Technische Hochschule Köln
Campus Gummersbach
Fakultät für Informatik und Ingenieurwissenschaften
Studiengang Wirtschaftsinformatik

Bachelorarbeit

zur Erlangung
des akademischen Grades
Bachelor of Science (B. Sc.)
im Studiengang Wirtschaftsinformatik

„Untersuchung des In-Memory-Konzepts in einer Oracle-Database-12c anhand der Analyse von Aktienwerten durch Indikatoren“

Autor: Sebastian Otto
Johann-Kierspel Str. 24
51491 Overath

Telefon-Nr. +491705485764
E-Mail basti.otto@onlinehome.de
Matrikelnummer: 11090185

Erstprüfer: Prof. Dr. Birgit Bertelsmeier
Zweitprüfer: Prof. Dr. Heide Faeskorn-Woyke

Ort: Gummersbach
Abgabetermin: 15.08.2016

Ehrenwörtliche Erklärung und Einverständniserklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig und ohne unerlaubte Hilfe Dritter verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe. Alle Stellen, die inhaltlich oder wörtlich aus Veröffentlichungen stammen, sind kenntlich gemacht. Diese Arbeit lag in der gleichen oder ähnlichen Weise noch keiner Prüfungsbehörde vor und wurde bisher noch nicht veröffentlicht.

_____ _____

Ort, Datum Unterschrift

Abstract

Das Ziel der vorliegenden Bachelorarbeit war es, das In-Memory-Konzept innerhalb einer Oracle Datenbank auf neue Mechanismen, Funktionen und Methoden zu untersuchen. Dazu wurde eine Datenbank erstellt und mit Beispieldaten bestückt. Diese Beispieldaten sind Aktienwerte der DAX-30 Unternehmen, welche durch eine Reihe von Indikatoren aus der technischen Analyse eine komplexe Möglichkeit der Analyse bieten. Die Ergebnisse bestätigen, dass das In-Memory-Konzept neben dem bekannten Spaltenformat eine Reihe von Techniken und Funktionen bietet, welche sich positiv bei der Verarbeitung von Daten durch Data Query Language-Befehle auswirken. Es kommen auch Nachteile, wie der flüchtige Speicher zum Vorschein, dennoch überwiegen die Vorteile stark. Nach einer Reihe von Tests wird deutlich, dass Objekte, die in den In-Memory-Column-Store geladen werden, nur 30 % der Zeit benötigen, um gelesen zu werden. Dies ist für die Verarbeitung von großen und komplexen Daten eine deutliche Verbesserung. Die Bachelorarbeit richtet sich an Studierende der Fachbereiche Informatik und BWL sowie an Interessierte im Bereich Datenbanken.

Inhaltsverzeichnis

Abkürzungsverzeichnis

ARO	Aroon
ATR	Average True Range
BB	Bollinger Band
CDB	Container Database
CPU	Central Processing Unit
CU	Column Unit
DB	Datenbank
DML	Data Manipulation Language
DQL	Data Query Language
EMA	Exponential Moving Average
GD	Gleitender Durchschnitt
IMCO	In-Memory Coordinator
IMCS	In-Memory Column Store
IMCU	In-Memory Compression Unit
MA	Moving Average
MACD	Moving Average Convergence/Divergence
MOM	Momentum
OBV	On Balance Volume
PDB	Pluggable Database
PGA	Programm Global Area
RAM	Random-Access Memory
ROC	Rate of Change
RSI	Relative Strength Index
SGA	System Global Area
SIMD	Single Instruction Multiple Data
SMCO	Space Management Coordinator
SMU	Snapshot Metadata Unit
SQL	Structured Query Language
SW	Standardabweichung
UGA	User Global Area
WPR	Williams Percent Range
W%R	Williams Percent Range

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcodeverzeichnis

Formelverzeichnis

1. Einleitung

In der heutigen Zeit sind Themen wie Big Data, Data Mining und analytische Datenbanken im Bereich *Datenbanken* unvermeidlich. Zu diesem Zweck muss eine große Anzahl an Daten untersucht werden, die durch ineffiziente Suchanfragen immer langsamer und oft auch doppelt verarbeitet werden. Der Wunsch von Reporting in Echtzeit und Real-Time-Analysen förderte die Entwicklung von Lösungsansätzen.

Nachdem der Hauptspeicher immer günstiger und die Prozessoren immer schneller geworden sind, müssen nun die alten Zugriffsverfahren neu modelliert werden. Die Option, dass die Daten im Hauptspeicher liegen, gibt es schon länger. Dieser Bereich wird als Buffer-Pool oder auch Buffer-Cache bezeichnet. Mit der Neuerung der spaltenorientierten Datenbanken wurde das *In-Memory*-Konzept entwickelt. Im Gegensatz zu anderen *In-Memory*-Techniken können bei der Oracle-Variante auch einzelne Objekte in den *In-Memory*-Speicher geladen werden, sodass nicht alle Daten verschoben werden müssen.

Die vorliegende Arbeit untersucht daher das *In-Memory*-Konzept in einer Oracle-Datenbank.

Zuerst wird dafür das Grundkonzept der *In-Memory*-Option in Kapitel 2, sowie die möglichen Vorteile, die von Oracle genannt werden, erläutert.

In Kapitel 3 werden daraufhin die theoretischen Grundkenntnisse über die Speicherarchitektur innerhalb einer Oracle-Datenbank (DB) geschaffen. Dafür werden die Speicherbereiche *User Global Area, Program Global Area, System Global Area und Software Code Area* im Detail erläutert. Zusätzlich sind die verschiedenen Teilbereiche der Speicherbereiche sowie deren Funktionen näher beschrieben.

Kapitel 4 untersucht im Detail das *In-Memory*-Konzept und erläutert Methoden, Funktionen und Mechanismen, auf denen dieses basiert. Zu Beginn des Kapitels wird der *In-Memory-Column-Store* vorgestellt und dieser dann mit dem Zeilenformat (*ROW*) verglichen. Weiterhin werden die für die Aktivierung der *In-Memory*-Option nötigen Schritte sowie der Ablauf des Ladens der Daten in den *In-Memory-Column-Store* dargestellt. Danach werden die Mechanismen *In-Memory-Storage-Index*, *Single Instruction Multiple Data* sowie die Hintergrundprozesse *In-Memory-Coordinator*, *Space Management Coordinator* und *Wnnn* erklärt. Diese sind ein wichtiger Bestandteil des Konzepts und bringen eine Performancesteigerung mit sich. Als weitere Neuerung der *In-Memory*-Option werden die neuen Kompressionstechniken, die Initialisierungsparameter, der Einsatz in einer multimandantenfähigen Umgebung sowie die Prioritäten-Funktion vorgestellt. Zum Abschluss des Kapitels werden die Vor- und Nachteile der *In-Memory*-Option aufgezählt. In Kapitel 5 werden nun auf Grundlage der Datenbasis durch ausgewählte Indikatoren Prognosen für

Aktienkurse erstellt. Dazu werden die Indikatoren sowie deren Berechnung im Detail erläutert und in ein Punktesystem integriert. Die Berechnung der Prognosen gibt daraufhin einen Einblick in die Performancesteigerung von Suchanfragen, die durch die *In-Memory*-Option entsteht, und schafft dadurch neue Möglichkeiten der Real-Time Analyse.
Kapitel 6 erklärt die Aktivierung der *In-Memory*-Option für die Tabelle Aktien und beschreibt wie anhand der Indikatoren die *In-Memory*-Option getestet wird.

1.1 Datenbasis

Die Datenbasis besteht in dieser Arbeit aus Aktienwerten, die von der öffentlich zugänglichen Yahoo-Finance-Webseite[1] exportiert wurden. Diese Aktienwerte bestehen aus Tageswerten, die in den letzten Jahren erfasst wurden. Zu den Tageswerten zählen der Startkurs, der Höchstkurs, der Tiefstkurs, der Schlusskurs, der angepasste Schlusskurs sowie das Volumen für ein bestimmtes Datum. Diese Daten wurden in eine Excel-Tabelle importiert und dort für die spätere Untersuchung vorbereitet. Zusätzlich wurden in Excel alle Zellen mit leeren Aktienkursen entfernt. Danach sind die Daten in die Oracle Database 12c importiert worden.
Die Eingrenzung der Aktienwerte erfolgte auf die DAX 30 – Unternehmen, die zu den größten und umsatzstärksten an der Frankfurter Wertpapierbörse gelisteten Unternehmen zählen. Bei manchen Unternehmen sind teilweise Tageswerte noch vor dem Jahr 2000 vorhanden. Der aktuellste Tageswert ist vom 15.04.2016.

1.2 Technisches Umfeld

Das technische Umfeld für diese Arbeit besteht aus einer virtuellen Maschine mit 16 GB Arbeitsspeicher, 126 GB Festplattenspeicher und dem Prozessor „Intel(R) Xeon(R) CPU E5-2660 0 @ 2.20GHz (4 CPUs), ~2.2GHz“, welche von der TH-Köln zu Verfügung gestellt wurde. Auf dieser virtuellen Maschine wurde vorher das Betriebssystem „Windows Server 2012 R2 Datacenter 64-bit“ eingerichtet. In der folgenden Abbildung ist die Version der installierten Oracle-DB dargestellt:

NLSRTL	12.1.0.2.0	Production
Oracle Database 12c Enterprise Edition	12.1.0.2.0	64bit Production
PL/SQL	12.1.0.2.0	Production
TNS for 64-bit Windows:	12.1.0.2.0	Production

Abbildung 1-1: Oracle-DB-Version[2]

[1] *[I_YA_16]*
[2] *Aus der Datenbank importiert*

2. Grundkonzept der In-Memory-Option

Mit Einführung des *In-Memory*-Konzeptes in die Oracle-DB 12.1.0.2 besteht für eine einzelne DB die Möglichkeit, den gemischten Arbeitsaufwand effizient zu unterstützen und eine optimierte Leistung für Befehle der *Data Query Language (DQL)* zu liefern. Zusätzlich wird die Real-Time-Analyse und Berichtserstattung unterstützt.[3] Die *In-Memory*-Option gibt den Endbenutzern die Möglichkeit, mehrere Abfragen in der Zeit auszuführen, die vorher benötigt wurde, um eine Abfrage auszuführen. Die Neuerung dieses Features erlaubt es, Tablespaces, Tabellen, Partitionen und materialisierte Views im *Column*-Format und nicht im typischen *ROW*-Format zu speichern. Da das *Column*-Format ein spaltenbezogener Speicher ist, wird der *In-Memory*-Speicher auch als *Column-Store* (*IMCS*)[4] bezeichnet. Als Folge dessen, kann die Oracle-DB 12c zwei verschiedene Formate der Architektur unterstützen.[5] Für den Fall, dass Daten zum Lesen oder Schreiben (*Data Manipulation*) aufgerufen werden, werden diese Daten im traditionellen ROW-Speicher verändert. Der ROW-Speicher verwendet dafür einen Zwischenspeicher, der häufig verwendete Daten für einen schnelleren Zugriff bereithält.

Daten, die nur zum Lesen bzw. zum Analysieren angefragt werden, sind meistens im neuen *In-Memory*-Spaltenspeicher angesiedelt.[6] Dafür werden ausgewählte Daten vorher aus dem ROW-Format in das *Column*-Format geladen. Das bedeutet, dass, sobald eine Transaktion etwas einfügt, aktualisiert oder löscht, die neuen Daten gleichzeitig in beiden Speichern erscheinen. Außerdem speichern beide Methoden die Daten transaktionsorientiert konsistent.[7] Die Oracle-Datenbank entscheidet selber, welches Format für einen Befehl optimal ist.

Da der *In-Memory*-Speicher ein Teil der *System Global Area* ist und eine zusätzliche Speicherzuweisung benötigt, gibt es weitere Aspekte zu beachten. Im Folgenden Kapitel 3 werden dazu unter anderem die Komponenten der *System Global Area* und die entsprechenden Zuweisungen erläutert.

[3] *Vgl. [ORA_WP_15] Kap. Introduction, S. 4.*
[4] *Vgl. [B_AG_15] S. 702.*
[5] *Vgl. [B_SG_15] Kap. 2, S. 68.*
[6] *Vgl. [ORA_MC_15] Kap. Introduction, S. 4.*
[7] *Vgl. Kap. 4.2.*

3. Speicherarchitektur in einer Oracle-Datenbank

Bei der Verwendung der *In-Memory*-Option müssen ggfs. Speicherbereiche bzw. Speicherpools angepasst werden. Damit diese optimal angepasst werden können, stehen in diesem Kapitel die Speicherstrukturen, die innerhalb einer Oracle-DB vorhanden sind, im Mittelpunkt. Hierfür werden die vier verschiedenen Speicherstrukturen, die die Basis für das Speichermanagement bilden, beschrieben. Zusätzlich werden diese noch einmal in dessen Unterstrukturen unterteilt und erklärt, für welche Funktion diese Speicher gedacht sind. Insbesondere wird in Kapitel 3.2 die Lage des *In-Memory-Column-Stores* innerhalb der DB beschrieben.

3.1 User Global Area (UGA)

Die *User Global Area* ist ein Speicherbereich für die aktuelle Session des Datenbankbenutzers. Diesem Speicher werden Session-Variablen wie Login-Informationen und andere Informationen über die aktuelle Datenbank-Session zugewiesen. Im Wesentlichen speichert die *User Global Area* den Status der Session für Benutzerprozesse.[8]

In der *User Global Area* kann sich zusätzlich der *OLAP-Pool* befinden.[9] Dieser ist für die Verwaltung von OLAP-Datenseiten, welche äquivalent zu Datenblöcken sind, zuständig.[10]

3.2 Software Code Area

Die *Software Code Area* ist ein Teil des Speichers, welcher den Code speichert, der gerade ausgeführt wird oder noch ausgeführt werden kann. Dieser Bereich ist normalerweise in höherem Maße exklusiver und besser geschützt als die Lage der Benutzerprogramme. Außerdem ist er auch statisch und wird in der Größe nur verändert, wenn die Software aktualisiert oder neuinstalliert wird.

Außerdem handelt es sich um einen *Read-Only*-Bereich, der entweder für die gemeinsame oder nicht gemeinsame Nutzung installiert werden kann. Generell sollte eine gemeinsame Benutzung bevorzugt werden, da so der verbrauchte Speicher verringert und damit die Leistung verbessert wird.

3.3 Program Global Area (PGA)

Die *Program Global Area* ist ein Speicherbereich, der speziell von ausführenden und nicht geteilten Prozessen und oder Threads, verwendet wird. Damit ist die *PGA* prozessspezifisch und wird niemals im *SGA*[11] zugewiesen. Sie ist ein dynamischer Speicher, welcher auf die

[8] *Vgl. [B_SD_13] Kap. 5, S. 172.*
[9] *Vgl. [B_MH_16] Kap. 2.3, S. 81.*
[10] *Vgl. [ORA_TK_15] Kap. 14, S. 2.*
[11] *Vgl. Kap. 3.3.*

Session bezogene Variablen bereithält, die von Server-Prozessen benötigt werden. Der Serverprozess wiederum teilt auch Speicherstrukturen zu, die er aus der *PGA* benötigt. Die Bereiche der *PGA* haben alle verschiedene Verwendungszwecke.[12]

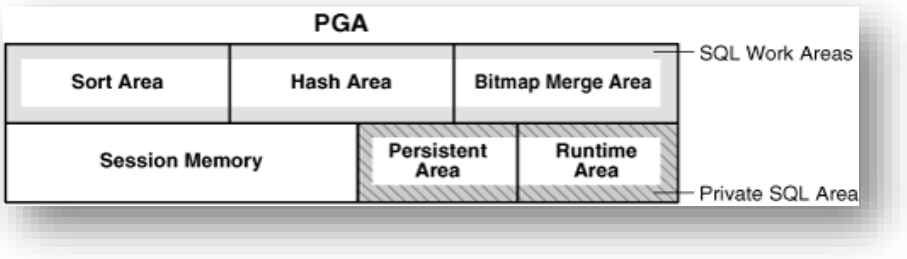

Abbildung 3-1: Speicherbereiche innerhalb der PGA[13]

Die Art und Weise, wie der *PGA*-Speicher zugewiesen werden kann, wird durch das *Automatic PGA Memory Management* vereinfacht und verbessert. Diese automatische *PGA*-Speicherverwaltung ist standardmäßig aktiviert und sorgt dafür, dass die Speichergröße der *PGA* innerhalb der Oracle-DB nicht manuell festgelegt werden muss. Dabei werden die Mengen an *PGA*-Speicher dynamisch so angepasst, dass sie auf 20 % des *SGA*-Speichers, der den *Work Areas* zugehörig ist, basieren.

