MULTIPLE VIEW, MULTIPLE DATA MODEL SUPPORT IN THE CHEOPS DATABASE MANAGEMENT SYSTEM

by

Anthony Klug

Computer Sciences Technical Report #418

January 1981

Multiple View, Multiple Data Model Support in the CHEOPS Database Management System

Anthony Klug

University of Wisconsin

Abstract

CHEOPS is an experimental database system which supports multiple user views and multiple user data languages. It currently supports relational, hierarchical and network high-level interfaces and a very general view facility. There may be many views on one central schema, and one view may be bound to the central schema in many ways. Queries and updates through user views are guaranteed to give correct results because their bindings to the main schema have been verified.

Keywords and Phrases: multiple views, multiple data models, mapping, structure mapping, operation mapping, mapping correctness, relational semantics

CR Categories: 3.5Ø, 3.7Ø, 4.22, 4.33, 4.34

Contents

1. Introduction

An important problem in database management systems today is the development of effective user interfaces. The mass of published work on the relational data model is ample example of this [Kim]. In addition, many other interfaces are being developed for different classes of users. For example, an improved Codasyl-like interface is being developed for application programmers [Clem2], and for naive users, high-level graphics [Hero] and natural language interfaces [Codd3] have been presented.

With these many interfaces available, each serving a particular class of users, it is desirable to have a number of them available simultaneously on the same database. In the "jargon of the field", this is called support of multiple data models and multiple views. The desirability of multiple user data models and multiple user views has been well documented (e.g., [Nijs], [Berg], [SWKH], [KlTs], [TsKl], [Clem1]). Having multiple data models allows the user to choose a data model with the structures and language features best suited to the application at hand. Having multiple views lets the user see just that data which is needed for the application.

Providing these facilities requires a special DBMS design. The ANSI/SPARC DBMS framework [TsKl] provides (among other things) features through which multiple views and multiple data models may be implemented. As depicted in Figure 1, a DBMS is functionally partitioned into three levels: the external, at which users work, the central, at which the database administrators work, and the internal, at which the efficiency experts work. The external level may have many schemas describing various views of the database using various data models. The central level has one schema which describes the totality

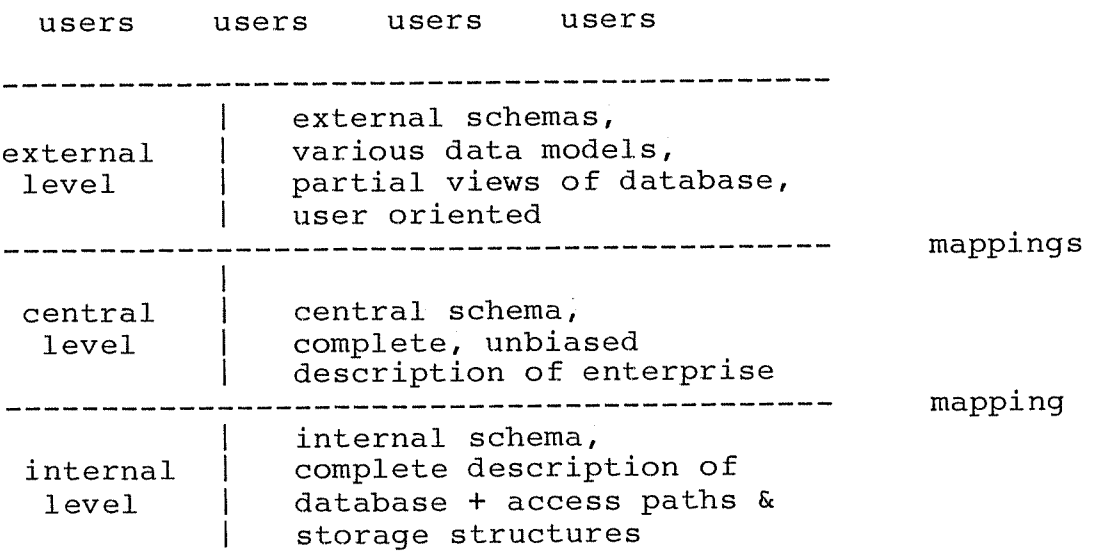


FIGURE 1. Levels in ANSI/SPARC Framework

of the information in the database. The internal level has one schema which describes the implementation of the information in the central schema. The three levels are connected by "bindings" or "mappings".

Having multiple views means that the information in the database may be filtered and transformed before presentation to the user. Having multiple user data models means that users may interact with the database through several different languages having quite different syntax and semantics. Although these two concepts are related in that seeing the database through a different language can be said to be a different view, they are not the exactly same concept. For example, suppose a user, "A", has a tree structured data model and sees a presidential database[1] with the structure of Figure 2. This user sees a simple hierarchy in which the children of presidents are the congresses with which they have served. Note that because of deaths or resignations, the same congress may appear under more than one president. However, under each president, the record for a particular

[1] Our running example is a presidential database as in [Sibl].

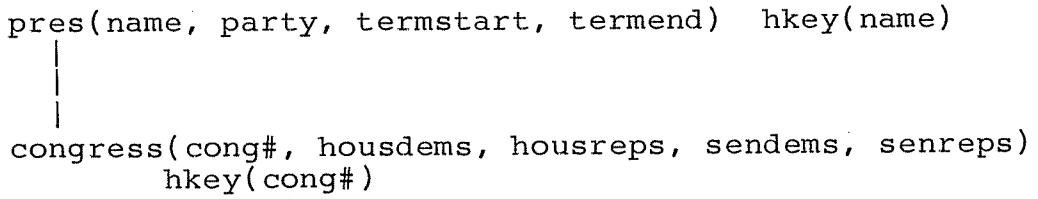


FIGURE 2. User "A"s View

will appear only once. (This is the meaning of a hierarchical key.)

There could be another user, user "B", who interacts with the database through the network schema of Figure 3. In this network view, the "link" record provides the many-to-many relationship between presidents and congresses. User "A" sees the same information (i.e. view) that user "B" sees, but the data models are different.

Now consider user "C" who uses the network schema of Figure 4. This user sees information different from that which the first two users see. User "C" has the same data model as user "B" but different view.

In general, both data models and views can be different.

1.1. Overview of Paper

The CHEOPS[2] database management system is being implemented at the University of Wisconsin to investigate the potentials and the problems of multiple view, multiple data model DBMSs. This paper presents and discusses the main design features of the CHEOPS DBMS. The subject matter of the paper is limited to the external level, the central level and the mappings between these two levels as they are

---

[2] The Egyptian king Cheops built the Great Pyramid at Giza. We chose this name because the CHEOPS structure can be likened to two pyramids end-to-end (as in the ANSI/SPARC framework, see [TsLo], p.97).

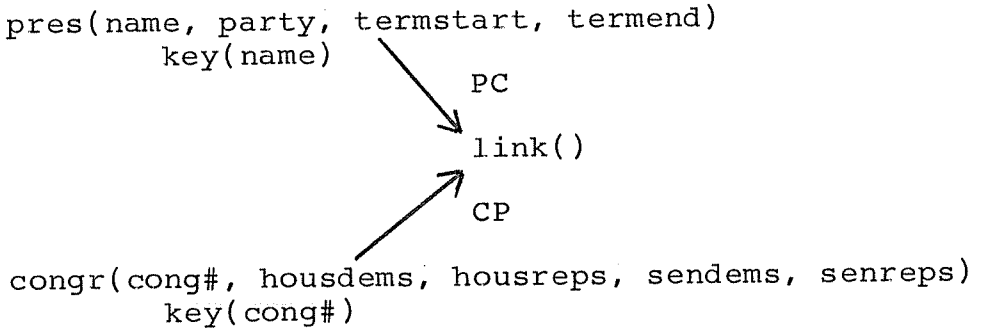


FIGURE 3. User "B"s View

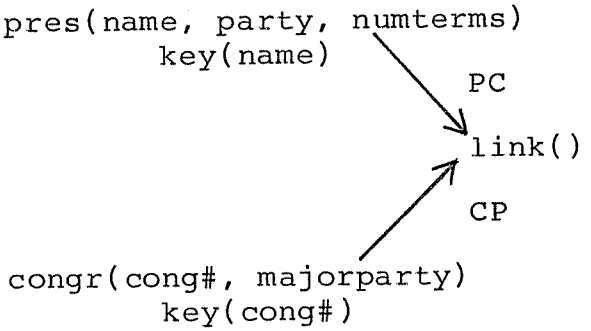


FIGURE 4. User "C"s View

implemented in CHEOPS. These are the components which are mainly responsible for providing multiple view and multiple data model support.

The design decisions and problems which must be addressed in implementing a DBMS which supports multiple views and multiple data models are several. Some of the questions we are addressing in this paper are:

(1) The multiple user data models must be brought together at some point. This can happen at the external level, the central level or the internal level or somewhere in between. Where is the best place? How is this to be done?

