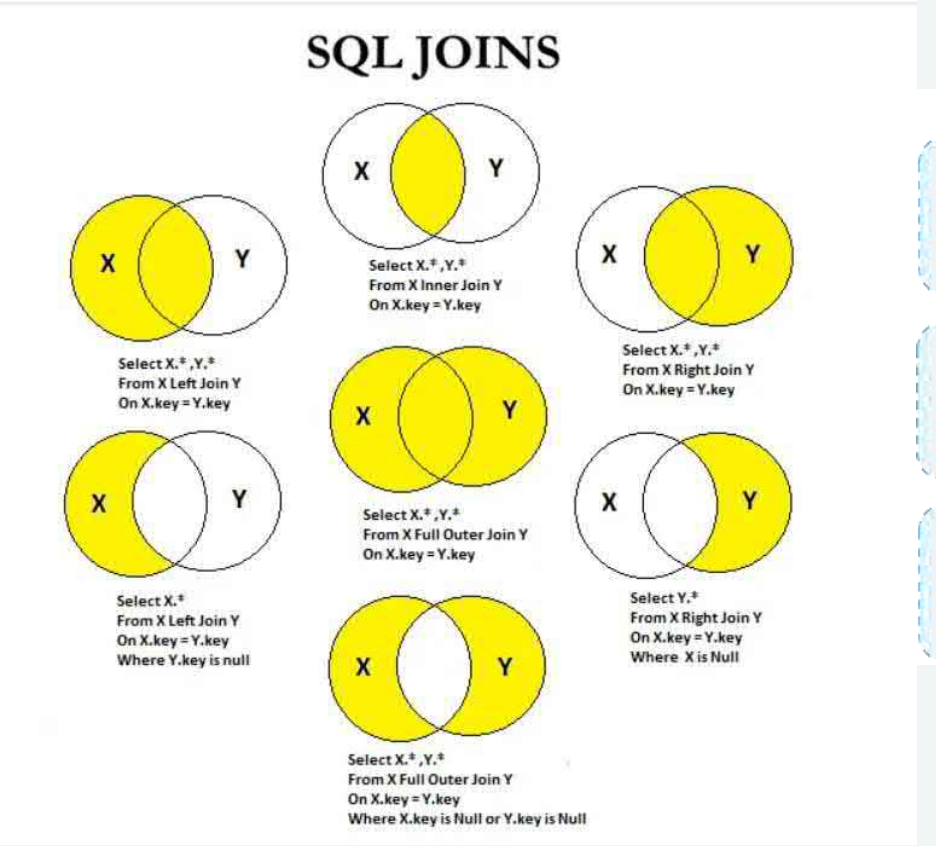
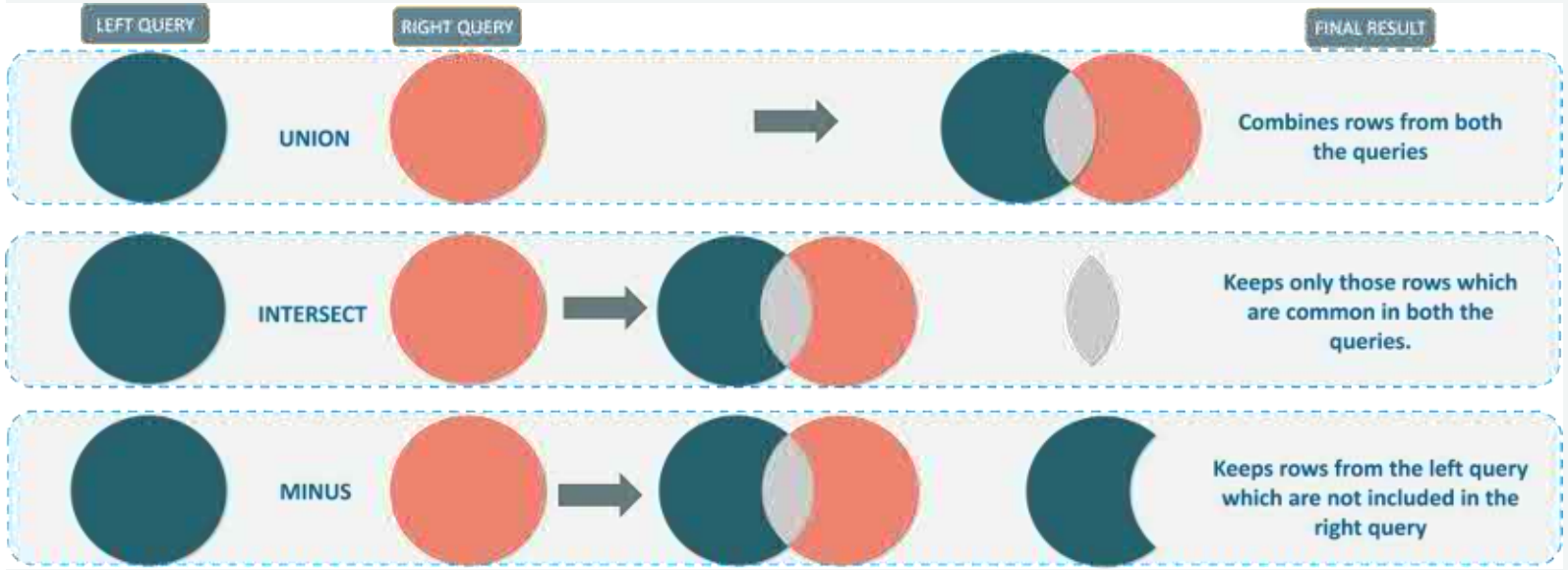
Oracle Database Design - 2024

