An Integrated Approach to Data Base Design

M. HAMED
University of Bahrain, College of Science,
Dept. of Comp. Sci., Isa Town, Bahrain

ABSTRACT. In this paper, a new method for data base design is proposed. The method makes use of some good features in different design approaches. Based on the results of a preliminary phase of determining user requirements, an initial design is first developed. Then the design is tuned according to some pre-defined changes in both the logical model and the physical representation. Response time is used as a measure of the effectiveness of the design.

Algorithms to implement the method are developed. Selection of the best design model (logical and physical) is left to the data base administrator (DBA). Results obtained from running the proposed algorithms can be used to help the DBA in taking the appropriate decision.

1. Introduction

Design of data bases is mostly done through two main consecutive phases: logical and physical design phases.

Logical design can be defined as: grouping of attributes into entities and building relationships between entities. Logical design may be based upon some specific theory as dependency theory\cite{1-3}, relationships between entities\cite{4-5}, logical record access\cite{6} ... etc. The model in which the design can be expressed is different in each case (e.g. relations, entity-relationship model, tree ... etc.). Transformation between models\cite{7} can also be done.

The physical design may involve some activities such as: selection of file organization to each entity, selection of a method of implementing relationships between entities and implementing inverted files. For example, the goal of the design of a physical model could be the minimization of retrieval response time.
The separation of the two phases (logical and physical) is meant mainly to assure independence. Despite that, it can be claimed that the mapping process from logical to physical models could affect the logical model if some specific objective is to be met (e.g. splitting an entity to two entities to enhance overall response time).

In this paper, a new method for data base design is proposed. The method is based upon an iterative process of tuning the logical and physical models till achieving the best result for some specific goal. For the sake of tuning, different changes are suggested to be applied on each model as both models affect the final design goal. The main design goal is to minimize retrieval (update) response time; while the secondary goal is to get the best possible value for update (retrieval) response time.

In section 2, different logical design approaches are surveyed.

In section 3, the design principles and the new suggested method itself are given. The method is described in the form of a procedure to be developed as a design aid.

In section 4, a case study is considered to illustrate the effectiveness of the method.

2. Different Approaches to Logical Design

Normalization of relations\[^{1-3,8}\] can be used as a basis for logical design. The theory emphasizes the associations (dependencies) between attributes (data items). The main goal of the design is to remove update anomalies. The main point here is how to remove update anomalies only in the logical model without considering any corresponding physical model. To illustrate this point consider the following example:

The relation CTX (C, T, X) is in 3NF where the attributes have the following conventions: C (Course), T (Teacher), X (Text). In this relation two MVDs hold, namely

\[ C \rightarrow T \quad \text{and} \quad C \rightarrow X \]

According to some normalization rules, the relation should be decomposed to two 4NF relations to remove update anomalies. The resulting relations are: CT (C, T) and CX (C, X). Now, if the original relation CTX is to be stored physically as an inverted file for all its attributes, no update anomalies would occur, and consequently further normalization steps become unnecessary. Thus, designing the logical model without considering a corresponding physical model is not a good way of design.

The entity-relationship approach\[^{4,5}\] can also be used as a basis for logical design. This approach handles entities and their relationships, but it pays no attention to the associations between attributes.

Hubbard\[^{9}\] introduces a methodology for computer-assisted design which is limited to DL/1 applications. In this method, only one-to-many relationships are considered, whereas many-to-many relationships are treated as ‘logical relation candidates’ (according to DL/1 terminology).

Henver and Yao\[^{10}\] introduce some design methods for network data bases. These
methods depend mainly on the analysis of queries to deduce ‘owner-member’ relationships between entities. Normalized relations are considered only if they were introduced in this structure from the first beginning and are converted to the main structure of the network (owner-member relationships).

Teory and Fry⁶ propose a method of design based on the idea of minimizing logical record accesses. The method suffers from the following disadvantages: (1) The measure of response time is not accurate, (2) Minimization of response time cannot be considered for both retrieval and update operations together as these two factors may oppose each other, (3) Associations between attributes are not taken into consideration.