(2) An external schema may have both explicit and implicit constraints which help to define the expected behavior of the

defined view. What is necessary to guarantee the expected behavior?

(3) What is the nature of the objects which bind external views to the central level database? Are different constructs needed for each different user data model? For example, how does the process of extracting a hierarchical view differ from the process of extracting a network view, and how do these differ from the process of extracting a relational view?

(4) How can these many interfaces be implemented efficiently?

In succeeding sections we will show how these questions are dealt with in the CHEOPS DBMS. In the next section we begin by describing the three user data models currently available. Section 3 discusses the concept of common semantics. The data model used at the central level is described in Section 4, and in Section 5 we show how common semantics are given to the three external data models mentioned above. Mappings and their use are the subject of the sixth section. The Section 7 briefly covers the method of reducing the complexity of the central level statements produced, the treatment of update operations and the actual CHEOPS interfaces. Finally, Section 8 summarizes the paper.

2. Three External Data Models in CHEOPS

The purpose of supporting several data models at the user level is to give each user the constructs best suited to that user's needs and skills. To the questions of what data models should be included, and what facilities they should have, we cannot give definitive answers since there is a large amount of subjectivity involved. We can, however, make the following important assumption: Navigational

languages [Coda] are not needed; high-level data languages will suffice. Note that:

(1) "High-level" does not necessarily imply "relational". For example, the language proposed by Fehder [Fehd] (which we also use) is high level hierarchical data language.

(2) "High-level" does not imply "stand-alone". For example, RIGEL [RoSh] provides a relational database interface embedded in a modular programming language.

(3) We do not imply that with current technology, high-level languages will suffice. (Although, we believe this will be the case in the future.)

CHEOPS currently supports three external data models with high-level data languages. They are based on relations, hierarchies, and networks. Two other external data models are under development. These are a binary relational model with a data manipulation language similar to FORAL [Senk], and an entity relationship model with a language similar to CABLE [Shos]. These five models are the major approaches to data modeling.

In this paper, only retrievals are discussed in detail. This is solely due to lack of space. CHEOPS supports a full range of database updates which will be discussed briefly in Section 7. We assume that the reader is familiar with the basic concepts of the relational, hierarchical, and network data models. (See, for example, [Sibl].)

2.1. Relational External Model

A CHEOPS relational schema contains relation declarations and constraint declarations (keys, other functional dependencies and

subset constraints). An example relational external schema is given in Figure 5. (Data types of attributes are omitted for brevity.) It refers to a presidential database which is used throughout this paper. This schema describes information about states, elections ("repcand" is the name of the Republican candidate, and so on), administrations, and congresses. The key declarations specify that the listed domains uniquely identify tuples [Coddl]. The subset constraints specify that values in the listed domains of the left-hand relation will occur as values of the listed domains of the right-hand relation. Subset constraints are sometimes called "foreign key constraints" [Coddl], "subset dependencies" [SaWa] or "existence constraints" [Lienl]. These constraints are essential for representing information structure such as "there is a one-to-one correspondence between senates and houses of congresses".

The relational data manipulation language (DML) is based on relational algebra [Codd2]. A retrieval has the form:

retrieve <expr>

A retrieval simply returns the result of evaluating the expression. The expression <expr> is built from schema relations using the relational algebra operations of projection "R(X)" (X a list of domains), cross product "R X S", restriction "R[XΘY]" (X,Y lists of domains, Θ one of '=', '<', etc.), union "R U S" and literals "{<a,b,c>, ... }". Joins "R[XΘY]S" and selections "R[XΘV]" (V a constant) may be defined in terms of these more basic operations. We reference domains in expressions by their name, if that is unique in the expression, or by

```
state (name, capital, adpreslname, adpresfname)
election (year, winnerfname, winnerlname, loserlname)
admin (admnum, preslname, presfname, presstate, presparty)
congadmin (congrnum, admnum)
senate (congrnum, numdems, numreps)
house (congrnum, numdems, numreps)

key state (name)
key election (year)
key admin (admnum)
key senate (congrnum)
key house (congrnum)

subset admin (presstate) in state (name)
subset congadmin (congrnum) in senate (congrnum)
subset senate (congrnum) in house (congrnum)
subset house (congrnum) in congadmin (congrnum)
subset state (adpreslname, adpresfname)
        in admin (preslname, presfname)
```

FIGURE 5. Schema STELCONG (State, ELection, CONGress)
A Relational External Schema

a number identifying a cross product term followed by a domain name.

Example 1. Find the names of states admitted by Republican administrations.

```
(state X admin[presparty = "republican"])
        [admitpres = pres] (name)
```

2.2. Hierarchical External Model

Figure 6 shows an example of a hierarchical external schema for the presidential database. Nested parenthesized blocks indicate the tree structure. Each hierarchical segment must have a hierarchical key specified. The hierarchical key specifies that under a given parent the key fields are unique.

The DML for the hierarchical model is similar to that proposed by Fehder [Fehd]. Statements have the form:

```
election(year)  hkey (year)
(
        runner(name, party, evotes)
                hkey(name)
)
```

FIGURE 6. Schema ELECHIST (ELECtion HISTory)
A Hierarchical External Schema

for each <seg> having <qual> : list <outputlist>

The qualification is a boolean combination of terms, where terms compare domains with values or other domains. The domains in the qualification may reside in the target segment (the one mentioned in the for part) or in any ancestor segment. Domains of ancestor segments have the segment name prepended to them. The output list contains domains of the target segment or of any of its ancestors. It may also contain nested for constructs whose target segment is any descendant of the target segment of the enclosing for clause. The nested fors have the same form as above but without the list keyword.

Example 2. For each election of this century list the year and candidates who were Democrats.

```
for each election having year > 1899 : list year,
        (for each runner having party = "democratic" :
                fname, lname)
```

Another hierarchical external schema we will consider is given in Figure 7.

2.3. Network External Model

The CHEOPS network external model provides for definitions of record types and set types. This model allows reflexive sets, and sets can be either optional or mandatory (mandatory is default)

```
pres(fname, lname, party)  hkey(fname, lname)
(
        congr(cong#)  hkey(cong#)
        (
                house(party, seats)  hkey(party)
                senate(party, seats)  hkey(party)
        )
)
```

FIGURE 7. Schema PRESCONG (PRESident, CONGress)
A Hierarchical External Schema

[Coda]. A network external schema for the presidential database is given in Figure 8. Aids to understanding this schema are provided in Figure 9.

The statements for the network model have the same overall structure as hierarchical statements:

for each <record> having <qual> : list <outputlist>

In a qualification, instead of referenced domains belonging to either the target record or to an ancestor as in the hierarchical external model, a domain can belong to any record connected to the target record by a directed path (in the owner direction only) of sets. In a list statement, a nested for clause specifies a directed path (in member direction only) of sets.

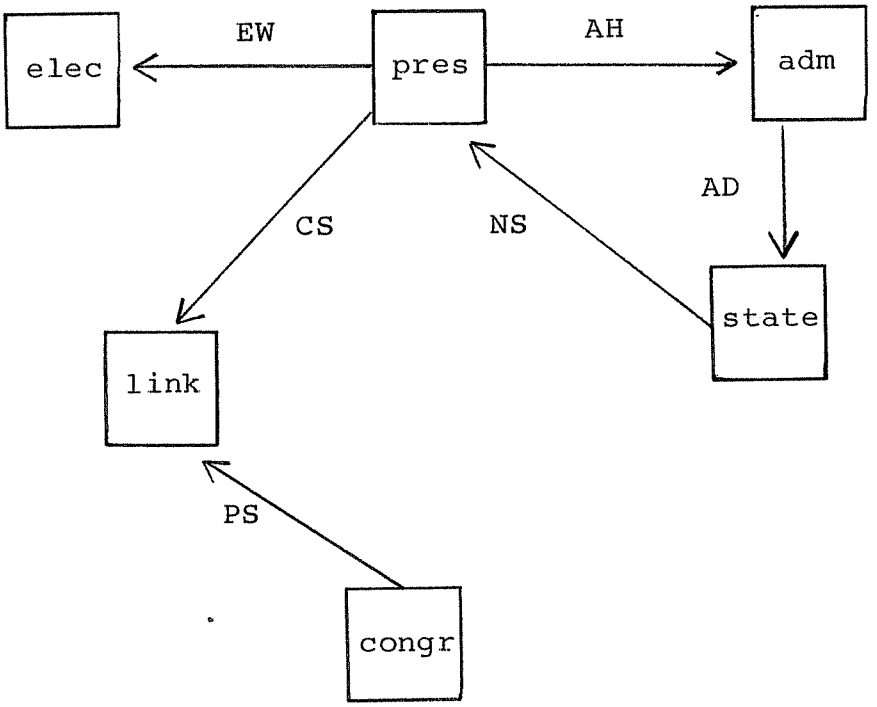
Example 3. List all presidents names and names of their native states.

```
for each pres: list lname, fname, NS.name
```

```
records
        pres (fname, lname, party) key(fname, lname)
        adm (ano, startyr, startmo, startday)
        elec (year, votes) key(year)
        state (name, capital, yradm) key(name)
        congr (congrno, housdem, housrep,
                  sendem, senrep) key(congrno)
        link    ()


sets
        EW ( pres->elec )
        CS ( pres->link )
        PS ( congr->link )
        NS ( state->pres )
        AH ( pres->adm )
        AD ( adm->state )
```

FIGURE 8. Schema PRADESC (President, ADministration, Election, State, Congress)
A Network Schema for Presidential Database.



