INTEGRATION OF MODELLING CONCEPTS

W. Kent

Rapporteur: Mr. P. Rautenbach

Abstract

There is considerable competition these days among a variety of approaches to information modelling. All of them have some valid motivation. For some purposes of the conceptual schema, it may in fact be best to integrate all the approaches, since they all represent aaspects and perceptions of reality which need to be modelled. Various steps in such a direction can be discussed.

Introduction

There is considerable competition these days among a variety of approaches to information modelling. All of them have some valid motivation. For some purposes of the conceptual schema, it may in fact be best to integrate all the approaches, since they all represent aspects and perceptions of reality which need to be modelled.

What is interesting to observe is that this integration pattern can be applied to many discussion points, and seems to suggest a general approach to resolving differences. In particular, it tends to preserve everyone's viewpoint, more or less, and it seems to capture an innate characteristic of reality: the same phenomenon is perceived in different ways by different viewers. The challenge is to devise modelling constructs exhibiting analogous behaviour.

Entities and Relationships

See Figure 1.

Differences:

- Relationships perform a linking function among entities, while entities don't.
- Relationships have associated concepts of degree, domains, functionality, totality, etc., which are not relevant to entities.
- Instances of relationships are rarely named (other than by composition of the relation type name and the names of the related entities).

Similarities:

- They are both organised in terms of instances and types.
- They can both be thought of as being explicitly created and destroyed.
- Like entities, relationships can themselves have attributes: duration, importance, intensity, etc.
- Relationships can be related to entities: who authorised, who knows about it, where it is valid, etc.
- Relationships can even be related to each other: more important, implies, etc.

Thus it would seem appropriate to treat relationships as also being entities, with some additional characteristics. Relationships are thus not disjoint from entities, but a subset of entities.

Attributes, Entities and Relationships

See Figure 2.

Similarity: an atrribute of a subject can be viewed in exactly the same structure as another entity having some relationship to that subject.

Difference: not generally agreed. Some that have been suggested, concerning the related entity:

- "Existence" dependence on the subject.
- Existence assertion not required before reference.
- Not allowed to have attributes of its own.
- Not represented by "surrogates".
- Not allowed to have synonyms, not subject to unique naming rules.

There is not much advantage to making "attribute" and "entity" mutually exclusive categories, and there is considerable disadvantage. The distinction, being hard to make, requires the data administrator to make a rather arbitrary decision without any clear motivation. In most systems, some capability is lost, whichever choice is made. And, over time, the decision often has to be changed, due to changing requirements and perceptions: so why make it in the first place?

The natural integration seems to be to treat attributes as being configured of entities and relationships, with a capability for declaring auxiliary behavioural parameters (of the kind suggested above) as needed.

<u>Types and Attributes</u>

See Figures 3 and 4.

Differences:

. Philosophical: categories, taxonomy.

. Data system implementations:

- Entity type perceived as record type.

  Implies record formats, naming conventions (for example, uniqueness), consistency constraints, etc. Establishing or changing a "type" fact about an entity has enormous consequences for the system.

- The definition of each type must be known to the system, while attributes really need not be so described.

  For example, some systems don't require a description of every field in a record.

- Type information is typically segregated for the protection of the sytstem, and specially formatted to optimise the performance of the system (as in catalogues).

- Type information is thus not available as an ordinarily retrievable fact about an entity.

Similarities:

. Types and attributes both represent facts about entities (Biller).

  They ought to be accessible for data retrieval in much the same way. All such facts provide a basis for the classification of entities.

. Type information itself can be organised within the framework of entities and relationships.

. Types are named and, as with other entities, ambiguity and synonyms can occur.

  "Human", "human being", and "person" may be synonymous. "Program" might signify different categories for a computer installation, a radio station, and a government agency.

Thus, at a basic level, types can be perceived simply as certain atrributes which have special significance to the data management system, but which are in the first place facts like any others.

The term "type" can continue to be used at human factored interfaces, allowing for economy of expression, as well as catering to fuzzy concepts which we feel intuitively bound to utilise.

Types and Constraints

See Figure 5.

For information modelling, the principal purpose of types is to be able to assert what kinds of things may occur in which relationships with each other. But this is merely a special case of constraints, although that's not yet generally recognised because we have little experience with constraints.

If we can assert that things participating in certain relationships must have certain attributes, then we can subsume the type function within the constraint capability.

Redundant Relationships

See Figures 6 and 7.