Finally, we remark that: Although the relational and entity relationship approaches seem to be more appropriate than others; neither any of them proved to be superior than the other¹¹. Consequently, we can say that there is no such approach that can be considered the best.

3. The Suggested Design Method

3.1 General Description

The goal of the design is “minimization of response time”. It is convenient¹² to consider either retrieval or update response times separately.

The proposed changes to physical model are: (1) Selection of file organizations, (2) Implementing hierarchical relationships between entities, (3) Inverting some attributes, and (4) Implementing reverse access path. The suggested changes to logical model are: (1) Normalization, (2) Splitting an entity to two entities, and (3) Joining two entities in one entity.

Refinement of the models is done on successive steps by trying different alternatives of physical and logical states in turn till achieving the final model. This makes the design process complete as this method does not suffer from problems of other methodologies. Some important problems that are tackled here are:

i – In the relational approach, removal of update anomalies should not be considered without any reference to a physical model. Also update anomalies¹³ should not be taken as the only goal of the design.

ii – In the entity-relationship model, the model has to be mapped to some data structure in all cases.

Factors that affect the design are: (1) Entities, attributes and relevant relationships, (2) Frequencies of reference in all views, (3) Available access paths.

3.2 The Method of Design

The suggested design method is based upon developing a design $D_{L,i}$ for logical model $M_i$ and physical model $P_j$. Other designs are obtained by applying changes to both models. Selection of best design is left to the data base administrator. The worst design is that one whose response time is the largest and the ratio between this re-
M. Hamed

response time and the next smaller time exceeds some specified value. The method can be described in the following sequence of steps (symbols are explained in the appendix):

1. Develop a logical model $M_0$ using algorithm-1.
2. Calculate retrieval and update times $RT_{0,j}$ and $UT_{0,j}$ for all possible physical models $P_j$ (each model consists of several physical options).
3. Select the worst design $D_{0,w}$ corresponding to logical model $M_0$ and physical model $P_w$ with a set of physical states $Z_w$.
4. Modify the logical model $M_0$ by applying normalization steps. Assuming that there are $m_1$ new models, then for each model $i$ calculate $RT_{i,j}$ and $UT_{i,j}$ for all possible physical models $P_j$ (excluding those models that contain the set of physical states $Z_w$).
5. Select the worst designs $\{D_r\}$ for logical models $\{M_r\}$ and physical models $\{P_r\}$ with a set of physical states $Z_r$.
6. Modify the remaining logical models by applying Split/Join steps. Assuming that there are $m_2$ new models, then for each model $i$ calculate $RT_{i,j}$ and $UT_{i,j}$ for all possible physical models $P_j$ (excluding those models that contain the sets of physical states $Z_w$ and $Z_l$).

It should be noted that:

(a) An entity $E$ is split to two entities $E_1$ and $E_2$ if $TV_1 >> TV_2$ ($TV$ is the transport volume) and $FJ_{12} < C_1$
(b) Two entities $E_1$ and $E_2$ are joined in one entity if $RS_1 << RS_2$ and $FJ_{12}$ is high ($>> C_2$).

where $RS$ is the record size, and $FJ$ is the joint frequency of reference of the two entities.

(7) Select the best design $D_{e,h}$ for logical model $M_e$ and physical model $P_h$.

3.3 Formulae of Calculation

Given below are the formulae of calculating $RT$ and $UT$. Symbols used are:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$FV_i$</td>
<td>frequency of view $i$.</td>
</tr>
<tr>
<td>$AC_{ij}$</td>
<td>access time of a record of entity $j$ in view $i$.</td>
</tr>
<tr>
<td>$FR_{ij}$</td>
<td>number of references of entity $j$ in view $i$.</td>
</tr>
<tr>
<td>$NE_i$</td>
<td>number of entities in view $i$.</td>
</tr>
<tr>
<td>$u$</td>
<td>number of update views.</td>
</tr>
<tr>
<td>$r$</td>
<td>number of retrieval views.</td>
</tr>
<tr>
<td>$q$</td>
<td>number of query views.</td>
</tr>
<tr>
<td>$NR_j$</td>
<td>number of records of entity $j$.</td>
</tr>
<tr>
<td>$NB_j$</td>
<td>number of blocks of entity $j$.</td>
</tr>
<tr>
<td>$TB$</td>
<td>time to access a block.</td>
</tr>
<tr>
<td>$AS_{jk}$</td>
<td>size of attribute $k$ in entity $j$.</td>
</tr>
<tr>
<td>$NN$</td>
<td>total number of views.</td>
</tr>
<tr>
<td>$C_1$ and $C_2$</td>
<td>are constants.</td>
</tr>
</tbody>
</table>
The formulae are:

