

ORACLE®

# Oracle TEXT

## Internals, Tipps & Tricks

**Ulrike Schwinn**  
**Carsten Czarski**  
**Business Unit Database**

ORACLE Deutschland GmbH

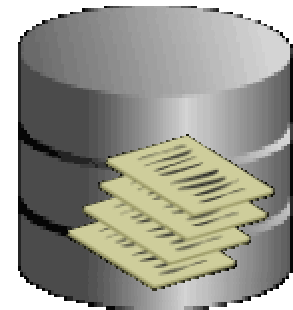


ORACLE®

# Text-Dokumente in der Datenbank

## Oracle TEXT

- Volltextrecherche "State-of-the-Art"
  - Linguistische Suchfunktionalität  
*Stemming, Base-Letter, Thesaurus-Suche, Phrasensuche, Stopwort-Listen, ...*
  - Dokumente innerhalb und außerhalb der Datenbank  
*FILE\_DATASTORE, URL\_DATASTORE*
  - Erweiterungen für Text Mining  
*Klassifizierung, Clustering*
- Ansprache mit SQL
  - CREATE INDEX
  - SELECT ... WHERE CONTAINS ...



# Oracle TEXT

## Indizierung ...

- Unterstützung aller gängigen Datentypen
  - XMLTYPE, VARCHAR2, CLOB, BLOB, ...
  - Filter bei Binärdaten möglich
- Auch mehrere Indizes pro Tabelle

```
SQL> create index idx_textindex  
2 on dokument_tab (dokument)  
3 indextype is CTXSYS.CONTEXT  
4 /
```

Index created.

### HINTERGRUND:

- Extensible Indexing Framework
- Spatial, TEXT, Expression Filter
- Eigene Indizes möglich

# Oracle TEXT

## Volltextrecherche

- Abfrage mit SQL
  - CONTAINS-Funktion
  - Kombinierbar mit relationalen Abfragen
- Relevanz-Ranking anhand Wort-Häufigkeiten
  - SCORE()-Funktion
- Ergebnis-Aufbereitung
  - Highlighting
  - "Keyword-in-Context"

```
select score(1), dokument
from dokument_tab
where CONTAINS(dokument, 'Software AND Oracle')>0
/
```

# Häufige Fragen ...

- Was passiert eigentlich „im Index“ ...?
- Wie kann ich den Index überwachen ...?
- Indexfragmentierung: Was tun ...?
- Strategien zur Index-Synchronisierung ...?
- Mixed Queries: Was tun ...?
- Sortierungen: Was tun ...?

# Grundlagen

## Aufbau eines TEXT Index

- DR\$[Indexname]\$I
  - *Token-Tabelle*: Enthält alle Tokens mit (binären) Informationen über die Dokumente, in denen sie vorkommen.
- DR\$[Indexname]\$R
  - Mapping-Tabelle DOCID → ROWID
- DR\$[Indexname]\$K
  - Mapping-Tabelle ROWID → DOCID
- DR\$[Indexname]\$N
  - *Negativliste*: Enthält alle gelöschten DOCID's
- DR\$[Indexname]\$P
  - *Substring-Index (wenn aktiviert)*

# Aufbau eines Oracle TEXT-Index Token-Tabelle (\$I)

- Tabellenaufbau

Name	Null?	Type
TOKEN_TEXT	NOT NULL	VARCHAR2 (64)
TOKEN_TYPE	NOT NULL	NUMBER (3)
TOKEN_FIRST	NOT NULL	NUMBER (10)
TOKEN_LAST	NOT NULL	NUMBER (10)
TOKEN_COUNT	NOT NULL	NUMBER (10)
<b>TOKEN_INFO</b>		<b>BLOB</b>

# Aufbau eines Oracle TEXT-Index

## Token-Tabelle (\$I)

- Eine Ausgangssituation

ID	TEXT
1001	Nacht und Tag, Tag und Na
1002	Es war eine stürmische Na...