KRYSTIAN WOJTKIEWICZ

HORIZONTAL and 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;
```

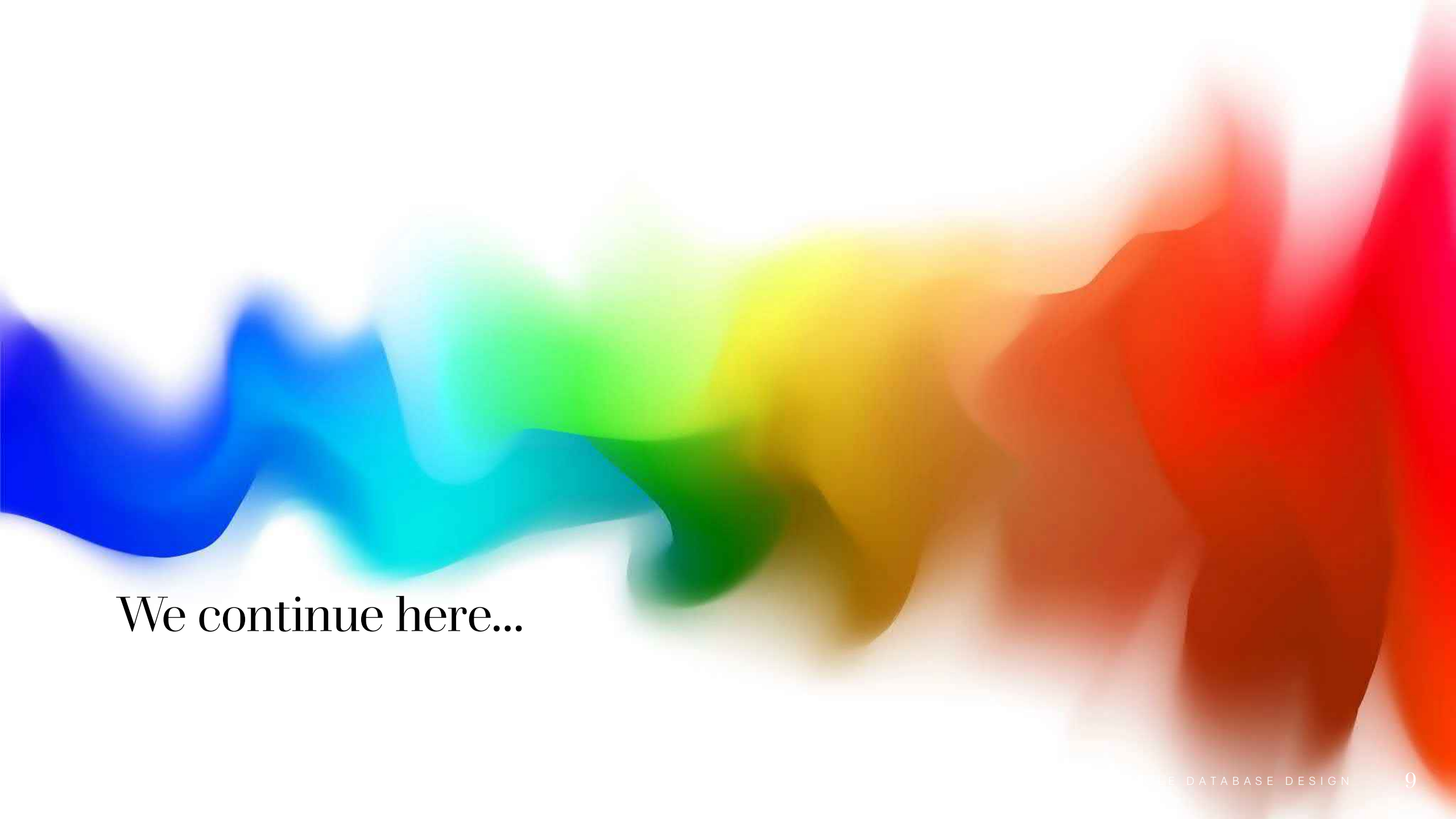
```
SELECT nickname "Has enemy in the field", band_no

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
```


We continue here...

Example

Task 29. *Find the names of cats who joined the herd earlier than their immediate superiors.*

```
SELECT C1.name "In the herd before
the boss" FROM Cats C1,Cats C2
WHERE C1.chief=C2.nickname AND
C1.in_herd_since<C2.in_herd_since;
```

```
In the herd before the boss
---------------------------
ZUZIA
```

```
SELECT C1.name "In the herd before
the boss" FROM Cats C1 JOIN Cats C2
ON C1.chief=C2.nickname AND
C1.in_herd_since<C2.in_herd_since;
```

Example

Task 30. *Display the names of cats that have not yet participated in the incidents.*

```
SELECT name "No incident„

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

WHERE I.nickname IS NULL;
```

```
No incident

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

MICKA

PUCEK

LUCEK
```

```
SELECT name "No incident"

