M.TECH.(COMPUTER SCIENCE) DISSERTATION

DESIGN OF A DYNAMIC FEDERATED DATABASE.

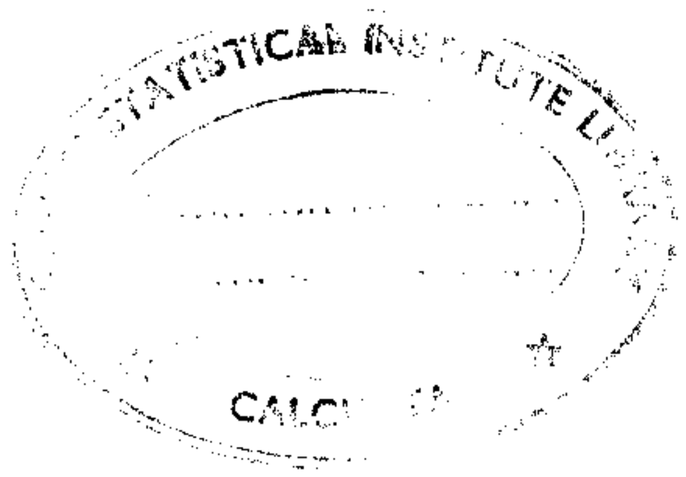
A DISSERTATION SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE M.TECH.(COMPUTER SCIENCE) DEGREE OF THE INDIAN STATISTICAL INSTITUTE.

submitted by :

Nabin Chandra Daskhan.

under the supervision of

Prof. Aditya Bagchi.



INDIAN STATISTICAL INSTITUTE
203 , BARRACKPUR TRUNCK ROAD
CALCUTTA - 700 035

CONTENTS :

PROBLEM :

TO FORM THE DYNAMIC FEDERATED DATABASE STRUCTURE FROM A NUMBER OF LOCAL RELATIONAL DATABASES. THE FEDERATED DATABASE SCHEMA WILL SUPPORT THE FUNCTIONS OF EACH PARTICIPATING DATABASE AS WELL AS THE SERVICES THAT CAN BE OBTAINED FROM THE FEDERATED DATABASE. IF NECESSARY A PARTICIPATING DATABASE MAY BE INCLUDED OR WITHDRAWN. THE FEDERATED DATABASE SHOULD BE USED FOR ANSWERING THE GLOBAL QUERIES.

****

INTRODUCTION :

A federated database system (FDBS) is a collection of cooperating databases (DBSs) that are autonomous. Here we will develop a methodology for building such a FDBS . The component DBSs are integrated to varying degrees . One of the significant aspects of a FDBS is that a component DBS can continue its local operations and at the same time participate in a federation . A key characteristic of the federation is the cooperation among the independent systems. At present there is no ideal model for federation. To allow controlled sharing while preserving the autonomy of the component DBSs and continued execution of the existing applications an FDBS supports two types of operations : local and global. Global operations involve data access using the FDBS and may involve data managed by multiple component DBSs. Local operations may be submitted to a component DBS directly.The proposed structue is dynamic in nature where a new database can easily be added to or withdrawn from the federation.

We normally form a single database for an enterprise (an enterprise is a reasonably self_contained

commercial, scientific,or other organization; e.g., manufacturing company ,bank,hospital,university etc.). .The database contains all the relevant data of the enterprise.In a relational system we can update the schema by adding new attribute(s) to a relation or by forming a new relation according to the requirement.Update also includes deletion of attrbute(s) or relation(s).In this way an enterprise,however complicated ,may be modelled by a single database. The above approach has the following limitaions :

1. A big enterprise in general consists of a number of subenterprises. Though these subenterprises contain overlapping information (i.e. some relations are fully or partly common in the sense that they refer to same set of attributes)each such subenterprise is large enough to be considered as a separate database.
2. Each subenterprise has users who are interested only in that subenterprise and may not be interested in other subenterprises.
3. So each such database modelling may need local autonomy for efficent maintenance and query processing.
4. Subenterprises should act independent of each other so that in the event of error/crash in one subenterprise local operations in other subenterprises should not get affected. If we keep a single database for the whole enterprise,we cannot provide such facility.
5. Data may be distributed among multiple databases. Benefits of data distribution such as increased availability as well as improved access time is evident.

