



# SQL language: basics

## Nested queries

# Nested queries

- Introduction
- The IN operator
- The NOT IN operator
- The tuple constructor
- The EXISTS operator
- The NOT EXISTS operator
- Correlation among queries
- The division operation
- Table functions



# Nested queries

## Introduction

- A nested query is a **SELECT** statement contained within another query
  - query nesting allows decomposing a complex problem into simpler subproblems
- **SELECT** statements may be introduced
  - within a predicate in the **WHERE** clause
  - within a predicate in the **HAVING** clause
  - in the **FROM** clause

## Supplier and part DB (1/2)

- P (PId, PName, Color, Size, Store)
- S (SId, SName, #Employees, City)
- SP (SId, PId, Qty)



# Supplier and Product database

P

<u>PId</u>	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

Foreign keys

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400


## Nested queries (no.1)

- Find the codes of the suppliers that are based in the same city as S1
  
- By using a formulation with nested queries, the problem may be decomposed into two subproblems
  - city of supplier S1
  - codes of the suppliers based in the same city

## Nested queries (no.1)

- Find the codes of the suppliers that are based in the same city as S1

```
SELECT SId
FROM S
WHERE City = (SELECT City
              FROM S
              WHERE SId='S1');
```



- The '=' operator may be used only if it is known in advance that the inner SELECT statement always returns a single value



## Equivalent formulation (no.1)

- Find the codes of the suppliers that are based in the same city as S1
- An equivalent formulation may be defined using a join operation

# Equivalent formulation

- The equivalent formulation with join is characterized by
- a FROM clause including the tables referenced by the FROM clauses of each SELECT statement
  - appropriate join conditions in the WHERE clause
  - possible selection predicates added in the WHERE clause

## FROM clause (no.1)

➤ Find the codes of the suppliers that are based in the same city as S1

```
SELECT SId
FROM (S) ← SX
WHERE City = (SELECT City
              FROM (S) ← SY
              WHERE SId='S1');
```

## FROM clause (no.1)

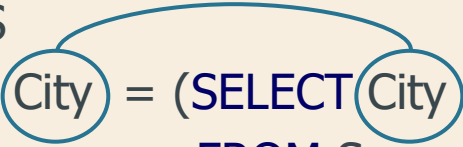
- Find the codes of the suppliers that are based in the same city as S1

```
SELECT ...  
FROM S AS SX, S AS SY  
...
```

## Join condition (no.1)

- Find the codes of the suppliers that are based in the same city as S1

```
SELECT SId
FROM S
WHERE City = (SELECT City
               FROM S
               WHERE SId='S1');
```

A blue curved line connects the 'City' column in the outer query's WHERE clause to the 'City' column in the subquery's WHERE clause, indicating the join condition.

## Join condition (no.1)

- Find the codes of the suppliers that are based in the same city as S1

```
SELECT ...  
FROM S AS SX, S AS SY  
WHERE SX.City=SY.City  
...
```



## Selection predicate (no.1)

- Find the codes of the suppliers that are based in the same city as S1

```
SELECT SId
FROM S
WHERE City = (SELECT City
              FROM S
              WHERE SId='S1');
```

## SELECT clause (no.1)

➤ Find the codes of the suppliers that are based in the same city as S1

```
SELECT SY.SId
FROM S AS SX, S AS SY
WHERE SX.City=SY.City AND
      SX.SId='S1';
```

## Equivalent formulation (no.2)

- Find the codes of the suppliers whose number of employees is smaller than the maximum number of employees

```
SELECT SId
FROM S
WHERE #Employees < (SELECT MAX(#Employees)
                     FROM S);
```

- Is it possible to define an equivalent formulation with join?

## Equivalent formulation (no.2)

- Find the codes of the suppliers whose number of employees is smaller than the maximum number of employees

```
SELECT SId
FROM S
WHERE #Employees < (SELECT MAX(#Employees)
                     FROM S);
```

- An equivalent formulation with join is not possible



# Nested queries

## The IN operator

## The IN operator (no.1)

- Find the names of the suppliers that provide product P2
  
- Decomposition of the problem into two subproblems
  - codes of the suppliers of product P2
  - names of the suppliers with such codes



# The IN operator (no.1)

➤ Find the names of the suppliers that provide product P2

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400



<u>SId</u>
S1
S2
S3

```
(SELECT SId  
FROM SP  
WHERE PId='P2')
```

*Codes  
of the  
suppliers  
of P2*


## The IN operator (no.1)

➤ Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId='P2')
```

# The IN operator (no.1)

➤ Find the names of the suppliers that provide product P2


```
SELECT SName
FROM S
WHERE SId  (SELECT SId
      FROM SP
      WHERE PId='P2')
```

# The IN operator (no.1)

➤ Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId='P2');
```

*Set membership*



# The IN operator

- It expresses the concept of membership to a set of values
  - *AttributeName* IN (*NestedQuery*)
- It allows writing a query by
  - decomposing the problem into subproblems
  - following a “bottom-up” procedure