In the real world (or whatever object system is being modelled), we can perceive an enormous number of overlapping relationships. The set is highly redundant, because these include all relationships which are implied by or derivable from others. There are all sorts of implications, derivations, transformations, etc. by which one set of relationships can generate another.

A relationship of degree n implies the existence of a whole family of relationships of lower degree (the projections). The existence of any two relationships involving a common entity implies the existence of a composite, and a join. Sometimes one relationship is a "special case" of another, obtainable by combinations of projection and selection: "birthdate (person, date)" is a special case of "personal-event (person, event-type, date)". (This latter type is especially perverse: a binary and a ternary representation of the same fact, wwhere a term in one case has been absorbed into the relationship name in the other. This kind of equivalence is quite common, but rarely discussed.) And of course there are all the semantic inferences, derivable from whatever axioms are known about the real world.

One, of the prime motivators in modelling, both in terms of choosing a model type and in modelling a particular object system, is to select some minimal non-redundant subset of such relationships from which the others can be inferred.

Unfortunately, there are many ways to select such a "canonical" subset, with each such group being transformable into the others by an appropriate set of operations (I won't try to define that). Some groups will consist of a few relationshiips of high degree, while some others will contain many relationships of low degree. In some groups all relationships will be in third normal form, in others they will all be irreducible (Hall, Falkenberg), in others they will all be binary. Different groups will be minimal in different senses: some will have the fewest relationships; some will be the smallest groups containing only irreducible relationships, and so on.

A central issue in modelling theory is this: which group shall be selected as a canonical representation of the relationships involved? Most replies presume that it should be a minimal (non-redundant) set; the main debate seems to be between minimal binary and minimal irreducible. But maybe "minimal" is not the right answer in the first place. So long as the dependences among the redundant relationships were expressed in the model, a redundant set might be useful, and would allow some benefits. The derivable relationships could be named, and could be the targets of "view mappings". The argument over which form ought to be used can be avoided. Thos with strong leanings could limit themselves to one or another of the minimal forms; all would still be using consistent subsets of the general model. Of course, with redundant relationships, it may become necessary to specify (perhaps not in the conceptual schema) which are directly changeable, and to be sure that the propagated consequence of change are well defined.

We might want to explore some sort of "implication structure" as a system of possibly redundant relationships together with implications among them. One interesting question is whether such implication structures can be partitioned into clusters such as, for example, a ternary relation and its three binary projections. These clusters might then be considered as various manifestations of a single fact. We might thus achieve the metaphysical phenomenon of multiple constructs behaving as one.

In some sense such a cluster is a single phenomenon which can look different in different situations. For example, different relationships from the cluster can be explicitly manipulated in different situations, with the others being automatically maintained consistent via the implications. In particular, various combinations of relationships in such a cluster can be selected for mapping to data structures in various implementations.

Schema and Data Base

See Figure 8.

Some recent work in ISO/TC97/SC5/WG3 (a working group addressing the definition of the conceptual schema) suggests that even the boundary between schema and data base may not be as well defined as we thought.

Differences

The differences fairly leap out at us, since we perceive schema and data base as such different constructs. Primarily, the schema defines types and the data base contains instances; the schema specifies rules and the data base contains things governed by the rules.

Furthermore, taking a certain point of view about the conceptual schema in particular, the schema is data independent while the data base is realised in very specific data structures and formats. One might go so far as to consider the conceptual schema to be an "enterprise description", expressed in such semantic terms as entities and relationships, with perhaps no mention of such things as records and fields.

Also, as with types, changes to the schema imply some inpact on the way the system operates (for example, perhaps requiring physical reorganisation), while changes to the data base are simply accepted passively.

Similarities

Both schema and data base provide information about the real world. As before, the "type" of something is useful information which ought to be available along with other facts. So also should the name of the relationship existing between certain things. Even constraints may contain useful information which ought to be retrievable, such as the maximum allowed of something. From an artificial intelligence point of view, "employees are assigned to departments" is a single fact, part of the "knowledge base". In some systems (Chang, Hemphill), queries are answered using the combined information in both schema and data base. Virtual relations, accessible to users, can be defined in the schema, having no direct realisations in the data base.

The earlier argument for merging types and attributes in itself argues for the merging of schema and data base.

Rules often span the two domains. The allowable set of values for a department field consists of the keys of the current set of department records.