FROM Cats C, Incidents I

WHERE C.nickname=I.nickname(+) AND
I.nickname IS NULL;
```

Example

Task 31. *Display a report that returns information about male cats' subordinates and superiors. If the cat does not have a subordinate, this should be indicated in the report. Similarly, the report should indicate the absence of a superior.*

```
SELECT NVL(C1.nickname,'No superior') "Superior",
       NVL(C2.nickname,'No subordinate') "Subordinate"
FROM Cats C1 FULL JOIN Cats C2 ON C1.nickname=C2.chief
WHERE DECODE(C1.nickname,NULL,'M',C1.gender)='M' AND
      DECODE(C2.nickname,NULL,'M',C2.gender)='M’
ORDER BY 1;
```

```
Superior        Subordinate
--------------- ---------------
BALD            CAKE
BALD            TUBE
BOLEK           No subordinate
CAKE            No subordinate
MAN             No subordinate
No superior     TIGER
REEF            MAN
REEF            SMALL
SMALL           No subordinate
TIGER           BALD
TIGER           ZOMBIES
TIGER           BOLEK
TIGER           REEF
TUBE            No subordinate
ZERO            No subordinate

 15 rows selected
```

Vertical joining

In the case of vertical joining, the joined relations are treated as sets of rows and the resulting relation is the result of a set operation on these sets.

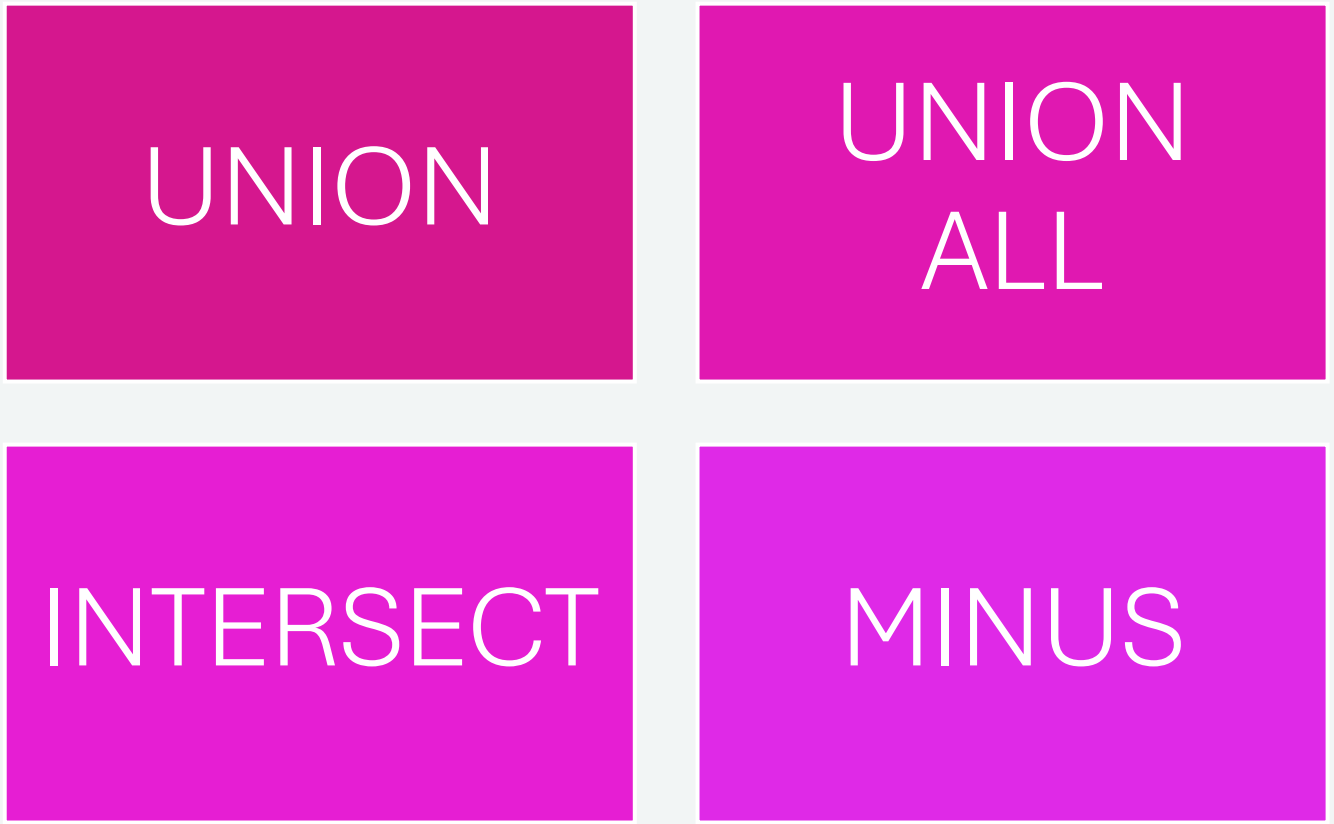
To perform the vertical join, the joining relations must have the same number of attributes and their types must be the same, respectively (relations must have the same scheme).

The attribute names of the resulting relation always come from the first joined relation.

In query with vertical join, ORDER BY clause may appear only once, as its last.

Ordering operation can only be done according to the expression numbers that appear in the SELECT clause of the joined relations.

Oracle set operators



Example

Task 32. *Specify the functions of cats in bands 1 and 2.*

