JKAU: Eng. Sci., **vol. 5**, pp. 31-44 (1413 A.H. / 1993 A.D.)

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[1-3], relationships between entities[4-5], logical record access[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[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

$$C \twoheadrightarrow T \qquad \text{and} \qquad C \twoheadrightarrow X$$

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 correponding 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 Fry[6] 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 other[11]. 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 convenient[12] 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 anomalies[13] 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 $D_{i,j}$ 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-

sponse 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 j (excluding those models that contain the set of physical states Z_w).
(5) Select the worst designs $\{D_t\}$ for logical models $\{M_t\}$ and physical models $\{P_t\}$ with a set of physical states Z_t.
(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 j (excluding those models that contain the sets of physical states Z_w and Z_t).

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 $D_{c,h}$ for logical model M_c and physical model P_h.

3.3 Formulae of Calculation

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

FV_i frequency of view i.
AC_{ij} access time of a record of entity j in view i.
FR_{ij} number of references of entity j in view i.
NE_i number of entities in view i.
u number of update views.
r number of retrieval views.
q number of query views.
NR_j number of records of entity j.
NB_j number of blocks of entity j.
TB time to access a block.
AS_{jk} size of attribute k in entity j.
NN total number of views.
C_1 and C_2 are constants.

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 $FR_{ij} = NR_j$ and access is sequential then $AC_{ij} \cdot FR_{ij} = NB_j \cdot TB$ otherwise it is equal to $FR_{ij} \cdot TB \cdot CF_j$

$$(3)\ TV_j = \sum_i FV_i \cdot FR_{ij} (AS_{11} + AS_{12} + \cdots + AS_{1k})$$

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

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 ← 0;
    For I = 1 to MM do
      if A (I, J) ≠ NULL then SW ← 1;
    If SW = 0 then L (J) ← 1;
  End { J }
```

Step 2. Assign levels to other nodes

```
CL ← 1; SW ← 0;
Repeat
  For I = 1 to MM do
    Begin
      K ← 0;
      If L (I) ≠ CL then
        For J = 1 to MM do
          If A (I, J) ≠ NULL and not (A (I, J) = = 'M' and A (J, I) = 'M') then
            Begin
              K ← K + 1; OUT (I, K) ← J; L (J) ← CL + 1; SW ← 1;
            End; { If }
  End { I }
  If SW = 1 then
    Begin
      SW ← 0; CL ← CL + 1
    End; { If }
  Until SW = 0; { End Repeat }
```

Step 3. Find intersections

```
K ← 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 ← K + 1; INT (K, 1) ← I; INT (K, 2) ← 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
  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 (*CHK*)

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*).

Output : A *FLAG* which is true if there is no conflict; otherwise it is false.

Steps :

Step 1 . Build binary relationship matrix
For *I* = 1 to *MM* do
  For *J* = 1 to *MM* do
    If *A* (*I, J*) <> *NULL then B* (*I, J*) ← 1
                  else *B* (*I, J*) ← 0;

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

3.4.3. Algorithm–3: Check Connection (*CON*)[16]

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 }

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 $B^Y = B^{Y-1} . B$.
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_{Kl} (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 $\{D_t\}$ with logical models $\{M_t\}$ and physical states Z_t.

(4) Excluding logical models $\{M_l\}$, 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 (WORK-STATION)–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 1. Relationships between entities.

	S	P	D	W	J	U	E	T
S		M						
P			M		M			
D				M				
W					M		M	M
J		M		M			M	M
U							M	
E				M	M			M

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 2. Attributes of entities.

Attr.	Meaning	Char.	Attr.	Meaning	Char.
SNO	State code	2	SNAME	State name	20
PNO PTOT1 - 5	Plant number Totals 1 - 5	2 10 (each)	PNAME	Plant name	20
UNO UT1 - 5	Union number Totals 1 - 5	2 6 (each)	UNAME	Union name	20
ENO EADR ESTTS EDTH ESAL ETOT1 - 8	Employee number Address Status Hire date Salary Totals 1 -8	10 30 3 6 6 6 (each)	ENAME EAREA EDTB EDTT	Employee name Area Date of birth Termination date	20 2 6 6
WNO	Work-station number	3	WTOT1	Total	6
JNO JTOT1	Job number Total	2 6	JTIT	Job title	20
THR	Hours	4	TDAT	Date	6
DNO	Department number	2	DNAME	Department name	20

The resulting network is given in Fig. 1.

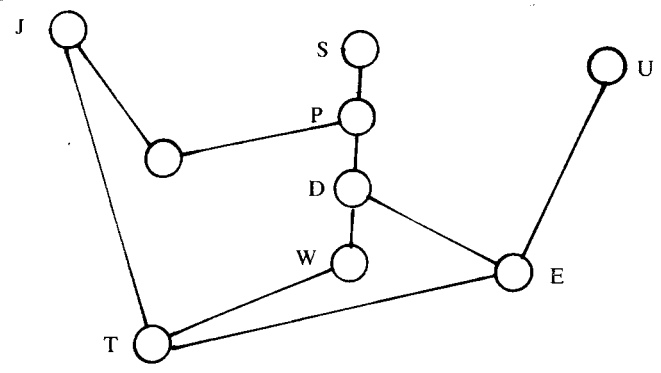

FIG. 1. The resulting network.

TABLE 3. Different views.