# Equivalent formulation

- The equivalent formulation with join is characterized by
- a FROM clause including the tables referenced by the FROM clauses of each SELECT statement
  - appropriate join conditions in the WHERE clause
  - possible selection predicates added in the WHERE clause



## The IN operator (no.1)

➤ Find the names of the suppliers that provide product P2

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId='P2');
```

## Equivalent formulation (no.1)

➤ Find the names of the suppliers that provide product P2

```
SELECT SName  
FROM S, SP  
WHERE S.SId=SP.SId  
      AND PId='P2';
```

## The IN operator (no.2)

- Find the names of the suppliers that supply at least one red product
  
- Decomposition of the problem into subproblems
  - codes of the red products
  - codes of the suppliers of such products
  - names of the suppliers with such codes

## The IN operator (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId IN (SELECT PId
                            FROM P
                            WHERE Color='Red'));
```

## Equivalent formulation (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId IN (SELECT PId
                           FROM P
                           WHERE Color='Red'));
```

## FROM clause (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM (S)
WHERE SId IN (SELECT SId
              FROM (SP)
              WHERE PId IN (SELECT PId
                            FROM (P)
                            WHERE Color='Red')));
```

## FROM clause (no.2)

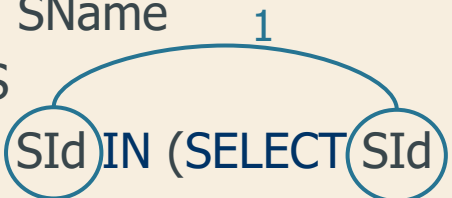
➤ Find the names of the suppliers that supply at least one red product

```
SELECT ...  
FROM S, SP, P  
...
```

## Join conditions (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM S
WHERE (Sid) IN (SELECT (Sid)
                FROM SP
                WHERE PId IN (SELECT PId
                              FROM P
                              WHERE Color='Red')));
```





## Join conditions (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT ...  
FROM S, SP, P  
WHERE S.SId=SP.SId
```

1

## Join conditions (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
```

```
FROM S
```

```
WHERE SId IN (SELECT SId
```

```
FROM SP
```

```
WHERE PId IN (SELECT PId
```

```
FROM P
```

```
WHERE Color='Red'));
```

2



## Join conditions (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT ...  
FROM S, SP, P  
WHERE S.SId=SP.SId AND  
       SP.CodP=S.CodP  
...
```

2

## Selection predicate (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId IN (SELECT PId
                           FROM P
                           WHERE Color='Red'));
```

## SELECT clause (no.2)

➤ Find the names of the suppliers that supply at least one red product

```
SELECT DISTINCT SName  
FROM S, SP, P  
WHERE S.SId=SP.SId AND  
       SP.PId=P.PId AND  
       Color='Red';
```

## A complex example (no.3)

➤ Find the names of the suppliers that supply at least one product supplied by suppliers of red products

P

<u>PId</u>	PName	Color	Size	Store
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

## A complex example (no.3)

- Find the names of the suppliers that supply at least one product supplied by suppliers of red products



## A complex example (no.3)

- Find the names of the suppliers that supply at least one product supplied by suppliers of red products
- The formulation with join is awkward
  - it is easier to decompose the problem into subproblems through nested queries



## A complex example (no.3)

- Find the names of the suppliers that supply at least one product supplied by suppliers of red products

*Codes of the  
red products*

```
SELECT PId  
FROM P  
WHERE Color='Red'
```

## A complex example (no.3)

- Find the names of the suppliers that supply at least one product supplied by suppliers of red products

```
SELECT SId
FROM SP
WHERE PId IN
    (SELECT PId
     FROM P
     WHERE Color='Red')
```

*Codes of the suppliers  
of red products*

## A complex example (no.3)

➤ Find the names of the suppliers that supply at least one product supplied by suppliers of red products

*Codes of the products supplied by suppliers of red products*

```
SELECT PId
FROM SP
WHERE SId IN
    (SELECT SId
     FROM SP
     WHERE PId IN
         (SELECT PId
          FROM P
          WHERE Color='Red'))
```

## A complex example (no.3)

```
SELECT SId
FROM SP
WHERE PId IN
    (SELECT PId
     FROM SP
     WHERE SId IN
         (SELECT SId
          FROM SP
          WHERE PId IN
              (SELECT PId
               FROM P
               WHERE Color='Red'))))
```

*Codes of the suppliers  
of the products  
supplied by suppliers  
of red products*

## Complete query (no.3)

```
SELECT SName
FROM S
WHERE SId IN
    (SELECT SId
     FROM SP
     WHERE PId IN
         (SELECT PId
          FROM SP
          WHERE SId IN
              (SELECT SId
               FROM SP
               WHERE PId IN
                   (SELECT PId
                    FROM P
                    WHERE Color='Red'))));
```

# Formulation with join (no.3)



## Formulation with join (no.3)