ROWID:

ROWID:

AAASWTAAFAAACGAAC4

DOCID:

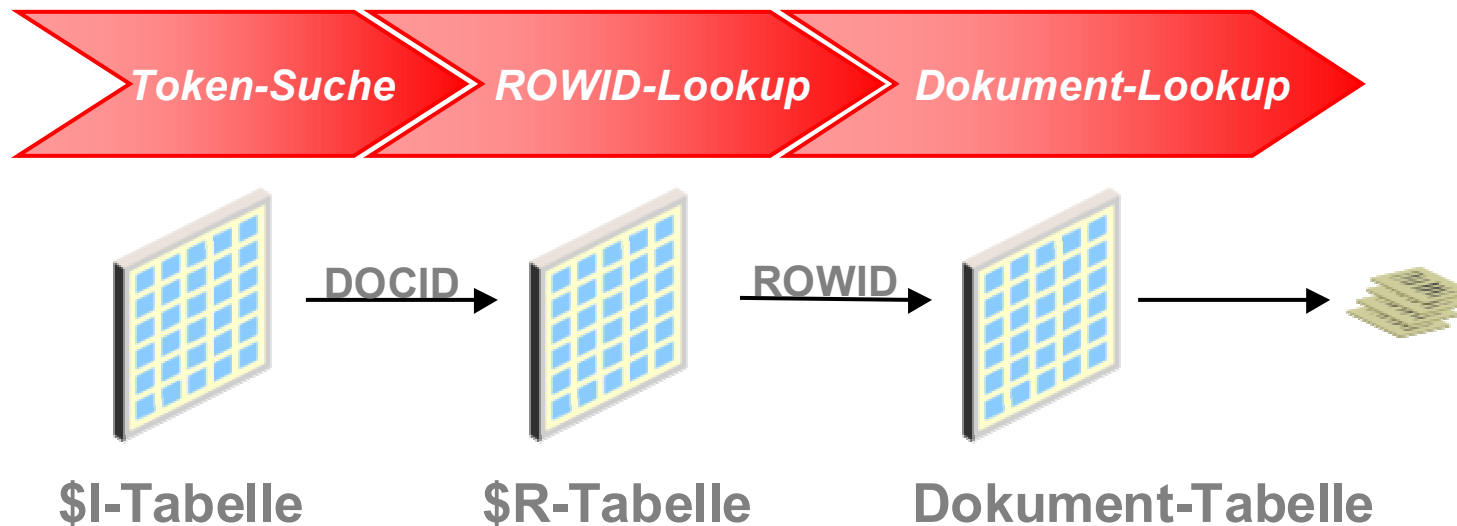
2

TOKEN_TEXT	TOKEN_INFO (BLOB)
NACHT	(DOCID 1, INFO 1,2) (DOCID 2, INFO 3)
TAG	(DOCID 1, INFO 4,5)
STÜRMISCHE	(DOCID 2, INFO 6)

# Grundlagen

## Verwendung von DOCID's

- Oracle TEXT verwendet intern DOCID's
  - DOCID ist kompakter als ROWID
  - Faktor ~10:1
- Prozeß bei einer Query:



# Verwendung von DOCID's

## Verhalten bei DML

- INSERT
  - Erzeugung einer neuen DOCID
  - Neue Zeile(n) in Token-Tabelle (\$I) für neue Tokens
  - Neue Zeile in Mapping-Tabellen (\$N,\$R)
- Vorgehensweise Token-Tabelle ...
  - Idealfall wäre:  
Einpfelegen in bestehende Zeilen, wenn Token bereits vorhanden sind
  - Problem dabei:  
Einpfelegevorgang (SQL UPDATE auf BLOB) zu teuer, daher neue Zeilen (SQL INSERT)

