OPTIMIZING THE LOGICAL DATABASE DESIGN

DICK ONEL

DCE DATABASE CONSULTANTS EUROPE
PRINSENGRACHT 747-751
AMSTERDAM
HOLLAND

ABSTRACT

The logical database design of an IMAGE database consists of those aspects of the design that determine it's functionality namely it's structure, in terms of data sets, and the entry points to those data sets. When the logical database design has been enhanced with the physical parameters and reorganisation procedures which determine the physical organisation of the database, we speak of the physical database design.

The physical database design process is oriented at obtaining optimal performance. However, there are also many ways to optimize in the logical database design which result in better performing and longer lasting database applications. If these applications have unsatisfactory performance, or if they use more resources then necessary or available, it is, among others, the logical database design which must be re-evaluated. In order to do this, we must understand the transaction(s) causing the problems, know Turbo-Image, and have resources, such as disk space, or CPU cycles, to make design trade-offs with.

The logical database design is made on the basis of assumptions of how the data will be used, and should support all required accesses to the data. When the database is in production the real usage of the data should be re-evaluated. Based on these re-evaluations the access structures should be revised , i.e. remove not used or infrequent used paths and/or add new paths. The logical database model should theoretically have no redundant data. If this model shows performance problems we should optimize it. We can add redundant data in masters and/or details or add redundant data and access structures. We can split entries , distinguish between files and tables and between current and historical data.

These measures should make a stronger foundation for the future of our databases.

INTRODUCTION

The logical database design of an IMAGE database consists of those aspects of the design that determine it's functionality - namely it's structure, in terms of data sets, and the entry points to those data sets. When the logical database design has been enhanced with the physical parameters and reorganisation procedures which determine the physical organisation of the database, we speak of the physical database design. The physical database design process is oriented at obtaining optimal performance.

However, there is no point to optimizing the physical storage of the database if the logical database design does not match the requirements of the business. A poor logical database design is often the fundamental cause of performance problems in Turbo Image databases.

In order to produce good logical database designs, we need methods and techniques which will enable us to identify what the business requirements mean in terms of what data should be stored in the database, what the inherent logical structure of that data is, and how it will be accessed.

All design begins with analysis , even if prototyping techniques are used. The analysis should be carried out in a structured manner and produce documentation and diagrams that feeds into the design process.

The Systems Development Cycle, shown in figure 1, is a proven approach for developing database systems, and forms the basis of the techniques discussed in this paper.

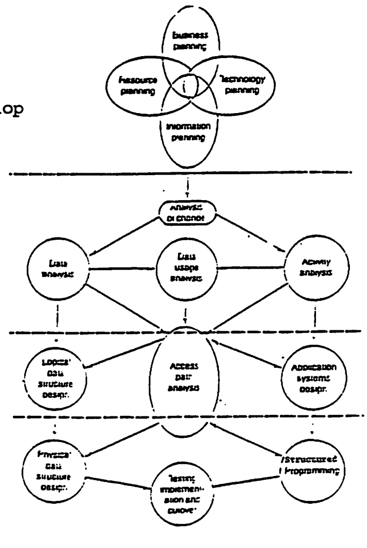


Figure 1.
The Systems Develop
Cycle

The right hand side deals with activities, functions or processes. The left hand side deals with data. Recent studies show that the data side of an application system is far more stable than the activity side. It is therefore <u>imperative</u> to have separate analysis and design processes for determining the data structure, and not just let the data structure "fall out" of the functional analysis and design.

The first part of the analysis phase is defining the application-system aims and the scope of the analysis; this is called the BUSINESS ANALYSIS or FEASIBILITY-STUDY.

The second step is the analysis of the data resources used and the analysis of the user's information handling processes, the DATA ANALYSIS and the ACTIVITY ANALYSIS. To cross-reference, and check the consistency of the Data Analysis and Activity Analysis, DATA USAGE ANALYSIS is performed.

At the next step, the LOGICAL DATA STRUCTURE DESIGN (or LOGICAL DATABASE DESIGN) is performed, using the results of the ACCESS PATH ANALYSIS, which determines <u>how</u> the activities use the data.

