SQL: A Language for Database Applications

Imperial College London

Bank Branch Database

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'	[illegible]

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
[illegible]	[illegible]	[illegible]	LL	34
[illegible]	[illegible]	[illegible]	LL	56
[illegible]	[illegible]	[illegible]	50	56
[illegible]	[illegible]	[illegible]	LL	56

mid	no	[illegible]	[illegible]
1000	100	[illegible]	[illegible]
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

key branch(so

key branch(b

key movemen

key account(n

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

account(sortcode) $\overset{fk}{\Rightarrow}$ branch(sortcode)

SQL WHERE expressions in more detail

Testing Strings against a Pattern

```
WHERE column LIKE pattern ESCAPE escape_char
```

Will return TRUE where pattern matches column. The escape_char may be used before any of t ormal text.

- _ to m
- % to m
- Tran
- TransactSQL Only: [ABC] to match a character

List customers whose first initial is P, and have one more ini

```
SELECT  DISTINCT  cname
FROM    account
WHERE   cname LIKE '%, P._.'
```

SQL WHERE expressions in more detail

Testing Strings against a Pattern

```
WHERE column LIKE pattern ESCAPE escape_char
```

Will return TRUE where pattern matches column. The escape_char may be used before any of t ormal text.

- _ to m
- % to m
- Tran
- TransactSQL Only: [ABC] to match a character

List customers whose first initial is between A and L

```
SELECT  DISTINCT cname
FROM    account
WHERE   cname LIKE '%, [A–L].%'
```

Processing the result of project

Modifications to data

Any processing of data to appear in a result set must be placed in the SELECT clause

- Many functions proposed in ANSI SQL, *e.g.*
  - ABS(number) returns the absolute value of any number
  - ROUND(value,dp) rounds a numeric value to dp decimal places
  -
- Tend [illegible] QL compliant, *e.g.*
  - Postgres: LENGTH(object) returns the length of any object (including strings)
  - TrasnsactSQL: LEN(str) returns the length of an

Display accounts with just surnames and rounded rate

PostgresSQL

```
SELECT no,
       ROUND(rate,1) AS rate_1dp,
       SUBSTRING(cname FROM 1 FOR POSITION(',' IN cname)−1) AS surname
FROM   account
```

Quiz 1: SQL extensions to RA select and project

customer				
cname	phone	address	joined	salary
'McBrien, P.'	'02077651234'	'123 Strand, London WC1A'	1999-01-03	30000
'Boyd, M.'	'02077656666'	'33 Aldwych, London'	1999-01-05	NULL
'Poulovassilis, A.'	'02089474321'	'13 Haydons Rd, London SW19'	1999-01-05	40000
'Bailey, J.'	'02089461111'	'22 Queens Rd, London SW19'	1999-01-07	45000

```
SELECT  cname,
        SUBSTRING(address,CHARINDEX(',',address)+2,LEN(address)) AS area
FROM    cust
WHERE   pho
```

What is the r

A

cname	area
Bailey, J.	London SW19
Poulovassilis, A.	London SW19

B

cna	
Bail	
Poulovassilis, A.	13 Haydons Rd

C

cname	area
Poulovassilis, A.	London SW19

D

cname	area
Poulovassilis, A.	13 Haydons Rd

Processing the result of project: CASE statements

CASE statements

A CASE statement may be put in the SELECT clause to process the values being returned.

Display ac

```
SELECT n
       C
       CASE
       WHEN rate>0 AND rate<5.5
       THEN 'low rate'
       WHEN rate>=5.5
       THEN 'high rate'
       ELSE 'zero rate'
       END AS interest_class
FROM   account
```

		est_class
125	0.00	zero rate

Need for yet another type of Join?

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				

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
			15/1/1999
			15/1/1999
			18/1/1999

Listing of m

```
SELECT cname,
       mid
FROM   account NATURAL JOIN
       movement
```

cname	mid
M[illegible]	[illegible]
M[illegible]	[illegible]
M[illegible]	[illegible]
Poulovassilis, A.	1004
Boyd, M.	1005
McBrien, P.	1006
Poulovassilis, A.	1007
McBrien, P.	1008
Poulovassilis, A.	1009

Need for yet another type of Join?

