

SQL

P.J. McBrien

Imperial College London

Topic 5: SQL, An Implementation of The RA

P.J. McBrien

Imperial College London

SQL

Development of Relational Database Systems

- Relation Model and Algebra proposed by C.J.Codd in 1970
- IBM developed a prototype relational database called **System R** with a query language **Structured English Query Language (SEQUEL)**
- SEQUEL later renamed **SQL**
- Various commercial versions of SQL launched in late 1970's/early 1980s
 - **DB2**
 - **Oracle**
 - **Sybase**
 - ...

SQL

Development of Relational Database Systems

- Relation Model and Algebra proposed by C.J.Codd in 1970
- IBM developed a prototype relational database called **System R** with a query language **Structured English Query Language (SEQUEL)**
- SEQUEL later renamed **SQL**
- Various commercial versions of SQL launched in late 1970's/early 1980s
 - **DB2**
 - **Oracle**
 - **Sybase**
 - ...

SQL Language Components

Data Definition Language (DDL): a relational schema with data

Data Manipulation Language (DML): a relational query and update language

SQL DDL: Definition of Tables

```
CREATE TABLE branch
(  sortcode INTEGER NOT NULL,
   bname VARCHAR(20) NOT NULL,
   cash DECIMAL(10,2) NOT NULL
)
```

branch		
sortcode	bname	cash

```
CREATE TABLE account
(  no INTEGER NOT NULL,
   type VARCHAR(8) NOT NULL,
   cname VARCHAR(20) NOT NULL,
   rate DECIMAL(4,2) NULL,
   sortcode INTEGER NOT NULL
)
```

account				
no	type	cname	rate?	sortcode

SQL DDL: SQL Data Types

Some SQL Data Types	
Keyword	Semantics
BOOLEAN	A logical value (TRUE, FALSE, or UNKNOWN)
BIT	1 bit integer (0, 1, or NULL)
INTEGER	32 bit integer
BIGINT	64 bit integer
FLOAT(n)	An n bit mantissa floating point number
REAL	32 bit floating point number (\equiv FLOAT(24))
DOUBLE PRECISION	64 bit floating point number (\equiv FLOAT(53))
DECIMAL(p,s)	A p digit number with s digits after the decimal point
CHAR(n)	A fixed length string of n characters
VARCHAR(n)	A varying length string of upto n characters
DATE	A calendar date (day, month and year)
TIME	A time of day (seconds, minutes, hours)
TIMESTAMP	time and day together
ARRAY	An ordered list of a certain datatype
MULTISET	A bag (<i>i.e.</i> unordered list) of a certain datatype
XML	XML text

SQL DDL: Definition of Keys

```
CREATE TABLE branch
(
  sortcode INTEGER NOT NULL,
  bname VARCHAR(20) NOT NULL,
  cash DECIMAL(10,2) NOT NULL,
  CONSTRAINT branch_pk PRIMARY KEY (sortcode)
)
```

branch		
<u>sortcode</u>	bname	cash

```
CREATE TABLE account
(
  no INTEGER NOT NULL,
  type VARCHAR(8) NOT NULL,
  cname VARCHAR(20) NOT NULL,
  rate DECIMAL(4,2) NULL,
  sortcode INTEGER NOT NULL,
  CONSTRAINT account_pk PRIMARY KEY (no),
  CONSTRAINT account_fk FOREIGN KEY (sortcode)
  REFERENCES branch
)
```

account				
<u>no</u>	type	cname	rate	sortcode

account(sortcode) \xrightarrow{fk} branch(sortcode)

Keys and the Primary Key

Keys

The alternative keys of a table are called **candidate keys**

Keys and the Primary Key

Keys

The alternative keys of a table are called **candidate keys**

Primary Key

- Choose the key most often used to access a table as the **primary key**
- Has no logical impact on the relational model
- Has an operation impact: index created that accesses the data faster
- All other keys are called **secondary keys**

Keys and the Primary Key

Keys

The alternative keys of a table are called **candidate keys**

Primary Key

- Choose the key most often used to access a table as the **primary key**
- Has no logical impact on the relational model
- Has an operation impact: index created that accesses the data faster
- All other keys are called **secondary keys**

Declaring Primary Keys after table creation

```
ALTER TABLE branch  
ADD CONSTRAINT branch_pk PRIMARY KEY (sortcode);
```

Keys and the Primary Key

Keys

The alternative keys of a table are called **candidate keys**

Primary Key

- Choose the key most often used to access a table as the **primary key**
- Has no logical impact on the relational model
- Has an operation impact: index created that accesses the data faster
- All other keys are called **secondary keys**