```
SELECT SName
FROM S
WHERE SId IN
    (SELECT SId
     FROM SP
     WHERE PId IN
         (SELECT PId
          FROM SP
          WHERE SId IN
              (SELECT SId
               FROM SP
               WHERE PId IN
                   (SELECT PId
                    FROM P
                    WHERE Color='Red'))));
```

## FROM clause (no.3)

```
SELECT SName
FROM (S)
WHERE SId IN
    (SELECT SId
     FROM (SP)
     WHERE PId IN
         (SELECT PId
          FROM (SP)
          WHERE SId IN
              (SELECT SId
               FROM (SP)
               WHERE PId IN
                   (SELECT PId
                    FROM (P)
                    WHERE Color='Red'))));
```



# FROM clause (no.3)

```
SELECT SName
FROM (S)
WHERE SId IN
  (SELECT SId
   FROM (SP)
   WHERE PId IN
     (SELECT PId
      FROM (SP)
      WHERE SId IN
        (SELECT SId
         FROM (SP)
         WHERE PId IN
           (SELECT PId
            FROM (P)
            WHERE Color='Red'))));
```

SPA

SPB

SPC

The diagram illustrates a nested SQL query structure. The main query is 'SELECT SName FROM (S) WHERE SId IN'. It contains three nested subqueries, each starting with 'SELECT' and ending with 'WHERE'. The first subquery is '(SELECT SId FROM (SP) WHERE PId IN)'. The second subquery is '(SELECT PId FROM (SP) WHERE SId IN)'. The third subquery is '(SELECT SId FROM (SP) WHERE PId IN)'. The fourth subquery is '(SELECT PId FROM (P) WHERE Color='Red')'. Red circles are drawn around the 'S', 'SP', 'SP', 'SP', and 'P' in the FROM clauses of the subqueries. Red arrows point from the labels SPA, SPB, and SPC to these circles respectively.

## FROM clause (no.3)

SELECT ...

FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P

...

## Join conditions (no.3)

```
SELECT SName      1
FROM S
WHERE SId IN
  (SELECT SId
   FROM SP
   WHERE PId IN
     (SELECT PId
      FROM SP
      WHERE SId IN
        (SELECT SId
         FROM SP
         WHERE PId IN
           (SELECT PId
            FROM P
            WHERE Color='Red'))));
```

The diagram illustrates the join conditions in the provided SQL query. It highlights the following elements:

- The **SId** attribute in the main query's **WHERE** clause is circled in blue.
- The **SId** attribute in the subquery's **WHERE** clause is circled in blue.
- The **SP** table name in the subquery's **FROM** clause is circled in red.
- The **SP** table name in the main query's **WHERE** clause is circled in red.
- A red arrow labeled **SPA** points from the circled **SP** in the main query to the circled **SP** in the subquery.
- A blue arc labeled **1** connects the circled **SId** in the main query to the circled **SId** in the subquery.

## Join conditions (no.3)

```
SELECT ...  
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P  
WHERE S.SId=SPA.SId    ①  
...  
...
```

## Join conditions (no.3)

```
SELECT SName  
FROM S  
WHERE SId IN
```

```
(SELECT SId  
FROM SP  
WHERE PId IN  
(SELECT PId  
FROM SP  
WHERE SId IN  
(SELECT SId  
FROM SP  
WHERE PId IN  
(SELECT PId  
FROM P  
WHERE Color='Red')))))));
```

The diagram illustrates the join conditions in the SQL query. It shows a sequence of nested subqueries. The first subquery selects SId from SP, where PId is in a subquery that selects PId from SP, where SId is in a subquery that selects SId from SP, where PId is in a subquery that selects PId from P, where Color is 'Red'. Red arrows labeled SPA and SPB point to the SP tables in the first and second subqueries, respectively. A blue arc labeled '2' connects the two SP tables, indicating a join condition. Blue circles highlight the PId columns in the subqueries.

## Join conditions (no.3)

```
SELECT ...  
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P  
WHERE S.SId=SPA.SId AND  
      SPA.PId=SPB.PId  
      ...
```

2

## Join conditions (no.3)

```
SELECT SName  
FROM S  
WHERE SId IN
```

```
(SELECT SId  
FROM SP
```

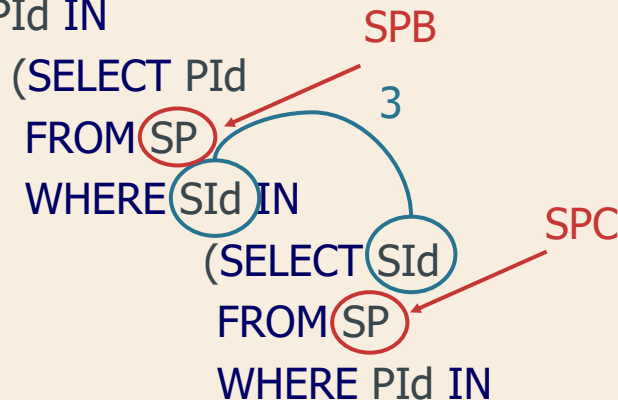
```
WHERE PId IN
```