1. \( UT = \sum_{i=1}^{u} FV_i \left( \sum_{j=1}^{NE(i)} AC_{ij} \cdot FR_{ij} \right) \)

2. \( RT = \sum_{i=1}^{r} FV_i \left( \sum_{j=1}^{NE(i)} AC_{ij} \cdot FR_{ij} \right) \) + \( \sum_{i=1}^{q} FV_i \left( \sum_{j=1}^{NE(i)} AC_{ij} \cdot FR_{ij} \right) \)

where \( RT \) is subject to the following constraint:

If \( FR_{jj} = NR_j \) and access is sequential then \( AC_{jj} \cdot FR_{ij} = NB_j \cdot TB \) otherwise it is equal to \( FR_{jj} \cdot TB \cdot CF_j \).

3. \( TV_j = \sum_i FV_i \cdot FR_{ij} (AS_{i1} + AS_{i2} + \cdots + AS_{ik}) \)

4. \( FJ_{jk} = \sum_{i \in NN} FV_i / \sum_{i=1}^{NN} FV_i \)

3.4. Algorithms

In this section, algorithms are given to the following functions: (1) Build the logical model, (2) Check the validity of views, (3) Calculate response time. Other necessary algorithms and formulae can be found in the literature. For example, a decomposition algorithm is found in reference [14], and calculation of access time is found in reference [15].

3.4.1 Algorithm - 1: Build the Logical Model (BLD)

Input:
1. A set of entities \( E(MM) \).
2. A matrix \( A(MM, MM) \) of relationships between entities.
3. Matrices describing the views: \( VW(NN, MM, KK) \), \( NE(NN) \), \( NOD(NN, MM) \), \( AA(NN, MM) \), \( RL(NN, MM, MM) \).

Output:
A multi-level network in the form of arrays \( OUT(MM, MM) \), \( L(MM) \), and \( INT(MM, 2) \), where

\( OUT(j, k) \) k = 1 \cdots m, represent a group of nodes directly emanating from node \( j \).

\( L(j) \) is the level of node \( j \).

\( INT(m, 1) \) and \( INT(m, 2) \) are two nodes, the relationship between which is many-to-many.

Description:
First, the top level nodes (level 1) are determined. Next, for each node at level \( i \), all nodes emanating from it are determined (assumed that this set of nodes is \( S \)). Each node in \( S \) is assigned a level number equal to \((i + 1)\) except those having backward relationship for one to many to the original node. In the next step, the many to many relationships are indicated (for possible creation of intersection...
nodes). Note that a node having more than one parent will be assigned the largest level number among those assigned to it due to parent-child relationships. In the last step, a check is made to discover any conflict between a view and the conceptual schema.

**Steps:**

Step 1. Select first level nodes

For $J = 1$ to $MM$ do

Begin

$SW \leftarrow 0$;

For $I = 1$ to $MM$ do

if $A(I, J) \neq NULL$ then $SW \leftarrow 1$;

If $SW = 0$ then $L(J) \leftarrow 1$;

End { $J$ }

End { $I$ }

Step 2. Assign levels to other nodes

$CL \leftarrow 1$; $SW \leftarrow 0$;

Repeat

For $I = 1$ to $MM$ do

Begin

$K \leftarrow 0$;

If $L(I) \neq CL$ then

For $J = 1$ to $MM$ do

if $A(I, J) \neq NULL$ and not $A(I, J) = 'M'$ and $A(J, I) = 'M'$ then

Begin

$K \leftarrow K + 1$; $OUT(I, K) \leftarrow J$; $L(J) \leftarrow CL + 1$; $SW \leftarrow 1$

