Oracle Database Design - 2024

KRYSTIAN WOJTKIEWICZ

Last example from the previous lecture

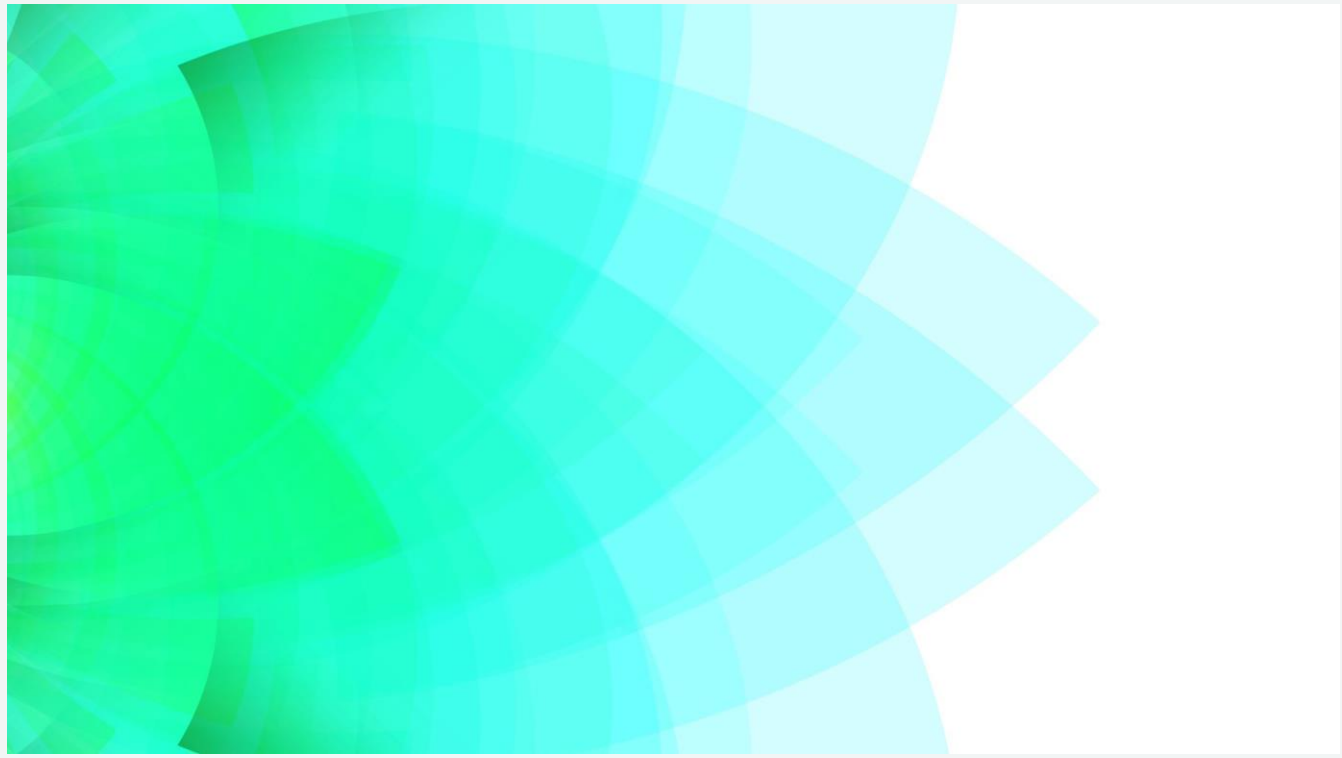
Task 19. *Display data of cats for which the ration of mice exceeds 60. Sort data first ascending by gender and name of the band and then descending by date of join to the herd and then ascending by function name.*

```
SELECT nickname "Nickname",gender "Gender",
       band_no "Band",in_herd_since "Join
date",
       mice_ration "Eats"
FROM Cats WHERE mice_ration>60
ORDER BY 2,"Band",in_herd_since
DESC,function;
```

```
Nickname   Gender Band      Join date           Eats
---------- ------ ---------- ------------------ ----
TIGER      M      1          2002=01-01         103
CAKE       M      2          2008=12-01         67
BALD       M      2          2006=08-15         72
ZOMBIES    M      3          2004=03-16         75
REEF       M      4          2006=10-15         65
FAST       W      2          2006=07-21         65
HEN        W      3          2008=01-01         61

7 rows selected
```

GROUP BY



GROUP BY - remarks

Allows grouping the output according to value of attribute or expression and displays one result row for each group

The SELECT clause, if grouping is used, can only contain

- attributes/expressions listed after the GROUP BY clause,
- aggregate functions,
- pseudo-columns, or
- expressions, which include these elements.

