ORDIX® seminare

einfach. gut. geschult.

Analytische Funktionen

Analytische Funktionen in Oracle

seminare@ordix.de https://seminare.ordix.de

Agenda

ORDIX AG

- Was sind Analytische Funktionen?
- Anwendungsgebiete und Vorteile
- Syntax Analytische Funktionen
- Teilmengenextraktion mit WINDOWING-Klausel
- Analytische Funktionen und GROUP BY-Anweisung
- Anwendungsbeispiele aus der Praxis
- MATCH_RECOGNIZE (12c)
- Top N-Klausel (12c)

Übersicht Analytische Funktionen

ORDIX AG

- Eingeführt mit Oracle 8i und ständig erweitert
- Vermeidung von prozeduralem und sehr umständlichem SQL-Code
- Bessere Performance
- Analytische Funktionen ermöglichen Zugriff auf benachbarte Datensätze
- Aufteilung der Datensätze in Partitionen (ähnlich einer Gruppenverarbeitung)
- Mehrere unterschiedliche Partitionen innerhalb einer SELECT-Anweisung
- Sortierung innerhalb der Partitionen möglich
- Teilmengenextraktion (WINDOWING-Klausel)

Anwendungsgebiete und Vorteile von Analytischen Funktionen

ORDIX AG

- Anwendungsgebiete:
  - Reporting und Analyse
  - laufende Aufsummierung von Werten
  - Gleitende Durchschnitte
  - Rankings
  - Statistische Analysen (Regression)

- Vorteile:
  - Vermeidung von Self-Joins, Unterabfragen und Inline-Views
  - Kompakte Syntax
  - hohe Performance-Gewinne

Beispiel 1: Anwendungsgebiete

ORDIX AG

- Optimierung von SELECT-Abfragen mit mehrfachen Zugriffen auf dieselbe Tabelle:

```
-- Durschnittsgehalt pro Abteilung
select m1.*, m2.avg_gehalt
from tb_ma m1, (
  select abtnr, avg(gehalt) avg_gehalt from tb_ma group by abtnr
) m2 where m1.abtnr=m2.abtnr
order by manr;


-- Bestverdienende Mitarbeiter pro Abteilung
select *
from tb_ma where (abtnr, gehalt) in (
  select abtnr, max(gehalt) from tb_ma group by abtnr
) order by 1;
```

Beispiel 2: Anwendungsgebiete

ORDIX AG

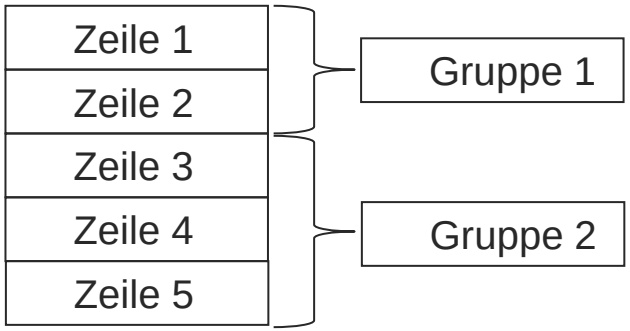
- Optimierung von SELECT-Abfragen mit mehrfachen Zugriffen auf dieselbe Tabelle:

```
-- Kumuliertes Gehalt pro Abteilung
select manr, maname, abtnr, gehalt, (
  select sum(gehalt) from tb_ma m2
  where m2.abtnr=m1.abtnr and m2.manr<=m1.manr
 ) sum_gehalt
from tb_ma m1
order by manr;

-- Zugriff auf Vorgängerdatensatz
select manr, maname, abtnr, gehalt, (
  select * from (
    select m2.gehalt from tb_ma m2 where m2.manr<m1.manr
    order by m2.manr desc
  ) where rownum<2
) gehalt_vor
from tb_ma m1 order by manr;
```

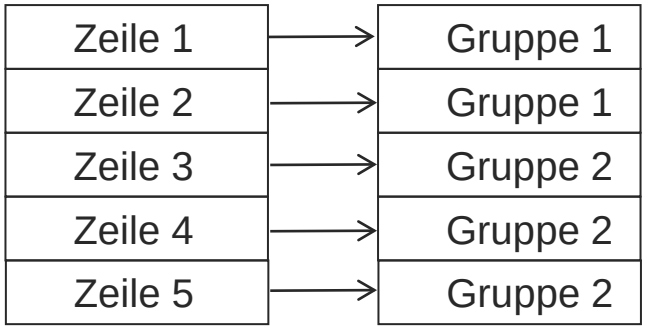
Gruppierungen versus Analytische Funktionen

ORDIX AG

- Gruppierungen (GROUP BY)



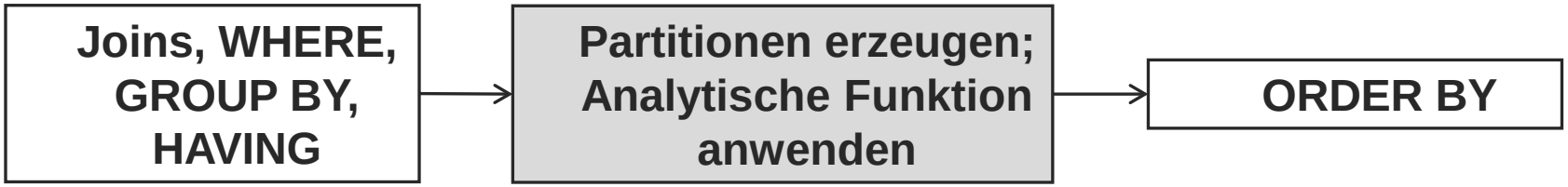
- Analytische Funktion



Besonderheiten der Analytischen Funktionen

ORDIX AG

- Analytische Funktionen sind nur in der SELECT-Liste und ORDER BY-Klausel zulässig.
- Reihenfolge der SQL-Verarbeitung:



Syntax Analytische Funktionen

ORDIX AG

```
select <spalte>,
  funktion([argumente]) OVER (
    [PARTITION BY <spalte>]
    [ORDER BY <spalte> |
     ORDER BY <spalte> WINDOWING-Klausel]
  ) ana_spalte
from <tabelle>
where ...
```

PARTITION BY

ORDIX AG

- Die PARTITION BY - Klausel dient der Gruppierung der Ergebnismenge, ähnlich der GROUP BY - Klausel bei Gruppenfunktionen.
- Alle Zeilen einer Ergebnismenge werden zusammengefasst, die dem Gruppierungskriterium entsprechen.
- Ohne Angabe der PARTITION BY - Klausel, wird die gesamte Ergebnismenge als Partition betrachtet.

ORDER BY

ORDIX AG

- Die ORDER BY - Klausel gibt eine Sortierung innerhalb einer Partition vor.
- Die Sortierung muss nicht zwingend der Reihenfolge der Ausgabe entsprechen.
- **Durch die Angabe der ORDER BY - Klausel werden nicht alle Zeilen zu einer Partition zusammengefasst, sondern nur ein bestimmtes "Fenster".**
- Das standardmäßige Fenster besteht aus allen Zeilen der jeweiligen Partition bis zur aktuellen Zeile. Dies entspricht dem Standardverhalten der WINDOWING-Klausel.

WINDOWING-Klausel

ORDIX AG

- Ein Fenster definiert den Bereich von Datensätzen auf dem eine Berechnung für die aktuelle Zeile vorgenommen werden soll (Teilmengenextraktion).
- Voraussetzung zur Verwendung der WINDOWING-Klausel ist die ORDER BY - Klausel (sortierte Ergebnismenge), d. h. mit der ORDER BY - Klausel wird die WINDOWING-Klausel aktiviert.
- Die Einschränkung des Fensters ist auf Zeilenebene mit dem Schlüsselwort ROWS und auf logischer Ebene mit RANGE möglich.

Start- und Endpunkt bei WINDOWING-Klausel

ORDIX AG

- In der WINDOWING-Klausel werden entweder Start- und Endpunkt oder nur der Startpunkt des Fensters angegeben. Wird lediglich der Startpunkt angegeben, so ist die aktuelle Zeile automatisch der Endpunkt.
- Mögliche Werte für Start- und Endpunkt:

Syntax	Beschreibung
UNBOUNDED PRECEDING	alle bisherigen Zeilen ab Beginn der Partition
<N> PRECEDING	die letzten <N> Zeilen vor der aktuellen Zeile
UNBOUNDED FOLLOWING	alle nachfolgenden Zeilen bis zum Ende der Partition
<N> FOLLOWING	die nächsten <N> Zeilen ab der aktuellen Zeile
CURRENT ROW	die aktuelle Zeile

ROWS versus RANGE

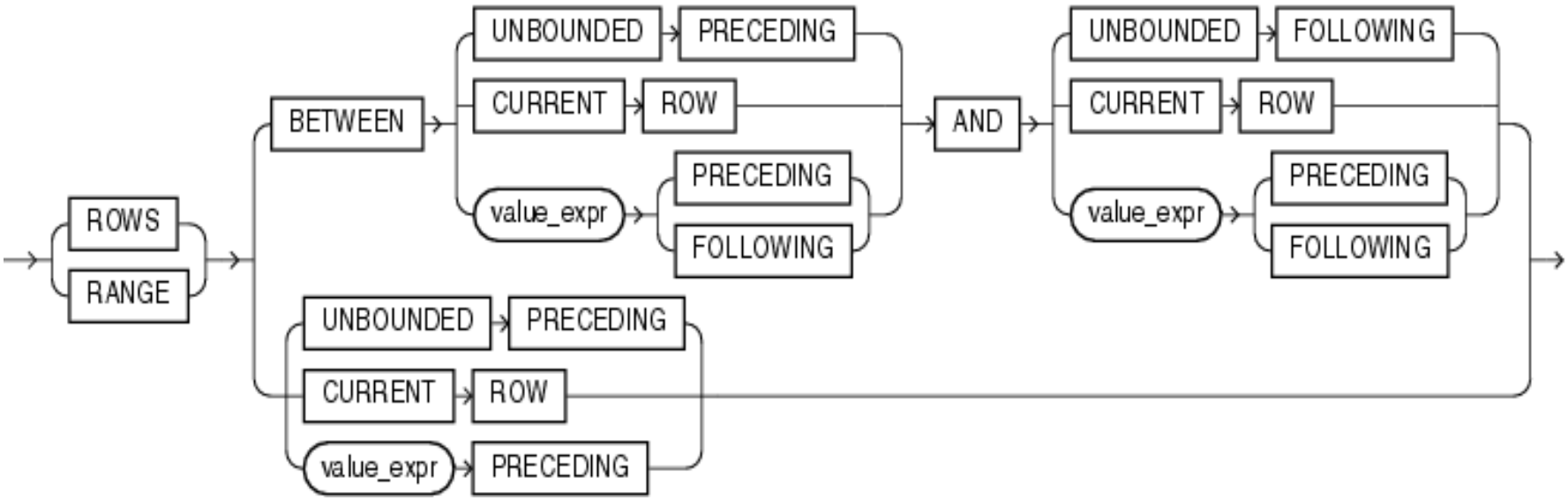
ORDIX AG

- Einschränkung des Fensters
- ROWS:
  - auf physikalischer Zeilenebene
  - Offset ist vom Datentyp NUMBER (>=0)
- RANGE:
  - auf logischer Wertebene (Angabe der Spalte über ORDER BY-Klausel)
  - Einträge die den selben Wert haben werden immer zusammen betrachtet.
  - Bei der ORDER BY-Klausel ist nur eine Spalte vom Datentyp NUMBER, DATE oder TIMESTAMP erlaubt
  - Offset ist vom Datentyp:
    - NUMBER (>=0)
    - INTERVAL YEAR TO MONTH oder INTERVAL DAY TO SECOND
  - Ein Offset kann auch eine Stored Function sein.
    HINWEIS! Damit lässt sich ein variables Fenster in Abhängigkeit von bestimmten Werten erstellen.

