



Lahmlegen der Datenbank für Fortgeschrittene

Gute Performance durch oder trotz Funktionen

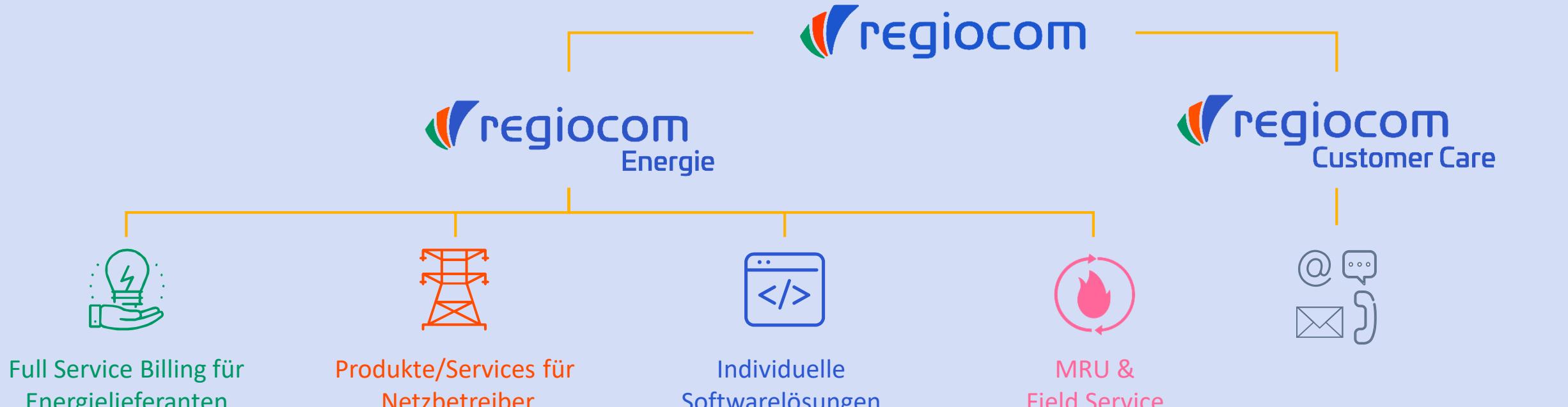
Über mich

Even Langer

- PO und Oracle Certified Professional
- seit 12 Jahren regiocom SE
- Arbeitsschwerpunkte:
 - Performancetuning
 - Verarbeitung von Massendaten
 - Applikationsdesign/Systemarchitektur
 - Moderation TechTalk-Vortragreihe
- Hobbies:
 - People-Fotografie www.evenlanger.de
 - Reisen



Die Geschäftsfelder und Partnerunternehmen



> 6500 Mitarbeiter



Kunden



Agenda

1

Rückblick / Vorbemerkung

5 min

2

Analytische Funktionen

25 min

3

(Pipelined) Table Functions

10 min

4

Performance bei NVL

5 min

5

SQL-Performance mit eigenen Funktionen

10 min

6

Diskussion

Rückblick



Wie lege ich als Entwickler die Datenbank definitiv lahm?

SQL-Performance-Fallstricke



09.09.2024

 regiocom

 regiocom

Rückblick

Wichtigste Optimierungsregeln:

Passendes Datenmodell

Nicht benötigte Abfragen entfernen

Selektivität, Selektivität, Selektivität!

Rückblick

ja

Hohe Selektivität erwartet

nein

Indexe
Statistiken
Constraints
Hints

Komprimierung Tabellen/Indexe
Partitionierung
Alle relevanten Attribute im Index
Aufsplittung des Statements
Bitmap-Indexe (DWH)

Vorbemerkung

Funktionen in der Datenbank können die Performance massiv verbessern,
Funktionen in der Datenbank können die Performance massiv verschlechtern!*

* „it depends“ ;-)