SYSTEM :

As a remedial measure a dynamic conceptual multidatabase has been proposed. The proposed system maintains aset of local databases one for each subenterprise. Each such local database is capable of handling a local query involving its own relations only. A global query which may involve more than one local database ,may be decomposed into a number of local queries; each accessing a separate local database. The proposed system provides an integrated view of the local databases and an user can make a global query. This query will then be analyzed and decomposed and will be distributed to the different local databases. Later answers from the local databases will be merged to get the answer for the global query. The integration provides just a conceptual multidatabase and does not contain any data. This is a dyanamic structure where a new schema may be added or an existing one may be deleted.

THE TECHNIQUE TO REALISE THE MULTIDATABASE IS DESCRIBED BELOW :

To incorporate a number of databases in a conceptual multidatabase a conteptual integration is required over the databases. Unless the schemas are represented in the same model ,analyzing and comparing their schema objects are extremely difficult. It is important to note that comparision of schema objects is primarily guided by their semantics and not by their syntax . Analyzing and comparing schema objects are followed by specifying the interrelationships among the schema objects. In the relational model we assume that basic relationships may exist between relations .A attributes of two relations can be related if and only if the relations themselves be related . Two attributes a1 and a2 of two relations may b related by :

a1 is_synonym_of a2 or

a2 is_disjoint_to a2 .

Determination such relationships can be time consuming and tedeous . (Seth and Larsen argue that if each schema has 100 entity types with an average of five attributes per entity type then 250,000 pairs of attributes must be compared). This task cannot be automated and we may need to depend on heuristics to identify a few attribute pairs that may be potentially related (synonym).A completely automatic schema integration process is not possible is because it requires all the semantics of the schema to be specified. The current semantic data models are unable to capture a real world state completely It will be necessary to capture much more information than is typically available in a schema . Also there can be multiple interpretation of a real world.

The program developed takes a new incoming database as input and incorporate it into the existing multidatabase. The new database is formed in an interactive session with the user. The program forms an SQL file that contains SQL commands which when executed produces the new database. After the new database is formed, conceptual integration begins. If the database is the first one then no integration is to be done and it is directly entered into the multidatabase. Otherwise integration is to be done. Before giving the integration process let us discuss the structure of the multidatabase. The multidatabase consists of two files :-

A. Attribute_file(Attb file) and

B. Global_attribute_header_file(HAHD file).

A. Attb file :-

The Attb file is actually a synonym lists file. For each globalattribute (an attribute is global if it maintains the same meaning throughout the multidatabase) there exists a

list of synonyms i.e. attributes of the participating databases having the same meaning. Each synonym entry corresponds to a record in the Attb file. Each such record consists of the following fields :-

(dirty_bit,dbase,relation,attb,unit,conv_factor,constr_type, constr_no , constr_file_position,next,prev,header_pointer)

giving the following information :

dirty_bit :

whether the record is valid or not.

dbase :

the name of the participating database containing the synonym.

relation :

name of the relation in the above database containing the synonym.

Attb :

local name of the synonym.

unit :

unit of the synonym,if any (eg. cm.,foot,kg etc.).

conv_factor :

global to local transformation factor,if any.

constr_type,constr_no,constr_file_position :

The three integer fields encode the constraints information of the synonym.

constr_type ▪ 0 means no constraints.

▪ 1 means that it is not null.

▪ 2 means that constraint is of set type.

▪ 3 means constraint is of range type.

e.g. set type constraint :

sex in (male,female) .

day in (sun,mon,....,sat).

range type constraint :

2000 <= salary <= 10000.

when the constr_type is 2 or 3 then only other two constraint fields are relevent. The field constr_no gives the total number of such constraints imposed on the synonym. e.g., for the constraint sex in (male,female) the constraint number is 2. The actual constr_information is stored in a file called Con file. The constraint field gives the starting position of the constraint information in the Con file corresponding to the synonym.

next :

pointer to the next synonym record.

prev :

pointer to the prev synonym record.

header_pointer :

pointer to the global attribute in the header file.