# Verwendung von DOCID's

## Verhalten bei DML

- DELETE
  - Löschen der DOCID aus den Mapping-Tabellen (\$K, \$R)
  - Eintrag in Negativlisten-Tabelle (\$N)
  - Token-Tabelle (\$I) bleibt unverändert
- Vorgehensweise ...
  - Idealfall wäre:  
Löschen aller Einträge direkt aus der Token-Tabelle
  - Problem dabei:  
Löschvorgang (SQL UPDATE auf BLOB) zu teuer, daher keine Aktion

# Oracle TEXT und SQL UPDATE

- Default-Verhalten
  - Behandlung wie DELETE und INSERT
  - Grund: In-Place-Pflege der Token-Tabelle zu teuer
- Problem:
  - UPDATES führen zur Index-Fragmentierung
  - Bedarf nach OPTIMIZE bzw. REBUILD
- Lösungsansatz:
  - MDATA-Sections (Oracle10g)

# MDATA-Sections

... oder Metadaten-Bereiche

- Anwendung für "getaggte" Dokumente
  - XML, HTML, ...
- Keine Zerlegung in *Tokens*
- Eigenständiges UPDATE möglich

```
begin
  ctx_ddl.add_mdata_section(
    group_name => 'mysg',
    section_name => 'branchid',
    tag => 'branch-id');
end;
```

# Metadaten-Bereiche

## Ausgangssituation: ohne MDATA-Sections

### 1a. Ausgangssituation

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>92</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	1 (NORMAL)
92	1 (NORMAL)

# Metadaten-Bereiche

Ausgangssituation: ohne MDATA-Sections

## 1b. Reguläres SQL-UPDATE

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>91</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	1 (NORMAL)
92	1 (NORMAL)

# Metadaten-Bereiche

Ausgangssituation: ohne MDATA-Sections

## 1c. Index Synchronize

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>91</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	1 (NORMAL)
92	1 (NORMAL) (N-Table)
91	1 (NORMAL)

# Metadaten-Bereiche

## Ausgangssituation: ohne MDATA-Sections

### 1d. Index Optimize

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>91</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	1 (NORMAL)
91	1 (NORMAL)

# MDATA Sections

## Effizientes Metadaten-UPDATE

### 2a. Ausgangssituation mit MDATA

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>92</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	400 (MDATA)
92	400 (MDATA)

# MDATA Sections

## Effizientes Metadaten-UPDATE

### 2b. MDATA-Update

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>92</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	400 (MDATA)
92	400 (MDATA)
92	-400 (MDATA)
91	-400 (MDATA)

# MDATA Sections

## Effizientes Metadaten-UPDATE

### 2c. Index Optimize

ID	TEXT
1001	<text>Nacht und Tag, Tag und Nacht</text> <branch-id>90</branch-id>
1002	<text>Es war eine stürmische Nacht</text> <branch-id>92</branch-id>

TOKEN_TEXT	TOKEN_TYPE
NACHT	1 (NORMAL)
TAG	1 (NORMAL)
90	400 (MDATA)
91	400 (MDATA)

# MDATA Sections

## PL/SQL Prozeduren für Update

- MDATA-Section ändern

```
ctx_ddl.remove_mdata(  
  idx_name      => 'myidx',  
  section_name => 'branchid',  
  mdata_value  => '1',  
  mdata_rowid  => v_rowid  
);  
  
ctx_ddl.add_mdata(  
  idx_name      => 'myidx',  
  section_name => 'stocklevel',  
  mdata_value  => '0',  
  mdata_rowid  => v_rowid  
);
```

# MDATA Sections

## Seiteneffekte und Hinweise ...

- MDATA-Update betrifft **nur** den Index
  - Betroffen ist nur die Token-Tabelle
  - Index und Originaldaten sind "*out-of-sync*"
- MDATA-Update fügt **neue** Zeilen hinzu
  - Negative Token-Types (-400) sind Deltas
  - Eliminierung bei Index Optimize
- MDATA-Update ist **immer** *Remove* und *Add*

