

2. The Conceptual Data Model.

- [Syllabus](#)
- [End of Page](#)

CONCEPTUAL DATA MODELLING is the first stage in the process of TOP-DOWN DATABASE DESIGN. The aim is to describe the information used by an organisation in a way which is not governed by implementation-level issues and details. It should make it easy to see the overall picture so that non-technical staff can contribute to discussions.

A common method of analysis involves identifying:

1. the ENTITIES (persons, places, things etc.) which the organisation has to deal with.
2. the ATTRIBUTES - the items of information which characterise and describe these entities.
3. the RELATIONSHIPS between entities which exist and must be taken into account when processing information.

In the abstract, or illustrated by superficial examples, this looks a very simple idea. An explanation may put forward a car as a typical entity, point to make, colour, registration number as obvious attributes, and suggest owning and driving as relationships in which it takes part. But applying the method of analysis to some useful purpose in a working organisation will be difficult, simply because the world does not fit so neatly into boxes.

2.1. Entities.

2.1.1. Defining.

In a global data model for a hospital, for example, the entity names employee and patient are likely to occur. In some situations the same individual may be both an employee and a patient; is s/he one entity or two? In such a case it is more accurate to define a fundamental entity such as a person able to take a role in the relationship.

Note: the name given to an entity should always be a singular noun descriptive of each item to be stored in it. e.g. patient NOT patients.

2.1.2. Composite entities.

Sometimes an entity as defined in 2.1.1. may be considered as a single entity or as multiple entities. For example, a firm may have a number of different premises for the delivery of goods or supply of services - multiple entities - but one head office for the payment of accounts - a single entity. Similarly, a single object - e.g. an aeroplane - may be made up of

many components. These components have a hierarchical / recursive relationship: e.g. A is made up of B and C which in turn are made up of X, Y and Z. It is impossible to treat such entities at a single level; for example the CAA has rules about the expected life of different aircraft components but these must be related to the flight history (take-offs, landings, flight hours) of the plane in which they are fitted.

2.1.4. Aggregates.

For some applications, e.g. stock control of low cost items, we are not concerned with individuals but with AGGREGATES - e.g. how many cans of baked beans are in the warehouse? It is easy to overlook the fact that a definition refers not to an individual but to a type of object. More problems arise when manufactured objects change status part-way through their lives - a car on an assembly line may be indistinguishable from those immediately before and after it, but it must at some point acquire an individual identity for sale and registration.

2.1.5. Sub-classification of entities.

The need for sub-classification of entities arises frequently. For example, a transport pool handles resources which in a global data model might be all classed as vehicles. Although certain attributes and relationships may be common to all vehicles, they are not inter-changeable and for practical purposes must be subdivided into vans, lorries, limousines, etc. each with their own particular characteristics.

2.1.6. Physical boundaries.

It may be difficult to determine the physical boundaries of entities which are identified primarily by geographical location. For example, a motorway is a road which is part of one continuous system; it may, however, be necessary to specify points on the motorway, where one section stops and the next begins.

2.1.7. Events.

So far we have been talking about entities which are fairly tangible objects; however, there are many more that are more abstract, for example a sale or a birth. In terms of the overall model these are events which trigger off processes. Nevertheless, we may wish to count them, attach attributes to them, and relate them to one another, as if they were entities. The more abstract the entity, the more difficult it is to decide whether it should really be classed as an event or a relationship.

2.1.8. Messages.

Similar difficulties occur with entities which exist only as messages within an existing clerical information system. As a piece of paper, a receipt is a tangible object, but it is doubtful whether it should be a first class citizen of a conceptual data model, since in different technological situations its functions could equally well be performed by a notch on a stick or a series of pulses on an EDI network.

As these examples indicate, entities are not easy to pin down and compromise is required

between a fruitless search for some ultimate truth and a short-term expedient which will break down as soon as it is seriously tested. It is necessary to consider the purpose for which the database is being designed: what sort of questions are likely to be asked and how future developments may require the initial requirements specification to be extended.

2.2. Attributes.

Attributes are pieces of information ABOUT entities. The analysis must of course identify those which are actually relevant to the proposed application. Attributes will give rise to recorded items of data in the database so it is important that nothing potentially useful is omitted here even though not all may be included later.

At this level we need to know such things as:

- attribute name.
- the domain from which attribute values are taken.
- whether the attribute is part of the entity identifier.
- whether it is permanent or time-varying.
- whether it is required or optional for the entity.

2.2.1. Attribute names.

Names should be explanatory words or phrases; codings and/or abbreviations should be postponed until the implementation level. They should enable users of the data model to see what is being recorded, and identify where the same piece of information occurs in different places (e.g. through a DATA DICTIONARY.)