End { If }

End { $J$ }

If $SW = 1$ then

Begin

$SW \leftarrow 0$; $CL \leftarrow CL + 1$

End { If }

Until $SW = 0$; { End Repeat }

Step 3. Find intersections

$K \leftarrow 0$;

For $I = 1$ to $MM$ do

For $J = 1$ to $MM$ do

if $A(I, J) = 'M'$ and $(J, I) = 'M'$ then

Begin

$K \leftarrow K + 1$; $INT(K, 1) \leftarrow I$; $INT(K, 2) \leftarrow J$

End { End If }

Step 4: Run algorithm–2 to discover any discrepancy between a view and the conceptual model for relationships between entities.
Repeat
\[ \text{CHK}(E, A, VW, NOD, NE, AA, RL, FLAG) \];
If \( FLAG = .FALSE. \) then
\( \) (Update information of conflicting views).
Until \( FLAG = .TRUE. ; \) \{ End Repeat \}

3.4.2. Algorithm–2: Check Discrepancies (\textit{CHK})

\textbf{Input} : (1) A set of entities \( E (MM) \).
(2) A matrix \( A (MM, MM) \) of relationships between entities.
(3) Arrays describing the views: \( VW (NN, MM, KK) \), \( NE (NN) \), \( NOD (NN, MM) \), \( AA (NN, MM) \), \( RL (NN, MM, MM) \).

\textbf{Output} : A \( FLAG \) which is true if there is no conflict; otherwise it is false.

\textbf{Steps} :

Step 1. Build binary relationship matrix
For \( I = 1 \) to \( MM \) do
For \( J = 1 \) to \( MM \) do
If \( A (I, J) \neq NULL \) then
\( B (I, J) \leftarrow 1 \)
else \( B (I, J) \leftarrow 0 ; \)

Step 2. Check for discrepancies
\( FLAG \leftarrow .TRUE. ; \)
For \( I = 1 \) to \( NN \) do
For \( JJ = 1 \) to \( NE (I) \) do
Begin
\( JN \leftarrow NOD (I, JJ) ; \)
For \( K = 1 \) to \( AA (I, JN) \) do
Begin
\( KN \leftarrow RL (I, J, K) ; \)
\{ Run algorithm–3 for nodes \( JN \) and \( KN \) \}
\( CON (B, JN, KN, T) ; \)
If \( T = 0 \) then Begin
\( FLAG \leftarrow .FALSE. ; \) Print \( I, JN, KN ; \)
End
End \{ End K \}
End \{ End JJ \}

3.4.3. Algorithm–3: Check Connection (\textit{CON})\textsuperscript{[16]}

\textbf{Input} : Binary relationship matrix \( B (MM, MM) \) and two node subscripts \( i \) and \( j \).
\{ \( B_{ij} = 1 \) if there is a direct connection between nodes \( E_i \) and \( E_j \) for \( i = j \)
and \( B_{ij} = 0 \) otherwise \}

\textbf{Output} : An indicator \( T \) representing the number of levels through which \( E_i \) and \( E_j \) may be connected.
Steps:

\[ T \leftarrow 0 \]

If \( B_{ij} = 1 \) then \( T \leftarrow 1 \)
else Begin

\[ Y \leftarrow 2; \]

Repeat

Find \( B^y = B^{y-1} \cdot B \).

if \( B_{ij} = 1 \) then \( T \leftarrow Y \)
else \( Y \leftarrow Y + 1; \)

Until \( B = [0] \) or \( B_{ij} = 1 \);

3.4.4. Algorithm-4: Calculate Response Time (CRT)

Input:
- (1) Logical model \( M_i \) in the form of:
  - (a) Network structure given by arrays \( E(MM) \), \( OUT(MM, MM) \), \( L(MM) \), \( AT(MM, KK) \)
  - (b) Information about views given by arrays \( FV(NN) \), \( FR(NN, MM) \), \( TYP(NN) \)
- (2) Physical model \( P_j \) in the form of arrays:
  - \( PO(MM) \), \( PR(MM, MM) \), \( INV(TA) \), \( RV(MM, MM) \)

Output:
- \( UT \) and \( RT \) (for the model) – \( TV \) and \( FJ \) (for entities).