Zu den *Work Areas* gehören unter anderem die *Sort Area*, die *Hash Area* und die *Bitmap Merge Area*. Diesen Bereichen wird der privat zugewiesene *PGA*-Speicher zur Bewältigung von komplizierten Abfragen, die den Speicher intensiv beanspruchen, zur Verfügung gestellt.[14]

Den Sortier-Operator verwendet die *Sort Area* für das Sortieren einer Anzahl von Zeilen. Dazu äquivalent wird die *Hash Area* zum Aufbau einer Hash-Tabelle, und die *Bitmap Merge Area*, zum Verdichten von Daten, die aus mehreren Untersuchungen von Bitmap-Indizes stammen, verwendet.[15]

In der *Private SQL Area* werden Informationen über geparste SQL-Ausdrücke und andere session-spezifische Informationen für die Verarbeitung bereitgehalten.[16] Wenn ein Server-Prozess einen SQL- oder PL/SQL-Code ausführt, speichert dieser Prozess die Werte von Bind-Variablen, den Status von ausgeführten Abfragen sowie ausgeführte Abfragen aus den *Work Areas* in der *Private SQL Area* ab.[17] Die *Private SQL Area*, welche sich in der *PGA* befindet, ist von der *Shared SQL Area*, welche Ausführungspläne in der *SGA* bereithält[18], zu

[12] *Vgl. [ORA_LA_15] Kap. 14, S. 5/6.*
[13] *[ORA_LA_15] Kap. 14, S. 6 Abbildung „Figure 14-4 PGA Contents“.*
[14] *Vgl. [ORA_RB_16] Kap. 16, S. 1.*
[15] *Vgl. [B_MA_13] Kap. 20.4, S. 386.*
[16] *Vgl. [B_CA_14] Kap. 2, S. 21.*
[17] *Vgl. [ORA_LA_15] Kap. 14, S. 6.*
[18] *Siehe Kapitel 6.3.3.*

unterscheiden. Diese sind jedoch miteinander verknüpft, da mehrere *Private SQL Areas* in der gleichen oder in verschiedenen Sessions auf einen einzigen Ausführungsplan zeigen können.[19] Der Cursor wird häufig in Zusammenhang mit der Handhabung einer spezifischen *Private SQL Area* gebracht.[20] Dies erklärt sich aufgrund der Tatsache, dass der Cursor auf der Seite des Clients als ein Zeiger und auf der Seite des Servers als ein Status dienen kann.
Da der Begriff des Cursors sehr eng mit der *Private SQL Area* verbunden ist, können diese Bezeichnungen ausgetauscht werden.[21] Die Anzahl an *Private SQL Areas*, die durch einen Prozess des Benutzers erstellt werden können, ist durch den *open_cursors* Parameter limitiert. Der Standardwert dieses Parameters liegt bei Anzahl von 50 *Private SQL Areas*.[22]

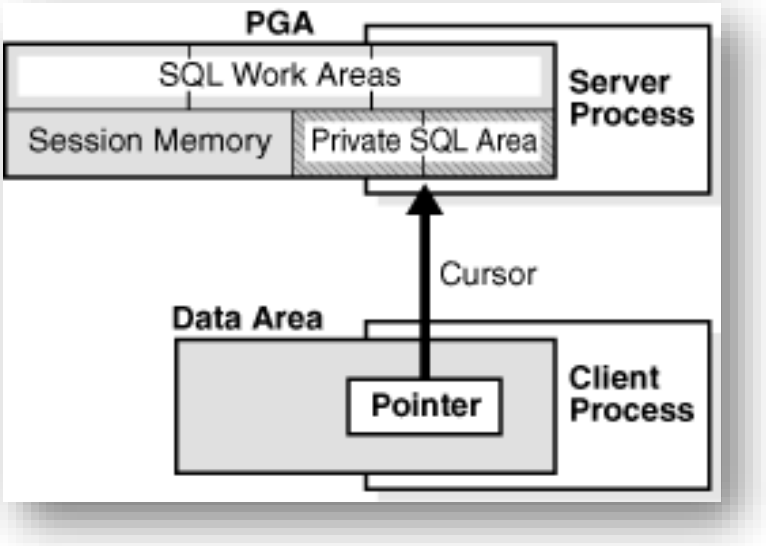

Abbildung 3-2: Private SQL Area – Cursor Konzept[23]

Die *Private SQL Area* ist in die *Run-Time Area* und die *Persistent Area* unterteilt.[24] Eine *Run-Time Area* beinhaltet die Statusinformationen zu einer ausgeführten Abfrage und wird von der Oracle-DB, sobald eine Abfrage gestartet wurde, erstellt. Bei DML-Ausdrücken wird die *Run-Time Area* befreit, sobald der SQL-Ausdruck geschlossen ist. Die *Persistent Area* enthält Werte von Bindevariablen[25], die einen SQL-Ausdruck bei der Ausführung unterstützen und Informationen über den Status der zuletzt ausgeführten Befehle im Cursor enthalten.[26] Sobald

[19] *Vgl. [B_AK_05] Kap.2, S. 27.*
[20] *Vgl. [B_GI_02] Kap. 9, S.226.*
[21] *Vgl. [ORA_LA_15] Kap.14, S. 6.*
[22] *Vgl. [B_AK_05] Kap.2, S. 27.*
[23] *[ORA_LA_15] Kap. 14, S. 6 Abbildung „Figure 14-5 Cursor".*
[24] *Vgl. [B_BT_11] Kap. 8, S. 407.*
[25] *Vgl. [I_FH_11] S. 4.*
[26] *Vgl. [B_CR_06] Kap. 11, S.449.*

der Cursor geschlossen wird, ist dieser Bereich von seiner Aufgabe für diesen Cursor entbunden.

Der Client-Prozess ist zwar für die Verwaltung der *Private SQL Areas* verantwortlich, die Zuweisung und die Auflösung hängen aber in hohem Maß von der Anwendung des Benutzers ab. Hierbei ist zu beachten, dass die meisten Benutzer sich auf die automatische Cursor-Abwicklung verlassen. Es gibt aber die Möglichkeit, über das *Oracle Database programmatic interface* mehr Kontrolle über den Cursor zu erhalten. Dennoch ist es empfehlenswert Cursor immer zu schließen, um den entsprechenden Speicher der Persistent Area freizugeben.[27]

3.4 System Global Area (SGA)

Die *System Global Area* ist ein Speicherbereich, dem der gemeinsam genutzte *Random-Access Memory* (RAM) zugewiesen wird, sobald eine Instanz von Oracle gestartet wird.[28] Hier befinden sich unter anderem Daten und SQL-Ausdrücke, welche zwischen den Oracle-Hintergrundprozessen und den Sever-Prozessen geteilt werden.
Die üblichen Komponenten sind:

- **Database Buffer Cache:** Ist ein Speicher für Daten und Index-Blöcke, um beim erneuten Zugriff auf diese Blöcke den Zugang deutlich schneller zu machen,[29] als beim Zugriff über die normale Festplatte.
- **Shared Pool:** Dieser Speicherpool beinhaltet zuletzt genutzte SQL-Befehle, Auswertungspläne, System-Parameter und *Data Dictionary*-Informationen.[30]
    - **Dictionary Cache:** Ein Speicher über die Informationen zu den *Data Dictionary*-Objekten.
    - **Library Cache:** Er ist eine Speicherstruktur, die den ausführbaren SQL- und PL/SQL-Code speichert.
    - **Server Result Cache:** Dieser Cache hält die Ergebnismenge von SQL-Abfragen und PL/SQL-Funktionen bereit.
    - **Reserved Pool:** Ein Speicherbereich, der dazu genutzt werden kann, um große aneinander liegende Chunks einem Speicher zuzuweisen.

[27] *Vgl. [ORA_LA_15] Kap14, S. 7.*
[28] *Vgl. [B_SR_03] Kap. 5, S. 167.*
[29] *Vgl. [B_MA_15] Kap. 2.7, S. 22.*
[30] *Vgl. [B_MA_15] Kap. 2.16, S. 29.*

- **Redo Log Buffer:** Ein Kreisförmiger Speicher[31], der Informationen über abgeschlossene Transaktionen bzw. Änderungen in der Datenbank, die noch nicht in die Online-REDO-LOG-Dateien übertragen wurden, bereithält.[32]
- **JAVA Pool:** Ein optionaler Speicherbereich, der zur Speicherung von session-spezifischem Java-Code verwendet wird.
- **Streams Pool:** Speicherpool für gepufferte Query-Nachrichten und bietet Speicher für Oracle-Streams-Prozesse.[33]
- **Large Pool**: Separater, optionaler Speicherbereich, der dafür geeignet ist, Speicher zuzuweisen, welcher größer als der Shared Pool ist.[34]
- **Fixed SGA:** Interne Verwaltungsarea, die z.B. allgemeine Informationen über den Status der Datenbank und über Prozesse enthält.[35]
- **In-Memory Column Store:** Optionaler Bereich, der Kopien von Tabellen, Partitionen und anderen Datenbankobjekten in einem Spaltenformat speichert, welches für schnelles Scannen optimiert ist.

Der Speicher für den *Buffer Cache*, *Shared Pool*, *Large Pool* sowie *Java Pool* wird über die „granules"-Einheit zugewiesen. Falls die SGA einen Wert kleiner als 1 GB annimmt, ist die „granules"- Einheit 4 MB groß. Wenn die *SGA* eine Größe von mehr als 1 GB aufweist, wird die „granules"- Einheit auf 16 MB erweitert.[36] Die Größe der „granules"- Einheit wird beim Start der Datenbank berechnet sowie festgelegt und kann während einer aktuellen Instanz nicht mehr geschlossen werden. Die Größe der *SGA* wird durch die Initialisierungsparameter kontrolliert. Diese Parameter befinden sich in der *INIT.ORA*-Datei oder in einer *SPFILE*-Datei[37] und heißen *SGA_TARGET* und *SGA_MAX_SIZE*.

Eigenschaft	Beschreibung
Parametertyp	Big Integer
Syntax	***SGA_TARGET* = integer [K|M|G]**
Standardwert	0 (SGA Autotuning ist deaktiviert).
Modifizierbarkeit	***ALTER SYSTEM***
Wertebereich	Von 64 bis zum betriebssystemabhängigen Maximum.
Grundinitialisierungsparameter	Ja

Tabelle 3-1: Eigenschaften von SGA_TARGET[38]

[31] *Vgl. Abbildung 3 3: Darstellung von PGA und SGA.*
[32] *Vgl. [B_MV_03] Kap. 3.2, S. 73/74.*
[33] *Vgl. [B_TK_14] Kap. 4, S.165.*
[34] *Vgl. [ORA_LA_15] Kap. 14, S. 33.*
[35] *Vgl. [ORA_LA_15] Kap. 14, S. 34/35.*
[36] *Vgl. [B_TK_10] Kap. 4, S.151. & [B_BT_11] Kap. 8, S. 407.*
[37] *Vgl. [B_AH_04] Kap. A.1, S. 498.*
[38] *Vgl. [ORA_BR_16] Kap. 1, S. 240*

Der Parameter *SGA_TARGET* spezifiziert die totale Größe von allen *SGA*-Komponenten und passt die Speicherpools automatisch an, falls der *SGA_TARGET*-Wert festgelegt wurde. Zusätzlich wird, wenn der *SGA_TARGET*-WERT nicht Null ist, das *automatic SGA tuning*, welches in der Oracle-Version 10g eingeführt wurde[39], aktiviert. Sofern die automatisch angepassten Speicherpools auf ungleich Null gesetzt sind, werden diese Werte als Minimumwerte vom *Automatic Shared Memory Management* verwendet.[40] Diesen Speicherpools wird dann Speicher, so wie er benötigt wird, zugewiesen.[41] Als dynamischer Parameter kann *SGA_TARGET* solange erhöht werden, bis die Größe des Parameters *SGA_MAX_TARGET* erreicht ist.[42]

Eigenschaft	Beschreibung
Parametertyp	Big Integer
Syntax	**SGA_MAX_SIZE = *integer* [K | M | G]**
Standardwert	Anfangsgröße der SGA beim Hochfahren der Datenbank. Abhängig von den verschiedenen Größen der Speicherpools innerhalb der SGA.
Modifizierbarkeit	Nein
Wertebereich	Von 0 bis zum systemabhängigen Maximum.

Tabelle 3-2: Eigenschaften von SGA_MAX_SIZE

Der Parameter SGA_MAX_SIZE spezifiziert die maximale Größe der SGA für die Dauer der Instanz.[43] Die Besonderheit bei 64-Bit-Plattformen bzw. Nicht-32-Bit-Plattformen besteht darin, dass, wenn einer der beiden Parameter MEMORY_TARGET oder MEMORY_MAX_TARGET festgelegt wurde, der Standardwert von SGA_MAX_SIZE auf den größeren Wert der beiden dynamisch angepasst wird[44]. Als Folge dessen wird viel mehr Speicher für die SGA-Erweiterung reserviert. Bei 32-Bit-Plattformen ist der Standardwert der Größte der folgenden Werte:

- 60% von MEMORY_TARGET (falls festgelegt)
- 60% von MEMORY_MAX_TARGET (falls festgelegt)
- 25% vom totalen verfügbaren virtuellen Speicher

[39] *Vgl. [B_DB_10] Kap. 11, S. 425.*
[40] *Vgl. [ORA_RU_16] Kap. 6, S. 6.*
[41] *Vgl. [B_TK_14] Kap. 4, S. 169.*
[42] *Vgl. [ORA_RU_16] Kap. 6, S. 10.*
[43] *Vgl. [B_MA_15] Kap. 19.3, S. 388.*
[44] *Vgl. [B_RF_14] S. 3.*

In der nachfolgenden Abbildung 3-3 wird der Zusammenhang der Speicherbereiche im Detail visualisiert. Es ist deutlich, dass der Großteil der Speicherpools in der *SGA* vorhanden ist und das die *PGA* von außen über Prozesse auf die *SGA* zugreift.

