Relational Algebra Solutions

The following relational schema (in which primary keys are underlined) describes part of the operations of a bank. Note that each account may be owned by several customers and that each employee may be associated with several branches.

customer(ID, forename, surname, address, occupation)
account(accountno, type, balance, inBranch)   - type is "deposit", "current", ...
owner(accno, custID)
branch(branchNo, braddress)
employee(staffNo, forename, surname, empbranch, supervisor)

Notation:

∏(relation, attr1, ... attrn) means **project** the n attributes out of the relation.
σ(relation, condition) means **select** the rows which match the condition.
⊗(relation1,attr1,relation2,attr2) means **join** the two relations on the named attributes.
relation1 – relation2 is the **difference** between two relations.
relation1 ÷ relation2 is the **division** of one relation by another.

a) All the types of account currently in existence.

∏(account, type).

b) The account number, type and balance of all accounts at branch number 20.

∏(σ(account, inBranch = 20), accountno, type, balance)

c) The ID and surname of the owner(s) of account number 23519.

Owner23519 = σ(owner, accno = 23519)
Customer23519 = ⊗(Owner23519, custID, customer, ID)
∏(Customer23519, custID, surname)

Note common sequence: select to concentrate on specific data, join to connect to other data and the project to pick out the values required.

d) The account number and balance of any accounts owned by customers with the surname, "Lalmas".

Lalmas = σ(customer, surname = "Lalmas")
LalmasOwns = ⊗(Lalmas, ID, owner, custID)
LalmasAcs = ⊗(LalmasOwns, accno, account, accountno)
∏(LalmasAcs, accno, balance)

The owner relation is used to join the information from customer and account explicitly referred to.

e) The types of account for which there are no instances with a negative balance.

NegativeAccTypes = ∏(σ(account, balance < 0), type)
AllTypes = ∏(account, type)
AllTypes – NegativeAccTypes.

Typical use of difference - find all the types associated with a negative balance and then remove these from the set of all types.

f) The types of account for which there is at least one instance with a negative balance.

$\prod$(σ(account, balance < 0), type) (i.e. same as Negative AccTypes)

g) The employee numbers of any employees associated with every branch.

AllBranchNumbers = $\prod$(branch, branchNo)
EmployeeNumberBranch = $\prod$(employee, staffNo, empbranch)
EmployeeNumberBranch ÷ AllBranchNumbers

Uses relational division with the relations having the following columns:

EmployeeNumberBranch		AllBranchNumbers
staffNo	empbranch	branchNo

so we end up with a one column relation containing the staff numbers of those employees associated with every branch.

h) The branch number and address of any branches in which all accounts of type "deposit" have a positive balance.

depositAccs = σ(account, type = "deposit")
badDepAccs = σ(depositAccs, balance < 0)
GoodBranchNos = $\prod$(depositAccs, inBranch) –
$\prod$(badDepAccs, inBranch)
⊗(GoodBranchNos, inBranch, branch, BranchNo)

Typical use of double negative resulting from using the word "all" - we find first the set of negative deposit accounts. From this we extract the set of branch numbers and remove this from the set of all branch numbers. Then we join these back with the rest of the branch information to get the result.

i) The full details of any accounts owned by customers giving their occupation as "lecturer".

Lects = σ(customer, occupation = "Lecturer")
LectsAccs = ⊗(Lects, ID, owner, custID)
LectsAccs2 = ⊗(LectsAccs, accno, account, accountno)
$\prod$(LectsAccs2, accno, type, balance, inBranch)

j) The full details of any customers having accounts with balances over £1,000,000, where the account is at a branch employing someone with the same surname as the customer.

BigAccs = σ(account, Balance > 10000000)
Millionaires = ⊗(BigAccs, accountno , owner, accno)
Millionaires2 = ⊗(Millionaires, custID, customer, ID)
Millionaires3 = $\prod$(Millionaires2, surname, inBranch, custID)
BranchName = $\prod$(employee, surname, empbranch)
Crooks = ⊗(Millionaires3, (surname, inBranch),
BranchName(surname, empbranch)
$\prod$(Crooks, Customer.forename, Customer.surname, address,
occupation)

More complex - find the customers with big accounts and all surname, empbranch pairs. Now join these simultaneously on name and branch number. This will match up customers and employees as requested. Finally extract the customer information.Crooks can be defined in two steps (two joins).