If grouping is performed by of an attribute/expression that can have a null value (NULL), this value is treated as an additional value for the attribute/expression (an associated group is created)

Example

Task 20. *Find nicknames of cats with subordinates.*

```
SELECT chief

FROM Cats

GROUP BY chief;
```

```
CHIEF

---------------

TIGER

ZOMBIES

BALD

REEF

HEN


 6 rows selected
```

Example

Task 21. *Find the number of female cats performing a specific function in each band.*

```
SELECT COUNT(*)||

' number of female cats in the

'||band_no||

' band with the function of '||function

"Statistics for functions"

FROM Cats

WHERE gender='W'

GROUP BY band_no,function;
```

```
Statistics for functions

--------------------------------------------------------------------

2 number of female cats in the 1 band with the function of NICE

1 number of female cats in the 2 band with the function of CATCHING

1 number of female cats in the 2 band with the function of NICE

1 number of female cats in the 3 band with the function of CATCHING

1 number of female cats in the 3 band with the function of NICE

1 number of female cats in the 4 band with the function of CAT

1 number of female cats in the 4 band with the function of CATCHER

7 rows selected
```

COUNT (* | {[DISTINCT | ALL] expression})

The COUNT function always counts rows in a group.

If the argument of the function is *, all rows are counted.

If the argument is an expression (its special case is e.g. an attribute), only those rows for which the expression is different from NULL are counted (repetitions of the value of the expression are taken into account - default ALL).

If, additionally, the DISTINCT qualifier appears before the expression rows with a repetitive expression values are omitted in the count.

More aggregate functions

SUM ([DISTINCT | ALL] expression)

returns the sum of the values of non-NULL expressions taken from each row of the group; DISTINCT omits the repetitive expression value from the calculation (the default is ALL),

AVG ([DISTINCT | ALL] expression)

returns the arithmetic average of the values of non-NULL expressions taken from each row of the group; DISTINCT omits the repetitive expression value from the calculation (the default is ALL),

MAX (expression)

returns the maximum value among non-NULL expression values, taken from each row of the group,

MIN (expression)

returns the minimum value of non-NULL expressions values taken from each row of the group.

Example

Task 22. *Find the average consumption of mice for each gender (including additional rations).*

```
SELECT DECODE(gender,'W','Female
cats',’Male cats’)"Gender",

AVG(NVL(mice_ration,0)+NVL(mice_
extra,0)) "Average consumption„

FROM Cats

GROUP BY gender;
```

```
Gender      Average consumption
----------- --------------------
Female cats 57,5
Male cats   68,9
```

```
SELECT CASE gender
WHEN 'W' THEN 'Female cats’
ELSE 'Male cats' END "Gender",
AVG(NVL(mice_ration,0)+NVL(mice_extra,0))
"Average consumption"
FROM Cats GROUP BY gender;
```

HAVING clause

The HAVING clause is used to select groups resulting from the grouping operation (GROUP BY).

It cannot occur without the GROUP BY clause.

The condition after HAVING determines which groups are selected.

This condition can be built only on the basis of

- the attribute/expression (attributes/expressions) appearing in the GROUP BY clause,
- constants or
- aggregate functions.

Example

Task 23. *Find bands with "mice chimneys" (the ratio of mice of a small number of cats far exceeds the ration of other cats).*

```
SELECT band_no "Chimney band",

       AVG(mice_ration) "Average ration",

       (MAX(NVL(mice_ration,0))+
MIN(NVL(mice_ration,0)))/2

       "(MAX+MIN)/2"

FROM Cats

GROUP BY band_no

HAVING (MAX(NVL(mice_ration,0))+

        MIN(NVL(mice_ration,0)))/2>

        AVG(NVL(mice_ration,0));
```

```
Chimney band     Average ration   (MAX+MIN)/2
---------------- ---------------- -----------
1                50               62,5
4                49,4             52,5
```

CONNECT BY and START WITH clauses

Off-topic

Problem: define a tree structure where every parent has a one-to-many relationship to its children.