```
SELECT function FROM Cats WHERE band_no=1

UNION

SELECT function FROM Cats WHERE band_no=2;
```

```
FUNCTION

----------

BOSS

CATCHER

CATCHING

DIVISIVE

NICE

THUG


6 rows selected
```

The fun begins...

LET'S USE SUBQUERIES

A few facts

SELECT query clauses may contain nested SELECT clauses, i.e., subqueries. This applies to the following clauses:

- SELECT
- FROM
- WHERE
- HAVING

For the SELECT clause the subquery must return only one value (the resulting relation of the subquery must consist of one row and one attribute).

The subquerries might be correlated (bound) and uncorrelated (unbound).

A correlated subquery is performed for each row of the external query, an uncorrelated subquery only once at the beginning of the external query action.

The subquery cannot contain the ORDER BY clause (except for the subquery in the FROM clause).

A correlated subquery cannot appear in the FROM clause.

Example

Task 33. *Find cats that perform the same function as a cat with nickname LOLA.*

```
SELECT name "LOLA's deputy",band_no "its band"
FROM Cats
WHERE function=(SELECT function
                FROM Cats
                WHERE nickname='LOLA’)
      AND nickname!='LOLA';
```

```
LOLA's deputy   its band
--------------- -----------
SONIA           3
RUDA            1
BELA            2
```

```
SELECT C1.name "LOLA's
deputy",C1.band_no "its band"
FROM Cats C1,Cats C2
WHERE C1.function=C2.function
AND C2.nickname='LOLA'
      AND C1.nickname!='LOLA';
```

Example

Task 34. *Find cats for whose mice ration is greater than the average ration throughout the herd.*

```
SELECT nickname "Nickname",mice_ration "Eats"
FROM Cats
WHERE mice_ration>(SELECT
AVG(NVL(mice_ration,0))
                  FROM Cats);
```

```
Nickname        Eats
--------------- --------------------
CAKE            67
TUBE            56
TIGER           103
ZOMBIES         75
BALD            72
FAST            65
REEF            65
HEN             61
 8 rows selected
```

```
SELECT nickname "Nickname",mice_ration "Eats"
FROM Cats,(SELECT AVG(NVL(mice_ration,0)) av
                  FROM Cats)
WHERE mice_ration>av;
```

```
SELECT nickname "Nickname",mice_ration "Eats"
FROM Cats JOIN (SELECT AVG(NVL(mice_ration,0))
av FROM Cats) ON mice_ration>av;
```

Example

Task 35. *Find cats whose ration of mice belongs to the list of smallest rations from each bands.*

```
SELECT    name    "Name",NVL(mice_ration,0)
"Eats",band_no "Band"

FROM Cats

WHERE    NVL(mice_ration,0)    IN    (SELECT
MIN(NVL(mice_ration,0))

                              FROM Cats

                              GROUP BY

band_no)
```

Name	Eats	Band
RUDA	22	1
BELA	24	2
SONIA	20	3
LATKA	40	4
DUDEK	40	4

Example

Task 36. *Find the cats with the lowest mice ration in their bands.*

```
SELECT name "Name",mice_ration "Eats",band_no "Band"
FROM Cats
WHERE (NVL(mice_ration,0),band_no) IN
                        (SELECT
MIN(NVL(mice_ration,0)),band_no
                         FROM Cats
                         GROUP BY band_no)
ORDER BY band_no;
```

Name	Eats	Band
RUDA	22	1
BELA	24	2
SONIA	20	3
LATKA	40	4
DUDEK	40	4

Example

Task 36. *Find the cats with the lowest mice ration in their bands.*

```
SELECT name "Name",mice_ration "Eats",band_no "Band"
FROM Cats
WHERE (NVL(mice_ration,0),band_no) IN
                        (SELECT
MIN(NVL(mice_ration,0)),band_no
                         FROM Cats
                         GROUP BY band_no)
ORDER BY band_no;
```

```
SELECT name "Name",mice_ration "Eats",band_no "Band"
FROM Cats,(SELECT MIN(NVL(mice_ration,0)) mi,band_no nb
           FROM Cats
           GROUP BY band_no)
WHERE band_no=nb AND NVL(mice_ration,0)=mi
ORDER BY band_no;
```

```
SELECT name "Name",mice_ration "Eats",band_no "Band"
FROM Cats JOIN (SELECT MIN(NVL(mice_ration,0))
mi,band_no nb
                FROM Cats
                GROUP BY band_no) ON band_no=nb AND

NVL(mice_ration,0)=mi ORDER BY band_no;
```

```
SELECT name "Name",mice_ration "Eats",band_no "Band"
FROM Cats C
WHERE NVL(mice_ration,0)= (SELECT
MIN(NVL(mice_ration,0))
                           FROM Cats
                           WHERE band_no=C.band_no)
ORDER BY band_no;
```

Remark

One can use the ANY operator or the ALL operator for subqueries placed in the WHERE clause that returns only one column. These operators always occur with relation operators:

=, <,>, <=,> =

For example, expressions:

a) X> ANY subquery,

b) X <ALL subquery,

will be TRUE if X is greater than at least one value returned by the subquery (expression a)) or if X is less than each value returned by the subquery (expression b)).

Similarly, one can use the ANY and ALL operators for other relationship operators.

Example

Task 37. *Find cats whose ration of mice is greater than the lowest mice ration in band 4. Use the ANY operator to solve the task.*