```
(SELECT PId  
FROM SP  
WHERE SId IN
```

```
(SELECT SId  
FROM SP  
WHERE PId IN
```

```
(SELECT PId  
FROM P
```

```
WHERE Color='Red'))));
```



## Join conditions (no.3)

```
SELECT ...  
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P  
WHERE S.SId=SPA.SId AND  
      SPA.PId=SPB.PId AND  
      SPB.SId=SPC.SId  
...
```

3



## Join conditions (no.3)

```
SELECT SName
FROM S
WHERE SId IN
    (SELECT SId
     FROM SP
     WHERE PId IN
         (SELECT PId
          FROM SP
          WHERE SId IN
              (SELECT SId
               FROM SP
               WHERE PId IN
                   (SELECT PId
                    FROM P
                    WHERE Color='Red')))))));
```

Diagram illustrating join conditions in a SQL query. The query is shown with annotations: a red arrow points from 'SPC' to the 'SP' table in the innermost subquery, and a blue arrow points from '4' to the same 'SP' table. The 'SP' table in the innermost subquery and the 'PId' table in the next level are circled in blue. A blue arc connects the 'PId' table in the next level to the 'SP' table in the innermost subquery, indicating a join condition.

## Join conditions (no.3)

```
SELECT ...  
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P  
WHERE S.SId=SPA.SId AND  
      SPA.PId=SPB.PId AND  
      SPB.SId=SPC.SId AND  
      SPC.PId=P.PId  
...
```

4

## Selection predicate (no.3)

```
SELECT SName
FROM S
WHERE SId IN
    (SELECT SId
     FROM SP
     WHERE PId IN
         (SELECT PId
          FROM SP
          WHERE SId IN
              (SELECT SId
               FROM SP
               WHERE PId IN
                   (SELECT PId
                    FROM P
                    WHERE Color='Red'))));
```

## Selection predicate (no.3)

```
SELECT ...  
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P  
WHERE S.SId=SPA.SId AND  
       SPA.PId=SPB.PId AND  
       SPB.SId=SPC.SId AND  
       SPC.PId=P.PId AND  
       Color='Red'
```

## SELECT statement (no.3)

```
SELECT DISTINCT SName
FROM S, SP AS SPA, SP AS SPB, SP AS SPC, P
WHERE S.SId=SPA.SId AND
      SPA.PId=SPB.PId AND
      SPB.SId=SPC.SId AND
      SPC.PId=P.PId AND
      Color='Red';
```



## Nested queries

### The NOT IN operator

## Concept of exclusion (no.1)

- Find the names of the suppliers that *do not* supply product P2
- is it possible to express the query with a join operation?

```
SELECT SName
FROM S, SP
WHERE S.SId=SP.SId
      AND PId<>'P2';
```

## Wrong solution (no.1)

- Find the names of the suppliers that *do not* supply product P2
- the query may not be expressed by means of a join

```
SELECT SName  
FROM S, SP  
WHERE S.SId=SP.SId  
AND PId<>'P2';
```




# Wrong solution (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

R



SName
Smith
Jones
Clark

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

## Wrong solution (no.1)

```
SELECT SName  
FROM S, SP  
WHERE S.SId=SP.SId  
AND PId<>'P2';
```

➤ Which query does it answer?

## Wrong solution (no.1)

```
SELECT SName
FROM S, SP
WHERE S.SId=SP.SId
      AND PId<>'P2';
```



Find the names of the suppliers that supply  
at least one product other than P2

## Concept of exclusion (no.1)

- Find the names of the suppliers that *do not* supply product P2
- We need to exclude from the result
  - the suppliers that supply product P2

# Concept of exclusion (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

```
SELECT SId  
FROM SP  
WHERE PId='P2'
```


*Codes of the suppliers  
that supply P2*

# Concept of exclusion (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

```
SELECT SName  
FROM S  
WHERE SId
```

```
(SELECT SId  
FROM SP  
WHERE PId='P2');
```



*Codes of the suppliers  
that supply P2*

# The NOT IN operator (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