The end deliverable of this step is the logical database design. The logical database design is then used as input to the PHYSICAL DATA STRUCTURE DESIGN or PHYSICAL DATABASE DESIGN process.

In the following sections of this paper, the various steps of the Systems Development Cycle, which result in the production of the logical database design, are discussed in more detail, using the breakdown shown in figure 2.

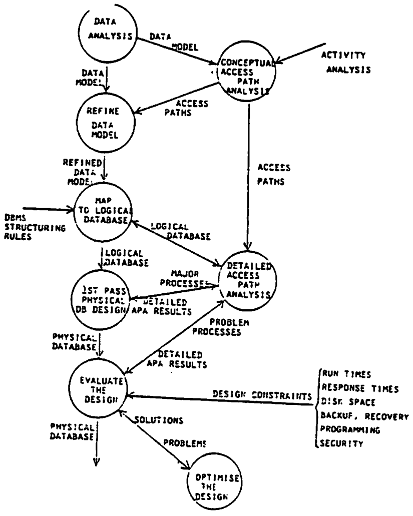


Figure 2.
The Database Design Process

In this diagram Logical Data Structure Design has been broken down into two steps, namely 'REFINING THE DATA MODEL' and 'MAP TO THE LOGICAL DATABASE'. Physical Data Structure Design has been decomposed into '1ST PASS PHYSICAL DATABASE DESIGN', 'EVALUATE THE DESIGN' and 'OPTIMIZE THE DESIGN' steps. Access Path Analysis has been broken down into 'CONCEPTUAL ACCESS PATH ANALYSIS' (used to support the Logical Data Structure Design) and 'DETAILED ACCESS PATH ANALYSIS' (used to support the Physical Data Structure Design).

This discussion is followed by a detailed examination of the steps that can be taken to optimize the logical database design.

DATA ANALYSIS

DATA ANALYSIS is a method used to understand and document a complex environment in terms of its data resources.
I use here the ENTITY-ATTRIBUTE-RELATIONSHIP (EAR) method, a more rigorous form of Peter Chen's Entity-Relationship approach.
The output of the Data Analysis consists of a Data Model Diagram which describes the data resources in terms of entities and relationships, and a data dictionary describing the entities, attributes and relationships.

An ENTITY is something of fundamental importance to a company. It is thus something about which data will be kept in an information processing system. An entity is shown on the data model by means of a rectangular box, with the name of the entity inside. Examples are : objects, people, places and abstractions such as an event.

An ATTRIBUTE is a basic unit of information which describes an entity. Within the company environment an attribute cannot usefully be subdivided into other units of information. An entity must have attributes if it is of interest to the company. Examples are : policy number, date policy started, person's name, person's date of birth.

A RELATIONSHIP is an association between entities. It is shown on the data model by means of a line drawn between the entities. Examples are : the entity order is related to the entity "order-line", the entity "car" is related to the entity "part".

Whether something is an attribute or a entity is dependent on the company environment. For example in a car factory, colour is an attribute of the entity car because colour is a basic unit of information .In a paint factory, however, colour is an entity because it probably has a number of attributes.

A relationship has a degree, this indicates how many occurrences of an entity are related to one occurrence of the other entity. The degrees are one-to one, one-to-many and many-to-many.

A relationship also has a type, e.g. OPTIONAL, this indicates that an occurrence of one entity MAY be related to one or more occurrences of the other entity in the relationship.

ORDER PROCESSING DATA MODEL

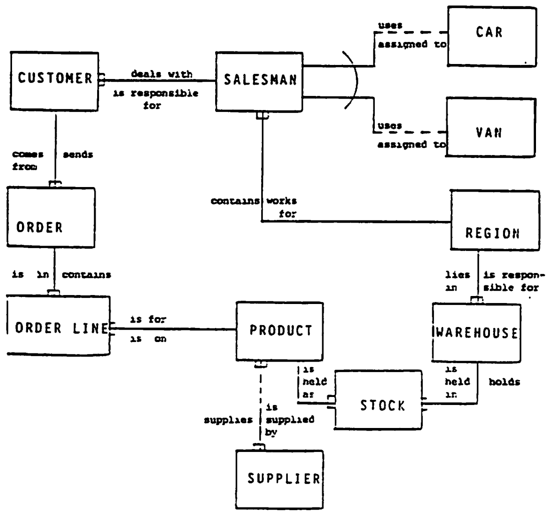


Figure 3. Data Model

This is shown in the data model by a dotted line between the entities, an example is the relation between PRODUCT and SUPPLIER (see figure 3). It means that there can be a product without a supplier (this company also makes their own products).