```
SELECT name "Name", NVL(mice_ration,0) "Eats",
       band_no "Band"
FROM Cats
WHERE mice_ration>ANY(SELECT DISTINCT NVL(mice_ration,0)
                      FROM Cats
                      WHERE Band_no=4)
ORDER BY "Eats" DESC;
```

```
Name            Eats        Band
--------------- ----------- -----------
MRUCZEK         103         1
KOREK           75          3
BOLEK           72          2
JACEK           67          2
PUCEK           65          4
ZUZIA           65          2
PUNIA           61          3
BARI            56          2
KSAWERY         51          4
MELA            51          4
CHYTRY          50          1
LUCEK           43          3

 12 rows selected
```

Example

Task 38. *Find bands in which the average mice ration is higher than the average ration in band 3.*

```
SELECT band_no "Band better than a band 3",
       AVG(NVL(mice_ration,0)) "Average ration in band"
FROM Cats
HAVING AVG(NVL(mice_ration,0))>(SELECT
AVG(NVL(mice_ration,0))
       FROM Cats
       WHERE band_no=3)
GROUP BY band_no;
```

```
Band better than a band 3 Average ration in band
------------------------- ----------------------
1                         50
2                         56,8
```

Example

Task 39. *Find cats whose ration of mice is higher than the highest ration in the PINTO HUNTERS band.*

```
SELECT name "Better than any of PINTO",mice_ration "Eats"
FROM Cats
WHERE mice_ration>(SELECT MAX(NVL(mice_ration,0))
                   FROM Cats
                   WHERE band_no=(SELECT band_no
                                  FROM Bands
                                  WHERE name='PINTO HUNTERS'))
ORDER BY mice_ration DESC;
```

```
Better than any of PINTO Eats
------------------------ -------
MRUCZEK                  103
KOREK                    75
BOLEK                    72
JACEK                    67
```

Example

Task 40. *Find cats from the band WHITE HUNTERS and PINTO HUNTERS, whose ration of mice is higher than the average ration of the whole herd.*

```
SELECT nickname "WHITE and PINTO above!"
FROM Cats
WHERE Band_no IN (SELECT band_no
                  FROM Bands
                  WHERE name IN
                            ('WHITE HUNTERS','PINTO HUNTERS'))
                  AND
                            mice_ration>(SELECT
                                     AVG(NVL(mice_ration,0))
              FROM Cats);
```

```
WHITE and PINTO above!
----------------------
ZOMBIES
REEF
HEN
```

Example

Task 41. *For each male cat, find the average ration of mice released monthly to the cat in his band.*

```
SELECT nickname "Cat",(SELECT
AVG(NVL(mice_ration,0))
       FROM Cats
       WHERE band_no=C.band_no)
"Average in the band"
FROM Cats C
WHERE gender='M';
```

Cat	Average in the band
CAKE	56,8
TUBE	56,8
ZERO	49,75
SMALL	49,4
TIGER	50
BOLEK	50
ZOMBIES	49,75
BALD	56,8
REEF	49,4
MAN	49,4

10 rows selected

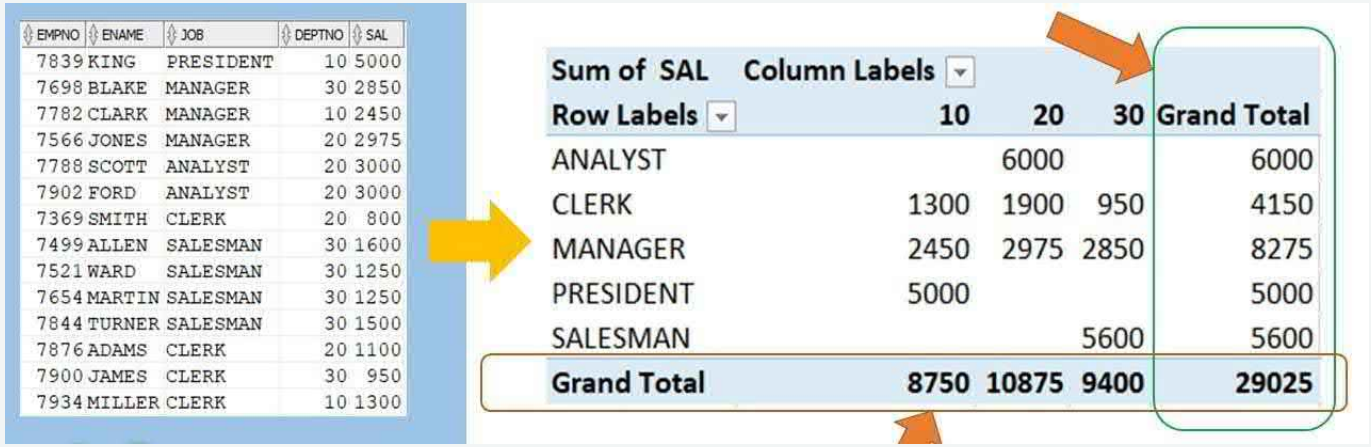
Example

Task 42. *The whole leadership elite of the cats herd came to the conclusion that the potential threat to their power are cats, which do not have subordinates, but are sometimes feisty (have enemies) and in addition their ration of mice is at least equal to the value min_mice+(max_mice-min_mice)/3 (they had to stand out in the past), where min_mice and max_mice is determined by their function. Find these cats.*