```
SELECT SName
FROM S
WHERE SId NOT IN (SELECT SId
                  FROM SP
                  WHERE PId='P2');
```

*does not belong to*

*Codes of the suppliers  
that supply P2*

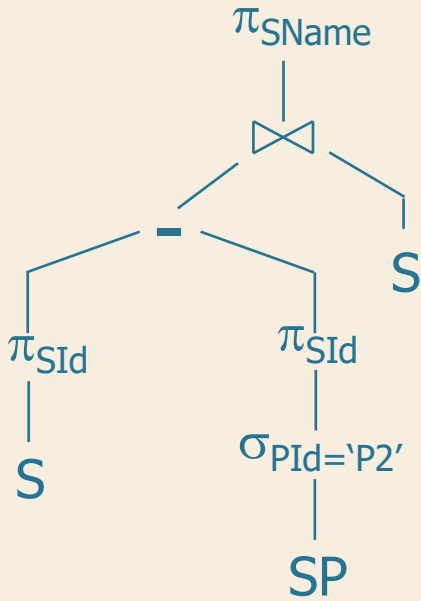
# The NOT IN operator

- It expresses the concept of exclusion from a set of values
  - *AttributeName* NOT IN (*NestedQuery*)
- It requires the identification of an appropriate *set to be excluded*
  - defined by the nested query



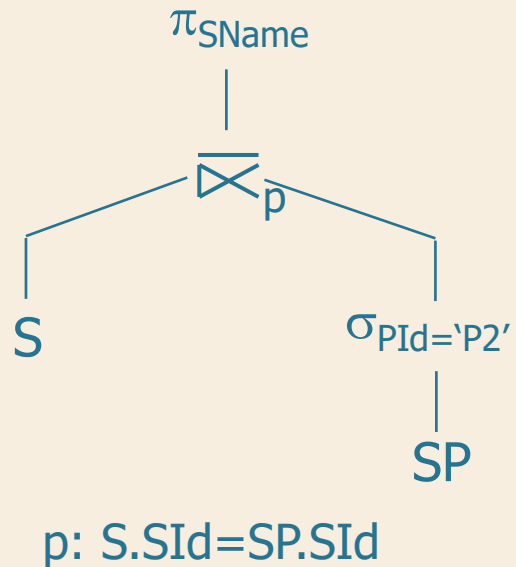
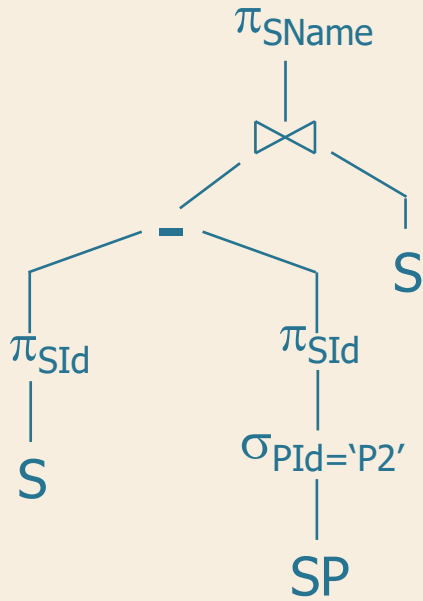
# NOT IN and relational algebra (no.1)

➤ Find the names of the suppliers that *do not* supply product P2



# NOT IN and relational algebra (no.1)

➤ Find the names of the suppliers that *do not* supply product P2



## The NOT IN operator (no.2)

- Find the names of the suppliers that *only* supply product P2



*Find the names of the suppliers of P2 that have never supplied products other than P2*

- Set to be excluded
  - suppliers of products other than P2

## The NOT IN operator (no.2)

➤ Find the names of the suppliers that only supply product P2

```
SELECT SId  
FROM SP  
WHERE PId <> 'P2'
```

*Codes of the suppliers  
that supply  
at least one product  
other than P2*

## The NOT IN operator (no.2)

- Find the names of the suppliers that only supply product P2

```
SELECT SName
FROM S
WHERE SId NOT IN (SELECT SId
                  FROM SP
                  WHERE PId<>'P2')
```

...

## The NOT IN operator (no.2)

- Find the names of the suppliers that only supply product P2

```
SELECT SName
FROM S, SP
WHERE S.SId NOT IN (SELECT SId
                    FROM SP
                    WHERE PId<>'P2')
AND S.SId=SP.SId;
```

## Alternative solution (no.2)

- Find the names of the suppliers that only supply product P2

```
SELECT SName
FROM S
WHERE S.SId NOT IN (SELECT SId
                    FROM SP
                    WHERE PId<>'P2')
AND S.SId IN (SELECT SId
              FROM SP);
```

## The NOT IN operator (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

➤ P (PId, PName, Color, Size, Store)

➤ S (SId, SName, #Employees, City)

➤ SP (SId, PId, Qty)



## The NOT IN operator (no.3)

- Find the names of the suppliers that *do not* supply any red products
- Set to be excluded?
  - suppliers of red products, identified by their codes

## The NOT IN operator (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

*Codes of the suppliers  
of red products*

```
(SELECT SId
FROM SP
WHERE PId IN (SELECT PId
FROM P
WHERE Color='Red'))
```

## The NOT IN operator (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

```
SELECT SName
FROM S
WHERE SId NOT IN (SELECT SId
                  FROM SP
                  WHERE PId IN (SELECT PId
                               FROM P
                               WHERE Color='Red'));
```

## Alternative (correct?) (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

*Codes of the suppliers that supply at least one non-red product*