(a) Data-Structure Diagram for PRADESC Schema

RECORD TYPES

pres	president's name, party, and college
adm	administration number (per president), starting year, month and day, and length
elec	year of election, electoral votes of winner
state	name, capital, and year admitted to union
congr	number of congress, number of Democrats and Republicans in house and senate
link	linking record for many-to-many congress-president relationship

SET TYPES

EW	elections won by president owner
CS	congresses served (via link) by president owner
PS	presidents served (via link) by congress owner
NS	native sons for state owner
AH	administrations for president owner
AD	states admitted during administration owner

(b) Description of Records and Sets

FIGURE 9.

Example 4. List the presidents who admitted their own native states to the union (no presidents actually did this), and list information on the congresses they served.

```
for each pres having lname = NS.AD.AH.lname and
   fname = NS.AD.AH.fname :
        list fname, lname, ( for each link via CS :
                                  PS.congrno,
                                  PS.housdem, PS.housrep,
                                  PS.sendem, PS.senrep)
```

3. The Common Semantics Approach in CHEOPS

The previous section presented three different data models which are presently available at the CHEOPS user level. In this section we consider the general question of how different data models are to be supported, and we justify the common semantics approach taken in CHEOPS.

Consider the problem of binding (mapping) of user views and user data models to the central schema and central data model. This binding must do two things: It must change data models, since the user and the central data models will in general not be the same. The binding must also extract the view, masking and transforming whatever is necessary.

In addition, in whatever manner the mapping is done, certain mapping processors must be available. There must be a compiler for the mapping, and there must be a mapping verifier. (Mapping verification is discussed in Section 6).

Changing data models and extracting the view can be done in three ways:

(1) The mapping can do both simultaneously. In this case the external schemas, regardless of the models to which they belong, would be connected directly by the mappings to the central schema as in Figure 1Ø. For each different external data model there must be a different mapping language. The mapping language must associate one kind of structure (external) with another type (central), and must also perform transformations such as joining, averaging, projecting, and so on. This double duty will complicate the syntax and semantics of a "cross data model" mapping language. The necessary mapping processors (see Section 6) will also be complicated.

(2) We can change the data model at the central level and extract the view using constructs of the external data model. Then for every external data model there is a transformation which from the central schema produces an equivalent schema conforming to that external data model as in Figure 11. Here, the syntax and semantics of a mapping language will not be so complicated since it will map the data model constructs to constructs of the same type. On the other hand, many mapping languages and processors must still be written. This will not be easy. Consider only the language design problem. A designer of, say, a hierarchical mapping language must ask: "What are all the possible ways to derive a hierarchical view from a given hierarchy?" A designer of a network mapping language must ask: "What are all the possible ways to derive a network view from a given network schema?" A designer of an entity/relationship mapping language must ask: "What are all the possible ways in which one view of entities and relationships can be derived from a given set of entities and relationships?" Each time a new external data model is to be

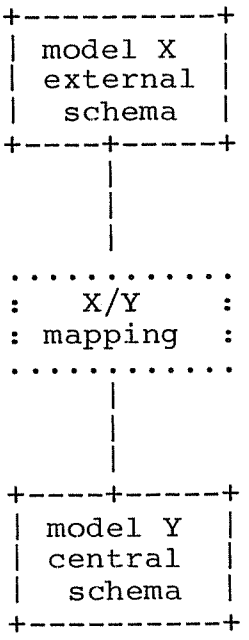


FIGURE 1Ø. Direct Connection, No Transformations

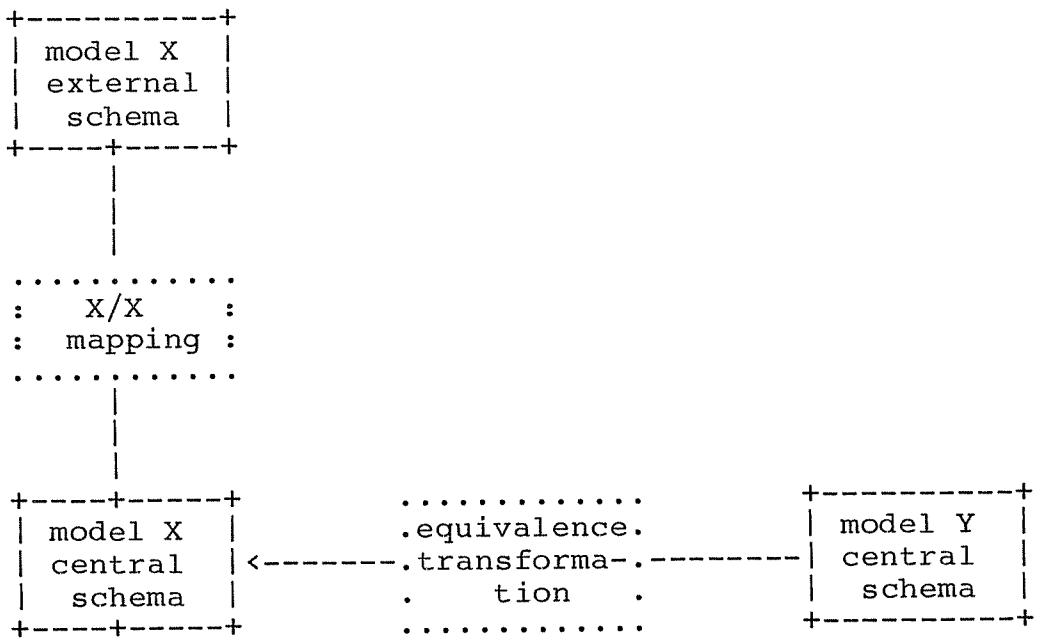


FIGURE 11. Equivalence Transformation of Central Schema

added, we would have to consider this difficult question.

Another problem is that the equivalence transforms are going in the "wrong direction". That is, they must take a schema from a model which is specifically designed to be used at the central level, i.e., is designed to represent general information structures, and produce an equivalent schema in a model which is end-user oriented and may be restrictive and simplified. For example, from an entity/relationship [Chen] central schema it is harder to produce an equivalent hierarchical schema than it is to produce an equivalent relational schema.

(3) We can change the data model at the external level and extract the view using constructs of the central data model. In this approach to designing the central/external interface, all external schemas regardless of their data model are transformed into schemas of the central data model as in Figure 12. Now there is a need for only a single mapping language and a single mapping processor. In addition, a transform represents the less general, end-user data models by constructs of the more general central data model. Clearly, this is an easier transformation than the previous one. Finally, the question "What are all the possible ways to derive a view?" needs to be answered only once.

The schema transformation we have just described is one component of common semantics for an external data model. This transformation we call the schema semantics. Since a data model also has a data manipulation language, we also need DML semantics. These functions[3] are depicted in Figure 13.