Another relationship type is EXCLUSIVE; this means that an entity is related to either one of a number of other entities. For example the entity SALESMAN is related to the entity CAR or to the entity VAN, this is shown by an arc (see figure 3).

The Data Model diagram contains all entities and relationships in the scope of the project. The relationships lines can have at end special symbols as arrows or tridents indicating the relationship degree (see figure 3). In addition to the data model all entities, attributes and relationships should be documented in a data dictionary. This documentation should also include precise descriptions, volumes, characteristics, special conditions and values etc. Figure 3 shows a Data Model for an order processing system.The Data Model is used to enable the users and the information-analysts to quickly understand a complex data area (one diagram is more than thousand words). The Data Model is the basis for the logical database design.

Building the Data Model is an iterative process. By definition, the analyst does not know the details of every entity, attribute and relationship at the start of the analysis . Intermediate Data Models will be wrong, but should allow the analyst to extract further information from the user in order to arrive at a correct model.

CONCEPTUAL ACCESS PATH ANALYSIS

Before a database can be designed we must understand the data structure (Data Analysis) and how the data is used. Access Path Analysis is the process of showing how the business activities defined in the Activity (or Functional) Analysis uses the data units defined in Data Analysis.

The functions identified during the Activity Analysis can be grouped as retrieval and creation. The retrieval functions just read the data, while the creation functions may, apart from reading, also insert, delete and modify the data.

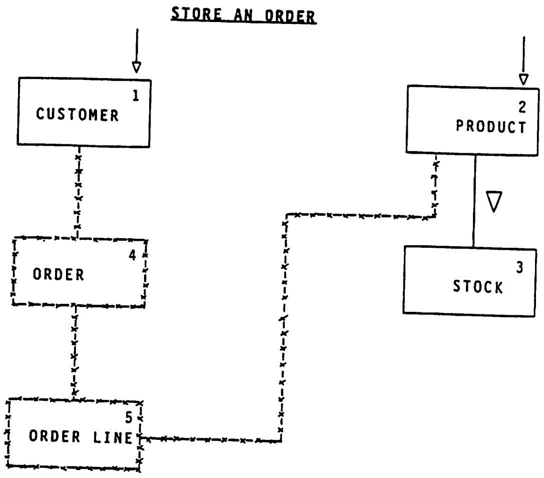
For every function a profile must be made indicating the initial access to the data and the way the function uses the data structure.

An ENTRY POINT shows the initial access to the data. It is a search for one occurrence of an entity (e.g. customer-code 159) or a search for several occurrences (e.g. all customers called Smith).

A NAVIGATION PATH shows the usage of the data structure. It consists of one or more entities and their relationships.

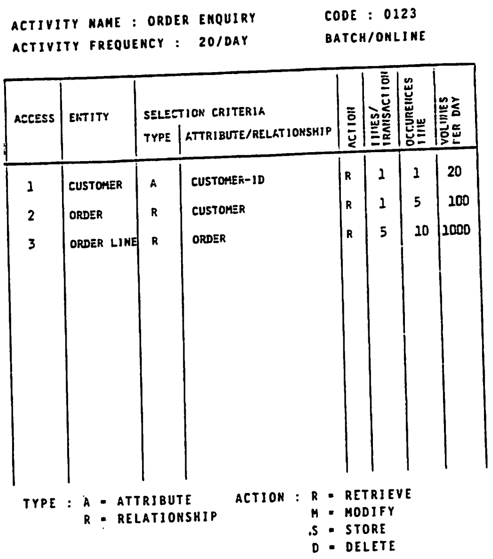
An example is shown in figure 4; here the creation profile of the "Store An Order" function is drawn. The numbers in the entities shows the order of access. The first entry point is Customer-no in the entity Customer, to check if the Customer exists. The second entry point is Product-no in the entity Product to check if the required product exists, then following the path to the entity Stock is followed, in order to check if there is stock for the required product etc.