Description: The algorithm calculates update and retrieval times for the given logical and physical models according to formulae given before. Note that \( AC_{K_i} \) (access time) is a function of the input parameters.

3.4.5. The Main Routine

Input:
- to the main routine are:
  - (1) A set of entities \( E(MM) \).
  - (2) A matrix \( A(MM, MM) \) of relationships between entities.
  - (3) Attributes of entities \( AT(MM, KK) \), and their sizes \( AS(MM, KK) \).
  - (4) Description of views (entities, attributes, and frequencies) \( VW(NN, MM, KK) \), \( FR(NN, MM) \), \( TYP(NN) \), \( FV(NN) \), \( NE(NN) \), \( NOD(NN, MM) \), \( RL(NN, MM, MM) \), \( AA(NN, MM) \).
  - (5) Associations between attributes \( DEP(G, X, W) \).

Description:
- (1) Call routine \( BLD \) to develop a logical model (\( BLD \) calls \( CHK \) and \( CON \)).
- (2) Vary the parameters of the physical model (\( PO, PR, INV, RV \)) and for each instance, calculate the response time (using routine \( CRT \)). Point out the worst physical states \( Z_w \).
- (3) Use \( DEP(G, X, W) \) to decompose some entities to normalized ones. This may be done on steps; and for each step, a different logical model is developed. Varying the physical parameters with each model, calculate the response time for each design case. Point out the set of worst design cases \( \{D_i\} \) with logical models \( \{M_i\} \) and physical states \( Z_r \).
(4) Excluding logical models \{ M_i \}, apply SPLIT/JOIN operations on the remaining models. For each new logical model, vary the physical parameters (excluding states \( Z_w \) and \( Z_t \)) and calculate the response time for each design case.

(5) The last step is to select the best design case among those generated in the previous step according to the criteria mentioned before.

4. A Case Study

Algorithms to run the given procedure were implemented. An experiment was carried out on a simple case study (modified version of the example of reference [6] as given below.

**Example**

The following entities were included (the related figure represents the number of records): S (STATE) - 40, P (PLANT) - 50, D (DEPARTMENT) - 1000, W (WORKSTATION) - 500, J (JOB) - 20, U (UNION) - 5, E (EMPLOYEE) - 100000, T (TASK) - 200000.

The relationships between entities are given in Table 1, where \( M \) denotes a many-to-one relationship. The attributes belonging to each entity are given in Table 2 (the initial of the attribute name is taken to be the abbreviation of the entity name itself).

<table>
<thead>
<tr>
<th>S</th>
<th>P</th>
<th>D</th>
<th>W</th>
<th>J</th>
<th>U</th>
<th>E</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
<td>M</td>
</tr>
</tbody>
</table>

The input for views is given in Table 3. Changes to physical model were limited to:

1. Either using pointers or embedded key values to implement hierarchical relationships,
2. Inverting some attributes. Changes to logical model were applied as given earlier. It should be noted that only one file organization was considered, namely the indexed-sequential organization. Results of retrieval and update times for different design cases are given in Table 4.
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SNO</td>
<td>State code</td>
<td>2</td>
<td>SNAME</td>
<td>State name</td>
<td>20</td>
</tr>
<tr>
<td>PNO</td>
<td>Plant number</td>
<td>2</td>
<td>PNAME</td>
<td>Plant name</td>
<td>20</td>
</tr>
<tr>
<td>PTOT1-5</td>
<td>Totals 1-5</td>
<td>10 (each)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UNO</td>
<td>Union number</td>
<td>2</td>
<td>UNAME</td>
<td>Union name</td>
<td>20</td>
</tr>
<tr>
<td>UT1-5</td>
<td>Totals 1-5</td>
<td>6 (each)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENO</td>
<td>Employee number</td>
<td>10</td>
<td>ENAME</td>
<td>Employee name</td>
<td>20</td>
</tr>
<tr>
<td>EADR</td>
<td>Address</td>
<td>30</td>
<td>EAREA</td>
<td>Area</td>
<td>2</td>
</tr>
<tr>
<td>ESTTS</td>
<td>Status</td>
<td>3</td>
<td>EDTB</td>
<td>Date of birth</td>
<td>6</td>
</tr>
<tr>
<td>EDTH</td>
<td>Hire date</td>
<td>6</td>
<td>EDIT</td>
<td>Termination date</td>
<td>6</td>
</tr>
<tr>
<td>ESAL</td>
<td>Salary</td>
<td>6</td>
<td>EDIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ETOT1-8</td>
<td>Totals 1-8</td>
<td>6 (each)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WNO</td>
<td>Work-station number</td>
<td>3</td>
<td>WTOT1</td>
<td>Total</td>
<td>6</td>
</tr>
<tr>
<td>JNO</td>
<td>Job number</td>
<td>2</td>
<td>JTIT</td>
<td>Job title</td>
<td>20</td>
</tr>
<tr>
<td>JTOT1</td>
<td>Total</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THR</td>
<td>Hours</td>
<td>4</td>
<td>TDAT</td>
<td>Date</td>
<td>6</td>
</tr>
<tr>
<td>DNO</td>
<td>Department number</td>
<td>2</td>
<td>DNAME</td>
<td>Department name</td>
<td>20</td>
</tr>
</tbody>
</table>

The resulting network is given in Fig. 1.

![Fig. 1. The resulting network.](image-url)
### Table 3. Different views.

<table>
<thead>
<tr>
<th>View no.</th>
<th>Type</th>
<th>View freq.</th>
<th>Entity freq.</th>
<th>Entities and attributes used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U</td>
<td>100</td>
<td>1</td>
<td>E(ENO, ENAME, EADR, EDTB, EDTH, ESAL) P(PNO), U(UNO), S(SNO)</td>
</tr>
<tr>
<td>2</td>
<td>U</td>
<td>100</td>
<td>2</td>
<td>E(ENO, ENAME, EDTH)</td>
</tr>
<tr>
<td>3</td>
<td>U</td>
<td>200000</td>
<td>2</td>
<td>E(ENO, ETOT1 - ETOT8), W(WNO), J(JNO), T(THR, TDAT), P(PNO, PTOT1)</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>1/75</td>
<td>5</td>
<td>U(UNO, UNAME, UT1 - UT5) E(ENO, ETOT1 - ETOT8), E(ENO, ENAME, EADR, ESAL, ETOT3, ETOT6, ETOT8)</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>1/300</td>
<td>40</td>
<td>S(SNO, SNAME) E(ENO, ENAME, ETOT8)</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>1/25</td>
<td>40</td>
<td>S(SNO, SNAME) P(PNO, PNAME, PTOT2)</td>
</tr>
<tr>
<td>7</td>
<td>R</td>
<td>1/25</td>
<td>50</td>
<td>P(PNO, PNAME, PTOT2) J(JNO, PTTT)</td>
</tr>
<tr>
<td>8</td>
<td>R</td>
<td>1</td>
<td>50</td>
<td>P(PNO, PNAME) E(ENO, ENAME) T(THR), J(JNO, JTTT), W(WTOT1)</td>
</tr>
<tr>
<td>9</td>
<td>R</td>
<td>1/25</td>
<td>50</td>
<td>P(PNO, PNAME) E(ENO, ENAME, EADR, ETOT5)</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>100</td>
<td>1</td>
<td>E(ENO, ENAME, ETOT3, ETOT5, ETOT6)</td>
</tr>
<tr>
<td>11</td>
<td>Q</td>
<td>100</td>
<td>1</td>
<td>E(ENO, ENAME, ESAL)</td>
</tr>
<tr>
<td>12</td>
<td>U</td>
<td>100000</td>
<td>2</td>
<td>E(ENO, ENAME, ESAL)</td>
</tr>
</tbody>
</table>

### Table 4. Response time / TB (in thousands of blocks).

<table>
<thead>
<tr>
<th>Time</th>
<th>Using pointers</th>
<th>Using key values</th>
<th>Inversion</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No more changes</td>
<td>No more changes</td>
<td>Normalization</td>
</tr>
<tr>
<td></td>
<td>Using pointers</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Retrieval</td>
<td>85</td>
<td>45</td>
<td>37</td>
</tr>
<tr>
<td>Update</td>
<td>135</td>
<td>35</td>
<td>23</td>
</tr>
</tbody>
</table>
5. Conclusion

The suggested data base design method can be considered as a step towards an optimal design methodology since it makes use of good features in different design approaches. The method starts by developing an initial logical model. The next step is to consider all possible physical models to determine the worst of them and discard its physical states. After that both logical and physical models are tuned according to proposed changes.