---

[3] The schema is needed as a parameter to the DML transform, and the dots under the arrows indicate that several relational DML statements and results may be needed with intermediate processing for translating a DML statement of data model X. (Such complications are

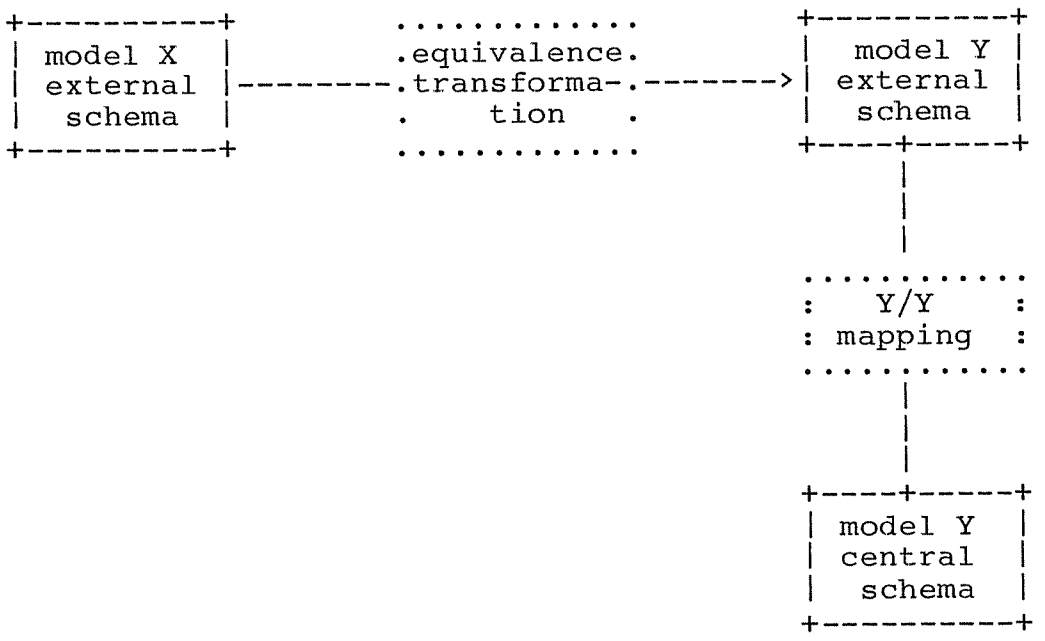


FIGURE 12. Equivalence Transformation of External Schema

---

ignored in this paper.) Also, we have anticipated the next section by calling the central data model relational rather than just "model Y".

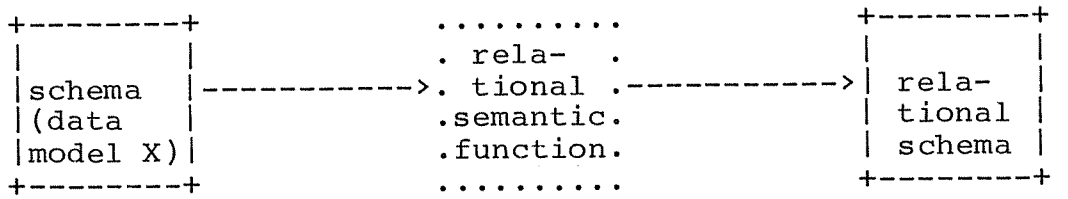


FIGURE 13a. Schema Semantic Function

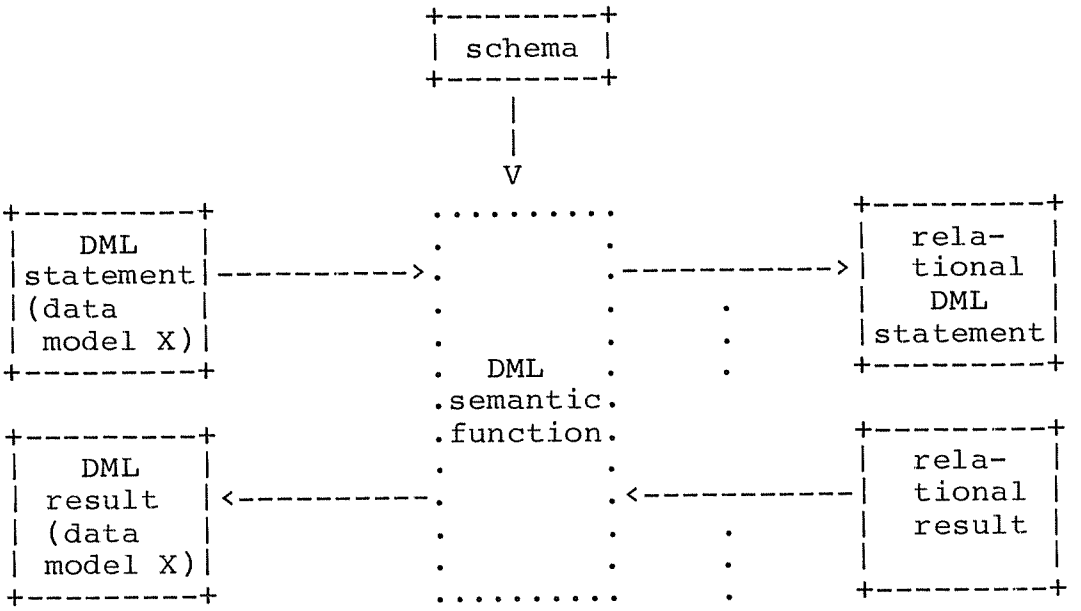


FIGURE 13b. DML Semantic Function

We can contrast the common semantics approach to accommodating different data models with another approach which might be called the "subsetting" approach [Date]. In this approach, there is one "super-model" from which all other data models are derived by adding restrictions on language syntax. For example (in terms of schemas only), Date considers a hierarchy to be a special case of a network -- one in which each child record has exactly one parent record, and he considers a relation to be a special case of a hierarchy -- a hierarchy consisting of a root only. While subsetting may work well for particular network, hierarchical, and relational models, it cannot be generalized. For example, even relational algebra and relational calculus cannot reasonably be considered to be subsets of the same language. With the subsetting approach, all user models must be made mutually

consistent in their syntax and semantics. With our common semantics approach, we may use any sort of procedure to define any external data model within the expressive power of the central data model.

Another advantage of the common semantics approach to supporting multiple data models is that the "cross data model interference" [PaPe] problem disappears. This is the problem of how to handle conflicting update semantics from different data models. For example, deletion in hierarchical data models implies deletion of subordinate segments, while deletion in relational models has no implied deletions. Presumably, simultaneous updates by users of these models could cause consistency problems. With the common semantics approach all model-specific operation effects, whether implied or explicit, are explicitly phrased in terms of operations of the common data model.

4. The Central Data Model in CHEOPS

The last section explained why CHEOPS defines common semantics for all of its external data models. In this section we describe the CHEOPS central data model (the target model of the semantic functions).

Several complete papers could be (and have been, see [BrPP], [Hawr], [Kent], [Schm], for example) written on the subject of what should be or should not be in the central data model. This debate lies beyond the scope of this paper, and we have attempted to indicate this by our choice of terms: "central data model" rather than the more traditional "conceptual data model". The central data model must, nevertheless, have at least the facility to support, via the semantic functions of the previous section, the external data models. It has been shown that relations plus functional dependencies and subset con-

straints can represent a large class of other structures such as hierarchies [Coddl], networks [Zani] [Lien2], and entities and relationships [Klug4] [ZaMe] [Chen]. Thus the CHEOPS central data model is relational (with functional dependencies and subset constraints): A CHEOPS central schema contains relation declarations and constraint declarations (keys, other functional dependencies and subset constraints). The central schema for the presidential database is given in Figure 14. Some explanations are given in Figure 15.

The DML at the central level is based on relational algebra. The language is thus identical with that of the external relational data language.

5. Some Common Semantics in CHEOPS

The algorithm for giving relational semantics to a CHEOPS hierarchical schema is essentially Codd's First Normal Form algorithm [Coddl]. For example, the PRESCONG schema (Figure 7) is represented by the relational schema in Figure 16. Each hierarchical segment generates a corresponding relation. The relational keys are generated by recursively bringing down keys of "parent" relations and concatenating them with domains of the given hierarchical key. (Segment names are added to guarantee uniqueness of domain names). The subset constraints are generated to model the fact that hierarchical children always have parents. It is important to note that the subset constraints are an essential part of the semantics of the hierarchical schema. If they were omitted, there would be legal instances of the relational schema in which parts were children of nonexistent suppliers. This would not be the intended semantics of a hierarchical model.

```
Pres(lname, fname, party, state)
Admin(admink, imonth, iday, iyear, plname, pfname)
State(sname, yearad, capital, admitadm)
Elecloser(year, name, party, evotes)
Senate(congky, party, seats)
House (congky, party, seats)
Congpres(congky, plname, pfname)
Elecwinner(year, plname, pfname, evotes)

key Pres(lname, fname)
key Admin(admink)
key State(sname)
key Elecloser(year)
key Senate(congky)
key House(congky)
key Elecwinner(year)

subset Admin(plname, pfname) in Pres(lname, fname)
subset Senate(congky) in House(congky)
subset House(congky) in Senate(congky)
subset Congpres(congky) in Senate(congky)
subset Senate(congky) in Congpres(congky)
subset Congpres(plname, pfname) in Pres(lname, fname)
subset Elecwinner(plname, pfname) in Pres(lname, fname)
```

FIGURE 14. Central Schema for Presidential Database

Pres	last name, first name, party, native state of president
Admin	president's administration with number, inauguration date and president's name
State	state name, year became state, capital city, administration admitting
Elecloser	year of election, loser's name, party and electoral votes won
Senate	congress number, party name, senate seats for party
House	congress number, party name, house seats for party
Congpres	congress number and president name serving together
Elecwinner	year of election, winner (i.e., president) name and electoral votes

FIGURE 15. Description of Central Schema.

```
pres(fname, lname, party)
congr(pres^fname, pres^lname, cong)
house(congr^cong#, pres^fname, pres^lname, party, seats)
senate(congr^cong#, pres^fname, pres^lname, party, seats)

key pres(fname, lname)
key congr(pres^fname, pres^lname, cong#)
key house(congr^cong#, pres^fname, pres^lname, party)
key senate(congr^cong#, pres^fname, pres^lname, party)

subset congr(pres^fname, pres^lname) in pres(fname, lname)
subset house(congr^cong#, pres^fname, pres^lname) in
       congr(cong#, pres^fname, pres^lname)
subset senate(congr^cong#, pres^fname, pres^lname) in
       congr(cong#, pres^fname, pres^lname)
```

FIGURE 16. Relational Semantics of PRESCONG Schema (Fig. 7)

The relational semantics for the ELECHIST schema (Figure 6) are shown in Figure 17.