Figure 4. Creation Profile



In addition to drawing the access profiles in the Data Model, forms should be filled in for each process showing frequency,volumes of data accessed, specific attribute usage within the entities, whether it is a response time critical function etc.(see figures 5 and 6). From these forms a number of statistics and matrices can be generated :

- Usage statistics show how frequently entities and relation are used.
- Usage matrices show in which functions the entities and relationships are used and/or created. Throughout the database design process it is essential to have an easy cross-reference of activities (or processes) to entities (or files). This is the only way to evaluate the total effect of design decisions.
- Attribute usage matrices show, per entity, where attributes are used and/or created.


ACTIVITY NAME : ORDER ENQUIRY CODE : 0123

ACTIVITY FREQUENCY : 20/DAY BATCH/ONLINE

ACCESS	ENTITY	SELECTION CRITERIA		ACTION	TIMES/ TRANSACTION	OCCURENCES TIME	VOLUMES PER DAY
		TYPE	ATTRIBUTE/RELATIONSHIP				
1	CUSTOMER	A	CUSTOMER-ID	R	1	1	20
2	ORDER	R	CUSTOMER	R	1	5	100
3	ORDER LINE	R	ORDER	R	5	10	1000

TYPE : A = ATTRIBUTE
R = RELATIONSHIP

ACTION : R = RETRIEVE
M = MODIFY
S = STORE
D = DELETE

Figure 5.
Access Path Form

ACTIVITY NAME : ORDER ENQUIRY CODE : 0123

Figure 6.
Attributes used per Activity

ACCESS	ENTITY	ATTRIBUTES REQUIRED	SORT FIELDS
1	CUSTOMER	CUSTOMER-ID	
		CUSTOMER NAME	
		CUSTOMER ADDRESS	
2	ORDER	ORDER-ID	
		TOTAL VALUE	
		BTW	
		DATE	DESCENDING
3	ORDER LINE	LINE-ID	ASCENDING
		PRODUCT-ID	
		QUANTITY	
		PRICE	

The matrices and statistics can be compiled manually but since the design process is iterative it will be a lot of work. A better solution is the use a good Data Dictionary. Most analyst-workbenches generate the matrices and statistics automatically.

REFINING THE DATA MODEL

The aim of this phase is to create a data model that contains only the required data and access structures, using the results of the conceptual access path analysis. What is required to be implemented may differ from what is conceptually present, because it is not normally practical to automate everything.

The transformations carried out in this phase are as follows:

1) Relationships

   - An added relationship may be redundant in the conceptual model, but important as a result of access requirements.
   - A relationship may exist in the conceptual model but will be eliminated when conceptual access path analysis does not find a requirement for it.

2) Entities and Attributes

   - If access path analysis shows that an entity is never used, it can be eliminated. Similarly, attributes can be eliminated if access path analysis shows that they are never used. If you decide to leave them in, remember that they may need to be maintained. If you leave them out ask yourself the questions : will it be required later and how easily can they be added. Make sure you document your decisions.
   - Entities which are always accessed together can possibly be mapped to one record type. This depends on the number of occurrences, the volume and the variability.
   - Exclusive entities with similar attributes could be combined. An indicator must be added to show the difference in the occurrences.

3) Many-to many relationships

   Because few database management systems will handle 'many-to-many' relationships, the should be eliminated at this stage. They are eliminated by the introduction of a new "in-between" entity (sometimes called a "relation entity") .The new entity is related to the original two entities with two 'one-to-many' relationships (see SUPPLIER-PRODUCT)

Starting from your conceptual data model you have selected everything that was required using the access path analysis results. You have simplified and optimised the required parts of the data model. You have created the REFINED DATA MODEL; now you are ready to map to a Turbo-Image database. Make sure you have documented all changes in the Entity-Attribute-Relationship descriptions.

