Quaestiones Informaticae

Vol. 2 No. 2 May, 1983

Quaestiones Informaticae

An official publication of the Computer Society of South Africa
'n Amptelike tydskrif van die Rekenaarvereniging van Suid-Afrika

Editor: Prof G. Wiechers
Department of Computer Science and Information Systems
University of South Africa
P.O. Box 392, Pretoria 0001

Subscriptions

Annual subscriptions are as follows:

	SA	US	UK
Individuals	R6	$7	£3,0
Institutions	R12	$14	£6,0

Circulation manager
Mr E Anderssen
Department of Computer Science
Rand Afrikaanse Universiteit
P O Box 524
Johannesburg 2000
Tel.: (011) 726-5000

Quaestiones Informaticae is prepared for publication by Thomson Publications South Africa (Pty) Ltd for the Computer Society of South Africa.

NOTE FROM THE EDITOR

Three points must be made by way of introduction to the second issue of Volume 2 of Quaestiones Informaticae.

Firstly, an apology is in order for the mistake in the date (November 1983 instead of 1982) at the foot of my note introducing the preceding issue. Lacking the services of a professional proof reader, printing errors are bound to show up from time to time, but it is hoped that their number will be kept to a minimum!

Secondly, it is a pleasure to announce that this journal will not only serve to publish papers of a scientific or technical nature on computing matters under the auspices of the Computer Society of South Africa. An agreement has been reached to share the facilities of Quaestiones Informaticae between the CSSA and SAICS, the South African Institute of Computer Scientists. Henceforth this journal will also be used to publish the Transactions of this Institute. This implies certain changes to the cover pages which will be implemented in future issues. I shall continue to serve as editor, but on behalf of SAICS Prof R. J. van den Heever will share some of my duties and act as co-editor.

Finally Mr Edwin Anderssen, of Rand Afrikaanse Universiteit, has agreed to serve as circulation manager for Quaestiones Informaticae. I am grateful indeed that he is willing to serve the journal in this capacity, and look forward to a long period of fruitful cooperation.

G WIECHERS

May, 1983

Restructuring of the Conceptual Schema to produce DBMS Schemata

S. Wulf
Comcon (Pty) Limited, Sandton

Abstract
An overall methodology for database design is presented. This includes creation of a data dictionary and relational analysis. An algorithm is presented to create a conceptual schema or logical database design from the set of third normal form relations. The primary emphasis of this paper is to present algorithms for restructuring the conceptual schema to create first pass designs (FPD) for particular DBMS. These FPDs can then be used as the basis for iterative physical database design and performance predictions. Examples are given for creating FPDs for TOTAL and IMS. These can be extended for any other DBMS. Indications are given for future research in extending the methodology.

1. INTRODUCTION

In practice, database design has been found to be complex and difficult. The resulting success of the database is a function of the knowledge of the initial information requirements, the robustness of the design and the subsequent performance of the database. It has been pointed out [1,2,3] that the overall design process should be structured and formalised. The following iterative phases have been proposed [4] to cover the database design activity from gathering of information requirements to physical implementation of the database:

1 Information requirements specification
2 Creation of the data dictionary
3 Relational Analysis
4 Creation of the conceptual schema
5 Structured analysis and local views
6 First pass DBMS design
7 Performance modelling
8 Physical design

In this paper we review briefly the process of relational analysis culminating in the creation of a refined conceptual schema. The conceptual schema is a network diagram with nodes corresponding in general to entities and the links between them to the relationships between these entities. An entity is a person, location or object of interest to the organisation being modelled. It is clearly a logical model — that is, independent of any particular hardware and software implementation constraints.

This paper presents algorithms for specifying operations upon the conceptual schema to modify it to conform to the data structuring constraints of certain target database management systems (DBMS). The objective is to create a DBMS schema [5] as close in flexibility to the conceptual schema as possible. We call this the first pass design (FPD). This design serves as the starting point for iterative physical design.