So it leads to this:

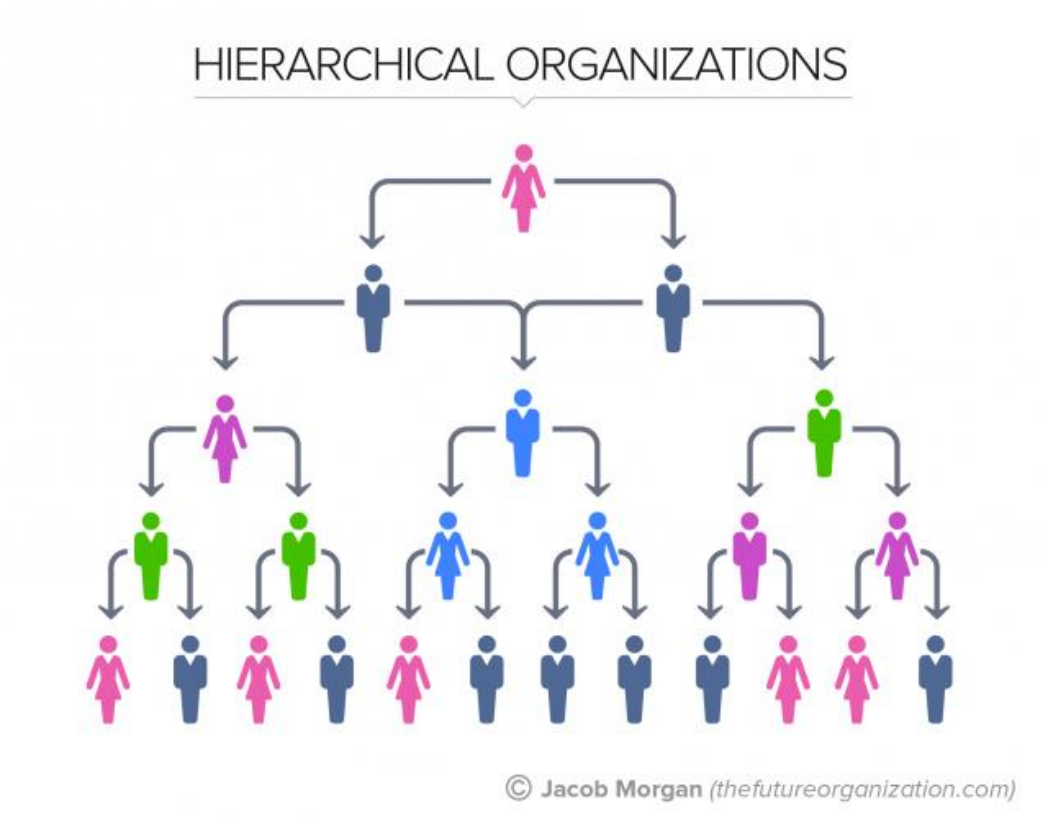
<parent> owns <children>

<parent> is boss of <children>

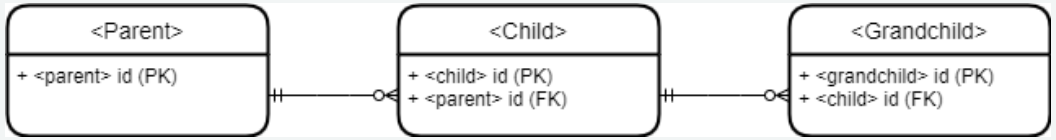
<parent> is derived from <children>

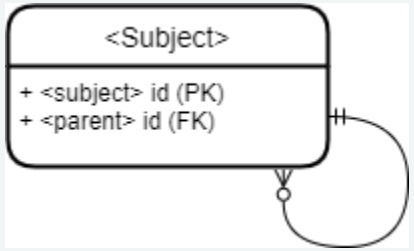
<parent> is composed of <children>

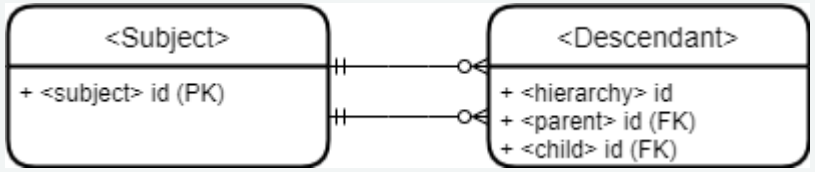
<parent> is generalization of <children>



Hierarchy - solution







The basic SELECT command syntax presented earlier can be supplemented with additional CONNECT BY and START WITH clauses.