We will illustrate here only the semantics of simple hierarchical queries which require no intermediate processing by the semantic function. Given a hierarchical query containing no nested for-constructs[4], the semantic function generates a relational query which joins on hierarchical keys all relations corresponding to referenced segments, restricts according to the qualification and projects according to the output list. Consider the following query for the ELECHIST schema:

```
for each runner having party = "democratic" and
        election.year > 1899 :
                list election.year, fname, lname)
```

The meaning of this query (using the relations from the equivalent relational schema) is the following relational query:

---

[4] Intermediate processing is required when there nested for's, and the output must be made to look "hierarchical" rather than "flat".

```
elec(year)
runner(elec^year, name, party, votes)

key elec(year)
key runner(elec^year, name)

subset runner(elec^year) in elec(year)
```

FIGURE 17. Relational Semantics for ELECHIST Schema (Figure 6)

```
retrieve  (runner X elec)
                [elec^year = year and party = "democratic"
                 and year > 1899] (year, fname, lname)
```

The relations are joined on the hierarchical keys, the proper restrictions are made, and the requested domains are projected out.

The schema semantic function for the network model is driven by two goals: One or more keys must be determined for each record type to make it into a relation, and each set must be relationally represented in the member relation by new domains referring to key attributes of the owner relation plus some subset constraints. (New domain names are made unique by concatenating to them the name of the set which caused their creation.) The algorithm in CHEOPS is similar to but simpler than the one given by Zaniolo. An outline of the algorithm for the schema semantic function is given in Figure 18. The PRADESC network schema (Figure 8) is mapped by the semantic function to the relational schema of Figure 19.

In general, the query semantic function for simple network queries forms a join containing all paths referenced in the query and selects out the target records satisfying the qualification.

Example 5. Consider the query using the PRADESC schema to list congress numbers which served presidents who admitted to the union their own native states:

```
mark TO-DO all records with keys declared in schema;
repeat
        while (there exists record R TO-DO)
        {
                for each set X owned by R
                {
                        concatenate 'X^' to key attrs and
                        make these new attrs in mem rec S of X;
                        if (X has key-clause)
                        {
                                add new key-decl to rel schema
                                consisting of new attrs and
                                attrs in key-clause;
                                if (S not DONE)
                                        mark S TO-DO;
                        }
                }
                mark R DONE;
        }
        while (there exists record R not DONE)
        {
                add new key decl to rel schema
                consisting of all new attrs;
                mark R TO-DO;
        }
until (no more changes)
```

FIGURE 18. Network Schema Semantic Function

```
pres(fname, lname, party, NS^name)
adm(ano, startyr, startmo, startday, AH^fname, AH^lname)
elec(year, votes, EW^fname, EW^lname)
state(name, capital, yradm, AD^ano)
congr(cong#, housdem, housrep, sendem, senrep)
link(CS^fname, CS^lname, PS^cong#)

key pres(fname, lname)
key adm(ano)
key elec(year)
key state(name)
key congr(cong#)
key link(CS^fname, CS^lname, PS^cong#)

subset pres(NS^name) in state(name)
subset adm(AH^fname, AH^lname) in pres(fname, lname)
subset elec(EW^fname, EW^lname) in pres(fname, lname)
subset state(AD^ano) in adm(ano)
subset link(CS^fname, CS^lname) in pres(fname, lname)
subset link(PS^cong#) in congr(cong#)
```

FIGURE 19. Relational Semantics of PRADESC Schema (Fig. 8)

```
for each link having
        CS.lname = CS.NS.AD.AH.lname and
        CS.fname = CS.NS.AD.AH.fname :
                list PS.cong#
```

The semantic function produces the statement:

```
retrieve  (link X pres X state X adm X pres X congr)
          [ CS^fname = 2.fname and CS^lname = 2.lname and
            2.NS^name = name and
            AD^ano = ano and
            AH^fname = 5.fname and AH^lname = 5.lname and
            PS^cong# = cong# and
            2.lname = 5.lname and 2.fname = 5.fname ]
          (cong#)
```

The terms on lines 1,2,3, and 4 of the restriction "follow", respectively, the path from link via CS to pres, from pres via NS to state, from state via AD to adm, and from adm via AH to pres. The last term equates the first occurrence of pres (the one who is the native son) with the second occurrence of pres (the one who admitted the state), and the term on line 5 joins link with congr to get the output domain.

6. Schema Mappings in CHEOPS

In the last section we saw how schemas and queries of different data models could be represented by schemas and queries of a single common data model (a relational one). In this section we describe how user views are bound to the central schema in CHEOPS via the common semantics.

An important result of the Section 3 is that CHEOPS may pretend, for the purposes of binding external user models to the central model, that all external models are relational. Thus, this section on mappings between schemas refers only to relational external schemas. To bind a network, hierarchical or any other external schema, we bind the equivalent relational schema which is produced by the schema semantic

function.

A mapping specifies relations between schema (syntactic) objects. The CHEOPS schema mapping language is based on relational algebra. A structure mapping consists of a set of equations

$$R = e$$

where R is an external relation and e is a relational algebra expression over relations in the central schema. For each R there is exactly one such equation in a mapping, and the type of R defined in the external schema is the same as the type of e as determined by the relations in the central schema.

In the terminology of, say, INGRES [SWKH], R is a view relation. There is an important difference, however, between view relations in the style of INGRES and CHEOPS external relations. The definition and declared constraints of an external relation are completely independent of the mapping equation which "implements" the external relation, whereas an INGRES view relation has no existence independent of its defining query.

A structure mapping completely determines the results of external retrieval statements. The structural part of the mapping is used to to substitute central schema expressions for the external schema relations. This is analogous to query modification in INGRES [Ston].

In Section 2, where three user data models were presented, we gave four example schemas which belonged to different data models (2 were hierarchical), which all represented some view of a presidential database, and whose structures were quite different. In the following paragraphs we will show:

(1) How the schema semantics work with the schema mapping to produce output.

(2) How these four different views expressed in different data models can all be bound to the same central schema (Figure 14).

(3) How a single user schema (we use the ELECHIST schema) can be bound to the central schema in several different ways producing quite different views. (The view acts as a movable window over the database.)

In Figures 2Ø through 22 we give mappings for the STELCONG, PRESCONG, and PRADESC schemas. In Figures 23 and 24 two different mappings are given for the ELECHIST schema. Commentary on these mappings follows:

The STELCONG schema is relational. Thus it can be bound directly to the central schema by a mapping. The mappings for state, admin and congadmin are simple joins to get the correct attributes into one relation. The senate view contains attributes for Democrat numbers and for Republican numbers. (The case for house is similar.) However, the Senate central relation has a different structure. Thus the mapping must join two copies of the Senate relation (on year) and select Democrat tuples from one copy and Republican tuples from the other. (Note that congresses without both Democrats and Republicans will not appear.) To derive a desired structure for the election view, we must first join Elecwinner with Pres to correlate the winner's party with the vote and year attributes. Then we will add a join with Elecloser. Thus far, there will be as many tuples for a given year as there are losers. Only those tuples are selected involving Democrats and Republicans. A mapping could also be written which places in the view other combinations of winners and losers. The only restriction is that only one loser can appear for each year.

