may therefore require extensions to the present approach. The vendor has to cover the database *maintenance* cost by the query charges (e.g., as an overhead in the coefficient transforming the query cost into monetary terms).

The *results* provided by the proposed approach are necessary prerequisites for the viability of processing-based charging methods. Without information on query output, the users cannot compare the query costs and utility. In many cases this would hinder database use. Those libraries and information services which charge their customers for the services may encounter difficulties in 'selling' search services to customers if the costs cannot be predicted. Moreover tight budgets do not foster risk-taking. [12]

In addition to query formulation, the results provided by the approach are useful in vendor or database selection if, as is often the case, the requested data are available at several sources. Some vendors may not like the possibility that users may find some queries too expensive in advance, and, as a consequence, may not want to execute them. In such situations, the vendor would lose income. On the other hand, much more income will be lost, and many dissatisfied users will turn up if the user cannot have an advance estimate of the query costs.

6. Conclusions

Recent studies on charging for online information retrieval emphasize the importance of charging users for the information they actually retrieve from the database, and their retrieval cost, and not for their connect-time to the online hosts. The foreseeable developments in the microcomputer, mass storage, and data communication technologies will lead to a new generation of online information systems where item- and processing-based charging is the standard [51]. In this article it has been argued that such charging methods are viable in NDBs only if users have the possibility of estimating the query costs in advance. This is due to the characteristics of modern DBMSs, the variability of queries and their results, and the necessarily nontechnical view the users have of the database. Therefore it is necessary to develop query cost estimation tools to be included in the query interfaces of public online NDBs. Without them query negotiaiton has no firm and rational basis.

A general approach to query cost modelling in NDBs is proposed here. The approach is based on (1) extending standard techniques used in query cardinality estimation, file design, and query optimization in the context of the RDM, (2) using file descriptions to describe and convey the characteristics of the operand, intermediate and result files of queries, and (3) developing estimation models for the implementation procedures of RA operations. The proposed approach contains all components, functions and relationships among them which are necessary for developing automatic query cost estimation tools. It provides the possibility of specifying these tools *generally, exactly,* and *systematically*. A number of requirements to be met by query cost modelling tools are put forward. The proposed approach seems to meet these requirements well. It is obviously straightforward to incorporate even more user charging items in the approach, e.g., the estimation of telecommunication costs. Tools providing such estimates are required in the user interfaces if user charging is based on such items. They can be used for the benefit of the database user, supplier, and vendor.

References

1. *Directory of Online Databases*. Santa Monica, CA: Cuadra Associates; 1984.