At the top of the attribute file we keep a pointer to the next free record where the insertion of a record may be done. The free records themselves make a link list of free records. The list is dynamically updated as soon as an insertion or deletion is done.

If a record is deleted then the record is added to the free link list, the prev and next fields of its concerned records are suitably adjusted. If both the prev and next pointers of a record are null then it is the only synonym record present in the file. So when such a record is deleted then no synonym exists for it in the file. Then the global attribute in the header file referring to the synonym is also deleted. The prev and the header pointers are kept for this updation purpose.

B. Global_attribute_header_file(GAHD file). This file stores the global attributes along with a pointer referring to the first synonym in the Attb file. Each record in the file

corresponds to a global attribute. Such a record contains the following fields :

(dirty_bit,record_no,attb_name,ptr) giving the following information :-

dirty_bit : whether the record is a valid one.
record_no : the record number of the current record.
Attb_name : global attribute name.
ptr : pointer to the first synonym in the attribute file.

For each new global attribute in the multidatabase one entry is made in the header file. Entry position of an incoming attribute is determined by a near perfect hash function and a collision resolution method attribute to the attribute name. The hash table size is to be chosen on the basis of expected maximum number of attributes that may have to be stored. In case the expected limit is crossed in due course , we have to increase the hash table size and rehash the entire hash table.

The above two files along with the Con file forms the multidatabase. Apart from them we also maintain two more files :-

C. Relation_header file and

D. Relation file.

These two files are not used for query processing but used for integration of the incoming database with the existing multidatabase.

C. Relation_header file :

The file keeps the global relations of the multidatabase along with a pointer to the first relation in the relation file connected by some relationship. Each record of the file is of the form :-

(dirty_bit,record_no,pointer,rel,attbno,attributes[])

keeping following information :

dirty_bit : whether the record is valid one.

record_no : giving its position in the file.

pointer : points to the first related relation in the relation file.

rel : global relation name.

attbno : number of global attributes in the relation.

attributes[] : set of global attribute names.

D. Relation file :-

This file keep lists where each list contains the relations having some relationship with the global relation. Each list consists of the records of the following form :

(dirty_bit,dbase,rel,relationship,next,prev,head_pointer)

giving the following information :

dirty_bit : whether the record is a valid one.

dbase : name of the containing database.

relation : local name of the related relation.

relationship : name of the relationship with the global relation. It may be self ,subset,overlap etc.

next and prev : pointers to the next and previous related record.

head_pointer : points to the global relation in header file to which it is related.

We also maintain a subfile within the relation header file having each record of the form (relation_name,position) giving

the following information :

relation_name : Name of the global relation formed at the time of integration process.

position : pointer to the header file record

corresponding to the above relation name.

This subfile is mainly meant for the user who is going to integrate an incoming database. The user is shown the intermediate relations formed along with the attributes to reach the existing multidatabase.

The integration process is as follows :-

The user (integrater) is shown the existing global relations formed. For each incoming relation the user is interactively asked whether it is related to an existing global relation or a new one. If it is new then a global entry is made corresponding to it.

Otherwise,the relation file is updated by incorporating it; if required, some new global attributes are inserted in the global relation record. Method of updation (updation and deletion) of relation file and relation_header file are similar to the Attb file and Attb_header file. For each global attribute added corresponding entries are made in the attribute file. The rest of the attributes of the incoming relation are synonyms of some global attributes. Hence they are inserted into the corresponding position of the synonym list. To determine the synonym pairs the interaction with the user may be required to differentiate between homonyms and homonyms and also to get synonyms which are not homonyms. After the integration process is completed,the updated multidatabase is formed. The deletion operations in the multidatabase is more involving than insertion. Deletion may be of three types :-

1. Deletion of database :

In this case all the entries corresponding to the database are deleted. The multidatabase file scanned throughout to find the records corresponding to the database. These records are made invalid by resetting the dirty bit and also the records are

added to free record list.

ii. Deletion of a relation :

In this case both the database name and relation name are to be matched to delete a record.

iii. Deletion of an attribute :

Here the database name,relation name and attribute name are to be matched to declare a record to be invalid.