They are most often used when we consider hierarchical relationship, i.e., relationship that defines the hierarchy, e.g., superior - subordinate.

The Oracle uses such a relationship to build a tree that reflects this hierarchy.

CONNECT BY and START WITH clauses

The START WITH clause specifies the condition indicating the row which is to be the root of the tree.

If n rows satisfies the condition, n trees are built.

The CONNECT BY clause indicates the condition defining the way the tree is built, i.e., determining which row should be attached as the current leaf of a given node.

The CONNECT BY clause is followed by the PRIOR keyword indicating the so-called the parent attribute (in the row of the current node) from which the value is taken to compare with the value of the second attribute of condition, called the child attribute (from the row that can become a leaf).

CONNECT_BY_ROOT attribute

returns, for the indicated attribute from the currently supported row in the tree, the value of this attribute in the row of the root,

SYS_CONNECT_BY_PATH function (attribute, separator)

returns, for the supported row, the path in the tree from the row of the root to this row, in the form of a string built from the successive values of the indicated attribute at each intermediate node, each value separated by a string separator.

Example

Task 24. *Determine the hierarchy in a herd of cats from the herd leader. In the built tree, skip the cat named KOREK together with all his subordinates and cats with the function NICE.*

```
SELECT name "Name",level "Position",
       band_no "Band",NVL(mice_ration,0) "Eats"
FROM Cats WHERE function!='NICE'
CONNECT BY PRIOR nickname=chief AND name!='KOREK'
START WITH chief IS NULL
ORDER BY band_no,level;
```

Name	Position	Band	Eats
MRUCZEK	1	1	103
CHYTRY	2	1	50
BOLEK	2	2	72
JACEK	3	2	67
ZUZIA	3	2	65
BARI	3	2	56
PUCEK	2	4	65
LATKA	3	4	40
MELA	3	4	51
KSAWERY	3	4	51
DUDEK	3	4	40