```
SELECT nickname "For crawling",B.name "Band name"
FROM Cats C JOIN Bands B USING(band_no)
WHERE NOT EXISTS (SELECT nickname
                  FROM Cats
                  WHERE chief=C.nickname)
INTERSECT
SELECT nickname,B.name
FROM Cats JOIN Bands B USING (band_no)
          JOIN Functions USING(function)
WHERE mice_ration>NVL(min_mice,0)+
                  (NVL(max_mice,0)-NVL(min_mice,0))/3
INTERSECT
SELECT nickname,B.name
FROM Cats C JOIN Bands B USING(band_no)
WHERE EXISTS (SELECT nickname
              FROM Incidents
              WHERE nickname=C.nickname)
ORDER BY 2,1;
```

For crawling	Band name
CAKE	BLACK KNIGHTS
FAST	BLACK KNIGHTS
MISS	BLACK KNIGHTS
TUBE	BLACK KNIGHTS
BOLEK	SUPERIORS

Pivot Tables



EMPNO	ENAME	JOB	DEPTNO	SAL
7839	KING	PRESIDENT	10	5000
7698	BLAKE	MANAGER	30	2850
7782	CLARK	MANAGER	10	2450
7566	JONES	MANAGER	20	2975
7788	SCOTT	ANALYST	20	3000
7902	FORD	ANALYST	20	3000
7369	SMITH	CLERK	20	800
7499	ALLEN	SALESMAN	30	1600
7521	WARD	SALESMAN	30	1250
7654	MARTIN	SALESMAN	30	1250
7844	TURNER	SALESMAN	30	1500
7876	ADAMS	CLERK	20	1100
7900	JAMES	CLERK	30	950
7934	MILLER	CLERK	10	1300

Sum of SAL	Column Labels			
Row Labels	10	20	30	Grand Total
ANALYST		6000		6000
CLERK	1300	1900	950	4150
MANAGER	2450	2975	2850	8275
PRESIDENT	5000			5000
SALESMAN			5600	5600
Grand Total	**8750**	**10875**	**9400**	**29025**

```
SELECT ...
FROM DATA_SOURCE
        PIVOT
        (
         AGGREGATE_FUNCTION
FOR COLUMN_TO_ROTATION
         IN RANGE_OF_DATA_TO_ROTATION
        )
 WHERE...
```

Example

Task 43. *Display functions and total rations of mice for cats from the bands WHITE HUNTER and BLACK KNIGHTS. As a result, do not include the BOSS function.*

```
SELECT function, B.name "Band name",
       NVL(mice_ration,0)+NVL(mice_extra,0)
           "Total mice ration"
FROM Cats JOIN Bands B USING(band_no)
WHERE B.name IN ('BLACK KNIGHTS','WHITE HUNTERS') AND
function != 'BOSS';
```

```
FUNCTION   Band name       Total mice ration
---------- --------------- -----------------
THUG       BLACK KNIGHTS   93
CATCHING   BLACK KNIGHTS   65
CATCHING   BLACK KNIGHTS   67
CATCHER    BLACK KNIGHTS   56
NICE       BLACK KNIGHTS   52
NICE       WHITE HUNTERS   55
CAT        WHITE HUNTERS   43
THUG       WHITE HUNTERS   88
CATCHING   WHITE HUNTERS   61

 9 rows selected
```

Example

Task 43*. *Display the total rations of mice for bands WHITE HUNTERS and BLACK KNIGHTS, divided into functions performed by cats. Skip in the list the BOSS function.*

```
SELECT function, B.name "Band name",
       SUM(NVL(mice_ration,0)+NVL(mice_extra,0))
              "Total ration for the band"
FROM Cats JOIN Bands B USING(band_no)
WHERE B.name IN ('BLACK KNIGHTS','WHITE HUNTERS')
      AND function != 'BOSS'
GROUP BY function,B.name;
```

```
FUNCTION   Band name      Total ration for the band
---------- -------------- ------------------------
THUG       WHITE HUNTERS  88
CATCHER    BLACK KNIGHTS  56
CATCHING   WHITE HUNTERS  61
THUG       BLACK KNIGHTS  93
NICE       WHITE HUNTERS  55
CATCHING   BLACK KNIGHTS  132
NICE       BLACK KNIGHTS  52
CAT        WHITE HUNTERS  43

 8 rows selected
```

```
SELECT *
FROM (SELECT function, B.name band_name,
      NVL(mice_ration,0)+NVL(mice_extra,0) mice_total
      FROM Cats JOIN Bands B USING(band_no))
      PIVOT
      (
       SUM(mice_total)
       FOR band_name
       IN ('BLACK KNIGHTS' "Band BLACK KNIGHTS",
           'WHITE HUNTERS' "Band WHITE HUNTERS")
              )
        WHERE function != 'BOSS'
        ORDER BY "Band BLACK KNIGHTS" DESC;
```