The Refined data model is shown in figure 7.

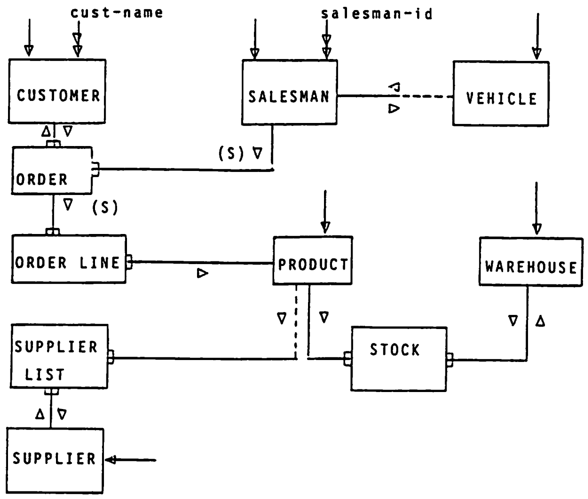


Figure 7.
Refined Data
Model

MAPPING TO THE LOGICAL DATABASE

The Turbo-Image database management system has a set of "structure rules" which does not allow direct implementation of the refined data model. We must therefore translate the refined model to the logical model. We must also implement the access requirements into this logical model. This step is called mapping.

The Structure Rules for Turbo-Image are as follows:

1) DATA SET

   A DATA SET is a collection of records with the same record format. In conventual terms it is a file with one record type. Turbo-Image supports two types of data-sets, MASTER and DETAIL.

2) PATH

   Relationships are supported as PATH's between masters and details. These path's consists of one entry in a master set linked to zero, one or more entries in a detail set. The linking is done via a common value in the search field of a master set entry and the search field of a detail set entry.

A path is a means of connecting entries in master data sets to entries in a detail data set. Paths only exist between masters and details. The search field in the master data set is linked to one of the search fields in the detail set if they contain the same value. The physical implementation of the path is called a chain, which is a pointer in one entry pointing to another entry in the same set (as in details) or in a different set (from master to detail). The entries in a chain in a detail set may be sorted on a data field in that entry. This data field is then called the SORT field.

3) MASTER DATA SET

A master data set has one unique data field called the search field. Turbo-Image supports calculated (hashed) access to the master set using the value of the search field. A master data set can be linked to up to 16 detail sets.
Two types of master sets are supported, the MANUAL master set and the AUTOMATIC master set.

4) MANUAL MASTER SET

The manual master may contain more data fields besides the search field. The master data set must be maintained by the application programs. If an entry must be added to a detail set, linked to a manual master set, an entry with the same search value must first be created in the manual master set.

5) AUTOMATIC MASTER SET

An automatic master set entry contains only one data field, the search field. The automatic master set is maintained by Turbo-Image. When a data entry is added to the associated detail set using a value of the search field that does not already exits in the automatic data set, an entry with that search value is created in the automatic data set. When the last entry with a certain search value is deleted for the associated detail set, the corresponding automatic master set entry is deleted by Turbo-Image.

6) DETAIL DATA SET

The detail dat sets can contain several data and search fields. A detail data set can be linked to maximum 16 master data sets.

7) ACCESS METHODS

Several access method are supported by Turbo-Image. For the logical database design the most important are shown grouped by data set type:

Access on masters :	SERIAL	- the sequence in which the entries are physical stored
	HASHED	- the value of the search field is used to find the entry
Access on details :	SERIAL	- the sequence in which the entries are physical stored
	CHAINED	- all detail entries belonging, via the same search value, to one master entry. The sequence of the chain may be controlled via a sort field.

For other "structural limitations" I refer to the Turbo-Image reference manual (32215-90050).

The mapping process consists of 13 steps which map the refined data model to a logical data model.

1. Select all entities from the refined data model that only have 1:1 or 1 : M (many) relationships with other entities and map them to a manual master data set.

2. For each manual master determine, from the access profiles, the search field. If more than one search field is required, choose the search field required most often for on-line access.

