Information Lifecycle Management

Dr. Frank Haney

Vorstellung

Selbständiger Oracle-Berater seit 2002

- OCP DBA
- Oracle University zertifizierter Trainer
- Auditleiter für geprüfte IT-Sicherheit

Administration

- Implementierung und Test
- Hochverfügbarkeit
- Backup und Recovery
- Migration und Upgrade
- Performance Diagnostik und Tuning
- Sicherheit

Applikationsentwicklung

- Design
- SQL und PL/SQL
- Oracle Text
- Oracle und XML
- Apex
- Objektrelationale Erweiterungen

Agenda

- Anforderungen an das Information Life Cycle Management (ILM)
- Basisfeatures für das ILM
  - Flashback Data Archive
  - Partitionierung
    - Hilfsmittel für eine optimale Partitionierungsstrategie
      - ILM Assitant
      - SQL Access Advisor
  - Komprimierung
- Neu in Oracle 12c
  - Heat Maps
  - Automatic Data Optimization (ADO)
  - Temporal Validity
  - In-Database Archiving

Worum geht es bei ILM?

Definition:

ILM beschreibt eine Speicherstrategie. Ziel dieser Strategie ist die Speicherung von Informationen entsprechend ihrem Wert auf dem jeweils günstigsten Speichermedium einschließlich der Regeln und Prozesse, wie Information auf die geeigneten Speichermedien verschoben wird.

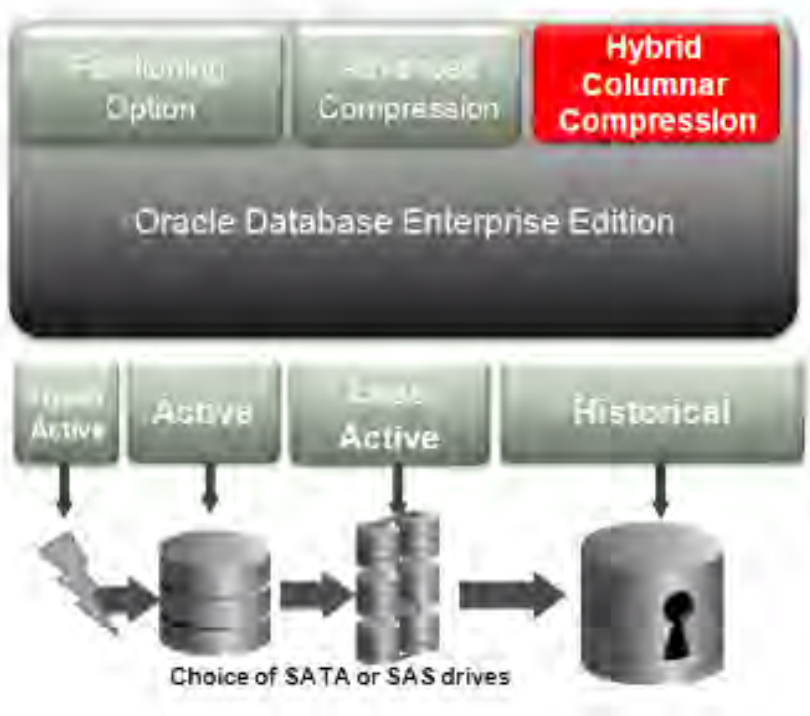
Die Steuerungsmechanismen der Verwaltung und der Speicherung orientieren sich an *Wichtigkeit*, *Wertigkeit* und *Kosten* der elektronischen Information. Hierfür wird eine Klassifizierung der Daten, Quellen und Speichersysteme vorgenommen, die innerhalb einer Speicherhierarchie die automatisierte Bereitstellung erlauben.

Quelle:
https://de.wikipedia.org/wiki/Informationslebenszyklusmanagement#Definition

Lifecycle Management mit Oracle

- Definition der Anforderungen (Datenklassen)
  - Relevanz der Daten für das Geschäft
  - Workflow der Daten innerhalb der Firma
  - Sichtbarkeit und Manipulierbarkeit der Daten entsprechend dem Lebenszyklus
  - Grad an Verfügbarkeit und Sicherheit
  - Erforderliche Performance beim Zugriff
  - Aufbewahrungsfristen aus geschäftlicher und rechtlicher Sicht
- Definition logischer Speicherbereiche für die Datenklassen
- Definition des Lifecycle (Was wird wo, wie und wie lange gespeichert)
- Zuweisung des Lifecycle zu Tabellen und Partitionen
- Definition und Erzwingen der entsprechenden Policies

Features für das ILM



Different portions of the data are assigned to different tiers based on the cost, performance, availability, and security requirements for each portion.