Syntax WINDOWING-Klausel

ORDIX AG

- Syntax der WINDOWING-Klausel:



Quelle: Oracle Dokumentation 12c

Default bei WINDOWING-Klausel

ORDIX AG

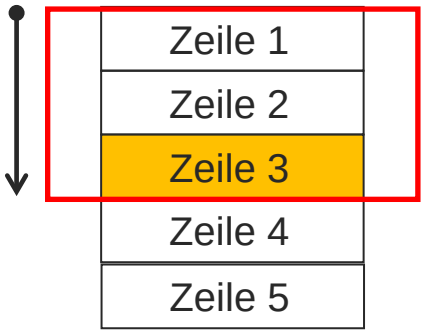
- Standardmäßig, wenn die ORDER BY - Klausel ohne WINDOWING-Klausel spezifiziert ist, besteht das Fenster aus allen Zeilen vom Start der Partition bis zur aktuellen Zeile **plus alle folgenden Zeilen, die denselben Wert** wie die aktuelle Zeile haben. Das Verhalten entspricht folgender Syntax:

  RANGE BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW

- Wird die ORDER BY - Klausel weggelassen, besteht das Fenster standardmäßig aus allen Zeilen der Partition.

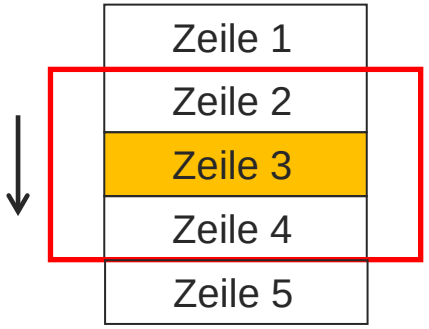
Beispiel WINDOWING-Klausel

ORDIX AG



RANGE BETWEEN
UNBOUNDED PRECEDING
AND CURRENT ROW

← Default bei Angabe von ORDER BY



ROWS BETWEEN
1 PRECEDING
AND 1 FOLLOWING

Aktuelle Zeile: → Zeile X

Beispiel WINDOWING-Klausel II

ORDIX AG

```
select tb_ma.*,
  count(*) over (
    partition by abtnr
    order by manr
    range between 1 preceding
    and 1 following
  ) cnt_abtnr
from tb_ma order by 1;

 MANR MANAME ABTNR GEHALT  CNT_ABTNR
----- ------ ----- ------ ----------
    1 ma1       10   1000          1

    2 ma2       20   2000          2
    3 ma3       20   3000          2

    4 ma4       30   4000          2
    5 ma5       30   5000          3
    6 ma6       30   5000          3
    7 ma7       30   5500          2
```

Partition

Aktueller Datensatz

Fenster

Analytische Funktionen

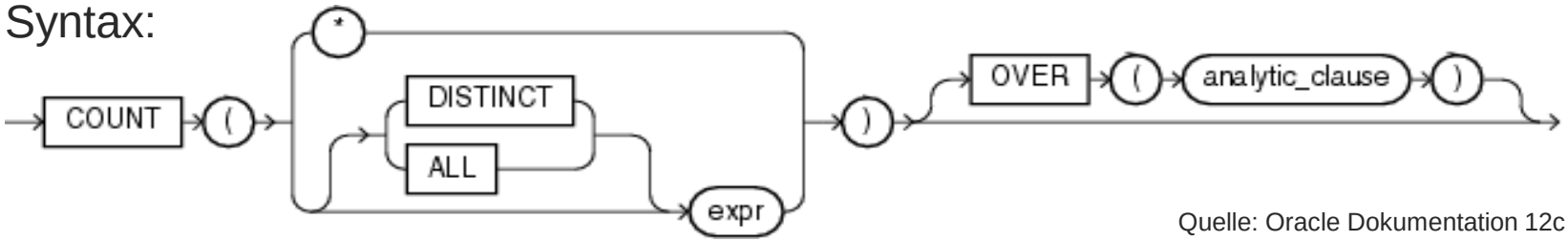
ORDIX AG

Funktion	WINDOWING-Klausel zulässig	ORDER BY-Klausel
AVG	Ja	erlaubt
COUNT	Ja	erlaubt
CUME_DIST	Nein	notwendig
DENSE_RANK	Nein	notwendig
FIRST_VALUE	Ja	erlaubt
LAG	Nein	notwendig
LAST_VALUE	Ja	erlaubt
LEAD	Nein	notwendig
LISTAGG	Nein	nicht erlaubt
MAX	Ja	erlaubt
MIN	Ja	erlaubt
NTH_VALUE	Ja	erlaubt
NTILE	Nein	notwendig
PERCENT_RANK	Nein	notwendig
RANK	Nein	notwendig
RATIO_TO_REPORT	Nein	nicht erlaubt
ROW_NUMBER	Nein	notwendig
SUM	Ja	erlaubt

COUNT

ORDIX AG

- Liefert die Anzahl der Datensätze.
- Bei der Übergabe von einem Ausdruck werden nur Datensätze berücksichtigt, bei denen der Ausdruck NOT NULL ist.
- Bei der Übergabe von * werden alle Datensätze inkl. Dubletten und NULL-Werten berücksichtigt.
- COUNT liefert kein NULL zurück.
- DISTINCT berücksichtigt nur eindeutige Werte ungleich NULL.
- Bei DISTINCT ist nur die PARTITION BY - Klausel erlaubt.
  Die ORDER BY -Klausel ist nicht zulässig.
- Bei ALL ist PARTITION BY-, ORDER BY- und WINDOWING-Klausel erlaubt.
- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel I COUNT

ORDIX AG

```
select manr, abtnr, gehalt,
  count(*)                 over ()                       cnt_ges,
  count(distinct abtnr) over ()                          cnt_dist_ges,
  count(*)                 over (partition by abtnr)     cnt_part_abtnr
from tb_ma order by 1;

 MANR ABTNR GEHALT CNT_GES CNT_DIST_GES CNT_PART_ABTNR
----- ----- ------ ------- ------------ --------------
    1    10   6000       6            3              1
    2    20   1000       6            3              2
    3    20   2000       6            3              2
    4    30   2000       6            3              3
    5    30   3000       6            3              3
    6    30   4000       6            3              3
```

Beispiel II COUNT

ORDIX AG

```
select manr, abtnr, gehalt,
  count(*) over (order by abtnr)      cnt_order_asc_ges,
  count(*) over (order by abtnr desc) cnt_order_desc_ges
from tb_ma order by 1;

 MANR ABTNR GEHALT CNT_ORDER_ASC_GES CNT_ORDER_DESC_GES
----- ----- ------ ----------------- ------------------
    1    10   6000                 1                  6
    2    20   1000                 3                  5
    3    20   2000                 3                  5
    4    30   2000                 6                  3
    5    30   3000                 6                  3
    6    30   4000                 6                  3
```

AVG

ORDIX AG

- Liefert den Durschnittswert von einem Ausdruck
- DISTINCT berücksichtigt nur eindeutige Werte ungleich NULL.
- Bei DISTINCT ist nur die PARTITION BY - Klausel erlaubt.
  Die ORDER BY - Klausel ist nicht zulässig.
- Bei ALL ist PARTITION BY-, ORDER BY- und WINDOWING-Klausel erlaubt.
- Syntax:

Quelle: Oracle Dokumentation 12c

Beispiel AVG

ORDIX AG

```
select tb_ma.*,
  avg(gehalt) over () avg_gesamt,
  avg(gehalt) over (partition by abtnr) avg_pro_abtnr
from tb_ma order by 1;

 MANR MANAME ABTNR GEHALT AVG_GESAMT AVG_PRO_ABTNR
----- ------ ----- ------ ---------- -------------
    1 ma1        10   1000       2000          1500
    2 ma2        10   2000       2000          1500
    3 ma3        20   2000       2000          2500
    4 ma4        20   3000       2000          2500
```

MAX

ORDIX AG

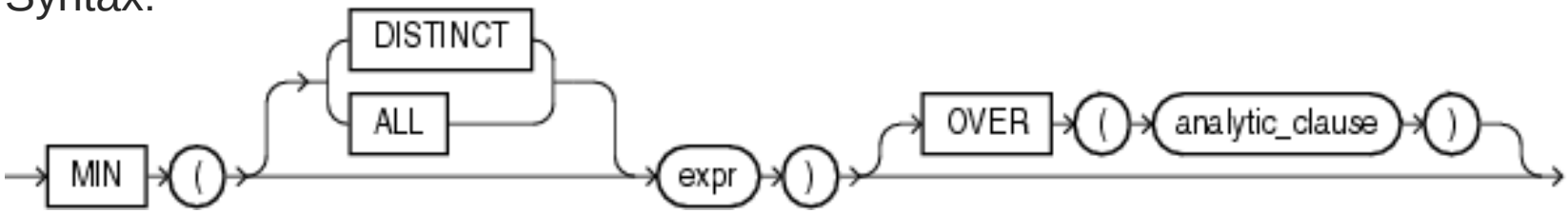
- Liefert den maximalen Wert von einem Ausdruck.
- DISTINCT berücksichtigt nur eindeutige Werte ungleich NULL.
- Bei DISTINCT ist nur die PARTITION BY - Klausel erlaubt.
  Die ORDER BY - Klausel ist nicht zulässig.
- Bei ALL sind die PARTITION BY-, die ORDER BY- und die WINDOWING-Klausel erlaubt.
- Syntax:

Quelle: Oracle Dokumentation 12c

MIN

ORDIX AG

- Liefert den minimalen Wert von einem Ausdruck.
- DISTINCT berücksichtigt nur eindeutige Werte ungleich NULL.
- Bei DISTINCT ist nur die PARTITION BY - Klausel erlaubt.
  Die ORDER BY - Klausel ist nicht zulässig.
- Bei ALL sind die PARTITION BY-, die ORDER BY- und die WINDOWING-Klausel erlaubt.
- Syntax:

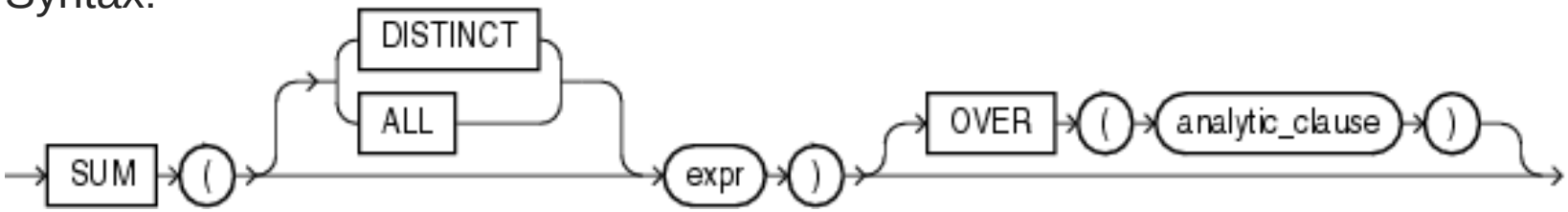


Quelle: Oracle Dokumentation 12c

SUM

ORDIX AG

- Liefert die Summe von einem Ausdruck.
- DISTINCT berücksichtigt nur eindeutige Werte ungleich NULL.
- Bei DISTINCT ist nur die PARTITION BY - Klausel erlaubt.
  Die ORDER BY - Klausel ist nicht zulässig.
- Bei ALL sind die PARTITION BY-, die ORDER BY- und die WINDOWING-Klausel erlaubt.
- Syntax:



Quelle: Oracle Dokumentation 12c

DENSE_RANK, RANK und ROW_NUMBER

ORDIX AG

- Rangfolge ermitteln
- Bei gleichem Wert geben RANK und DENSE_RANK den gleichen Rang zurück. ROW_NUMBER setzt dagegen die Zählung einfach fort.
- Der Unterschied zwischen RANK und DENSE_RANK besteht darin, dass RANK nach gleichen Werten eine Lücke in der Rangfolge lässt, DENSE_RANK dagegen nicht.
- Keine Übergabeparameter
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.

Funktion	Beschreibung
DENSE_RANK()	Liefert einen Rang (bei gleichen Werten entsteht bei Rangfolge keine Lücke)
RANK()	Liefert einen Rang (bei gleichen Werten entsteht bei Rangfolge eine Lücke)
ROW_NUMBER()	Liefert eine Zeilennummer

Beispiel DENSE_RANK, RANK und ROW_NUMBER

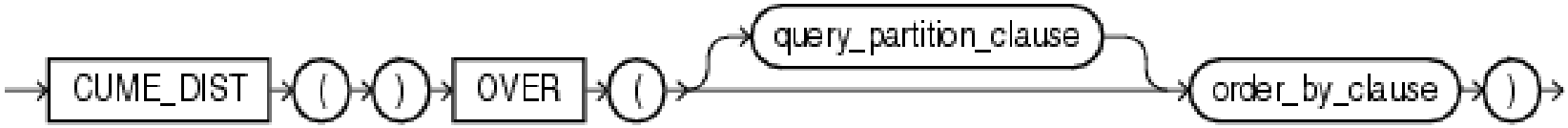
ORDIX AG

```
select tb_ma.*,
  dense_rank() over (order by gehalt) dense_rank,
  rank()       over (order by gehalt) rank,
  row_number() over (order by gehalt) row_number
from tb_ma order by 1;

 MANR MANAME ABTNR GEHALT DENSE_RANK       RANK ROW_NUMBER
----- ------ ----- ------ ---------- ---------- ----------
    1 ma1       10   1000          1          1          1
    2 ma2       20   2000          2          2          2
    3 ma3       20   2000          2          2          3
    4 ma4       20   3000          3          4          4
```

CUME_DIST

ORDIX AG

- Ermittelt eine kumulative Werteverteilung anhand der Zeilenrangfolge.
- Keine Übergabeparameter
- Zurückgeliefert wird der Anteil der Zeilen, die kleiner oder gleich dem jeweiligen Wert des ORDER BY - Ausdrucks sind.
- Der Rückgabewert liegt zwischen 0 und 1 (>0, <=1).
- Gleiche Werte werden derselben höchsten kumulativen Werteverteilung zugeordnet.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.
- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel CUME_DIST

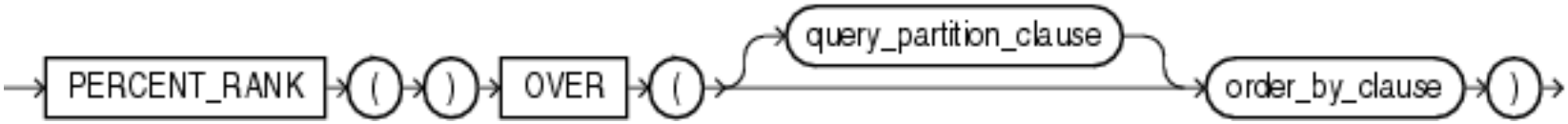
ORDIX AG

```
select manr, abtnr, gehalt,
  cume_dist() over (order by manr)   cd_manr,
  cume_dist() over (order by abtnr)  cd_abtnr,
  cume_dist() over (order by gehalt) cd_gehalt
from tb_ma order by 1;

 MANR ABTNR GEHALT    CD_MANR   CD_ABTNR  CD_GEHALT
----- ----- ------ ---------- ---------- ----------
    1    10   1000         ,2         ,4         ,2
    4    10   2000         ,4         ,4         ,6
    5    20   2000         ,6          1         ,6
    7    20   4000         ,8          1         ,8
    8    20   5500          1          1          1
```

PERCENT_RANK

ORDIX AG

- Ermittelt eine kumulative Werteverteilung anhand der Zeilenrangfolge.
- Keine Übergabeparameter
- Zurückgeliefert wird der Anteil der Zeilen, die kleiner oder gleich dem jeweiligen Wert des ORDER BY-Ausdrucks sind.
- Der Rückgabewert liegt zwischen 0 und 1 (>=0, <=1).
- Gleiche Werte werden derselben niedrigsten kumulativen Werteverteilung zugeordnet.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.
- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel PERCENT_RANK

ORDIX AG

```
select manr, abtnr, gehalt,
  percent_rank() over (order by manr)   pr_manr,
  percent_rank() over (order by abtnr)  pr_abtnr,
  percent_rank() over (order by gehalt) pr_gehalt
from tb_ma order by 1;

 MANR ABTNR GEHALT    CD_MANR   CD_ABTNR  CD_GEHALT
----- ----- ------ ---------- ---------- ----------
    1    10   1000          0          0          0
    4    10   2000        ,25          0        ,25
    5    20   2000         ,5         ,5        ,25
    7    20   4000        ,75         ,5        ,75
    8    20   5500          1         ,5          1
```

NTILE

ORDIX AG

- Teilt einen Wertebereich in gleich große Buckets auf.
- Die Anzahl der Buckets wird als Parameter übergeben.
- Der Rückgabewert ist die Nummer des Buckets, zu dem der Wert gehört. Gleiche Werte können dabei auch unterschiedlichen Buckets zugeordnet werden.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.

- Syntax:

Quelle: Oracle Dokumentation 12c

Beispiel NTILE

ORDIX AG

```
select manr, abtnr, gehalt,
  ntile(2) over (order by manr)   ntile_2_manr,
  ntile(3) over (order by abtnr)  ntile_3_abtnr,
  ntile(7) over (order by gehalt) ntile_7_gehalt
from tb_ma order by 1;

MANR ABTNR GEHALT NTILE_2_MANR NTILE_3_ABTNR NTILE_7_GEHALT
---- ----- ------ ------------ ------------- -------------
   1    10   1000            1             1             1
   2    10   2000            1             1             2
   3    20   3000            2             2             3
   4    20   4000            2             3             4
```

RATIO_TO_REPORT

ORDIX AG

- Anteil des Wertes an der Gesamtsumme
- Der Übergabeparameter bestimmt, wonach der Anteil berechnet werden soll.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.

- Syntax:

Quelle: Oracle Dokumentation 12c

Beispiel RATIO_TO_REPORT

ORDIX AG

```
select manr, gehalt,
  ratio_to_report(manr)   over () ratio_to_report_manr,
  ratio_to_report(gehalt) over () ratio_to_report_gehalt
from tb_ma order by 1;

  MANR     GEHALT RATIO_TO_REPORT_MANR RATIO_TO_REPORT_GEHALT
------ ---------- -------------------- ----------------------
     1       1000           ,083333333                   ,125
     2       3000           ,166666667                   ,375
     3       2000                  ,25                    ,25
     6       2000                   ,5                    ,25
```

