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

**Tutorial Week 8 – More SQL**

Richard Cooper

November 19th 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:

h) The types of account for which there is at least 1 instance with a negative balance.

**RA** **NegAccs ← σ balance < 0 (Account)** ***// all accounts with a negative balance***
**NegTypes ← π type (NegAccs)** ***// pick out types - duplicates removed***

**SQL**
```
SELECT DISTINCT Type          // In SQL we have to say remove
FROM Account                  // duplicates  with the DISTINCT keyword
WHERE Balance < 0;
```

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

***We can't just turn the "<" to a ">" as we get all the account types with both positive and negative balances, we must removed the previous answer from the set of account types:***

**RA** **NegAccs ← σ balance < 0 (Account)**
**NegTypes ← π type (NegAccs)** ***// i.e. the answer to (h)***
**AllTypes ← π type (Account)** ***// all the account types***
**Result ← AllTypes – NegTypes** ***// all the types not in answer (h)***

**SQL** ***Three alternatives are given here:***

**a)**
```
SELECT DISTINCT type          // select types which are not in the set of
FROM Account
        MINUS
SELECT DISTINCT type FROM Account  //  types with a negative balance
WHERE balance < 0;
```

**b)**
```
SELECT DISTINCT type          // select type
FROM Account
WHERE type NOT IN             // if not in
(SELECT type                  // the list of types with a –ve balance
FROM Account WHERE balance < 0);
```



**c)**
```
SELECT DISTINCT A1.type       // select type
FROM Account A1
WHERE NOT EXISTS              // if there is not an account
(SELECT *                     // with the same type and –ve balance
FROM ACCOUNT A2
WHERE A2.balance < 0  AND A1.type = A2.type);
```

i) The branch number and address of all branches except those which have deposit accounts with a negative balance.

***Find the set of branch numbers which have a deposit account with a negative balance and remove these from the set of all branches, before joining back to get the columns we need.***

**RA**
**NegDepAccs ← σ type= 'deposit' and balance < 0 (Account)**
**PosBrNum ← π branchNo (Branch) - π inBranch (NegDepAccs)**
**PosBrs ←= PosBrNums ⋈ inBranch = branchNo Branch)**
**Result ← π branchNo, brAddress (PosBrs)**

**SQL**
```
SELECT branchNo, brAddress FROM Branch
WHERE branchNo NOT IN
(  SELECT inBranch FROM ACCOUNT          // the branch numbers we
                                         //       don't want
        WHERE type = 'deposit' AND balance < 0 );
```

j) The types of account that occur at every branch.

***Clue - Question :  Where? Answer : Every***
***For the bank database,  the type 'current' is the only one to occur in every branch.***

**RA**
**AccBr ← π type, inBranch (Account)** ***pairs of type/branch***
**AllBr ← π branchNo (Branch)** ***list of all branches***
**Result ← AccBr ÷ AllBr** ***The divide gives any account types that occur in the paired list for every branch***

***SQL using double negative***
```
SELECT DISTINCT A1.type          Select account type
FROM Account A1
WHERE NOT EXISTS                 if there isn't any branch which
    (SELECT * FROM Branch              doesn't have this type
       WHERE branchNo NOT IN
            (SELECT inBranch     List of branches with this  type
                FROM Account A2
                    WHERE A1.type = A2.type)
    );

SELECT DISTINCT A1.type          Select account type
FROM Account A1
WHERE NOT EXISTS                 if there isn't any branch which
((SELECT DISTINCT branchNo FROM Branch)
    MINUS                              doesn't have this type
(SELECT inBranch  FROM Account A2 WHERE A2.Type = A1.Type));
```

***This query selects all the type/branch pairs that might exist and subtracts from them all the type/branch pairs that do exist, thus leaving all the type/branch pairs that do not exist. Then it removes the types that are in this list from the list of all types.***

***Suggestion : alter bank database so that there is one (or more) branches containing every type of account. Find them with SQL.***

k) The staff number and name of all the employees including, for managers, the branch number and address of the branch that they manage (one query).

*This needs an outer join to keep the employee data even those who are not managers:.*

RA EmpMans ← Employee ***left Outer Join*** $_{\text{staffno = manager}}$ Branch
Result ← π $_{\text{staffno, forename, surname, branchno, brAddress}}$ EmpMans

SQL SELECT staffno, forename, surname, branchno, braddress
FROM Employee LEFT OUTER JOIN Branch ON manager = staffno;

2. Write and run SQL queries to perform the following:

**a)** Find the full names and addresses of all customers who have accounts which have a negative balance (i) in name order, (ii) in overdraft order (largest first), (iii) in overdraft order (smallest first)

```
SELECT DISTINCT forename, surname, address, balance
        FROM Customer, Owner, Account
                WHERE id = custId AND accountNo = accno AND balance < 0
(i)  ORDER BY surname, forename;
(ii) ORDER BY balance;
(iii)ORDER BY balance DESC;
```

**b** Find how many current accounts owned by students have positive balances and how many have negative balances

```
SELECT COUNT(DISTINCT accountNo)
        FROM Account, Customer, Owner
                WHERE accno = accountNo AND id = custId AND type =
'current'
                        AND occupation = 'Student' AND balance < 0;
```

***Repeat for balance > 0***

***NB COUNT(*) counts the rows, so you can always use * unless you are counting distinct occurrences, when you need to specify DISTINCT and the column name. Otherwise you can specify any column name, all the rows will be counted.***