# MDATA-Sections

## Strategie zur Vermeidung von "*mixed Queries*"

- Mehrere Tabellen bzw. Tabellenspalten
  - Unstrukturierte Daten in CLOB/BLOB/XMLTYPE
  - Strukturierte Metadaten in relationalen Spalten
- Zusammenfassung per *User-Data-Store*
  - Volltextindex erfasst **alle** Daten
  - Metadaten als MDATA-Sections
- Metadaten-Update
  - SQL-UPDATE auf relationale Tabellenspalte
  - MDATA-Update (per Trigger)
- Vorteile:
  - Vermeidung von "*mixed Queries*"
  - Metadaten-Update ohne Index-Sync möglich

# Indexfragmentierung

- DML-Aktivität führt zu fragmentiertem Index
  - TOKEN\_INFO-Felder (BLOB) suboptimal befüllt
  - Gelöschte Dokumente (Garbage) in Token-Tabelle
- Lösungsansätze ...
  - Geeignetes Synchronisierungs-Intervall  
ACHTUNG: Neue Option: ON\_COMMIT in 10g
  - Regelmäßiges OPTIMIZE\_INDEX
  - **Wichtig dabei: Regelmäßiges Monitoring**
- Zusätzlich: Transaktionale Indizes
  - In-Memory-Indizierung der Zeilen in CTX\_PENDING

# Indexfragmentierung

## OPTIMIZE\_INDEX

- CTX\_DDL.OPTIMIZE\_INDEX

- Optimierungsebene

– OPTLEVEL_FAST	• "Verdichtung" der Token-Tabelle
– OPTLEVEL_FULL	• "Verdichtung" der Token-Tabelle Entfernen veralteter Token
– OPTLEVEL_REBUILD	• Index Rebuild • schneller als OPTLEVEL_FULL

- Optimierungstiefe

- OPTLEVEL\_TOKEN
- OPTLEVEL\_TOKEN\_TYPE

# Oracle TEXT Monitoring

## Allgemeines ...

- Optimizer-Statistiken sammeln (empfohlen)
  - DBMS\_STATS.GATHER\_...\_STATS
- PL/SQL Paket CTX\_REPORT
  - DESCRIBE\_INDEX
  - INDEX\_SIZE
  - INDEX\_STATS
- Ausgabeformate
  - Plain TEXT
  - XML

