DB Worksheet 1: Primitive Relational Algebra Operators

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement

mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)
movement(no) $\overset{fk}{\Rightarrow}$ account(no)
account(sortcode) $\overset{fk}{\Rightarrow}$ branch(sortcode)

1. What is the result of the RA query $\pi_{\text{cname,sortcode}}\,\sigma_{\text{type='deposit'}}\,\text{account}$

cname	sortcode
McBrien, P	67
Poulovassilis, A	56

2. What is the result of the RA query

$$\pi_{\text{account.no,cname,mid}}\,\sigma_{\text{account.no=movement.no}\wedge\text{movement.amount}<0}(\text{account} \times \text{movement})$$

no	cname	mid
100	McBrien, P	1002
107	Poulovassilis, A	1004

3. Write an RA query to return the scheme (cname) that lists the cname of all customers that have made a withdrawal from an account.

$\pi_{\text{cname}}\,\sigma_{\text{account.no}=\text{movement.no}\wedge\text{amount}<0}(\text{account} \times \text{movement})$

4. Write an RA query to return the scheme (cname,bname) that lists the cname of all deposit account holders, together with bname where the account is held.

$\pi_{\text{cname,bname}}\,\sigma_{\text{branch.sortcode}=\text{account.sortcode}\wedge\text{type}=\text{'deposit'}}(\text{account} \times \text{branch})$

5. Write an RA query to return the scheme (sortcode) that lists the sortcodes of branches that either have less than £10,000 of cash, or that hold deposit accounts.

$\pi_{\text{sortcode}}\,\sigma_{\text{cash}<10{,}000}\,\text{branch} \cup \pi_{\text{sortcode}}\,\sigma_{\text{type}=\text{'deposit'}}\,\text{account}$

6. Write an RA query to return the scheme (sortcode) that lists those branches where no deposit account is held

$\pi_{\text{sortcode}}\,\text{branch} - \pi_{\text{sortcode}}\,\sigma_{\text{type}=\text{'deposit'}}\,\text{account}$

DB Worksheet 2: Derived Relational Algebra Operators

1. What is the result of the RA query $\pi_{\text{bname,cname}}(\text{branch} \bowtie \text{account})$

bname	cname
Strand	McBrien, P
Wimbledon	Poulovassilis, A
Wimbledon	Bailey, J
Goodge St	Boyd, M

2. What is the result of the RA query $\pi_{\text{bname,cname}}(\text{branch} \bowtie \text{account} \bowtie \text{movement})$

bname	cname
Strand	McBrien, P
Wimbledon	Poulovassilis, A
Goodge St	Boyd, M

3. What is the result of the RA query $\pi_{\text{sortcode,type}}\,\text{account} \div \pi_{\text{sortcode}}\,\text{branch}$

$\pi_{\text{sortcode, type}}$ account

67	current
67	deposit
34	deposit

π_{sortcode} branch

56
34
67

type
current

4. Write an RA query returning the scheme (bname,mid) that lists branch names, together with mids that have occurred at that branch.

$\pi_{\text{bname, mid}}$ (branch ⋈ account ⋈ movement)

5. Write an RA query returning the scheme (sortcode,bname,cash) that lists rows of all branches that have at least one deposit account.

branch ⋈ $\sigma_{\text{type='deposit'}}$ account

6. Write an RA query returning the scheme (sortcode) that lists the sortcodes of branches that have both have less than £10,000 of cash, and that hold deposit accounts.

① $\pi_{\text{sortcode}}\sigma_{\text{cash}<10000}$ branch ∩ $\pi_{\text{sortcode}}\sigma_{\text{type='deposit'}}$ account

② $\pi_{\text{sortcode}}\sigma_{\text{cash}<10000}$ (branch ⋈ $\sigma_{\text{type='deposit'}}$ account)

$\pi_{\text{sortcode, bname, cash}}$ (branch ⋈ account on branch.sortcode = account.sortcode ∧ type='deposit')

DB Worksheet 3: Equivalences Between RA Expressions

Consider the following queries over the bank_branch database.

1. Put a tick or cross to indicate if each equivalence holds:

Equivalence	Correct?
$\pi_{no,type}\,\sigma_{type='deposit'}\,account \equiv \sigma_{type='deposit'}\,\pi_{no,type}\,account$	✓
$\pi_{no,type}\,\sigma_{type='deposit'}\,account \equiv \sigma_{type='deposit'}\,\pi_{type,no}\,account$	✓
$\pi_{type}\,\sigma_{type='deposit'}\,account \equiv \sigma_{type='deposit'}(\pi_{no}\,account)$	✗
$\sigma_{sortcode=56}(account \times movement) \equiv \sigma_{sortcode=56}\,account \times movement$	✓
$\pi_{sortcode}(account \times movement) \equiv \pi_{sortcode}\,account \times movement$	✗
$\pi_{no,type}\,\sigma_{type='deposit'}\,account \equiv \pi_{no,type}\,\sigma_{type<>'current'}\,account$	✗

2. Simplify the following RA query to contain as few RA operators as possible

$\pi_{no,type}\,\sigma_{sortcode=56}\,\pi_{no,type,sortcode}\,\sigma_{type='deposit'}\,account$

$\Pi_{no,type}\,\sigma_{sortcode=56 \wedge type='deposit'}\,account$

3. Rewrite the following RA query into the form $\pi_{...}\,\sigma_{...}\,R \times S$, the general form of which we will call a **project select product (PSP)** query.