QUERY PROCESSING :

Query processing involves converting a query against a federated schema into several queries against the participating schemas. Query processing in FDBS can be divided into global processing and local processing. Global processing and optimization related to processing a query or transaction submitted by a federation user ,called global transaction and it is divided into multiple subtransactions .Local query processing and optimization relate to processing a local transaction on a single component DBS. For global optimization we have followed a simple approach -
transform the global transaction into the minimum number of possible local transaction.

The multidatabase will support the query of the form :

SELECT attribute_list
WHERE predicate_expression.

The predicate expresselon contains attributes and boolean/relational operators. From this query we extract all the attributes referred. These attributes are all global in

the multidatabase and are all distinct. Now our task is :

1. To distribute the query to the local databases and
2. Answers obtained from the local databases are merged to give the complete answer to the user.

1. To distribute the query to the local databases, we have to identify the local databases. For each global attribute referred in the query we get a set of databases from the synonym list for the global attribute. Only those databases for which attribute constraints matche should be in the set. For this purpose we may have to consult the Con file (when constraint type is 2 or 3) Then for optimum query processing we need to know the minimum number of databases that cover all the attributes referred. It is the well known set cover problem which is NP complete . We can use some hearistic method which gives an approximate solution in polynomial time. But as the number of participating databases is not much (expected upper limit <= 10) we proceed to get the suboptimal solution using greedy algorithm.One such algorithm is as follows :-

First see whether any single database can cover all the attributes. If so then return . Otherwise see whether any two databases taken at a time can cover all the attributes or not .and .....so on. More formally the algorithm runs as follows:

Let a minumum of r number of databases out of total n databases can cover all the attributes in the query. There are nCr such possible distinct combinations. Find these combinations and proceed to check whether any one of them can give the desired set. If such a set is obtained then our job is done. Otherwise increase r by one and repeat the above procedure. Initially r takes the value 1 and in the worst case we may have to increase r upto n.

The beauty of the above algorithm is that the first answer the algorithm obtain is the desired set and the algorithm terminates. So the average complexity of the algorithm is reasonably acceptable.

A predicate_expression(P) is of the form :-

P = x1 bop x2 bop x3 .....bop xn.

where xi is an expression containing attribute(s) and arithmetic operators. bop is some boolean operator.

It is assumed that each xi contain attributes which are available in a single database. Otherwise we cannot impose predicate conditions in the local database level. But attributes in xi and xj (i ≠ j) may be in two different databases.

Though the user query does not conform to SQL syntax ,the query for a local database must be in SQL which is of the form :

SELECT attribute_list

FROM relation_list

WHERE predicate_expression.

We get the relation_list which is not present in the user query from the synonym lists of the attributes referred.

EXAMPLE :

We consider the following three participating databases :

1. Factory :

employee (emp-no,emp-name,address,basic,dept-no);

dept (dept-no,dept-name,manager,budget);

dependent (dep-name,emp-no,relation);

2. School :

teacher(t-no,t-name,address,basic,subject);

student(roll-no,class,st-name,guar-no);

3. Hospital :

doctor (d-no,d-name,address,speclz);

patient (p-no,p-name,p-address,d-no);

Here the 1st incoming database is the factory which is directly entered into the multidatabase. The contents of the multidatabase now contains the following entries :

in GAHD file ->

(emp_no,emp-name,address,basic,dept_no,dept_name,manager,budget,
dep-name,relation)

in Attb file ->

(emp_no,emp-name,address,basic,dept_no
;dept_no,dept-name,manager,budget;
dep-name,emp_no,relation).

in Relation header file ->

(employee,dept,dependent);

in Relation file ->

(employee,dept,dependent);

The next database to be incorporated is 'school'. It has two relations : teacher and student . Teacher is_a employee and they share the following synonyms :

(emp_no,t_no)

(emp-name,t-name)

(address,address)

(basic,basic)

The attribute subject in teacher becomes a new global attribute.