Declaring Primary Keys after table creation

```
ALTER TABLE branch  
ADD CONSTRAINT branch_pk PRIMARY KEY (sortcode);
```

Declaring Secondary Keys for a table

```
CREATE UNIQUE INDEX branch_bname_key ON branch(bname)
```

SQL DML: Inserting, Updating and Deleting Data

```

INSERT INTO account
VALUES (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)

```

account				
<u>no</u>	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

SQL DML: Inserting, Updating and Deleting Data

```

INSERT INTO account
VALUES (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)

```

```

UPDATE account
SET type='deposit'
WHERE no=100

```

account				
<u>no</u>	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

account				
<u>no</u>	type	cname	rate	sortcode
100	'deposit'	'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

SQL DML: Inserting, Updating and Deleting Data

```

INSERT INTO account
VALUES (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)

```

```

UPDATE account
SET type='deposit'
WHERE no=100

```

```

DELETE
FROM account
WHERE no=100

```

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

account				
no	type	cname	rate	sortcode
100	'deposit'	'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

account				
no	type	cname	rate	sortcode
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


SQL DML: An Implementation of the RA

SQL **SELECT** statements: Rough Equivalence to RA

```

SELECT A1, ..., An
FROM R1, ..., Rm
WHERE P1
AND ...
AND Pk

```


 $\pi_{A_1, \dots, A_n} \sigma_{P_1 \wedge \dots \wedge P_k} R_1 \times \dots \times R_m$

SQL **SELECT** implements RA π , σ and \times

$\pi_{\text{bname, no}} \sigma_{\text{branch.sortcode}=\text{account.sortcode} \wedge \text{account.type}=\text{'current'}} (\text{branch} \times \text{account})$

```

SELECT branch.bname,
       account.no
FROM   account, branch
WHERE  account.sortcode=branch.sortcode
AND    account.type='current'

```


SQL DML: An Implementation of the RA

SQL **SELECT** statements: Rough Equivalence to RA

```

SELECT A1, ..., An
FROM R1, ..., Rm
WHERE P1
AND ...
AND Pk

```



$$\pi_{A_1, \dots, A_n} \sigma_{P_1 \wedge \dots \wedge P_k} R_1 \times \dots \times R_m$$

SQL **SELECT** implements RA π , σ and \times

$\pi_{\text{bname, no}} \sigma_{\text{branch.sortcode}=\text{account.sortcode} \wedge \text{account.type}=\text{'current'}} (\text{branch} \times \text{account})$

```

SELECT branch.bname,
       account.no
FROM   account JOIN branch
       ON account.sortcode=branch.sortcode
WHERE  account.type='current'

```


Naming columns in SQL

Column naming rules in SQL

- You must never have an ambiguous column name in an SQL statement
- You can use **SELECT *** to indicate all columns (*i.e.* have no projection)
- You can use **tablename.*** to imply all columns from a table



```
SELECT branch.bname,
       account.sortcode
FROM   account JOIN branch
       ON account.sortcode=
         branch.sortcode
WHERE  account.type='current'
```



```
SELECT bname,
       sortcode
FROM   account JOIN branch
       ON account.sortcode=
         branch.sortcode
WHERE  type='current'
```



```
SELECT bname,
       account.sortcode
FROM   account JOIN branch
       ON account.sortcode=
         branch.sortcode
WHERE  type='current'
```



```
SELECT branch.*,
       no
FROM   account JOIN branch
       ON account.sortcode=
         branch.sortcode
WHERE  type='current'
```



sortcode	bname	cash	no
67	'Strand'	34005.00	100
34	'Goodge St'	8900.67	103
56	'Wimbledon'	94340.45	107
56	'Wimbledon'	94340.45	125

Quiz 5.1: Translating RA into SQL

Which SQL query implements $\pi_{\text{bname, no}} \sigma_{\text{type}='deposit'}(\text{account} \bowtie \text{branch})$?

A

```
SELECT *
FROM account , branch
WHERE type='deposit'
```

B

```
SELECT bname , no
FROM account , branch
WHERE type='deposit'
```

C

```
SELECT bname , no
FROM branch , account
WHERE branch.sortcode=
       account.sortcode
AND type='deposit'
```

D

```
SELECT bname , no
FROM account , branch
WHERE branch.sortcode=
       account.no
AND type='deposit'
```

Connectives Between SQL **SELECT** statementsBinary operators between **SELECT** statements

- SQL **UNION** implements $RA \cup$
- SQL **EXCEPT** implements $RA -$
- SQL **INTERSECT** implements $RA \cap$

Note that two tables must be **union compatible**: have the same number and type of columns