$\sigma_{account.no=movement.no}(\pi_{no,cname}\,account \times \pi_{mid,no}\,\sigma_{amount>1000}\,movement)$

$\Pi_{no,cname,mid}\,\sigma_{account.no=movement.no \wedge amount>1000}(account \times movement)$

4. Rewrite the following RA to be a union between two PSP queries

$\sigma_{account.no=movement.no}(\pi_{no,cname,rate}\,account \times (\pi_{mid,no}\,\sigma_{amount>1000}\,movement \cup \pi_{mid,no}\,\sigma_{amount<100}\,movement))$

$\Pi_{no,cname,rate,mid}\,\sigma_{account.no=movement.no \wedge amount>1000}(account \times movement) \cup \Pi_{"}\,\sigma_{"\wedge amount<100}("")$

5. Rewrite the following query to an equivalent form that minimises the number of tuples, and the number of attributes within those tuples, that are handled by the $\times$ operator.

$\pi_{no,cname,tdate}\,\sigma_{amount<0 \wedge account.no=movement.no}(account \times movement)$

$\sigma_{account.no=movement.no}(\Pi_{no,cname}\,account \times \Pi_{no,tdate}\,\sigma_{amount<0}\,movement)$

DB Worksheet 4: Datalog

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement

mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)
movement(no) $\overset{fk}{\Rightarrow}$ account(no)
account(sortcode) $\overset{fk}{\Rightarrow}$ branch(sortcode)

1. Write a Datalog query returning the scheme (cname,bname) listing the cname of all deposit account holders, together with bname where the account is held.

```
deposit_holders(CName, BName) :-
    account(_, 'deposit', CName, _, SortCode),
    branch(SortCode, BName).
```

2. Write a Datalog query returning the scheme (no,bname) listing the account number and branch name of all accounts that have no movements recorded for the account.

```
no_movements(No, BName) :-
    account(No, _, _, _, SortCode),
    branch(SortCode, BName, _),
    ¬movement(_, No, _, _).
```

3. Write a Datalog query that returns the scheme (no) listing the 'target account' numbers, defined as those accounts that have made a withdrawal, or are of type deposit.

```
target_accts(No) :-
    account(No, _, _, _, _),
    movement(_, No, Amnt, _),
    amnt < 0.
target_accts(No) :-
    account(No, 'deposit', _, _, _).
```

4. List what the following Datalog query returns, and explain its semantics.

```
query(CName) :-
        account(_, _, , CName, _, _),
        ¬sub_query(CName).
sub_query(CName) :-
        account(_, _, CName, _, _),
        account(_, Type, _, _, _),
        ¬account(_, Type, CName, _, _).
```

Semantics: List customers with every type of account found in the database.

CName
McBrien, P
Poulovassilis, A

DB Worksheet 5: Translating Between RA and SQL

Consider the following fragment of the **bank_branch** database:

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

movement(no) $\overset{fk}{\Rightarrow}$ account(no)

movement

mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

1. Write the RA expression using project, select and product operators equivalent to the SQL query below.

```
SELECT   account.cname, movement.amount
FROM     account JOIN movement ON account.no=movement.no
WHERE    account.rate>2.0
AND      movement.amount>1000
```

$\pi_{cname, amount}\ \sigma_{account.no = movement.no \wedge rate>2.0 \wedge amount>1000}\ (account \times movement)$

2. Modify your RA expression to use any other RA operators you have been shown to write a mimimal RA expression (*i.e.* one that uses as few operators as possible).

$\pi_{cname, amount}\ \sigma_{rate>2.0 \wedge amount>1000}\ (account \bowtie movement)$

3. Write a minimal SQL query (*i.e.* as compact as posible) equivalent to the RA expression π_{no} movement $- \pi_{no}$ account

```
SELECT no
FROM   movement
EXCEPT
SELECT no
FROM   account
```

4. Write a minimal SQL query equivalent to the RA expression $\pi_{no,type}$ account

```
SELECT   no, type
FROM     account
```

5. Write a minimal SQL query equivalent to the RA expression π_{type} account

```
SELECT DISTINCT type
FROM   account
```

6. Write a minimal SQL query equivalent to the RA expression $\pi_{type,cname}$ account

```
SELECT DISTINCT type, cname
FROM   account
```

7. Write a minimal SQL query equivalent to the RA expresss account $\ltimes$ movement

```
SELECT   account.*
FROM     account NATURAL JOIN movement
```

DB Worksheet 6: SQL Set Operators

The following questions again use the **bank_branch** database:

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement

mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)
movement(no) $\stackrel{fk}{\Rightarrow}$ account(no)
account(sortcode) $\stackrel{fk}{\Rightarrow}$ branch(sortcode)

1. Write an SQL query returning the scheme (mid,no,amount,tdate) listing all details of movements for accounts 100,101,103 and 107.

```
SELECT *
FROM movements
WHERE no IN (100, 101, 103, 107)
```

2. Write an SQL query returning the scheme (sortcode) listing the sortcode of all branches without any deposit accounts.

```
SELECT sortcode
FROM branch
WHERE sortcode NOT IN (SELECT sortcode
                       FROM account
                       WHERE type = 'deposit')
```

3. Write an SQL query without using any negation (*i.e.* without the use of NOT or EXCEPT) returning the scheme (no) listing accounts with no movements on or before the 11-Jan-1999.

4. Write an SQL query returning the scheme (cname) listing customers that have every type of account that appears in account.