Also we assume that student is_a dependent and they share the following synonyms :-

(st-name,dep-name)

(emp_no,guar_no)

The attributes roll_no and class in student becomes new global attributes.

So the multidatabase is updated as follows :

in GAHD file ->

(emp_no, emp-name, address, basic, dept_no, dept_name, manager, budget,dep-name,relation, subject,roll_no,class)

in Attb file ->

(emp_no,emp-name,address,basic,dept_no

;dept_no,dept_name,manager,budget;

dep-name,emp_no,relation;

t_no,t-name,address,basic,subject;

roll_no,st_name,class,guar_no).

with the synonyms forming linked list.

in Relation header file ->

(employee,dept,dependent);

in Relation file ->

(employee,dept,dependent,teacher,student);

with related relations forming linked list.

The next incoming database is hospital having relations doctor and patient. We have doctor is_a employee having the following synonym pairs :

(d_no,emp_no)

(d-name,emp-name)

(address,address)

and speclz becomes a new global attribute.

The patient is a new global relation.

So the multidatabase is updated as follows :

in GAHD file ->

(emp_no, emp-name, address, basic, dept_no, dept_name, manager, budget,dep-name,relation, subject,roll_no,class,speclz,p-name,p_no)

in Attb file ->

(emp_no,emp-name,address,basic,dept_no

;dept_no,dept_name,manager,budget;

dep-name,emp_no,relation;

t_no,t-name,address,basic,subject;

roll_no,st_name,class,guar_no

d_no,d-name,address,speclz;

p_no,p-name,address,_no).

with the synonyms forming linked list.

in Relation header file ->

(employee,dept,dependent,patient);

in Relation file ->

(employee,dept,dependent,teacher,student,doctor,patient);

Please See OUTPUT for final federated database structure.

REFERENCES :

1. A.P.Seth & j.A. Larson,"Federated Database Systems for managing Distributed Heterogeneous and Autonomous Databases",ACM Computing Surveys, vol. 22, No. 3,PP 183-236, sept. 1990.

2. J.A.Larson,S.B.Navathe & R.Elmasri,"A Thory of Attribute Equivalence in Databases with Application to Schema Integration", IEEE Trans. on Software Engg., vol. SE 15, No.4, PP 449-463,April 1989.

3. M.T. Ozsu & P. Valduriez,"Principles of Distributed Database Systems",Prentice Hall,1991.

OUTPUT

****

Attb file :-

record : (dirty_bit,database,relation,attribute,domain,unit,cons_type,cons_size,
cons_location and pointers for next record,prev record,global record)
** the field conv_factor is not considered here

27

35	factory	employee	emp_no	integer	null	3	0	0	-1	12	43
35	factory	employee	emp_name	char	null	0	0	0	-1	13	87
35	factory	employee	address	char	null	0	0	0	-1	14	49
35	factory	employee	basic	integer	Rs.	3	0	1	-1	15	88
35	factory	employee	dept_no	integer	null	3	0	2	-1	5	79
35	factory	dept	dept_no	integer	null	3	0	0	4	-1	79
35	factory	dept	dept_name	char	null	0	0	0	-1	-1	22
35	factory	dept	manager	char	null	0	0	0	-1	-1	46
35	factory	dept	budget	integer	Rs.	3	0	3	-1	-1	84
35	factory	dependent	dep_name	char	null	0	0	0	-1	-1	64
35	factory	dependent	emp_no	integer	null	1	0	0	0	-1	43
35	factory	dependent	relation	char	null	0	0	0	-1	-1	11
35	school	teacher	t_no	integer	null	3	0	0	0	17	43
35	school	teacher	t_name	char	null	0	0	0	1	-1	87
35	school	teacher	address	char	null	0	0	0	2	21	49
35	school	teacher	basic	integer	Rs.	3	0	0	3	-1	88
35	school	teacher	subject	char	null	0	0	0	-1	-1	80