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				

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
			15/1/1999
			15/1/1999
			18/1/1999

Listing an

```
SELECT cname,
       mid
FROM   account NATURAL LEFT JOIN
       movement
```

cname	mid
[illegible]	1000
[illegible]	1001
[illegible]	1002
Poulovassilis, A.	1004
Boyd, M.	1005
McBrien, P.	1006
Poulovassilis, A.	1007
McBrien, P.	1008
Poulovassilis, A.	1009
Bailey, J.	NULL

Left and Right Joins

Left Join

A left join $R \overset{\text{L}}{\bowtie} S$ returns every row in R, even if no rows in S match. In such cases where no row in S matches a row from R, the columns of S are filled with NULL values.

Right Join

A right join [illegible] In such cases where no row in R [illegible] S [illegible] R [illegible] NULL values.

Outer Join

An outer join $R \overset{\text{O}}{\bowtie} S$ returns every row in R, even if no rows in S match, and also returns every row in S even if no row in R matches.

$$R \overset{\text{O}}{\bowtie} S \equiv (R \overset{\text{L}}{\bowtie} S) \cup (R \overset{\text{R}}{\bowtie} S)$$

RA equivalent of LEFT JOIN

```
SELECT A1, ..., An
FROM   R1 LEFT JOIN R2 ON O1 AND ... AND Oi
```

⇓

$$\pi_{A_1,\ldots,A_n}\sigma_{P_1\wedge\ldots\wedge P_k}(\sigma_{O_1\wedge\ldots\wedge O_i}(R_1 \times R_2) \cup (R_1 \qquad {}_2)))$$

- $\omega(R_2)$ returns a row of NULLs with the same number of columns as R_2

Quiz 2: SQL LEFT JOIN ... ON (1)

```
SELECT account.no,
       movement.amount
FROM   account LEFT JOIN movement
                ON  account.no=movement.no
                AND movement.amount<0
```

What is the r[illegible]

A

no	amount

B

no	amount
100	-223.45
107	-100.00

C

no	[illegible]
100	[illegible]
101	[illegible]
103	NULL
107	-100.00
119	NULL
125	NULL

[illegible]	nt
[illegible]	.45
[illegible]	.00
103	0.00
107	-100.00
119	0.00
125	0.00

Quiz 3: SQL LEFT JOIN ... ON (2)

```
SELECT  account.no,
        movement.amount
FROM    account LEFT JOIN movement
                ON    account.no=movement.no
WHERE   movement.amount<0
```

What is the r

A

no	amount

B

no	amount
100	-223.45
107	-100.00

C

no	[illegible]
100	[illegible]
101	[illegible]
103	NULL
107	-100.00
119	NULL
125	NULL

[illegible]	amount
[illegible]	[illegible].45
[illegible]	[illegible].00
103	0.00
107	-100.00
119	0.00
125	0.00

Worksheet: Left, Right and Outer Joins

worksheet_null database

movement			
mid	no	amount	tdate
099			
100			
100			
100			
100			
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

				ortcode
100	'current'	'McBrien, P.'	null	67
101	'deposit'			7
119	'deposit'			6
125	'current'			6

OLTP and OLAP

OLTP

- online transactional processing
- reads and writes to a few rows
- 'stan

```
BEGIN T
     U
     SET cash=cash-10000.00
     WHERE sortcode=56

     UPDATE branch
     SET cash=cash+10000.00
     WHERE sortcode=34
COMMIT TRANSACTION T1
```

OLAP

- online analytical processing
- reads many rows

```
     FROM branch
CO
```

SQL OLAP features: GROUP BY

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	108	145.50	12/1/1999
1006	100	10.23	15/1/1999
1007	107	345.56	15/1/1999
1008	101	1230.00	15/1/199
1009	119	5600.00	18/1/199

⇒

```
:
FROM movement
:
GROUP BY no
```

⇒

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

Aggregat

SUM	Sum the values of all rows in the group
COUNT	Count the number of non-NUL
AVG	Average of the non-NULL valu
MIN	Minimum value in the group
MAX	Maximum value in the group

⋮

GROUP BY rules

- Only one row output per group
- *ANSI SQL says must apply aggregate function to non grouped columns*

SQL OLAP features: GROUP BY

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	15/1/199
1009	119	5600.00	18/1/199