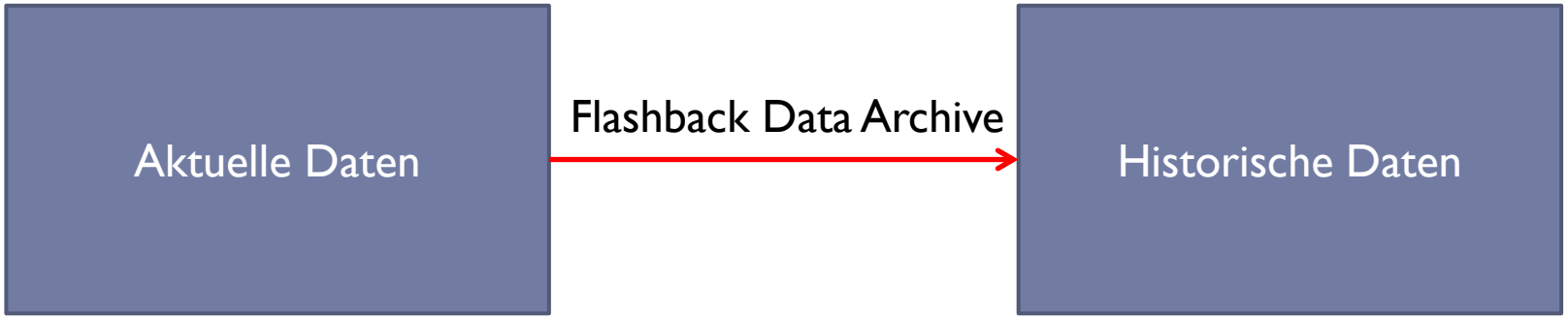
Data are compressed with Advanced Compression and/or Hybrid Columnar Compression across storage tiers – 3X to 15X compression is typical.

- *Hybrid Columnar Compression* erfordert ZFS (12.1 nur mit Exadata)
- *Partitioning* und *Advanced Compression* sind Optionen der EE

Basisfeatures für das ILM

- **Flashback Data Archive**
  - Langzeitspeicherung relevanter Transaktionen
  - Daten stehe für das Reporting zur Verfügung
  - Policy-gestütztes automatische Löschen alter Daten
- **Partitioning** (Partitionierung): Aufteilen von Tabellen mit dem Ziel
  - getrennter Speicherung entsprechend der ILM-Policy
  - performanteren Zugriffs auf Teile der Daten
  - besserer Wartbarkeit bei großen Tabellen
- **Compression** (Komprimierung der Daten) mit dem Ziel der
  - Reduktion des Speichervolumens und damit der Kosten
  - Neue Features Heat Map und ADO

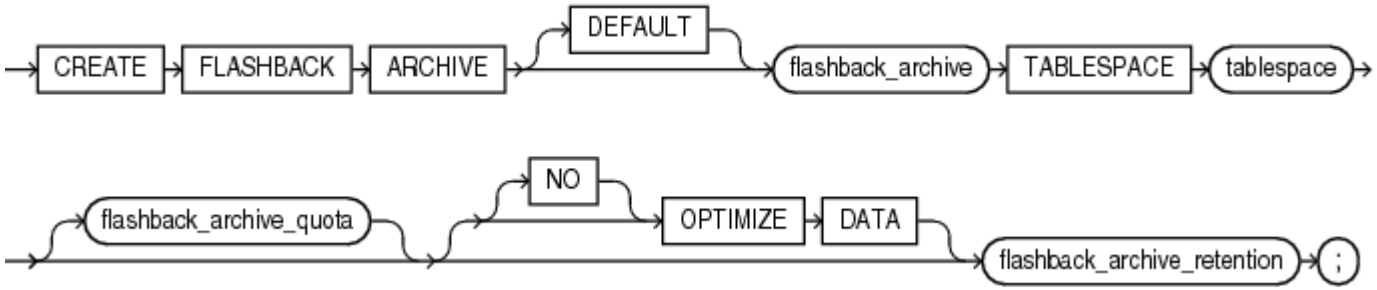
Flashback Data Archive (Total Recall)



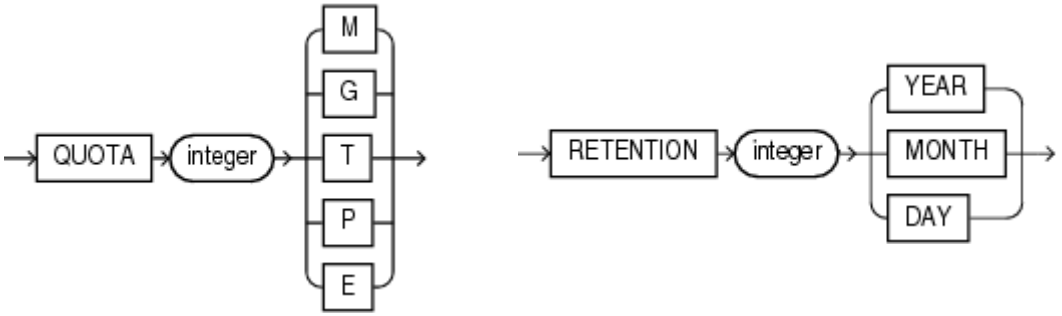
- Bestandteil der *Advanced Compression Option*
  (ohne Komprimierung ab 12.1 frei, auch in SE2, backported nach 11.2.0.4!)
- Neu in 12*c*:
  - User-Kontext der Änderungen nachverfolgbar
  - Tabellen können in einer „Application“ zusammengefasst und gemeinsam dem Flashback Data Archive hinzugefügt werden.
  - Import und Export der History

Anlegen eines Flashback Data Archive

- Syntax:



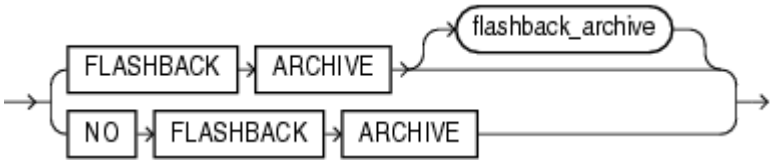
- Platz für das FDA und Aufbewahrungsdauer



- **`OPTIMIZE DATA`** bewirkt, daß die archivierten Daten unter Nutzung eines der folgenden Features optimiert werden: Advanced Row Compression, Advanced LOB Compression, Advanced LOB Deduplication, segment-level compression und row-level compression. (Dann aber *Advanced Compression Option* zu lizenzieren.)