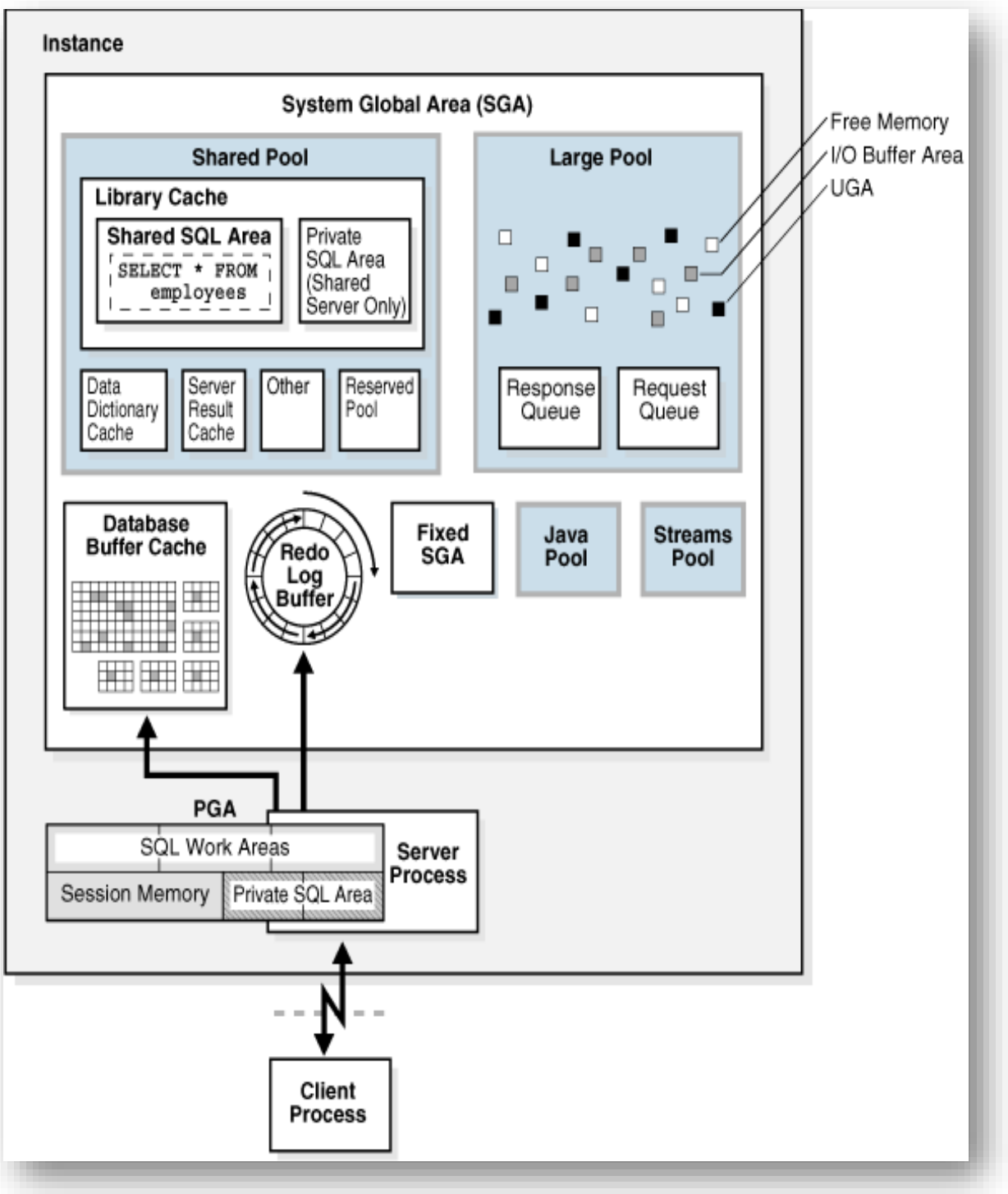

Abbildung 3-3: Darstellung von PGA und SGA[45]

[45] *[ORA_LA_15] Kap. 14, S. 3 Abbildung „Figure 14-1 Oracle Database Memory Structures“.*

4. In-Memory-Konzept

Das *In-Memory*-Konzept basiert auf verschiedenen Mechanismen und Methoden. Als Grundlage für das Konzept wird der *In-Memory-Column-Store* verwendet. Dieser stellt das neue Spaltenformat dar und verändert so die Zugriffsverfahren auf die Objekte. Zum Vergleich werden in diesem Kapitel unter anderem die Unterschiede und Gemeinsamkeiten zwischen dem neuen Spaltenformat und dem alten Zeilenformat erläutert. Danach werden die Anforderungen an die Datenbank, die zur Aktivierung benötigt werden, und den Ablauf der *Population* genauer beschrieben. Die veränderten Zugriffsverfahren verwenden neben dem *In-Memory-Storage-Index* auch die *Single Data Multiple Data*-Methode. Diese Methoden ermöglichen es der Datenbank, die Menge der verarbeiteten Daten zu verringern und die übrig gebliebenen Daten bis zu einem bestimmten Punkt gleichzeitig zu verarbeiten. Weiterhin werden Prozesse, die den *In-Memory-Column-Storage* im Hintergrund unterstützen, auf ihre Funktionsweise durchleuchtet. Als weitere Neuerung können für Objekte, die in den *In-Memory-Column-Storage* geladen werden sollen, eine Priorität und eine Kompressionstechnik ausgewählt werden. Die Auswahlmöglichkeiten unterscheiden sich hinsichtlich der Geschwindigkeit und der Kompressionsrate. Zum Abschluss des Kapitels werden die Vor- und Nachteile der *In-Memory*-Option zusammengefasst.

4.1 In-Memory-Column-Store (IMCS)

Datenbanken, die Gebrauch des *In-Memory*-Formats machen, benutzen das *In-Memory*-Spaltenformat, welches eine neue Komponente der *System Global Area* ist. In der *SGA* ist es unter dem Namen *In-Memory Column Store* als ein optionaler Speicherbereich angelegt.[46] Die *In-Memory-Area* ist ein statischer Pool innerhalb der *SGA*, der vom Parameter *INMEMORY_SIZE* (Standardwert 0)[47] kontrolliert wird.[48] Die aktuelle Größe dieses Bereiches ist in der View *V$SGA* sichtbar. Daten, die in diesem Bereich abgelegt werden, besitzen das nicht traditionelle Zeilenformat. Stattdessen wird ein neues Spaltenformat verwendet, in das Kopien von Tabellen, Partitionen und anderen Datenbankobjekten geladen werden. Der Speicher ersetzt aber nicht den Zwischenspeicher, sondern agiert als Unterstützung dessen.[49] So können die Daten im Row- und/oder Column-Format gespeichert werden

[46] *Siehe Kap. 3.3*
[47] *Vgl. [B_SG_16] Kap. 2, S. 70.*
[48] *Vgl. [B_MB_15] Kap. 2, S. 22.*
[49] *Vgl. [B_MH_16] Kap. 8.6, S. 519.*

Der *In-Memory Column Store* ist weiterhin in zwei Speicherpools unterteilt:

1. **1 MB Pool**: Ein Speicherpool für die aktuell im Column-Format vorhandenen Daten, die in In-Memory geladen werden.
2. **64k Pool**: Speicherort für Metadaten über Objekte, die in *In-Memory* geladen wurden.[50]

Die Menge an vorhandenem Speicher in jedem Pool wird durch folgende Abfrage in Erfahrung gebracht:

```
SELECT pool, alloc_bytes, usede_bytes, populate_status
FROM V$INMEMORY_AREA;
```

SQL-Code 4-1: Menge der Daten im IMCS erfahren[51]

4.2 Unterschied zwischen Row- und Column-Format

Daten werden traditionell im Zeilenformat gespeichert, was unter anderem bedeutet, dass jede neue Transaktion und neuer Record, die in der Datenbank gespeichert werden, wie eine neue Zeile in einer Tabelle zu betrachten sind. Jede Zeile besteht wiederum aus mehreren Spalten, in denen die verschiedenen Attribute dargestellt werden. Ein Zeilenformat eignet sich besonders für *Online-Transaction-Processing* (OLTP)-Systeme, da es einen schnellen Zugriff zu allen Spalten in einem Record ermöglicht. Dies wurde, seitdem alle Daten eines gegebenen Records zusammen im *IN-Memory* und traditionellen Speicher gehalten werden, verbessert.[52]

Ein Speicher im Spaltenformat speichert hingegen jedes Attribut einer Transaktion oder eines Record in einer gesonderten Spaltenstruktur. Das Spaltenformat ist besonders für die Analytik geeignet, da es Daten schnell aus der DB abfragt, wenn nur wenige Spalten ausgewählt sind und eine große Datenmenge angesprochen wird.[53]

Als weiterer Unterschied zwischen den beiden Formaten ist zu nennen, dass DML-Aktionen bezüglich der Effizienz anders verarbeitet werden. In einem Zeilenformat sind DML-Aktionen äußert effizient, weil die Manipulation eines Records durch eine Änderung in der Zeile möglich ist. Dagegen können im Spaltenformat *DML*-Aktionen durch den *In-Memory-Storage-Index* zeilenweise behindert und deshalb äußert ineffizient ausgeführt werden, da für die Bearbeitung eines Records die gesamte Spaltenstruktur der Tabelle verändert werden muss.[54]

[50] *Vgl. [ORA_MC_15] S. 8.*
[51] *Vgl. [B_SG_16] Kap. 2, S. 69.*
[52] *Vgl. [ORA_MC_15] S. 5.*
[53] *Vgl. [B_MV_14] Kap. 9, S. 311.*
[54] *Vgl. [ORA_RU_14] Kap. 6, S. 28.*

Eine wichtige Neuerung der *In-Memory*-Option ist der Aspekt, dass die Daten nicht nur *im In-Memory*-Spaltenformat gespeichert werden können, sondern dass gleichzeitig die Daten auch im *In-Memory*-Zeilenformat (Buffer Cache) geladen werden. Der *In-Memory*-Spaltenspeicher sollte so angepasst sein, dass alle im Speicher anzulegenden Objekte aufgenommen werden können. Mit der Dual-Format-Architektur wird aber nicht die doppelte Menge an Speicher verwendet. Der *Buffer Cache* wurde so optimiert, dass er sehr effektiv mit einer kleineren Größe als der Größe der Datenbank arbeiten kann. In der Praxis wird deshalb erwartet, dass die Dual-Format-Architektur weniger als 20 % Overhead[55] in Bezug auf den Gesamtspeicher erstellt.[56] Mit diesem Ansatz bleibt nur eine einzige Kopie der Tabelle im Speicher, weshalb z.B. keine Kosten für weiteren Speicher oder Synchronisationsprobleme notwendig sind. Die Datenbank beinhaltet weiterhin vollständige Transaktionskonsistenz zwischen dem Zeilen- und dem Spaltenformat sowie die Konsistenz zwischen Tabelle und Index.

Abbildung 4-1: Dual-Format im IMCS[57]

4.3 Aktivierung der In-Memory-Option

Die *In-Memory*-Option benötigt keine weitere Installation, sondern ist eine neue Komponente der *System Global Area* (SGA). Deshalb kann sie auch nicht entfernt werden. Dennoch ist die Option nicht automatisch aktiviert.[58] Falls der *In-Memory*-Speicher aktiviert ist, kann man über den Parameter *INMEMORY* den Status des zugewiesenen Speichers erfahren. Es gibt weiterhin sechs Parameter, die mit dem Präfix *INMEMORY_* beginnen.[59] Diese sind in Kapitel 4.11 im Detail erläutert. Der *In-Memory*-Speicher wird auch nicht vom *Automatic Memory Management* (AMM) kontrolliert oder beeinflusst.[60] Die minimale Größe des *In-Memory*-

[55] *Overheads sind keine primären Daten, sondern speichern nur Information bezüglich der gespeicherten Daten und gelten als Zusatzinformationen (Vgl. [B_AH_04] Kap. 4.4, S. 98).*
[56] *Vgl. [ORA_MC_15] S. 5.*
[57] *[ORA_MC_15] S.5 „Figure 1. Oracle's unique dual-format architecture.".*
[58] *Vgl. [ORA_RU_16] S. 39.*
[59] *Vgl. [B_SG_16] Kap. 2, S. 70*
[60] *Vgl. [ORA_MC_15] S. 6. & Vgl. [B_SG_16] Kap. 2, S. 69.*

Speichers liegt bei 100 MB und wird durch den Parameter INMEMORY_SIZE festgelegt. Folglich ist, wenn der Parameter INMEMORY_SIZE den Wert Null hat, die *In-Memory*-Option deaktiviert, sodass kein Speicher bereitgestellt wird.[61] Ein anderer Weg, um in Erfahrung zu bringen, ob die *In-Memory*-Option aktiv ist und Speicherplatz reserviert wurde, ist die Abfrage über die View *V$SGA*.[62] Wenn festgestellt wurde, dass sie nicht aktiv ist, sind einige Parameter anzupassen, um diese zu aktivieren. Da der *In-Memory*-Spaltenspeicher Teil der *SGA* ist, muss sichergestellt werden, dass der Parameter SGA_TARGET groß genug ist, um den neuen *In-Memory*-Speicher und alle anderen Komponenten unterzubringen. Sobald der Parameter *SGA_TARGET* festgelegt wird, werden weitere Speicherpools automatisch größenmäßig angeordnet. Diese sind der *Database Buffer Cache, Shared Pool, Large Pool, Java Pool,* sowie *Streams Pool.*[63] Sollten für diese Speicherpools keine Null-Werte gesetzt sein, werden die vorhandenen Werte als Minimumwerte vom *Automatic Shared Memory Management* verwendet.[64] Die Größe der Speicherpools *Redo log Buffer*, andere Zwischenspeicher wie *KEEP* oder *RECYCLE* Pool und Fixed SGA werden manuell festgelegt und nicht vom *Automatic Shared Memory Management* beeinflusst. Der reservierte Speicher für diese Pools muss dabei vom komplett verfügbaren Speicher abgezogen werden.[65] Nachdem der Speicher reserviert wurde, werden nun bestimmte Objekte für die In-Memory-Option benötigt.

Im Gegensatz zu anderen Datenbanken werden nicht alle Objekte der Oracle-DB im *IMCS* benötigt. Der *In-Memory*-Speicher sollte nur mit Daten bestückt sein, die sich kritisch auf die Performance auswirken. Daten, die eine geringere Auswirkung auf die Performance haben, können im preisgünstigeren Plattenspeicher liegen. Nur Objekte mit dem *INMEMORY*-Attribut sind auch im *In-Memory*-Spaltenspeicher vorhanden. Dieses Attribut wird durch die DDL-Anweisung ALTER verändert und kann für Tabellen, Tablespaces, Unter-Partitionen und materialisierte Views spezifiziert werden.[66] Es folgt ein Beispiel, in dem für eine Tabelle die *In-Memory*-Option aktiviert wurde:

Beispiel:

```
ALTER TABLE XY INMEMORY;
```

SQL-Code 4-2: Beispiel - In-Memory für Tabelle aktivieren

[61] *Vgl. [ORA_BR_16] Kap.1, S. 114.*
[62] *Vgl. [ORA_RU_16] Kap. 5, S. 19 & Vgl. [B_CG_08] Kap. 1, S.71.*
[63] *Vgl. [ORA_RB_16] Kap. 2, S. 3.*
[64] *Vgl. [B_LF_08] Kap. 3-3, S. 66.*
[65] *Vgl. [ORA_MC_05] Kap. 8, S. 6.*
[66] *Vgl. [ORA_RU_14] Kap. 6, S. 28.*

Als Besonderheit ist hier hervorzuheben, dass Oracle automatisch entscheidet, wann die Tabelle in den *In-Memory*-Speicher geladen wird. Dies geschieht nach dem Grundsatz von *on demand* und wird durch die Prioritäten geregelt. D.h., dass, wenn der Zugriff auf diese Tabelle angefordert wird, diese auch in den In-Memory-Speicher geladen wird. Wenn man eine Tabelle wieder in den normalen Speicher verschieben möchte, verwendet man das Attribut *NO INMEMORY*.