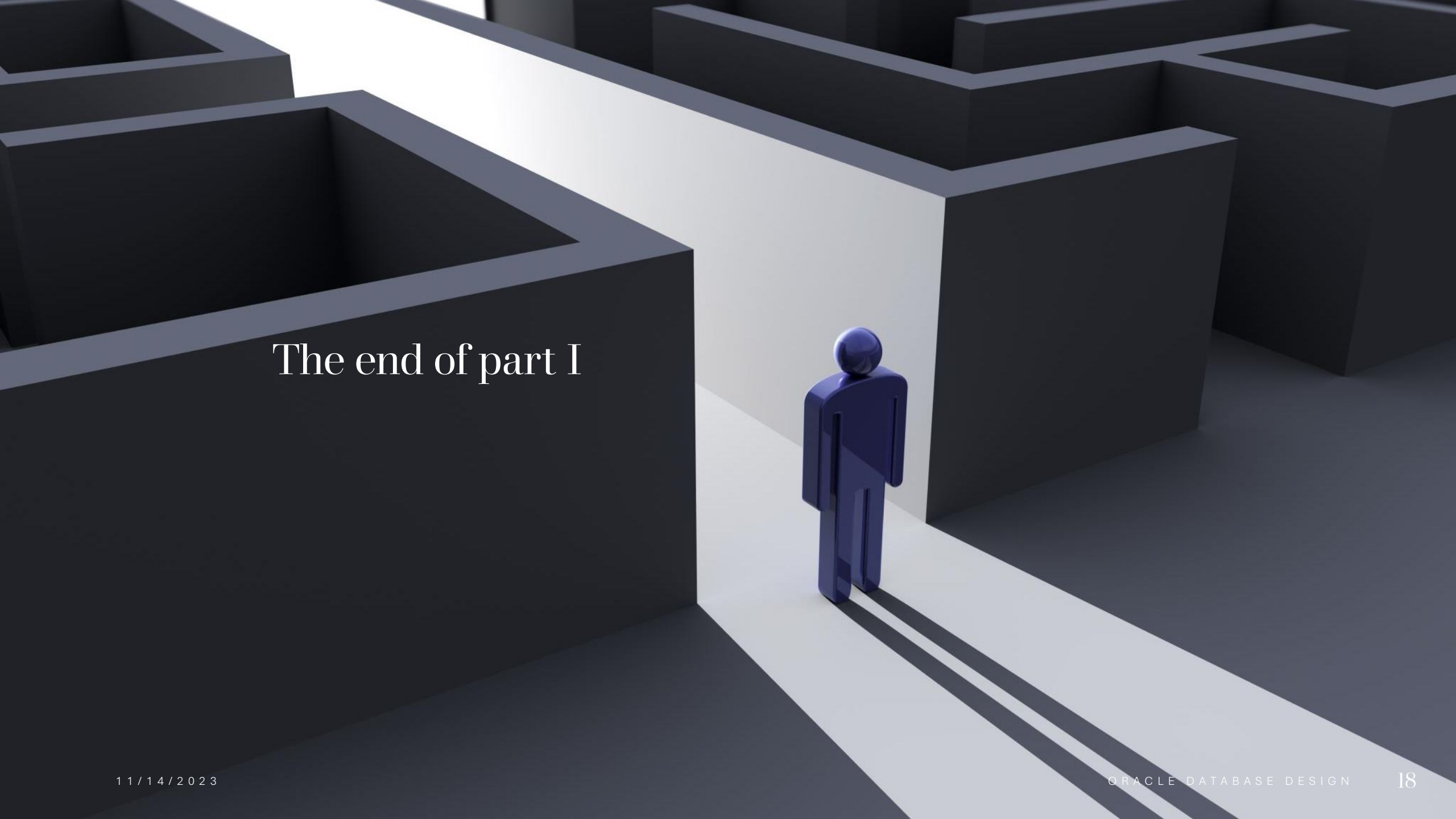
11 rows selected

Example

Task 25. *For each cat belonging to the subtrees with roots ZOMBI and RAFA (nicknames of cats) present the nickname of the cat from the root of the subtree and, in the form of further nicknames, paths from the nickname of the cat from the root to the nickname of the served cat.*

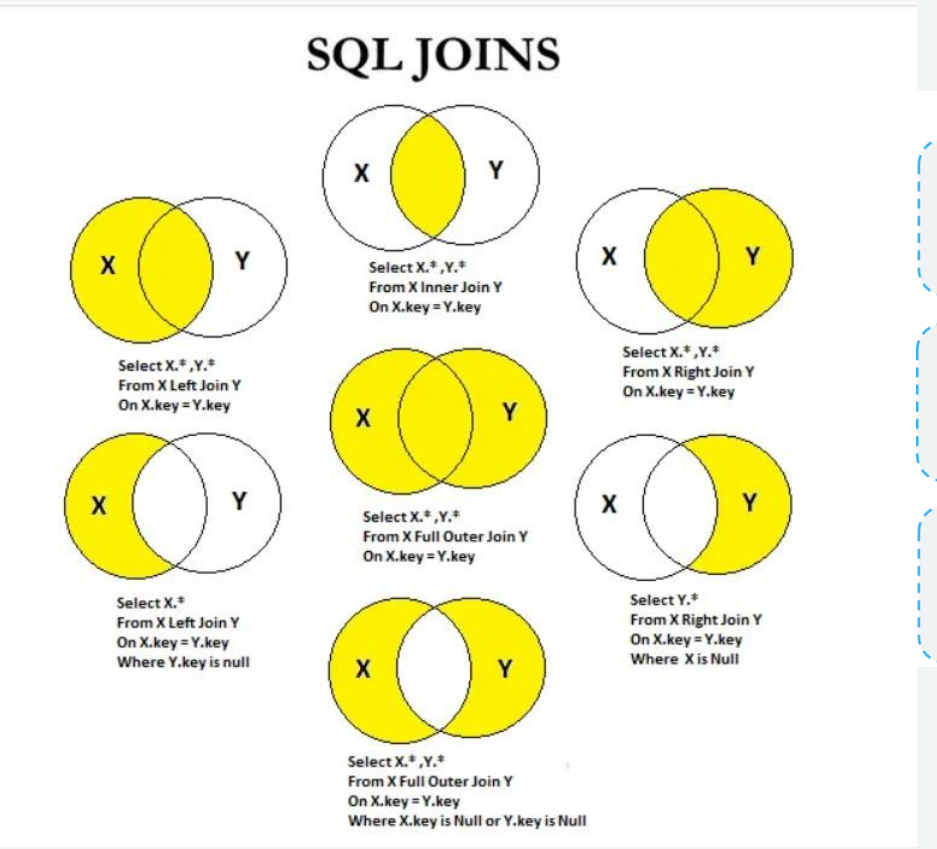
```
SELECT nickname "Cat",
 DECODE(CONNECT_BY_ROOT nickname,
 nickname,NULL,CONNECT_BY_ROOT nickname) "Chief",
 SYS_CONNECT_BY_PATH(nickname,'/') "Nickname path"
FROM Cats
CONNECT BY PRIOR nickname=chief
START WITH nickname IN ('ZOMBIES','REEF');
```