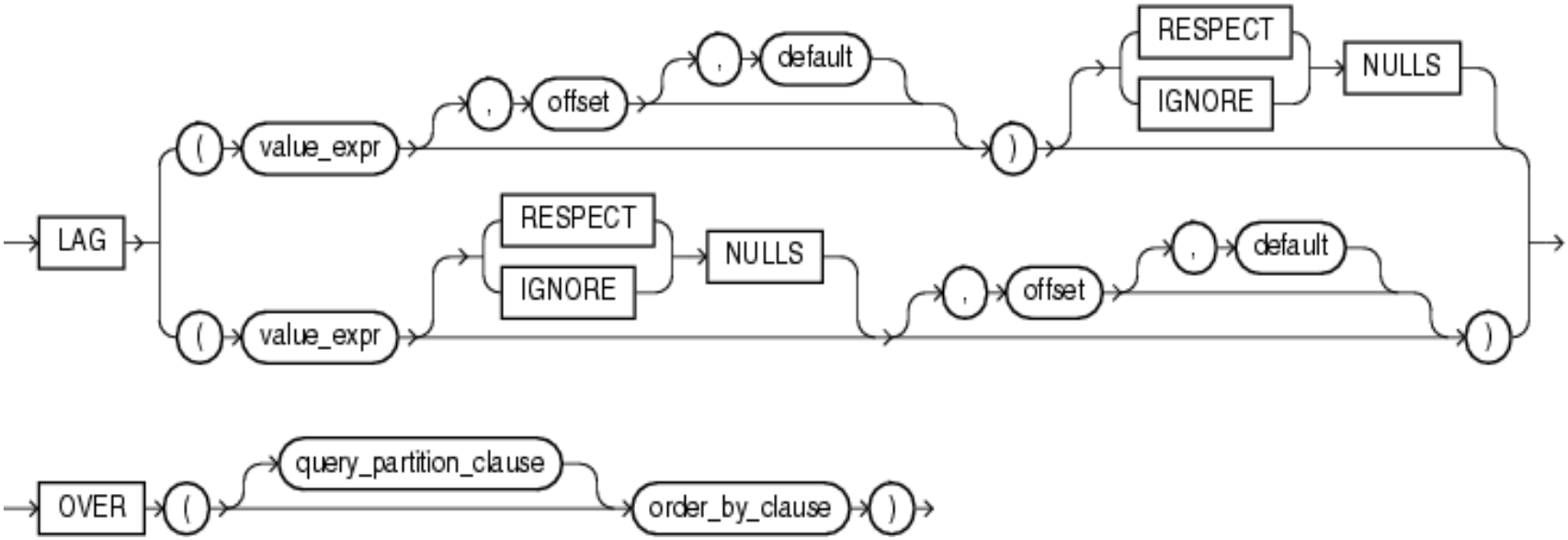
LAG

ORDIX AG

- Mit LAG kann ausgehend vom aktuellen Datensatz auf vorhergehende Datensätze der Ergebnismenge zugegriffen werden.
- Es ist kein Self-Join notwendig.
- Zugriff auf vorhergehende Datensätze über offset Parameter (Default 1).
- Angabe vom Default-Wert möglich, falls der Fensterbereich überschritten wurde (Default-Wert ist NULL).
- NULL-Werte können über die Angabe RESPECT NULLS (Default) berücksichtigt oder über die Angabe IGNORE NULLS ignoriert werden.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.

Syntax LAG

ORDIX AG



Quelle: Oracle Dokumentation 12c

Beispiel LAG

ORDIX AG

```
select manr, maname, gehalt,
  lag(gehalt)              over (order by manr) lag,
  lag(gehalt, 1, 0)        over (order by manr) lag_default,
  lag(gehalt) ignore nulls over (order by manr) lag_ignore_nulls
from tb_ma order by 1;

 MANR MANAME GEHALT        LAG LAG_DEFAULT LAG_IGNORE_NULLS
----- ------ ------ ---------- ----------- ---------------
    1 ma1      1000                     0
    2 ma2      2000       1000        1000            1000
    3 ma3      3000       2000        2000            2000
    4 ma4                 3000        3000            3000
    5 ma5      5000                                   3000
    6 ma6      6000       5000        5000            5000
```

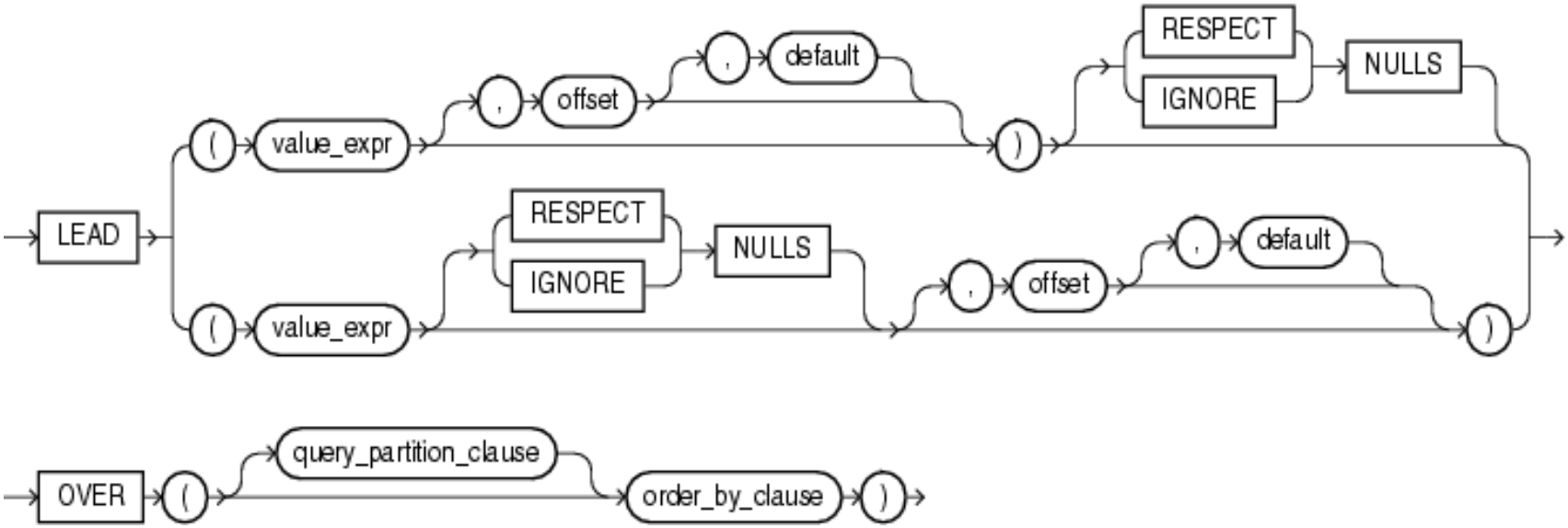
LEAD

ORDIX AG

- Mit LEAD kann ausgehend vom aktuellen Datensatz auf nachfolgende Datensätze der Ergebnismenge zugegriffen werden.
- Es ist kein Self-Join notwendig.
- Zugriff auf nachfolgende Datensätze über offset Parameter (Default 1).
- Angabe vom Default-Wert möglich, falls der Fensterbereich überschritten wurde (Default-Wert ist NULL).
- NULL-Werte können über die Angabe RESPECT NULLS (Default) berücksichtigt oder über die Angabe IGNORE NULLS ignoriert werden.
- Die ORDER BY - Klausel ist notwendig, die WINDOWING-Klausel ist dagegen nicht zulässig.

Syntax LEAD

ORDIX AG



Quelle: Oracle Dokumentation 12c

Beispiel LEAD

ORDIX AG

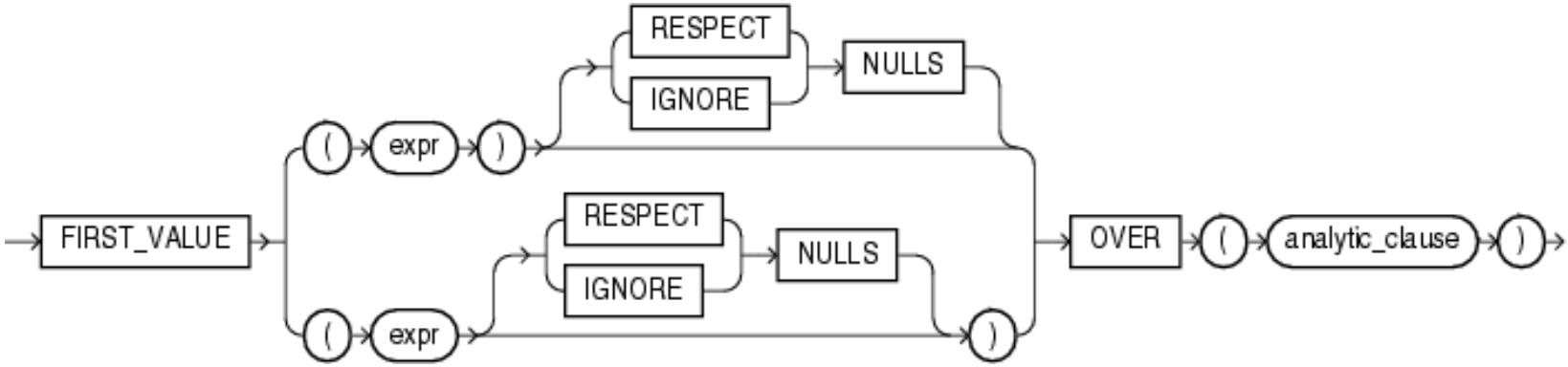
```
select manr, maname, gehalt,
 lead(gehalt)              over (order by manr) lead,
 lead(gehalt, 1, 0)        over (order by manr) lead_default,
 lead(gehalt) ignore nulls over (order by manr) lead_ignore_nulls
from tb_ma order by 1;

MANR MANAME GEHALT       LEAD LEAD_DEFAULT LEAD_IGNORE_NULLS
----- ------ ------ ---------- ------------ ----------------
    1 ma1      1000       2000         2000             2000
    2 ma2      2000       3000         3000             3000
    3 ma3      3000                                     5000
    4 ma4                 5000         5000             5000
    5 ma5      5000       6000         6000             6000
    6 ma6      6000                       0
```

FIRST_VALUE

ORDIX AG

- Liefert den ersten Wert in der Partition
- NULL-Werte können über die Angabe RESPECT NULLS (Default) berücksichtigt oder über die Angabe IGNORE NULLS ignoriert werden.
- ORDER BY-Klausel ist erlaubt und die WINDOWING-Klausel ist zulässig.

- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel FIRST_VALUE

ORDIX AG

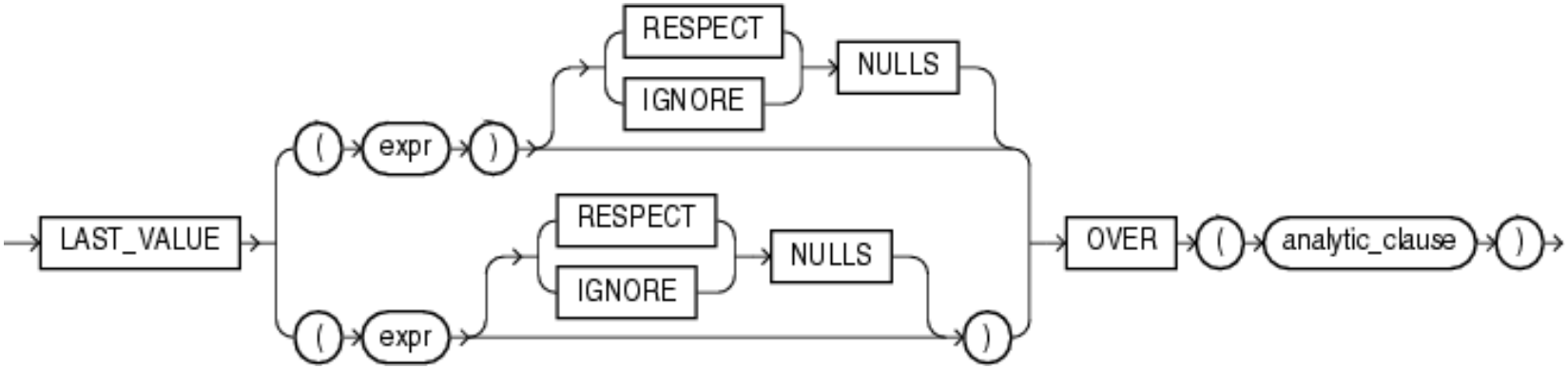
```
select manr, abtnr, gehalt,
  first_value(gehalt) over (order by manr)      fv_1,
  first_value(gehalt) over (order by manr desc) fv_2,
  first_value(gehalt ignore nulls) over (
    partition by abtnr order by manr
    rows between unbounded preceding and unbounded following
  ) fv_3
from tb_ma order by 1;

 MANR ABTNR GEHALT       FV_1       FV_2       FV_3
----- ----- ------ ---------- ---------- ----------
    1    10   3000       3000       4000       3000
    2    20   3200       3000       4000       3200
    3    20   3500       3000       4000       3200
    4    30   3600       3000       4000       3600
    5    30   3800       3000       4000       3600
    6    30   4000       3000       4000       3600
    7    30   4000       3000       4000       3600
```

LAST_VALUE

ORDIX AG

- Liefert den letzten Wert in der Partition
- NULL-Werte können über die Angabe RESPECT NULLS (Default) berücksichtigt oder über die Angabe IGNORE NULLS ignoriert werden.
- ORDER BY-Klausel ist erlaubt und die WINDOWING-Klausel ist zulässig.

- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel LAST_VALUE

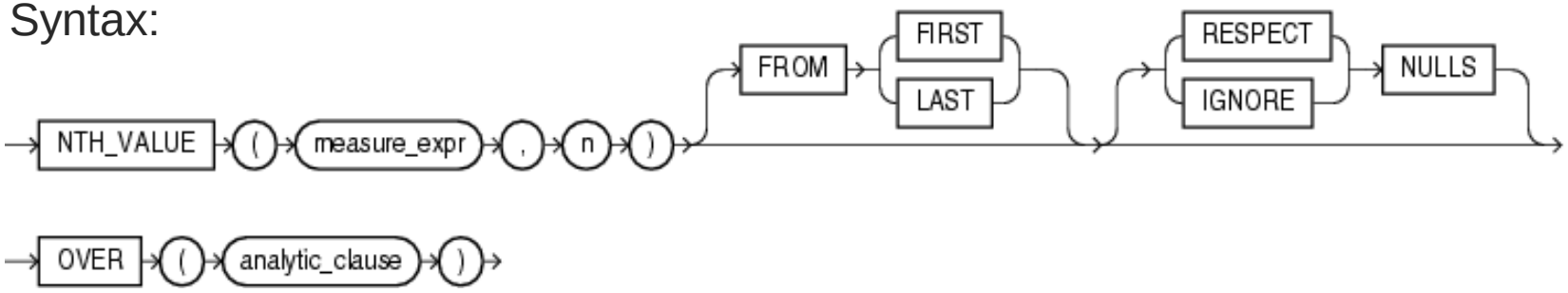
ORDIX AG

```
select manr, abtnr, gehalt,
  last_value(gehalt) over (order by manr) lv_1,
  last_value(gehalt) over (order by manr desc) lv_2,
  last_value(gehalt ignore nulls) over (
    partition by abtnr order by manr
    rows between unbounded preceding and unbounded following
  ) lv_3
from tb_ma order by 1;

 MANR ABTNR GEHALT       LV_1       LV_2       LV_3
----- ----- ------ ---------- ---------- ----------
    1    10   3000       3000       3000       3000
    2    20   3200       3200       3200       3500
    3    20   3500       3500       3500       3500
    4    30   3600       3600       3600       4000
    5    30   3800       3800       3800       4000
    6    30   4000       4000       4000       4000
    7    30   4000       4000       4000       4000
```

NTH_VALUE

ORDIX AG

- Liefert n-ten Wert in der Partition.
- n wird als zweiter Übergabeparameter übergeben. n muss ein positiver Wert >0 sein, sonst wird eine Exception geliefert.
- Reihenfolge kann mit FROM FIRST und FROM LAST bestimmt werden. Default ist FROM FIRST.
- NULL-Werte können über die Angabe RESPECT NULLS (Default) berücksichtigt oder über die Angabe IGNORE NULLS ignoriert werden.
- ORDER BY-Klausel ist erlaubt und die WINDOWING-Klausel ist zulässig.
- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel NTH_VALUE

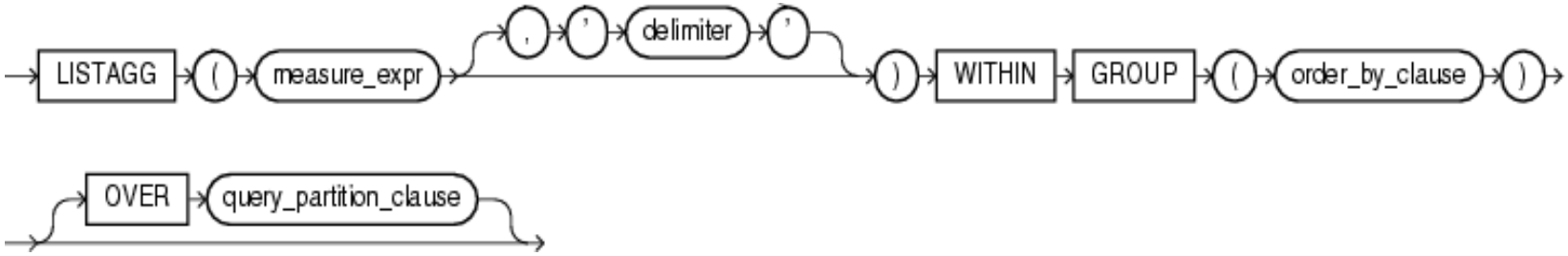
ORDIX AG

```
select t.*,
  nth_value(gehalt, 2) over (
    partition by abtnr order by manr
    rows between unbounded preceding and unbounded following
  ) nth_value_part_order_by
from tb_ma t order by 1;

 MANR MANAME ABTNR GEHALT NTH_VALUE_PART_ORDER_BY
----- ------ ----- ------ -----------------------
    1 ma1       10   1000                    2000
    2 ma2       10   2000                    2000
    3 ma3       20   3000                    4000
    4 ma4       20   4000                    4000
    5 ma5       20   5000                    4000
    6 ma6       20   6000                    4000
```

LISTAGG

ORDIX AG

- Erzeugt eine Liste mit Werten.
- NULL-Werte werden ignoriert.
- Trennzeichen ist ein String (Default NULL).
- Rückgabewert ist VARCHAR2 (bei RAW wird RAW zurück geliefert).
- ORDER BY- und WINDOWING-Klausel sind in der OVER-Klausel nicht erlaubt.
- Syntax:



Quelle: Oracle Dokumentation 12c

Beispiel LISTAGG

ORDIX AG

```
select manr, maname, abtnr, gehalt,
  listagg(maname, ';' ) WITHIN GROUP (ORDER BY manr     )
    over (partition by abtnr) ma_list_asc,
  listagg(maname, '; ') WITHIN GROUP (ORDER BY manr desc)
    over (partition by abtnr) ma_list_desc
from tb_ma order by 1;

 MANR MANAME ABTNR GEHALT MA_LIST_ASC     MA_LIST_DESC
----- ------ ----- ------ --------------- -------------
    1 ma1        10   1000 ma1             ma1
    2 ma2        20   2000 ma2;ma3         ma3; ma2
    3 ma3        20   3000 ma2;ma3         ma3; ma2
    4 ma4        30   4000 ma4;ma5;ma6     ma6; ma5; ma4
    5 ma5        30   5000 ma4;ma5;ma6     ma6; ma5; ma4
    6 ma6        30   5000 ma4;ma5;ma6     ma6; ma5; ma4
```

Analytische Funktionen und GROUP BY-Anweisung

ORDIX AG

- Analytische Funktionen können auch zusammen mit einer GROUP BY-Anweisung verwendet werden.
- In der Analytischen Funktion Syntax sind nur Aggregate und Aggregatsfunktionen erlaubt.

Beispiel Analytische Funktionen und GROUP BY-Anweisung I

ORDIX AG

```
select abtnr, sum(gehalt) sum_gehalt
  -- Groesste Abteilungsnummer
  , max(abtnr) over () max_abtnr_ana
  -- Anzahl Eintraege Gesamt nach der GROUP BY-Anweisung
  , count(*) over () count_ges
  -- Anzahl Eintraege pro Abteilung nach der GROUP BY-Anweisung
  , count(*) over (partition by abtnr) count_part
from tb_ma
group by abtnr order by 1;

     ABTNR SUM_GEHALT MAX_ABTNR_ANA  COUNT_GES COUNT_PART
---------- ---------- ------------- ---------- ----------
        10       1000            20          2          1
        20       5000            20          2          1
```

Beispiel Analytische Funktionen und GROUP BY-Anweisung II

ORDIX AG

```
select abtnr, sum(gehalt) sum_gehalt
  -- Liefert das groesste Gruppensummengehalt. Analytische Fkt.
  -- bezieht sich auf das Resultset vom GROUP BY.
  , max(sum(gehalt)) over () max_sum_gehalt_ana
  -- Liefert das Gesamtsummengehalt. Analytische Fkt.
  -- bezieht sich auf das Resultset vom GROUP BY.
  , sum(sum(gehalt)) over () sum_sum_gehalt_ana
from tb_ma
group by abtnr order by 1;

ABTNR SUM_GEHALT MAX_SUM_GEHALT_ANA SUM_SUM_GEHALT_ANA
----- ---------- ------------------ ------------------
   10       1000              14000              20000
   20       5000              14000              20000
   30      14000              14000              20000
```

Beispiel Analytische Funktionen und GROUP BY-Anweisung III

ORDIX AG

```
select abtnr, sum(gehalt) sum_gehalt
  -- ist gleichbedeutend mit sum(gehalt)
  , sum(sum(gehalt)) over (partition by abtnr)
    sum_sum_gehalt_part_ana
from tb_ma
group by abtnr order by 1;

ABTNR SUM_GEHALT SUM_SUM_GEHALT_PART_ANA
----- ---------- ------------------------
   10       1000                     1000
   20       5000                     5000
   30      14000                    14000
```