DB Worksheet 7: Null Values in SQL

In a modified version of the **bank_branch** database, called **bank_branch_null**, there are the following two tables:

movement

mid	no	amount	tdate
0999	119	45.00	null
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999
1010	100	null	20/1/1999
1011	null	null	20/1/1999
1012	null	600.00	20/1/1999
1013	null	-46.00	20/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	null	67
101	'deposit'	'McBrien, P.'	5.25	67
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	null	56

1. Write an SQL query returning the scheme (mid) to find movements known not to have occurred on 5/1/1999

```
SELECT  mid
FROM    movement
WHERE   NOT tdate = '5/1/1999'
```

2. Write an SQL query returning the scheme (mid) to find movements that have or might have occurred on 5/1/1999.

```
SELECT  mid
FROM    movement
WHERE   (tdate = '5/1/1999') IS NOT FALSE
```

3. Write an SQL query returning the scheme (no,mid) that lists account numbers, and any movements mids that have or might have occurred on that account

```
SELECT  account.no,
        mid
FROM    account JOIN movement
        ON (account.no = movement.no) IS NOT FALSE
```

4. What is the result of the following query:

```
SELECT  account.no
FROM    account
WHERE   account.no NOT IN (SELECT movement.no FROM movement)
```

no ✓

no
125

5. Write an SQL query returning the scheme (no) that lists those account numbers that might not have any movements.

```
SELECT no
FROM   account
WHERE  no NOT IN (SELECT no
                  FROM   movement
                  WHERE  no IS NOT NULL)
```

DB Worksheet 8: Left, Right, Outer and Inner Joins

The following question uses the **bank_branch_null** database:

movement

mid	no	amount	tdate
0999	119	45.00	null
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999
1010	100	null	20/1/1999
1011	null	null	20/1/1999
1012	null	600.00	20/1/1999
1013	null	-46.00	20/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	null	67
101	'deposit'	'McBrien, P.'	5.25	67
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	null	56

1. What is the result of

```
SELECT account.no, movement.mid
FROM   account NATURAL LEFT JOIN movement
```

no	mid
100	1000
100	1002
100	1006
100	1010

no	mid
101	1001
101	1008
119	0999
119	1009

no	mid
125	null

2. What is the result of

```
SELECT account.no, movement.mid
FROM   account NATURAL FULL OUTER JOIN movement
```

Answer to Q1 +

no	mid
null	1004
null	1005
null	1011
null	1012
null	1013

3. Write an SQL query returning the scheme (no,cname,mid) that lists all account numbers in the database (including just those in movement), and lists any known mid or cname for each account. The result for the current data should be:

no	cname	mid
100	McBrien, P.	1000
100	McBrien, P.	1006
100	McBrien, P.	1010
100	McBrien, P.	1002
101	McBrien, P.	1008
101	McBrien, P.	1001
103	null	1005
107	null	1004
119	Poulovassilis, A.	999
119	Poulovassilis, A.	1009
125	Bailey, J.	null

```
SELECT COALESCE(account.no, movement.no) AS no,
       account.cname,
       movement.mid
FROM   account NATURAL FULL OUTER JOIN
       movement
WHERE  COALESCE(account.no, movement.no)
       IS NOT NULL
```

π σ — (table x table)

```
SELECT no
       account.cname,
       movement.mid
FROM   account NATURAL FULL OUTER JOIN
                                movement
WHERE  no IS NOT NULL
```

DB Worksheet 9: OLAP Queries in SQL

The following questions should be written to run on the **bank_branch** database listed below.

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

movement

mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1001	101	4000.00	5/1/1999
1002	100	-223.45	8/1/1999
1004	107	-100.00	11/1/1999
1005	103	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999

account

no	type	cname	rate?	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
103	'current'	'Boyd, M.'	NULL	34
107	'current'	'Poulovassilis, A.'	NULL	56
119	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

key branch(sortcode)
key branch(bname)
key movement(mid)
key account(no)
movement(no) $\overset{fk}{\Rightarrow}$ account(no)
account(sortcode) $\overset{fk}{\Rightarrow}$ branch(sortcode)

1. Write an SQL query returning the scheme (no,balance,avg_trans) that lists for each account that has transactions the account no, the balance (computed as the sum of the movements for the account), and the average value of transactions on that account.

```
SELECT   no
         SUM(amount) AS balance,
         AVG(amount) AS avg_trans
FROM     movement
GROUP BY no
```

2. Alter the query for (1) so that it includes the customer name of each account, and includes *all* accounts held at the bank in the listing of balances, even if the account has no movements.

```
SELECT   account.no,
         account.cname,
   COALESCE(SUM(movement.amount), 0) AS balance,
         AVG(movement.amount) AS avg_trans
FROM     account LEFT JOIN movement ON account.no = movement.no
GROUP BY account.no,
         account.cname
```

3. Write an SQL query returning the scheme (cname,current_balance,deposit_balance) that lists one row for each customer (*i.e.* each distinct cname), with a column for the net balance of all current accounts held by the customer, and a column for the net balance of all deposit accounts held by the customer.

4. Write an SQL query returning the scheme (no,cname,type,pc_cust_funds,pc_type_funds) that lists one row for each account, and for each account, lists the no, cname and type of the account, and in pc_cust_funds the percentage of the customer funds held in the account, and in pc_type_funds the percentage of the total funds in this particular type of account. For the current data this should result in:

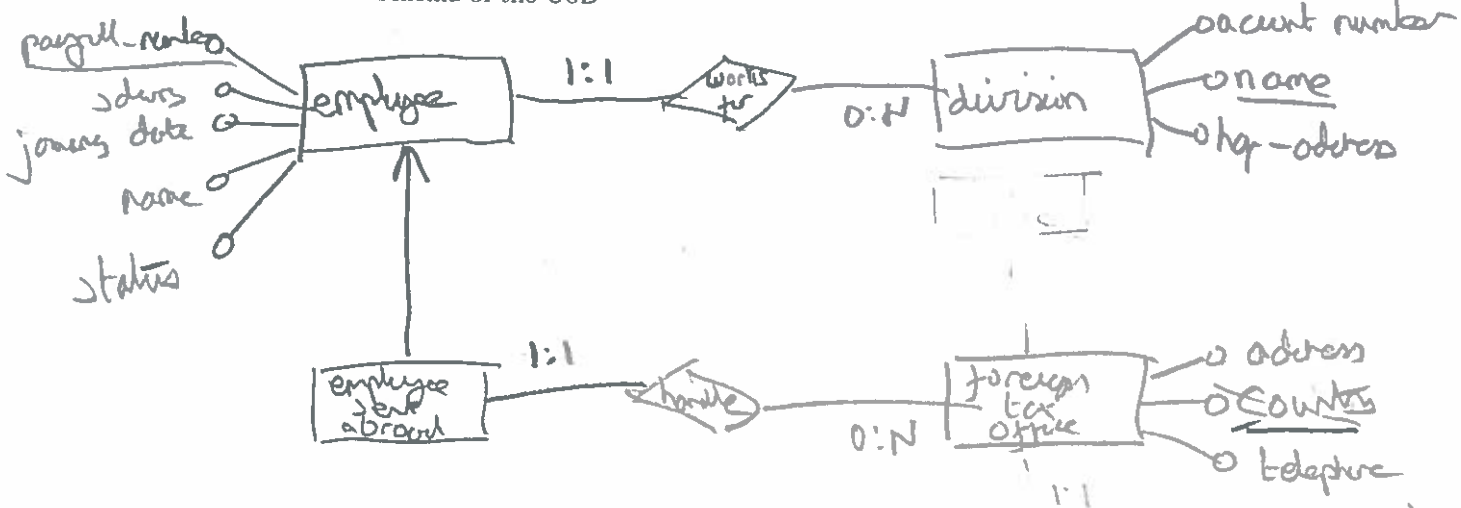
no	cname	type	pc_cust_funds	pc_type_funds
100	McBrien, P.	current	28.52	84.22
101	McBrien, P.	deposit	71.48	48.29
103	Boyd, M.	current	100.00	5.87
107	Poulovassilis, A.	current	4.20	9.91
119	Poulovassilis, A.	deposit	95.80	51.71
125	Bailey, J.	current	NULL	0.00

DB Worksheet 10: Constructing an ER^{KLMOS} Schema

The following text gives a description of a UoD, which you have been asked to build a relational database that stores data associated with the UoD.

> The payroll system for *BIG Inc* records the salaries, status, joining date, name, and payroll number for all of the corporations 30,000 employees. Each employee works for one division, and each division has an account number for paying its staff. We identify divisions by their name, and record the address where the division's HQ is located.
>
> For employees sent abroad by *BIG Inc*, we record the address, country and telephone number of the foreign tax office that will handle the employee. It is assumed that each country has one central tax office that we have to deal with. All other employees have their tax affairs dealt with by the Inland Revenue.

Draw an ER^{KLMOS} schema of the UoD



employee(payroll_number, salary, joining_date, name, status, dname)

employee(dname) ⊆ division(name)

division(account_number, name, hq_address)

employee_sent_abroad(country, payroll_number)

employee_sent_abroad(country) ⊆ foreign_tax_office(country)

employee_sent_abroad(payroll_number) ⊆ employee(payroll_number)

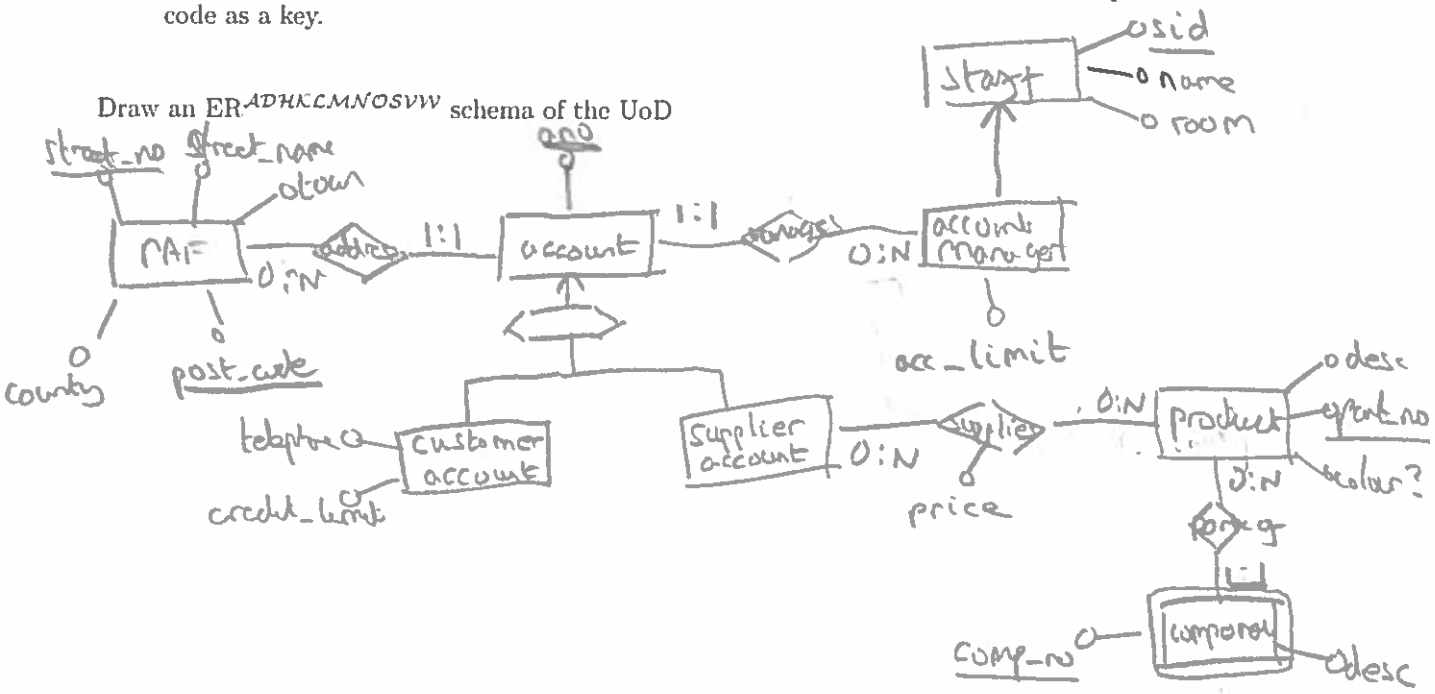
foreign_tax_office(address, country, telephone)

DB Worksheet 11: Constructing an ER^{ADHKLMNOSVW} Schema

The following text gives a description of a UoD, which you have been asked to build a relational database that stores data associated with the UoD.

