Section 2.2: Relational Databases

Databases

A database is a set of records that can be manipulated by a computer. Database management systems allow users of the system to perform a variety of operations, such as retrieving records or adding, changing, or deleting records. Most commercial database products are written in the computing language SQL, which is based on the **relational database model**. In this section, we will look at **relational algebra**, which is the collection of fundamental operations used to manipulate databases.

The set of records in a database is essentially a relation. Up until now, we have been looking at binary relations – relations that have two columns when written as a table. However, for many practical applications, we are interested in relations with many more than two columns. For example,

STUDENT

number	name	course	grade
000123	John Smith	Math 161	C+
000124	Jane Doe	Math 161	B+
000125	Richard Jones	Comp 112	A-
000126	John Smith	Comp 112	D

This relation can also be expressed as a set of n-tuples: STUDENT = {(000123, John Smith, Math 161, C+),(000124, Jane Doe, Math 161, B+),(000125, Richard Jones, Comp 112, A-), (000126, John Smith, Comp 112, B+)}. However, the table is much easier to read.

The rows of a relation are called records. Each record is a unique combination of the elements. For example, each record of the STUDENT relation given above corresponds to one individual student at Camosun College. One database in a bank might have each banking transaction (withdrawal, deposit, transfer) as a record, and another database in which each bank account was a record.

The columns of a relation are called attributes. For example, the attributes of the STUDENT relation are the number, name, course, and grade.

A key is a single attribute or combination of attributes that uniquely identifies a record. For example, in the STUDENT relation above, it would not be a good idea to try to use the student name as a key, since two students have the same name. Rather, the college assigns each student a unique student number as a key. (In the table above, we could also use the grade as a key, since in this very short table, no two students have the same grade.)

Example

Is your drivers licence number a key for the records kept by the Motor Vehicle Branch?

Answer: Yes, because each individual is assigned a unique number.

Example

Is your postal code a key for the post office's database of all possible street addresses?

Answer: No, because there is more than one street address for each postal code.

The idea of a key ties in very nicely with the idea of a function that we discussed in previous sections. For a binary relation, the relation is a function if for each value of the first component, there is only one value of the second component. Therefore, the first component can serve as a key to identify the ordered pair.

Queries

A query is a request for information from a database. In order to answer a query, the database management system has to perform one or more fundamental operations on the database. We will be looking at the operations Select, Project, and Join.

Select

The Select operator selects certain records (rows of the table, or n-tuples) from a relation. Looking once again at the STUDENT relation, the operation

$$\sigma_{\text{course = “Math 161”}}(\text{STUDENT})$$

will select the n-tuples (000123, John Smith, Math 161, C+) and (000124, Jane Doe, Math 161, B+).

Example

What is the result of the operation $\sigma_{\text{grade = “A-”}}(\text{STUDENT})$?

Answer: (000125, Richard Jones, Comp 112, A-)

Project

The Project operator selects attributes, or columns, from a relation. For example, the projection $\pi_{\text{name}}(\text{STUDENT})$ gives (John Smith), (Jane Doe), (Richard Jones), (John Smith). More than one column may be projected, as in the following example.

Example

What is the result of the operation $\pi_{name,grade}$(STUDENT)?

Answer: (John Smith, C+), (Jane Doe, B+), (Richard Jones, A-), (John Smith, D).

When you use the project operator in order to choose certain rows of a relation, you may end up with identical n-tuples. For example, suppose you had the relation

INVENTORY

ID	Item	Brand Name	Colour	In Stock
1	hockey stick	Cooper	black	3
2	football	Trooper	pink	5
3	football	Trooper	yellow	0
4	football	Cooper	yellow	2
5	snowboard	Nike	white	0

and you used the command $\pi_{Item,Colour}$ (INVENTORY). You should then get the relation

Item	Colour
hockey stick	black
football	pink
football	yellow
football	yellow
snowboard	white

Note that there are two instances of the ordered pair (football, yellow). According to set theory, the duplicate should be removed and the table collapsed to only four rows.

Item	Colour
hockey stick	black
football	pink
football	yellow
snowboard	white

This is how the Project operator would be handled according to relational algebra. As a practical matter, what real databases often do is give you either option – either preserving all of the original rows intact, or collapsing duplicate rows. SQL, for example, uses the keyword `distinct` to collapse duplicate rows.

Join

The Join operator (also called Natural Join) joins two relations. Suppose we had a second relation, EMAIL

number	email
000123	johnny@yahoo.com
000124	jane@telus.net
000125	rick@badboy.ca
000126	c000126@camosun.bc.ca

We wish to match the "number" column in the STUDENT relation with the "number" column in the EMAIL relation, and so the operation

$$\text{STUDENT} \bowtie \text{EMAIL}$$

will give the resulting relation

number	name	course	grade	email
000123	John Smith	Math 161	C+	johnny@yahoo.com
000124	Jane Doe	Math 161	B+	jane@telus.net
000125	Richard Jones	Comp 112	A-	rick@badboy.ca
000126	John Smith	Comp 112	D	c000126@camosun.bc.ca

Note that the two relations joined by the $\bowtie$ operator must share one common attribute (in this example, each relation has the column "number").

Sequences of Operations

Suppose the query which we wish to make requires more than one fundamental operations. Our approach then is to make a series of operations that will yield the desired result. The first operation will result in a new relation, which we then perform another operation on. We could either give that temporary relation a name to be specified in the next operation, or the more compact way is to nest two operations.

For example, consider the following two relations.

INVENTORY

ID	Item	Brand Name	Colour	In Stock
1	hockey stick	Cooper	black	3
2	football	Trooper	pink	5
3	football	Trooper	yellow	0
4	football	Cooper	yellow	2
5	snowboard	Nike	white	0

CONTACT INFO

Brand Name	Phone
Cooper	555-1212
Trooper	370-4542
Nike	370-3001

Suppose we wished to list the ID and phone number for all items. We will first have to join the two databases using INVENTORY $\bowtie$ CONTACT INFO, giving

ID	Item	Brand Name	Colour	In Stock	Phone
1	hockey stick	Cooper	black	3	555-1212
2	football	Trooper	pink	5	370-4542
3	football	Trooper	yellow	0	370-4542
4	football	Cooper	yellow	2	555-1212
5	snowboard	Nike	white	0	370-3001

And then we'll want to use the Project operator to choose the ID and Phone columns.

The sequence of operations would then look like

$\pi_{\text{ID, Phone}}$ (INVENTORY $\bowtie$ CONTACT INFO)

giving the relation

ID	Phone
1	555-1212
2	370-4542
3	370-4542
4	555-1212
5	370-3001

which is our desired result.

Please note that for this example, if we tried to do the Project operator first, to get rid of the unnecessary columns, that we would have to leave the Brand Name column in for the Join operation. Then we'd have to strip it out using *another* Project operator. So, it's simpler to do the Join first.

Example

For the relation INVENTORY, write an operation or sequence of operations that will list the ID number for all out-of-stock items.

Answer:

π_{ID} ($\sigma_{\text{In Stock} = 0}$ (INVENTORY))