Beispiel:

ALTER TABLE XY NO INMEMORY:

SQL-Code 4-3: Beispiel - In-Memory für Tabelle deaktivieren

Wenn eine Tabelle auf den *In-Memory*-Modus umgestellt wird, wird automatisch für alle Spalten der Tabelle die *In-Memory*-Option standardmäßig aktiviert. Es ist aber auch möglich, diese Funktion für einzelne Spalten zu deaktivieren, selbst wenn diese für die Tabelle aktiv sind. Zusätzlich zu dem Attribut *INMEMORY* werden ein *NO INMEMORY* und der Spaltenname in Klammern dahinter angefügt.

Beispiel:

ALTER TABLE XY INMEMORY NO INMEMORY(AA);

SQL-Code 4-4: Beispiel - In-Memory für nur eine Spalte der Tabelle deaktivieren

4.4 Das Laden der Daten in den IMCS

Das Laden der Objekte in den *In-Memory*-Speicher wird auch als *Population* bezeichnet.[67] Wenn nun aufgrund von Änderungen an den Attributen der Objekte, die Objekte neu geladen werden müssen, wird dies als *Repopulation* bezeichnet. Dies kann auch durch eine zu große Anzahl an IMCUs[68] entstehen.[69] Abhängig von der Größe der Datenbank ist es grundsätzlich möglich, dass alle Tabellen in den *In-Memory*-Spaltenspeicher geladen werden. Sobald die *In-Memory-Area* jedoch voll ist, werden erst wieder Objekte in den *ICMS* geladen, wenn ein anderes Objekt gelöscht wird oder nicht mehr als In-Memory-Objekt markiert ist.[70]

Das INMEMORY-Attribut kann bei einer Tablespace, einer Tabelle, bei Sub-Partitionen und materialisierten Sichten aktiviert werden. Wenn dieses Attribut auf dem Level einer Tablespace aktiv ist, dann wird die *In-Memory*-Option für alle Tabellen und materialisierte Sichten

[67] *Vgl. [B_SG_16] Kap. 2, S. 70 & Vgl. [ORA_RU_16] Kap. 6, S. 38.*
[68] *Vgl. Kap. 4.6.*
[69] *Vgl. [ORA_MC_15] S. 17/18.*
[70] *Vgl. [I_MC_14] Abschnitt. „What happens when the In-Memory column store becomes full?“.*

innerhalb dieser Tablespace standardmäßig aktiviert. Wenn das Attribut bei einer Tabelle aktiviert ist, bedeutet dies, dass alle Spalten in den *In-Memory*-Speicher geladen werden. Generell ist es auch möglich, nur eine Teilmenge von Spalten im *In-Memory*-Spaltenspeicher anzulegen.

Die *Population* des *In-Memory*-Spaltenspeichers erfolgt durch einen Satz von Hintergrundprozessen, die als Arbeitsprozesse (ora_w001_orcl) bezeichnet werden.[71] Jedem Arbeitsprozess wird eine Teilmenge der Datenblöcke des Objektes zugewiesen, welches in den *In-Memory*-Spaltenspeicher geladen werden soll. Die Befüllung des *In-Memory*-Spaltenspeichers ist ein Streaming-Mechanismus, der simultan spalten-erstellend und daten-verdichtend ist.[72]

Die Kompression, die während der Befüllung des Speichers verwendet wird, unterscheidet sich von allen Kompressionstypen, die in vorherigen Oracle-Versionen verwendet wurden. Der neue Algorithmus hilft nicht nur dabei Speicherplatz zu sparen, sondern verbessert auch die Leistung bei Abfragen dadurch, dass direkte Abfragen auf komprimierte Spalten möglich sind. Folglich werden alle suchenden und filternden Aktionen für eine geringere Menge von Daten ausgeführt. Eben diese Daten werden nur dekomprimiert, wenn eine Ergebnismenge benötigt wird.[73] Die *In-Memory*-Kompression wird durch den Parameter *MEMCOMPRESS*, welcher eine Unterklausel des Parameters *INMEMORY* ist, festgelegt. Es gibt sechs mögliche Kompressionsverfahren, die sich in ihrer Kompressionsrate und Leistung unterscheiden.[74] Die Kompression jeder Tabelle steht in der Spalte *INMEMORY_COMPRESSION* in der Tabelle **_Tables*.[75]

Objekte, die den *IMCS* befüllen sollen, werden entweder der Priorität nach oder direkt nachdem die Datenbank geöffnet wurde, in den Speicher geladen. Außerdem gibt es die Möglichkeit, Objekte erst in den Speicher zu laden, wenn das erste Mal eine Abfrage auf das Objekt ausgeführt wurde. Durch das Schlüsselwort *PRIORITY*, welches eine Unterklausel des *INMEMORY*-Attributs ist, werden die Prioritäten für einzelne Objekte festgelegt.[76]

Beispiel:

```
ALTER TABLE XY INMEMORY PRIORITY LOW;
```

SQL-Code 4-5: Beispiel - In-Memory für Tabelle mit der Priorität Low aktivieren

[71] *Vgl. [I_AR_15] & Vgl. Kap. 4.5.*
[72] *Vgl. [ORA_MC_15] S. 7.*
[73] *Vgl. [B_SG_16] Kap. 2, S. 70/71.*
[74] *Vgl. [B_AG_15] S.702 & Vgl. [ORA_RU_14] Kap. 6, S. 37.*
[75] *Vgl. Kap. 4.6.*
[76] *Vgl. [ORA_RU_14] Kap. 6, S. 38.*

Ist dem Parameter *PRIORITY* der Standardwert *NONE* zugewiesen, wird der Speicher erst befüllt, wenn der erste Zugriff auf das Objekt ausgeführt wurde. Die Priorität einer Tabelle kann über die Spalte I*NMEMORY_PRIORITY* in der Dictionary-Tabelle **_TABLES* eingesehen werden.[77]

4.5 In-Memory Storage Index

In diesem Kapitel wird der Speicherindex in Bezug auf den *In-Memory*-Bereich beleuchtet, der eine deutliche Einsparung der Datenmenge bietet. Der *In-Memory*-Speicherindex wird automatisch erstellt und für jede Spalte des *In-Memory*-Speichers beibehalten.[78] In einem Speicherindex sind Informationen über die angelegten Regionen im physischen Speicher, in dem sich Daten befinden, vorhanden. Anhand dieser Informationen kann die Speicherzelle feststellen, welche Bereiche des Speichers für eine Abfrage nicht relevant sind und deshalb auch keinen Zugriff benötigen.[79]

Mittels des Speicherindex können Daten auf Grundlage von Filterprädikaten, die in SQL-Ausdrücken enthalten sind, angepasst werden. Um den Überblick zu behalten, wird der minimale und maximale Wert jeder Spalte in einer *In-Memory Compression Unit (IMCU)* gesichert. Nachdem alle *IMCUs* angelegt sind, können bei einer Abfrage die Bedingungen effektiver geprüft werde. Mittels dem *In-Memory*-Spaltenindex werden die Verweise der Spalten auf Einträge, die mit dem angegebenen Spaltenwert in einer der *IMCU's* übereinstimmen, untersucht.[80] Dies geschieht durch das Vergleichen des angegebenen Werts mit den minimalen und maximalen Werten des Spaltenindex.[81] Für den Fall, dass der gesuchte Spaltenwert außerhalb dieses Intervalls liegt, wird die entsprechende *IMCU* nicht weiter untersucht.

Nachdem die Daten optimiert wurden, kann die *SIMD*-Methode[82] dafür sorgen, dass die übrig gebliebenen Daten entsprechend schnell verarbeitet werden.

4.6 Single Instruction Multiple Data (SIMD)

Single Instruction Multiple Data (SIMD) gehört zur flynnschen Klassifikation, welche Rechnerarchitekturen anhand der verfügbaren Befehle und Datenströme unterteilt.[83] *SIMD* dient zur schnellen Verarbeitung von gleichen Operationen auf gleichzeitig mehreren

[77] *Vgl. Kap. 4.8 & Vgl. [ORA_RU_14] Kap. 6, S. 38.*
[78] *Vgl. [I_MCO_14].*
[79] *Vgl. [B_SG_16] Kap. 2, S. 72.*
[80] *Vgl. [I_AR_16].*
[81] *Vgl. Vgl. [ORA_MC_15] S. 11.*
[82] *Vgl. Kap. 4.6.*
[83] *Vgl. [B_SK_13] Kap. 2.1, S. 12 & [ORA_MC_15] S. 11.*

Datenströmen. In einer Oracle-Datenbank wird die Verarbeitung durch das Spaltenformat verwendet und unterstützt die Verarbeitung von Datenwerten durch eine CPU-Anweisung.[84] *SIMD* basiert auf spezifischen Vektor-Registern, spezifischen CPU-Anweisungen und einem bestimmten Satz von Anweisungen für die Ausführung. Ein *SIMD*-Register entspricht einem Register in der CPU, welches jedoch breiter als die traditionellen Register ist[85]. Der Ablauf einer Aktion lässt sich am besten durch den Unterschied zwischen der Skalaren Methode und der *SIMD*-Verarbeitung erläutern. Eine Skalar-Operation, in der zu einem Array mit bestimmten Werten immer der Wert 1 dazu addiert werden soll, arbeitet so, dass jede Aktion ein Register benutzt. In einem Register wird demnach die Zahl geladen, in einem weiteren steht der Wert, der dazu addiert werden soll, und in einem weiteren steht das Ergebnis. Das Ergebnis wird zunächst in den RAM als Ausgabewert geladen, bevor die Register mit den nächsten Werten befüllt werden. In einer *SIMD*-Operation wird ein *SIMD*-Register direkt mit den Werten aus dem Array gefüllt. Dies bedeutet, dass alle Aktionen für die Werte gleichzeitig erfolgen. So werden alle gleichzeitig geladen, dann wird auf alle vier gleichzeitig der Wert 1 addiert und zum Schluss werden alle Ergebnisse gleichzeitig gespeichert. Folglich ist hier der Ablauf deutlich schneller, da die Berechnung direkt abgeschlossen ist und nicht wie bei der Skalar-Operation mehrere Durchgänge benötigt.

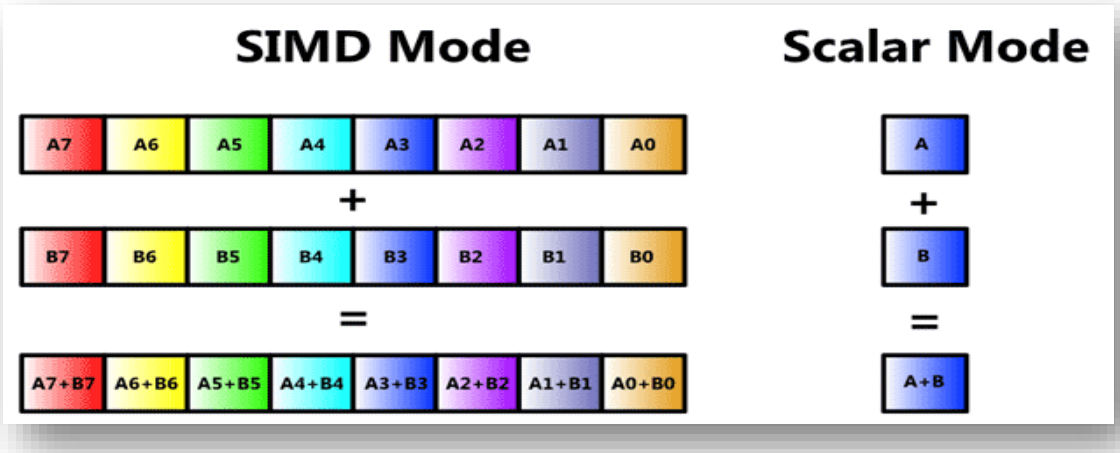

Abbildung 4-2: Unterschied Skalar und SIMD[86]

Innerhalb einer Oracle-Datenbank werden SIMD-Anweisungen durch die Datenstruktur *In-Memory Compression Unit* (IMCU) verwaltet. Die IMCU ist die Einheit für die Zuweisung des Spaltenspeichers und gehört deshalb zum 1 MB-Pool[87]. Eine IMCU kann mehr als eine Spalte

[84] *Vgl. [B_SG_16] Kap. 2, S. 72.*
[85] *Vgl. [B_NF_09] Kap. 5.1.3, S. 62.*
[86] *[I_CL_11] „Figure 3. SIMD versus scalar operations“*
[87] *Vgl. Kap. 4.1.*

beinhalten, wobei jede Spalte als eine sogenannte *Column Unit (CU)* zu sehen ist.[88] Die Metadaten über die *IMCU* werden im 64k-Pool[89] *Snapshot Metadata Unit (SMU)* aufbewahrt. Der *In-Memory*-Spaltenspeicher speichert Indizes, wo von jeder *CU* der minimale und der maximale Wert in einem Speicherindex aufbewahrt wird.[90] Ein Speicherindex macht das sogenannte *CU-pruning* möglich, welches die *CU's* vereinfacht bzw. optimiert.[91] Genaue Informationen über die *CU's* sind in den *Dynamic Performance*-Views *GV$IM_COL_CU und V$IM_HEADER* enthalten. Die SIMD-Erweiterung erfolgt, nachdem die *CU* optimiert wurde und wendet dann effizient Filterprädikate an. Oracle 12c verwendet spezielle Bibliotheken für SIMD-Erweiterungen und die Kompression der Daten.

Die Datenbank verwendet für Daten, die im *In-Memory-Column-Store* analysiert werden, das *SIMD vector processing*. Vektorprozesse zeichnen sich dadurch aus, dass sie eine Kalkulation über mehrere Daten gleichzeitig auszuführen vermögen. Somit können im In-Memory-Spaltenspeicher eine Menge von Spaltenwerten zusammen durch eine CPU-Anweisung ausgewertet werden. Besonders das Spaltenformat wurde dafür ausgelegt, die Anzahl an Spalteneinträgen, die in die Vektorenregister der CPU geladen und durch eine CPU-Anweisung analysiert werden, zu maximieren. Dadurch einsteht die Fähigkeit, dass die Datenbank eine Milliarde Zeilen pro Sekunde zu scannen vermag.[92]

4.7 Hintergrundprozesse

Eine Oracle-DB besitzt viele Prozesse und Hintergrundprozesse, die gewisse Funktionalitäten arrangieren oder eine unterstützende Funktion haben. In den folgenden Kapiteln werden drei Hintergrundprozesse erläutert, die besonders für die *In-Memory*-Option eine wichtige Bedeutung sind.