3. Map all remaining entities to detail data sets.

4. For every detail set, implement the M : 1 relationship that exists in the refined data model with the entities that were mapped to manual masters in step 1, and include 'the search field of the master and the detail data record.

5. Select all entities that were mapped to detail data sets in step 3 but have 1 : 1 or 1 : M relationships with other entities in the refined data model.
   Choose a suitable search field for these entities (refer to the access path analysis).

6. Create automatic masters containing the selected search field for the detail data sets selected in step 5.

7. Implement the 1 : M relationships that exists in the refined data model between two entities that were mapped to detail data sets by relating them to the automatic masters created in step 6.
   Include the search field in the detail data record.

8. Implement the relationship between two manual master sets (1 : M and 1 : 1 relationships in the refined data model) by creating a detail data set containing only the search fields of the manual masters.

9. Consider extra automatic masters for details on which direct entry is required (see access path analysis) or access via the existing master sets is clumsy.

10.Remove all paths (not the search fields) when the access profiles show that you only go from detail to master and never from master to detail.

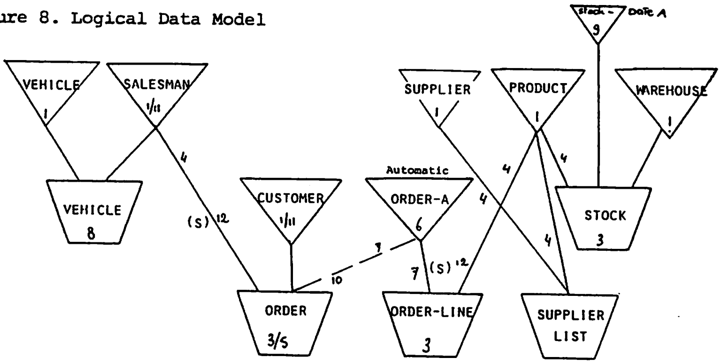
11.If more than one search field on a master set is required, consider mapping the master to a detail with automatic masters for the needed extra search fields.

12.Determine for the access profiles which master to detail paths should be sorted, and on what key. If more than one sort order is required, choose the one required most often for on-line retrieval.

13.If a detail data set has an optional relationship with a master set implement a extra automatic master and an extra detail, because Turbo-Image will not allow non-owned detail entries.

The logical data model shown in figure 8 is produced via the mapping rules from the refined data model and the access profiles shown in figure 7.
The number of the appropriate mapping rule is shown for reference only.

Figure 8. Logical Data Model



1ST PASS PHYSICAL DATABASE DESIGN

The logical data model must now converted to a physical model, on the basis of the following considerations:

- performance optimisation
- environmental constraints (CPU, disk-io's, disk space)
- user constraints (response times, run times, security, backup and recovery)

This process is outside the scope of this paper.

DETAILED ACCESS PATH ANALYSIS

This process supports physical database design, and optimization, by evaluating how a particular function will use the database in terms of :

- use of entry points
- records read
- records selected
- use of paths
- sorts
- frequencies

The detailed access path analysis differs from the conceptual access path analysis that it looks at volumes, importance and time-criticality.
An analysis is done for all major processes to check for problem areas in the logical model.

Figure 9. Evaluation of physical I.O.

ACTIVITY NAME: ORDER STOCK CHECK　　　　CODE : 0123

FREQUENCY : 20/DAY　　　　~~BATCH~~/ONLINE

RESPONSE TIME: 2 SECS

ACCESS	RECORDTYPE	VIA RECORD-TYPE	ACCESS TYPE	ACTION	SELECTION (SEQUENCE)	TIMES/TRAN	RECORDS READ/TIME	RECORDS SELECTED TIME	BLOCKS READ/TIME	BLOCKS READ/TRAN
1	ORDER-A	-	M	R	ORDER-NO	1	1	1	1	1
2	ORDER-LINE	ORDER-A	D	R	-	1	10	10	5	5
3	PRODUCT	-	M	R	PROD-CODE	10	1	1	1	10
4	STOCK	PRODUCT	D	R	WAREHOUSE	10	2	1	1	10
										26

TOTAL I/O = 26 x 0,1 Sec = 2,6 Secs

For every major process a form, as in figure 9 for the "order stock check" function, is filled out. This form then shows, per user process, the expected database response times.