Beispiel Analytische Funktionen und GROUP BY-Anweisung IV

ORDIX AG

```
-- ORA-00979: Kein GROUP BY-Ausdruck
select abtnr, sum(gehalt) sum_gehalt,
  count(*) over (partition by gehalt)
from tb_ma
group by abtnr order by 1;


 count(*) over (partition by gehalt)
                                *
FEHLER in Zeile 2:
ORA-00979: Kein GROUP BY-Ausdruck
```

Anwendungsbeispiele I

ORDIX AG

1. a) Ermitteln Sie aus der Mitarbeitertabelle TB_MA(MANR, MANAME, ABTNR, GEHALT) die bestverdienenden Mitarbeiter pro Abteilung. Geben Sie dabei die Mitarbeiternummer, die Abteilungsnummer, den Namen und das Gehalt der bestverdienenden Mitarbeiter pro Abteilung aus. Bei gleichem Gehalt soll der Mitarbeiter mit der niedrigen MA-Nr ausgegeben werden.
   b) Erstellen Sie eine aufsteigende Liste aller Mitarbeiter. Geben Sie zu jedem Mitarbeiter folgendes aus:
   - das Durchschnittsgehalt seiner Abteilung,
   - die Gehaltsdifferenz zu seinem Vorgänger,
   - die Gehaltsdifferenz zu seinem Nachfolger.
2. Ermitteln Sie anhand des Änderungszeitstempel (Spalte ZEIT) den jüngsten Datensatz pro Mitarbeiter aus der Mitarbeitertabelle TB_MA(MANR, MANAME, ABTNR, GEHALT, ZEIT) mit mehreren Versionen der Mitarbeiterdaten. Löschen Sie anschließend alle doppelten alten Einträge.

Anwendungsbeispiele II

ORDIX AG

3. Ermitteln Sie aus der Tabelle Gruppenmitglieder TB_GRUPPE_MTGL(NR, NAME, GRUPPE, TYP, UMSATZ) folgendes:

   a) Für jedes Mitglied soll die Anzahl seiner Gruppenmitglieder ermittelt werden.

   b) Für jedes Mitglied soll die Anzahl seiner A-, B- und C-Gruppenmitglieder ermittelt werden.

   c) Es sollen alle Mitglieder einer Gruppe ermittelt werden, die ein A-Mitglied haben.

   d) Für jedes Mitglied soll ein neuer Typ seiner Gruppe ermittelt werden. Der Typ wird in Abhängigkeit des Gruppenmitgliederumsatzes festgelegt:

   ```
   Typ A: Umsatz >= 3000
   Typ B: Umsatz < 3000 und >1000
   Typ C: Umsatz <= 1000
   ```

Anwendungsbeispiele III

ORDIX AG

4. Ermitteln Sie aus der Umsatztabelle TB_UMSATZ(JAHR, MONAT, UMSATZ) folgendes:

   a) Neben dem Umsatz pro Monat soll jeweils der gesamte Jahresumsatz bis zum entsprechenden Monat aufsteigend nach Monat angezeigt werden (kumulierte Summierung).

   b) Neben dem Gesamtumsatz pro Jahr soll jeweils die prozentuale Veränderung gegenüber dem Vorjahresumsatz angezeigt werden (kumulierte Summierung).

5. Ermitteln Sie aus der Mitarbeitertabelle TB_MA alle Mitarbeiter, die seit der letzten Tabellenreorganisation gelöscht wurden, mit dem Ziel, Lücken in der Tabelle zu finden. Verwenden Sie dabei die Funktion dbms_rowid.rowid_row_number(rowid).

Anwendungsbeispiele IV

ORDIX AG

6. Ermitteln Sie aus der Log-Tabelle TB_LOG(NR, TEXT, ZEIT) mit einem Zeitstempel:

   a) Dauer der einzelnen Verarbeitungsschritte

   b) Welcher Schritt hat am längsten gedauert?

   c) Verarbeitungsdauer pro Kunde (Kunde steht in der Spalte TEXT)

7. Datensätze aus der TB_MA_STG-Tabelle sollen in die TB_MA-Tabelle übertragen werden. Dabei gelten bei Dubletten folgende Regeln:

   a) Alle Datensätze mit doppelten PK in TB_MA_ERR-Tabelle eintragen

   b) Beim gleichen PK und gleichen Datenteil, soll nur ein Datensatz in die TB_MA-Tabelle eingefügt werden.

   Tabellenaufbau:

   - TB_MA_STG(RUN_ID, MANR, MANAME, ABTNR, GEHALT)
   - TB_MA(RUN_ID, STICHTAG, MANR, MANAME, ABTNR, GEHALT)

Anwendungsbeispiele V

ORDIX AG

8. Ermitteln Sie aus der Status-Tabelle TB_STATUS(COL_LFDNR, COL_JOBNAME, COL_START, COL_ENDE) mit Start- und Ende-Zeitstempel, wie viele Jobs parallel laufen, bzw. wie viele Aktivitäten gleichzeitig statt finden.

9. Ermitteln Sie aus den AWR-Tabellen:

   a) Wartezustand "db file sequential read" (DBA_HIST_SYSTEM_EVENT)

   b) Verhältnis von logischen zu physischen I/O „Buffer Hit Ratio“ (DBA_HIST_SYSSTAT) anhand der folgenden Formel:

   [Buffer Hit Ratio] = 100 - [physical reads] * 100 / ([db block gets] + [consistent gets])

Anwendungsbeispiele VI

ORDIX AG

10. a) Ermitteln Sie aus der Mitarbeitertabelle TB_MA(MANR, MANAME, GEHALT, GUELTIG_AB, GUELTIG_BIS) die Änderungen an den Datensätzen. Diese Tabelle beinhaltet historische Daten und verfügt über eine GUELTIG_AB und GUELTIG_BIS Tabellenspalte. Dabei soll der folgende Zeitraum betrachtet werden:
    - Deltazeitraum: 05.11.2016 - 06.11.2016 (TEV)

    Zusätzlich soll ein DML-Status mit der Kennzeichnung der Änderung ermittelt werden:
    - DML-Status: I-INSERT, U-UPDATE und D-DELETE

    b) Erweitern Sie die Delta-Verarbeitung um die Ermittlung von sogenannten deltainduzierenden Spalten. Eine deltainduzierende Spalte ist eine Spalte, auf der Änderungen bei UPDATE-Operationen untersucht werden. Der DML-Status U-UPDATE wird dabei nur dann gesetzt, wenn sich nur in den deltainduzierenden Spalten etwas geändert hat.
    Folgende Spalte soll als deltainduzierende Spalte implementiert werden:

    - MANAME

Übersicht Mustererkennung ab Oracle12c

ORDIX AG

- Neue SQL-Mustererkennung-Klausel MATCH_RECOGNIZE über Datensätze hinaus
- Angabe nach dem Tabellennamen in der SELECT-FROM-Klausel
- Keine Funktion
- Zeilenübergreifend
- Deklarative Musterbeschreibung mit Regulären Ausdrücken
- Variablendefinition für Musterbeschreibung möglich
- Anwendungsbeispiele:
  - Muster in Logdateien, Sensordaten
  - Muster in Prozessketten
  - Muster in Preisentwicklung, Kaufgewohnheiten

Syntax MATCH_RECOGNIZE

ORDIX AG

```
SELECT <COL> FROM <TABLE>

MATCH_RECOGNIZE (

  [ ROW_PATTERN_PARTITION_BY ]

  [ ROW_PATTERN_ORDER_BY ]

  [ ROW_PATTERN_MEASURES ]

  [ ROW_PATTERN_ROWS_PER_MATCH ]

  [ ROW_PATTERN_SKIP_TO ]

  PATTERN (ROW_PATTERN)

    [ ROW_PATTERN_SUBSET_CLAUSE ]

  DEFINE ROW_PATTERN_DEFINITION_LIST

)
```

◀ 1. Aufteilung der Daten in Partitionen

◀ 2. Sortierung der Daten innerhalb der Partitionen

◀ 5. Ausgabe-Felder und Berechnungen

◀ 6. Rückgabe der Zeilen (eine Zeile oder alle Zeilen pro Mustertreffer)

◀ 7. Wiederaufnahme der Suche nach einem Mustertreffer

◀ 3. Musterdefinition

◀ 4. Mustervariablen

ROW_PATTERN_PARTITION_BY

ORDIX AG

- Aufteilung der Daten in Partitionen
- Optional
- Wenn nicht angegeben, dann ist die gesamte Ergebnismenge eine große Partition
- Ähnlich der Analytischen Funktionen Syntax
- Syntax:

```
row_pattern_partition_by ::=
  PARTITION BY column[, column]...
```

ROW_PATTERN_ORDER_BY

ORDIX AG

- Sortierung der Daten innerhalb der Partitionen
- Optional
- Sollte immer angegeben werden, sonst nicht deterministische Ergebnisse
- Ruft nicht zwingend zusätzliche Sortierungen hervor
- Ähnlich der Analytischen Funktionen Syntax
- Syntax:

```
row_pattern_order_by ::=
  ORDER BY column[, column]...
```

ROW_PATTERN_MEASURES

ORDIX AG

- Ausgabe-Felder und Berechnungen
- Optional
- Syntax:

```
row_pattern_measures ::=
  MEASURES row_pattern_measure_column
         [,row_pattern_measure_column]...

row_pattern_measure_column ::= expression AS c_alias
```

- Beispiel:

```
MEASURES
  avg(gehalt) as avg_gehalt,
  sum(gehalt) as sum_gehalt
```

Ausdrücke in MEASURES

ORDIX AG

- Fenster der Ergebnismenge (Teilmengenextraktion):
  - RUNNING (**Default**)
    - alle Zeilen vom Start der Partition bis zur aktuellen Zeile
  - FINAL
    - alle Zeilen der Partition
- Beispiel:

```
MEASURES
  running avg(gehalt) as run_avg_gehalt,
  final   avg(gehalt) as fin_avg_gehalt
```

ROW_PATTERN_ROWS_PER_MATCH I

ORDIX AG

- Rückgabe der Zeilen
- Optional
- ONE ROW PER MATCH **(Default)**
  - Pro Mustertreffer wird nur die letzte Zeile ausgegeben.
  - Es sind nur PARTITION BY - und MEASURES-Spalten sichtbar, d. h. nur diese Spalten können in SELECT-, WHERE- und der äußeren ORDER BY - Klausel verwendet werden.
- ALL ROWS PER MATCH
  - Es werden alle zu dem Muster passenden Zeilen ausgegeben, inkl. Zeilen, die einem leeren Muster entsprechen.
  - Alternative Syntax: ALL ROWS PER MATCH SHOW EMPTY MATCHES

ROW_PATTERN_ROWS_PER_MATCH II

ORDIX AG

- ALL ROWS PER MATCH OMIT EMPTY MATCHES
  - Es werden alle zu dem Muster passenden Zeilen ausgegeben, aber ohne Zeilen, die einem leeren Muster entsprechen.
- ALL ROWS PER MATCH WITH UNMATCHED ROWS
  - Es werden immer alle Zeilen ausgegeben.
  - „Ausschluss von der Ausgabe“-Syntax {- … -} ist hier nicht erlaubt.