This entails modelling the proposed database system using queueing network techniques based on algorithms developed in [6] and [7] and modifying the current DBMS schema is arrived at which satisfies minimal performance constraints of the application. Physical design is outside the scope of this paper. In Section 2, a brief review of relational analysis is presented. In Section 3, the production of the conceptual schema is discussed. This serves as the source schema for FPD. Data restructuring algorithms for FPD are presented in Section 4 for two major DBMS — IMS [8] and TOTAL [9]. A sample conceptual schema is then used in Section 5 to prepare a FPD for both DBMS. Future work is discussed in Section 6 and conclusions are presented in Section 7.

2. RELATIONAL ANALYSIS

The process of information requirements specification culminates in the production of a data dictionary. This is essentially a list of data elements sorted alphabetically by name of the data element together with information about the functional dependencies between pairs of data elements [10]. Relational analysis consists of applying successively the procedure of normalisation [11] on this data to produce a set of third normal form relations. By definition each named relation consists of a set of data elements, one or more of which constitute the primary key. We define a foreign key in a relation to be one or more data elements of that relation which are a primary key of another relation. Following [3] we use the following notation to document a relation:

relation-name
** data-element$_1$
** data-element$_2$
,
,
,
** data-element$_n$
data-element$_{n+1}$
,
,
,
data-element$_m$
* data-element$_{m+1}$
,
,
,
* data-element$_p$

where double-starred data elements comprise the primary key of the relation and single-starred data elements comprise foreign keys. This set of relations is used to produce the conceptual schema.

3. THE CONCEPTUAL SCHEMA

The initial conceptual schema network diagram is created by applying the following rules systematically:

1. Each relation becomes a node with the relation name as node name primary key of the relation as key of the node and data elements of the relation of data elements of the node.
2. If a subset of the primary key of any relation is not the key of any existing node then create a new index node with that subset of the primary key as key of the node.
3. If the key of any node is a subset of the key of another node then draw a link between the two nodes with a single headed arrow (→), pointing into the node with the superset key.
4. If the key of any node is a foreign key of another node then draw a link between the two nodes with a double-headed arrow (→≫) pointing into the node which has the key data elements as a foreign key.
5. If more than one single-headed arrow is incident on a node, we designate one link as the primary link and make all the others double-headed arrows.

The result will be a network diagram consisting of a set of

nodes with a single and double headed arrows joining them.

The procedure of structured analysis [4] culminates in the production of a data flow diagram based on the functional requirements of the proposed application system. From this overall system data flow diagram, a set of linear data flow diagrams such as those described in [12] can be constructed. There is one linear data flow diagram per application transaction (input-transformation-output). Associated with each linear data flow diagram, that portion of the overall conceptual schema which it needs can be derived.

The overall conceptual schema is based on the complete data dictionary and the functional dependencies between data elements without regard to the functional requirements of the application system. It is likely, therefore, that many of the nodes and relationships of the conceptual schema will not be required on the implemented database. Subsets of the conceptual schema used in each individual linear data flow diagram constitute local views or subschemata of the conceptual schema. By consolidating all these views, we can produce a refined conceptual schema. This is the logical model of the database which will be the basis for physical database design. In practice we have found that the initial conceptual schema for an organisation is large (70-100 nodes with 120-150 links) while the refined conceptual schema is smaller and hence more manageable (10-40 nodes with 15-50 links).

4. RESTRUCTURING OPERATIONS FOR FPD

4.1 Hierarchical and non-hierarchical relations

In [13] analysis of the logical and physical structures of databases is discussed. Hierarchical and non-hierarchical data relationships are defined and application of the schema analysis methodology to restructuring is presented. This work has been used as the basis for the techniques presented in this section for deriving the FPD for various DBMS given a particular refined conceptual schema.

Consider a set relation (A,B) where A and B are nodes of the conceptual schema and there exists a link between A and B with the arrow head pointing into B. If the key of B is a subset of the key of A, we call (A,B) a hierarchical relation. A is called hierarchically superior and B is the hierarchically dependent node. Clearly, every relation in the conceptual schema which has a single-headed arrow is a hierarchical relation. If the set relation (A,B) is not hierarchical, we call the relation non-hierarchical. Every relation in the conceptual schema which has a double-headed arrow is non-hierarchical.