⇨

```
:
:
FROM movement
:
:
GROUP BY no
```

⇨

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

Example o

```
SELECT  no ,
        SUM(amount) AS balance ,
        COUNT(amount) AS no_trans
FROM    movement
GROUP BY no
```

⇨

no	balance	no_trans

GROUP BY rules

- Only one row output per group
- *ANSI SQL says must apply aggregate function to non grouped columns*

Quiz 4: GROUP BY in ANSI SQL

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

Which SQ

A

```
SELECT    no,
          cname,
          AVG(rate)
FROM      account
GROUP BY  no
```

B

```
SELECT    no,
          MIN(cname),
          AVG(rate)
FROM      account
GROUP BY  no
```

C

```
SEL

              MAX(rate)
FROM      account
GROUP BY  no
```

```
(rate)
count
```

SQL OLAP features: Aggregate operators

- Normally use GROUP BY on all non-aggregated attributes.

```
SELECT no,
       SUM(amount) AS total,

FRO
GRO
```

no	total	trans
119	5600.00	1
107	245.56	2

- Don't f

```
SELECT COUNT(DISTINCT no) AS active_acc
FROM   movement
```

- NULL attributes don't count!

```
SELECT COUNT(rate) AS no_rates
FROM   account
```

no_rates
2

Quiz 5: GROUP BY over NULL values (1)

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
1006	100	10.23	15/1/1999
1008	101	1230.00	15/1/1999
1009			
1010			
1011			
1012			
1013			

account				
no	type	cname	rate	sortcode
100	'current'	'McBrien, P.'	NULL	67
101	'deposit'	'McBrien, P.'	5.25	67
107	'deposit'	'Poulovassilis, A.'	5.50	56
125	'current'	'Bailey, J.'	NULL	56

```
SELECT movement.no,
                                    _trans,
                                   _value
```

What is the result of the above query?

A

no	no_trans	min_value
119	2	45.00
101	2	1230.00
107	1	-100.00
100	3	-223.45
103	1	145.50

B

no	no_trans	min_value
101	2	1230.00
100	4	-223.45
119	2	45.00

C

no	no_trans	min_value
101	2	1230.00
100	4	NULL
119	2	45.00

no	no_trans	min_value
101	2	1230.00
100	3	-223.45
119	2	45.00

Quiz 6: GROUP BY over NULL values (2)

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
1006	100	10.23	15/1/1999
1008	101	1230.00	15/1/1999
1009	119	5600.00	18/1/1999
1010			
1011			
1012			
1013			

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

```
SELECT movement.no,
                                      e
```

What is the result of the above query?

A

no	balance
NULL	NULL
NULL	600.00
NULL	-46.00
119	5645.00
101	5230.00
100	2086.78

B

no	balance
NULL	600.00
NULL	-46.00
119	5645.00
101	5230.00
100	2086.78

C

no	balance
NULL	554.00
119	5645.00
101	5230.00
100	2086.78

no	balance
119	5645.00
101	5230.00
100	2086.78

Selecting results from aggregates: HAVING

GROUP BY in the RA

- An extension to the RA includes a group by operator
- In SQL, the GROUP BY operator is applied *outside* the $\sigma_P(\ldots \times \ldots)$
- To execute a σ_P *outside* the GROUP BY, you must place the predicates P in a HAVING clause

```
SELECT


FROM
GROUP BY no
HAVING   SUM(amount)>2000
```

⇒

ns
3
2
1

Ordering of SQL clauses

- HAVING is executed after GROUP BY, but before SELECT
- Can be used to avoid divide by zero errors

```
SELECT    no,
          MAX(amount)/MIN(amount) AS variance_ratio
FROM      movement
GROUP BY  movement.no
HAVING    MIN(amount)<>0
```

Quiz 7: HAVING

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		
1009	119		

```
SELECT   account.no,
         account.cname,
         SUM(movement.amount) AS balance
FROM     account NATURAL JOIN movement
                                       00
                                       )>
                                       t)>1000
```

no	type			
100	'curr			
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

What is the result of the above query?