> The customer and supplier database of *Big Inc* will hold all accounts of the company, divided into customer accounts and supplier accounts. All accounts have an account number, and one account manager assigned from the company's staff. *Big Inc* identifies staff by a sid, and records the staff member's name and room. The account managers have a limit on the number of accounts they can manage. Only certain staff members are permitted to be account managers.
>
> For customer accounts we need to record a credit limit on the balance of the account, and the telephone number of the accounts department at the customer.
>
> For supplier accounts we need to record which *Big Inc* products are supplied, and at what price.
>
> *Big Inc* products are identified by the company standard part_no and all have a description. For some we record the colour. Some products have a record of the components, each component identified by a combination of part_no and component number, and again each has a description. Some products do not have a supplier.
>
> *Big Inc* has purchased a copy of the Post Office address file, and associates every account to an address from this file. The address data includes street number, street name, town, county and post code, and uses a combination of street number and post code as a key.

Draw an ER^{ADHKLMNOSVW} schema of the UoD



PAF (street_no, post_code, street_name, town, county)

staff (sid, name, room)

account_manager (sid, acc_limit)

account_manager (sid) ⇒ staff (sid)

account (ano, street_no, post_code, sid)

account (street_no, post_code) ⇒ PAF (street_no, post_code)

account (sid) ⇒ account_manager (sid)

customer_account (sid, telephone, credit_limit)

customer_account (sid) ⇒ account (sid)

supplier_account (sid)

supplier_account (sid) ⇒ account (sid)

product (part_no, desc, colour)

supplies (sid, part_no, price)

supplies (sid) ⇒ supplier_account (sid)

supplies (part_no) ⇒ product (part_no)

component (part_no, comp_no, desc)

component (part_no) ⇒ product (part_no)

DB Worksheet 12: Minimal Cover of FDs and Candidate Keys

Suppose a relation $R(A, B, C, D, E, F, G, H)$ has the FDs

$S = \{AB \to DEH, BEF \to A, FGH \to C, D \to EG, EG \to BF, F \to BH\}$

1. Rewrite S to an equivalent set of FDs which only have a single attribute on the RHS of each FD.

   $AB \to D$, $AB \to E$, $AB \to H$, $BEF \to A$, $FGH \to C$, $D \to E$, $D \to G$
   $EG \to B$, ~~$AB \to B$~~, $EG \to F$, $F \to B$, $F \to H$

2. Consider each FD $X \to A$, and for each $B \in X$, consider if $X \to B$ from the other FDs. If so, replace $X \to A$ by $(X - B) \to A$ in S.

   Since $F \to B$
   $BEF \to A \Rightarrow EF \to A$
   Since $F \to H$
   $FGH \to C \Rightarrow FG \to C$

3. Consider each FD $X \to A$, and compute X^+ without using $X \to A$. If $A \subseteq X^+$, delete $X \to A$ since it is rundundant. This will give a minimal cover S_c of S.

(1) $AB^+ = ABCDEFGH$
Try removing $AB \to D$: $AB^+ = ABEH$ so can't remove
Try removing $AB \to E$: $AB^+ = ABDHEGFC$ ∴ remove $AB \to E$
Try removing $AB \to H$: $AB^+ = ABDEGFHC$ ∴ remove $AB \to H$

$EF^+ =$

(2) Since $EG \to F, F \to B \models EG \to B$
$EG \to B \Rightarrow \emptyset$
Since $AB \to D, D \to E \models AB \to E$
$AB \to E \Rightarrow \emptyset$
Since $AB \to D, D \to EG, EG \to F, F \to H \models AB \to H$
$AB \to H \Rightarrow \emptyset$

$S_c = \{AB \to D, EF \to A, FG \to C, D \to EG, EG \to F, F \to BH\}$

4. Justify what are the minimal candidate keys of R constrained by S_c

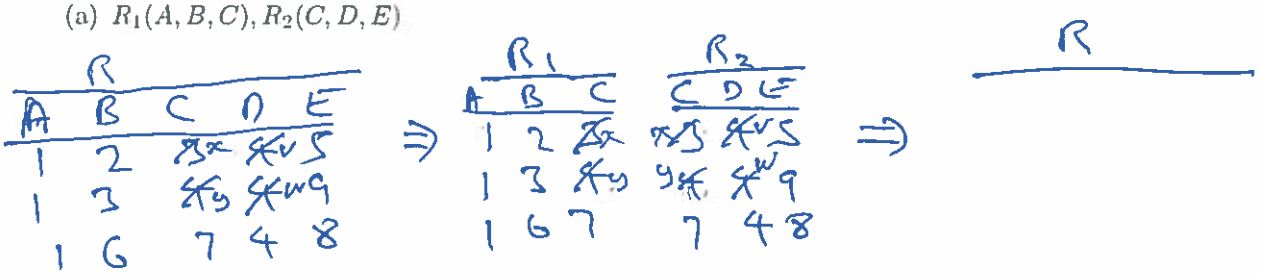
Since $D^+ = DEGFBHAC$ — D is minimal key
Since $F^+ = FBH$ — F is not a key
Since D is a key, and $AB \to D$, AB is a minimal key
Since $EF \to A, F \to B$ ∴ EF is a minimal key
Since $EG \to F$, and EF is key, so is EG a minimal key