$\pi_{no}account - \pi_{no}movement$

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

SQL NATURAL JOIN

Natural join performs a join on the attributes that appear in both relations, and returns each attribute once

e.g. branch ⋈ account joins on sortcode

Best to use **JOIN ... ON** or **USING**

SQL NATURAL JOIN

```
SELECT *  
FROM   branch NATURAL JOIN  
       account
```



sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	null
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Goode St	8900.67	103	current	Boyd, M.	null
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	null
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	null

SQL NATURAL JOIN

Natural join performs a join on the attributes that appear in both relations, and returns each attribute once

e.g. branch ⋈ account joins on sortcode

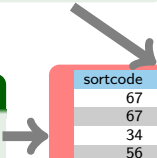
Best to use **JOIN ... ON** or **USING**

SQL JOIN ... ON

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
ON branch.sortcode=account.sortcode
```

SQL NATURAL JOIN

```
SELECT *
FROM branch NATURAL JOIN
account
```



sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	null
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Goodge St	8900.67	103	current	Boyd, M.	null
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	null
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	null

SQL NATURAL JOIN

Natural join performs a join on the attributes that appear in both relations, and returns each attribute once

e.g. branch ⋈ account joins on sortcode

Best to use **JOIN ... ON** or **USING**

SQL JOIN ... ON

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
ON branch.sortcode=account.sortcode
```

SQL NATURAL JOIN

```
SELECT *
FROM branch NATURAL JOIN
account
```

sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	null
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Goodge St	8900.67	103	current	Boyd, M.	null
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	null
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	null

SQL JOIN ... USING

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
USING (sortcode)
```

SQL NATURAL JOIN

Natural join performs a join on the attributes that appear in both relations, and returns each attribute once

e.g. branch ⋈ account joins on sortcode

Best to use **JOIN ... ON** or **USING**

SQL JOIN ... ON

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
ON branch.sortcode=account.sortcode
```

Classic SQL

```
SELECT branch.*, no, type, cname, rate
FROM branch, account
WHERE branch.sortcode=account.sortcode
```

SQL NATURAL JOIN

```
SELECT *
FROM branch NATURAL JOIN
account
```

sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	null
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Goodge St	8900.67	103	current	Boyd, M.	null
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	null
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	null

SQL JOIN ... USING

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
USING (sortcode)
```

Natural join performs a join on the attributes that appear in both relations, and returns each attribute once

e.g. branch ⋈ account joins on sortcode

Best to use JOIN ... ON or USING

SQL NATURAL JOIN

SQL JOIN ... ON

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
ON branch.sortcode=account.sortcode
```

Classic SQL

```
SELECT branch.*, no, type, cname, rate
FROM branch, account
WHERE branch.sortcode=account.sortcode
```

SQL NATURAL JOIN

```
SELECT *
FROM branch NATURAL JOIN
account
```

sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	null
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Godge St	8900.67	103	current	Boyd, M.	null
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	null
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	null

SQL JOIN ... USING

```
SELECT branch.*, no, type, cname, rate
FROM branch JOIN account
USING (sortcode)
```

SQL CROSS JOIN

```
SELECT branch.*, no, type, cname, rate
FROM branch CROSS JOIN account
WHERE branch.sortcode=account.sortcode
```


Overview of RA and SQL correspondences

RA and SQL	
RA Operator	SQL Operator
π	SELECT
σ	WHERE
$R_1 \times R_2$	FROM R_1, R_2 <i>or</i> FROM R_1 CROSS JOIN R_2
$R_1 \bowtie R_2$	FROM R_1 NATURAL JOIN R_2
$R_1 \bowtie_{\theta} R_2$	FROM R_1 JOIN R_2 ON θ
$R_1 - R_2$	R_1 EXCEPT R_2
$R_1 \cup R_2$	R_1 UNION R_2
$R_1 \cap R_2$	R_1 INTERSECT R_2

Table and Column Aliases

AS: Alias

- A logical copy of a column or table can be produced using **AS**
- When applied to a table, allows to compare one row of a table with another row of the same table in a join

Table and Column Aliases

AS: Alias

- A logical copy of a column or table can be produced using **AS**
- When applied to a table, allows to compare one row of a table with another row of the same table in a 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	'current'	'Bailey, J.'	NULL	56

AS current_account

current_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

AS deposit_account

deposit_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

Table and Column Aliases

```

SELECT current_account.cname,
       current_account.no AS current_no,
       deposit_account.no AS deposit_no
FROM   account AS current_account
JOIN   account AS deposit_account
ON     current_account.cname=
       deposit_account.cname
AND    current_account.type='current'
AND    deposit_account.type='deposit'

```

cname	current
McBrien, P.	
Poulovassilis, A.	