A

no	cname	balance
101	McBrien, P.	5230.00

B

no	cname	balance
101	McBrien, P.	5230.00
119	Poulovassilis, A.	5600.00

C

no	cname	balance
100	McBrien, P.	2086.78
101	McBrien, P.	5230.00

D

no	cname	balance
100	McBrien, P.	2086.78
101	McBrien, P.	5230.00
119	Poulovassilis, A.	5600.00

SQL OLAP features: PARTITION

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
1009	119	5600.00	18/1

```
⋮
OVER (PARTITION BY no)
FROM movement
⋮
```

movement			
mid	no	amount	tdate
1000	100	2300.00	5/1/1999
1002	100	-223.45	8/1/1999
1006	100	10.23	15/1/1999
1001	101	4000.00	5/1/1999
1008	101	1230.00	15/1/1999
1004	107	-100.00	11/1/1999
1007	107	345.56	15/1/1999

/1999

/1999

```
SELECT mid,
       no,
       amount,
       SUM(amount) OVER (PARTITION BY no) AS balance
FROM   movement
```

			balance
			2086.78
1002	100	-223.45	2086.78
1006	100	10.23	2086.78
			0
			0
			0
			0
			0
			0

PARTITION BY

- One row output per input row
- Aggregates apply to partition

Relationally Complete SQL

Relational Completeness

- Relational completeness in SQL means being able to fully support the RA in SQL
- 'pure' RA can be fully supported by SQL
- Aggregates require 'relationally complete' SQL
  -
  - S

```
SELECT   SUM
INTO     #to   _
FROM     movement
```

```
13307.84
```

```
SELECT     movement.no,
           SUM(movement.amount) AS balance,
           ROUND(100*SUM(movement.amount)/
                #total_balance.total,1) AS pc
FROM       movement,
           #total_balance
GROUP BY   movement.no,#total_balance.total
ORDER BY   movement.no
```

```
101  5230.00  39.3
103   145.50   1.1
107   245.56   1.8
119  5600.00  42.1
```

Relationally Complete SQL

Relational Completeness

- Relational completeness in SQL means being able to fully support the RA in SQL
- ‘pure’ RA can be fully supported by SQL
- Aggregates require ‘relationally complete’ SQL
  -
  - S

```
SELECT

         ROUND(100*SUM(movement.amount)/total_balance.total,1) AS pc
FROM     movement,
         (SELECT SUM(amount) AS total FROM
GROUP BY movement.no,total_balance.total
ORDER BY movement.no
```

no	balance	pc
100	2086.78	15.7
101	5230.00	39.3
103	145.50	1.1
107	245.56	1.8
119	5600.00	42.1

SQL OLAP features: Ordering Rows

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

```
SELECT    mid,
          tdate,
          amount
FROM      movement
ORDER BY  mid
```

mi		
100		
100		
100		
1004	1999-01-11	-100.00
1005	1999-01-12	145.50
1006	1999-01-15	10.23
1007	1999-01-15	345.56
1008	1999-01-15	1230.00
1009	1999-01-18	5600.00

SQL OLAP features: Ranking Rows

```
SELECT mid
       tdate ,
       a
       R

FROM
```

mid	tdate	amount	rank
1009	1999-01-18	5600.00	1
1001	1999-01-05	4000.00	2
1000	1999-01-05	2300.00	3
		.00	4
		.56	5
		.50	6
		.23	7
		.00	8
		.45	9

- RANK function provides normal concept of ranking
- DENSE_RANK function will not skip numbers wher ntical
- Only in Postgres since verison 9.0

Quiz 8: Execution of SQL clauses

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

What orde

A

SELECT
FROM
WHERE
GROUP BY
HAVING
ORDER BY

B

FROM
WHERE
SELECT
GROUP BY
HAVING
ORDER BY

C

FROM
WHERE
GROUP BY
HAVING
SELECT
ORDER BY

[illegible]
[illegible]
GROUP BY
WHERE
FROM
SELECT

OLAP: Pivot

- for presentation purposes useful to change layout of table
- information spread over rows is instead spread over columns

```
SELECT

FROM     account JOIN branch
         ON account.sortcode=
              branch.sortcode
GROUP BY branch.sortcode,
         branch.bname,
         account.type
ORDER BY branch.sortcode,
         branch.bname
```

			qty
34	Goodge St	current	1

SQL OLAP: Pivot using CASE statements

```
SELECT    branch.sortcode,
          branch.bname,
          COUNT(CASE WHEN type='current' THEN no ELSE NULL END) AS current,
          COUNT(CASE WHEN type='deposit' THEN no ELSE NULL END) AS deposit,
          COUNT(CASE WHEN type NOT IN ('current','deposit') THEN no
          ELSE NULL END) AS other
FROM
GROUP BY
ORDER BY
```

branch account types pivot				
sortcode	bname	current	deposit	other
34	Goodge St	1	0	0
56	Wimbledon	2	1	0
67	Strand	1	1	0