Generally, hierarchical relations denote one-to-many relationships between owner nodes A and member nodes B of a set relationship. Non-hierarchical relatins are generally many-to-many. If a node participates as the member of more than one non-hierarchical relation (ie. two or more double-headed arrows point to the node) it in fact contains the "intersection data" or "link data" for logical occurrences of the implicit many-to-many relationship between the two owner nodes. In [13] and in the various representations of the entity-relationship model [14], this data is not represented explicitly as a node but as data associated with the relation. We have found the representation used here to be closer to DBMS schemata and hence easier to use.

4.2 Restructuring algorithms for FPD

The objective of FPD is to manipulate the source conceptual schema network schematic in a programmed way to ensure that the target DBMS schema is as conformable as possible. The conceptual schema structure is a generalised network with possibly mixed hierarchical and non-hierarchical relations while DBMS schema specifications usually place restrictions on the allowed schema structures. For example, TOTAL does not allow a given node to be both an owner and a member. IMS supports only strictly hierarchical "physical" databases but allows these to be linked together to form "logical" databases. We present here algorithms for both TOTAl and IMS to restructure the source conceptual schema to valid FPDs.

Algorithm for TOTAL

1. For all nodes i,
   If source-node (i) = any owner and source-node (i) not = any member
   create target-node (i, 'master') = source-node (i)
2. For all source nodes i, j with i ≠ j
   If source-node (i) = any member of source-node (j) and target-mode (j, 'master') or target-node (j), 'master', 'index') exists
   create target-node (i, 'variable') = source-node (i)
   link target-node (i) to target-node (j, 'master', 'index')
3. For all source nodes i,
   If source-node (i) = any member and target-node (i, 'variable') exists and source-node (i) = any owner
   create target-note (i, 'master', 'index')
4. Repeat steps 2-3 for all nodes i.

In the above algorithm, 'master' and 'variable' denote the creation of TOTAL master and variable data sets., 'index' denotes creation of a master data set with one key record for each record as the associated variable data set.

Algorithm for IMS

1. For all nodes i,
   If source-node (i) is not = hierarchical member
   create target-node = source-node (i)
2. For all source-nodes i, j and i ≠ j,
   If source-node (i) = hierarchical member of source-node (j) and target-node (j) exists
   create target-node (i) = source-node (i)
   link target-node (1) to target-node (j)
3. For all source-nodes i, j and i ≠ j
   If source-node (i) = non hierarchical member of source-node (j)
   link 'logical' target-note (i) to target-node (j)
4. Repeat 2 and 3 for all nodes i.

In the above algorithm 'logical' denotes the creation of a logical child link between the two nodes.

5. DATA BASE EXAMPLE

Suppose that the following set of third normal form relations is given:

Agent
**agent-no
agent-details

Customer
**customer-no
customer-details

Customer-Account
**customer-no
month-no
account-details

Customer-Address
**customer-no
**address-type
customer-address

Order
**order-no
order-details
*customer-no
*agent-no

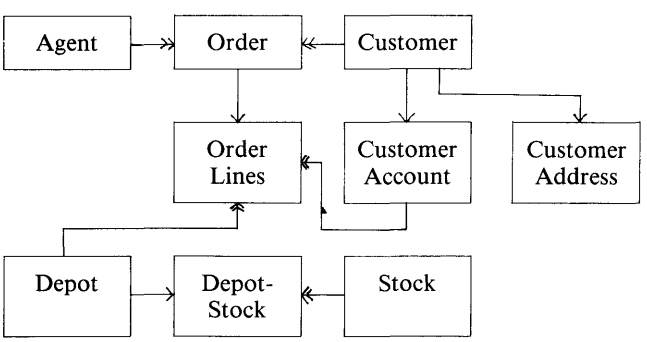
Order-Lines
**order-no
**stock-no
*depot-no
line-details