4.7.1 In-Memory Coordinator

Der Hintergrundprozess *In-Memory Coordinator (IMCO)* initialisiert das Laden von Objekten, bei denen die *In-Memory*-Option aktiviert wurde. Diese können die möglichen Prioritäten *LOW*, *MEDIUM*, *HIGH*, *CRITICAL* aufweisen.[93] *In-Memory*-Objekte, die mit der Priorität NONE markiert werden, werden nicht vorab in den *In-Memory*-Spaltenspeicher geladen,

[88] *Vgl. [ORA_LA_15] Kap. 14, S. 23.& [I_AR_16]*
[89] *Vgl. Kap. 4.1.*
[90] *Vgl. [I_MCO_14].*
[91] *Vgl. [ORA_MC_15] S. 11.*
[92] *Vgl. [I_MCO_14]. & Vgl. [ORA_MC_15] S. 11.*
[93] *Vgl. Vgl. [B_SG_16] Kap. 2, S. 71.*

sondern erst auf Anfrage über Wnnn-Prozesse.[94] Zusätzlich kann der Hintergrundprozess *IMCO* noch dafür sorgen, dass Objekte erneut in den *IMCS* geladen werden.[95]

4.7.2 Space Management Coordinator

Der *Space Management Coordinator (SMCO)*-Prozess ist ein Hintergrundprozess, der für die Verwaltung von verschiedenen Aufgaben der Speicherverwaltung zuständig ist. Er beinhaltet sowohl die proaktive Speicherzuteilung, als auch die Zurückforderung von Speicher.[96] Dafür erstellt der *SMCO* dynamische Hilfsprozesse (*Wnnn*), die dieser dann in Aufgaben einbindet.[97]

4.7.3 Space Management Slave Process

Die sogenannten *Wnnn*-Prozesse führen im Auftrag der Speicherverwaltung und der Oracle *In-Memory*-Option Aufgaben und Aufträge im Hintergrund aus. Sie werden dafür dynamisch vom *SMCO* erstellt. Basierend auf der Wachstumsanalyse der Speicherbenutzung und der Speicherzurückgewinnung durch verworfene Segmente zählt zu den Aufgaben der Wnnn-Prozesse die Reservierung von Speicher für lokal verwaltete Tablespaces und SecureFile-Segmente.[98]

Nachdem die *Wnnn*-Prozesse einmal gestartet wurden, agieren sie als autonome Erfüllungsgehilfe und beginnen nach Abschluss der aktuellen Aufgabe automatisch mit der nächsten Aufgabe aus der Wartschlange. Sobald ein *Wnnn*-Prozess länger nicht aktiv ist, schließt sich dieser selber.

Wnnn-Prozesse führen Aufgaben der *In-Memory*-Option aus, welche dafür sorgen, dass der Speicher mit den Daten aus den *In-Memory*-Objekten befüllt wird. In diesem Zusammenhang nutzt sowohl der *IMCO*-Hintergrundprozess als auch der Vordergrundprozess die *Wnnn*-Prozesse für die Befüllung und Neubefüllung mit Daten. Der Vordergrundprozess befüllt den Speicher, als Reaktion auf eine Abfrage oder eine DML-Transaktion auf In-Memory-Objekte, mit den entsprechenden Daten.[99]

4.8 Prioritäten

Objekte werden im *IMCS* in einer priorisierten Liste entweder direkt oder nachdem sie das erste Mal gescannt wurden, angelegt. Die Reihenfolge, in der die Objekte angelegt werden, wird durch den Parameter PRIOTITY festgelegt.

---

[94] Vgl. Kap. 4.7.3.
[95] Vgl. [ORA_BR_16] Kap. F, S. 9.
[96] Vgl. [ORA_LA_15] Kap. 12, S. 16.
[97] Vgl. [ORA_MC_15] S. 11. & Vgl. [ORA_BR_16] Kap. F, S. 18.
[98] Eine neuere Version der bisher bekannten LOB-Technologie, die eine der besten Möglichkeiten bietet, Daten wie Bilder, Videos, PDFs und ähnliches zu speichern. (Vgl. [ORA_RR_07] S. 2ff).
[99] Vgl. [ORA_BR_16] Kap. F, S. 21.

Dieser Parameter hat folgende fünf Level:

1. **CRITICAL**

   Daten mit dieser Priorität werden zuerst geladen. Danach folgen Daten mit den Prioritäten NONE, LOW, MEDIUM oder HIGH.[100]

2. **HIGH**

   Die Daten, die mit dieser Priorität gekennzeichnet sind, werden erst nach den CRITICAL-Daten geladen. Sie werden jedoch vor den Daten mit der Priorität NONE, LOW oder MEDIUM geladen.[101]

3. **MEDIUM**

   Das Level dieser Priorität steht über den Prioritäten NONE und LOW und kann erst geladen werden, wenn alle Daten mit den Prioritäten CRITICAL und HIGH fertig geladen sind.[102]

4. **LOW**

   Die Priorität LOW ist die schwächste Priorität, bei der die Daten noch direkt in den Speicher geladen werden. Erst nachdem CRITCAL, HIGH und MEDIUM fertig geladen sind, können diese Daten geladen werden.[103]

5. **NONE**

   Im Gegensatz zu den anderen Prioritäten werden die Daten mit der Priorität NONE erst geladen, wenn ein Zugriff auf die Daten erfolgt ist. Erfolgt dieser, wird dieses Objekt ungeachtet der Priorität anderer Objekte in den Speicher geladen.[104]

Objekte, die kleiner als 64 KB sind, werden nicht im Speicher angelegt, da diese eine zu große Menge an Speicher verbrauchen. Dies kommt daher, dass der *ICMS* in 1 MB-Chunks unterteilt ist. [105]

4.9 Kompressionstechnik

Die traditionelle Kompression war ursprünglich ein Mechanismus, der Speicherplatz sparen sollte.[106] Nun werden Daten, die den *IMCS* befüllen, durch eine neue Reihe von Kompressionsalgorithmen komprimiert, welche nicht nur Speicher sparen können, sondern

[100] *Vgl. [I_AR_15].*
[101] *Vgl. Vgl. [B_SG_16] Kap. 2, S. 71.*
[102] *Vgl. [ORA_MC_15] S. 8.*
[103] *Vgl. [ORA_BR_16] Kap. 1, S. 108.*
[104] *Vgl. [ORA_RU_14] Kap. 6, S. 31.*
[105] *Vgl. [ORA_RU_16] Kap. 6, S. 39.*
[106] *Vgl. [B_SG_07] Kap. 3, S. 154.*

auch dafür gedacht sind, die Leistung von Abfragen zu verbessern. Das neue IM-Kompressionsformat erlaubt, direkt Abfragen gegen komprimierte Spalten durchzulaufen. Somit werden alle scannende und filternde Operationen auf eine deutlich kleinere Menge an komprimierten Daten ausgeführt. Diese Daten werden erst dann dekomprimiert, wenn es für die Ergebnismenge benötigt wird.[107]

Die *In-Memory*-Kompression wird über das Schlüsselwort MEMCOMPRESS[108], welches ein Unterpunkt des INMEMORY-Attributs ist, gesteuert. MEMCOMPRESS kann sechs verschiedene Stufen annehmen, wobei jede eine andere Art der Kompression oder Leistung hat[109]:

1. **NO MEMCOMPRESS**
   Die Daten werden ohne Kompression geladen.[110]

2. **MEMCOMPRESS FOR DML**
   Eine Minimale Kompression, die für die Leistung von DML-Operationen optimiert ist und die Daten im *IMCS* am wenigsten komprimiert.[111]

3. **MEMCOMPRESS FOR QUERY LOW**
   Dies ist der Standardwert. Er bietet die beste Leistung für Abfragen und verwendet außerdem allgemeine Kompressionstechniken wie Wörterbuchkompression, Lauflängenkodierung und *Bit-Packing*.[112]

4. **MEMCOMPRESS FOR QUERY HIGH**
   In dieser Stufe werden sowohl die Leistung von Abfragen als auch das Sparen von Speicherplatz optimiert.[113]

Die *FOR CAPACITY*-Optionen wenden eine zusätzliche Kompressionstechnik auf die *FOR QUERY*-Kompression an. Dies hat einen entscheidenden Einfluss auf die Leistung eines jeden

[107] *Vgl. [ORA_MC_15] S. 8.*
[108] *Vgl. [ORA_LA_15] Kap. 14, S. 25.*
[109] *Vgl. [ORA_RU_14] Kap. 6, S. 29/30.*
[110] *Vgl. [B_SG_16] Kap. 2, S. 70.*
[111] *Vgl. [ORA_RB_16] Kap. 12, S. 9.*
[112] *Vgl. [ORA_LA_15] Kap. 14, S. 25 & Vgl. [ORA_MC_15] S.9.*
[113] *Vgl. [ORA_BR_16] Kap. 1, S. 107.*

Eintrags, der dekomprimiert werden muss, bevor die WHERE-Klausel verwendet werden kann.[114]

5. **MEMCOMPRESS FOR CAPACITY LOW**
   Eine ausgeglichene Stufe, wobei die Tendenz stark zum Speicherplatzsparen geht, denn hier wird eine geschützte Kompressionstechnik mit dem Namen OZIP[115] verwendet, die eine extrem schnelle und speziell auf die Oracle-Datenbank angepasste Dekompression anbietet. Es ist der Mittelwert zwischen Kompression und Leistung.[116]

6. **MEMCOMPRESS FOR CAPACITY HIGH**
   Diese Stufe ist optimiert, um Speicherplatz zu sparen.[117] Sie verwendet einen schwereren Gewichtskompressionsalgorithmus mit einer langsameren Dekompression an, um eine höhere Kompression zu liefern.[118]

Die Kompressionsraten können vom Zweifachen bis zum Zwanzigfachen variieren und sind abhängig von der gewählten Option, dem Datentyp und dem Inhalt der Tabelle. Die Kompressionstechnik kann sich auch innerhalb einer Tabelle zwischen den verschiedenen Spalten und Partitionen unterscheiden. Somit ist es denkbar, manche Spalten für Leistung und manche für das Sparen von Speicherplatz zu optimieren.[119] Einen Weg, die Kompression besser kontrollieren zu können, wird über den *Oracle Compression Advisor* ermöglicht. Ebendies ist in der Datenbank als Paket *DBMS_COMPRESSION* vorhanden. Dieser Ratgeber gestattet eine Schätzung über die Kompressionsverhältnisse, die durch die Verwendung von MEMCOMPRESS möglich sind.[120] Genau diese Schätzung basiert auf der Analyse der Daten eines Abschnitts einer Tabelle und liefert eine gute Schätzung der tatsächlichen Ergebnisse, sobald die Tabelle in den IM Spaltenspeicher geladen wird.[121] Der neue MEMCOMPRESS-Algorithmus kann erst ab der Oracle-Datenbank-Version 12.1.0.2 oder höher verwendet werden.[122]

[114] *Vgl. [ORA_MC_15] S. 9.*
[115] *Vgl. [I_OV_16] S. 11.*
[116] *Vgl. [I_MRM_2016] & Vgl. [ORA_BR_16] Kap. 1, S. 107.*
[117] *Vgl. [ORA_BR_16] Kap. 1, S. 107.*
[118] *Vgl. [ORA_MC_15] S. 9.*
[119] *Vgl. [ORA_MC_15] S. 9.*
[120] *Vgl. [I_ORA1] & [I_US_11]*
[121] *Vgl. [B_SG_16] Kap. 2, S. 72.*
[122] *Vgl. [ORA_MC_15] S. 9.*

4.10 Überwachung von In-Memory-Objekten

Die Aktivierung der *In-Memory*-Option kann abgeschlossen sein, aber um zu überprüfen, ob die Objekte tatsächlich in den *In-Memory*-Speicher geladen wurden, wurden neue Views hinzugefügt. Über die *Dynamic Performance Views V$VIEWS*[123], *V$IM_SEGMENTS*[124] und *V$IM_USER_SEGMENTS*[125] kann in Erfahrung gebracht werden, welche Objekte sich im *In-Memory-Column-Store (IMCS)* befinden. Die Views werden wie folgt aufgerufen:

```
SELECT * FROM V$IM_SEGMENTS
SELECT * FROM V$IM_USER_SEGMENTS
SELECT * FROM V$IM_COLUMN_LEVEL
```

SQL-Code 4-6: Drei Views zur Überwachung der In-Memory-Objekte

Mit diesen VIEWS kann man unter anderem herausfinden, wie viele Objekte momentan im *IMCS* angelegt sind. Die View *V$IM_SEGMENTS* stellt alle Segmente, die im *IMCS* vorhanden sind, dar.[126] Im Gegensatz dazu zeigt die VIEW *V$IM_USER_SEGMENTS* nur die *In-Memory*-Segmente des aktuellen Benutzers an.[127] Als weitere Neuerung ermittelt die View *V$IM_COLUMN_LEVEL* den Kompressionstyp für Objekte, die im *IMCS* vorhanden sind.[128]

4.11 Initialisierungsparameter

In diesem Kapitel wird ein Einblick über die verschiedenen Initialisierungsparameter, die einen Bezug zum *IMCS* haben, gegeben. Diese werden bei jedem Start einer Oracle-Instanz gelesen. Diejenigen, die keinen direkten Einfluss auf den *In-Memory*-Bereich haben, sind an dieser Stelle nicht erwähnenswert. Mit folgender Abfrage können die verschiedenen relevanten Parameter angezeigt werden:

```
SHOW PARAMETER INMEMORY;
```

SQL-Code 4-7: Alle In-Memory-Parameter anzeigen

[123] *Vgl. [B_DK_13] Kap. 10, S. 262.*
[124] *Vgl. [ORA_BR_16] Kap. 8, S. 2.*
[125] *Vgl. [ORA_BR_16] Kap. 8, S. 4.*
[126] *Vgl. [ORA_LA_15] Kap. 14, S. 23*
[127] *Vgl. [ORA_BR_16] Kap. 8, S. 4.*
[128] *Vgl. [ORA_RU_14] Kap. 6, S. 36.*

- **INMEMORY_SIZE**

  Dieser Parameter legt die Größe des IMCS in einer Datenbank-Instanz fest. Der Standardwert liegt bei Null, was bedeutet, dass die *In-Memory*-Option in der momentanen Version 12c nicht standardmäßig aktiviert wird. Dies könnte daran liegen, dass die *In-Memory*-Option relativ neu und noch nicht so stark verbreitet ist. Der Parameter muss einen Wert größer als Null annehmen und mindestens 100 MB groß sein.[129] In einer multimandantenfähigen Umgebung ist dieser Parameter als Grundlage für die gesamte *Container Database* (CDB)[130] zu sehen. Es kann aber auch für jede einzelne *pluggable Database* ein Wert gesetzt werden.