Agenda

1

Rückblick / Vorbemerkung

5 min

2

Analytische Funktionen

25 min

3

(Pipelined) Table Functions

10 min

4

Performance bei NVL

5 min

5

SQL-Performance mit eigenen Funktionen

10 min

6

Diskussion

Demo 1



Analytische Funktionen

Database / Oracle / Oracle Database / Release 19

SQL Language Reference

- ▶ 2 Basic Elements of Oracle SQL
- ▶ 3 Pseudocolumns
- ▶ 4 Operators
- ▶ 5 Expressions
- ▶ 6 Conditions
- ▶ 7 Functions
 - About SQL Functions
 - Single-Row Functions
 - Aggregate Functions
 - Analytic Functions**
 - Object Reference Functions
 - Model Functions
 - OLAP Functions
 - Data Cartridge Functions
 - ABS
 - ACOS

Analytic Functions

Analytic functions compute an **aggregate value based on a group of rows**. They differ from aggregate functions in that they return multiple rows for each group. The group of rows is called a **window** and is defined by the **analytic_clause**. For each row, a **sliding window of rows is defined**. The window determines the range of rows used to perform the calculations for the current row. Window sizes can be based on either a physical number of rows or a logical interval such as time.

Analytic functions are the last set of operations performed in a query except for the final ORDER BY clause. All joins and all WHERE, GROUP BY, and HAVING clauses are completed before the analytic functions are processed. Therefore, analytic functions can appear only in the select list or ORDER BY clause.

Analytic functions are commonly used to compute cumulative, moving, centered, and reporting aggregates.

analytic_function ::=

Description of the illustration *analytic_function.eps*

analytic_clause ::=

Description of the illustration *analytic_clause.eps*

Analytische Funktionen

- Vorgehensweise
 1. Gewünschte Funktion auswählen
 2. Ein oder mehrere Partitionierungsattribute auswählen
 3. Ein oder mehrere Attribute zur Sortierung innerhalb der Partition festlegen
 4. Ggf. noch ein Window definieren (bspw. nur die 3 Zeilen vor und hinter der aktuellen Zeile)
- Vorteile:
 - Deutlich bessere Performance, da Tabelle(nausschnitte) nur einmal gelesen werden
 - Meist übersichtlichere Statements ohne Unterabfragen
- Erweiterung: MATCH_RECOGNIZE-Klausel

Demo 2



Analytische Funktionen

AVG	LISTAGG	RANK
CORR	MAX	RATIO_TO_REPORT
COUNT	MIN	REGR_
COVAR_POP	NTH_VALUE	ROW_NUMBER
COVAR_SAMP	NTILE	STDDEV
CUME_DIST	PERCENT_RANK	STDDEV_POP
DENSE_RANK	PERCENTILE_CONT	STDDEV_SAMP
FIRST	PERCENTILE_DISC	SUM
FIRST_VALUE	PREDICTION	VAR_POP
LAG	PREDICTION_COST	VAR_SAMP
LAST	PREDICTION_DETAILS	VARIANCE
LAST_VALUE	PREDICTION_PROBABILITY	
LEAD	PREDICTION_SET	

Agenda

1

Rückblick / Vorbemerkung

5 min

2

Analytische Funktionen

25 min

3

(Pipelined) Table Functions

10 min

4

Performance bei NVL

5 min

5

SQL-Performance mit eigenen Funktionen

10 min

6

Diskussion

(Pipelined) Table Functions

- Liefern Tabellenstruktur als Ergebnis zurück
- Können in FROM-Klauseln verwendet werden
- Joins mit „normalen“ Tabellen möglich
- mengenbasiertes „Weiterarbeiten“
- Vereinfacht Separierung von Verarbeitung und Persistierung -> einfachere UNIT-Tests
- Vorteil Pipelined-Functions:
 - Einzelne Datensätze werden sofort zurückgegeben
 - Geringerer Speicherverbrauch
 - Bessere Performance insbesondere bei Verwendung von Parallelisierung