35	school	student	guar_no	integer	null	0	0	0	12	20	43
35	school	student	roll_no	integer	null	3	0	0	-1	-1	89
35	school	student	class	integer	null	3	0	0	-1	-1	90
35	hospital	doctor	d_no	integer	null	3	0	0	17	-1	43
35	hospital	doctor	address	char	null	0	0	0	14	-1	49
35	hospital	doctor	speclz	char	null	0	0	0	-1	-1	17
35	hospital	patient	p_no	integer	null	0	0	0	-1	-1	50
35	hospital	patient	p_name	char	null	0	0	0	-1	-1	83
35	hospital	patient	address	char	null	0	0	0	-1	-1	51
35	hospital	patient	d_no	char	null	1	0	0	-1	-1	7
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	28	-1	-1
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	29	-1	-1
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	30	-1	-1
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	31	-1	-1
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	32	-1	-1
...		...	...	...	...			...			
...		...	...	...	...			...			
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	99	-1	-1
-1	xxx	xxx	xxx	xxx	xxx	-1	-1	-1	100	-1	-1

file : GAHD file :-

record : (dirty_bit,own record no,attb_name,first synonym pointer)

-1	0	xxx	-1
-1	1	xxx	-1

-1	2	xxx	-1
-1	3	xxx	-1
-1	4	xxx	-1
-1	5	xxx	-1
-1	6	xxx	-1
35	7	d_no	26
-1	8	xxx	-1
-1	9	xxx	-1
-1	10	xxx	-1
35	11	relation	11
-1	12	xxx	-1
-1	13	xxx	-1
-1	14	xxx	-1
-1	15	xxx	-1
-1	16	xxx	-1
35	17	spec1z	22
-1	18	xxx	-1
-1	19	xxx	-1
-1	20	xxx	-1
-1	21	xxx	-1
35	22	dept_name	6
-1	23	xxx	-1
-1	24	xxx	-1
-1	25	xxx	-1
-1	26	xxx	-1

-1	27	xxx	-1
-1	28	xxx	-1
-1	29	xxx	-1
-1	30	xxx	-1
-1	31	xxx	-1
-1	32	xxx	-1
-1	33	xxx	-1
-1	34	xxx	-1
-1	35	xxx	-1
-1	36	xxx	-1
-1	37	xxx	-1
-1	38	xxx	-1
-1	39	xxx	-1
-1	40	xxx	-1
-1	41	xxx	-1
-1	42	xxx	-1
35	43	emp_no	20
-1	44	xxx	-1
-1	45	xxx	-1
35	46	manager	7
-1	47	xxx	-1
-1	48	xxx	-1
35	49	address	21
35	50	p_no	23
35	51	address	25

-1	52	xxx	-1
-1	53	xxx	-1
-1	54	xxx	-1
-1	55	xxx	-1
-1	56	xxx	-1
-1	57	xxx	-1
-1	58	xxx	-1
-1	59	xxx	-1
-1	60	xxx	-1
-1	61	xxx	-1
-1	62	xxx	-1
-1	63	xxx	-1
35	64	dep_name	9
-1	65	xxx	-1
-1	66	xxx	-1
-1	67	xxx	-1
-1	68	xxx	-1
-1	69	xxx	-1
-1	70	xxx	-1
-1	71	xxx	-1
-1	72	xxx	-1
-1	73	xxx	-1
-1	74	xxx	-1
-1	75	xxx	-1
-1	76	xxx	-1

-1	77	xxx	-1
-1	78	xxx	-1
35	79	dept_no	5
35	80	subject	16
-1	81	xxx	-1
-1	82	xxx	-1
35	83	p_name	24
35	84	budget	8
-1	85	xxx	-1
-1	86	xxx	-1
35	87	emp_name	13
35	88	basic	15
35	89	roll_no	18
35	90	class	19
-1	91	xxx	-1
-1	92	xxx	-1
-1	93	xxx	-1
-1	94	xxx	-1
-1	95	xxx	-1
-1	96	xxx	-1
-1	97	xxx	-1
-1	98	xxx	-1
-1	99	xxx	-1