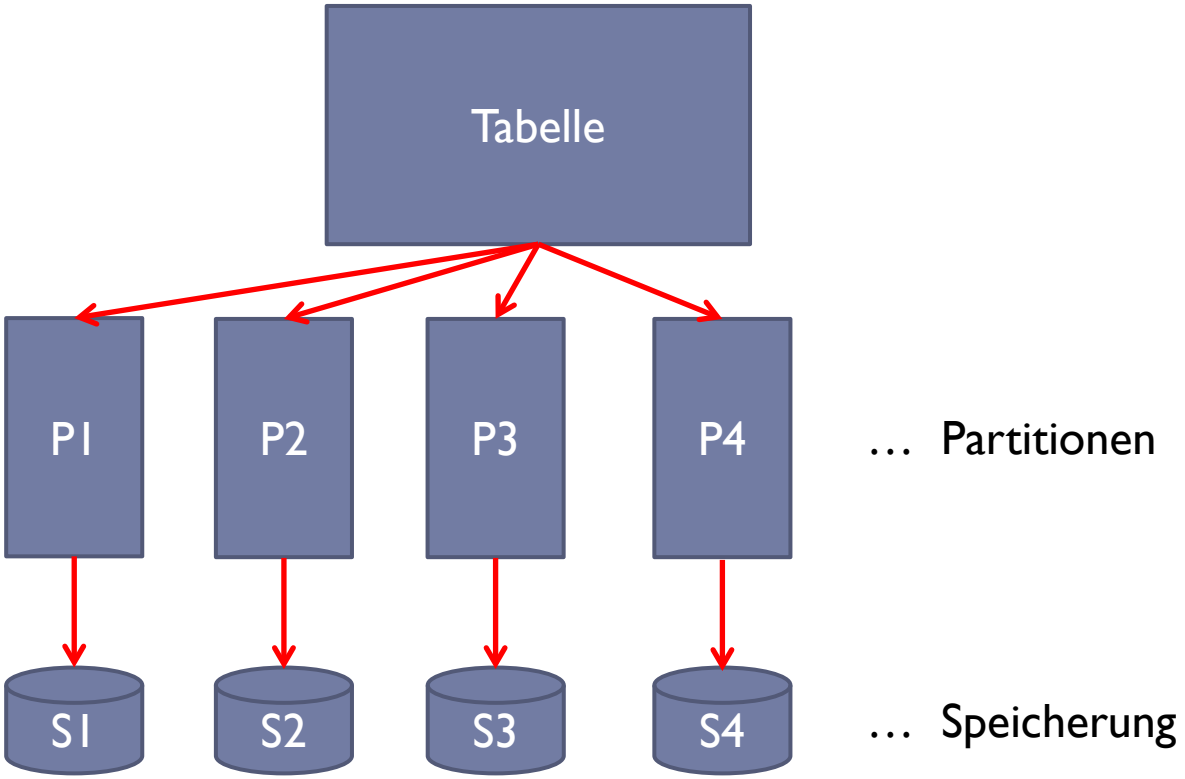
Verwendung des Flashback Data Archive

- **`FLASHBACK ARCHIVE`**-Klausel von **`CREATE`** und **`ALTER TABLE`**



- Abfragen mit der Klausel **`AS OF TIMESTAMP`** timestamp von SELECT
- *Flashback Version Query* wird unterstützt: **`VERSIONS BETWEEN { SCN | TIMESTAMP } start AND end`**
- Erlaubte DDL
  - **`ALTER TABLE`**:
    - Hinzufügen, Löschen, Umbenennen und Ändern von Spalten
    - Hinzufügen, Löschen und Umbenennen von Constraints
    - Löschen oder Leeren von Partitionen oder Subpartitionen
  - **`TRUNCATE TABLE`**
  - **`RENAME`** (Umbenennen von Tabellen)
- Verbotene DDL
  - Verschieben, Splitten, Mergen und Zusammenführen von Partitionen oder Subpartitionen,
  - Tabellen verschieben
  - LONG in LOB umwandeln

Partitionierung (Prinzip)



Partitionierung

- Traditionell
  - Range (Oracle 8)
  - Hash (Oracle 8i)
  - List (Oracle 9i)
- Neu in 11g
  - Reference (Partitionierung über Fremdschlüssel)
  - Interval (Auto-Range)
  - Virtual Column
  - System (manuell)
  - Beliebige Kombinationen von Range, Hash und List (Subpartitionen)