2.2.2. Domains.

A DOMAIN is a set of values from which attribute values may be taken. Examples are: dates, sums of money, temperatures, grid references, colours, gradings and nationalities. In one way the more definite we can be here the better - it is more helpful to know that an attribute is a linear measurement than that it is a number - but precise formatting specifications are not required. A date is a date, however it is stored or presented, and the conceptual data model is not the place to specify that it is a six-digit number in the format ddmmyy. On the other hand it will be useful to know the kind of accuracy which may be expected for this attribute - is the measurement to the nearest day, week or year?

2.2.3. Identifier attributes.

We must distinguish between attributes which just describe an entity (and perhaps change over time), and those which help to identify it uniquely. This will certainly influence the lower level design. But some care is required. In natural discourse very few things have one unique name or identifying attribute - they are distinguished by a combination e.g. name, address and date of birth for a person. In formal information systems, it is convenient to give unique but arbitrary labels to persons or objects (e.g. National Insurance Number, Vehicle Registration Number) so as to have quicker and easier means of identification. But however convenient such KEYS are at the implementation level they should not be introduced into the conceptual design unless they are already familiar to users of the information, and there is an

understandable process for getting from the entity itself to its identifying attribute.

2.2.4. Time-varying attributes.

It is important to know which attributes may change their values over time, as they will have to be updated when the system is implemented. It may also be necessary to hold previous values somewhere for AUDITING or the production of HISTORICAL REPORTS.

2.2.5. Optional attributes.

Entities may have attributes whose values will sometimes be unknown or irrelevant. This happens with very general entities such as vehicle more often than with those more narrowly defined. A record of which attributes are OPTIONAL or MANDATORY will help when defining general consistency constraints for the data base at lower levels.

2.3. Relationships.

In many applications one external event or process may affect several related entities, requiring the setting of LINKS from one part of the database to another. Important information to be recorded is:

2.3.1. Number and type of roles.

Like an attribute, a relationship should be named by a word or phrase which explains its function. The same applies to the role names within the relationship, e.g. owner, driver etc. Role names are different from the names of entities forming the relationship: one entity may take on many roles, the same role may be played by different entities. Where a relationship is symmetrical (e.g. 2 roads intersecting) role names are arbitrarily assigned.

An important point about a relationship is how many entities participate in it. In practice BINARY RELATIONSHIPS are most convenient for data modeling; however it is by no means the case that all relationships must necessarily be binary in this sense. For example, consider a model built around vehicle manufacturers, types of vehicle and garages. In such a model we may wish to represent the information specified by a TERNARY RELATIONSHIP e.g.

- (Vehicle_Manufacturer, Vehicle_Type, Garage)
- Vehicle_Manufacturer supplies Vehicle_Type to Garage.

This ternary relationship is not in general equivalent to the combination of the three binary relationships:

- Vehicle_Manufacturer SUPPLIES Vehicle_Type
- Garage SELLS Vehicle_Type
- Garage IS SUPPLIED BY Vehicle_Manufacturers

For example the information that:

1. Ford supplies cars to Hills

tells us more than the combination:

2. Ford supplies cars
3. Hills sells cars
4. Hills is supplied by Ford

Relationships (2), (3) and (4) only tell us that:

- Ford supplies cars to some garage(s)
- Hills sells cars supplied by some supplier
- Ford supplies some type of vehicle to Hills.

One cannot validly infer (1) from (2), (3) and (4); e.g. it may be that Hills only sells Ford trucks, not cars. The false inference (2, 3, 4) \Rightarrow (1) is sometimes called the connection trap.

2.3.2. Degree.

Relationships may be:

- ONE-TO-ONE, e.g. Building - Location,
- ONE-TO-MANY, e.g. hospital - patient,
- MANY-TO-MANY, e.g. Author - Book.
- RECURSIVE, e.g. Manager - Employee.

To decide the DEGREE of a relationship one must be clear about whether individuals or types are involved. The relationship (vehicle_manufacturer, vehicle_type) is 1-to-many if we are talking about an individual manufacturer, but many-to-many where (vehicle_manufacturer, vehicle_type) are types.

Consider also the time dimension - whether the relationship is being recorded at a single point in time or over a period of time. For example, in Britain the legal relationship husband-wife is one to one in the first case, but potentially many to many in the second.

2.3.3. Optionality.

A relationship may be required or optional for either participant, e.g. a piece of property must be owned by a person but not all persons need own a piece of property.

2.3.4. Permanence.

A relationship may be defined as permanent (e.g. parenthood) or temporary (e.g. ownership or employment). If temporary but required by one of the participants (e.g. ownership) it must be transferable.

2.3.5. Dependencies.

One relationship may necessarily exclude or imply another, or be excluded or implied by another. Membership of the City University Students Union necessarily implies membership of the University itself.

Many of the above constraints will require consistency checks on incoming data. Some DBMSs allow such checks to be declared as part of the logical data model so that they are applied automatically whenever the data base is updated.

- [Syllabus](#)
 - Previous: [An Introduction to Databases and DBMSs.](#)
 - Next: [The Relational Data Model](#)
 - [Top of page.](#)
-