- one case for each value
- wise to have a default case

Worksheet: OLAP Queries in SQL

movement			
mid	no	amount	tdate
1000	1		
1001	1		
1002	1		
1004	1		
1005	1		
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
125	'current'	'Ba	N	56

movement.no

Worksheet: OLAP Queries Questions 3 & 4

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 accou

4 Writ (no,c nt, and for e 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 p ccount. For the current data this should result in:

no	cname	type	pc_cust_fun	
100	McBrien, P.	current	28.	
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

Worksheet: OLAP Queries in SQL (3)

```
SELECT     account.cname,
                                                                    ,
                          ELSE null END),0.0) AS deposit_balance
FROM       account LEFT JOIN movement ON
GROUP BY   account.cname
```

Worksheet: OLAP Queries in SQL (4)

```
SELECT DISTINCT account.no,
       accou
       accou
       ROU                                                                   .no),0.0)/
       AS
       ROU                                                                   .no),0.0)/
                SUM(movement.amount) OVER (PARTITION BY account.type),2
       AS pc_type_funds
FROM   account LEFT JOIN movement ON account.no=mov
```

SQL OLAP: Un-pivot using UNION statements

Un-pivot the account table to triple format

```
SELECT  no,
        'cname' AS col,
        cname AS value
FROM    account
UNION
SELECT  no
        'typ
        typ
FROM    acc
UNION
SELECT  no,
        'rate',
        CAST(rate AS VARCHAR)
FROM    account
WHERE   rate IS NOT NULL
UNION
SELECT  no,
        'sortcode',
        CAST(sortcode AS VARCHAR)
FROM    account
```

no	col	value
100	cname	McBrien, P.
100	sortcode	67
100	type	current
101	cname	McBrien, P.
103	type	current
107		
107		
107		
119		
119		
119	sortcode	56
119	type	deposit
125	cname	Bailey, J.
125	sortcode	56
125	type	current

SQL Functions

FUNCTION

- Most SQL implementations support some variant of ANSI SQL FUNCTION
- Details vary . . .

TransactSQL function to return cnames reformatted

```
CREATE FUN
       RE
BEGIN
  DECLA

  SELECT @ifcname=
       SUBSTRING(@cname,CHARINDEX(',',
       SUBSTRING(@cname,1,CHARINDEX(','

  RETURN @ifcname
END
```

```
SELECT no,
       dbo.cname_to_initial_first(
           account.cname) AS cname
FROM   account
```

no	cname
100	P.McBrien
101	P.McBrien
103	M.Boyd
107	A.Poulovassilis
119	A.Poulovassilis
125	J.Bailey

SQL Procedures

PROCEDURE

- No specific PROCEDURE construct in Postgres
- TransactSQL supports PROCEDURE definition, and generally refers to them a stored procedure

Transact

```
CREATE PR
(   @from
    @to_b
    @total DECIMAL(10,2)
) AS
BEGIN
   UPDATE branch
   SET cash=cash−@total
   WHERE sortcode=@from_branch

   UPDATE branch
   SET cash=cash+@total
   WHERE sortcode=@to_branch
END
```

SQL Constraints

$\forall No, Rate.account(No, \_, \_, Rate, \_) \rightarrow Rate \geq 0.00$

```
ALTER TABLE account
ADD CONSTRAINT check_account_rate
CHECK (rate >=0.00)
```

IF account [illegible] C)

```
CREATE FUN[illegible]C INT)
RETURNS BIT A
BEGIN
   IF EXISTS (SELECT *
              FROM    current_account
              WHERE   no=@NO
              AND     cname=@CN
              AND     sortcode=@SC)
      RETURN 1

   RETURN 0
END;

ALTER TABLE account
ADD CONSTRAINT check_current_account
CHECK (type<>'current' OR dbo.is_in_current_account(no,cname,sortcode)=1);
```