So far, tuning is done in consecutive steps, the order of which is done heuristically. Further work may include some investigation on varying the sequence of tuning steps and testing various physical representations.

List of Symbols

- $A_{jk}$: Relationship between entities $j$ and $k$.
- $AA_{ij}$: Number of entities related to entity $j$ in view $i$.
- $AC_{ij}$: Access time of a record of entity $j$ in view $i$.
- $AS_p$: Size of attribute $p$ in entity $j$.
- $AT_{jk}$: Attribute $k$ of entity $j$.
- $B_{jk}$: Binary relationship between entities $j$ and $k$.
- $BS$: Block size.
- $CF_j$: Clustering factor ($< 1$) for accessing more than one record directly in the same block (of entity).
- $D$: Complete design alternative.
- $DEP_{j|XW}$: Dependency between sets of attributes $X$ and $W$ in entity $j$.
- $E_j$: Name of entity $j$.
- $F_{jk}$: Joint frequency of reference of entities $j$ and $k$ in different views.
- $FR_{ij}$: Number of reference of entity $j$ in view $i$.
- $FV_i$: Frequency of view $i$.
- $INT$: Intersection between entities.
- $INV$: Indication of inverted attributes.
- $KK$: Maximum number of attributes of an entity.
- $L_j$: Level of node $j$.
- $M$: Logical model.
- $MM$: Number of entities.
- $NB_j$: Number of blocks of entity $j$.
- $NE_i$: Number of entities in view $i$.
- $NR_j$: Number of records of entity $j$.
- $NN$: Total number of views.
- $NOD_{ij}$: An entity $j$ in view $i$.
- $OUT_{jk}$: A node $k$ output from node $j$ in the network.
- $P$: Physical model.
- $PO_j$: Physical file organization of entity $j$.
- $PR_{jk}$: Method of physical implementation of relationship between entities $j$ and $k$.
- $q$: number of query views.
- $r$: number of retrieval views.
- $RL_{ijk}$: Entity $k$ related to entity $j$ in view $i$.
- $RS_j$: Record size of entity $j$.
- $RT$: Retrieval time.
- $RV_{jk}$: Indication of reverse access path between entities $j$ and $k$.
- $TA$: Total number of attributes.
An Integrated Approach to Data Base Design

$TB$  Time to access a block.
$TV_j$  Transport volume of entity $j$.
$TYP_i$  Type of view $i$.
$u$  Number of update views.
$VW_{ijk}$  Attribute $k$ of entity $j$ in view $i$.
$Z$  Set of physical states.

References

أسلوب متكامل لتصميم قواعد البيانات

محمد مصطفى حامد
قسم علوم الحاسب ، كلية العلوم ، جامعة البحرين
مدينة عيسى - البحرين

المتخلص: يقدم هذا البحث طريقة جديدة لتصميم قواعد البيانات . تعتمد هذه الطريقة على الاستفادة من الخواص الجيدة المتوافرة من الطرق السابقة . بدأ الدراسة بعمل تصميم مبدئي بناء على متطلبات مستعملين قاعدة البيانات . ثم يتم ضبط التصميم بعمل تغييرات متابعة في كل النموذجيين المتلقفي الطبيعي . ويستخدم زمن الاستجابة كمقياس لمدى كفاءة التصميم .

ويشتمل البحث على خوارزميات لتنفيذ الطريقة المفترضة . وقد تركت حريات اختيار التصميم المناسب لل مدير قاعدة البيانات الذي يمكن للمناوج المشترك من الخوارزميات المعروفة مساعدة في اتخاذ القرار المناسب .