



**COMMITTED TO ICT.
INVOLVED IN PEOPLE.**

Patrick.Barel@AMIS.nl
technology.amis.nl/blog
blog.bar-solutions.com

COMMITTED TO ICT. INVOLVED IN PEOPLE.





PUZZELLEN MET SQL SET





THE RULES - FEATURES

➤ Color



➤ Symbol



➤ Number



➤ Shading





THE RULES

A 'Set' consists of three cards in which each feature is EITHER the same on each card OR is different on each card. That is to say, any feature in the 'Set' of three cards is either common to all three cards or is different on each card.



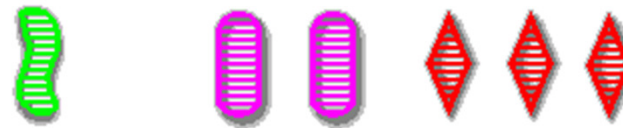
EXAMPLES

Correct



Feature	Result
Color	Same
Number	Same
Shape	Same
Shading	Different

Feature	Result
Color	Different
Number	Different
Shape	Different
Shading	Same



Feature	Result
Color	Different
Number	Different
Shape	Different
Shading	Different



EXAMPLES

Incorrect



Feature	Result
Color	Different
Number	Same
Shape	Same
Shading	2 out of 3

Feature	Result
Color	2 out of 3
Number	Same
Shape	Same
Shading	Different





START WITH EXAMPLES

Create a puzzle (taken from the SET box)

```
with cards as
(select 1 cardno, 'oval' symbol, 'red' color, 'open' shading, 2 qty from dual union all
select 2,'squiggle', 'purple', 'solid', 1 from dual union all
select 3,'diamond', 'red', 'solid', 3 from dual union all
select 4,'squiggle', 'green', 'solid',3 from dual union all
select 5,'diamond', 'red', 'striped', 1 from dual union all
select 6,'oval', 'purple', 'striped', 1 from dual union all
select 7,'oval', 'red', 'open',1 from dual union all
select 8,'oval', 'purple', 'open',1 from dual union all
select 9,'oval', 'green', 'solid',2 from dual union all
select 10,'oval', 'purple', 'solid',1 from dual union all
select 11,'oval', 'red', 'open',3 from dual union all
select 12,'diamond', 'purple', 'striped',1 from dual
)
select *
  from cards
```



ADD THE PREDICATE(S)

Rewrite the rule into a SQL predicate

```
(    [feature card 1] = [feature card 2]
  and [feature card 2] = [feature card 3])
or (    [feature card 1] <> [feature card 2]
  and [feature card 1] <> [feature card 3]
  and [feature card 2] <> [feature card 3])
```

Notice the rule [feature card 1] = [feature card 3] is missing.



CONVERT PSEUDO TO REAL CODE

4 features so 4 parts in the predicates

```
where ( ( cards1_symbol = cards2_symbol  
        and cards1_symbol = cards3_symbol)  
        or ( cards1_symbol <> cards2_symbol  
            and cards2_symbol <> cards3_symbol  
            and cards1_symbol <> cards3_symbol))
```

```
and ( ( cards1_color = cards2_color  
       and cards1_color = cards3_color)  
      or ( cards1_color <> cards2_color  
          and cards2_color <> cards3_color  
          and cards1_color <> cards3_color))
```

```
and ( ( cards1_shading = cards2_shading  
       and cards1_shading = cards3_shading)  
      or ( cards1_shading <> cards2_shading  
          and cards2_shading <> cards3_shading  
          and cards1_shading <> cards3_shading))
```

```
and ( ( cards1_qty = cards2_qty  
       and cards1_qty = cards3_qty)  
      or ( cards1_qty <> cards2_qty  
          and cards2_qty <> cards3_qty  
          and cards1_qty <> cards3_qty))
```



SOLVING THE PUZZLE

Steps to take:

- Build the deck
- Select the cards
- Solve the puzzle



BUILD THE DECK

Define the features
using subquery
factoring.

No tables needed

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
)
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
)
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
)
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
)
)
```

4) numbers.sql

3) shadings.sql

2) symbols.sql

1) colors.sql



BUILD THE DECK

Combine the features using a cartesian product.

This is NOT an error, this is intended.

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
)
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
)
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
)
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
)
, all_cards as
(select *
  from colors
    , symbols
    , shadings
    , numbers
)
```



BUILD THE DECK

Combine the features using a cartesian product.

This is NOT an error, this is intended.

Using ANSI SQL this is more verbose.

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
)
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
)
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
)
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
)
, all_cards as
(select *
  from colors
      cross join symbols
      cross join shadings
      cross join numbers
)
```



SELECT THE CARDS

Using a simple query will 'always' select the same cards.