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

AS current_account

AS deposit_account

current_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

deposit_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

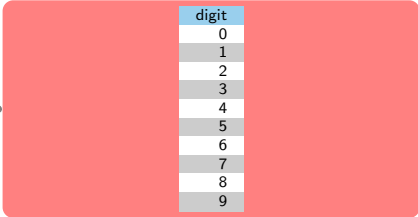
Set or Bag Based Semantics: No Iteration

SQL allows either set (**DISTINCT**) or bag (**ALL**) based semantics

- Lack of procedural semantics means all queries can execute in parallel
- Might need to think about 'programming' in different ways

Table of Constants

```
SELECT *  
FROM (VALUES (0),(1),(2),(3),(4),  
            (5),(6),(7),(8),(9))  
     AS decimal(digit)
```



digit
0
1
2
3
4
5
6
7
8
9

Set or Bag Based Semantics: No Iteration

SQL allows either set (**DISTINCT**) or bag (**ALL**) based semantics

- Lack of procedural semantics means all queries can execute in parallel
- Might need to think about 'programming' in different ways

Generating 0..999

```
SELECT hundred.digit*100+ten.digit*10+unit.digit AS n
FROM   (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS unit(digit)
       CROSS JOIN (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS ten(digit)
       CROSS JOIN (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS hundred(digit)
ORDER BY n
```



n	n	n	n	n	n
0	10	20	30	980	990
1	11	21	31	981	991
2	12	22	32	982	992
3	13	23	33	983	993
4	14	24	34	984	994
5	15	25	35	985	995
6	16	26	36	986	996
7	17	27	37	987	997
8	18	28	38	988	998
9	19	29	39	989	999

Common Table Expressions: SQL WITH

```
SELECT hundred.digit*100+ten.digit*10+unit.digit AS n
FROM (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS unit(digit)
CROSS JOIN (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS ten(digit)
CROSS JOIN (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9)) AS hundred(digit)
```



```
WITH decimal(digit) AS (VALUES (0),(1),(2),(3),(4),(5),(6),(7),(8),(9))
SELECT hundred.digit*100+ten.digit*10+unit.digit AS n
FROM decimal AS unit
CROSS JOIN decimal AS ten
CROSS JOIN decimal AS hundred
```



n	n	n	n	n	n
0	10	20	30	980	990
1	11	21	31	981	991
2	12	22	32	982	992
3	13	23	33	983	993
4	14	24	34	984	994
5	15	25	35	985	995
6	16	26	36	986	996
7	17	27	37	987	997
8	18	28	38	988	998
9	19	29	39	989	999

A **Common Table Expression (CTE)** may define a table that is used in one query

`WITH table AS def SELECT ...`

Use to avoid repeating definitions

Topic 6: SQL Bags and Sets

P.J. McBrien

Imperial College London

SQL: Bags and Sets

```
SELECT sortcode
FROM account
```

$\approx \pi_{\text{sortcode}} \text{account}$
sortcode
67
67
56
56
56
34

```
SELECT DISTINCT sortcode
FROM account
```

$\pi_{\text{sortcode}} \text{account}$
sortcode
34
56
67

SQL **SELECT**: Bag semantics

- By default, an SQL **SELECT** (equivalent to an RA π) does *not* eliminate duplicates, and returns a **bag** (or **multiset**) rather than a set.
- Any **SELECT** that does not cover a key of the input relation, and requires a set based answer, should use **DISTINCT**.

SQL: Bags and Sets

```
SELECT ALL sortcode
FROM account
```

$\approx \pi_{\text{sortcode}} \text{account}$
sortcode
67
67
56
56
56
34

```
SELECT DISTINCT sortcode
FROM account
```

$\pi_{\text{sortcode}} \text{account}$
sortcode
34
56
67

SQL **SELECT**: Bag semantics

- By default, an SQL **SELECT** (equivalent to an RA π) does *not* eliminate duplicates, and returns a **bag** (or **multiset**) rather than a set.
- Any **SELECT** that does not cover a key of the input relation, and requires a set based answer, should use **DISTINCT**.

Quiz 6.1: Correct use of **SELECT DISTINCT** (1)

branch(sortcode,bname,cash)

key branch(sortcode)

key branch(bname)

Which SQL query requires the use of **DISTINCT** in order to avoid the possibility of a bag being produced?

A

```
SELECT *  
FROM branch  
WHERE cash > 10000
```

B

```
SELECT sortcode  
FROM branch  
WHERE cash > 10000
```

C

```
SELECT bname, cash  
FROM branch
```

D

```
SELECT cash  
FROM branch  
WHERE cash > 10000
```

Quiz 6.2: Correct use of **SELECT DISTINCT** (2)

branch(sortcode,bname,cash)
 account(no,type,cname,rate,sortcode)

key branch(sortcode)
 key branch(bname)
 key account(no)

Which SQL query requires the use of **DISTINCT** in order to avoid the possibility of a bag being produced?