### **ACHTUNG:**

- Vollständiger Index-Scan
- Lange Laufzeit bei großen Indizes



# Monitoring

## Ausgabe von CTX\_REPORT

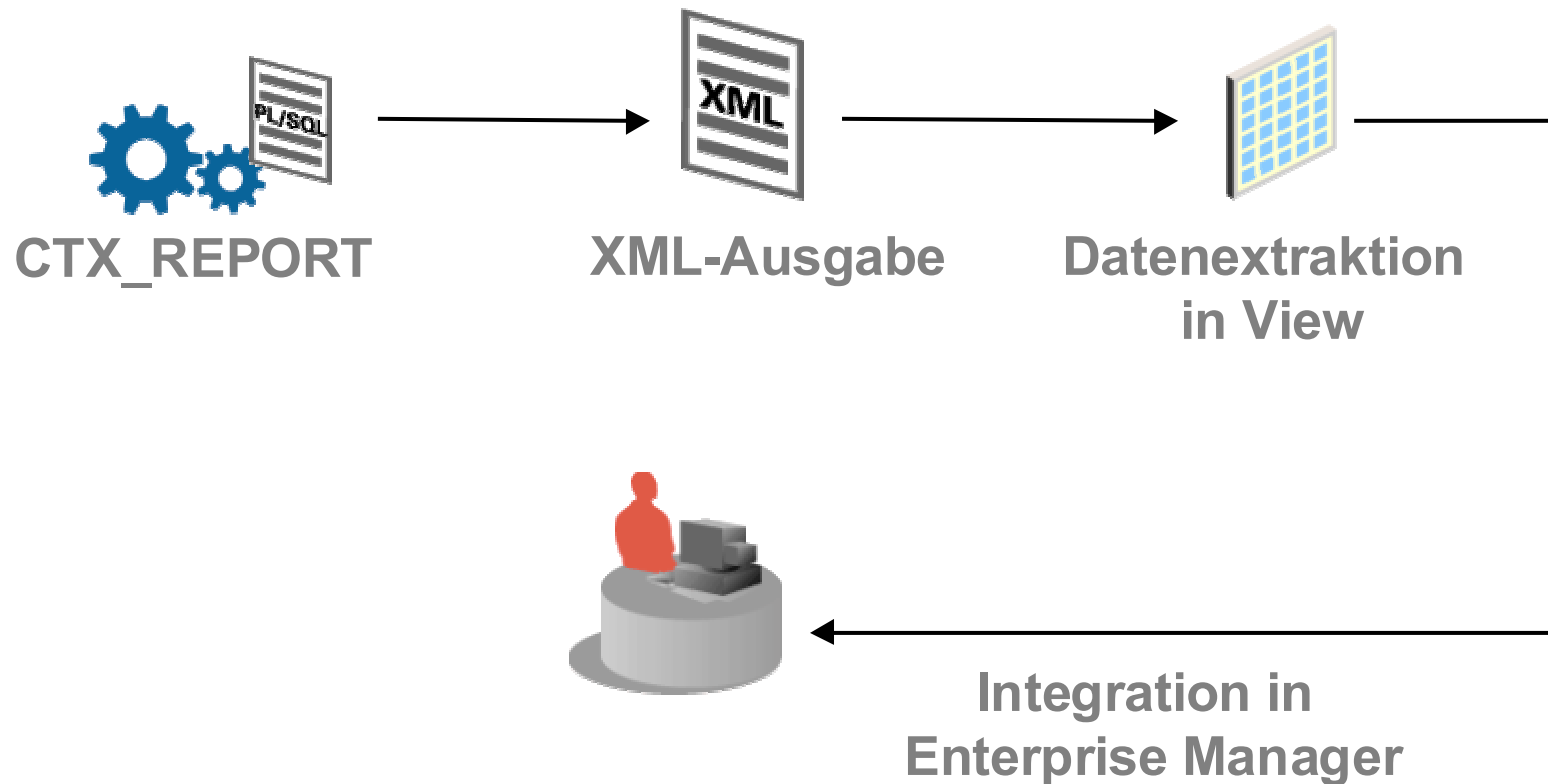
- Plain-Text-Ausgabe

```
-----  
FRAGMENTATION STATISTICS  
-----  
total size of $I data:                278,859 (272.32 KB)  
  
$I rows:                               28,975  
estimated $I rows if optimal:          17,776  
estimated row fragmentation:           39 %  
  
garbage docids:                        0  
estimated garbage size:                 0  
most fragmented tokens:  
  GE (0:TEXT)                           92 %  
  ERMOGLICHT (0:TEXT)                    92 %  
  DURCH (0:TEXT)                         92 %  
  DE (0:TEXT)                            92 %  
  ALLEN (0:TEXT)                         92 %
```

# Oracle TEXT Monitoring

## Beispiel: Enterprise-Manager-Integration

- Vorgehensweise



# Monitoring Enterprise-Manager-Integration




- Benutzerdefinierte Metriken
  - Definition der Schwellenwerte
  - Definition Aktualisierungsintervall

## ▼ Alerts

Kategorie   Kritisch 0

### **ACHTUNG:**

- Laufzeit beachten bei großen Indizes
- Weiträumige Intervalle (Tage / Wochen)