- **INMEMORY_FORCE**

  Dieser Parameter ermöglicht es, Tabellen und materialisierte Views im *In-Memory*-Speicher zu verwenden. Außerdem kann damit bei allen Tabellen oder materialisierten Views die *In-Memory*-Option deaktiviert werden. Der Wert „DEFAULT“ entspricht dem Standardwert und erlaubt es, die zwei möglichen Attribute INMEMORY oder NO INMEMORY auf verschiedene Objekte der Datenbank anzuwenden. Mit dem Wert „OFF“ wird festgelegt, dass alle Tabellen und materialisierte Views für den *In-Memory*-Bereich deaktiviert werden.[131]

- **INMEMORY_CLAUSE_DEFAULT**

  Dieser Initialisierungsparameter gestattet das Aufstellen einer Standardregel für den *IMCS* in Bezug auf neue Tabellen und materialisierte Views.[132]

  Für den Fall, dass dieser Parameter nicht festgesetzt ist oder einen Null-Wert aufweist, gibt es keine Standardregel für neue Tabellen und materialisierte Views im *In-Memory*-Bereich. Eine andere Handhabe ist es, den Wert auf *NO INMEMORY* zu setzen, da dies dem Standardwert entspricht. Dieser ist in diesem Fall ein leeres Wort.[133]

  Eine Regel wird durch eine *INMEMORY*-Klausel wirksam und kann mit anderen Klauseln über die Kompression und Priorität verbunden werden. In dieser Situation werden alle neuen Tabellen und materialisierten Views in den *IMCS* geladen, unabhängig davon, ob bei der Entstehung die *INMEMORY*-Klausel auf die Tabellen

---

[129] Vgl. [B_MV_14] Kap. 9, S. 313 & [B_AG_15] S. 702 & Vgl. [B_SG_16] Kap. 2, S. 70 & Vgl. [ORA_LA_15] Kap. 14, S. 21.
[130] Vgl. Kapitel 4.12.
[131] Vgl. [B_SG_16] Kap. 2, S. 70 & [ORA_RU_16] Kap.6, S. 40.
[132] Vgl. [ORA_MB_16] Kap. 16, S. 48.
[133] Vgl. [ORA_RU_14] Kap. 6, S. 32.

angewendet wurde. Zusätzlich können Regeln ohne die *INMEMORY*-Klausel angefertigt werden. Dann wird die Regel auch nur auf die Tabellen und Views angewendet, bei denen die *In-Memory*-Option aktiv ist.[134]

- **INMEMORY_QUERY**

  Dieser Parameter spezifiziert, ob Abfragen auf *In-Memory*-Objekte erlaubt sind. Der Wert *ENABLE*, hier der Standardwert, erlaubt es, Abfragen auf Objekte der Datenbank, welche in den *IMCS* geladen werden, durchzuführen. Um den Zugang zu deaktivieren, muss der Parameter auf den Wert *DISABLE* gesetzt werden. Sobald der Parameter INMEMORY_SIZE gesetzt wurde, wird dieser Wert automatisch festgelegt.[135]

- **INMEMORY_MAX_POPULATE_SERVERS**

  Als Absicherung grenzt dieser Parameter die maximale Anzahl an Hintergrundservern, die dazu verwendet werden, den *In-Memory*-Speicher mit Daten zu befüllen, ein. Somit kann der Rest des Systems nicht überladen werden. Der Standardwert ist entweder die effektive Anzahl an CPU-Threads oder der Wert des *PGA_AGGREGATE_TARGET* Parameters dividiert durch 512 MB.[136]

- **INMEMORY_TRICKLE_REPOPULATE_SERVERS_PERCENT**

  Dieser Initialisierungsparameter bezieht sich auf den Parameter INMEMORY_MAX_POPULATE_SERVERS. Er beschränkt im Gegensatz dazu aber die maximale Anzahl an Hintergrundservern, die dazu verwendet werden, den Speicher nach Änderung der Daten, teilweise wieder neu zu befüllen. Eine Neubefüllung benötigt aber nur einen Teil der Server. Deshalb wird mit diesem Parameter ein kleiner prozentualer Anteil, der von Hintergrundservern genutzt werden kann, festgelegt. Die Skala reicht von Null bis 50 Prozent. Abhängig von dem Wert des Parameters INMEMORY_MAX_POPULATE_SERVERS beschreibt dieser Wert dann die prozentuale Nutzung des Hintergrundservers bzw. die Nutzung der Leistung der CPU.[137]

---

[134] Vgl. [I_BU_15] & Vgl. [ORA_RU_14] Kap. 6, S. 32.
[135] Vgl. [ORA_RU_14] Kap. 6, S. 32. & Vgl. [B_SG_16] Kap. 2, S. 70 & Vgl. [ORA_MC_15] S. 22.
[136] Vgl. [ORA_RU_14] Kap. 6, S. 33 & Vgl. [ORA_BR_16] Kap. 1, S. 111.
[137] Vgl. [I_JD_15] & Vgl. [ORA_RU_14] Kap. 6, S. 33 & Vgl. [ORA_MC_15] S. 18 & Vgl. [ORA_BR_16] Kap. 1, S. 115.

- **OPTIMIZER_INMEMORY_AWARE**

  Mit diesem Initialisierungsparameter vom Typ *Boolean* wird die Erweiterung des Modells *Cost-Based Optimization* entweder aktiviert oder deaktiviert. Diese Erweiterung ist ein allgemeiner Prozess zur Optimierung von Abfragen, der dafür sorgt, dass SQL-Ausdrücke am effizientesten ausgeführt werden. Die *Cost-Based Optimization* optimiert hierfür die Ein- und Ausgabe, die CPU und Netzwerkressourcen, die von der Datenbank benötigt werden. Der Parameter kann dann festlegen, ob Objekte dieser Prozesse auch optimiert oder ignoriert werden sollen.[138]

4.12 Container Database und Pluggable Database

In Zusammenhang mit dem Oracle-*In-Memory*-Konzept gilt es zu erwähnen, dass die Verwendung einer multimandantenfähigen Architektur auch mit dem *IMCS* möglich ist. Die *Container Database (CDB)* ist der Kern der multimandantenfähigen Architektur. In dieser Datenbank befinden sich die Hauptressourcen und die Daten der DB. Mit der *CDB* können eine oder mehrere *Pluggable Databases (PDB)* versorgt werden. Eine *Pluggable Database* kann als ein spezieller Typ eines Containers, welche eine Ansammlung von Dateien und Metadaten ist, definiert werden.[139] Während einzelne Aspekte einer einzigen Datenbank behalten werden, ermöglicht diese Architektur die Teilung der *Shared Global Area* und der *PDB's*. Zusätzlich werden die Hintergrundprozesse einer allgemeinen *CDB* sowie ein einziger IMCS von den *PDB's* geteilt.[140]

In Abbildung 4-3 wird eine *Container Database* dargestellt, in der bis zu drei *PDB's* eingesteckt werden, welche alle einer anderen Aufgabe zugeordnet sind. Die allgemeinen Operationen laufen vollständig über die gemeinsame *CDB*, den gemeinsamen Speicher und die Hintergrundprozesse. Wie in einer normalen Datenbank wird auch hier die *In-Memory*-Option zuerst über den Parameter INMEMORY_SIZE aktiviert. Der Unterschied besteht darin, dass einerseits der Parameter in der *CDB* den gesamten *In-Memory*-Speicher bestimmt in der *PDB* allerdings nur den Anteil, den die *PDB* vom *CDB* mitbenutzen kann. Andererseits ist es auch möglich, dass die Summe des IMCS aller *PDB's* die Größe des *IMCS* der *CDB* überschreitet. Dies dient dazu, verfügbaren Speicher nicht zu verschwenden, falls eine *PDB* heruntergefahren

---

[138] Vgl. [I_DB2_15] & Vgl. [I_MC_16] & Vgl. [I_JD_15] & Vgl. [B_SG_16] Kap. 2, S. 70 & Vgl. [ORA_RU_14] Kap. 6, S. 32.
[139] Vgl. [B_DK_13] Kap. 23, S. 667.
[140] Vgl. [ORA_MC_15] S.20.

oder aus der *CDB* entfernt wird.[141] Denn eine Konsequenz hiervon kann sein, dass später hinzugefügte *PDB's* keinen *IMCS* mehr bekommen, da diese der zuerst geladenen *PDB* zugewiesen werden.[142]

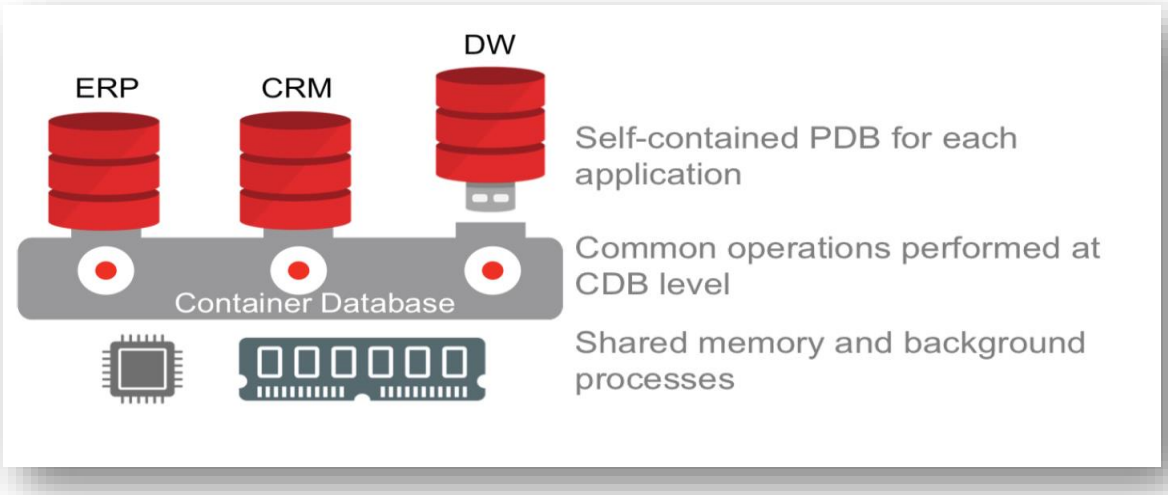

Abbildung 4-3: Beispiel einer multimandantenfähigen Architektur[143]

4.13 Vor- und Nachteile

Das *In-Memory*-Konzept kommt mit vielen neuen Funktionen und Mechanismen, die alle Ihre Vor- und Nachteile haben.

1. Zugriff nur auf benötigte Spaltendaten

Es müssen für die Abfrage nur die wirklich genutzten Spalten untersucht werden. Hingegen werden bei dem traditionellen Zeilenspeicher alle Spalten in jeder Zeile der Tabelle solange analysiert, bis die gesuchten Spalten als erreicht gelten. In der folgenden Tabelle werden Beispiel-Daten dargestellt, mit denen das Prinzip erläutert wird:

Nr.	Unternehmen	Jahr	Aktien(Close)
1	Adidas	2016	112
2	BMW	2015	98
3	XY	2014	45

Tabelle 4-1: Beispiel Tabelle

Zunächst werden die Daten durch die verschiedenen Formate Unterschiedlich von der Datenbank gelesen.[144] Im ROW-Format werden die Daten Zeile für Zeile gelesen. Das bedeutet, dass viel mehr Daten durchsucht werden müssen. Um dies zu verdeutlichen, wird die Beispiel-

[141] *Vgl. [I_MC_16].*
[142] *Vgl. [B_MH_16] Kap. 5.1.2, S. 290.*
[143] *[ORA_MC_15] S.20, Abbildung „Figure 22.".*
[144] *Vgl. Kap. 4.1.*

Tabelle 4.1 untersucht, wobei die Daten dann in folgender Reihenfolge in den ROW- und Column-Formaten gelesen werden:

Zeilenorientierter Speicher (ROW Store)

1, Adidas, 2016, 112; 2, BMW, 2015, 98; 3, XY, 2014, 45

Spaltenorientierter Speicher (In-Memory Column Store)

1, 2, 3; Adidas, BMW, XY; 2016,2015,2014; 112, 98, 45

Folglich kann das Column-Format effizientere Ergebnisse für eine Abfrage liefern, wenn Daten aus einer bestimmten Spalte gesucht werden. Dennoch ist es weniger effizient bei DML-Transaktionen, da diese oft eine ganze Zeile verändern. Als Lösung wird aber die Dual-Format-Architektur geboten. In dieser existieren beide gleichzeitig und die Datenbank entscheidet selbst, welches Format genutzt werden soll.

2. Komprimiertes Format der analysierten und gefilterten Daten

Daten, die in den *IMCS* geladen werden, werden automatisch mit einem neuen Kompressionsalgorithmus komprimiert. Dies erlaubt es, die Eigenschaft der WHERE-Klausel auf das komprimierte Format anzuwenden. Somit ist das Volumen der Daten, die im Spaltenspeicher bei einer Anfrage analysiert werden, um einiges kleiner als das der Daten im Zeilenspeicher. Die nicht komprimierten Daten aus dem Zeilenspeicher können bis zu zwanzigmal größer sein, als die komprimierten Daten im Spaltenspeicher. Die Kompressionsraten des *IMCS* lassen sich so anpassen, dass sich ein Mittelmaß für Geschwindigkeit und Kompression anbietet.

3. Schnellere Verarbeitung durch das SIMD-Konzept

Die Datenbank verwendet SIMD-Vektoren-Instruktionen, um eine Reihe von Spaltenwerten in einem CPU-Uhrenzyklus zu bearbeiten. Somit kann die Datenbank mehrere Werte in einem Vektor speichern, was die Leistungsvorteile beim Verarbeiten mit dem SIMD-Vektor maximiert. Zusätzlich wird vorher die zu bearbeitende Menge an Daten durch den *In-Memory*-Storage-Index verringert. Dieser ist dagegen weniger effektiv, wenn keine Bedingung an die Abfrage gestellt ist.

4. Hohes Risiko bei der Einführung

Die *In-Memory*-Option ist noch nicht so verbreitet wie die ursprünglichen Systeme. Folglich sind Erfahrungen in diesem Bereich nur selten und somit können nicht auf Erfahrungswerte

verwiesen werden. Also können schon kleinere Probleme mit der *In-Memory*-Option einen hohen Aufwand erfordern.

5. Flüchtiger Speicher

Der Hauptspeicher ist ein flüchtiger Speicher und somit können Daten verloren gehen, wenn die Datenbank abstürzt oder aus einem anderen Grund irregulär geschlossen wird. Dafür bietet Oracle zwar ein Backup-Tool, dennoch ist die Gefahr gegeben, dass bei einem Absturz Daten verloren gehen.

5. Technische Analyse

Die nun erläuterten Aspekte der technischen Analyse dienen als Grundlage für die Untersuchung der Datenbasis. Neben den grundlegenden Methoden werden acht Indikatoren dargestellt, die für die Erfassung von Kaufs- und Verkaufssignalen anhand der Beobachtung der Kursentwicklung verwendet werden. Besonders die Beobachtung über einen längeren Zeitraum schafft einen hohen Datendurchsatz, der durch die *In-Memory*-Methode beschleunigt wird.

5.1 Grundlegende Methoden

Im Folgenden werden die grundlegenden mathematischen Methoden aufgeführt, die von den Indikatoren und Oszillatoren genutzt werden. Diese selbst können zwar auch selbst Prognosen erstellen, jedoch verbessert die Weiterentwicklung in andere Indikatoren die Erkennung von Trends.

5.1.1 Gleitender Durchschnitt

Der *gleitende Durchschnitt (GD)* –im englischen *Moving Average (MA)*– gehört zu den am häufigsten verwendeten Methoden der technischen Analyse. Hiermit kann der Durchschnittskurs innerhalb einer Periode dargestellt werden. Da es ein gleitender Durchschnitt ist, fällt der älteste Kurs aus der Berechnung raus, sobald ein neuer Kurs hinzugefügt wird. Generell ergibt sich eine Glättung des gegebenen Kursverlaufes, welche einen Trend andeutet.[145]

Auf Grundlage des gleitenden Durchschnitts wurden einige Erweiterungen entwickelt. Diese sind *simple*, *weighted*, *exponential*[146], *triangular*[147], *variable*[148], die jeweils einen anderen Fokus auf die Daten haben. Der vom gleitenden Durchschnitt angedeutete Trend kann sich, abhängig von der Länge der gewählten Periode, unterschiedlich verhalten[149]. So lässt sich eine kurvige Linie[150] erstellen, welche als Kauf- oder Verkaufssignal zu interpretieren ist, wenn sich diese mit der Linie des Grundwertes oder eines anderen *Moving Average* kreuzt.[151]
Besonders wichtig für später folgende Indikatoren ist der *exponentiell gleitende Durchschnitt* (EMA). Dieser verwendet einen Glättungsfaktor, der sich aus der ausgewählten Zeitspanne errechnen lässt. Die Formel dafür lautet:

[145] *Vgl. [B_DS_13] S. 119.*
[146] *Vgl. [B_ JM_06] S. 204/205.*
[147] *Vgl. [B_ET_09] S. 38, Kap. 3.1.*
[148] *Vgl. [B_CK_10] S. 285. Kap. 14.*
[149] *Vgl. [B_ JM_06] S. 204.*
[150] *Vgl. [B_MW_07] S. 144, Kap. 8.4.*
[151] *Vgl. [B_CA_06] S. 122 & [B_ JM_06] S. 206.*

Glättungsfaktor : $\frac{2}{(x+1)}$

x: = Anzahl Tage

Formel 5-1: Glättungsfaktor[152]

Der gesamte exponentielle gleitende Durchschnitt wird anhand folgender Formel errechnet:

$$EMA(t) = EMA(t-1) + (GF * (SK(t) - EMA(t-1))$$

EMA: = exponential moving average = exponentielle gleitender Durchschnitt,

t: = Ausgewähltes Datum

GF: = Glättungsfaktor

SK: = Schlusskurs vom Datum **t**

Formel 5-2: exponentieller gleitender Durchschnitt[153]

5.1.2 Standardabweichung

„Die Standardabweichung ist in der Stochastik ein Maß für die Streuung der Werte einer Zufallsvariable um ihren Mittelwert.“[154] Sie berechnet sich aus der Quadratwurzel der Varianz und erzeugt ein Intervall, in welchem normalverteilte Werte zu 67 % liegen.[155] Je mehr die Werte von ihrem Mittelwert abweichen, desto größer wird die Standardabweichung.[156]

5.2 Technische Indikatoren und Oszillatoren

Es gibt acht bekannte Indikatoren und Oszillatoren, die einen statistischen Trend aus dem Verlauf einer Aktie ermitteln können. Auch wenn der Fokus dieser Arbeit auf dem *In-Memory*-Konzept liegt, müssen hier auch die Vorgehensweisen für die verschiedenen Indikatoren thematisiert werden, da diese die Grundlage für die Untersuchung der Datensätze bilden. Aus diesen Ergebnissen lassen sich Rückschlüsse auf die Vorteile der *In-Memory*-Option ziehen, wenn die verschiedenen Speicherformate getestet werden.