Stock
**stock-no
stock-details

Depot
**depot-no
depot-details

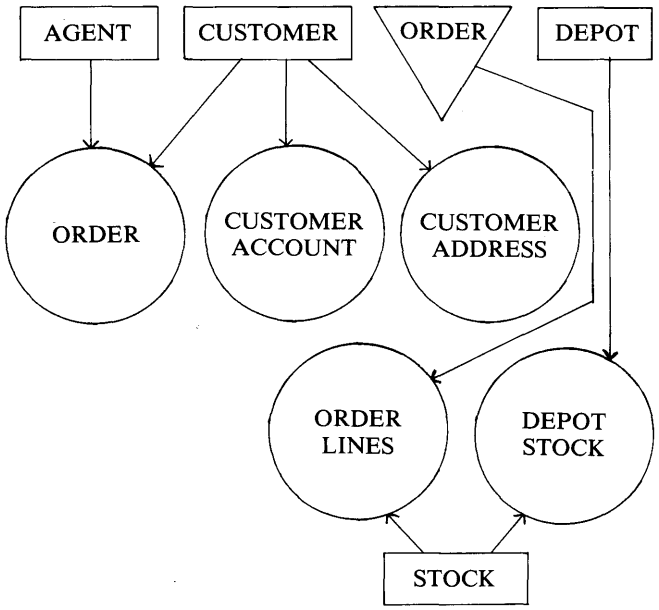
Depot-Stock
**depot-no
**stock-no
quantity

This is a typical set of relations in a commercial accounting environment. Applying the algorithm presented in Section 3 we obtain the following conceptual schema:

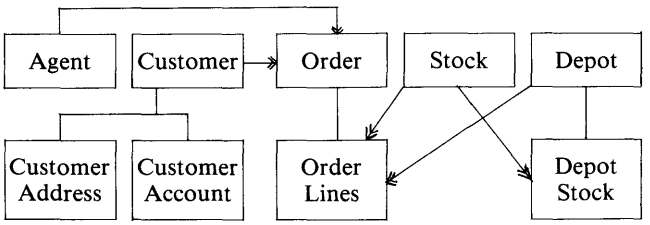


The next step is to complete the data flow diagram for the proposed application. From the data flow diagram, linear data flow diagrams together with their associated local view diagrams can be prepared. The union of all the data flow diagrams corresponds to the refined conceptual schema. We assume that this refined conceptual schema turns out to be equivalent to the above conceptual schema.

If the TOTAL algorithm is applied, the following FPD is obtained (master data sets are designated by □, index master data sets by ∇ and variable data sets by ○):



If the IMS algorithm is applied, the following FDP is obtained:



The double-headed arrow indicates a logical parent-child relationship.

6. Future Developments

In continuing work by the author, application system functional requirements as summarised in the data flow diagram together with the FPD are used to produce a structured design [12] hierarchy diagram. Similar to the algorithms for FDP, algorithms to control subsequent iterative modification of the DBMS schema can be defined. These incorporate the restructuring operations of [15]. Depending on the results of a prediction model at each iteration, appropriate restructuring of the current DBMS schema can be performed to meet either disk space constraints or the response/throughput time constraint. This iteration continues until the DBMS schema most closely resembling the conceptual schema and meeting performance specifications is produced.

7. Conclusions

The algorithms for FPD have been specified here for TOTAL and IMS only. It is relatively simple to produce simple algorithms for other DBMS. For example, Codasyl-type systems such as IDMS need only specifiy procedures for restructuring bijunctive relations to produce an FPD.

They offer a disciplined method for initial physical database design to the database administrator and ensure that the benefits of formal relational analysis are not lost in tailoring the conceptual schema to a DBMS compatible one.

References

[1] E. Aurdal, A multi-level procedure for file design, cascade working paper no. 39, Univ. of Trondheim, 1975.

[2] B. H. Kahn, A method for describing information required by the database design process, Proc ACM Sigmod, 1976.

[3] S. Wulf, A database project methodology, Internal report, Comcon (Pty) Limited, 1977.

[4] S. Wulf, A database project methodology, Internal working paper, Comcon (Pty) Limited, 1980.

[5] Report of the Codasyl DBTG, 1971.

[6] Baskett et al., Open, closed and mixed networks of queues with different classes of customers, JACM 22, no. 2, 1975.

[7] S. B. Yao, An attribute based model for database access cost analysis, ACM TODS, 2, no. 1, 1977.

[8] IMS/VS General Information Manual, Form GH20-1260, IBM, 1982.

[9] TOTAL Application Programmer's Guide, Pub PO2-1236, Cincom Systems, 1981.

[10] E. F. Codd, Further normalisation of the database relation model, Courant Computer Science Symposia, 6, 1971.