Dringlichkeit ▼	Kategorie	Name	Meldung	ausgelöst
	User Defined Metrics	User Defined Numeric Metric	User Defined Metric Fragmentierung IDX_VOLLTEXT returned a value of 39.	14.01.2006 21:49:10
	Invalid Objects by Schema	Owner's Invalid Object Count	5 Objekte sind im Schema XDB ungültig.	14.01.2006 21:45:48
	Invalid Objects by Schema	Owner's Invalid Object Count	3 Objekte sind im Schema RAPTOR_DEMO ungültig.	14.01.2006 21:45:48

# Indexerstellung

## Weitere Tipps & Tricks

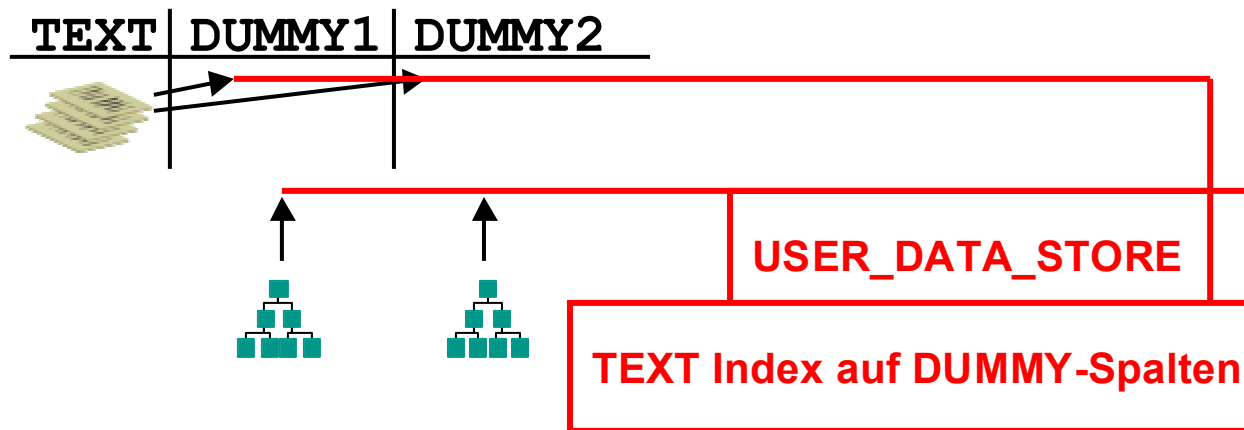
- Präferenzen
  - CTX\_REPORT.CREATE\_INDEX\_SCRIPT
  - Direkt nach Indexerstellung laufen lassen
  - Grund: Änderungen an den Präferenzen
- Stopwörter
  - So viele wie möglich, so wenig wie nötig
  - Indexgröße
- Speicherverbrauch
  - MEMORY-Parameter
  - So viel wie möglich ... aber bitte ohne Paging



# Indexerstellung

## Backup-Index ...

- Problem:
  - Index Rebuild dauert im Verlustfall zu lange
  - Default: **Ein** Index pro Tabellenspalte
- Lösungsansatz: USER\_DATA\_STORE



# Indexerstellung

## Backup-Index mit USER\_DATA\_STORE

- SQL Update auf Dokument-Spalte
  - Per RDBMS-Trigger UPDATE auf DUMMY-Spalten
  - Index Sync
- Index-Auswahl bei Query

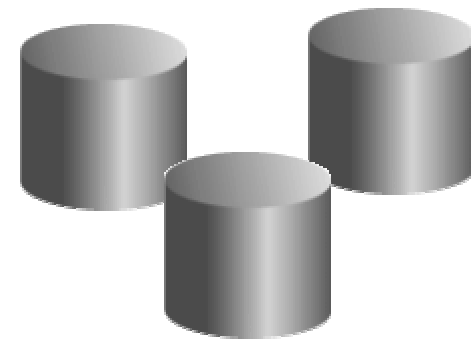
```
select score(1), dokument
from dokument_tab
where CONTAINS(dummy1, 'Software AND Oracle')>0
/
```

```
select score(1), dokument
from dokument_tab
where CONTAINS(dummy2, 'Software AND Oracle')>0
/
```