```
SELECT SId
FROM SP
WHERE PId NOT IN (SELECT PId
                  FROM P
                  WHERE Color='Red')
```

## Alternative (correct?) (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId NOT IN (SELECT PId
                                FROM P
                                WHERE Color='Red'));
```

## Wrong alternative (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

```
SELECT SName  
FROM S  
WHERE SId IN (SELECT SId  
              FROM SP  
              WHERE PId NOT IN (SELECT PId  
                                FROM P  
                                WHERE Color='Red')));
```

## Wrong alternative (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

```
SELECT SName
FROM S
WHERE SId IN (SELECT SId
              FROM SP
              WHERE PId NOT IN (SELECT PId
                                FROM P
                                WHERE Color='Red'));
```

*Codes of the suppliers of non-red products*

## Wrong alternative (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

P

<u>PId</u>	<u>PName</u>	<u>Color</u>	<u>Size</u>	<u>Store</u>
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

S

<u>SId</u>	<u>SName</u>	<u>#Employees</u>	<u>City</u>
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	<u>Qty</u>
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400



## Wrong alternative (no.3)

➤ Find the names of the suppliers that *do not* supply any red products

P

<u>PId</u>	<u>PName</u>	<u>Color</u>	<u>Size</u>	<u>Store</u>
P1	Jumper	Red	40	London
P2	Jeans	Green	48	Paris
P3	Blouse	Blue	48	Rome
P4	Blouse	Red	44	London
P5	Skirt	Blue	40	Paris
P6	Shorts	Red	42	London

S

<u>SId</u>	<u>SName</u>	<u>#Employees</u>	<u>City</u>
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	<u>Qty</u>
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

## Wrong alternative (no.3)

- Find the names of the suppliers that *do not* supply any red products

```
SELECT SName  
FROM S  
WHERE SId IN (SELECT SId  
              FROM SP  
              WHERE PId NOT IN (SELECT PId  
                                FROM P  
                                WHERE Color='Red')));
```

- The set of elements to be excluded is incorrect



## Nested queries

### The tuple constructor

# The tuple constructor

- It allows defining a temporary structure for a tuple
  - the attributes belonging to the tuple must be listed within ()

*(AttributeName<sub>1</sub>, AttributeName<sub>2</sub>, ...)*

- It enhances the expressive power of the **IN** and **NOT IN** operators

## Example (no.1)

TRIP (TId, StartingPlace, Destination,  
DepartureTime, ArrivalTime)

- Find the pairs of starting places and destinations for which none of the trips lasts more than 2 hours

## Example (no.1)

TRIP (TId, StartingPlace, Destination,  
DepartureTime, ArrivalTime)

- Find the pairs of starting places and destinations for which none of the trips lasts more than 2 hours

```
SELECT StartingPlace, Destination
FROM TRIP
WHERE (StartingPlace, Destination) NOT IN
      (SELECT StartingPlace, Destination
       FROM TRIP
       WHERE ArrivalTime-DepartureTime>2);
```

*Tuple  
constructor*





## Nested queries

The EXISTS operator

# The EXISTS operator (no.1)

➤ Find the names of the suppliers of product P2



*Find the names of the suppliers for which there exists a product supply for P2*



# Correlation condition (no.1)

➤ Find the names of the suppliers of product P2

```
SELECT SName
FROM S
WHERE EXISTS (SELECT *
              FROM SP
              WHERE PId='P2'
              AND SP.SId=S.SId );
```

*Correlation condition*

# How EXISTS works (no.1)

➤ Find the names of the suppliers of product P2

S

SId	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

SId	PId	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

```
SELECT *  
FROM SP  
WHERE PId='P2'
```

```
AND SP.SId='S1'
```

↑  
*Value of SId in the  
current line of table S*

# How EXISTS works (no.1)

➤ Find the names of the suppliers of product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

➤ The predicate including **EXISTS** is true for S1 since there exists a supply for P2 by S1

- S1 belongs to the result of the query

# How EXISTS works (no.1)

➤ Find the names of the suppliers of product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

➤ The predicate including **EXISTS** is false for S4 since there exists no supply for P2 by S4

- S4 is not included in the result of the query

## Result of the query (no.1)

➤ Find the names of the suppliers of product P2

R

SName
Smith
Jones
Blake

# Predicates with EXISTS

- The predicate including EXISTS is
- true if the inner query returns at least one tuple
  - false if the inner query returns the empty set

# Predicates with EXISTS

- The predicate including EXISTS is
  - true if the inner query returns at least one tuple
  - false if the inner query returns the empty set
- In the query inside EXISTS, the SELECT clause is mandatory but irrelevant, since its attributes are never displayed
- The correlation condition binds the execution of the inner query to the values of attributes of the current tuple in the outer query

## Scope of attributes

- A nested query may reference attributes defined within outer queries
- A query may not reference attributes defined
  - within a nested query at an inner level
  - within a different query at the same level





## Nested queries

**The NOT EXISTS operator**

# The NOT EXISTS operator (no.1)

➤ Find the names of the suppliers that *do not* supply product P2



*Find the names of the suppliers for which there does not exist a product supply for P2*