Neu in 12c

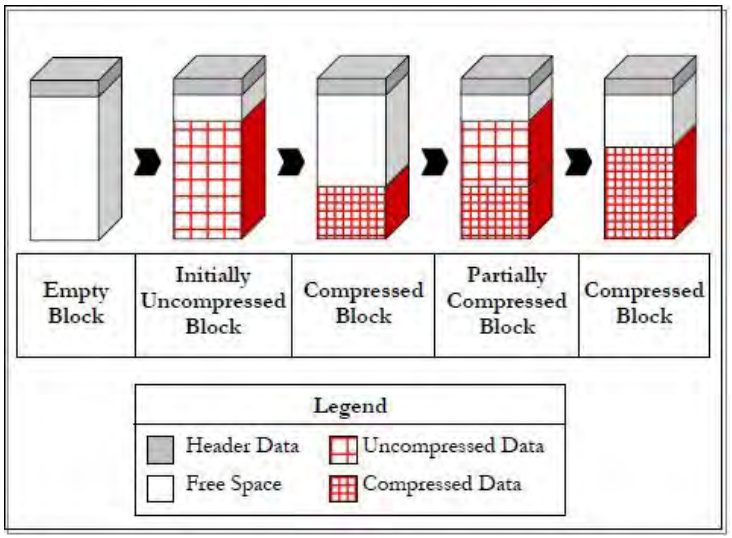
- Interval-Reference (12.1)
- Auto-List (12.2.)
- Online-Partitionierung nichtpartitionierter Tabellen (12.2.)
- Partitionierung externer Tabellen (12.2.)
- Read-only Partitionen (12.2.)

Hilfsmittel für die Partitionierung

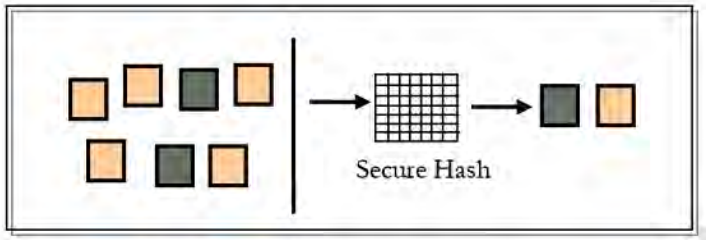
- SQL Access Advisor
  - Empfiehlt Tabellen als Partitionierungskandidaten
  - Aber keine ILM-spezifischen Empfehlungen
  - Bestandteil des *OEM Tuning Pack*
- ILM Assistant
  - Gibt Empfehlungen auf der Basis von ILM Policies
  - Wird als Schema in die Datenbank installiert
  - APEX-Applikation
  - Nicht mehr taufrisch (2009)
  - Aufgabe wird von *Heat Map* und *Automatic Data Optimization* (ADO) übernommen und automatisiert – siehe dort

Komprimierung – Beispiele

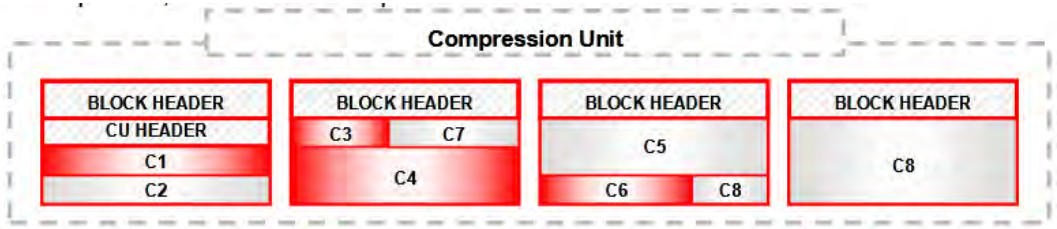
Advanced Row Compression



LOB Deduplication



Hybrid Columnar Compression

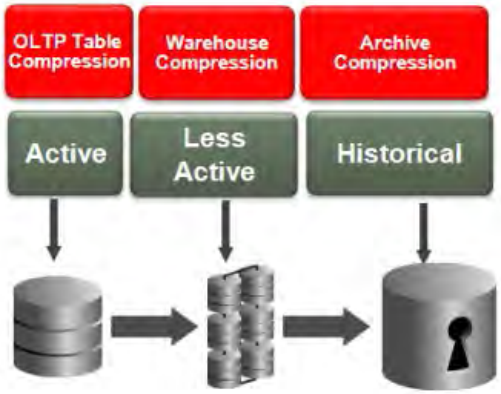


Komprimierung – Verwendung

Methode	Kompressions-rate	CPU Overhead	Verwendung	Bemerkungen
Basic table compression	Hoch	Minimal	DSS	Keine
Advanced row compression	Hoch	Minimal	OLTP, DSS	Keine
Warehouse compression (Hybrid Columnar Compression)	Am höchsten	Höher	DSS	Kompressionsniveau und CPU Overhead hängen von der Einstellung LOW oder HIGH ab.
Archive compression (Hybrid Columnar Compression)	Am höchsten	Am höchsten	Archivierung	Kompressionsniveau und CPU Overhead hängen von der Einstellung LOW oder HIGH ab.