(Pipelined) Table Functions

```
create or replace type exchange_rate as object (currency varchar2(3), rate number);

create or replace type exchange_rate_tab as table of exchange_rate;

create or replace function get_exchange_rates (p_date in date)
return exchange_rate_tab pipelined
as
begin
  for x_rates in (<Umrechnungsfaktoren aus historisierter Tabelle oder Webservice ermitteln>)
  loop
    pipe row(exchange_rate(x_rates.currency, x_rates.rate));
  end loop;
  return;
end;
/

Select b.price as price_in_eur, b.price * r.rate prince_in_currency, r.currency
from bestellung b
cross join get_exchange_rates(bestelldatum) r
where bestellnummer = 4711;
```

Demo 3



Agenda

1

Rückblick / Vorbemerkung

5 min

2

Analytische Funktionen

25 min

3

(Pipelined) Table Functions

10 min

4

Performance bei NVL

5 min

5

SQL-Performance mit eigenen Funktionen

10 min

6

Diskussion

Performance bei NVL

Ergebnis der Oracle-eigenen Funktion:

Gibt ersten Parameter zurück, wenn der nicht null ist, ansonsten wird den zweiten Parameter ausgegeben

Beispiel: nvl(l_startdatum, sysdate)

Problem: Die Funktion **NVL** wertet immer beide Parameter aus!

Demo 4



Agenda

1

Rückblick / Vorbemerkung

5 min

2

Analytische Funktionen

25 min

3

(Pipelined) Table Functions

10 min

4

Performance bei NVL

5 min

5

SQL-Performance mit eigenen Funktionen

10 min

6

Diskussion

SQL-Performance mit eigenen Funktionen

- Konkretes Beispiel bei der Kommunikation im Energiemarkt
- Gemeldetes Problem:
 - Job-Laufzeit wesentlich länger als sonst
 - Die Verarbeitung einer einzigen Nachricht dauert ca. 20-30 Minuten
- Active Session History/AWR-Report liefert scheinbares Problemstatement

SQL-ID: ax8mzv9rgk90z

```
select max(eme.prozessdatum), max(eme.prozessdatum_dat),
       max(eme.prozessdatum_anfang), max(eme.prozessdatum_anfang_dat),
       max(emm.prozessdatum), max(emm.prozessdatum_dat),
       max(emm.prozessdatum_anfang), max(emm.prozessdatum_anfang_dat)
  into v_datumsfelder
 from sst_exp_msc_erfassung eme join sst_exp_msc_zaehlwerk on emz_eme_id = eme_id
                                  join sst_exp_msc_messwerte emm on emm_emz_id = emz_id
 where eme_emsid = p_ems_id;
```

- Statement lokalisiert im Package workflow_mscons.f_get_ablesedatum

SQL-Performance mit eigenen Funktionen

- Aber: Durchschnittliche Ausführungszeit im unteren Millisekundenbereich
- Dafür: Extrem viele Ausführungen