A

```
SELECT *
FROM branch NATURAL JOIN
account
```

B

```
SELECT branch.sortcode , type , rate
FROM branch NATURAL JOIN
account
```

C

```
SELECT branch.sortcode , no
FROM branch NATURAL JOIN
account
```

D

```
SELECT branch.sortcode , no , cash
FROM branch NATURAL JOIN
account
```

Quiz 6.3: Operators that might produce bags

If R and S are sets, which RA operator could produce a bag result if the implementation did not check for duplicates?

A

 σR

B

 $R \cup S$

C

 $R - S$

D

 $R \times S$

Bag and Set operations in SQL

RA Operator	Set Based SQL	Bag Based SQL
π_{A_1, \dots, A_n}	SELECT DISTINCT A_1, \dots, A_n	SELECT ALL A_1, \dots, A_n
$R_1 \times \dots \times R_m$	FROM R_1, \dots, R_m	FROM R_1, \dots, R_m
σ_{P_1, \dots, P_k}	WHERE P_1 AND \dots AND P_k	WHERE P_1 AND \dots AND P_k
$R_1 \cup R_2$	R_1 UNION DISTINCT R_2	R_1 UNION ALL R_2
$R_1 - R_2$	R_1 EXCEPT DISTINCT R_2	R_1 EXCEPT ALL R_2
$R_1 \cap R_2$	R_1 INTERSECT DISTINCT R_2	R_1 INTERSECT ALL R_2

Choosing between set and bag semantics

If you omit **DISTINCT** or **ALL**, then the defaults are:

SELECT ALL

UNION DISTINCT

EXCEPT DISTINCT

INTERSECT DISTINCT

No **FROM DISTINCT** or **WHERE DISTINCT**?

There is no need for **DISTINCT** or **ALL** around **FROM** (\times) and **WHERE** (σ) cannot introduce any duplicates, and any existing duplicates can be removed in the **SELECT**

Project-Select-Product Queries

SQL **SELECT** statements: Exact Equivalence to RA

```

SELECT DISTINCT A1, ..., An
FROM   R1, ..., Rm
WHERE  P1
AND    ...
AND    Pk

```

$$\equiv \pi_{A_1, \dots, A_n} \sigma_{P_1 \wedge \dots \wedge P_k} R_1 \times \dots \times R_m$$

- SQL **SELECT** implements RA π, σ and \times
- Omit **DISTINCT** when either
 - you know A_1, \dots, A_n cover a key
 - you want a bag (rather than set) answer

Quiz 6.4: SQL EXCEPT

```

SELECT no
FROM movement
EXCEPT
SELECT no
FROM account

```

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

What is the result of the above SQL query?

A

no
100
101
103
107
110

B

no
100
101
103
107
110

C

no
100
100
101
107

D

no
100

Quiz 6.5: SQL EXCEPT ALL

```

SELECT no
FROM movement
EXCEPT ALL
SELECT no
FROM account

```

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

What is the result of the above SQL query?

A

no
100
101
103
107
110

B

no
100
101
103
107
110

C

no
100
100
101
107

D

no
100

Worksheet: Translating Between Relational Algebra and SQL

account				
<u>no</u>	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			
<u>mid</u>	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

$\text{movement}(\text{no}) \xrightarrow{fk} \text{account.no}$

Topic 7: Experiment with SQL in DoC

P.J. McBrien

Imperial College London

Try some examples yourself ...

```
medusa-s2(pjm)-4$ psql -h db -U lab -d bank_branch -W
Password:
bank_branch=> SELECT *
bank_branch-> FROM branch NATURAL JOIN account;
```

sortcode	bname	cash	no	type	cname	rate
67	Strand	34005.00	100	current	McBrien, P.	
67	Strand	34005.00	101	deposit	McBrien, P.	5.25
34	Goodge St	8900.67	103	current	Boyd, M.	
56	Wimbledon	94340.45	107	current	Poulovassilis, A.	
56	Wimbledon	94340.45	119	deposit	Poulovassilis, A.	5.50
56	Wimbledon	94340.45	125	current	Bailey, J.	