```
, cards as
(select rownum cardno
 , color
 , symbol
 , shading
 , qty
 from (select color
 , symbol
 , shading
 , qty
 from all_cards
 )
 where rownum <= 12
)
```



SELECT THE CARDS

Using a simple query will 'always' select the same cards.

We need to select random cards.

```
, cards as
(select rownum cardno
, color
, symbol
, shading
, qty
from (select color
, symbol
, shading
, qty
from all_cards
order by dbms_random.value
)
where rownum <= 12
)
```



SELECT THE CARDS

Display the selection we just made.

```
select cards.cardno cards1_cardno
      , cards.symbol cards1_symbol
      , cards.color cards1_color
      , cards.shading cards1_shading
      , cards.qty cards1_qty
      , null cards2_cardno
      , null cards2_symbol
      , null cards2_color
      , null cards2_shading
      , null cards2_qty
      , null cards3_cardno
      , null cards3_symbol
      , null cards3_color
      , null cards3_shading
      , null cards3_qty
      , 'puzzle' what
from cards
```

Needed for the
union all with
the solution

the solution
union all with
needed for the



SOLVE THE PUZZLE

We need to select three cards from the puzzle.

Can't select same card twice

```
select *
  from (select cards1.cardno cards1_cardno
        , cards1.symbol cards1_symbol
        , cards1.color cards1_color
        , cards1.shading cards1_shading
        , cards1.qty cards1_qty
        , cards2.cardno cards2_cardno
        , cards2.symbol cards2_symbol
        , cards2.color cards2_color
        , cards2.shading cards2_shading
        , cards2.qty cards2_qty
        , cards3.cardno cards3_cardno
        , cards3.symbol cards3_symbol
        , cards3.color cards3_color
        , cards3.shading cards3_shading
        , cards3.qty cards3_qty
        , 'solution' what
  from cards cards1
  join cards cards2 on (cards1.cardno < cards2.cardno)
  join cards cards3 on (cards2.cardno < cards3.cardno)
)
```



SOLVE THE PUZZLE

Apply the rules to the selected cards

```
where ((cards1_symbol = cards2_symbol and cards1_symbol = cards3_symbol)
      or
      (cards1_symbol <> cards2_symbol
       and cards2_symbol <> cards3_symbol
       and cards1_symbol <> cards3_symbol
      )
     )
```

```
and ((cards1_color = cards2_color and cards1_color = cards3_color)
     or
     (cards1_color <> cards2_color
      and cards2_color <> cards3_color
      and cards1_color <> cards3_color
     )
    )
```

```
and ((cards1_shading = cards2_shading and cards1_shading = cards3_shading)
     or
     (cards1_shading <> cards2_shading
      and cards2_shading <> cards3_shading
      and cards1_shading <> cards3_shading
     )
    )
```

```
and ((cards1_qty = cards2_qty and cards1_qty = cards3_qty)
     or
     (cards1_qty <> cards2_qty
      and cards2_qty <> cards3_qty
      and cards1_qty <> cards3_qty
     )
    )
```



SOLVE THE PUZZLE

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
)
, symbols as
(select 'squiggle' symbol from dual
union all
 select 'oval' from dual union all
 select 'diamond' from dual
)
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
)
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
)
, all_cards as
(select *
 from colors
   , symbols
   , shadings
   , numbers
)
```

Combine all the elements

➤ Create the deck



SOLVE THE PUZZLE

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
 )
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
 )
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
 )
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
 )
, all_cards as
(select *
 from colors
 , symbols
 , shadings
 , numbers
 )
, cards as
(select rownum cardno
 , color
 , symbol
 , shading
 , qty
 from (select color
 , symbol
 , shading
 , qty
 from all_cards
 order by dbms_random.value
 )
 where rownum <= 12
 )
```

Combine all the elements

- Create the deck
- Select the cards



SOLVE THE PUZZLE

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
 )
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
 )
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
 )
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
 )
, all_cards as
(select *
 from colors
 , symbols
 , shadings
 , numbers
 )
, cards as
(select rownum cardno
 , color
 , symbol
 , shading
 , qty
 from (select color
 , symbol
 , shading
 , qty
 from all_cards
 order by dbms_random.value
 )
 where rownum <= 12
 )
```

Combine all the elements

- Create the deck
- Select the cards
- Display the puzzle

```
select cards.cardno cards1_cardno
 , cards.symbol cards1_symbol
 , cards.color cards1_color
 , cards.shading cards1_shading
 , cards.qty cards1_qty
 , null cards2_cardno
 , null cards2_symbol
 , null cards2_color
 , null cards2_shading
 , null cards2_qty
 , null cards3_cardno
 , null cards3_symbol
 , null cards3_color
 , null cards3_shading
 , null cards3_qty
 , 'puzzle' what
from cards
```