# Indexerstellung

## Storage-Parameter

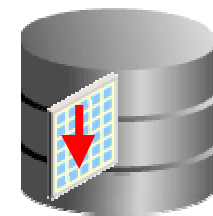
- Storage-Klauseln
  - Tablespaces
  - Extent-Verwaltung
- Achtung:
  - R\_TABLE\_CLAUSE  
**WICHTIG: "LOB (DATA) STORE AS (CACHE)"**
  - I\_INDEX\_CLAUSE  
**WICHTIG: "COMPRESS 2"**



# Oracle TEXT

## Sortieren der Ergebnismenge ... bspw. nach Datum

- Problem:
  - Oracle TEXT liefert Ergebnisse so zurück, wie sie gefunden wurden
  - Sortierung nach Datum muss extra erfolgen
- Lösungsansätze
  - Reverse DOCID searching
  - Partitionierung
  - MDATA-Sections und Progressive Relaxation



# Reverse DOCID Searching

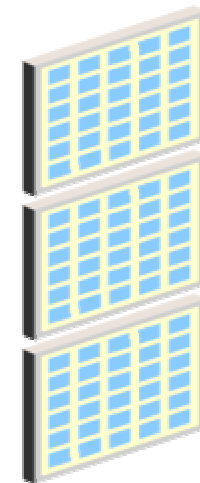
- Sortierung einer Trefferliste (Default)
  - Reverse DOCID (9.2)
  - Forward DOCID (10.x)  
Reverse DOCID als Option verfügbar
- Nutzung für Datumssortierung
  - Dokumente werden chronologisch sortiert
  - Keine UPDATES
  - Keine parallelen Index Builds

Nicht verfügbar bei

- Progressive Relaxation
- Transactional Queries

# Tabellen partitionieren

- Prinzip
  - Aufteilung in mehrere physikalische Einheiten
  - Nach außen **eine normale** Tabelle
  - Oracle TEXT unterstützt partitionierte Tabellen
- Nutzen: Performance
  - Abfragen können auf einzelne Partitionen beschränkt werden (Optimizer)
  - Partitionen können als Einheit gelöscht oder verschoben werden



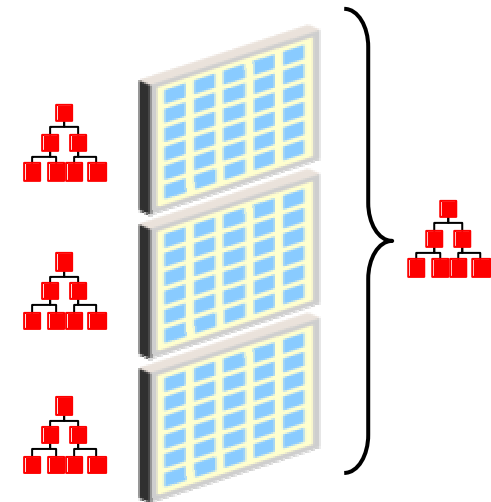
# Tabellen partitionieren

## Nutzen für Sortierung nach Datum

- Partitionierungsschlüssel: Datum
  - Relationaler Spalte
  - Bei XML: Im Dokument (O/R Speicherung)

### SQL-Abfrage

```
SELECT * FROM (  
  SELECT title FROM text_tab  
  WHERE CONTAINS(  
    dokument, 'textquery'  
  ) > 0  
  ORDER BY datum  
) WHERE ROWNUM < 20;
```