Try changing some examples ...

```
SELECT bname,  
       sortcode  
FROM   account, branch  
WHERE  account.sortcode=  
       branch.sortcode  
AND    type='current'
```

```
SELECT bname,  
       sortcode  
FROM   account  
       NATURAL JOIN branch  
WHERE  type='current'
```

Topic 8: SQL Set Operations

P.J. McBrien

Imperial College London

Set Operations: IN

IN operator tests for membership of a set

```
SELECT *  
FROM   account  
WHERE  type='current'  
AND    no IN (100,101)
```

Set Operations: IN

IN operator tests for membership of a set

```
SELECT *  
FROM account  
WHERE type='current'  
AND no IN (100,101)
```

Can use nested SELECT to generate set

```
SELECT no  
FROM account  
WHERE type='current'  
AND no IN (SELECT no  
           FROM movement  
           WHERE amount > 500)
```


Set Operations: IN

IN operator tests for membership of a set

```
SELECT *
FROM account
WHERE type='current'
AND no IN (100,101)
```

Can use nested SELECT to generate set

```
SELECT no
FROM account
WHERE type='current'
AND no IN (SELECT no
           FROM movement
           WHERE amount > 500)
```



no
100

Set Operations: IN

IN operator tests for membership of a set

```
SELECT *
FROM   account
WHERE  type='current'
AND    no IN (100,101)
```

Can use nested SELECT to generate set

```
SELECT no
FROM   account
WHERE  type='current'
AND    no IN (SELECT no
              FROM   movement
              WHERE  amount > 500)
```

```
≡
SELECT DISTINCT account.no
FROM   account JOIN movement
ON     account.no=movement.no
WHERE  type='current'
AND    amount > 500
```

Quiz 8.1: SQL Set Membership Testing

```

SELECT no
FROM account
WHERE type='current'
AND no NOT IN
( SELECT no
  FROM movement
  WHERE amount > 500)

```

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			
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

What is the result of the above SQL query?

A

no
100
103
107
125

B

no
100
103
107

C

no
103
107
125

D

no
103
107

Quiz 8.1: SQL Set Membership Testing

```

SELECT no
FROM account
WHERE type='current'
AND no NOT IN
( SELECT no
  FROM movement
  WHERE amount>500)

```

```

SELECT DISTINCT account.no
FROM account
JOIN movement
ON account.no=movement.no
WHERE type='current'
AND NOT amount>500

```

≠

(Gives answer B)

What is the result of the above SQL query?

A

no
100
103
107
125

B

no
100
103
107

C

no
103
107
125

D

no
103
107

Set Operations: EXISTS

Testing for Existence

- **IN** can be used to test if some value is in a relation, either listed, or produced by some **SELECT** statement
- **EXISTS** can be used to test if a **SELECT** statement returns any rows

List people without a deposit account

```

SELECT DISTINCT cname
FROM   account
WHERE  cname NOT IN
( SELECT cname
  FROM   account
  WHERE  type='deposit' )

SELECT DISTINCT cname
FROM   account
WHERE  NOT EXISTS
≡ ( SELECT *
    FROM   account AS deposit_account
    WHERE  type='deposit'
    AND    account.cname=cname )

```

cname
'Boyd, M.'
'Bailey, J.'

Correlated Subquery

Correlated Subquery

- A correlated subquery contains a reference to the columns of the outer query in which the subquery is contained
- Conceptually, result is as if the subquery were executed for each row considered by the **WHERE** clause

List people without a deposit account

```
SELECT DISTINCT cname
FROM account
WHERE NOT EXISTS (SELECT *
                  FROM account AS deposit_account
                  WHERE type='deposit'
                  AND account.cname=deposit_account.cname)
```

cname
'Boyd, M.'
'Bailey, J.'

Set Operations: EXISTS

NOT EXISTS and EXCEPT

- Most queries involving **EXCEPT** can be also written using **NOT EXISTS**
- **EXCEPT** relatively recent addition to SQL

$$\pi_{\text{no}}\text{account} - \pi_{\text{no}}\text{movement}$$

<pre>SELECT no FROM account EXCEPT SELECT no FROM movement</pre>	≡	<pre>SELECT no FROM account WHERE NOT EXISTS (SELECT no FROM movement WHERE no=account.no)</pre>
--	---	--

Set Operations: **SOME** and **ALL**

Can test a value against members of a set

- V op **SOME** S is **TRUE** if there is at least one $V_s \in S$ such that V op V_s
- V op **ALL** S is **TRUE** if there are no values $V_s \in S$ such that **NOT** V op V_s

names of branches that only have current accounts

```
SELECT bname
FROM branch
WHERE 'current' = ALL (SELECT type
                       FROM account
                       WHERE branch.sortcode = account.sortcode)
```

names of branches that have deposit accounts

```
SELECT bname
FROM branch
WHERE 'deposit' = SOME (SELECT type
                       FROM account
                       WHERE branch.sortcode = account.sortcode)
```


Worksheet: Set Operations

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) \xRightarrow{fk} account(no)

account(sortcode) \xRightarrow{fk} branch(sortcode)

Worksheet: Set Operations (3)