Conversely, instances often occur in the schema, in the definitions of some rules. For example, the list of valid colours might be defined in the schema. Asking for the list of currently valid departments and the list of currently valid colours would seem to be similar requests, yet one has to be directed to the data base and the other to the schema, through totally different kinds of interfaces.

Beyond that, certain information spans the boundary between schema and data. There can be relationships between types and instances (other than the obvious one). Authorisation information may have this form, as may records of ownership and creation.

If such things as triggered actions or other propagated effects are provided in the data base, then updates to either the data base or the schema might entail extensive system activity. Hence even the update sensitivity might not be such a sharp distinction.

Integration

The integration in this area is not so well developed, and ought to be characterised as work in progress. It is in a sense a test of the principle of merged concepts, to see if that principle can in fact generate new and useful results.

It is difficult to think of a merging of conceptual schema with data base in either the internal or external view senses. Part of the resolution may involve postulating something at the same level as the conceptual schema but consisting of the corresponding instances rather than the types. Such a construct is called an "information base" in ISO/TC97/SC5/WG3. The information base is what would be accessible if manipulative operations were provided at the conceptual schema level (for example, the "conceptual level programming language" of Olle). It is also those things which exist in principle as intermediaries in the materialisation path from internal to external data, although we always say that practical implementations will leap those two mappings in one jump.

It is somewhat easier to imagine the union of conceptual schema and information base, with the boundary between them being somewhat fuzzy. Thus we deal with a unified abstract information system, of which the conceptual schema and information base are subsets, possibly imperfectly defined and possibly overlapping.

Conclusion

We have suggested a different approach to the conceptual model. It avoids assuming that certain arguments need to be settled at the conceptual level, since the arguments themselves argue the existence of the alternative viewpoints in reality. We don't need to eliminate redundancy, but to represent and manage it, as an aspect of reality.

A model with merged concepts offers a more complete modelling of real situations, in the various ways they may be perceived. Also, the need to make certain arbitrary decisions is avoided.

What is lost is a certain simplicity -- but can any faithful model of real life be simple?

Discussion

Professor Schmidt commented that types are not simply sets, but are also related to their operational characteristics, that is their dynamic characteristics.

Mr. Kent thought this view was more related to data type, whereas he had been talking more about entity type. He pointed out, however, that operational restrictions can also be represented by relationships between entities. As far as the static/dynamic argument was concerned, Mr. Kent acknowledged that his work concentrated on the static view.

Dr. van Rijsbergen referred to Mr. Kent's dissatisfaction with the concept of record and asked what alternative he was working towards. Mr. Kent replied that he had not yet arrived at an alternative: he was just turning the corner from critisism to proposing alternatives.

Professor Kopetz wanted to know if Mr. Kent saw a fundamental difference between entity type and name.

Mr. Kent again saw naming falling into his entity/relation framework. He added that he had not considered the problems of names applied to program variables.

Professor Kopetz asked whether Mr. Kent distinguished between proper name and class name, to which Mr. Kent replied that he simply considered a class as having a proper name.

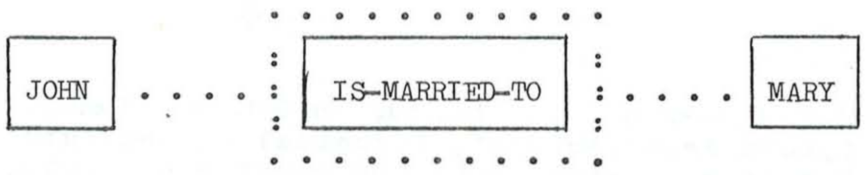
References

ANSI	The ANSI/X3/SPARC DBMS Framework, Report of the Study Group on Data Base Management Systems, (D. Tsichritzis and A. Klug, editors), AFIPS Press, 1977.
Astrahan	M.M. Astrahan et al., "System R: Relational Approach to Data Base Management", ACM Transactions on Data Base Systems 1 (2), June 1976, pp. 97-137.
Biller	H. Biller and E.J. Neuhold, "Semantics of Data Bases: The Semantics of Data Models", Information Systems 3 (1), 1978.
Bracchi	G. Bracchi, P. Paolini and G. Pelagatti, "Binary Logical Associations in Data Modelling", in Nijssen .
Chang	C.L. Chang, "DEDUCE 2: Further Investigations of Deduction in Relational Data Bases", in J. Minker and H. Gallaire (eds.), Logic and Data Bases Plenum Press (forthcoming).