5.2.1 Moving Average Convergence/Divergence

Der *Moving Average Convergence Divergence* – Indikator (MACD) wurde von G. Appel in den 70er Jahren entwickelt.[157] Dieser Indikator zeigt Auf- und Abwärtstrends anhand der Differenz von zwei exponentiellen gleitenden Durchschnitten an. Diese sind mit einer kurzen

[152] *Vgl. [B_TL_13] Kap. 4.2.1.3, S. 29.*
[153] *Vgl. [B_DJ_08] Kap. 8, S. 144.*
[154] *[B_MG_13] S. 40, Kap. 3.1.6.*
[155] *Vgl. [B_AB_08] S. 128, Kap. 5.*
[156] *Vgl. [B_PZ_00] S. 37, Kap. 3.3.1.*
[157] *Vgl. [B_PD_14] Kap. 12, S.195.*

Periode von zwölf Tagen und einer längeren Periode von 26 Tagen.[158] Als Signallinie wird eine Periode von neun Tagen verwendet.

Formel:

MACD= EMA (12) - EMA (26)

Formel 5-3: MACD[159]

Eine steigende MACD-Linie zeigt den Aufwärtstrend einer Aktie an, wenn sie die Signallinie von unten nach oben durchstößt. In anderer Richtung entsteht ein sinkender Abwärtstrend. Je größer der Abstand des MACD von seiner Nulllinie ist, desto kräftiger ist der Trend.[160] Wenn sich der errechnete Trend in eine komplett andere Richtung als der tatsächliche Aktienkurs entwickelt, wird von einer *MACD*-Divergenz gesprochen. Die negative Divergenz bezeichnet den Kurs, der unerwartet steigt und die MACD-Linie, die währenddessen einen Abwärtstrend andeutet.[161] Dies kann eine Änderung der Trendrichtung andeuten.[162]

Wie andere Marktinstrumente liefert auch der *MACD* manchmal Fehlsignale und sollte daher mit Vorsicht angewandt werden.[163] Dennoch kann der *MACD* abhängig zu der Nulllinie ein Indiz dafür sein, ob es sich um einen Bärenmarkt oder Bullenmarkt handelt[164]. Falls der *MACD* im positiven Bereich weiter steigt oder im negativen Bereich weiter fällt. wird ein beschleunigter Trend angezeigt. Ein fallender *MACD* im positiven Bereich suggeriert einen nachlassenden Aufwärtstrend, der ein Hinweis auf einen bald fallenden Aktienpreis ist.[165]

5.2.2 Momentum und Rate-of-Change

Der *Momentum*-Indikator (*MOM*) soll einen Impuls einer Preis- und Kursänderung anzeigen und Erkenntnisse über die Geschwindigkeit sowie Stärke dieser Kursbewegungen darstellen.[166] Für den Momentum-Indikator wird vom aktuellen Kurs der Kurs vor n-Tagen subtrahiert. Wenn dies für mehrere Tage mit gleicher Anzahl n wiederholt wird, kann aus den errechneten Werten eine Kurve erstellt werden, die die Ausprägungen der Hochs und Tiefs des Aktienkurses andeutet. Wendepunkte in der Kurve sind als Kaufs-/Verkaufssignal zu deuten.[167]

---

[158] *Vgl. [B_RR_06] Kap. 1, S. 294.*
[159] *Vgl. [B_RR_06] Kap.1, S. 307.*
[160] *Vgl. [B_PD_14] Kap. 12, S. 195.*
[161] *Vgl. [B_EP_16] Kap. 18, S. 227.*
[162] *Vgl. [B_RP_15] Kap. 7, S. 119.*
[163] *Vgl. [B_RR_06] Kap. 1, S. 300.*
[164] *„Der Begriff Bullenmarkt steht an der Börse für anhaltend steigende Kurse und Bärenmarkt für anhaltend sinkende Kurse.“ [B_TK_10]*
[165] *Vgl. [B_CH_10] Kap. 1.2.1, S. 41.*
[166] *Vgl. [B_MM_07] Kap. 1.9.9, S. 536.*
[167] *Vgl. [B_TL_13] Kap. 4.2.5, S. 33.*

„Häufig verwendete Periodeneinstellungen für das Momentum sind 5, 10, 21 oder 38 Tage. Es ist aber auch jede andere Einstellung möglich.“[168] Abhängig von der Anzahl der gewählten Tage entsteht eine andere Art der Betrachtung.

- 12-25 Tage = Betrachtung der kurzeitigen Entwicklung eines Aktienkurses[169]
- 20-35 Tage = Betrachtung der langwierigen Entwicklung eines Aktienkurses

Formel:

Momentum(t) = Schlusskurs(t) –Schlusskurs(t-n)

t = der ausgewählte Tag

n= Anzahl der Periodenlänge

Formel 5-4: Momentum

Ähnlich zu dem Momentum-Indikator verhält sich der *Rate-of-Change-Indikator* (*ROC*). Im Gegensatz zum *MOM* berechnet der *ROC* aber die verhältnismäßige Preisdifferenz, die in einer Prozentzahl ausgedrückt wird. Dies ergibt sich aus der Division des aktuellen Kurses mit dem Kurs von vor n-Tagen. Das Ergebnis wird dann mit 100 multipliziert.

Formel:

ROC= (C / Cx) *100

C: = Aktueller Schlusskurs der Aktie y

Cx: = Schlusskurs der Aktie y vor n-Tagen

Formel 5-5: Rate of Change[170]

Da der *ROC* eine prozentuale Darstellung des *MOM*-Indikators ist, zeigt dieser die *gleichen* Trends an. Der ROC kann einen fortgesetzten Aufwärtstrend anzeigen, wenn der ROC im positiven Bereich noch weiter steigt. Fällt der *ROC* dagegen im positive Bereich, wird das Ende eines bisherigen Aufwärtstrends angezeigt. Im negativen Bereich deutet der *ROC* genau das Gegenteil an. Steigt er im negativen Bereich, wird ein Ende eines bisherigen Abwärtstrends angedeutet. Fällt der *ROC* im negativen Bereich, ist es ein fortgesetzter Abwärtstrend.

Das Momentum ist im Normalfall mit der Aussage etwas weiter voraus. Das bedeutet, wenn das Momentum und der *ROC* aus dem negativen in den positiven Bereich steigen, ist dies ein Kaufsignal. Andersrum kann ein Verkaufssignal entstehen.[171]

[168] *[B_GB_05] Kap. 6.2, S. 163.*
[169] *Vgl. [I_CL_16].*
[170] *Vgl. [B_TL_13] Kap. 4.2.5, S. 33.*
[171] *Vgl. [B_UT_06] Kap. 3.2.2, S. 31/32.*

5.2.3 Average True Range

Der *Average True Range* - Indikator ist ein Standardindikator, der ein vielfältiges Einsatzgebiet hat und 1978 von Wilder entwickelt wurde. Der Ursprungsgedanke war die Abbildung der Schwankungsbreite von Rohstoffen- und Terminmärkten.[172] Anstatt jedoch nur auf die täglichen Handlungsgrenzen zwischen den Höchst- und Tiefstwert zu schauen, wurden die Kurslücken (Gaps) mitberücksichtigt. Für die Ermittlung der *Average True Range* werden folgenden drei Spannen berechnet:

Spanne 1: Höchstwert – Tiefstwert

Spanne 2: Heutiges Hoch – Schlusskurs von gestern

Bedingung von Spanne 2: Der heutige Schlusskurs liegt über dem von gestern

Spanne 3: gestriger Schlusskurs – heutiges Tief

Bedingung von Spanne 3: Heutiger Schlusskurs liegt unter dem von gestern

Formel 5-6: Average True Range[173]

Bei dieser Berechnung wird die Spanne mit dem höchsten Wert genommen, der unabhängig vom Vorzeichen zu betrachten ist. Durch diese Spannen wird die True Range dargestellt. Jedoch sind die Bereiche, die damit umfasst werden, zu grob und haben dadurch nur eine geringe Aussagekraft. Als Folge dessen wird die Average True Range verwendet. Hierfür wird ein gleitender Durchschnitt mit einer Periodenlänge von 5-30 Tagen erstellt.[174] Der ATR-Wert zeigt die wahre Handelsspanne und somit also die Volatilität[175]. Folglich steigt die Volatilität, welche eine Maßeinheit für Schwankungen des Preises zu einem bestimmten Basiswertes sind[176], wenn der ATR-Wert steigt.

5.2.4 Aroon

Der *Aroon*-Indikator (*ARO*) soll den Wechsel einer Phase anzeigen und die entsprechende Richtung für einen Trend ermitteln. Die Basis bilden zwei Linien, wovon die erste die Anzahl der Tage seit dem letzten Hoch und die zweite die Anzahl der Tage seit dem letzten Tief darstellen. Folglich heißen die Linien Aroon-UP und –Down.[177] Diese Komponenten können jeweils Werte zwischen Null und 100 annehmen.

[172] *Vgl. [B_TV_10] Kap. 1.2.1, S. 136.*
[173] *Vgl. [B_UG_04] Kap. 10.10.*
[174] *Vgl. [B_RR_06] Kap. 1, S. 94.*
[175] *Vgl. [B_OP_06] Kap. 2.4.1.5.*
[176] *Vgl. [B_AF_14] Kap. 2.1, S. 4.*
[177] *Vgl. [B_BK_00] Kap. 8, S. 8 – 6/ 8 – 7.*

Formel:

Aroon-UP = ((Anzahl Tage – letztes Hoch) / Anzahl Tage) * 100

Aroon-Down = ((Anzahl Tage – letztes Tief) / Anzahl Tage) * 100

Formel 5-7: Aroon[178]

Ein positiver Trend ist zu erkennen, wenn der Aroon-Up über dem Aroon-Down liegt. Dagegen lässt sich auf einen negativen Trend schließen, wenn der Aroon-Down über dem Aroon-Up liegt. Falls der höhere von beiden Werten zwischen 70 und 90 liegt, kann der Trend als solide bezeichnet werden.[179] „Der Aroon-Oszillator ist eine einfache und interessante Weiterentwicklung des Aroon up/down.“[180] In diesem Fall wird vom Aroon-Up Wert der Wert des Aroon-Down subtrahiert. Somit entsteht eine Bandbreite von +100 bis -100.[181]

Formel:

Aroon-Oszillator = Aroon-Up – Aroon-Down

Formel 5-8: Aroon-Oszillator

Dieser Wert zeigt noch deutlicher einen Trend oder einen Trendwechsel an. Sobald der Wert größer als Null ist, erwartet man einen Aufwärtstrend. Bei einem Wert unter Null liegt ein Abwärtstrend vor. Wenn der Wert ungefähr um den Wert Null liegt, dann findet ein Trendwechsel statt.[182]

5.2.5 Bollinger Band

Das *Bollinger Band* soll einen Preistrend der Aktienkurse angeben und die Schwankungsmöglichkeiten bestimmen[183]. Das *Bollinger Band* basiert auf der Standardabweichung[184], welche die Verteilung von Daten um ihren Mittelpunkt herum beschreibt und ein Maß für die Volatilität ist[185]. Neben der Standardabweichung wird noch der gleitende Durchschnitt[186] als Mittellinie verwendet. Aus der Kombination von beidem können Preisbänder oberhalb und unterhalb der Mittellinie konstruiert werden. Dafür wird die zweifache Standardabweichung nach oben und unten zu dem Durchschnitt hinzugefügt.[187].

[178] *Vgl. [I_RB_14].*
[179] *Vgl. [B_GB_05] S. 155.*
[180] *[B_BV_15] S. 33.*
[181] *Vgl. [B_OP_06] Kap. 2.3.2.*
[182] *Vgl. [B_BV_15] S. 34.*
[183] *Vgl. [B_RR_06] Kap. 1, S. 120.*
[184] *Vgl. [B_JM_06] Kap. 9, S.214.*
[185] *Vgl. [B_MB_16] Kap. 5, S. 63.*
[186] *Vgl. Kap. 5.1.*
[187] *Vgl. [B_HA_06] Kap. 3.7, S.332.*

Formel:

Oberes Band: MA + 2 * SW

Unteres Band: MA – 2 * SW

MA = Moving Average = gleitender Durchschnitt, **SW** = Standardabweichung

Formel 5-9: Bollinger Band[188]

Wenn nun der Kurs die Mittellinie kreuzt, ist der Trend als Kaufs- oder Verkaufssignal zu deuten. Dies ist davon abhängig, ob der Kurs vorher im negativen oder im positiven Bereich der Bollinger Bänder war.[189]

5.2.6 Relative Strength Index

Der Indikator *Relative Strength Index* (RSI) von J. Wallace Wilder[190] soll die „innere Stärke"[191] eines einzelnen Aktienkurses messen und wird als „Weiterentwicklung des Momentums"[192] angesehen. Der Indikator beschreibt den Quotienten aus der durchschnittlichen Aufwärtsveränderungen, dividiert durch die durchschnittlichen Abwärtsänderungen eines Aktienwertes.[193] Diese werden meistens in einem Zeitraum von 14 Tagen erfasst und ergeben eine Kurve die zwischen Null und 100 pendeln kann.[194] Dieser Wert zeigt im Idealfall einen Kaufs- oder Verkaufstrend an. Ab einem Wert unter 30 spricht man von einer überverkauften Aktie, was einen kommenden Kauftrend signalisiert. Dagegen zeigt ein Wert über 70 eine überkaufte Aktie an und deutet so einen auf zukünftigen Verkaufstrend hin.[195] Je näher dieser Wert an Null bzw. 100 liegt, desto wahrscheinlicher ist ein baldiger Trendwechsel in Bezug auf den Kauf und Verkauf einer Aktie. Erfahrene Händler ändern diese Grenzen auch oft in 40-80[196] oder 20-60 ab. Je nachdem ob die Kurse über einen längeren Zeitraum fallen oder steigen, wird von einem Bullen- und Bärenmarkt gesprochen.