ROW_PATTERN_SKIP_TO

ORDIX AG

- Wiederaufnahme der Suche nach einem Mustertreffer
- Optional
- Mögliche Werte:
  - AFTER MATCH SKIP TO NEXT ROW
    - Wiederaufnahme der Suche nach der ersten Zeile des aktuellen Mustertreffers.
  - AFTER MATCH SKIP PAST LAST ROW **(Default)**
    - Wiederaufnahme der Suche nach der letzten Zeile des aktuellen Mustertreffers.
  - AFTER MATCH SKIP TO FIRST <pattern_variable>
    - Wiederaufnahme der Suche ab der ersten Zeile der <pattern_variable> Mustervariable.
  - AFTER MATCH SKIP TO LAST <pattern_variable>
    - Wiederaufnahme der Suche ab der letzten Zeile der <pattern_variable> Mustervariable.
  - AFTER MATCH SKIP TO <pattern_variable> = AFTER MATCH SKIP TO LAST <pattern_variable>

Syntax ROW_PATTERN_SKIP_TO

ORDIX AG

- Syntax:

```
row_pattern_skip_to ::=
   AFTER MATCH {
     SKIP TO NEXT ROW
   | SKIP PAST LAST ROW
   | SKIP TO FIRST pattern_variable
   | SKIP TO LAST pattern_variable
   | SKIP TO pattern_variable
 }
```

Quelle: Oracle Dokumentation 12c

PATTERN

ORDIX AG

- Musterdefinition
- Obligatorisch, aber SUBSET-Klausel ist optional
- Variable ohne Definition in der DEFINE-Klausel entspricht jeder Zeile
  - Startbeschreibung eines Musters
- Syntax:

```
PATTERN (ROW_PATTERN) [ ROW_PATTERN_SUBSET_CLAUSE]
```

Syntax PATTERN

ORDIX AG

```
row_pattern ::= row_pattern_term | row_pattern "|" row_pattern_term

row_pattern_term ::= row_pattern_factor | row_pattern_term
   row_pattern_factor

row_pattern_factor ::= row_pattern_primary [row_pattern_quantifier]

row_pattern_primary ::= variable_name |$ |^ |([row_pattern])
   |"{-" row_pattern"-}" | row_pattern_permute

row_pattern_quantifier ::= *[?] |+[?] |?[?]
   |"{"[unsigned_integer ],[unsigned_integer]"}"[?]
   |"{"unsigned_integer "}"

row_pattern_permute ::= PERMUTE (row_pattern [, row_pattern] ...)

row_pattern_subset_clause ::= SUBSET row_pattern_subset_item [,
   row_pattern_subset_item] ...

row_pattern_subset_item ::= variable_name = (variable_name[ ,
   variable_name]...)
```

Quelle: Oracle Dokumentation 12c

Quantifier PATTERN

ORDIX AG

- Die Quantifier werden mit der Reguläre-Ausdrücke-Syntax beschrieben:

Syntax	Beschreibung
*	0 oder mehr Wiederholungen
+	1 oder mehr Wiederholungen
?	0 oder 1 Wiederholungen
{n}	n Wiederholungen (n>0)
{n,}	n oder mehr Wiederholungen (n>=0)
{n,m}	zwischen n und m Weiderholungen (0<=n<=m, 0>m)
{,m}	zwischen 0 und m Wiederholungen (m>0)
|	„ODER“-Verknüpfung
^	Anfang der Partition
$	Ende der Partition
?	reluctant, Gegensatz vom greedy (Default)
{- … -}	Ausschluss von der Ausgabe
()	Gruppierung

PERMUTE

ORDIX AG

- Permutation
- Syntax:

```
row_pattern_permute ::=
  PERMUTE (row_pattern [, row_pattern] ...)
```

- Beispiel:

```
PERMUTE (A, B, C)
-- ist gleich bedeutend mit
PATTERN (A B C | A C B | B A C | B C A | C A B | C B A)
```

Reluctant und Greedy Quantifier

ORDIX AG

- Greedy (gierig):
  - Muster (Quantifier) versucht, möglichst viele Treffer zu landen, wenn die Datensätze nicht eindeutig nur einem Muster zugewiesen werden können.
- Reluctant (zögernd):
  - Muster versucht dem darauf folgenden Muster Vorrang zu geben.
  - Wirkt sich bei Datensätzen aus, die nicht eindeutig nur einem Muster zugewiesen werden können.
  - Syntax: das Fragezeichen ?
- Beispiel:

```
pattern (strt ab+? auf+)
define
  ab  as (preis <= prev(preis)),
  auf as (preis >= prev(preis))
```

ROW_PATTERN_SUBSET_CLAUSE

ORDIX AG

- Ein Subset beinhaltet Mustervariablen Vereinigung
- Kommaseparierte Liste von Mustervariablen
- Ein Subset darf nicht auf einen anderen Subset referenzieren
- Optional
- Verwendbar in MEASURES- und DEFINE-Klausel
- Syntax:

```
PATTERN (ROW_PATTERN) [ ROW_PATTERN_SUBSET_CLAUSE ]
```

- Beispiel:

```
pattern ( x+ y+ z+ )
  subset xy = (x, y),
         xz = (x, z)
```

DEFINE

ORDIX AG

- Definition von Mustervariablen
- Unterstützt nur die RUNNING-Semantik (alle Zeilen vom Start der Partition bis zur aktuellen Zeile). Die Schlüsselwörter RUNNING und FINAL-Syntax sind nicht erlaubt.
- Obligatorisch
- Syntax:

```
DEFINE row_pattern_definition_list
row_pattern_definition_list ::= row_pattern_definition[,
   row_pattern_definition]...
row_pattern_definition ::= variable_name AS condition
```

- Beispiel:

```
DEFINE a as flag='a',
       b as (aktion='soll' and sum(betrag) < 4000)
```

MATCH_NUMBER-Funktion

ORDIX AG

- Liefert die Mustertreffernummer
- Alle Zeilen eines Mustertreffers innerhalb einer Partition haben dieselbe Nummer.
- Der erste Mustertreffer innerhalb einer Partition bekommt die Zahl 1.
- Erlaubt in MEASURES- und DEFINE-Klausel
- Beispiel:

```
MEASURES
  match_number()  as match_number
```

CLASSIFIER-Funktion

ORDIX AG

- Liefert die Mustervariable vom dazugehörigen Datensatz
- Erlaubt in MEASURES- und DEFINE-Klausel
- In der DEFINE-Klausel wird der Name der Mustervariable des aktuellen Datensatzes zurückgeliefert.
- In der MEASURES-Klausel wird
  - bei "ONE ROW PER MATCH" der Name der Mustervariable vom letzten Datensatz zurückgeliefert
  - bei "ALL ROWS PER MATCH" der Name der Mustervariable des aktuellen Datensatzes zurückgeliefert
- Bei einem leeren Treffer wird der Wert NULL zurückgeliefert
- Beispiel:

```
MEASURES
  classifier() as classifier
```

Gruppenfunktionen

ORDIX AG

- Gruppenfunktionen sind in MEASURES- und DEFINE-Klausel erlaubt:
  - COUNT, SUM, AVG, MAX und MIN
- DISTINCT-Schlüsselwort in Gruppenfunktionen ist nicht erlaubt:
  - ORA-62512: Dieses Aggregat wird in der MATCH_RECOGNIZE-Klausel noch nicht unterstützt.
- In der DEFINE-Klausel wird nur die RUNNING-Semantik unterstützt. Die Schlüsselwörter RUNNING und FINAL-Syntax sind nicht erlaubt.
- Syntax:

```
row_pattern_aggregate_function ::=
   [RUNNING | FINAL] aggregate_function
```

Navigation

ORDIX AG

- Navigation ist innerhalb von MEASURES- und DEFINE-Klausel möglich mit:
  - PREV, NEXT, FIRST und LAST
- In der DEFINE-Klausel wird nur die RUNNING-Semantik unterstützt. Die Schlüsselwörter RUNNING und FINAL-Syntax sind nicht erlaubt.
- Syntax:

```
row_pattern_navigation_logical ::=
   [RUNNING|FINAL] {FIRST|LAST} (expression[,offset])

row_pattern_navigation_physical ::=
   {PREV|NEXT}(expression[, offset])

row_pattern_navigation_compound ::=
   {PREV | NEXT} ( [RUNNING| FINAL] {FIRST|LAST}
  (expression[, offset]) [,offset])
```

Beispiel Navigation

ORDIX AG

Syntax	Beschreibung
`PREV(a.betrag)`	Betrag des vorhergehenden Datensatzes
`PREV(a.betrag, 2)`	Betrag des zwei Datensätze zurückliegenden Datensatzes (Offset von 2)
`NEXT(a.betrag)`	Betrag des nächsten Datensatzes
`NEXT(a.betrag, 2)`	Betrag des übernächsten Datensatzes (Offset von 2)
`FIRST(a.betrag)`	Betrag vom ersten Datensatz des Musters
`FIRST(a.betrag, 1)`	Betrag des zweiten Datensatzes des Musters (Offset von 1)
`LAST(a.betrag)`	Betrag vom letzten Datensatz des Musters
`LAST(a.betrag, 1)`	Betrag vom vorletzten Datensatz des Musters (Offset von 1)

```
final first(flag,1) => next(final first(flag))
final last (flag,1) => prev(final last(flag))
```

Beispiel 1: Suche nach Mustern in Ampelfarbendaten

ORDIX AG

1. Ermitteln Sie aus der Tabelle TB_MESSWERTE(NR, FARBE) mit Ampelfarbendaten (siehe unten) die folgenden Muster:

   a) gelb, gelb, rot

   b) 5 mal gelb

```
  NR FARBE
---- ------
   1 gelb
   2 rot
   3 gelb
   4 gelb
   5 rot
   6 gelb
   7 rot
   8 rot
   ...
```

Beispiel 1: Lösung für Unterpunkt A

ORDIX AG

```
select nr, farbe, match_number, classifier
from tb_messwerte
match_recognize (
  order by nr
  measures
    match_number() match_number,
    classifier()   classifier
  all rows per match with unmatched rows
  pattern (g g r)
  define g as g.farbe = 'gelb', r as r.farbe = 'rot'
) mr order by nr;
```