Dict	DB/CD Data Dictionary General Information Manual, GH20-9104, IBM, San Jose, California.
Falkenberg	E. Falkenberg, "Concepts for Modelling Information", in Nijssen .
Hall	P.A.V. Hall, J. Owlett and S.J.P. Todd, "Relations and Entities", in Nijssen .
Hammer	M.M. Hammer and D.J. McLeod, "On Data Base Management System Architecture", Technical Report TR79-4, Computer Science Dept., University of Southern California, April 1979.
Hemphill	L. Hemphill and J. Rhyne, "A Model for Knowledge Representation in Natural Language Query Systems", IBM Research Report RJ2304.
Nijssen	G.M. Nijssen, Modelling in Data Base Management Systems, North Holland, 1976.
Olle	T.W. Olle, "Multistage Data Definition in a Multi-component DBMS Architecture", in B. Sceneiderman, Data Bases: Improving Usability and Responsiveness, Academic Press, 1978.
Sharman	G.C.H. Sharman and N. Winterbottom, "The Data Dictionary Facilities of NDB", in Proc. Fourth Int'l. Conf. on Very Large Data Bases, Sept., 13-15, 1978, Berlin, Germany.

Is a marriage an entity or a relationship?

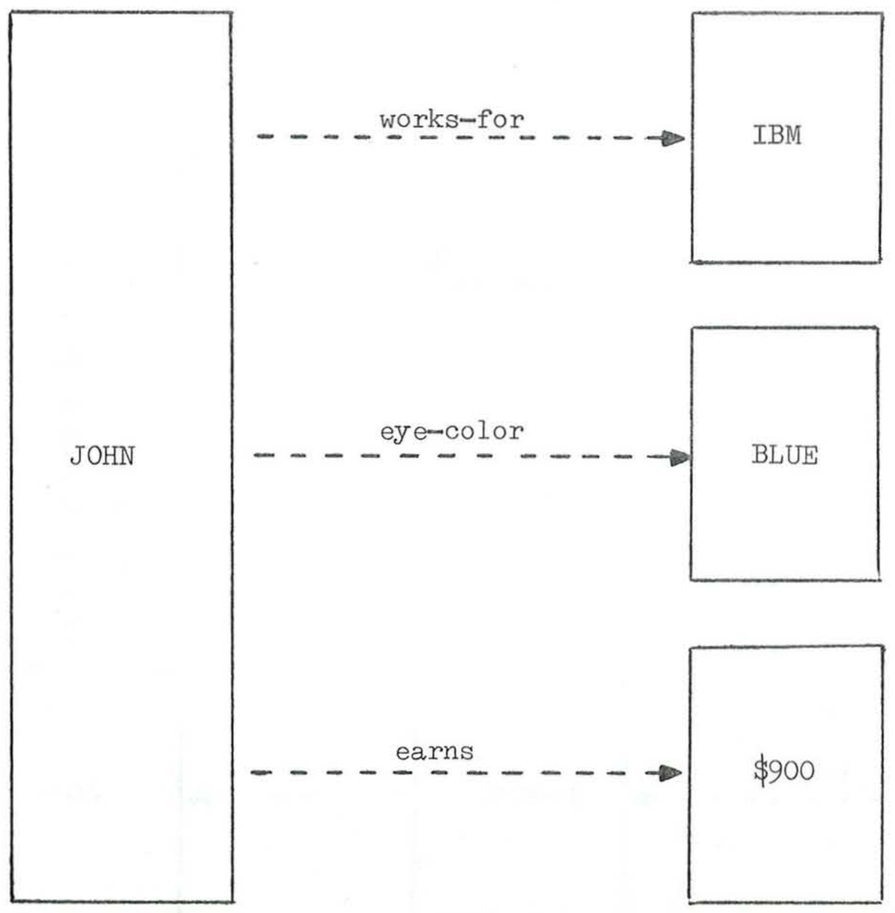
Yes.



A relationship can be an entity which also can link other entities.