The PRESCONG schema is hierarchical. Thus the mapping is defined between the central schema and the relational equivalent of the

```
state     = (State X Admitadmin X Admin)
                [1.sname=2.sname and 2.admink=3.admink]
                (1.sname, capital, plname, pfname)

election  = (Elecwinner X Pres X Elecloser)
              [1.plname=2.lname and 1.pfname=2.fname and
               1.year=3.year and
               (2.party="democratic" or 2.party="republican") and
               (3.party="democratic" or 3.party="republican")]
              (1.year, 1.pfname, 1.plname, 3.name)

admin     = (Admin X Pres) [plname=lname and pfname=fname]
                (admink, lname, fname, state, party)

congadmin = (Congpres X Admin) [plname=lname and pfname=lname]
                (congky, admink)

senate    = (Senate X Senate)
                [1.congky=2.congky and 1.party="democratic" and
                   2.party="republican"]
                (1.congky, 1.seats, 2.seats)

house     = (House X House)
                [1.congky=2.congky and 1.party="democratic" and
                   2.party="republican"]
                (1.congky, 1.seats, 2.seats)
```

FIGURE 2Ø. Mapping for STELCONG Schema (Figure 5)

```
pres   = Pres (fname, lname, party)

congr  = Congpres (pfname, plname, congky)

house  = (House X Congpres)
                [1.congky=2.congky]
                (1.congky, pfname, plname, party, seats)

senate = (Senate X Congpres)
                [1.congky=2.congky]
                (1.congky, pfname, plname, party, seats)
```

FIGURE 21. Mapping for PRESCONG Schema (Figures 7,16)

```
pres    = Pres (fname, lname, party, state)

adm     = Admin (admink, iyear, imonth, iday, pfname, plname)

elec    = Elecwinner (year, pfname, plname, evotes)

state   = (State X Admitadmin) [1.sname=2.sname]
                (1.sname, capital, yearad, admink)

congr   = (House X House X Senate X Senate)
                [1.congky=2.congky and 1.congky=3.congky and
                 1.congky=4.congky and 1.party="democratic"
                        and 2.party="republican" and
                 3.party="democratic" and 4.party="republican"]
                (1.congky, 1.seats, 2.seats, 3.seats, 4.seats)

link    = Congpres (pfname, plname, congky)
```

FIGURE 22. Mapping for PRADESC Schema (Figures 8,19)

```
elec   = Elecwinner(year)

runner = (Elecwinner X Pres) [plname=lname and pfname=fname]
                (year, lname, party, evotes)
         U
         Elecloser
```

FIGURE 23. Mapping 1 for ELECHIST Schema (Figures 6,17)

```
elec   = (Elecwinner X Pres)
                [plname=lname and pfname=fname
                 and party="whig"] (year)
       U
       Elecloser [party="whig"] (year)

runner = (Elecwinner X Pres X Elecloser)
                [ plname=lname and pfname=fname and
                  1.year=3.year and
                  (2.party="whig" or 3.party="whig") ]
                (1.year, lname, 2.party, 1.evotes)
            U
          (Elecwinner X Pres X Elecloser)
                [ plname=lname and pfname=fname and
                  1.year=3.year and
                  (2.party="whig" or 3.party="whig")]
                (3.year, 3.name, 3.party, 3.evotes)
```

FIGURE 24. Mapping 2 for ELECHIST Schema (Figures 6,17)

hierarchical view (Figure 16). This mapping happens to be as simple as possible. It merely joins appropriate relations to get the required attributes in one place.

The PRADESC schema is a network. The mapping is defined between the central schema and the relational equivalent of this network view (Figure 19). The mappings for pres, adm, elec, and link are simple reorderings of the attributes. The mapping for state is a join to get the needed attributes together. The mapping for congr performs a task similar to the mapping for the election relation above. It must join two copies each of House and Senate and select out Democratic and Republican numbers.

The ELECHIST schema is hierarchical. The relational equivalent is in Figure 17. For this schema we have defined two mappings. Mapping 1 extracts all elections and the winner and all losers in the elections. The set of all election years is the projection on the year attribute of Elecwinner (or Elecloser). The set of all runners (candidates) is the set of winners (presidents, with a join needed) plus the set of losers.

Mapping 2 illustrates how the election history schema can be used as a window over selected parts of the database. With mapping 2 we see only elections involving the Whig party. To get the elections years we take a union of years that winners were Whigs with years that losers were Whigs. To get the runners in years in which Whigs ran, we first join winners (with a join to Pres to get the party) with losers and select those joined tuples which have either a Whig winner or a Whig loser. We then take a union of a projection on winner attributes with a projection on loser attributes.

To illustrate the effect of these several views and mappings, we give below some sample output from typical queries.

Example 6. With the STELCONG schema and the given mapping, get information on elections since 1950.

query: retrieve election [year > 1950]

result:

year	winnerfname	winnerlname	loserlname
1952	dwight	eisenhowe	stevenson
1956	dwight	eisenhowe	stevenson
1960	john	kennedy	nixon
1964	lyndon	johnson	goldwater
1968	richard	nixon	humphrey
1972	richard	nixon	mc govern
1976	jimmy	carter	ford
1980	ronald	reagan	carter

Example 7. Using the PRESCONG schema and the given mapping, get congress information for recent congresses (the most recent five for which there is complete information in our database).

```
query:  for each congr having cong# >= 89 and cong#<=93 :
                  list cong#, (for each house: party, seats),
                              (for each senate: party, seats)
```

result:

congr	house		senate	
cong#	party	seats	party	seats
89	democratic	295	democratic	68
	republican	140	republican	32
90	democratic	248	democratic	64
	republican	187	republican	36
91	democratic	243	democratic	58
	republican	192	republican	42
92	democratic	254	conserve.	1
	independnt	1	democratic	53
	republican	180	independnt	2
			republican	44
93	democratic	255	democratic	57
	independnt	1	republican	43
	republican	179		

Example 8. Using the PRADESC schema and the given mapping, get names of presidents with last names beginning with 'N','P' or 'R' (to keep output small), the elections they won and the numbers of the congresses they served.

```
query:   for each pres having lname>"n" and lname<"s" :
           list fname,lname,
                (for each elec via EW : year),
                (for each link via CS : PS.cong#)
```

result:

pres		elec	link	
fname	lname	year	cong#	
franklin	pierce	1852	34	
franklin	roosevelt	1932	73	
		1936	74	
		1940	75	
		1944	76	
			77	
			78	
			79	
james	polk	1844		(no republican party
richard	nixon	1968	91	yet, see mapping)
		1972	92	
			93	
ronald	reagan	1980		(no house informa-
theodore	roosevelt	1904	57	tion in database)
			58	
			59	
			60	

Example 9. Using the ELECHIST schema with mapping 1, get election information for the elections since 1950.

query: for each elec having year > 1950 :
list year, (for each runner: name, party, votes)

result:

elec	runner		
year	name	party	votes
1952	eisenhowe	republican	442
	stevenson	democratic	89
1956	eisenhowe	republican	457
	stevenson	democratic	73
1960	kennedy	democratic	303
	nixon	republican	219
1964	goldwater	republican	52
	johnson	democratic	486
1968	humphrey	democratic	191
	nixon	republican	301
	wallace	american	46
1972	mc govern	democratic	17
	nixon	republican	521
1976	carter	democratic		(no info)
	ford	republican		(no info)
1980	anderson	indep.	0
	carter	democratic	49
	reagan	republican	489

Example 10. With the ELECHIST schema and mapping 2, get all information.

query: for each elec :
list year, (for each runner: name, party, votes)

result:

elec	runner		
year	name	party	votes
1836	harrison	whig	73
	mangum	indep.	11
	van buren	democratic	170
	webster	whig	14
	white	whig	26
1840	harrison	whig	234
	van buren	democratic	60
1844	clay	whig	105
	polk	democratic	170
1848	cass	democratic	127
	taylor	whig	163
1852	pierce	democratic	254
	scott	whig	42

To get the output in the above examples, the following process occurs:

(1) The user query is converted to relational form by the common semantic function.

(2) The mapping is applied to the resulting relational algebra retrievals. For every external relation in the view, the corresponding entry is found in the mapping, and the right-hand side of the equation is substituted.

(3) The resulting expression referring to central relations is executed.

We illustrate this process with an example:

Example 11. Using the PRESCONG schema and the given mapping, consider the query to list numbers of senates along with numbers of seats for each parts:

```
for each senate:list congr.cong#,party,seats
```

The DML semantic function of CHEOPS outputs the relational query[5] (referencing the relations of Figure 16):

```
retrieve  ( senate
            X
            congr
          ) [D1 = D8 and D2 = D6 and D3 = D7]
            (D8,D4,D5)
```

The mapping entries for senate and congr are found, and the right-hand sides are substituted:

```
retrieve  ( (Senate X Congpres)[D0 = D3](D0,D5,D4,D1,D2)
            X
            Congpres(D2, D1, D0)
          ) [D1 = D8 and D2 = D6 and D3 = D7]
            (D8,D4,D5)
```

This is the query which is sent to central level processors for evaluation.

