McLeod: This tutorial surveys recent research in data abstraction and conceptual modelling in the database area. In particular, I will present recent results in the area of higher level data models, sometimes called semantic data models.

The principal focus of database research has been on database management systems (DBMS's). A DBMS is a general purpose system which provides tools for logical structuring of formatted data, access interfaces, data control (protection and integrity) and efficient storage of and access to data. I would like to emphasize that results in database research, while applicable to tailored systems, e.g., an airline reservation system, are really of a general purpose nature.

Abstraction in Database Systems

First, I would like to turn to abstraction in database systems. In Dave Parnas's view, abstraction is a synonym for "good." To me abstraction is a knowledge representation technique in which the description of the essence of a concept is substituted for the concept itself, e.g., a detailed model of the concept is mapped into a higher level, less detailed one.

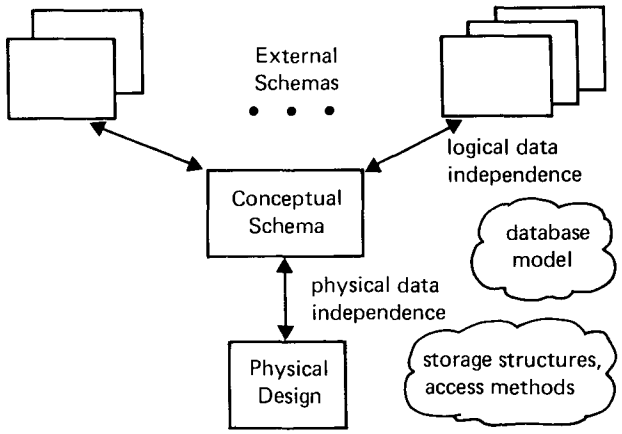
Cristian: How do essence and concept differ?

McLeod: The bs factor here is pretty high. It will take most of the conference to resolve issues like that. Let me just say that there is a one to many mapping between the essence of a thing and the ways a thing is looked at.

Three Kinds of Database Abstraction

McLeod: In database systems, abstraction concerns representation (or data) independence and is one of the main database objectives. Representation independence is the preparation of the physical representation of data from its logical specification, which is used to capture the meaning of data. The levels of database abstraction attempt to support evolution, suppression of detail, and multiple views.

The conceptual schema is a specification, using a database model, of a particular database. A database (or data) model is a general purpose formalism for describing logical properties of data in a schema. The physical design, or internal schema, is a physical representation of the data which uses storage structures and access methods. External schemas are specifications, derived from the conceptual schema, of different views of the same database. Physical data independence is attempted by separating the conceptual schema from physical design and relating them through a mapping. Hence,



ANSI-SPARC levels of Database Abstraction

it may be easier to alter the physical design without impacting the application logic. A similar separation of logical and conceptual schemas attempts to achieve logical data independence, reducing the impact of changes to views on the conceptual schema and vice versa.

Rich: Could you say what you mean by an external schema?

McLeod: Sure, an external schema is a restructuring or subsetting of a conceptual schema. For example, a freighter database is a subset of a ship database.

Rich: Why are there three levels? Why not four?

McLeod: The three levels are types of levels. You can have many levels of abstraction. In fact, you can define external schemas on top of external schemas.

Smith, J.: I think the three levels come from the fact that the conceptual schema is abstract. The internal schema represents those abstract properties in a way which is convenient for a computer to use. An external schema presents those appropriate abstract properties in a user-oriented way.

Rich: Oh, the external level concerns user interfaces.

Codd: External schemas involve more than syntactic sugar. A view at the external level could be an equi-join of two n-ary relations defined at the conceptual level. The properties of such a view are quite different from the properties of the relations they are made from.

Weber: Different primitives are used at each level. For example, the physical level is record-oriented.

The conceptual level may be concerned with, say, entities and relationships, or relations. The external level should present application-oriented primitives.

Data Models

McLeod: For the rest of the talk I will concentrate on the conceptual schema--how to define them, data models and so forth.

Classical DBMSs focus on formatted data. They are not intended for continuous signal data, text files or things of that sort. They stress efficient data storage and access and have very limited deductive and reasoning capabilities common in AI systems. They assume a rigid and relatively uniform organization of data. Hence, the size of the database is large compared to the size of the description; there are many instances of each type.

A data model is a set of logical structures plus a set of matching operations for manipulating those structures. The classical data models fall into three categories: hierarchical, network, and relational. They are not disjoint and not all analogous. The network model typically means the CODASYL DBTG network model. These models differ in their degree of data independence. The CODASYL DBTG data model is an example where there is poor physical data independence. There is a lot of implementation detail in logical specifications. Data models differ in the level of user views and types of access operations. The hierarchical model structures everything in the form of hierarchies, the network model, networks, and the relational model, in the form of n-ary relations.

In the database community, there is a changing view of performance. Originally performance was strictly related with run-time and access efficiency. It is now felt that the most costly aspect of the system life-cycle is system evolution, e.g., changing storage structures, as usage patterns change. So, the definition of performance now includes the amount of effort it takes to change, modify, and maintain a system over its life-cycle.

There are four main components of a data model. The first is a data space, a collection of elements and relationships among the elements. Primitive data objects are distinct from relationships among those primitive objects although no rigid distinction can be given. The second component is a collection of type definition constraints used to impose restrictions on and to structure relationships. Manipulation operators, the third component, support the creation, deletion and modification of elements. The fourth component is a predicate language used to identify and select elements from the database according to their properties.

Ships

ID	Captain	...
	⋮	
995	Brodie	...
	⋮	

Freighters

ID	Cargo
	⋮
995	Wheat
825	Steel

Hayes: Are the elements things in the "real world" that we talk about using symbols in the database? (McLeod) The elements are entities within the database system. They are intended to correspond to things in the "real world."