{Since $F \to B$, AB is a key ∴ AF is a minimal key

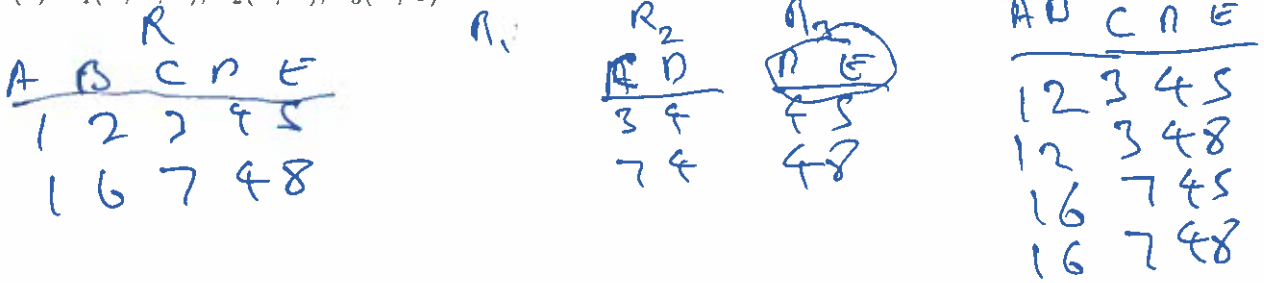
DB Worksheet 13: Lossless Decomposition of Relations

1. Suppose relation $R(A, B, C, D, E)$ has the FDs $S = \{AB \to C, C \to DE, E \to A\}$. For each of the following decompositions, determine if the decomposition if lossless, and if not lossless, illustrate a dataset for R that fails to decompose in a lossless manner over the relations.

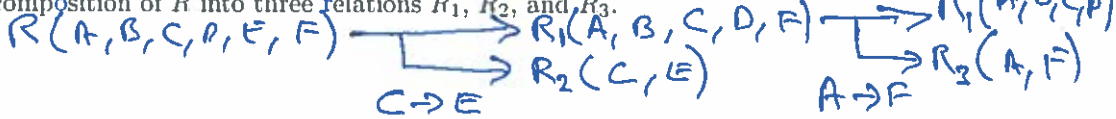
   (a) $R_1(A, B, C), R_2(C, D, E)$



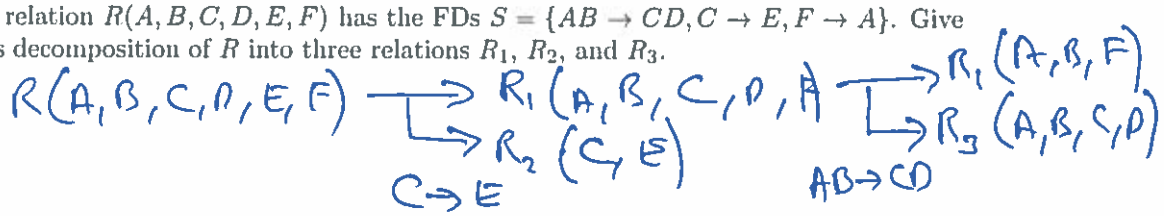
   (b) $R_1(A, B, C), R_2(C, D), R_3(D, E)$



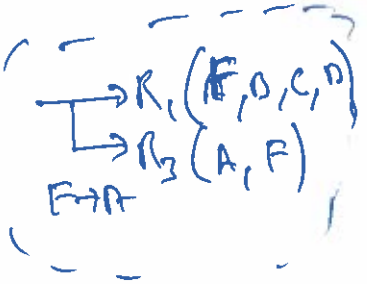
2. Suppose relation $R(A, B, C, D, E, F)$ has the FDs $S = \{AB \to CD, C \to E, A \to F\}$. Give a lossless decomposition of R into three relations R_1, R_2, and R_3.



3. Suppose relation $R(A, B, C, D, E, F)$ has the FDs $S = \{AB \to CD, C \to E, F \to A\}$. Give a lossless decomposition of R into three relations R_1, R_2, and R_3.



not a good answer, since it does not preserve FDs



DB Worksheet 14: Normal Forms

Suppose a relation $R(A, B, C, D, E, F, G, H)$ has the FDs

$S_c = \{AB \to D, EF \to A, FG \to C, D \to EG, EG \to F, F \to BH\}$

1. Decompose the relation into 3NF

Candidate keys are D, AB, AF, EF, EG ∴ CH non-prime

Since $F \to H$ has a non super-key determ a nonprime attribute

Decompose out $R_2(F, H)$, leaving $R'(A, B, C, D, E, F, G)$

$FG \to C$ violates 3NF

Decompose out $R_3(F, G, C)$ leaving $R_1(A, B, D, E, F, G)$

R_1, R_2, R_3 is 3NF

2. Decompose the relation into BCNF

$F \to B$ breaks BCNF in R_1

Decompose out $R_4(F, B)$ leaving $R_1'(A, D, E, F, G)$

Combine R_2 & R_4 to get $R_5(F, B, H)$. ∴ BCNF is R_5, R_3, R_1'

3. Determine if your decompositions in (1) and (2) preserve FDs, and if they do not, suggest how to amend you schema to preserve FDs.