Beispiel 1: Lösung für Unterpunkt B

ORDIX AG

```
select nr, farbe, match_number, classifier
from tb_messwerte
match_recognize (
  order by nr
  measures
    match_number() match_number,
    classifier()   classifier
  all rows per match with unmatched rows
  pattern ( g{5} )
  define g as (g.farbe = 'gelb')
) mr order by nr;
```

Beispiel 2: Vergleich der Gehälter

ORDIX AG

2. Erstellen Sie einen Vergleich der Gehälter der Mitarbeiter aus der Mitarbeitertabelle TB_MA(MANR, MANAME, ABTNR, GEHALT). Dabei soll sich der Vergleich eines Mitarbeiters immer auf den vorangegangenen Mitarbeiter beziehen. Folgende Vergleiche sollen durchgeführt werden:
   a) erster, besser, gleich und schlechter.
   b) Suche nach Muster mit mind. 2 mal besser und mind. 2 mal schlechter

Beispiel 2: Lösung für Unterpunkt A

ORDIX AG

```
select manr, maname, gehalt, match_number, classifier
from tb_ma
match_recognize (
  order by manr
  measures
    match_number() match_number,
    classifier()   classifier
  all rows per match with unmatched rows
  pattern (erster besser* schlechter* gleich*)
  define
    erster     as rownum = 1,
    besser     as besser.gehalt > prev (gehalt),
    gleich     as gleich.gehalt = prev (gehalt),
    schlechter as schlechter.gehalt < prev (gehalt)
) mr order by manr;
```

Beispiel 2: Lösung für Unterpunkt B

ORDIX AG

```
select manr, maname, gehalt, match_number, classifier
from tb_ma
match_recognize (
  order by manr
  measures
    match_number() match_number,
    classifier()   classifier
  all rows per match with unmatched rows
  pattern ( besser{2,} schlechter{2,} )
  define
    besser     as besser.gehalt > prev (gehalt),
    schlechter as schlechter.gehalt < prev (gehalt)
) mr order by manr;
```

Beispiel 3: Muster in den Überweisungsdaten

ORDIX AG

3. Ermitteln Sie aus der Tabelle TB_KONTO(LFDNR, BENUTZER_ID, KONTONR, ZEIT, AKTION, BETRAG) mit Überweisungsdaten das folgende Muster:
   a) 1 Überweisung <4000

   b) darauf folgende 2 oder mehr Überweisungen <4000 zu unterschiedlichen Konten (bezogen auf die Vorgängerüberweisung) innerhalb von 30 Tagen. Die Summe aller Überweisungen <4000 muss <18000 sein.

   c) darauf folgende Überweisung >500000 innerhalb von 10 Tagen seit der letzten Überweisung <4000

Beispiel 3: Lösung

ORDIX AG

```
select ...
from tb_konto
match_recognize (
  partition by benutzer_id
  order by lfdnr
  measures ...
  all rows per match with unmatched rows
  pattern ( a b{2,} c )
  define
    a as (aktion='soll' and betrag < 4000),
    b as (
      aktion='soll' and betrag < 4000   and
      prev(b.kontonr) <> b.kontonr      and
      last(b.zeit) - first(b.zeit) < 30 and
      sum(b.betrag) + a.betrag < 18000
    ),
    c as (
      aktion='soll' and betrag > 500000 and
     (c.zeit - last(b.zeit) < 10)
    )
) m order by 1,2,3;
```

Performancevergleich

ORDIX AG

- Zugriff auf Beispieltabelle mit 5 Mill. Datensätzen (624 MB)

Nr	Beschreibung	Analytische Funktionen (*)	MATCH_RECOGNIZE (*)
1	Zugriff auf Vorgängerwert	**8,35**	16,27
2	Summierung auf einer Partition	**28,46**	45,35
3	Kumulierte Summierung auf einer Partition	**26,14**	47,14
4	Suche nach 3 aufeinanderfolgenden gleichen Werten	**20,50**	22,65
5	Suche nach 80 aufeinanderfolgenden gleichen Werten	**22,04**	23,14
6	Suche nach folgendem Muster: (A{,9} | B{2,}) | C{6,}	😕	**24,36**

(*) Angaben in Sekunden

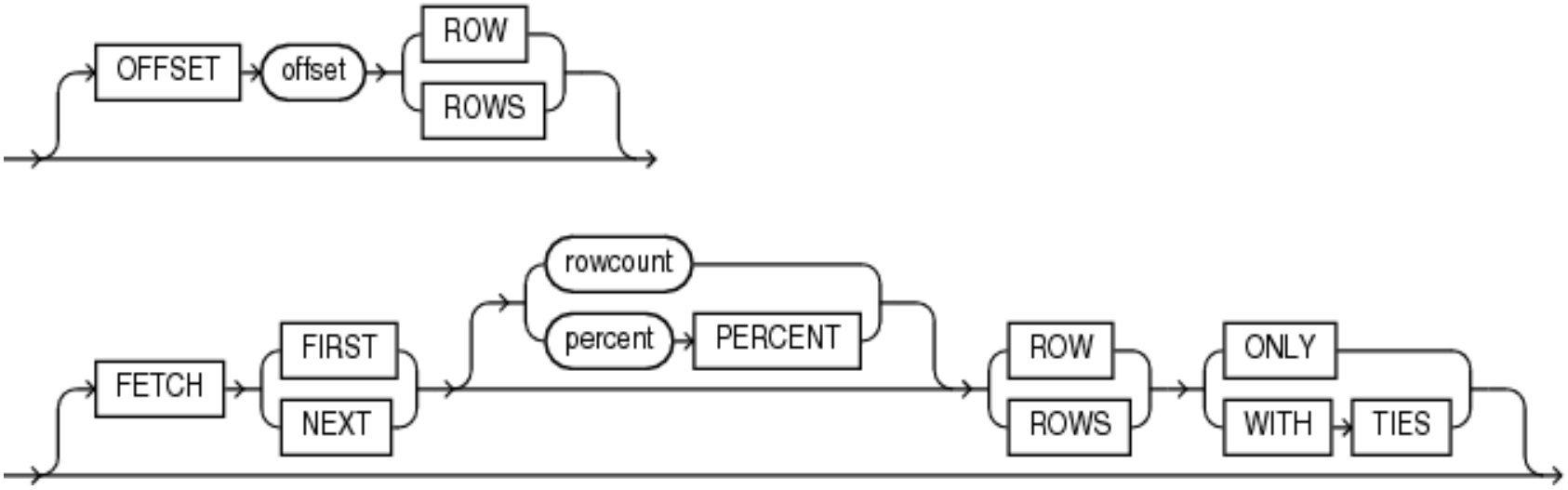
Top N-Klausel

ORDIX AG

- Limitierung von zurückgegebenen Zeilen einer SELECT-Abfrage.
- Abrufen einer bestimmten Teilmenge von Daten.
- Wird nach der ORDER BY-Klausel angegeben.
- Besser lesbar als die Alternative mit Inline-Views und mit der ROWNUM-Funktion.
- Oracle wandelt die Top-N-Klausel-Syntax intern in die Analytischen Funktionen Syntax um.
- Mit OFFSET können Zeilen zu Beginn der Rückgabe übersprungen werden.
- Mit FETCH kann die Anzahl oder der Prozentsatz der Zeilen angegeben werden, die ausgegeben werden sollen.
- Mit der WITH TIES-Klausel kann bestimmt werden, dass zusätzlich zu den selektierten Datensätzen, auch andere Datensätze ausgegeben werden, die den Werten des Sortierschlüssels entsprechen.

Syntax von Top N-Klausel

ORDIX AG



Quelle: Oracle Dokumentation 12c

Beispiel Top N-Klausel I

ORDIX AG

nr	name
1	ma1
2	ma2
3	ma3
4	ma4
5	ma5

```
SELECT ... ORDER BY ...
FETCH FIRST 2 ROWS ONLY
```

```
SELECT ... ORDER BY ...
OFFSET 2 ROWS FETCH NEXT 2
ROWS ONLY
```

Beispiel Top N-Klausel II

ORDIX AG

- Aufgabenstellung:
  - Ermitteln Sie die zwei bestverdienenden Mitarbeiter.

```
-- Beispiel ohne Top N-Klausel…
select * from (
  select manr, maname, abtnr, gehalt,
    row_number() over (order by gehalt desc) rn
  from tb_ma
) where rn <=2;


-- Beispiel mit Top N-Klausel…
select manr, maname, abtnr, gehalt
from tb_ma
order by gehalt desc fetch first 2 rows only;
```

Einschränkungen von Top N-Klausel

ORDIX AG

- Die Klausel FOR UPDATE kann bei einer SELECT-Abfrage mit Top N-Klausel nicht verwendet werden.
- Die Sequence Pseudospalten CURRVAL und NEXTVAL können bei einer SELECT-Abfrage mit Top N-Klausel nicht verwendet werden.
- Falls eine Materialized View auf einer SELECT-Abfrage mit Top N-Klausel basiert, dann kann diese Materialized View nicht mit einem „incremental refresh“ aktualisiert werden.

Fazit

ORDIX AG

- Analytische Funktionen:
  - Vereinfachen SQL-Code
  - Bessere Wartbarkeit
  - Weniger Fehler
  - Verbessern die Performance
  - Nicht nur für Data Warehouse
- MATCH_RECOGNIZE:
  - Ein mächtiges SQL-Feature für Mustererkennung
  - Nutzung von regulären Ausdrücken
  - Spezielle Syntax
  - Anforderungen lassen sich fast im Klartext ausformulieren
  - DISTINCT-Syntax fehlt
  - Keine Vergleiche auf logischer Wertebene (wie RANGE)

Vielen Dank für Ihre Aufmerksamkeit!

ORDIX AG

Zentrale Paderborn
Karl-Schurz-Str. 19a
33100 Paderborn
Tel.: 05251 1063-0
Fax: 0180 1 67349 0

Seminarzentrum Wiesbaden
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Weitere Geschäftsstellen
in Köln, Münster und
Gersthofen (bei Augsburg)

info@ordix.de
www.ordix.de

Wir informieren Sie stets aktuell in unserem Blog:

https://blog.ordix.de

ORDIX AG

Zentrale Paderborn
Karl-Schurz-Str. 19a
33100 Paderborn
Tel.: 05251 1063-0
Fax: 0180 1 67349 0

Seminarzentrum Wiesbaden
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Weitere Geschäftsstellen
in Köln, Münster und
Gersthofen (bei Augsburg)

info@ordix.de
www.ordix.de