```
FUNCTION   Band BLACK KNIGHTS Band WHITE HUNTERS
---------- ------------------ ------------------
CAT                           43
DIVISIVE
CATCHING   132                61
THUG       93                 88
CATCHER    56
NICE       52                 55

  6 rows selected
```

```
SELECT *
FROM (SELECT function, B.name band_name,gender,
      NVL(mice_ration,0)+NVL(mice_extra,0) mice_total
      FROM Cats JOIN Bands B USING(band_no))
      PIVOT
      (
       SUM(mice_total)
       FOR band_name
       IN ('BLACK KNIGHTS' "Band BLACK KNIGHTS",
           'WHITE HUNTERS' "Band WHITE HUNTERS")
              )
        WHERE function != 'BOSS'
        ORDER BY "Band BLACK KNIGHTS" DESC;
```

FUNCTION	GENDER	Band BLACK KNIGHTS	Band WHITE HUNTERS
CAT	W		
DIVISIVE	M		
CAT	M		43
CATCHER	W		
THUG	M	93	88
CATCHING	M	67	
CATCHING	W	65	61
CATCHER	M	56	
NICE	W	52	55

9 rows selected

WITH clause

```
WITH {SubqueryName AS (subquery) [, …]}
SELECT [DISTINCT | ALL] { expression [alias] [, ...]} | *
FROM {RelationViewNameSuqueryName [alias]
                [join_operator
                RelationViewNameSuqueryName [alias]
                [ON join_condition]] [, ...]}
Rest_of_SELECT_command
```

Placed before the SELECT query, the WITH clause lets to name relations effecting from subqueries in the FROM clause of that query. By optimizing the query, Oracle implements such a subquery in the form of a materialized view or temporary table.

Example

Task 44. *Find cats whose ration of mice is greater than the average ration of the whole herd.*

```
WITH Av AS
      (SELECT AVG(NVL(mice_ration,0)) avgmr
      FROM Cats)
SELECT nickname
"Nickname",NVL(mice_ration,0) "Eats"
FROM Cats JOIN Av ON mice_ration>avgmr;
```

```
Nickname        Eats
--------------- -------
CAKE            67
TUBE            56
TIGER           103
ZOMBIES         75
BALD            72
FAST            65
REEF            65
HEN             61

 8 rows selected
```

Example

Task 45. *Find female cats that have participated in incidents with enemies with hostility degree above 5.*

```
WITH Ladies AS
     (SELECT nickname
      FROM Cats
      WHERE gender='W'),
     Incidents5 AS
     (SELECT nickname
      FROM Incidents NATURAL JOIN Enemies
      WHERE hostility_degree>5)
SELECT DISTINCT nickname "Feisty female cats"
FROM Ladies NATURAL JOIN Incidents5;
```

```
Feisty female cats
------------------
EAR
MISS
LADY
```

```
WITH Ladies AS
     (SELECT nickname
      FROM Cats
      WHERE gender='W'),
     FeistyFemaleCats AS
     (SELECT DISTINCT nickname
      FROM Incidents NATURAL JOIN Enemies
                     NATURAL JOIN Ladies
      WHERE hostility_degree>5)
SELECT nickname "Feisty Female Cats"
FROM FeistyFemaleCats;
```

DML – Data Modification Language

INSERT

```
INSERT INTO RelationViewName [({attribute [, …]})]
{VALUES ({value [, …]})} | subquery
```

```
INSERT ALL {INTO RelationViewName [({attribute [, …]})]
VALUES ({value [, …]}) [ ...]}
{SELECT * FROM Dual }| subquery
```

Example

Task 46. *Tiger decided to punish the insubordination of his subordinates by sending them temporarily to cellars belonging to the inhabitants of the village of Wólka Mała. Define in Bands relation a new band called* ***Exiles****, whose hunting place will be cellars*

```
INSERT INTO Bands
(band_no,name,site,band_chief)
VALUES(6,'EXILES','CELLARS',NULL);
1 rows inserted.
```

Or

```
INSERT INTO Bands
VALUES(6,'EXILES','CELLARS',NULL);
1 rows inserted.
```

```
ROLLBACK;
rollback complete.
```

Example

Task 47. *Several cats, previously non-attached, decided to join the herd. To facilitate their admission, the relation called* ***New*** *with the attributes* ***nickname****,* ***name****,* ***sex****, where data of new cats were aggregate, was prepared. Using the* ***New*** *relation, add to the herd new members.*

```
INSERT INTO Cats

(nickname,name,gender,in_herd_since,chief)

SELECT nickname,name,sex,SYSDATE,'TIGER'
FROM New;

5 rows inserted.
```

```
ROLLBACK;

rollback complete.
```

UPDATE

UPDATE RelationViewName [alias]

SET {attribute_name = expression [, ...]}

[**WHERE** condition]

Example

Task 48. *Promote a female cat named LATKA to the CATCHER function, giving her a 30% increase of ration of mice.*

```
UPDATE Cats

SET funkcja='CATCHER’,

mice_ration=ROUND(NVL(mice_ration,0)*1.3,0)

WHERE name='LATKA';