Implemetierungsmöglichkeiten

- OLTP



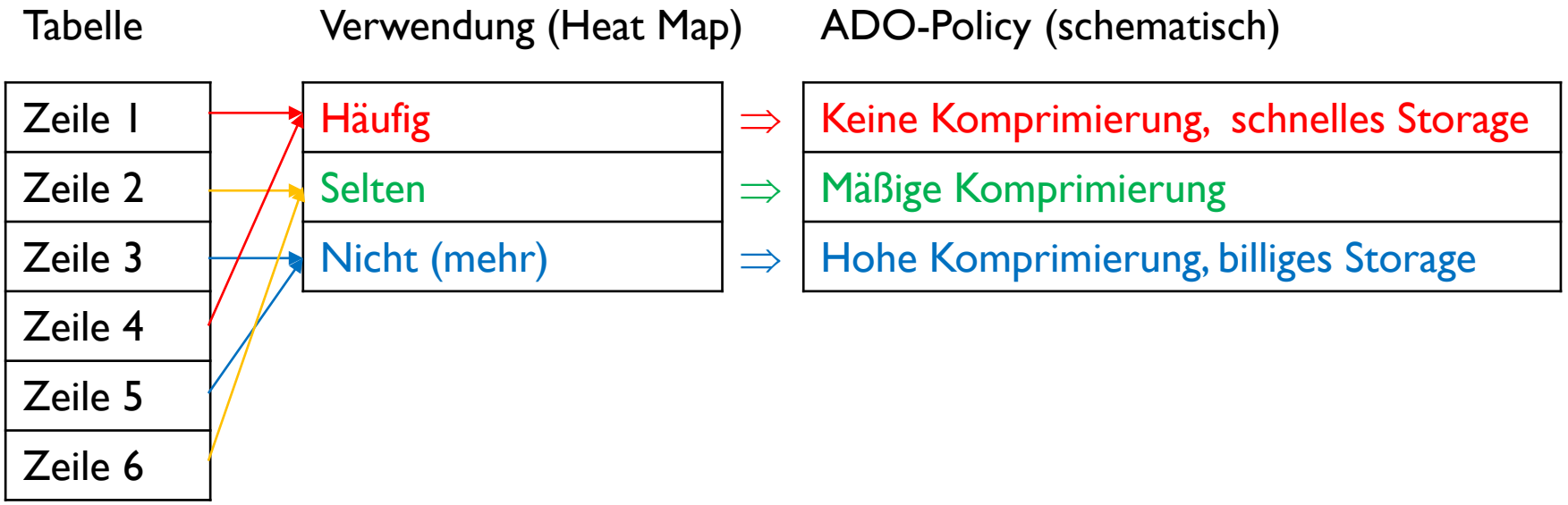
- Data Warehouse

Heat Map und ADO

Zwei zusammengehörige ILM-Features:

- Heat Map: Die Datenverwendung wird zeilen- und segmentweise aufgezeichnet
  Häufigkeit und letzter Zugriff bezüglich
  - Lesen und Schreiben
  - Full Table Scans
  - Index Lookups
- Automatic Data Optimization (ADO)
  Policies für Komprimierung und Umspeicherung von Daten entsprechend der in der Heat Map enthaltenen Informationen auf folgenden Ebenen:
  - Zeile
  - Segment
  - Tablespace
- Erfordert *Advanced Compression* **oder** *In-memory* Option

Heat Map und ADO (Prinzip)



- Einstellen mit

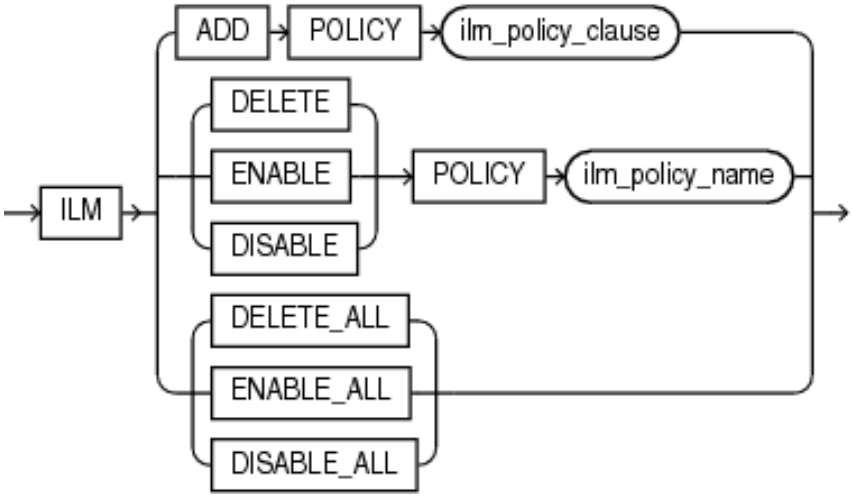
```
ALTER {SYSTEM|SESSION} SET HEAT MAP =
{ON|OFF};
```

Heat Map verwalten

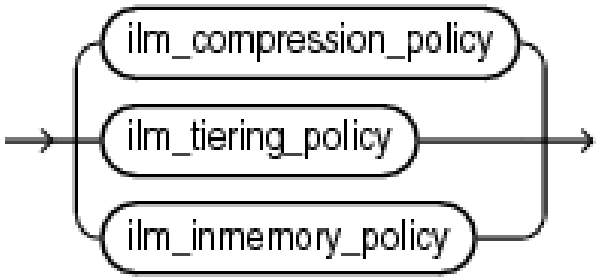
- Informationen erhält man mit:
  - V$HEAT_MAP_SEGMENT
    Echtzeitinformationen über den Segmentzugriff
  - {DBA|ALL|USER}_HEAT_MAP_SEGMENT
    Letzter Segmentzugriff für alle Segmente (Lesen, Schreiben, Full Scan, Index Lookup)
  - {DBA|ALL|USER}_HEAT_MAP_SEG_HISTOGRAM
    Zugriffsinformationen für alle Segmente
  - DBA_HEAT_MAP_TOP_OBJECTS
    Anzeige der 1000 meistverwendeten Objekte
  - DBA_HEAT_MAP_TOP_TABLESPACES
    Anzeige der 100 meistverwendeten Tablespaces