[11] C. J. Date, Normalisation — An introduction to database systems, Addison-Wesley, 1975, pp 95-114.

[12] Stevens et al., Structured design, IBM Systems J, 2, no. 2, 1974.

[13] S. B. Navathe, Schema analysis for database restructuring, ACM TODS, 5, no. 2, 1980.

[14] P. P. S. Chen, The entity-relationships model — Towards a unified view of data, ACM TODS, 1, no. 1, 1975.

[15] S. B. Navathe and J. P. Fry, Restructuring for large databases: Three levels of abstraction, ACM TODS, 1, No. 2, 1976.

Notes for Contributors

The purpose of this Journal will be to publish original papers in any field of computing. Papers submitted may be research articles, review articles, exploratory articles of general interest to readers of the Journal. The preferred languages of the Journal will be the congress languages of IFIP although papers in other languages will not be precluded.

Manuscripts should be submitted in triplicate to: Prof. G. Wiechers at:

Department of Computer Science
University of South Africa
P.O. Box 392
Pretoria 0001
South Africa

Form of manuscript

Manuscripts should be in double-space typing on one side only of sheets of A4 size with wide margins. The original ribbon copy of the typed manuscript should be submitted. Authors should write concisely.

The first page should include the article title (which should be brief), the author's name, and the affiliation and address. Each paper must be accompanied by a summary of less than 200 words which will be printed immediately below the title at the beginning of the paper, together with an appropriate key word list and a list of relevant Computing Review categories.

Tables and figures

Illustrations and tables should not be included in the text, although the author should indicate the desired location of each in the printed text. Tables should be typed on separate sheets and should be numbered consecutively and titled.

Illustrations should also be supplied on separate sheets, and each should be clearly identified on the back in pencil with the Author's name and figure number. Original line drawings (not photoprints) should be submitted and should include all relevant details. Drawings, etc., should be submitted and should include all relevant details. Drawings, etc., should be about twice the final size required and lettering must be clear and "open" and sufficiently large to permit the necessary reduction of size in block-making.

Where photographs are submitted, glossy bromide prints are required. If words or numbers are to appear on a photograph, two prints should be sent, the lettering being clearly indicated on one print only. Computer programs or output should be given on clear original printouts and preferably not on lined paper so that they can be reproduced photographically.

Figure legends should be typed on a separate sheet and placed at the end of the manuscript.

Symbols

Mathematical and other symbols may be either handwritten or typewritten. Greek letters and unusual symbols should be identified in the margin. Distinction should be made between capital and lower case letters between the letter O and zero; between the letter l, the number one and prime; between K and kappa.

References

References should be listed at the end of the manuscript in alphabetical order of author's name, and cited in the text by number in square brackets. Journal references should be arranged thus:

1. ASHCROFT, E. and MANNA, Z. (1972). The Translation of 'GOTO' Programs to 'WHILE' Programs, in *Proceedings of IFIP Congress 71,* North-Holland, Amsterdam, 250-255.
2. BÖHM, C. and JACOPINI, G. (1966). Flow Diagrams, Turing Machines and Languages with only Two Formation Rules, *Comm. ACM,* **9**, 366-371.
3. GINSBURG, S. (1966). *Mathematical Theory of context-free Languages,* McGraw Hill, New York.

Proofs and reprints

Galley proofs will be sent to the author to ensure that the papers have been correctly set up in type and not for the addition of new material or amendment of texts. Excessive alterations may have to be disallowed or the cost charged against the author. Corrected galley proofs, together with the original typescript, must be returned to the editor within three days to minimize the risk of the author's contribution having to be held over to a later issue.

Fifty reprints of each article will be supplied free of charge. Additional copies may be purchased on a reprint order form which will accompany the proofs.

Only original papers will be accepted, and copyright in published papers will be vested in the publisher.

Letters

A section of "Letters to the Editor" (each limited to about 500 words) will provide a forum for discussion of recent problems.

Hierdie notas is ook in Afrikaans verkrygbaar.

Quaestiones Informaticae

Contents/Inhoud

*Presented at the second South African Computer Symposium held on 28th and 29th October, 1981.

Printed by Studio Press, Johannesburg