**c** For each branch, find the total number of accounts held at the branch and the total sum of money contained in those accounts

```
SELECT inBranch Branch, COUNT(*) NumAccs, SUM(balance) TotBal
        FROM Account
                GROUP BY inBranch;
```



**d** Find the average number of employees associated with each branch

***Use nested query to get numbers of staff for each branch. Rename this column.***

```
SELECT avg(numStaff)
        FROM
                (SELECT count(*) numStaff, empBranch
                        FROM Employee
                                GROUP BY empbranch);
```

***Or could use a View for the nested query***

**e** Find the average balance in accounts for each branch which has more than 3 accounts.

```
SELECT inBranch, AVG(balance)
        FROM Account
                GROUP BY inBranch
                        HAVING COUNT(*) > 3;
```

**f** Find the average balance in current accounts for each branch in which there is more than one current account.

```
SELECT inBranch, AVG(balance)
        FROM Account
                WHERE type = 'current'
                        GROUP BY inBranch
                                HAVING COUNT(*) > 1;
```

**g** Create a View which shows the account number, customer names, and type of account. Then use the view to find the names of customers with a deposit account.

```
CREATE VIEW AccNameType AS
        SELECT accountNo, type, forename, surname
                FROM Account, Customer, Owner
                        WHERE accountNo = accNo AND custID = id;
```

***Here is an example of using it:***

```
SELECT forename, surname FROM AccNameType
        WHERE type = 'deposit';
```

**h** Create a View which shows the staff number and name of each employee together with the staff number and name of their supervisor.. (see (g) )

```
CREATE VIEW Supervisees AS
SELECT E1.staffNo, E1.forename, E2.Surname,
  E2.StaffNo SupSNo, E2.forename SupFName, E2.surname SupSName
FROM Employee E1, Employee E2
WHERE E1.supervisor = E2.staffNo;
```

**i** Write queries b & c as parameterised queries.

```
SELECT accountNo, type, balance
FROM Account
WHERE inBranch = &Branch;

SELECT id, surname
FROM Customer, Owner
WHERE custID = id AND accNo = &Account_Number;
```

**j** Transfer £200 from the deposit account 23507 to the current account 23505

```
This code illustrates transactions..
Do a different update first
        UPDATE Account Set balance = balance + 100 WHERE accountNo =
23507;
        SELECT balance from Account Where accountno = 23507;

Start a transaction by committing previous updates
        COMMIT;
Alternatively, setting transaction
        SET TRANSACTION READ WRITE;

Update balance
        UPDATE ACCOUNT SET balance = balance - 200
                WHERE accountNo = 23507;
        UPDATE ACCOUNT SET balance = balance + 200
                WHERE accountNo = 23505;
        COMMIT;
```

**k** Transfer member of staff no 287 from branch 6 to branch 3

```
        UPDATE EMPLOYEE
                SET empBranch = 3
                        WHERE StaffNo = 287;
```

*What would happen if it was to branch 7?*

*Answer: (ORA-02291: integrity constraint (IT02_MONICA.FK_EBRANCH) violated - parent key not found)*

**l** Close account number 23524

*What if you try to delete the account?*

```
        DELETE FROM ACCOUNT WHERE accountNo = 23524;
```

*Doesn't work because Owner record exists for this account*

*Therefore Delete Owner record then Account record*

```
        DELETE FROM Owner WHERE accNo = 23524;
        DELETE FROM Account WHERE accountNo = 23524;
```

*Thus works fine*



*Alternatively replace foreign key constraint with delete cascade, try delete again*

```
        ALTER TABLE Owner DROP CONSTRAINT fk_OAcc;
        ALTER TABLE OWNER ADD CONSTRAINT fk_OAcc
        FOREIGN KEY (Accno) REFERENCES Account (accountNo) ON
DELETE CASCADE;
        DELETE FROM ACCOUNT WHERE accountNo = 23524;
```

*Now account and owner record are both deleted*

**m** Close branch number 6 and move all its staff to branch number 5

*Just don't try to delete Branch until related records are deleted*

```
        UPDATE EMPLOYEE  SET empBranch = 5  WHERE empBranch = 6;

        UPDATE ACCOUNT  SET inBranch = 5  WHERE inBranch = 6;

        DELETE FROM Branch  WHERE branchNo = 6;
```

**n** Remove all mention of customer 193 from the database

*Either delete related records first (i.e. from owner)*

```
        COMMIT;
        DELETE FROM OWNER WHERE custID = 193;
        DELETE FROM CUSTOMER WHERE id = 193;
        COMMIT;
```

*Or alter foreign key constraint to cascade on delete*

```
        ALTER TABLE OWNER DROP CONSTRAINT fk_OCust;
        ALTER TABLE Owner ADD CONSTRAINT fk_OCust
                FOREIGN KEY (custID) REFERENCES Customer (id)
                                        ON DELETE CASCADE;
        DELETE FROM CUSTOMER WHERE id = 193;
```

*Don't leave behind accounts with no owners*

**o** Find full details of any account for which there is no registered owner

*There aren't any such records, so I removed one first.*
*I used COMMIT : ROLLBACK so that the delete wasn't permanent - I was just using it to test my sql.*

```
        COMMIT;
        DELETE FROM Owner where accno = 23512;

        SELECT *
        FROM ACCOUNT
        WHERE NOT EXISTS
                (SELECT * FROM OWNER
                        WHERE accno = accountNo);
```