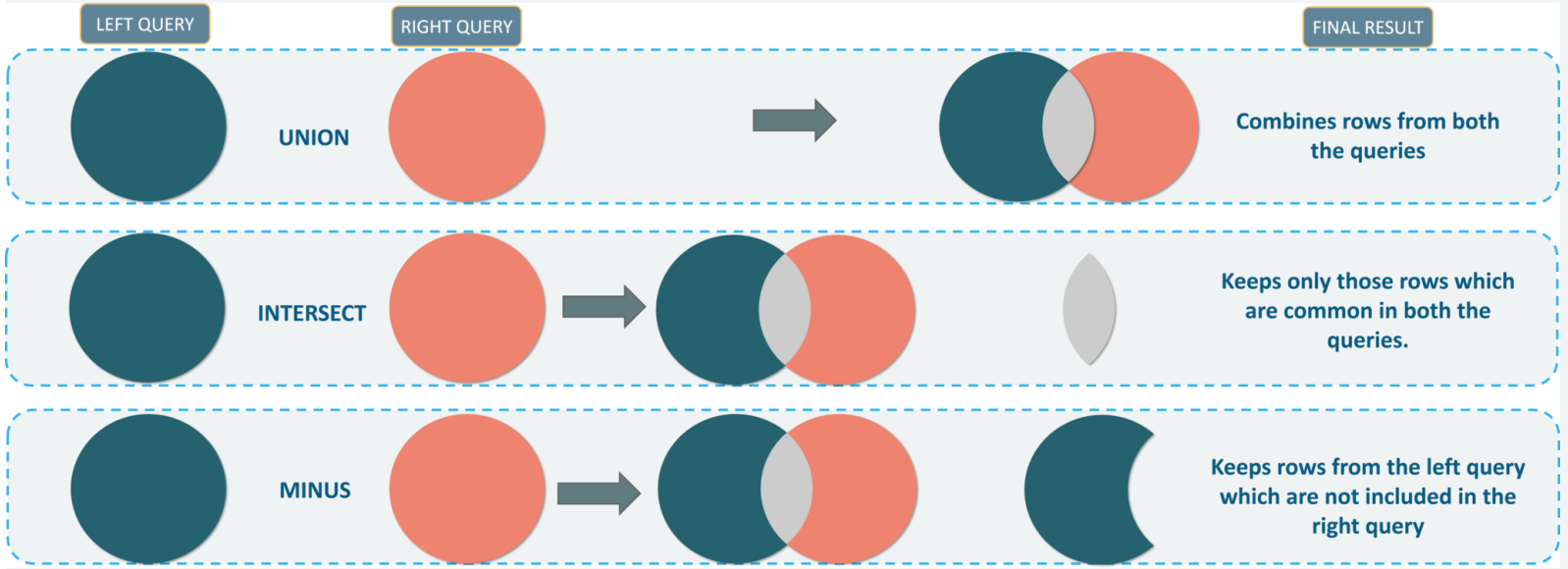
```
Cat        Chief      Nickname path
---------- ---------- ----------------
REEF                  /REEF
EAR        REEF       /REEF/EAR
LADY       REEF       /REEF/LADY
MAN        REEF       /REEF/MAN
SMALL      REEF       /REEF/SMALL
ZOMBIES               /ZOMBIES
FLUFFY     ZOMBIES    /ZOMBIES/FLUFFY
HEN        ZOMBIES    /ZOMBIES/HEN
ZERO       ZOMBIES    /ZOMBIES/HEN/ZERO
 9 rows selected
```



The end of part I

HORIZONTAL nad VERTICAL JOINS





Horizontal joining

In the case of a horizontal join, the result relation row is formed through concatenation of the rows of joined relations (listed in the FROM clause after the comma or as part of the JOIN operator) meeting the so-called joining condition.