# The NOT EXISTS operator (no.1)

➤ Find the names of the suppliers that *do not* supply product P2


```
SELECT SName
FROM S
WHERE NOT EXISTS (SELECT *
                  FROM SP
                  WHERE PId='P2'
                  AND SP.SId=S.SId );
```

*Correlation condition*

# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S



<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

```
SELECT *
```

```
FROM SP
```

```
WHERE PId='P2' AND
```


```
SP.SId='S1'
```

*value of SId in the  
current line of table S*

# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2


S



<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

```
SELECT *  
FROM SP  
WHERE PId='P2' AND  
       SP.SId='S1'
```

SP




<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

# How NOT EXISTS works (no.1)


➤ Find the names of the suppliers that *do not* supply product P2

S



<u>SId</u>	<u>SName</u>	<u>#Employees</u>	<u>City</u>
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP



<u>SId</u>	<u>PId</u>	<u>Qty</u>
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400


➤ The predicate including **NOT EXISTS** is false for S1 since there exists a supply for P2 by F1

- S1 *does not* belong to the result of the query

# How NOT EXISTS works (no.1)


➤ Find the names of the suppliers that *do not* supply product P2

S



<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP



<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400



# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens





# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

➤ The predicate including **NOT EXISTS** is true of S4 since there exists no supply for P2 by S4

- S4 belongs to the result of the query

# How NOT EXISTS works (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

S

<u>SId</u>	SName	#Employees	City
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SP

<u>SId</u>	<u>PId</u>	Qty
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P3	200
S4	P4	300
S4	P5	400

## Result of the query (no.1)

➤ Find the names of the suppliers that *do not* supply product P2

R

SName
Clark
Adams

# Predicates with NOT EXISTS

- The predicate including NOT EXISTS is
  - true if the inner query returns the empty set
  - false if the inner query return at least one tuple
- The correlation condition binds the execution of the inner query to the values of attributes of the current tuple in the outer query



## Nested queries

Correlation among queries

## Correlation among queries

- It may be required to bind the computation of a nested query to the value(s) of one or more attributes in an outer query
  - the binding is expressed by one or more correlation conditions



# Correlation condition

## ➤ A correlation condition

- must be specified in the WHERE clause of the nested query that requires it
- is a predicate that binds some attributes of tables appearing in the nested query's FROM clause to attributes of tables appearing in the FROM clause of outer queries

## ➤ Correlation conditions may not be expressed

- within queries at the same nesting level
- with references to attributes of a table appearing in the FROM clause of a nested query



## Correlation among queries (no.1)

- For each product, find the code of the supplier that provides the highest quantity

# Correlation among queries (no.1)

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT PId, SId  
FROM SP AS SPX  
WHERE Qty = (...
```

```
)
```

*Maximum  
quantity  
for the current  
product*

## Correlation among queries (no.1)

➤ For each product, find the code of the supplier that provides the highest quantity

```
SELECT PId, SId  
FROM SP AS SPX  
WHERE Qty = (SELECT MAX(Qty)  
             FROM SP AS SPY  
             ... )
```

*Maximum  
quantity*

## Correlation among queries (no.1)

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT PId, SId
FROM SP AS SPX
WHERE Qty = (SELECT MAX(Qty)
             FROM SP AS SPY
             WHERE SPY.PId=SPX.PId);
```

*Maximum  
quantity  
for the current  
product*

## Correlation among queries (no.1)

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT PId, SId
FROM SP AS SPX
WHERE Qty = (SELECT MAX(Qty)
             FROM SP AS SPY
             WHERE SPY.PId=SPX.PId);
```

*Correlation condition*

## Correlation among queries (no.1) – Alternative solution

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT Pid, Sid  
FROM SP AS SPX  
WHERE (PID, QTY) IN (SELECT PID, Max(Qty)  
                    FROM SP  
                    GROUP BY PID)
```

# Correlation among queries (no.1) – Wrong solution

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT Sid  
FROM SP AS SPX  
WHERE QTY = (SELECT Max(Qty)  
FROM SP  
GROUP BY PID)
```

# Correlation among queries (no.1) – Wrong solution

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT Sid  
FROM SP AS SPX  
WHERE QTY = (SELECT Max(Qty)  
FROM SP)
```



# Correlation among queries (no.1) – Wrong solution

- For each product, find the code of the supplier that provides the highest quantity

```
SELECT Sid  
FROM SP AS SPX  
WHERE QTY = (SELECT Max(Qty)  
FROM SP SPY  
WHERE SPY.PID=SPX.PID  
GROUP BY PID)
```

## Correlation among queries (no.2)

TRIP (TId, StartingPlace, Destination,  
DepartureTime, ArrivalTime)

- Find the codes of the trips whose duration is lower than the average duration of the trips on the same route (i.e., same starting place and destination)

```
SELECT TId  
FROM TRIP AS TA  
WHERE ArrivalTime-DepartureTime < (...
```