Mapping Correctness

The database mappings in CHEOPS share some properties with implementations of one data type (e.g., a stack) in terms of another data type (e.g., an array). Like data type implementations, database mappings must be "correct" for the user to see a view having the structure specified in the schema. If the mapping is not correct, the expected structure may not be present. Below are two examples showing

---

[5] Here, we have identified attributes by number rather than by name. This is how CHEOPS actually works, because by avoiding the renaming of attributes, query mapping is easier.

incorrect mappings and the unexpected results they give.

Example 12. Suppose the ELECHIST schema were bound with the mapping of Figure 25. The binding for elec erroneously selects only years in which the winner was a Whig. The subset constraint ("every child has a parent") in Figure 17 will fail. To see a manifestation of this error, consider the query:

```
for each elec : list year,
        (for each runner:name, party, votes)
```

The result with the incorrect mapping will be:

elec	runner		
year	name	party	votes
	harrison	whig	73
	mangum	indep.	11
	van buren	democratic	170
	webster	whig	14
	white	whig	26
1840	harrison	whig	234
	van buren	democratic	60
	clay	whig	105
	polk	democratic	170
1848	cass	democratic	127
	taylor	whig	163
	pierce	democratic	254
	scott	whig	42

Clay and Polk seem to have run in 1840, and a number of candidates do not even have a parent year.

Example 13. Suppose the STELCONG schema were bound with the mapping of Figure 26. Year is supposed to be a key for election, but since there can be many losers in an election, the given mapping will make this view constraint fail. To see this, consider the query:

```
retrieve election [year=1980]
```

```
elec   = (Elecwinner X Pres)
                [plname=lname and pfname=fname
                 and party="whig"] (year)

runner = (Elecwinner X Pres X Elecloser)
                [ plname=lname and pfname=fname and
                  1.year=3.year and
                  (2.party="whig" or 3.party="whig") ]
                (1.year, lname, 2.party, 1.evotes)
            U
          (Elecwinner X Pres X Elecloser)
                [ plname=lname and pfname=fname and
                  1.year=3.year and
                  (2.party="whig" or 3.party="whig")]
                (3.year, 3.name, 3.party, 3.evotes)
```

FIGURE 25. Incorrect Mapping for ELECHIST Schema

```
    .
    .
    .
election  = (Elecwinner X Elecloser)
                [1.year=2.year]
                (1.year, 1.pfname, 1.plname, 2.name)
    .
    . (others unchanged)
    .
```

FIGURE 26. Incorrect Mapping for STELCONG Schema

With incorrect mapping, this query gives the result:

year	winnerfname	winnerlname	loserlname
1980	ronald	reagan	anderson
1980	ronald	reagan	carter

The problem of determining correctness of structure mappings is not the problem of determining losslessness of joins [AhBU] or decomposability of relations [Riss]. External views are supposed to hide information and provide irreversible transformations; we do not want to require that a structure mapping be lossless or reversible.

In the CHEOPS system, constraints consist of key constraints (which are sets of functional dependencies), other functional dependencies and subset constraints. The structure mapping verifier must check that the validity of every constraint in the external schema E is implied by the mapping m plus the constraints of the central schema C. If the external constraint is a functional dependency R:Z->A, and the equation R = e is in m, the verifier must check that Z->A is valid on expression e over C. If the constraint is a subset constraint R1(X1) in R2(X2), and the equations R1 = e1 and R2 = e2 are in m, the verifier must check that e1(X1) in e2(X2) is valid over C.

In [KlPr] (with an earlier version in [Klugl]), an algorithm based on the tableau technique [AhSU1,2] has been given for the constraint verifier. The outline of the algorithm follows:

Constraint Verifier Algorithm

Given schema C and candidate FD e:Z->A, construct tables (a tableau) for relations of C such that two formal tuples t1, t2 having t1[Z] = t2[Z] will appear in expression e. If after ensuring that all constraints of C hold on the tables, it is found that t1[A] = t2[A], then Z->A is valid on e. If not, then Z->A is invalid because a counterexample state has been constructed. Given schema C and candidate subset constraint e1(X1) in e2(X2) construct tables for relations of S such that a formal tuple t will appear in e1(X1). If after ensuring that all constraints of C hold on the tables, it is found that t also appears in e2(X2), then e1(X1) in e2(X2) is valid. If not, then it is invalid because a counterexample has been constructed.

7. Optimization, Updates, and Other Topics

In this section, we briefly discuss other features of CHEOPS.

Logical Optimization

In a database system supporting multiple views and multiple data models, we can identify two kinds of new performance problems:

(1) Problems resulting from the extra levels through which queries, updates, and results must pass.

(2) Problems resulting from a view being radically different from the underlying database structure (either central or internal).

The second class of problems cannot really be termed "wasteful" or "inefficient" because by binding a view with a complicated mapping (e.g., mapping 2 for ELECHIST), we save the user a great deal of work in formulating queries. If the storage structure is not designed with a certain complicated mapping in mind, it cannot be expected that queries going through this mapping will be efficient. On the other hand, the first type of inefficiency should and can be dealt with.

The problem is that the process which starts with a query or operation in one of the external user models, which translates the statement to an equivalent relational form, and which maps the statement down to the central level, can result in an unnecessarily complicated query or operation. There may be cross product terms which contribute nothing to the result; there may be projections on projections, and so on. To solve this problem, the resulting expressions referring to central level relations are sent in CHEOPS through a "logical optimization" stage. By "logical" we mean that relational algebra equivalence transformations which are independent of access paths, storage structures or any other physical characteristics of the database are applied to expressions to reduce their complexity. We

illustrate the effect of logical optimization with an example.

Example 14. Consider the query, using the PRADESC schema, to retrieve the names of presidents who have admitted states to the union:

```
for each state :
        list AD.AH.fname, AD.AH.lname
```

The relational equivalent of this query and the mapped query are, respectively (again using attribute numbers for brevity):

```
retrieve  (state X adm X pres)
          [D4 = D5 and D9 = D11 and D10 = D12]
          (D11, D12)

retrieve  ( (State X Admitadmin)[D1 = D6](D1, D3, D2, D5)
            X Admin(D1, D4, D2, D3, D6, D5)
            X Pres(D2, D1, D3, D4))
          [D4 = D5 and D9 = D11 and D10 = D12]
          (D11, D12)
```

The CHEOPS logical optimization step produces a query containing just one join:

```
retrieve  (Admitadmin [D0 = D0] Admin)(D7, D6)
```

In words, the presidents who admitted states are the ones who have a tuple in Admin joining with at least one tuple in Admitadmin.

The current implementation of logical optimization is ad hoc. We are currently working on a theoretical basis. The problem is a generalization of the optimization discussed in [AhSU1,2] and [ChMe]. In our case. the presence of subset constraints changes the problem. The basic way in which a subset dependency can be used to reduce the size of an expression is as follows:

A projection of a join of the form (R[A=B]S)(X), where X contains only domains of R and S is a base relation (no selections), is equivalent to the projection R(X) if the subset constraint R(A) in S(B) is valid.

The expression is essentially a semi-join [BeGo], and the subset constraint guarantees that the semi-join is all of R.

Update Operations

In a future paper we will discuss treatment in CHEOPS of operations in different user data models which update the database. Here, we will simply note that the common semantics approach is still valid. That is, the DML semantic function (Figure 13) represents each updating operation of a user model by relational algebra operations.

Example 15. Suppose we wanted to delete all president information from the PRESCONG schema for president Smith. The update statement would be:

```
for each pres having lname="smith" : delete
```

The DML semantic function outputs the following sequence of statements referencing the relations of Figure 16:

```
delete  from senate
                (senate X pres)
                [pres^fname = fname and pres^lname = lname
                 and lname = "smith"]
                (congr^cong#, pres^fname, pres^lname,
                 party, seats)

delete  from house
                (house X pres)
                [pres^fname = fname and pres^lname = lname
                 and lname = "smith"]
                (congr^cong#, pres^fname, pres^lname,
                 party, seats)

delete  from congr
                (congr X pres)
                [pres^fname = fname and pres^lname = lname
                 and #4 = "smith"]
                (pres^fname, pres^lname, cong#)

delete  from pres
                pres [lname = "smith"]
```

To delete a root occurrence from a hierarchy, all descendants must

also be deleted. Thus the relational semantics of a delete of a pres occurrence consists of a sequence of four relational deletes. The first one deletes from the senate relation those senate tuples "under" the Smith pres tuple. (The expression is essentially a semi-join.) Similarly, the second and third statements delete house and congr tuples, respectively, which are under the Smith pres tuple. Finally, the Smith tuple itself is deleted.

The mappings we have specified do not supply enough information to bind update operations to the central level. "Operation mappings" are used for this purpose.