3NF preserves all FDs

BCNF schema loses $AB \to D$ so add $R_6(A, B, D)$ to preserve FD

DB Worksheet 15: Anomalies in Transactions

```
BEGIN TRANSACTION rental_charge
    UPDATE directory
    SET charge=charge+17
COMMIT TRANSACTION rental_charge

BEGIN TRANSACTION transfer_charge
    UPDATE directory
    FROM charge=charge+100
    WHERE telephone=1000

    UPDATE directory
    SET charge=charge-100
    WHERE telephone=1002
COMMIT TRANSACTION transfer_charge
```

```
BEGIN TRANSACTION total_charge
    SELECT SUM(charge)
    FROM directory
COMMIT TRANSACTION total_charge
```

directory

telephone	name	charge
1000	Adams	10.00
1001	Jones	120.25
1002	Black	344.00

`rental_charge` $H_1 = r_1[d_{1000}], w_1[d_{1000}], r_1[d_{1001}], w_1[d_{1001}], r_1[d_{1002}], w_1[d_{1002}]$

`transfer_charge` $H_2 = r_2[d_{1000}], w_2[d_{1000}], r_2[d_{1002}], w_2[d_{1002}]$

`total_charge` $H_3 = r_3[d_{1000}], r_3[d_{1001}], r_3[d_{1002}]$

For each of the following histories, identify if they are a concurrent execution of some pair of the above histories, and if so, then determine if any of the following three anomalies has occurred:

A lost update

B inconsistent analysis

C dirty read

1. $r_3[d_{1000}], r_1[d_{1000}], w_1[d_{1000}], r_1[d_{1001}], w_1[d_{1001}], r_1[d_{1002}], w_1[d_{1002}], c_1, r_3[d_{1001}], r_3[d_{1002}], c_3$

   inconsistent analysis

2. $r_1[d_{1000}], r_1[d_{1001}], r_1[d_{1002}], r_2[d_{1000}], w_2[d_{1000}], r_2[d_{1002}], w_2[d_{1002}], w_1[d_{1000}], w_1[d_{1001}], w_1[d_{1002}], c_1, c_2$

   not a concurrent execution of H_1 and H_2 because operations of H_1 reordered.

3. $r_1[d_{1000}], r_2[d_{1000}], w_1[d_{1000}], r_1[d_{1001}], w_1[d_{1001}], r_1[d_{1002}], w_2[d_{1000}], r_2[d_{1002}], w_1[d_{1002}], w_2[d_{1002}], c_1, c_2$

   $w_1[d_{1000}]$ between $r_2[d_{1000}]$ & $w_2[d_{1000}]$ implies lost update of $w_1[d_{1000}]$

4. $r_3[d_{1000}], r_3[d_{1001}], r_2[d_{1000}], w_2[d_{1000}], r_3[d_{1002}], r_2[d_{1002}], w_2[d_{1002}], c_2, a_3$

   no anomalies (and is correct execution)

5. $r_2[d_{1000}], w_2[d_{1000}], r_1[d_{1000}], r_2[d_{1002}], w_2[d_{1002}], a_2, w_1[d_{1000}], r_1[d_{1001}], w_1[d_{1001}], r_1[d_{1002}], w_1[d_{1002}], c_1$

   dirty read of d_{1000} by H_1

DB Worksheet 16: Serialisability

$H_1 = r_1[o_1], w_1[o_1], w_1[o_2], w_1[o_3], c_1$

$H_2 = r_2[o_2], w_2[o_2], w_2[o_1], c_2$

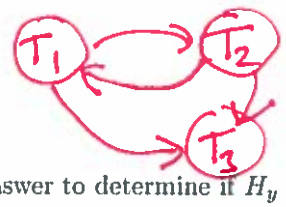
$H_3 = r_3[o_1], w_3[o_1], w_3[o_2], c_3$

$H_x = r_1[o_1], w_1[o_1], r_2[o_2], w_2[o_2], w_2[o_1], c_2, w_1[o_2], r_3[o_1], w_3[o_1], w_3[o_2], c_3, w_1[o_3], c_1$

$H_y = r_3[o_1], w_3[o_1], r_1[o_1], w_1[o_1], w_3[o_2], c_3, w_1[o_2], r_2[o_2], w_2[o_2], w_2[o_1], c_2, w_1[o_3], c_1$

1. Determine what are the ordered conflicting pairs in H_x, and write them down in the form $rw_i[o] \rightarrow rw_j[o]$ (where the rw should be replaced by a r or w in your answer as appropriate). Use your answer to determine if H_x is CSR.

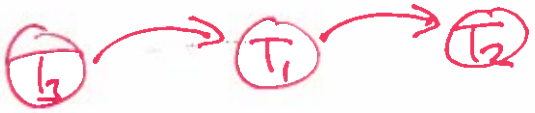
$w_1[o_1] \rightarrow w_2[o_1]$
$w_2[o_2] \rightarrow w_1[o_2]$
$w_1[o_2] \rightarrow w_3[o_2]$
$w_2[o_1] \rightarrow r_3[o_1]$



cycle in SG
∴ not serializable

2. Determine the conflicting pairs in H_y, and use your answer to determine if H_y is CSR.

$w_3[o_1] \rightarrow r_1[o_1]$
$w_1[o_1] \rightarrow w_2[o_1]$
$w_3[o_2] \rightarrow w_1[o_2]$
$w_1[o_2] \rightarrow r_2[o_2]$



acyclic SG ∴ serializable

DB Worksheet 17: Recoverability

$H_w = r_2[o_1], r_2[o_2], w_2[o_2], r_1[o_2], w_2[o_1], r_2[o_3], c_2, c_1$

$H_x = r_2[o_1], r_2[o_2], w_2[o_1], w_2[o_2], w_1[o_1], w_1[o_2], c_1, r_2[o_3], c_2$

$H_y = r_2[o_1], r_2[o_2], w_2[o_2], r_1[o_2], w_2[o_1], c_1, r_2[o_3], c_2$

$H_z = r_2[o_1], w_1[o_1], r_2[o_2], w_2[o_2], r_2[o_3], c_2, r_1[o_2], w_1[o_2], w_1[o_3], c_1$

1. Fill in for each history the list of pairs of operations where one transaction has read from another transaction ($w_i[o] \rightarrow r_j[o]$), and where one transaction has overwritten another transaction ($w_i[o] \rightarrow w_j[o]$).