} *Average  
duration  
of trips  
on the current  
route*

## Correlation among queries (no.2)

TRIP (TId, StartingPlace, Destination,  
DepartureTime, ArrivalTime)

- Find the codes of the trips whose duration is lower than the average duration of the trips on the same route (i.e., same starting place and destination)

```
SELECT TId
FROM TRIP AS TA
WHERE ArrivalTime-DepartureTime <
      (SELECT AVG(ArrivalTime-DepartureTime)
       FROM TRIP AS TB
       ... )
```

*Average  
duration  
of trips*

## Correlation among queries (no.2)

TRIP (TId, StartingPlace, Destination,  
DepartureTime, ArrivalTime)

- Find the codes of the trips whose duration is lower than the average duration of the trips on the same route (i.e., same starting place and destination)

```
SELECT TId
FROM TRIP AS TA
WHERE ArrivalTime-DepartureTime <
  (SELECT AVG(ArrivalTime-DepartureTime)
   FROM TRIP AS TB
   WHERE TB.StartingPlace=TA.StartingPlace
         AND TB.Destination=TA.Destination);
```

*Correlation conditions*



# Nested queries

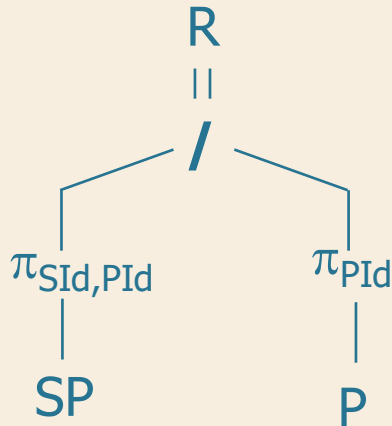
## The division operation

# The division operation (no.1)

- Find the codes of the suppliers that supply *all* products
- In relational algebra we must use the division operator

# The division operation (no.1)

- Find the codes of the suppliers that supply *all* products
- In relational algebra we must use the division operator





## Division in SQL (no.1)

➤ Find the codes of the suppliers that supply *all* products

➤ Remark

- all products that may be supplied are listed in table P



- a supplier is supplying all products if he is supplying a number of distinct products equal to the cardinality of P



## Division in SQL (no.1)

➤ Find the codes of the suppliers that supply *all* products

```
SELECT COUNT(*)  
FROM P
```

## Division in SQL (no.1)

➤ Find the codes of the suppliers that supply *all* products

```
SELECT COUNT(*)  
FROM P
```

} *Total  
number  
of products*

## Division in SQL (no.1)

➤ Find the codes of the suppliers that supply *all* products

```
SELECT SId  
FROM SP  
GROUP BY SId
```

...

```
(SELECT COUNT(*)  
FROM P)
```

## Division in SQL (no.1)

➤ Find the codes of the suppliers that supply *all* products

SP(SID, PID, Qty)

```
SELECT SId
FROM SP
GROUP BY SId
HAVING COUNT(*)=(SELECT COUNT(*)
                   FROM P);
```

# Division in SQL (no.1) – Different SP TABLE

➤ Find the codes of the suppliers that supply *all* products

SP(SID, PID, Date, Qty)

SELECT SId

FROM SP

GROUP BY SId

HAVING COUNT(**DISTINCT PID**)=(SELECT COUNT(\*)  
FROM P);

## Division in SQL: procedure (no.2)

- Find the codes of the suppliers that supply at least *all* of the products supplied by supplier S2
- We must count
  - the number of products supplied by S2
  - the number of products supplied both by an arbitrary supplier and by S2
    - (Alternative formulation)
    - Among the products supplied by S2, count the number of products supplied by an arbitrary supplier

➤ The two counts must be equal

## Division in SQL (no.2)

- Find the codes of the suppliers that supply at least *all* of the products supplied by supplier S2

```
SELECT COUNT(*)  
FROM SP  
WHERE SId='S2'
```

## Division in SQL (no.2)

- Find the codes of the suppliers that supply at least *all* of the products supplied by supplier S2

```
SELECT COUNT(*)  
FROM SP  
WHERE SId='S2'
```

*Number of  
products  
supplied by S2*



## Division in SQL (no.2)

➤ Find the codes of the suppliers that supply at least *all* of the products supplied by supplier S2

```
SELECT SId
FROM SP
WHERE PId IN (SELECT PId
              FROM SP
              WHERE SId='S2')
GROUP BY SId
... (SELECT COUNT(*)
      FROM SP
      WHERE SId='S2')
```

## Division in SQL (no.2)

➤ Find the codes of the suppliers that supply at least *all* of the products supplied by supplier S2

```
SELECT SId
FROM SP
WHERE PId IN (SELECT PId
              FROM SP
              WHERE SId='S2')
GROUP BY SId
HAVING COUNT(*)=(SELECT COUNT(*)
                  FROM SP
                  WHERE SId='S2');
```