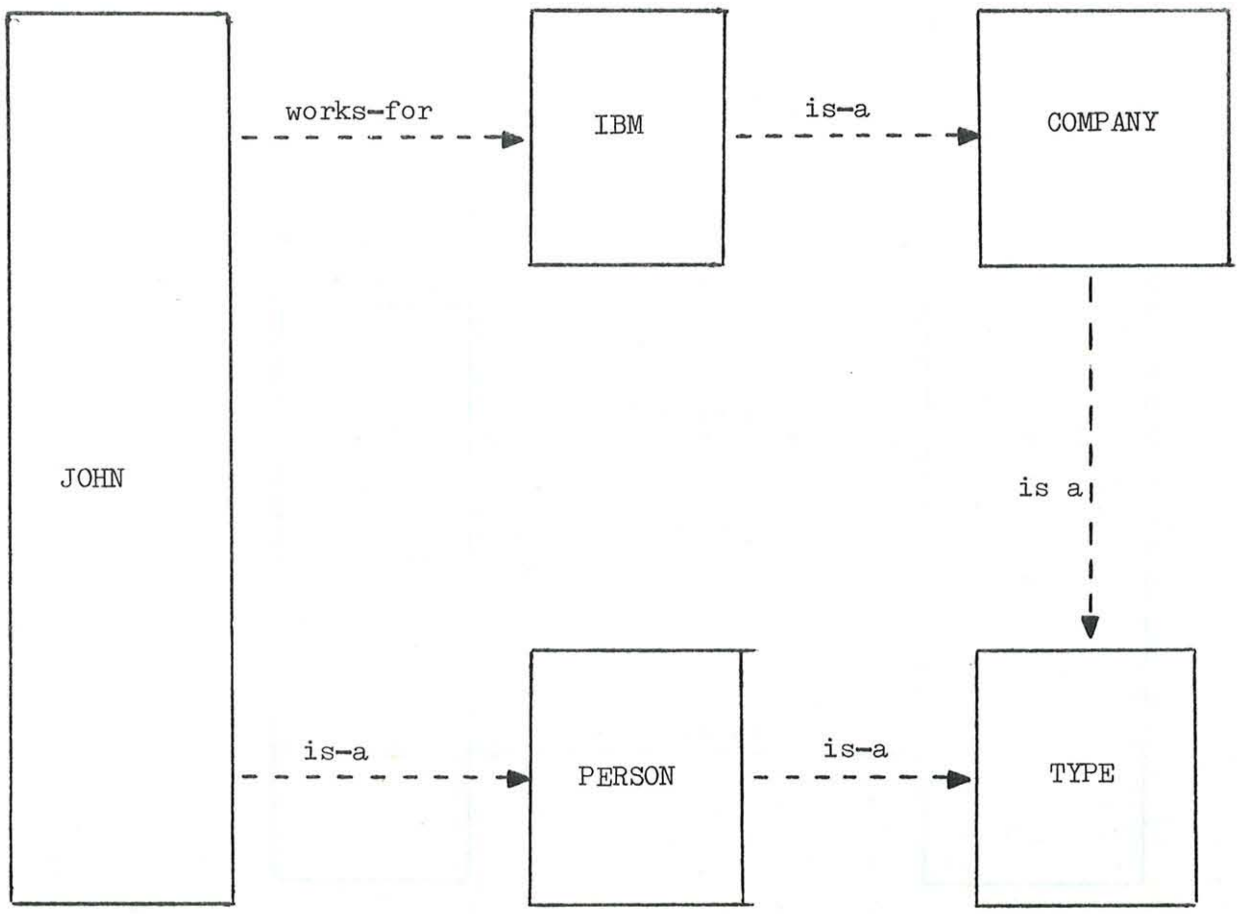
Ralationships are a subset of entities.

Figure 1 Entities and Relationships



We can <u>define</u> atrributes in terms of entities and relationships.

Figure 2 Attributes, Entities and Relationships



We can define types in terms of entities and relationships.

Figure 3

John is a:

Person	Vertebrate
Mammal	Male
Employee	Customer
Programmer	Redhead
Millionaire	American
Californian	Parent
Genius	Bore
Liberal	Golfer
etc.	

Which are types?

Why must we choose?

Figure 4 Types and Attributes

$$x \text{ earns } y \Rightarrow y < \$10000$$

$$x \text{ earns } y \Rightarrow \text{E}z \mid x \text{ works-for } z$$

$$x \text{ works-for } z \Rightarrow$$
$$x \text{ is-a person AND } z \text{ is-a company}$$

$$x \text{ owns } y \Rightarrow$$
$$x \text{ is-a person OR } x \text{ is-a company}$$
$$\text{AND}$$
$$y \text{ is-a vehicle OR } y \text{ is-a furniture}$$

Figure 5 Types and Constraints

There was a getting-married event.
    The event occurred in 1970.