RELATION FILE :-

```
record : (dirty_bit,database,relation,relationship and
          pointers for next,prev and global record )

 7

35 factory     employee    self          -1     3    10

35 factory     dept        self          -1    -1     4

35 factory     dependent   self          -1     4    47

35 school      teacher     is_a           0     5    10

35 school      student     is_a           2    -1    47

35 hospital    doctor      is_a           3    -1    10

35 hospital    patient     self          -1    -1    11

-1 xxx         xxx         xxx            8    -1    -1

-1 xxx         xxx         xxx            9    -1    -1

-1 xxx         xxx         xxx           10    -1    -1

   ...         ...         ...              ...

   ...         ...         ...              ...


-1 xxx         xxx         xxx           99    -1    -1

-1 xxx         xxx         xxx          100    -1    -1
```

RELATION HEADER FILE :-

```
record : (dirty bit,own record no,pointer to relation file,relation name, attribute

 3 employee     10 dept          4 dependent     47 patient      11 xxx

-1     0    -1 xxx          -1 xxx          xxx          xxx          xxx          xxx

-1     1    -1 xxx          -1 xxx          xxx          xxx          xxx          xxx

-1     2    -1 xxx          -1 xxx          xxx          xxx          xxx          xxx
```

-1	3	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
35	4	1	dept	3	dept_no	dept_name	manager	budget	xxx
-1	5	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	6	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	7	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	8	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	9	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
35	10	5	employee	6	emp_no	emp_name	address	basic	
			dept_no	subject		speclz	xxx	xxx	xxx
35	11	6	patient	3	p_no	p_name	address	d_no	xxx
-1	12	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	13	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	14	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	15	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	16	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	17	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	18	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	19	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	20	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	21	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	22	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	23	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	24	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	25	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	26	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx

-1	27	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	28	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	29	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	30	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	31	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	32	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	33	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	34	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	35	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	36	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	37	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	38	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	39	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	40	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	41	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	42	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	43	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	44	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	45	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
-1	46	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
35	47	4	dependent	4	dep_name	emp_no	relation	roll_no	class
-1	48	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx
	...		...		...	...	...	...	....
	...		...		...	...	...	...	....
-1	98	-1	xxx	-1	xxx	xxx	xxx	xxx	xxx

-1 99 -1 xxx -1 xxx xxx xxx xxx xxx

CONSTRAINTS FILE :-

```
3
1 1000
1000 9000
1 10
5000 9000
```

SQL FILE :-

```
create schema filename factory
create table employee
(
emp_no      integer ,
emp_name    char ,
address     char ,
basic       integer ,
dept_no     integer
check ( (emp_no > 1 and emp_no <  1000) check ( (basic > 1000 and basic < 9000)
check ( (dept_no > 1 and dept_no <  10) )
create table dept
(
dept_no     integer ,
```

```
dept_name   char ,

manager     char ,

budget      integer

check ( (dept_no > 1 and dept_no <  10) check ( (budget > 5000 and
budget <  9000) )

create table dependent

(

dep_name    char ,

emp_no      integer ,

relation    char

)

commit;



create schema filename school

create table teacher

(

t_no        integer ,

t_name      char ,

address     char ,

basic       integer ,

subject     char

check ( (t_no > 1 and t_no <  100) check ( (basic > 2000 and basic <  5000) )
```

```
create table student
(
roll_no     integer ,
st_name     char ,
class       integer ,
guar_no     integer
check ( (roll_no > 1 and roll_no <  500) check ( (class > 5 and class <  10) )
commit;

create schema filename hospital
create table doctor
(
d_no        integer ,
d_name      char ,
address     char ,
speclz      char
check ( (d_no > 100 and d_no <  150) )
create table patient
(
p_no        integer ,
p_name      char ,
address     char ,
d_no        char        not null
)
commit;
```