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

```
SELECT no
FROM account
WHERE '11-jan-1999' < ALL (SELECT tdate
                           FROM movement
                           WHERE movement.no=account.no)
```

Worksheet: Set Operations (4)

Write an SQL query that lists the `cname` of customers that have every type of account that appears in `account`

```
SELECT DISTINCT cname
FROM   account AS cust_account
WHERE  NOT EXISTS ( SELECT type
                    FROM   account
                    EXCEPT
                    SELECT type
                    FROM   account
                    WHERE  account.cname=cust_account.cname
                  )
```

Set Operations: NOT **SOME** NOT and **ALL**Equivalence between *exists* and *for all*In first order classical logic $\neg\exists\neg\equiv\forall$

accounts with all movements less than or equal to 500

```

SELECT no
FROM account
WHERE 500>=ALL (SELECT amount
                FROM movement
                WHERE account.no=movement.no)

```

Set Operations: NOT SOME NOT and ALL

Equivalence between *exists* and *for all*

In first order classical logic $\neg\exists\neg\equiv\forall$

accounts with all movements less than or equal to 500

```
SELECT no
FROM account
WHERE 500 >= ALL (SELECT amount
                  FROM movement
                  WHERE account.no = movement.no)
```

no
103
107
125

Set Operations: NOT **SOME** NOT and **ALL**Equivalence between *exists* and *for all*

In first order classical logic $\neg\exists\neg\equiv\forall$

accounts with all movements less than or equal to 500

```
SELECT no
FROM account
WHERE 500 >= ALL (SELECT amount
                  FROM movement
                  WHERE account.no = movement.no)
```

no
103
107
125

≡

```
SELECT no
FROM account
WHERE NOT 500 < SOME (SELECT amount
                     FROM movement
                     WHERE account.no = movement.no)
```

Topic 9: SQL Null Values

P.J. McBrien

Imperial College London

Null

Several definitions of null have been proposed, including:

- 1 null represents a something that is not present in the UoD

Null

Several definitions of null have been proposed, including:

- 1 null represents a something that is not present in the UoD
- 2 null represents something that might be present in the UoD, but we do not know its value at present

Null

Several definitions of null have been proposed, including:

- 1 null represents a something that is not present in the UoD
- 2 null represents something that might be present in the UoD, but we do not know its value at present
- 3 null represents something that is present in the UoD, but we do not know its value at present

Null

Several definitions of null have been proposed, including:

- 1 null represents a something that is not present in the UoD
- 2 null represents something that might be present in the UoD, but we do not know its value at present
- 3 null represents something that is present in the UoD, but we do not know its value at present

SQL handling of **NULL**

- SQL standard vague, but handling of **NULL** is nearest to option 2
- SQL uses a three valued logic to process **WHERE** predicate
- Truth values are **TRUE**, **FALSE**, and **UNKNOWN**

Quiz 9.1: SQL handling of NULL (1)

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

```
SELECT no
FROM account
WHERE rate=NULL
```

What is the result of the SQL query above?

A

no
100
101
103
107
119
125

B

no
100
103
107
125

C

no
101
119

D

no

Quiz 9.2: SQL handling of NULL (2)

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

```
SELECT no
FROM account
WHERE rate=NULL
OR rate <> NULL
```

What is the result of the SQL query above?

A

no
100
101
103
107
119
125

B

no
100
103
107
125

C

no
101
119

D

no

SQL implements three valued logic

AND

P_1 AND P_2		P_2		
	TRUE	TRUE	UNKNOWN	FALSE
P_1	UNKNOWN	TRUE	UNKNOWN	FALSE
	FALSE	UNKNOWN	UNKNOWN	FALSE
		FALSE	FALSE	FALSE

SQL implements three valued logic

AND

P_1 AND P_2		P_2		
P_1	TRUE	TRUE	UNKNOWN	FALSE
	UNKNOWN	TRUE	UNKNOWN	FALSE
	FALSE	UNKNOWN	UNKNOWN	FALSE
	FALSE	FALSE	FALSE	FALSE

OR

P_1 OR P_2		P_2		
P_1	TRUE	TRUE	UNKNOWN	FALSE
	UNKNOWN	TRUE	TRUE	TRUE
	FALSE	TRUE	UNKNOWN	UNKNOWN
	FALSE	TRUE	UNKNOWN	FALSE

SQL implements three valued logic

AND

P_1 AND P_2		P_2		
	TRUE	TRUE	UNKNOWN	FALSE
P_1	UNKNOWN	TRUE	UNKNOWN	FALSE
	FALSE	UNKNOWN	UNKNOWN	FALSE
		FALSE	FALSE	FALSE