Online Numeric Databases," *Business Information Reivew*. 1(1):
2. *Directory of Online Databases 8(1)*. New York, NY: Cuadra/Elsevier; 1987.
3. *EUSIDIC Database Guide 1983*. Learned Information, Oxford; 1983.
4. Chen, C.-C. Hernon, P., Eds. *Numeric Databases*. Norwood, NJ: Ablex; 1984.
5. Foster, A. "Business Information from Databanks: The Potential of 38–45; 1984.
6. Heim, K. M., Ed. "Data Libraries for the Social Sciences." *Library Trends Special Issue*. 30(3); 1982.
7. Rumble, J. R., Jr.; Hampel, V. E., Eds. *Database Management in Science and Technology: A CODATA Sourcebook on the Use of Computers in Data Activities*. Amsterdam: North-Holland; 1984.
8. Järvelin, K. "User Charge Estimation in Numeric Online Databases: A Methodology." PhD. Thesis, Tampere, Finland: University of Tampere, Acta Universitatis Tamperensis, Ser. A, Vol. 212, 1986.
9. Järvelin, K. "A Methodology for User Charge Estimation in Numeric Online Databases," *Journal of Information Science*. Part I: 14(1): 3–16; Part II: 14(2): 77–92.
10. Aitchison, T. M. "Online and the Database Producer," *Journal of Information Science*. 9(2): 75–80; 1984.
11. Dunn, R. G.; Boyle, H. F. "Online Searching: An Analysis of Marketing Issues," *Information Services & Use*. 4: 147–154; 1984.
12. Hull, D. "Marketing and Pricing of Full-text End-user Services," *Information Services & Use*. 4: 167–170; 1984.
13. Hunter, J. A. "What Price Information," *Information Services & Use*. 4:217–223; 1984.
14. "International Comparative Price Guide to Databases," *Online Review*. 9(1): 77–84; 1984.
15. Järvelin, K. "Cardinality Estimation in Numeric On-line Databases," *Information Processing and Management*. 22(6): 523–548.
16. Bernstein, P.; Wong, E.; Reeve, C.; Rothnie, J. "Query Processing in a System for Distributed Databases (SDD-1)," *ACM TODS*. 6(4): 602–625; 1981.
17. Piatetsky-Shapiro, G.; Connell, C. "Accurate Estimation of the Number of Tuples Satisfying a Condition." In: *Proceedings of the ACM SIGMOD Conference, Boston, MA, June 18–21, 1984:* 256–276.
18. Christodoulakis, S. "Estimating Record Selectivities," *Information Systems*. 8(2): 105–115; 1983.
19. Christodoulakis, S. "Estimating Block Transfers and Join Sizes," *ACM SIGMOD Record*. 13(4): 40–54; 1983.
20. Yu, C. T.; Lin, Y. C. "Some Estimation Problems in Distributed Query Processing," In: *Proceedings of the IEEE Distributed Computing Systems Conference, 1982:* 13–19.
21. Teorey, T. J. Fry, J. R. *Design of Database Structures*. Englewood Cliffs, NJ: Prentice-Hall; 1982.
22. Wiederhold, G. *Database Design*. New York, NY: McGraw-Hill; 1977.
23. Selinger, P. G. Astrahan, M. M., Chamberlin, D. D., Lorie, R. A., and Price, T. G., "Access path selection in a relational database management system." In: Bernstein, PA. ed. *Proceedings of the ACM SIGMOD Conference*, Boston, MA, May 30–June 1, 1979: 23–34.
24. Smith, J. M.; Chang, P. "Optimizing the Performance of a Relational Algebra Database Interface," *Communications of the ACM*. 18(10): 568–579; 1975.
25. Yao, S. B. "Optimization of Query Evaluation Algorithms," *ACM TODS*. 4(2): 133–155; 1979.
26. Järvelin, K. "A Straightforward Method for Advance Estimation of User Charges for Information in Numeric Databases," *Journal of Documentation*. 42(2): 65–83; 1986.
27. Järvelin, K. "An Approach to Query Cost Modelling in Numeric Databases." (Tampere, Finland: University of Tampere, Dept. of Library and Information Science, Rep. 22, 1986).
28. Järvelin, K. "A systematic approach to modelling the costs of flat files." (Thesis for the degree of Licentiate in Philosophy. Tampere, Finland: University of Tampere, Dept. of Mathematical Sciences, Rep. A152, 1985).
29. Järvelin, K. "A Systematic Approach to Query Cost Modelling," In: Kangassalo, H. ed. *Information Modelling and Data Base Management. Lecture Notes in Computer Science*. Berlin: Springer-Verlag, 1988 (in press).
30. Ullman, J. D. *Principles of Database Systems*. London: Pitman; 1980.
31. Astrahan, M. M. et al. "System R: Relational Approach to Database Management," *ACM TODS*. 1(2): 97–137; 1976.
32. Astrahan, M. M.; Chamberlin, D. D. "Implementation of a Structured English Query Language," *Communications of the ACM*. 18(10): 580–588; 1975.
33. Hanson, O. *Design of Computer Data Files*. London: Pitman; 1982.
34. Blasgen, M. W.; Eswaran, K. P. "Storage and access in relational data bases," *IBM Systems Journal*. 16(4): 363–377; 1977.
35. Chan, A. Y. "Index Selection in a Self-Adaptive Relational Data Base Management System." Cambridge, MA: MIT, Lab. for Computer Science, Report MIT/LCS/TR-166; 1976.
36. Schkolnick, M.; Tiberio, P. "Considerations in Developing a Design Tool for a Relational DBMS" in: *Proc. IEEE Computer Society's 3rd International Computer Software & Applications Conference*, Chicago, Ill., Nov. 6–8, 1979: 228–235.
37. Hall, P. "Optimization of a Single Relational Expression in a Relational Data Base System," *IBM Journal of Research and Development*. 20(3): 244–257; 1976.
38. Gelenbe, E. and Gardy, D. "On the Size of Projections: I." *Information Processing Letters*. 14(1): 18–21; 1982.
39. Yao, S. B.; Merten, A. G. "Selection of File Organization Using an Analytic Model" In: *Proceedings of the First VLDB Conference*, Sept. 1975, pp. 255–267.
40. Ausiello, G.; Batini, C.; Moscarini, M. "On the Equivalence Among Data Base Schemata" In: *Proceedings of the International Conference on Data Bases*, Aberdeen, Scotland, 2–4 July, 1980. London: Heyden, 1980: 34–46.
41. Niemi, T. "A Seven-tuple Representation for Hierarchical Data Structures," *Information Systems*. 8(3): 152–157; 1983.
42. Niemi, T.; Järvelin, K. "A Straightforward Formalization of the Relational Model," *Information Systems*. 10(1): 65–76; 1985.
43. Merret, T. H. "Why Sort-Merge Gives the Best Implementation of the Natural Join," *ACM SIGMOD Record*. 13(2): 39–51; 1983.
44. Järvelin, K. "Cardinalities and Attribute Descriptions of Result Relations of Relational Algebra Operations." 2nd ed. Tampere, Finland: University of Tampere, Dept. of Mathematical Sciences, Report A134, 1985.
45. Knuth, D. E. *The Art of Computer Programming, Vol. 3, Sorting and Searching*. Reading, MA: Addison-Wesley; 1973.
46. Yao, S. B. "An Attribute Based Model for Database Access Cost Analysis," *ACM TODS*. 2(1): 45–67; 1977.
47. Schkolnick, M. "Optimizing Partial Inversions for Files," San Jose, CA: IBM Research Labatory, Rep. RJ 1477 (#22576), 1974.
48. Merret, T. H.; Otoo, E. "Distribution Models of Relations" In: *Proceedings of the 5th International Conference on Very Large Data Bases*, Rio de Janeiro, Brazil, 1979: 418–425.
49. Gardy, D. and Puech, C. "On the Size of Projections: A Generating Function Approach," *Information Systems*. 9(3/4): 231–235; 1984.
50. Järvelin, K. "Finding functional dependencies for intermediate relations of relational algebra expressions." In: Kangassalo, H. ed. *Proceedings of the First Scandinavian Research Seminar on Information Modeling and Data Base Management*. Acta Universitatis Tamperensis ser. B 17. Tampere, Finland: University of Tampere, 1982: 407–441.
51. Fox, C. "Future Generation Information Systems," *Journal of the American Society for Information Science*. 37(4): 215–219; 1986.