Syntax für ADO Policies

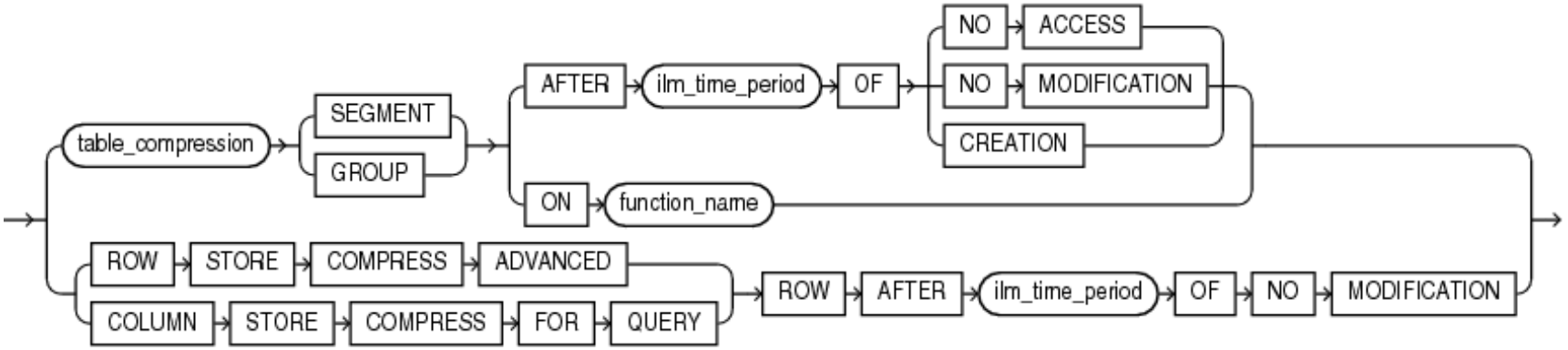
- ILM-Klausel von CREATE oder ALTER TABLE