Implementation Notes

CHEOPS runs on a Virtual VAX/UNIX system. There are separate interfaces for entering schemas, for entering mappings and for querying or updating the database. All interfaces use the same interactive monitor. Schemas, mappings and queries are first read into a temporary buffer, the buffer is edited if necessary to correct errors, and a GO monitor command is executed to send the buffer to the lower levels of the system.

The user interfaces are invoked with the commands:

```
rq <db> <sch> <map>     (relational interface)
hq <db> <sch> <map>     (hierarchical interface)
nq <db> <sch> <map>     (network interface)
```

Here, <db> is the database, <sch> is the external schema, and <map> is the mapping to be used to bind the external schema to the central schema.

8. Summary and Conclusions

CHEOPS provides facilities for multiple user data models and multiple user views. These facilities are provided through the following facilities:

(1) Central Data Model with functional dependencies and subset constraints. These constructs allow the support of the major data models at the external level.

(2) Relational Semantics. All external data models are given a semantics in terms of relations. This eliminates cross data model interference and the need for multiple mapping languages and verifiers.

(3) Structure and Operation Mappings. Explicit bindings of structures and operations of user views are independent of view definitions.

(4) Mapping Verifiers. The implementation of external views by their binding to the central level database can be proved correct by the mapping verifiers.

(5) Logical Optimization. Queries and operations produced at the central level have inefficiencies introduced by the mapping process removed.

Future Work

Future extensions to this work will include:

(1) Multiple Concurrent Users. This will demonstrate the effectiveness of relational semantics in eliminating the problem of cross data model interference. Some initial work has been done on a generalization of predicate locks [EGLT] for providing

concurrency control [Klug3].

(2) Aggregate functions will be incorporated into the user query languages and into the mapping language. Aggregate functions are important for providing summary-type user views. We have identified relational algebra constructs which can model aggregate or statistical operations in any user model [Klug2].

(3) An Internal Level. The internal level and mappings from the central level to the internal level will allow investigation of efficient compilation of mappings.

(4) Automatic Remapping. After a change to the central schema, mappings to the external schemas have to be rewritten. Automatic remappings may be possible. This would be a novel approach to the problem of application program conversion.

References

[AhBU] Aho A.V., Beeri C., and Ullman J.D. "The Theory of Joins in Relational Databases" ACM-TODS 4, 297-314 (1979)

[AhSU1] Aho A.V., Sagiv Y., and Ullman J.D. "Efficient Optimization of a Class of Relational Expressions", ACM-SIGMOD 1978 International Conference on Management of Data

[AhSU2] Aho A.V. Sagiv Y., and Ullman J.D. "Equivalences among Relational Expressions" SIAM J. Comptng. 8, 2, 218-246 (May 1979)

[BeGo] Bernstein P.A. and Goodman N. "The Theory of Semi-Joins", Tech. Rep. CCA-79-27, 1979, Computer Corp. of America

[Berg] Berg J.L. "A DBMS Architecture for Prudent Managers" Information & Management, 1 (1978) 265-276

[BrPP] Bracchi G., Paolini P., and Pelagatti G. "Binary Logical Associations in Data Modelling", IFIP Working Conference on Modelling in Data Base Management Systems, North Holland, 1976

[ChMe] Chandra A.K. and Merlin P.M. "Optimal Implementation of Conjunctive Queries in Relational Databases", Proc. 9-th Annual Symp. on Theory of Computing, May, 1976, 77-90

[Chen] Chen P.P.S. "The Entity-Relationship Model: Towards a Unified View of Data", TODS 1, pp. 9-36

[Clem1] Clemons E.K. "An external schema facility for Codasyl 1978"

Proc. 5th Int. Conf. Very Large Data Bases, 1979

[Clem2] Clemons E.K. "Design of a Prototype ANSI/SPARC Three-Schema Data Base System" Proc. NCC, 1979

[Coda] Codasyl Data Base Task Group, April 1971 report, ACM, New York

[Codd1] Codd E.F. "A Relational Model of Data for Large Shared Data Banks" CACM, 13, pp.377-387, 1970

[Codd2] Codd E.F. "Relational Completeness of Data Base Sub-languages" Data Base Systems, R. Rustin (ed.), Prentice Hall, 1972

[Codd3] Codd E.F., Arnold R.S., Cadiou J-M., Chang C.L., and Roussopoulos N. "Rendezvous version 1: an experimental english-language query formulation system for casual users of relational data bases" IBM Research Report RJ2144(29407) 1978

[Date] Date C.J. "An Introduction to the Unified Database Language" Proceedings Sixth International Conference on Very Large Databases, Montreal, 1980

[EGLT] Eswaran K.P., Gray J.N., Lorie R.A., and Traiger I.L. "The Notions of Consistency and Predicate Locks in a Database System" CACM 19, 11, pp.624-633

[Fehd] Fehder P.L. "HQL: A Set-Oriented Transaction Language for Hierarchically-Structured Data Bases" ACM '74, Proceedings of the Annual Conference, 1974

[Hawr] Hawryszkiewycz I.T. "Alternate Implementations of the Conceptual Schema" Information Systems, 5, pp.203-217 (1980)

[Hero] Herot C.F. "Spatial Management of Data" ACM TODS 5, pp.493-513 (1980)

[Kent] Kent W. "New Criteria for the Conceptual Model" Systems for Large Data Bases, Lockemann & Neuhold (eds.), North Holland Pub. Co., 1976

[Kim] Kim W. "Relational Database Systems", ACM Computing Surveys, 11, pp.185-212 (1979)

[KlTs] Klug A. and Tsichritzis D. "Multiple View Support within the ANSI/SPARC Framework", Third International Conference on Very Large Data Bases, 1977

[Klug1] Klug A. "Calculating Constraints on Relational Expressions", ACM Trans. on Database Systems, 5, #3 (1980)

[Klug2] Klug A. "Equivalence of relational algebra and relational calculus query languages having aggregate functions", to appear, available as UW CSD Technical Report #839 (1980)

[Klug3] Klug A. "Locking expressions for increased database concurrency" to appear, available as UW CSD Technical Report #400 (1980)

[Klug4] Klug A. "Entity-Relationship Views Over Uninterpreted

Enterprise Schemas", International Conference on Entity-Relationship Approach to Systems Analysis and Design, 1979

[KlPr] Klug A. and Price R. "Generalized Tableaux for Chasing Expression Constraints" to appear

[Lien1] Lien Y.E. "On the Equivalence of Database Models", Bell Laboratories Database Research Report No. 3, Holmdel NJ

[Lien2] Lien Y.E. "Hierarchical Schemata for Relational Databases", to appear ACM-TODS

[Nijs] Nijssen G.M. "A Gross Architecture for the Next Generation Database Management Systems" Modelling in Data Base Management Systems, G.M. Nijssen (ed.), North Holland Pub. Co., 1976

[PaPe] Paolini P. and Pelagatti G. "Formal Definition of Mappings in a Data Base" Proc.ACM-SIGMOD Conf. 1977

[Riss] Rissanen J. "Independent Components of Relations", ACM Trans. on Database Systems 2, 317-325 (1977)

[RoSh] Rowe L.A. and Shoens K.A. "Data Abstractions, Views, and Updates in RIGEL" Proc. ACM-SIGMOD Conf. 1979

[SaWa] Sagiv Y. and Walecka "Subset Dependencies as an Alternative to embedded multivalued Dependencies" UlUCDCS-R-79-980 (1979), Dept. of Computer Science, Univ. of Illinois

[Schm] Schmid H.A. "An Analysis of Some Constructs for Conceptual Models" (to appear, Information Systems)

[Senk] Senko M.E. "Specification of stored data structures and desired output results in DIAM II with FORAL" Proc. 1st Int. Conf. Very Large Data Bases, 1975

[Shos] Shoshani A. "Cable: a chain based language for the entity-relationship model" International Conference on Entity-Relationship Approach to Systems Analysis and Design, 1979

[Sibl] Sibley E.H.(ed) "Data-Base Management Systems" ACM Computing Surveys, 8, #1 (1976)

[Ston] Stonebraker M. "Implementation of Integrity Constraints and Views by Query Modifications" Proc. ACM-SIGMOD Conf.1975

[SWKH] Stonebraker M., Wong E., Kreps P., and Held G. "The Design and Implementation of INGRES", ACM-TODS 1, #3, 1976, pp.189-222

[TsKl] Tsichritzis D. and Klug A. (eds) "The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Database Management Systems", Information Systems, 3, #3, 1978

[TsLo] Tsichritzis D. and Lochovsky F. Data Base Management Systems, Academic Press, 1977

[Zani] Zaniolo C. "Design of Relational Views Over Network Schemas, Proc. ACM-SIGMOD Conf., 1979

[ZaMe] Zaniolo C. and Melkanoff M.A. "Decomposition of relations and synthesis of entity-relationship diagrams" Sperry Research Report SRC-RP-79-76