View no.	Type	View freq.	Entity freq.	Entities and attributes used
1	U	100	1	E (ENO, ENAME, EADR, EDTB, EDTH, ESAL) P (PNO), U (UNO), S (SNO)
2	U	100	2	E (ENO, ENAME, EDTT)
3	U	200000	2	E (ENO, ETOT1 - ETOT8), W (WNO), J (JNO), T (THR, TDAT), P (PNO, PTOT1)
4	R	1/75	5 100000	U (UNO, UNAME, UT1 - UT5) E (ENO, ENAME, EADR, ESAL, ETOT3, ETOT6, ETOT8)
5	R	1/300	40 100000	S (SNO, SNAME) E (ENO, ENAME, ETOT8)
6	R	1/25	40 50	S (SNO, SNAME) P (PNO, PNAME, PTOT2)
7	R	1/25	50 1000	P (PNO, PNAME, PTOT2) J (JNO, PTIT)
8	R	1	50 100000 200000	P (PNO, PNAME) E (ENO, ENAME) T (THR), J (JNO, JTIT), W (WTOT1)
9	R	1/25	50 100000	P (PNO, PNAME) E (ENO, ENAME, EADR, ETOT5)
10	R	100	1	E (ENO, ENAME, ETOT3, ETOT5, ETOT6)
11	Q	100	1	E (ENO, ENAME, ESAL)
12	U	100000	2	E (ENO, ENAME, ESAL)

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

Time	Using pointers	Using key values			
		No more changes	Inversion		
			No more changes	Normalization	+ SPLIT / JOIN
Retrieval	85	45	37	37	33
Update	135	35	23	22	20

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_{jp}	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 j).
D	Complete design alternative.
DEP_{jXW}	Dependency between sets of attributes X and W in entity j.
E_j	Name of entity j.
FJ_{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.

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

[1] **Zaniolo, C.** and **Melkanoff, M.A.,** On the design of relational data base schemata, *ACM-TODS,* **6**(1): 1-47 (1981).

[2] **Bernstein, P.A.,** Synthesizing third normal form relations from functional dependencies, *ACM-TODS,* **1**(4): 277-298 (1976).

[3] **Beeri, C.** and **Kifer, M.,** An integrated approach to logical design of relational data base schemes, *ACM-TODS,* **11**(2): 134-136 (1986).

[4] **Chen, P.P.,** The entity-relationship model: Towards a unified view of data, *ACM-TODS,* **1**(1): 9-36 (1976).

[5] **Ng, P.A.,** Further analysis of a entity-relationship approach to data base design, *IEEE Trans. on Softw. Eng.*, **SE-7**(1): 85-99 (1981).

[6] **Teorey, T.J.** and **Fry, J.P.,** The logical record access approach to data base design. *ACM Computing Surveys,* **12**(2): 179-211 (1980).

[7] ————, A practical approach to transforming extended ER diagrams into the relational model, *Inf. Sciences* (USA), **42**(2): 167-186 (1987).

[8] **Yang, C.C.,** *Relational Data Bases,* Prentice-Hall, USA, pp. 145-180 (1986).

[9] **Hubbard, G.U.,** *Computer-Assisted Data Base Design,* New York, N.Y.: Van Nostrand-Reinhold (1981).

[10] **Henver, A.R.** and **Yao, S.B.** Network data base design methods, *In:* **S.B. Yao (ed.)** *Principles of Data Base Design,* **Vol. 1,** Prentice-Hall, USA, pp. 294-323 (1986).

[11] **Jih, W.J.K.,** The effects of relational and entity-relationship models on query performance of end users, *Int. J. Man-Mach. Stud.,* **(3)** Sept.: 257-267 (1989).

[12] **Hamed, M.,** *Designing Multi-Level Schemas and Access Methods for Relational Data Bases,* Doctoral Dissertation, Ain Shams University, Cairo (1983).

[13] **Lozinski, E.L.,** Performance Consideration in Relational Data Base Design, *Proc. Int. Conf. on Data Bases: Improving Usability and Responsiveness, Israel,* Academic Press, London, pp. 273-294 (1978).

[14] **Hamed, M.** and **El-Karaksy, R.,** A decomposition procedure for the design of normalized relational schemata, *Computer Journal,* Vol. 35, pp. A329-A333 (1992).

[15] **Wiederhold, G.,** *Data Base Design,* McGraw-Hill, New York, pp. 86-162 (1979).

[16] **Ghost, G.P.,** *Data Base Organization for Data Management,* Academic Press, pp. 49-58 (1977).

أســـلوب متكامـــل لتصميم قواعـــد البيانات

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

المستخلص . يقـدم هذا البحث طريقة جديدة لتصميم قواعد البيانات . تعتمد هذه الطريقة على الاستفادة من الخواص الجيدة المتوافرة من الطرق السابقة . تبدأ الطريقة بعمل تصميم مبدئي بناء على متطلبات مستعملي قاعدة البيانات . ثم يتم ضبط التصميم بعمل تغييرات متتابعة في كلا النموذجين المنطقي والطبيعي . ويستخدم زمن الاستجابة كمقياس لمدى كفاءة التصميم .

ويشتمـل البحث على خوارزميات لتنفيذ الطريقة المقترحة . وقد تركت حرية اختيار التصميم المناسب لمدير قاعدة البيانات الذي يمكن للنتائج المستخرجة من الخوارزميات المعطاة مساعدته في اتخاذ القرار المناسب .