BEGIN_INTERVAL_TIME	PARSING_SCHEMA_NAME	PLAN_HASH_VALUE	EXECUTIONS_DELTA	ELAPSED_TIME	ELAPSED_PER_EXEC
06.05.2024 12:00:46,363	RCUTILDE	2749206032	3782589	6585,2378	0,0017
06.05.2024 11:00:30,814	RCUTILDE	2749206032	3365692	6610,1642	0,002
06.05.2024 10:00:15,071	RCUTILDE	2749206032	3891997	6731,6522	0,0017
06.05.2024 09:00:59,868	RCUTILDE	2749206032	4573051	7637,572	0,0017
06.05.2024 08:00:44,445	RCUTILDE	2749206032	3667140	6772,4222	0,0018
06.05.2024 07:00:28,833	RCUTILDE	2749206032	3933268	6459,1094	0,0016
06.05.2024 06:00:13,793	RCUTILDE	2749206032	4077425	6625,6687	0,0016
06.05.2024 05:00:58,782	RCUTILDE	2749206032	4098509	6558,9041	0,0016
06.05.2024 04:00:43,501	RCUTILDE	2749206032	4070874	6697,0625	0,0016
06.05.2024 03:00:28,128	RCUTILDE	2749206032	4062837	6670,4867	0,0016
06.05.2024 02:00:13,035	RCUTILDE	2749206032	4227428	6611,5425	0,0016
06.05.2024 01:00:57,567	RCUTILDE	2749206032	3584492	6145,0876	0,0017
06.05.2024 00:00:42,079	RCUTILDE	2749206032	3436397	6117,547	0,002
05.05.2024 23:00:26,659	RCUTILDE	2749206032	3669198	6740,7972	0,0018
05.05.2024 22:00:10,722	RCUTILDE	2749206032	3913259	6776,2316	0,0017
05.05.2024 21:00:55,516	RCUTILDE	2749206032	3491357	6696,854	0,0019
05.05.2024 20:00:40,032	RCUTILDE	2749206032	4048047	6746,8941	0,0017
05.05.2024 19:00:24,670	RCUTILDE	2749206032	4041991	6745,1218	0,0017
05.05.2024 18:00:08,798	RCUTILDE	2749206032	4034865	6771,2971	0,0017
05.05.2024 17:00:53,722	RCUTILDE	2749206032	4064821	6660,0339	0,0016
05.05.2024 16:00:38,493	RCUTILDE	2749206032	4081561	6782,6673	0,0017
05.05.2024 15:00:23,451	RCUTILDE	2749206032	4063375	6758,0474	0,0017
05.05.2024 14:00:07,612	RCUTILDE	2749206032	4207959	6708,1313	0,0016
05.05.2024 13:00:52,704	RCUTILDE	2749206032	4033700	6619,4633	0,0016
05.05.2024 12:00:37,494	RCUTILDE	2749206032	4307813	6716,0917	0,0016
05.05.2024 11:00:22,175	RCUTILDE	2749206032	4185276	6718,413	0,0016

SQL-Performance mit eigenen Funktionen

- Wo wir die Funktion ausgeführt?

```
select elapsed_time, sql_id, sql_fulltext
  from v$sql
 where lower(sql_fulltext) like '%f_get_ablesedatum%'
 order by elapsed_time desc;
```

row#	ELAPSED_TIME	SQL_ID	SQL_FULLTEXT
1	164152345503	dzqkmn9sq8dc9	SELECT NVL(MAX(EMS_ID),0) FROM SST_EXP_MSCONS JOIN SST_EXP_MSC_ERFASSUNG ON EME_EMSID = EMS_ID JOIN SST_EXP_MSC_ZAHLWERK ON EMZ_EME_ID = EME_ID WH
2	9412818	4s3d5a4n9nc4s	select 1 from wkf_vorgang join wkf_vorgang_mscons on vorm_vorid = vor_id...
3	8789194	aqvzn29rchzwz	SELECT VORM_VORID FROM WKF_VORGANG JOIN WKF_VORGANG_MSCONS ON VORM_VORID = VOR_ID JOIN SST_EXP_MSCONS ON EMS_ID = VORM_MSCONS_EXPID JOIN SST
4	4230046	c8m5221prq078	select 1...
5	2852904	dzqkmn9sq8dc9	SELECT NVL(MAX(EMS_ID),0) FROM SST_EXP_MSCONS JOIN SST_EXP_MSC_ERFASSUNG ON EME_EMSID = EMS_ID JOIN SST_EXP_MSC_ZAHLWERK ON EMZ_EME_ID = EME_ID WH
6	2401513	0kj0jjj21pg1j	--Kundenablesungen sollen nicht in den PK verschoben werden...
7	936808	8ncpzng4c0cq5	-- wenn Ausbau + es existiert ein Einbau/Neuanlage mit ident. Ablesedatum...
8	140107	7b7600u7bu0aa	wenn Ausbau + es existiert kein Einbau/Neuanlage mit Ablesedatum am gleichen oder folgenden Tag