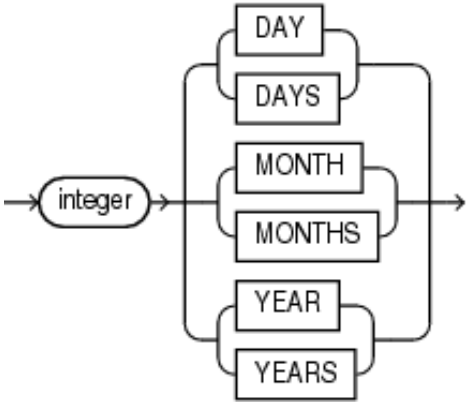
- ILM Policies



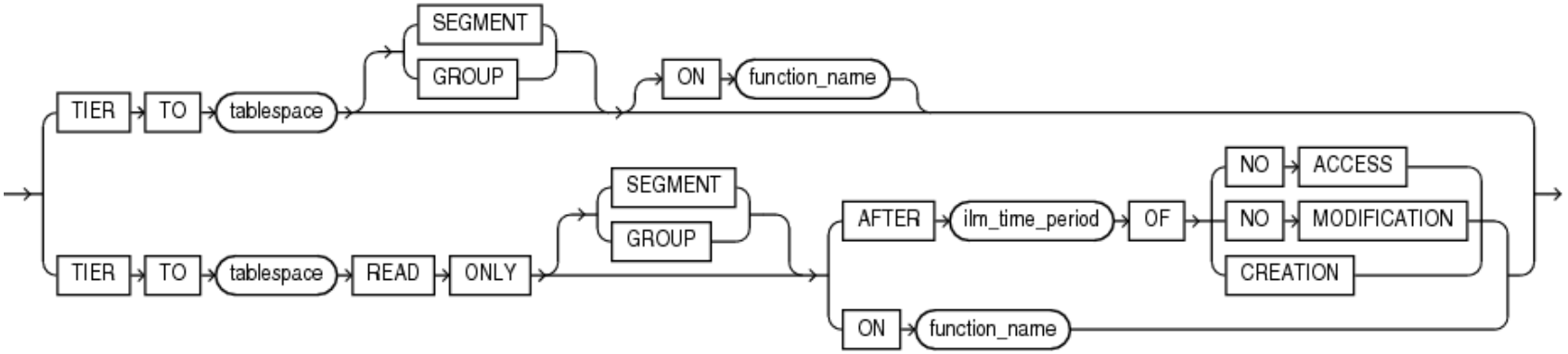
Komprimierungs-Policy



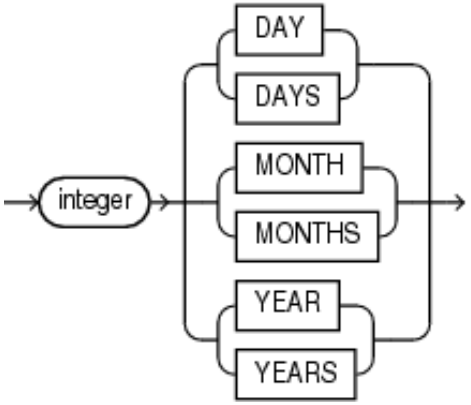
ilm_time_period::=



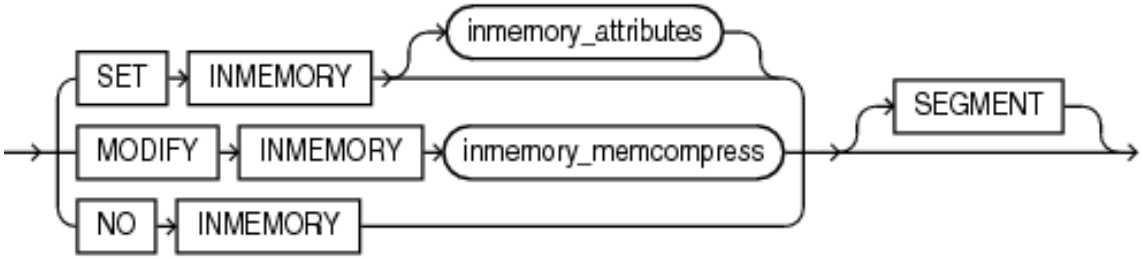
Tiering-Policy

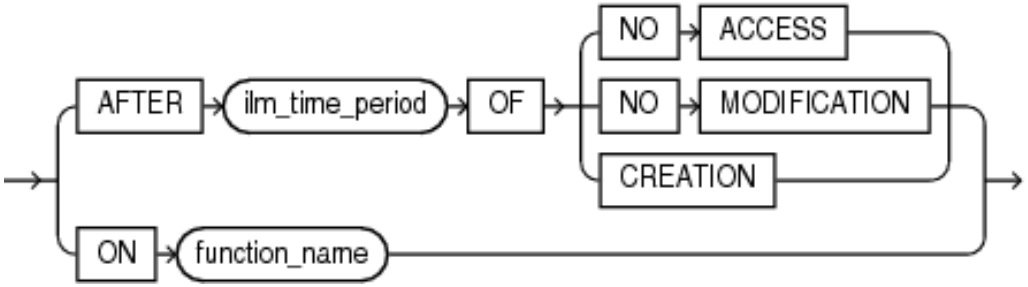


ilm_time_period::=

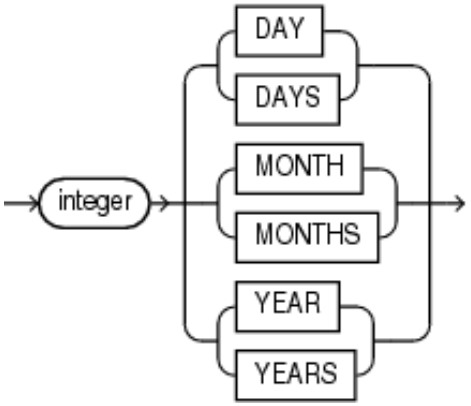


In-memory-Policy (erst 12.2)





ilm_time_period::=



Anmerkungen

- **`COMPRESS`** ist gleich **`ROW STORE COMPRESS BASIC`**
  Komprimierung findet nur bei Direct Path INSERT statt.
- **`ROW STORE COMPRESS ADVANCED`**
  Komprimierung bei allen DML-Operationen aktiv
- **`COLUMN STORE COMPRESS FOR {QUERY | ARCHIVE}`**
  Komprimierung in einem spaltenorientierten Verfahren (Hybrid Columnar Compression) jeweils **`LOW`** oder **`HIGH`** – Geeignet für wenig frequentierte Daten:
  - QUERY im DWH mit Komprimierung
  - ARCHIVE für Langzeitspeicherung ohne Zugriff
- Policies erhalten systemgenerierte Namen
- **Achtung: Advanced Compression Option!**

Temporal Validity

- Daten erhalten entlang historischer Dimensionen eine Gültigkeit und Sichtbarkeit
  - z.B. wird eine neue Kontonummer erst zu einem bestimmten Datum gültig und sichtbar
  - bei einem Umzug eines Mitarbeiters ist die alte Adresse nach einem Datum nicht mehr gültig und sichtbar
- Man kann sich aber sowohl aktuelle als auch historisch gültige Daten anzeigen lassen
- Klausel **`PERIOD FOR`** in SQL-Statements
- Zwei neue Pseudospalten für Beginn und Ende einer Version
- Achtung: Der gleiche Datensatz kann mit mehreren Versionen in der Tabelle stehen: Das hat Konsequenzen für die Primary Keys.

Temporal Validity - Einrichtung

- Beim CREATE oder ALTER TABLE werden ein oder mehrere Dimensionen mit folgender Syntax angegeben: **`PERIOD FOR dimension_name`** `[( start_time_column , end_time_column)]`
- `dimension_name` ist der Dimensionsname (ohne explizitem Datentyp)
- `start_time_column`, `end_time_column` sind zwei deklarierte Spalten für Beginn und Ende der Version
- Ohne explizite Deklaration wird der Default *`dimension_name`*`_start` und *`dimension_name`*`_end` verwendet. Es handelt sich dann um unsichtbare Spalten (hidden columns).