This is the first moment in the database design process that we can see the number of I/Os that a certain function is generating. We can also see if we can achieve the required response time.

EVALUATING THE DATABASE DESIGN

This phase takes the physical database design, and determines what is wrong with it. The design is wrong if the design constraints are not met; i.e. the run or response times are too long, the structure is so complicated that programmers have difficulty in using it properly, security is compromised, or the operational tasks cannot be performed effectively.

When evaluating the design, we concentrate on the problem areas. Performing Detailed Access Path Analysis takes a long time, and there is no point in doing so if there is little likelihood that the results will impact on the final database design. The following problems can be identified during the evaluation:

1. Access paths too long
2. Too many access structures
3. Wrong primary path
4. Locking too much data
5. No distinction between files and tables
6. No distinction between current and historic data
7. Large entries
8. Long back-up/recovery

The following section describes the Logical Database Optimization Solutions that can be used to address those problems.

OPTIMIZING THE LOGICAL DATABASE DESIGN

When optimizing the database design, we try the simplest solutions first. For example, physical database design decisions are re-examined, and where necessary reversed. However, the problems are often more fundamental, and, like it or not, the logical database design must be optimized. The possible measures discussed on a per-problem basis, as follows:

1) ACCESS PATH TOO LONG

   A function may have to go through a far too long access path to get the required data. Solutions are to add redundant data or to a redundant access structure.

Put data items you would otherwise have to determine by accessing one or more detail records in the master record.
If particular reports are often required, calculate them once and store them in a separate data structure.

2) TOO MANY ACCESS STRUCTURES

An "access structure" is a database structuring which is used to find a record, it is any access method other than serial search or sorting.
Every access structure in a database needs space and causes maintenance overhead.
Each one must be examined to determine if it is really necessary.
Consider external sorts and/or program internal tables against the overhead of maintaining sorted relationships.
If a detail set has too many masters (say more than four) the number of disk I/O's it takes to ADD an entry to this detail set will probably give unacceptably long response times .

3) WRONG PRIMARY PATH

The primary path is defined as the most used path to a certain detail set. This can be determined from the detailed access path analysis. It should however be evaluated when the database is in production. The Hewlett-Packard product PROFILER shows exactly what the most used path into a detail set is.
Keep in mind that the primary path is optimal only after reorganisation of a detail set.

4) LOCKING TOO MUCH DATA

When a small subset of attributes (fields) are subject to frequent change, causing the whole entry to be locked, think about splitting the data set.

5) NO DISTINCTION BETWEEN FILES AND TABLE

If a data set has only a few, small records which are infrequently updated, make it a look-up table or a data set with no relationships. The integrity constraints must then be handled by the application programs.

6) NO DISTINCTION BETWEEN CURRENT AND HISTORIC DATA

The critical questions to ask are as follows:

- When does current data become historical?
- What happens to it when it is historical?

- How can you recall it in the future?
- Should you split a data set?
- If so, what happens to the functions or queries that need all the information available?

7) LARGE ENTRIES

Large entries will require large buffers, and this will bring down the number of buffers, thus making the database slower in multi-user interactive environments. A remedy is to split the entry (make it two data sets).

8) LONG BACKUP/RECOVERY

If your database is very large it will take long time to back it up. Normally this means that the database is not available for users and if you do the backup at night it means extra operator hours.
If you split your database in two parts and if you have two tape-units you can do the back-ups concurrently in half the time it used to take.

CONCLUSION

Optimisation is a process in which you :

- pay the most attention to parts of the database with known problems,
- try the simplest solutions first, and
- preserve the data-structure as much as possible.

Optimisation is a process of making trade-offs. The major trade-offs are between update and retrieval efficiency, and between ease of use and performance. It is important to strike a balance between the effort required to optimize the database and the return on this investment. In short, designing a 100 % optimal database is not a 100 % optimal use of our time.