SQL-Performance mit eigenen Funktionen

- Problem gefunden: kippender Ausführungsplan!

```
procedure p_set_isu_daten(p_ems_id sst_exp_mscons.ems_id%type, p_meloid lok_messlokation.melo_id%type, p_vet_ende date, p_msc_absender sst_exp_mscons.ems_absender_ewsid%type, p_msc_Ablesegrund in out varchar2, p_msc_Zaehlerstandart in out varchar2, p_ms_isu_ablesegrund in out varchar2, p_postkorb in out number, p_zst number defa
)
is
    v_aehnliche_Zaehlernummern boolean := true;
    v_Einzugszaehlerstand sst_exp_mscons.ems_id%type := 0;
    v_Auszugszaehlerstand sst_exp_mscons.ems_id%type := 0;
    v_ZPT_Zaehlernummer net_geraet.ngr_geraetenummer%type;
    v_isu_zahlernummer net_geraet.ngr_geraetenummer%type;
    v_IstMarktpartnerExtern boolean;
    v_anz number;
    v_isu_ablesegrund sst_exp_mscons.ems_isu_ablesegrund%type;
    v_pi sst_exp_mscons.pruefidentifikator%type;
    v_sonderfall_mu_mrg number := 0;
begin
    v_isu_ablesegrund := p_isu_ablesegrund;
    -- Pruefung nur fuer Zaehlerstaende / BWZZ notwendig
    if p_zst is not null then
        begin
            select nvl(max(emis_id),0) into v_Einzugszaehlerstand
            from sst_exp_mscons
                join sst_exp_msc_erfassung on eme_emsid = ems_id
                join sst_exp_msc_zahlwerk on emz_eme_id = eme_id
            where ems_valid = p_vetid_1f
            and ems_isu_kopie = 1
            and ems_original_emsid <> p_ems_id
            and edis = sst_common.f_get_ISU_obis(edis, p_meloid, workflow_mscons.f_get_ablesedatum(emis_id))
            and obis.f_check_obis_brennwert(edis) = 0
            and obis.f_check_obis_zzahl(edis) = 0
            and ems_isu_ablesegrund = c_ISU_Ablesegrund_Einzug;

            select nvl(max(emis_id),0) into v_Auszugszaehlerstand
            from sst_exp_mscons
                join sst_exp_msc_erfassung on eme_emsid = ems_id
                join sst_exp_msc_zahlwerk on emz_eme_id = eme_id
            where ems_valid = p_vetid_1f
            and ems_isu_kopie = 1
            and ems_original_emsid <> p_ems_id
            and edis = sst_common.f_get_ISU_obis(edis, p_meloid, workflow_mscons.f_get_ablesedatum(emis_id))
            and obis.f_check_obis_brennwert(edis) = 0
            and obis.f_check_obis_zzahl(edis) = 0
            and ems_isu_ablesegrund = c_ISU_Ablesegrund_Auszug;
        end;
    end if;
end;
```

SQL-Performance mit eigenen Funktionen

- Mögliche Probleme:
 - Kontextswitch SQL – PLSQL
 - Einzelarbeitung statt mengenbasiertes Vorgehen
 - Extreme häufige Funktionsaufrufe durch kippende Ausführungspläne
- Lösungsmöglichkeiten:
 - Oracle 21c: skalare SQL Macros
 - Oracle 23ai: SQL Transpiler
 - Pipelined Functions
 - Bei Funktionen in der WHERE-Bedingung:
 - prüfen, ob andere, stärker einschränkende Bedingungen vorhanden sind
 - Im Problemfall WITH-Klausel mit MATERIALZE-Hint verwenden oder Statement splitten

Fazit

Nutzt Funktionen an der richtigen Stelle ;-)
Testen!

Reminder aus Vortrag vom letzten Jahr:
Denkt an die function-based Indexe!



Ist diese Vase
wasserdicht?

Die Auflösung erfährst du
heute Abend im Vortrag

**»3D-Druck
zum Anfassen«**



SCAN ME





**Vielen Dank für eure
Aufmerksamkeit**

www.regiocom.com

Even Langer

Product Owner & Oracle Certified Professional
even.langer@regiocom.com

 **regiocom**

The Regiocom logo consists of a stylized 'r' icon made of three curved lines in red, green, and blue, followed by the word 'regiocom' in a bold, lowercase, sans-serif font.

Diskussion