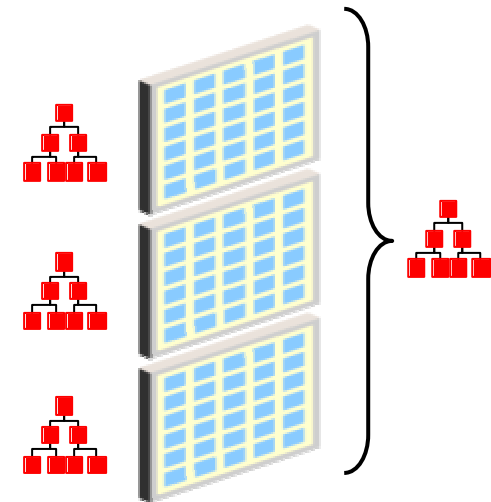
# Tabellen partitionieren

## Weiterer Nutzen ...

- "Rolling Window"-Operationen
  - Partitionen auf einmal löschen (mit Index)
  - Index Rebuild auf Partitionsebene
  - Paralleles Index Build (und Rebuild)

```
alter table dokument_tab  
drop partition part2005
```

```
alter index idx_volltext  
rebuild partition part2005
```



# Progressive Relaxation

- Was ist das?
  - Erweiterung der Textquery ...
  - ... schrittweise ...
  - ... bis die gewünschte Trefferzahl erreicht ist.

```
select score(1), title from test_table
where contains (
  dokument,
  '<query>
    <textquery>
      <progression>
        <seq>mdata (month, 122005)</seq>
        <seq>mdata (month, 112005)</seq>
        <seq>mdata (month, 102005)</seq>
        :
```

# Mehrsprachigkeit

- MULTI\_LEXER vs. WORLD\_LEXER

Feature	MULTI_LEXER	WORLD_LEXER
Definition and Setup	use ctx_ddl API to set up individual lexers and collect them into a multi-lexer	no programming needed
Multi-lingual corpora (mono-lingual documents)	Supported	Supported
Multi-lingual documents	Not Supported	Supported
Database charset	Any	Unicode only (AL32UTF8 and UTF8)
Set attributes of each language	Supported	Not Supported
Document language ID	User-identified, per document	Not needed
Query language identification	User-identified, per query	Not needed
Arabic	Keyword search only	Arabic-specific features
Chinese/Japanese	VGRAM or Segmentation	VGRAM only

*Quelle: Metalink-Note 249991.1*

# Anhang: XML und Oracle TEXT

- SECTION GROUPS
  - XML\_SECTION\_GROUP
  - AUTO\_SECTION\_GROUP
  - PATH\_SECTION\_GROUP
- Operatoren
  - WITHIN
  - INPATH / HASPATH

# Anhang: XML und Oracle TEXT

## AUTO\_SECTION\_GROUP

- Prinzip:
  - Standardmäßig vollständige Indizierung
  - Keine Deklaration von Abschnitten erforderlich
  - Deklaration von sog. "STOP"-Abschnitten möglich
- CONTAINS-Operatoren
  - WITHIN
- Vorteile
  - Implementierung einfacher
- Nachteile
  - Keine XPath-ähnliche Syntax möglich

# Anhang: XML und Oracle TEXT

## PATH\_SECTION\_GROUP

- Prinzip
  - Stets vollständige Indizierung des Dokumentes
- CONTAINS-Operatoren
  - WITHIN
  - INPATH und HASPATH
- Vorteile
  - XPath-Ähnliche Syntax möglich
  - Einfache Implementierung
- Nachteile
  - keine Möglichkeit zum Ausschließen bestimmter Abschnitte
  - Indexumfang größer

# INPATH / HASPATH

## XML-Standards

- XML Standards
  - keine 100%-Unterstützung des Standards
  - ABER: XPath definiert keine Volltextsuche
  - keine Namensräume
  - keine Unterstützung für Benutzerdefinierte Entities

# Weitere Informationen

- Oracle Dokumentation
  - Oracle TEXT Developers' Guide
  - Oracle TEXT Reference
- Oracle Technology Network (OTN)
  - Database → Content Management → Text
- Metalink
  - Note 249991.1: Oracle TEXT technical Overview
  - Note 150453.1: Strategy for creating Oracle TEXT indexes

Q U E S T I O N S  
A N S W E R S



ORACLE®