Types and Instances

McLeod: In classical data models the data space is partitioned into types and instances. Types correspond to the schema or the logical structure of the database and instances correspond to data, the population of the types. There is an unlimited variety and number of inter-instance relationships, that is, relationships among elements. There are no inter-type relationships. Types are disjoint. Types and elements are related by the instance-of relationship. One type can have many instances.

The classical data modes have operations associated with them, both general purpose set operations and separate operations for manipulating types and instances, e.g., create_instance_of (add tuple), create_type (add relation). In addition, these models have predicate languages for querying. Hierarchic and network models have (limited) boolean predicates. The relational model offers a first order predicate language.

Consider a relational database to keep track of ships, freighters, officers, and privileges. Each entity is represented in a relation. A tuple in the ships relation, represents a ship with id 995 and Captain Brodie. Freighters, we have 995 and the cargo is wheat. The relation officers currently represents two officers Brodie and Zilles, both of rank of admiral. The privileges relation states what officers can do. Admirals can sail and sink ships. This is a many to many association. Two tuples can be used to denote different properties of a single entity such as the association between officers and privileges based on rank. Brodie and Zilles rank as admirals; both can sail and sink ships.

There are four types in the example: ships, freighters, officers, and privileges. The instances are the tuples in the relations. There are an unlimited number of inter-instance relationships, e.g., relations between names and ranks, ranks and privileges, ids and cargos, etc.

Codd: Dennis, I think it is terribly misleading, if you're representing that as the relational model, to suggest that types are limited to the relations themselves. Domains, which you haven't mentioned, play a very important role in the relational model. (McLeod) I agree.

McLeod: One limitation of classical database models is that there is nothing in the data specification which says that, say, freighters are a subtype of ships. So there are very limited ways in which to interrelate types.

Officers

Name	Rank
Brodie	Admiral
Zilles	Admiral
⋮	

Privileges

Rank	Privilege
Admiral	Sail Ship
Admiral	Sink Ship
⋮	

Reiter: I'm afraid I'm confused. I can understand what a relation is but I don't know what a type is.

Hayes: Are all relations types?

Codd: I would say neither yes nor no. Relations have both a type aspect and a set aspect. We want to keep the type aspect separate in our thinking from the set aspect.

Capturing Meaning in the Database

McLeod: Recently there has been considerable work on higher level data models which try to remove some of the restrictions of classical data models to capture more meaning of data. Questions being considered are: what's a type? what's an element? what's a relationship? Are these things distinct or are they just multiple perspectives on the same notion?

One characteristic that has dominated the semantic model approach is to introduce relativism; that is, to look at things as attributes, types and instances. The notion of freighter as a subtype of ship which is a higher level type. One type of ship is a freighter, another is a tanker, so on. Freighter serves both the role of type with instances, namely ships of type freighter, and as an instance of another type, namely the type ship. One could also view freighter as a relationship among its attributes, such as hull, captain, and so forth. Recent work on semantic data models attempts to accommodate these relativistic issues or points of view and multiple perspectives on the same data. John and Diane Smith have done some work on these notions using aggregation and generalization. Ted Codd has extended the relational model to capture more meaning. I did some work with Mike Hammer on a model called SDM. There is also Peter Buneman's FQL, Dave Shipman's Daplex and the list goes on.

Semantic models classify relationships in the form of meta-relationships such as instance_of (classification), attribute_of (aggregation), and subtype_of (generalization).

Another thing of interest in semantic data models is abstract objects as opposed to just syntactic objects, similar to the programming languages distinction between abstract objects and syntactic objects such as integers and real numbers. Semantic models have tried to build into the model itself important types of constraints, such as subtypes or subset constraints, functionality, and so on.

Finally, with regard to operations or manipulation primitives associated with semantic models, the trend is toward the unification of the schema with the data--types with instances. Most semantic models have a generally applicable set of operations, which match the semantic model, but with very limited primitives for interconnecting those operations to define application_specific operations.

In closing, let me relate the database issues I have mentioned with the topics of the workshop sessions. The issues of type, element and relationship concern expressibility and relativism. They will be discussed in the session on types. The session on behaviour relates to the general purpose operations in classical data models for defining application specific operations. The session on specification relates to database models since their main purpose is for the specification of databases. The session on consistency relates to the levels of database abstraction and consistency among them, say between external schemas and conceptual schemas, as well as within the conceptual scheme. The session on relationships relates to structuring primitives for interconnecting both types and behaviour. Finally, the session on application will discuss experience with database tools and techniques.

Terminology

Balzer: Let me say two things. First, it seems that there is the same amount of confusion in the database area that we see in AI. Maybe there is a lot to be learned here. Second, if we continue to put off issues of terminology we're going to be in deep trouble.

McLeod: OK, type and instance are not terms typically used in the database area.

Codd: Oh, I disagree with that absolutely and completely. Type and instance are vital to distinguish in databases, and they've always been distinguished, as far as I'm concerned.

McLeod: I didn't mean that those notions are not critical, they're incredibly important. All I'm saying is that I don't normally use the terms type and instance very much with respect to databases.

Rowe: Let me say something that I think Ted was getting at earlier, and maybe this is what Dennis has been trying to say. The programming language point of view makes a distinction between type and the name of the type, e.g., pointer-to-node is the name of a type, it is not an instance of that type. Whereas if x is a variable whose type is pointer-to-node we have an object x whose name is x, but whose type has a different name, i.e., pointer-to-node. I think what Dennis was getting at is that very often in the database community, they do not distinguish a separate name to identify the type of the object. In fact, they have a single name. When you talk about people, you're talking both about a structure which contains instances of objects which are people.

McLeod: I don't see a unified notion of type in data model research. That is one of the important things we want to focus on in the workshop.