Dealing with databases

Recall what a database is mathematically. It is a table of data elements organised in a systematic way. Usually, a database will have a *key*, an index set, K. The actual data can be considered to be an array or vector of elements from a collection of sets.

KEY (ID) K	DATA SET A	DATA SET B	……..	DATA SET G
k_1	a_1	b_1		g_1
k_2	a_2	b_2		g_2
…				
…				
k_m	a_m	b_m		g_m

We can define this as a function that takes a subset of the set K and returns an array – i.e. an element of the product set A x B x …… x G which is the set of all vectors of the form (a, b, …, g).

So we could call this function D and so:

$$D : K \rightarrow A \times B \times \ldots\ldots \times G$$

or

$$D : K \rightarrow A_1 \times A_2 \times \ldots\ldots \times A_n$$

In the example the *domain* of D is the set $\{k_1, k_2, \ldots, k_m\}$ and D describes the **current** *state* of the database.

So a database is like a big matrix – here it is an *m x (n+1)* matrix where n+1 is the number of columns, there being n columns of actual data and one column of the key values.

If we put some more data, eg. a_{m+1}, b_{m+1}, …,g_{m+1}, into it then we will get a new state, the table will have a new row indexed by a new value of K, eg. k_{m+1}

and the contents of the new row will be

k_{m+1}	a_{m+1}	b_{m+1}	…	g_{m+1}

We could call this new state of the database D' and

$$\text{domain of D'} = \text{domain of D union } \{k_{m+1}\}$$

Of course, we can change data in the database which means replacing elements of a row by new elements – eg. b_2 is replaced by b'_2

One of the key things about databases is the ability to query them – to find out what values are in them.

How to describe the value in an arbitrary position of the database.

Suppose we want to find what the value is in the B column at key value k_i
We use a *projection* function.

For each row E we define a function $proj_E$ which takes a key value and returns the data element in column E and row k_i

So $proj_E (k_i) = e_i$

As usual we must remember that this only applies for the current state of the database D, if we change D then the value of this function will change. We should really regard the projection function as a function of both the set of all possible databases as well as the set of keys. However this makes the notation too complicated. Just remember that D is changing from time to time.

We can use the projection function to work things out.

Suppose that we had a database containing orders or collections of things – the columns could represent the numbers of components, calories of ingredients, prices of seats, results of matches, etc.

A given query might ask for weighted the sum of these data elements for a given row.

$$Total (k_i) = \sum_{j=1 \text{ to } n} weight_j \times proj_j (k_i)$$

where $weight_j$ is the fixed price/calorie level for the things in that column.

Thus Total is a function $K \rightarrow Z$ for a given state of the database (Z is the set of all integers).

Example.

Key	Name	Address	Current balance	Discount?
0001	Fred	Walkley	100.00	no
0002	Jane	Crookes	0.00	yes
0003	Pete	Barnsley	250.00	yes

A simple query is just to print out the value of a customer's name, eg.

Print_Name: Key → Name

Defined by

$$Print\_Name (k) = proj_1 (k) \text{ if k exists else 'error'}$$

So Print_Name (0001) = Fred

Note that

Print_Name (0006) = ‘error’

We want to print out a list of names that are in balance. First we find the customers in balance using the following formula:

List_in_balance = {all k in Key such that $proj_3(k) > 0$} so this would be the set {0001, 0003}

And then use Print_Name to get the actual names.

Print_Name ({0001, 0003}) = {Fred, Pete}

We can take a function that is defined on a set K say f: K → A and apply it to a *set* of elements of K to get a *set* of elements of A.

Thus we can define functions to manipulate the data in the databases, to insert data into the databases and to do many other things. These are part of the data specification.

Functions like Print_Name $proj_3$ are worth defining separately since we will use them in lots of other places.

The functions that are on the arrows of the X-machines may be more complicated – and they may depend on things like the current input and the current state of the memory (i.e. they may need to query a database or other set) so we would find that they are quite complicated and thus breaking them into a combination of simpler functions such as we have done here makes sense.

In Year 2 you will meet the special language used to deal with databases, SQL which is based on a generalization of the mathematical notion of a function – i.e. the relation.