A marriage exists.
    It has existed since 1970.

John is married to Mary.
    They have been so related since 1970.

John is a married person.
    He has been since 1970.

```
x is-a married-person <==>
Ey | x is-married-to y <==>
Ez | z is-a get-married-event
AND x was-married-in z
AND y was-married-in z
etc.
```

A model which doesn't capture all of these, with the inter-dependences, is missing something.

Figure 6 Logical Redundancy

N-ary vs. binary:

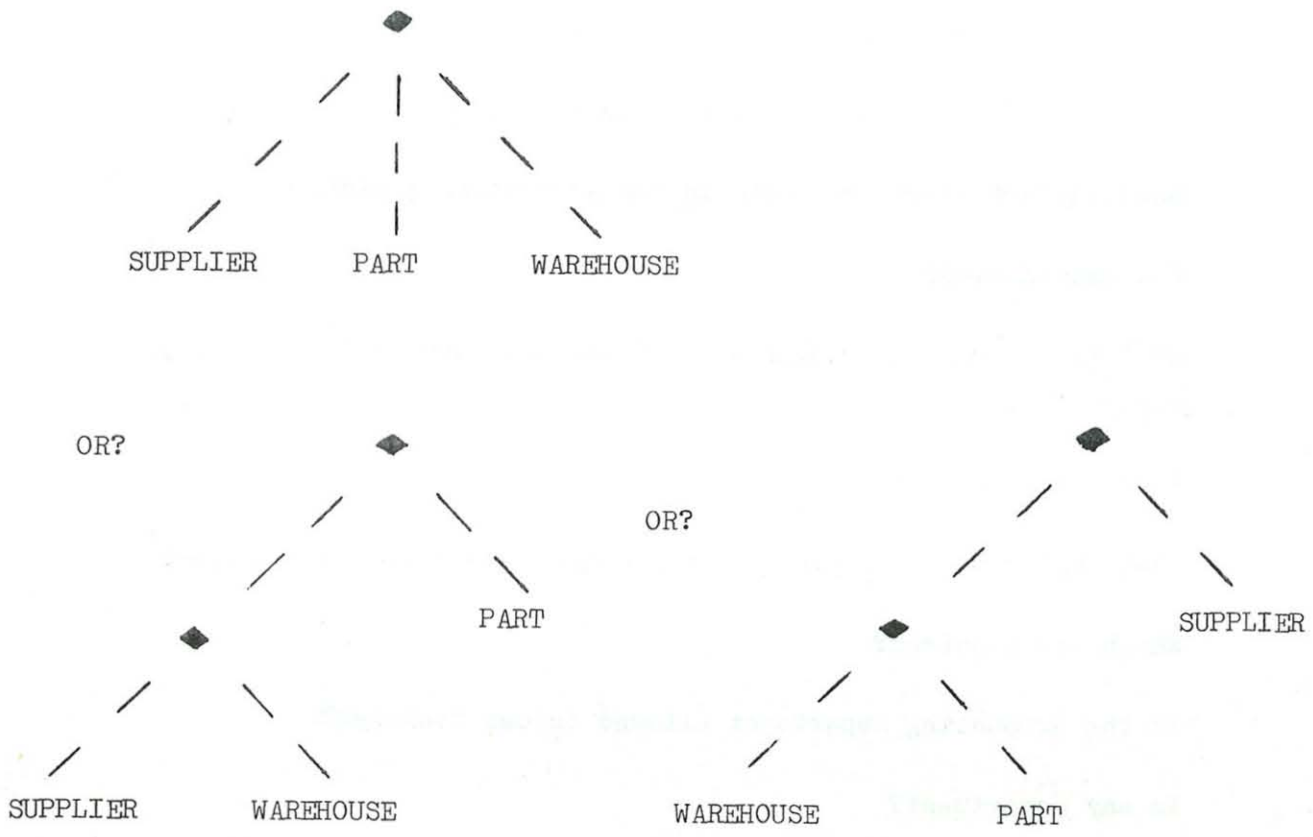


Figure 7 Logical Redundancy

The data/description continuum:

Does Fred Smith work in the Accounting department?

How is Fred Smith connected with the Accounting department?

How many employees are there in the Accounting department?

How many managers?

What is the maximum number of employees allowed in the Accounting department?

In any department?

What skills do the employees of the Accounting department have?

Which are required?

Is the Accounting department allowed to own vehicles?

Is any department?

Figure 8