OR

P_1 OR P_2		P_2		
	TRUE	TRUE	UNKNOWN	FALSE
P_1	UNKNOWN	TRUE	TRUE	TRUE
	FALSE	TRUE	UNKNOWN	UNKNOWN
		TRUE	UNKNOWN	FALSE

NOT

		$NOT P_1$
P_1	TRUE	FALSE
	UNKNOWN	UNKNOWN
	FALSE	TRUE

SQL implements three valued logic

AND

P_1 AND P_2		P_2		
		TRUE	UNKNOWN	FALSE
P_1	TRUE	TRUE	UNKNOWN	FALSE
	UNKNOWN	UNKNOWN	UNKNOWN	FALSE
	FALSE	FALSE	FALSE	FALSE

NOT

P_1	NOT P_1
TRUE	FALSE
UNKNOWN	UNKNOWN
FALSE	TRUE

OR

P_1 OR P_2		P_2		
		TRUE	UNKNOWN	FALSE
P_1	TRUE	TRUE	TRUE	TRUE
	UNKNOWN	TRUE	UNKNOWN	UNKNOWN
	FALSE	TRUE	UNKNOWN	FALSE

Truth values of SQL Formulae

Formula	Result
$x = \text{NULL}$	UNKNOWN
$\text{NULL} = \text{NULL}$	UNKNOWN
$x \text{ IS NULL}$	TRUE if x has a null value, FALSE otherwise
$x \text{ IS NOT NULL}$	TRUE if x does not have a null value, FALSE otherwise

Quiz 9.3: Possible answers to queries using **NULL**

```
SELECT name
FROM message
WHERE name=NULL
```

Suppose you have just connected to some database called **enigma**. Which of the following would not be a possible result of executing the query above?

A

name

B

name
NULL

C

Table **message** does not exist

D

No column **name** found in table **message**

'Correct' SQL Queries Using NULL

A query that always returns no rows

```
SELECT no
FROM account
WHERE rate=NULL
```

Find all accounts with no interest rate set

```
SELECT no
FROM account
WHERE rate IS NULL
```

Find accounts with a interest rate not equal to 5.5 or not set

```
SELECT no
FROM account
WHERE (rate=5.5) IS NOT TRUE
```

Quiz 9.4: SQL 'Might Be'

```
SELECT no
FROM account
WHERE (rate=5.25) IS NOT FALSE
```

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

What is the result of the above SQL query?

A

no
100
101
103
107
125

B

no
100
103
107
119
125

C

no
100
103
107
125

D

no

Worksheet: NULL values in SQL

movement			
<u>mid</u>	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				
<u>no</u>	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

Quiz 9.5: SQL EXCEPT and NULL

```

SELECT rate
FROM account
WHERE no < 105
EXCEPT
SELECT rate
FROM account
WHERE sortcode = 56

```

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

What is the result of the above SQL query?

A

rate

B

rate
5.25

C

rate
5.25
NULL

D

rate
5.25
NULL
NULL

Equivalences Between EXCEPT, NOT IN and NOT EXISTS

$R(\underline{A})$ and $S(B)$, A and B are not nullable

<pre>SELECT A FROM R EXCEPT SELECT B FROM S</pre>	\equiv	<pre>SELECT A FROM R WHERE NOT EXISTS (SELECT * FROM S WHERE S.B=R.A)</pre>	\equiv	<pre>SELECT A FROM R WHERE A NOT IN (SELECT B FROM S)</pre>
---	----------	---	----------	--

$R(A)$ and $S(B)$, A or B are nullable

<pre>SELECT A FROM R EXCEPT SELECT B FROM S</pre>	\neq	<pre>SELECT A FROM R WHERE NOT EXISTS (SELECT * FROM S WHERE S.B=R.A)</pre>	\neq	<pre>SELECT A FROM R WHERE A NOT IN (SELECT B FROM S)</pre>
---	--------	---	--------	--

Quiz 9.6: SQL **EXCEPT** and **NOT IN**

```

SELECT rate
FROM account
WHERE no < 105
AND rate NOT IN
      (SELECT rate
       FROM account
       WHERE sortcode = 56)

```

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

What is the result of the above SQL query?

A

rate

B

rate
5.25

C

rate
5.25
NULL

D

rate
5.25
NULL
NULL

Quiz 9.7: SQL **EXCEPT** and **NOT EXISTS**

```

SELECT rate
FROM account
WHERE no < 105
AND NOT EXISTS
  (SELECT *
   FROM account AS account_56
   WHERE sortcode = 56
   AND account_56.rate = account.rate )

```

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

What is the result of the above SQL query?

A

rate

B

rate
5.25

C

rate
5.25
NULL

D

rate
5.25
NULL
NULL