1 rows modified.
```

```
ROLLBACK;

rollback complete.
```

Example

Task 49. *Tiger, as an enlightened ruler, decided that he would not manage additional mice rations under the influence of current emotions. Instead, he decided to make monthly payments based on the "merits" remembered during a month in the* ***Extra_additions*** *relation. Modify the value of additional rations of mice on the base of "merits" of cats remembered in the* ***Extra_additions*** *relation.*

```
SELECT nickname "Nickname", extra_add "Extra
addition"
FROM Extra_additions;
```

```
Nickname  Extra addition
--------- --------------
TIGER                 10
LOLA                   5
BOLEK                 10
TIIGER                 5
LOLA                  -2
```

```
UPDATE Cats C
SET mice_extra=(SELECT NVL(C.mice_extra,0)+SUM(extra_add)
                FROM Extra_additions E
                WHERE E.nickname =C.nickname)
WHERE nickname IN (SELECT nickname
                   FROM Extra_additions);

3 rows modified.
```

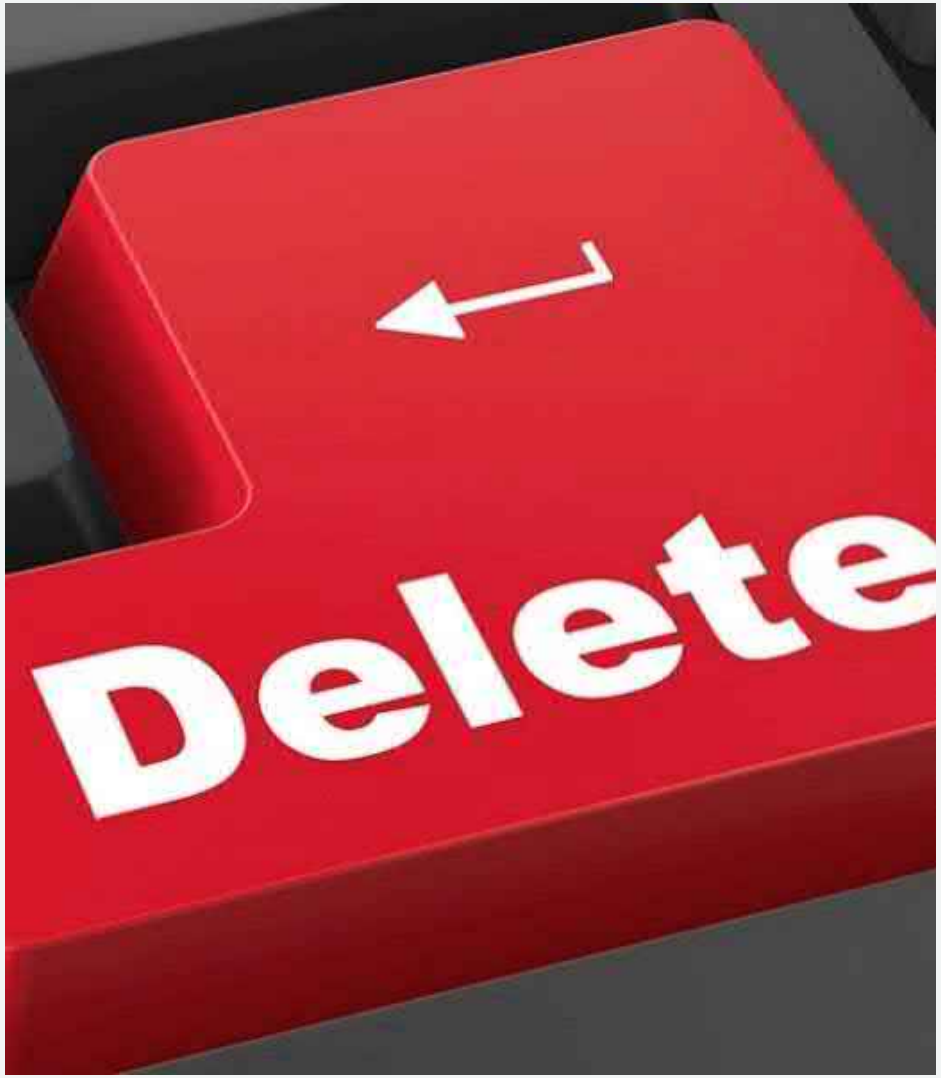
```
ROLLBACK;
rollback complete.
```

Example

Task 50. *As part of the fight against the crisis, Tiger ordered a temporary suspension of issuing additional rations of mice. Accomplish this task by appropriately modifying the* ***mice_extra*** *attribute in the* ***Cats*** *relation.*

```
UPDATE Cats

SET mice_extra=NULL

WHERE nickname<>'TIGER';


17 rows modified.
```

```
ROLLBACK;

rollback complete.
```



DELETE

DELETE FROM RelationViewName

[**WHERE** condition]

Example

Task 51. *As it turns out, however, being an enlightened ruler also has its limits. Present the highly understandable decision of the Tiger on removing his data from the* ***Extra_additions*** *relation when news arrived about of the visit of the Cat's Control Chamber.*

```
DELETE FROM Extra_additions

WHERE nickname='TIGER';

2 rows was deleted.
```

VIEWS are coming...

... but not today ☺