SOLVE THE PUZZLE

```
with colors
as
(select 'red' color from dual union all
 select 'green' from dual union all
 select 'purple' from dual
)
, symbols as
(select 'squiggle' symbol from dual union all
 select 'oval' from dual union all
 select 'diamond' from dual
)
, numbers as
(select 1 qty from dual union all
 select 2 from dual union all
 select 3 from dual
)
, shadings as
(select 'solid' from dual union all
 select 'outline' from dual union all
 select 'striped' from dual
)
, all_cards as
(select *
 from colors
 , symbols
 , shadings
 , numbers
)
, cards as
(select rownum cardno
 , color
 , symbol
 , shading
 , qty
 from (select color
 , symbol
 , shading
 , qty
 from all_cards
 order by dbms_random.value
)
where rownum <= 12
)
```

```
select cards.cardno cards1_cardno
 , cards.symbol cards1_symbol
 , cards.color cards1_color
 , cards.shading cards1_shading
 , cards.qty cards1_qty
 , null cards2_cardno
 , null cards2_symbol
 , null cards2_color
 , null cards2_shading
 , null cards2_qty
 , null cards3_cardno
 , null cards3_symbol
 , null cards3_color
 , null cards3_shading
 , null cards3_qty
 , 'puzzle' what
 from cards
 union all
```

Combine all the elements

- Create the deck
- Select the cards
- Display the puzzle
- Find the solution(s)

```
select *
 from ( ... )
 where ( ... )
 order by what
 , cards1_cardno
 , cards2_cardno
 , cards3_cardno
 ;
```



SELECT SOLUTION

```
select cards1.cardno cards1_cardno
      , cards1.symbol cards1_symbol
      , cards1.color cards1_color
      , cards1.shading cards1_shading
      , cards1.qty cards1_qty
      , cards2.cardno cards2_cardno
      , cards2.symbol cards2_symbol
      , cards2.color cards2_color
      , cards2.shading cards2_shading
      , cards2.qty cards2_qty
      , cards3.cardno cards3_cardno
      , cards3.symbol cards3_symbol
      , cards3.color cards3_color
      , cards3.shading cards3_shading
      , cards3.qty cards3_qty
      , 'solution' what
from cards cards1
join cards cards2 on (cards1.cardno < cards2.cardno)
join cards cards3 on (cards2.cardno < cards3.cardno)
```

Back



PREDICATES

```
(cards1_symbol = cards2_symbol and cards1_symbol = cards3_symbol)
  or
  (cards1_symbol <> cards2_symbol
   and cards2_symbol <> cards3_symbol
   and cards1_symbol <> cards3_symbol
  )
)
and ((cards1_color = cards2_color and cards1_color = cards3_color)
  or
  (cards1_color <> cards2_color
   and cards2_color <> cards3_color
   and cards1_color <> cards3_color
  )
)
and ((cards1_shading = cards2_shading and cards1_shading = cards3_shading)
  or
  (cards1_shading <> cards2_shading
   and cards2_shading <> cards3_shading
   and cards1_shading <> cards3_shading
  )
)
and ((cards1_qty = cards2_qty and cards1_qty = cards3_qty)
  or
  (cards1_qty <> cards2_qty
   and cards2_qty <> cards3_qty
   and cards1_qty <> cards3_qty
  )
)
```

Back



SOLVE THE PUZZLE IN SQL

Demo in PL/SQL Developer



SOLVE THE PUZZLE IN SQL

Demo in APEX with raffle

Demo in APEX

The image features a large, light gray 'FIMIS' logo in the background. A white swoosh underline is positioned behind the 'Q&A' text. The 'Q&A' text is rendered in a bold, red, sans-serif font with a slight drop shadow.

Q&A

COMMITTED TO ICT.
INVOLVED IN PEOPLE.



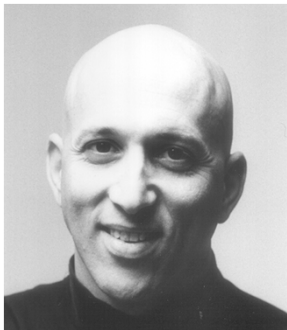
TRAININGEN EN MASTERCLASSES



8 en 9 december 2011

2 dagen Seminar van Steven Feuerstein,

Best of Oracle PL/SQL, Best Practices and new features



- Leer van zijn samenwerking met het PL/SQL product team
- Veel interactie en mogelijkheden tot het stellen van vragen

- inclusief het boek Oracle PL/SQL Programming (5th)
- inclusief 27 uur video training

Schrijf je in via www.amis.nl



COMMITTED TO ICT. INVOLVED IN PEOPLE.