The joining condition must include a reference to at least one attribute of each of the joined relations.

If the JOIN operator is not used in the query (relations are listed after the comma in the FROM clause), then it is implemented as the cartesian product of the joined relations (with its possible selection of the rows according to the joining condition or according to condition mentioned in the WHERE clause).

If the JOIN operator is used, the joining condition is defined after the ON clause of the operator (theta-join).

In most cases, we join relations referentially related (the joining condition is based on keys, i.e., on the primary and foreign key, of the related relations).

Example

Task 26. *Find female cats who participated in incidents. Display, in addition, the names of the enemies involved in the incidents and descriptions of incidents.*

```
SELECT C.nickname "Female cat", enemy_name "her enemy", incident_desc "Incident description"

FROM Cats C,Incidents I

WHERE C.nickname=I.nickname AND gender='W';
```

```
Female cat her enemy       Incident description
---------- --------------- ----------------------------------
EAR        UNRULY DYZIO    HE THREW STONES
FAST       STUPID SOPHIA   SHE USED THE CAT AS A CLOTH
FLUFFY     SLIM            SHE THREW CONES
HEN        DUN             HE CHASED
LADY       KAZIO           HE WANTED TO SKIN OFF
LITTLE     SLYBOOTS        HE RECOMMENDED HIMSEF AS A HUSBAND
MISS       KAZIO           HE CAUGHT THE TAIL AND MADE A WIND
MISS       WILD BILL       HE BITCHED

 8 rows selected
```

Example – other implementations

```
SELECT C.nickname "Female cat", enemy_name "her enemy", incident_desc "Incident description"

FROM Cats C,Incidents I

WHERE C.nickname=I.nickname AND gender='W';
```

```
SELECT C.nickname "Female cat", enemy_name "her enemy", incident_desc "Incident description"

FROM Cats C JOIN Incidents I ON C.nickname=I.nickname

WHERE gender='W';
```

```
SELECT nickname "Female cat",enemy_name "her enemy", incident_desc "Incident description"

FROM Cats JOIN Incidents USING(nickname)

WHERE gender='W';
```

```
SELECT nickname "Female cat",enemy_name "her enemy", incident_desc "Incident description"

FROM Cats NATURAL JOIN Incidents

WHERE gender='W';
```

Example

Task 27. *Find cats hunting on the site FIELD which have enemies with hostility degree above 5.*

```
SELECT DISTINCT C.nickname "Has enemy in
the field"
FROM Cats C,Incidents I,Enemies E,Bands B
WHERE C.band_no=B.band_no AND
C.nickname=I.nickname AND
I.enemy_name=E.enemy_name AND
site IN ('FIELD','WHOLE AREA') AND
hostility_degree>5;
```

```
Has enemy in the field
----------------------
TIGER
MISS
TUBE
BOLEK
```

Example – other implementations

```
SELECT DISTINCT C.nickname "Has enemy in the field"

FROM Cats C,Incidents I,Enemies E,Bands B

WHERE C.band_no=B.band_no AND C.nickname=I.nickname AND I.enemy_name=E.enemy_name AND site IN ('FIELD','WHOLE AREA')
AND hostility_degree>5;
```

```
SELECT DISTINCT C.nickname "Has enemy in the field"

FROM Cats C JOIN Incidents I ON C.nickname=I.nickname JOIN Enemies E ON I.enemy_name=E.enemy_name JOIN Bands B ON
C.band_no=B.band_no

WHERE site IN ('FIELD','WHOLE AREA') AND hostility_degree>5;
```

```
SELECT nickname "Has enemy in the field", band_no

FROM Cats NATURAL JOIN Incidents NATURAL JOIN Enemies JOIN Bands USING(band_no)

WHERE site IN ('FIELD','WHOLE AREA') AND hostility_degree>5;
```

Example

Task 28. *In each of the bands, apart from his own, Tiger has placed a spy. He can be recognized by the fact that in cat hierarchy he reports directly to the Tiger and not the boss of the band although he is not a member of the Tiger's band. Find all the Tiger spies.*

```
SELECT C1.nickname "Spy",C1.band_no
"Band„

FROM Cats C1 JOIN Cats C2 ON
C1.chief=C2.nickname AND
C1.band_no<>C2.band_no

WHERE C1.chief='TIGER';
```

```
Spy             Band
--------------- --------------------

ZOMBIES         3

BALD            2

REEF            4
```