Temporal Validity einrichten (Beispiel)

```
CREATE TABLE CUSTOMER_1
(
  CUST_NO    NUMBER(12) PRIMARY KEY,
  CUST_NAME  VARCHAR2(20) NOT NULL,
  CUST_ADDR  VARCHAR2(40),
  SERVICE_START      TIMESTAMP,
  SERVICE_END        TIMESTAMP,
  PERIOD FOR CUST_HIST_TIME(SERVICE_START, SERVICE_END)
  );

CREATE TABLE CUSTOMER_2
  (
  CUST_NO    NUMBER(12) PRIMARY KEY,
  CUST_NAME  VARCHAR2(20) NOT NULL,
  CUST_ADDR  VARCHAR2(40),
  PERIOD FOR CUST_HIST_TIME
  );
```

Temporal Validity – Gültigkeit

start_time_column	**end_time_column**	**Gültigkeit**
NULL	NULL	Gültig für **alle** Zeiten
NULL	Wert	Gültig für alle Zeiten **vor** dem end_time_column-Wert
Wert	NULL	Gültig für alle Zeiten **nach** dem start_time_column-Wert
Wert	Wert	Gültig für alle Zeiten **zwischen** den Werten von instart_time_column und end_time_column

Verwendung in Abfragen

- `SELECT * from CUSTOMER_1 AS OF PERIOD FOR CUST_HIST_TIME TO_TIMESTAMP('01-Jun-10');`
- `SELECT * from CUSTOMER_1 VERSIONS PERIOD FOR CUST_HIST_TIME BETWEEN TO_TIMESTAMP('01-Jun-10') AND TO_TIMESTAMP('02-Jun-10');`
- Einstellen der Sichtbarkeit auf Sessionebene mit:
  - `EXECUTE DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME ('ASOF', '31-DEC-12 12.00.01 PM');`
  - `EXECUTE DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME ('CURRENT');`
  - `EXECUTE DBMS_FLASHBACK_ARCHIVE.ENABLE_AT_VALID_TIME ('ALL'); -- Standard`

In-Database Archiving: Prinzip

- Tabellen**zeilen** werden vor der Applikation verborgen, um bestimmten Compliance-Anforderungen zu genügen.
- Das Vorhalten der von der Applikation nicht benötigten Zeilen beeinträchtigt die Performance NICHT.
- Einschalten mit Klausel **`ROW ARCHIVAL`** von CREATE oder ALTER TABLE

In-Database Archiving

Tabelle			Speicherung	
Zeile 1	`ORA_ARCHIVE_STATE=0`	⇒aktuell⇒	Partition 0	Tier 0
Zeile 2	`ORA_ARCHIVE_STATE=1`	⇒veraltet⇒	Partition 1	Tier 1
Zeile 3	`ORA_ARCHIVE_STATE=0`	⇒aktuell⇒	Partition 0	Tier 0
Zeile 4	`ORA_ARCHIVE_STATE=2`	⇒obsolet⇒	Partition 2	Tier 2
Zeile 5	`ORA_ARCHIVE_STATE=1`	⇒veraltet⇒	Partition 1	Tier 1

- Der Session-Parameter **`ROW ARCHIVAL VISIBILITY`** bestimmt, ob historische Zeilen sichtbar sind oder nicht.
- Der Inhalt der **verborgenen Spalte** **`ORA_ARCHIVE_STATE`** bestimmt, ob eine Zeile als historisch gilt oder nicht.
  - **`ROW ARCHIVAL VISIBILITY=ACTIVE`**
    Aktive Zeilen mit **`ORA_ARCHIVE_STATE = 0`** werden angezeigt
  - **`ROW ARCHIVAL VISIBILITY=ALL`**
    Alle Zeilen werden angezeigt, auch **`ORA_ARCHIVE_STATE ≥ 0`**

In-Database Archiving: Verwendung

- Die verborgene Spalte kann als Partitionierungsschlüssel verwendet werden, um ILM-Anforderungen zu berücksichtigen (Storage, Komprimierung etc.)
- CTAS übernimmt diese Informationen nicht!
- Werte setzen und auslesen lassen sich mit der Funktion:

```
DBMS_ILM.ARCHIVESTATENAME
    ( value IN VARCHAR2)
   RETURN VARCHAR2;
```

In-Database Archiving: Beispiel

- Sichtbarkeit einstellen auf aktive Zeilen setzen
```
ALTER SESSION SET ROW ARCHIVAL VISIBILITY= ACTIVE;
```
- Tabelle anlegen
```
CREATE TABLE arc_test (id NUMBER, text VARCHAR2(20))
ROW ARCHIVAL;
```
- Datensätze einfügen (implizit `ORA_ARCHIVE_STATE=0`)
```
INSERT INTO arc_test VALUES (1, 'Ein Text');
INSERT INTO arc_test VALUES (2, 'Noch ein Text');
```
- Alle Datensätze werden angezeigt.
- Datensatz ändern
```
UPDATE arc_test SET ORA_ARCHIVE_STATE=1 WHERE id=1;
```
- Nur Datensatz mit `id=2` wird angezeigt
- Sichtbarkeit auf alle Zeilen setzen
```
ALTER SESSION SET ROW ARCHIVAL VISIBILITY = ALL;
```
- Alle Zeilen werden angezeigt

Vielen Dank!

Lektüre:

- Database VLDB and Partitioning Guide
- PL/SQL Packages and Types Reference
- Database SQL Language Reference

Fragen?