University of Glasgow
Dip / MSc Information Technology
Information Systems and Databases

Tutorial Week 7 – Simple Relational Algebra and SQL

Richard Cooper

November 12th 2009

1.**The following is the schema of the bank account database:**

customer(ID, forename, surname, sex, address, occupation)
account(accountno, type, balance, dateOpened, *inBranch*)
owner(*accno*, *custID*)
branch(branchNo, braddress, *manager*)
employee(staffNo, forename, surname, *empbranch*, *supervisor*)

Give relational algebra programs and SQL queries to retrieve the following:

a) All the types of account currently in existence.

This is a simple projection of one of the columns from the Account table, i.e.:

RA π_{type} **(Account)**

```
SQL     SELECT DISTINCT type
        FROM Account;
```

Note that the relational algebra removes duplicates by definition, but you have to explicitly request this in SQL.

b) The account number, type and balance of all accounts at branch number 6.

Now you have to pick up the rows in Account for which the branch number is 6 and then project out the columns requested, i.e.:

RA $Branch6 \leftarrow \sigma_{inBranch = 6}$ **(Account)**
$Result \leftarrow \pi_{accountNo, type, balance}$ **(Branch6)**

```
SQL     SELECT accountNo, type, balance
        FROM Account
        WHERE inBranch = 6;
```

Selecting and then projecting is a very common pattern.

c) The ID and surname of the owner(s) of account number 23509.

Now having picked out one record from the Owner table, we must join the result with the Customer table toget to the surname required.

RA $Own23509 \leftarrow \sigma_{accno = 23509}$ **(Owner)**
$Cust23509 \leftarrow Own23509 \bowtie_{custid = id}$ **Customer**
$Result \leftarrow \pi_{id,surname}$ **(Cust23509)**

The pattern select-join-project is the dominant one for what I have called "simple" queries - i.e. ones in which you just have to connect up a set of tables and pick some rows and columns from the result

```
SQL     SELECT Customer.id, Customer.surname      using table & attribute name
        FROM Customer, Owner                      as tuple cursor to identify columns
        WHERE Owner.custID = Customer.id  AND Owner.accNo = 23509;

        SELECT C.id, C.surname                    using table alias as tuple cursor
        FROM Customer C, Owner O
        WHERE O.custID = C.id AND O.accNo = 23509;

        SELECT id, surname                        using an implicit tuple cursor for
        FROM Customer, Owner                      attributes as column names which
                                                  are unique
        WHERE custID = ID AND accNo = 23509;
```

There are three versions above, one for each of the three types of tuple cursor variable - all three work as there is no confusion about column names.

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

This is the same as the last one except that now you have to join three tables to get from the surname data to the account balance.

RA $GCust \leftarrow \sigma_{surname = 'Getty'}$ **(Customer)**
$GOwns \leftarrow GCust \bowtie_{id = custid}$ **Owner**
$GAccs \leftarrow GOwns \bowtie_{accno = accountno}$ **Account**
$Result \leftarrow \pi_{accno, balance}$ **(GAccs)**

```
SQL     SELECT Account.accountNo, Account.balance
        FROM Customer, Owner, Account
        WHERE Account.accountNo = Owner.accno
        AND Owner.custID = Customer.id  AND Customer.surname = 'Getty';
```

To avoid case problems with data values in Oracle use function UPPER or LOWER

```
Either      AND UPPER(Surname) = 'GETTY'
Or          AND LOWER(Surname) = 'getty'
```

Remember two things for cases like this:

You must include all table names in the FROM clause not just the ones explicitly mentioned in the question.

You must include the join condition without this you get everything printed out - possibly many times.

e) The full details of any accounts owned by customers giving their occupation as "turf accountant".

This is the same as the last one:

RA $TAs \leftarrow \sigma_{occupation = \text{“Turf Accountant”}}$ **(Customer)**
$TAOwn \leftarrow TAS \bowtie_{id=custId}$ **Owner**
$TAAcc \leftarrow TAOwn \bowtie_{accno = accountNo}$ **Account** ***(all account & customer info)***
$Result \leftarrow \pi_{accno, type, dateOpened, balance, inBranch}$ **(TAAcc)**

```
SQL  SELECT Account.*
     FROM Account, Owner, Customer
     WHERE accountNo = accNo        AND id = custID
                                    AND occupation = 'Turf Accountant';
```

f) The surname of each employee and his or her supervisor.

This is a bit trickier - in this case we must join a table with itself:

RA $EmpSup \leftarrow Employee \bowtie_{supervisor = staffno}$ **Employee**
$Result \leftarrow \pi_{esurname, ssurname}$ **(EmpSup)**

Actually, column renaming is more complicated - internally the system remembers which one comes from where.

```
SQL  SELECT E.surname AS 'Employee', S.surname AS 'Supervisor'
     FROM Employee E, Employee S
     WHERE E.supervisor = S. staffNo;
```

In SQL, it is now absolutely vital to define our own tuple cursor variables - E and S - to distinguish the two tables and which columns we mean when we use the column names.

g) 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.

If you want to try this out, insert records first so that data exists to be selected. Joins are usually done using foreign keys but here we are looking for matching surnames.

```
INSERT INTO Employee VALUES( 300,'Jane', 'Barr', 2, 217 );
INSERT INTO Employee VALUES( 301, 'Susan', 'Gibson', 3, 217 );
```

As usual, select using any constants in the question first, do some joins and then project out the columns required. The third join is the strange one. In the tutorial, we noted that using intersection would also work. This requires picking the surname and empBranch columns from MCusts and from Employee, then doing the intersection, but then you have to rejoin the result with Customer to get the information.

RA $MAccs \leftarrow \sigma_{balance > 1000000}$ **(Account)**
$MOwns \leftarrow MAccs \bowtie_{accountNo=accno}$ **Owner**
$MCusts \leftarrow Customer \bowtie_{id = custID}$ **MOwns** ***(all cust. acc info)***
$Crooks \leftarrow MCusts \bowtie_{surname, inBranch = surname, empBranch}$ **Employee**
$Answer \leftarrow \pi_{id, C.surname, C.Forename, sex, address, occupation}$ **(Crooks)**


```
SQL  SELECT Customer.*
     FROM Customer, Owner, Account, Employee
     WHERE accountNo = accNo
     AND id = custID
     AND Employee.surname = Customer.surname
     AND Employee.empBranch = Account.inBranch
     AND balance > 1000000;
```

Note the use of Customer.* to get all the columns from Customer and none of the others.