H_w	H_x	H_y	H_z
$w_2[o_2] \rightarrow r_1[o_2]$	$w_2[o_1] \rightarrow w_1[o_1]$ $w_2[o_2] \rightarrow w_1[o_2]$	$w_2[o_2] \rightarrow r_1[o_2]$	$w_2[o_2] \rightarrow r_1[o_2]$

2. Determine if H_w is RC, ACA, or ST.

Dirty read $r_1[o_2]$ means not ACA (nor ST). However RC, because c_2 before c_1

3. Determine if H_x is RC, ACA, or ST.

No Dirty reads means ACA (also RC). Dirty writes mean not ST

4. Determine if H_y is RC, ACA, or ST.

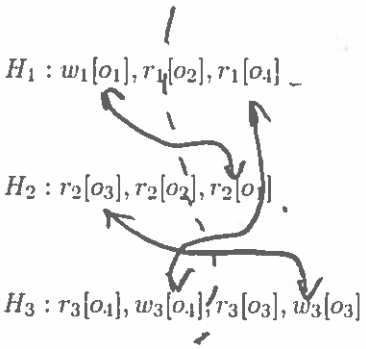
Dirty read $r_1[o_2]$ is committed whilst still dirty $\therefore$ not RC (nor ACA/ST)

5. Determine if H_z is RC, ACA, or ST.

No dirty reads in H_z nor dirty writes $\therefore$ ST (and ACA/RC)

DB Worksheet 18: Deadlocks and Waits-For Graphs

You are told that three transactions H_1, H_2, H_3 perform the following sequence of operations:



1. Write down all the possible conflict pairs from H_1, H_2, H_3 in the form $rw_i \rightarrow rw_j$.

$w_1[o_1] \rightarrow r_2[o_1]$ OR $r_2[o_1] \rightarrow w_1[o_1]$

$r_2[o_3] \rightarrow w_3[o_3]$ OR $w_3[o_3] \rightarrow r_2[o_3]$

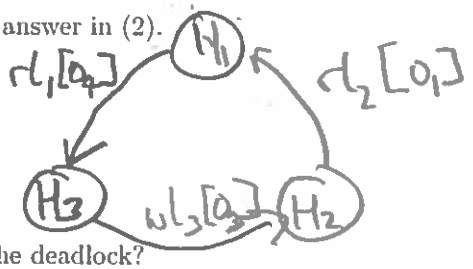
$w_3[o_4] \rightarrow r_1[o_4]$ OR $r_1[o_4] \rightarrow w_3[o_4]$

2. Write a concurrent execution of H_1, H_2, H_3 which results in a deadlock involving all three transactions. [Hint, use the conflict pairs to determine which transaction might have to wait for another, and then work out possible cycles of such waits-for].

$w_1[o_1], r_1[o_2], r_2[o_3], r_2[o_2],$

$r_3[o_4], w_3[o_4], r_3[o_3],$ <deadlock>

3. Draw the WFG for your answer in (2).



4. How would you resolve the deadlock?

Rollback anything

H_2 is best, because it has performed no work

DB Worksheet 19: Cache Consistent Checkpoint

The log belong is kept by a DM that is using cache consistent checkpointing and where the scheduler is using strict executions.

LOG	b_7
UNDO	$w_7[b_{67}, \text{cash}=34005.25]$
REDO	$w_7[b_{67}, \text{cash}=37005.25]$
LOG	b_2
UNDO	$w_2[b_{34}, \text{cash}=10900.67]$
REDO	$w_2[b_{34}, \text{cash}=8900.67]$
LOG	b_6
UNDO	$w_6[a_{101}, \text{rate}=5.25]$
REDO	$w_6[a_{101}, \text{rate}=6.00]$
LOG	b_1
UNDO	$w_1[b_{56}, \text{cash}=94340.45]$
REDO	$w_1[b_{56}, \text{cash}=84340.45]$
CP	$\{1, 2, 6\}$
UNDO	$w_6[a_{119}, \text{rate}=5.50]$
REDO	$w_6[a_{119}, \text{rate}=6.00]$
LOG	c_6
UNDO	$w_2[b_{67}, \text{cash}=34005.00]$
REDO	$w_2[b_{67}, \text{cash}=36005.25]$
LOG	b_8
LOG	c_2
UNDO	$w_1[b_{34}, \text{cash}=8900.67]$
REDO	$w_1[b_{34}, \text{cash}=18900.67]$
LOG	b_9
UNDO	$w_9[b_{67}, \text{cash}=36005.00]$
REDO	$w_9[b_{67}, \text{cash}=20000.00]$
LOG	c_9

You have to recover the database after a system failure using the above log.

1. In the backward scan for undos, what will be the set of committed transactions, the set of uncommitted transactions, and list of undo action(s) be when you reach the *cp* record?

   C = {9, 2, 6}
   I = {1, 8}
   UNDO $w_1[b_{34}, \text{cash} = 8900.67]$

2. Which actions(s) must you undo *before* the *cp* record?

   All from I on CP − C = {1}
   UNDO $w_1[b_{56}, \text{Cash} = 94340.45]$

3. Which actions(s) must you redo?

   All from in C after CP
   REDO $w_6[a_{119}, \text{Rate} = 6.0]$
   REDO $w_2[b_{67}, \text{cash} = 36005.25]$
   REDO $w_9[b_{67}, \text{cash} = 20000.00]$

4. Can you devise a more efficient algorithm that minimises the number of redo and undo actions performed?

   During UNDO phase
   For each REDO $w_i[o]$ where $i \in C$ put o into R
   Only do UNDO $w_j[o]$ if $o \notin R$