Formel:

RSI = 100 - (100/ (1 + RS))

RS = Quotient aus dem Durchschnitt der positiven und der negativen Tage

Formel 5-10: Relative Strength Index[197]

[188] *Vgl. [B_MM_07] Kap. 1.9.4, S. 519.*
[189] *Vgl. [B_LK_16] Kap. 8, S. 104/105.*
[190] *Vgl. [B_TK_09] Kap. 4.2.1, S.55.*
[191] *[B_TK_02] Kap. 3.2.1.4, S. 41.*
[192] *[B_TK_02] Kap. 3.2.1.4, S. 41.*
[193] *Vgl. [B_LK_16] Kap. 8.3, S. 102.*
[194] *Vgl. [B_RR_06] Kap. 1, S. 508.*
[195] *Vgl. [B_JS_05] Kap. 15, S. 576.*
[196] *Vgl. [B_TR_06] Kap. 2.3.2.4.3, S. 70.*
[197] *Vgl. [B_LS_13] Kap. 2.6.1, S. 22.*

Eine oft auch gesehene Darstellungsweise der Formel lautet:

RSI = 100 * (AVG UP) / (AVG UP + AVG DOWN)

AVG UP steht für die durchschnittlichen positiven Tage

AVG DOWN steht für durchschnittlichen negativen Tage

Formel 5-11: Relative Strength Index Variante 2

5.2.7 Williams Percent Range

Die *Williams Percent Range* (W%R) soll überkaufte und überverkaufte Märkte identifizieren.[198] Dafür ist der Schlusskurs in Verhältnis zu dem untersuchten Zeitraum zu setzen. Für diesen Wert wird die Differenz aus dem Höchstwert und dem Schlusskurs von heute gebildet. Das Ergebnis wird wiederrum durch die Differenz, die aus dem Höchstwert und dem Tiefstwert von der gewählten Periode gebildet wird, geteilt.[199]

Formel:

%R = 100 * ((H(t) - C) / (H(t) – T(t)))

H: = Höchstwert in einem Zeitraum t.

C: = Schlusswert vom heutigen Tag.

T: = Tiefstwert in einem Zeitraum t.

Formel 5-12: Williams Percent Range[200]

Der Indikator hat eine Skala von Null bis 100 %, wobei der Bereich von Null bis 20 auf einen überkauften Markt hindeutet. Der Bereich von 80 bis 100 deutet dagegen auf einen überverkauften Markt hin.[201] Als Folge dessen lässt sich frühzeitig durch diese Bereiche auch einen Trendwechsel erkennen.[202]

5.2.8 On Balance Volume

„Der OBV Indikator zeigt das Volumen, also die Menge von verkauften Aktien in der Abhängigkeit vom gestrigen Schlusskurs.“[203] Die Grundlage ist die These, dass das Handelsvolumen größer sein soll, wenn es in die Richtung des Trends geht.[204] Die Berechnung richtet sich nach dem Schlusskurs vom Vortag. Ist dieser positiv zum Tag davor, wird das Volumen vom entsprechenden Tag dazu addiert. Wenn dagegen der Aktienkurs fällt, wird das

[198] *Vgl. [B_MP_05] Kap. 2, S. 47.*
[199] *Vgl. [B_ML_11] Kap. 6, S. 154 und [B_LK_16] Kap. 8, S. 104.*
[200] *Vgl. [B_SH_11] Kap. APPENDIX G.2, S. 665.*
[201] *Vgl. [B_ML_11] Kap. 6, S. 154 und [B_GP_12] Kap. Indikator 16, S. 112.*
[202] *Vgl. [B_CL_12] Kap. 4.7.2.3, S. 65.*
[203] *[B_LK_16] Kap. 8, S. 104.*
[204] *Vgl. [B_ML_11] Kap. 6, S. 136.*

Volumen abgezogen. Wird diese Entwicklung über einen längeren Zeitraum betrachtet, kann durch die Höhe des *OBV*-Indikators ein Trend erkannt werden.[205]

Formel:

Fall 1: Close(t) > Close(t-1) ➔ OBV(t) = OBV(t-1) + Volume(t)

Fall 2: Close(t) < Close(t-1)➔ OBV(t) = OBV(t-1) - Volume(t)

Fall 3: Close(t) = Close(t-1) ➔ OBV(t) = OBV(t-1)

Close(t): = Schlusskurs vom Tag t

OBV(t): = On-Balance-Volumen-Wert vom Tag t

Volume(t): = Das Volumen der Aktien am Tag t

Formel 5-13: On Balance Volume[206]

5.3 Punktesystem für Indikatoren

Ein einziger Indikator gibt einen Hinweis auf einen Trend oder auf die mögliche Entwicklung einer Aktie. Folglich erhöht man die Wahrscheinlichkeit, dass die erstellte Prognose richtig ist, wenn mehrere Indikatoren gleichzeitig betrachtet werden. Dazu werden die bereits vorgestellten acht Indikatoren genutzt. Das *Bollinger Band* und die *Average True Range* bleiben bei diesem Zusammenschluss jedoch vorerst außen vor, da mit diesen Indikatoren die möglichen Grenzen angezeigt werden können, damit der Verlust nicht zu hoch ist. Die übrigen sechs Indikatoren werden interpretiert und erhalten ein Punktesystem. In der folgenden Tabelle wird dieses Punktesystem dargestellt:

Punkte	RSI	MACD	MOM/ROC	ARO	WPR	OBV
2	> 70		> 70		> 70	
1	50 - 70	> 0	50 - 70	> 0	50 - 70	> 0
0	50		50		50	
-1	30 - 50	< 0	30 - 50	< 0	30 - 50	< 0
-2	< 0		< 0		< 0	

Tabelle 5-1: Punktesystem der Indikatoren

Anhand dieser Tabelle lässt sich eine Punktzahl für die mögliche Entwicklung des Trends errechnen. Der maximal zu erreichende Wert liegt bei 10 Punkten. Dies bedeutet nicht, dass eine übermäßig starke Chance besteht, dass die Prognose richtig ist, sondern nur das alle sechs Indikatoren einen positiven Trend andeuten. Folglich werden auch hier nicht die Extremwerte

[205] *Vgl. [B_PT_12] Kap. 4, S. 96.*
[206] *Vgl. [B_RR_06] Kap. 1, S. 403.*

beachtet, da sonst die Aufteilung der Indikatoren in die verschiedenen Bereiche zu komplex ist. Dennoch zeigt dieses Punktesystem an, ob die Indikatoren alle in die gleiche Richtung deuten. Die wirklichen Werte sind zusätzlich noch selbst zu interpretieren, was jedoch ein großes Grundwissen und viel Erfahrung voraussetzt.

Auf Basis dieser Indikatoren und der zugewiesenen Punkte sind nun SQL-Prozeduren zu erstellen.[207] Im folgenden Kapitel werden die einzelnen Schritte, die für die Umstellung auf die *In-Memory*-Option benötigt werden, dargestellt und die Tests, die auf die Datenbasis angewendet werden, vorgestellt.

[207] *Vgl. Anhang S. 75 ff.*

6. Analyse der In-Memory-Option

In diesem Kapitel wird die *In-Memory*-Option hinsichtlich der Geschwindigkeit bei der Verarbeitung von Suchanfragen untersucht. Zuerst werden dafür die genauen Schritte zur Aktivierung in der Beispiel-Datenbank erläutert. Danach werden verschiedene Prozeduren, die einen möglichst hohen Aufwand erzeugen, ausgeführt. Diese Prozeduren erzeugen für einen gewünschten Tag aus der Datenbasis eine Prognose. Die Verarbeitung von mehreren Tagen und auch Jahren zeigen die Verbesserung durch den *IMCS.* Zum Schluss werden dann die Ergebnisse der Untersuchungen dargestellt.

6.1 Aktivierung des In-Memory-Column-Storage

Die *In-Memory*-Option ist standardmäßig nicht aktiviert und folglich ist noch kein Speicher reserviert. Damit der *In-Memory*-Speicher den vorhandenen Speicher nicht überschreitet, muss zuerst kontrolliert werden, wie viel Speicher die Objekte für den *In-Memory*-Speicher beanspruchen. Für die Aktivierung der *In-Memory*-Option sind folgende Schritte auszuführen:

1. Schritt – Größe der Tabelle Aktien

Die tatsächliche Größe der Tabelle Aktien kann über folgende Abfrage in Erfahrung gebracht werden:

```
SELECT bytes FROM user_segments WHERE segment_name = 'AKTIEN';
```

SQL-Code 6-1: Größe der Tabelle abfragen

Ergebnis: Die Tabelle Aktien ist ungefähr 12,5 MB groß.

2. Schritt - Ermittlung des vorhandenen Speichers in der PGA und SGA:

Der *IMCS*-Bereich befindet sich innerhalb der *SGA* und deshalb muss die Größe der *SGA* bestimmt werden. Zusätzlich sollte die *PGA* bestimmt werden, um zu kontrollieren, wie die Größe von beiden zusammen ist. Die Abfrage wird über folgenden Befehl ausgeführt:

```
SHOW PARAMETER SGA; SHOW PARAMETER PGA;
```

SQL-Code 6-2: Die Werte für die Parameter der SGA und PGA bestimmen

Als Ergebnis dieses Befehls werden die einzelnen Werte der entsprechenden Parameter angezeigt. In der folgenden Abbildung ist ein Ausschnitt über diese Werte aus der Beispieldatenbank dargestellt:

```
NAME                           TYPE        VALUE
------------------------------ ----------- -------
lock_sga                       boolean     FALSE
pre_page_sga                   boolean     TRUE
sga_max_size                   big integer 3G
sga_target                     big integer 0
unified_audit_sga_queue_size integer     1048576
NAME                 TYPE        VALUE
-------------------- ----------- -----
pga_aggregate_limit  big integer 0
pga_aggregate_target big integer 0
```

Abbildung 6-1: Werte von SGA und PGA

In dieser Datenbank ist der Parameter *SGA_MAX_SIZE* auf dem Wert „3G“. Folglich ist die maximale Größe für den *SGA*-Speicher 3 Gigabyte groß. Da der *SGA_TARGET*-Parameter den Wert Null hat, ist das „auto-tuning“ vorerst deaktiviert.

Über die folgende Abfrage können Vorschläge für die Größe der SGA aus der VIEW *V$SGA_TARGET_ADVICE* aus der Datenbank gelesen werden:

```
SELECT * FROM V$SGA_TARGET_ADVICE;
```

SQL-Code 6-3: Alle Werte der V$SGA_TARGET_ADVICE anzeigen

	SGA_SIZE	SGA_SIZE_FACTOR	ESTD_DB_TIME	ESTD_DB_TIME_FACTOR	ESTD_PHYSICAL_READS	ESTD_BUFFER_CACHE_SIZE	ESTD_SHARED_POOL_SIZE	CON_ID
1	512	0,25	25	0,463	18307	1408	272	0
2	1024	0,5	54	1	18307	384	336	0
3	1536	0,75	54	1	18307	1024	272	0
4	2048	1	54	1	18307	1408	272	0
5	2560	1,25	54	1	18307	1920	272	0
6	3072	1,5	54	1	18307	2432	272	0
7	3584	1,75	54	1	18307	2944	272	0
8	4096	2	54	1	18307	3328	272	0

Abbildung 6-2: Vorschläge für den Parameter SGA_TARGET

In Abbildung 6-3 sind mehrere Werte für den Parameter *SGA_SIZE* abgebildet. Die zusätzlichen Spalten geben Auskunft über andere Parameter, die durch die Änderung an der *SGA_SIZE* auch verändert werden. Da die Tabelle Aktien 12 MB groß ist, kann der Wert 1024M genommen werden. Der Parameter *SGA_MAX_SIZE* wird auf den Wert 2048M gesetzt, da somit der *SGA_TARGET*-Parameter noch genug Potential für Vergrößerung hat und trotzdem nicht zu langsam wird.

```
ALTER SYSTEM SET SGA_TARGET=1024M SCOPE=SPFILE;
ALTER SYSTEM SET SGA_MAX_SIZE=2048M SCOPE=SPFILE;
```

SQL-Code 6-4: Werte für SGA_TARGET und SGA_MAX_SIZE festlegen

Danach ist zu testen, welche Werte für die *In-Memory*-Parameter festgelegt sind. Über den folgenden Befehl sind die Werte aus der Abbildung 6-4 zu ermitteln:

```
SHOW PARAMETER INMEMORY;
```

SQL-Code 6-5: Alle Werte der In-Memory-Parameter

NAME	TYPE	VALUE
inmemory_clause_default	string	
inmemory_force	string	DEFAULT
inmemory_max_populate_servers	integer	0
inmemory_query	string	ENABLE
inmemory_size	big integer	0
inmemory_trickle_repopulate_servers_percent	integer	1
_optimizer_inmemory_access_path	boolean	TRUE
_optimizer_inmemory_autodop	boolean	TRUE
optimizer_inmemory_aware	boolean	TRUE
_optimizer_inmemory_bloom_filter	boolean	TRUE
_optimizer_inmemory_cluster_aware_dop	boolean	TRUE
_optimizer_inmemory_gen_pushable_preds	boolean	TRUE
_optimizer_inmemory_minmax_pruning	boolean	TRUE
_optimizer_inmemory_table_expansion	boolean	TRUE

Abbildung 6-3: Werte der In-Memory-Parameter

Aus den Werten der Abbildung 6-4 wird es ersichtlich, dass die *In-Memory*-Option deaktiviert ist. Die Option wird aktiviert, sobald der Parameter *INMEMORY_SIZE* größer als Null ist.

3.Schritt - Anpassung der Parameter

Die Datenbasis bietet eine Tabelle, die in der Datenbank ungefähr 12 MB groß ist. Der Mindestwert für den Parameter *INMEMORY_SIZE* liegt bei 100 MB. Folglich ist der Parameter so festgelegt:

```
ALTER SYSTEM SET INMEMORY_SIZE= 112M SCOPE=SPFILE;
```

SQL-Code 6-6. Festlegung des In-Memory-Speichers

Mit dem Attribut *SPFILE* wird festgelegt, dass die Änderungen erst nach einem Neustart der Datenbank übernommen werden. *SPFILE* steht hier für Server-Parameter-File. Wenn der Neustart erfolgt ist, sieht die Änderung wie folgt aus:

NAME	TYPE	VALUE
inmemory_clause_default	string	
inmemory_force	string	DEFAULT
inmemory_max_populate_servers	integer	2
inmemory_query	string	ENABLE
inmemory_size	big integer	112M
inmemory_trickle_repopulate_servers_percent	integer	1
_optimizer_inmemory_access_path	boolean	TRUE
_optimizer_inmemory_autodop	boolean	TRUE
optimizer_inmemory_aware	boolean	TRUE
_optimizer_inmemory_bloom_filter	boolean	TRUE
_optimizer_inmemory_cluster_aware_dop	boolean	TRUE
_optimizer_inmemory_gen_pushable_preds	boolean	TRUE
_optimizer_inmemory_minmax_pruning	boolean	TRUE
_optimizer_inmemory_table_expansion	boolean	TRUE

Abbildung 6-4: Werte der In-Memory-Parameter nach der Speicherreservierung

Nun ist der Speicherplatz für den *In-Memory*-Bereich reserviert und Objekte sind über folgenden Befehl zu markieren:

```
ALTER TABLE AKTIEN INMEMORY;
```

SQL-Code 6-7: In-Memory-Option für die Tabelle Aktien aktivieren

Um zu überprüfen, ob Objekte im *In-Memory*-Speicher vorhanden sind, kann dies mit folgender Abfrage untersucht werden:

```
SELECT POOL, ALLOC_BYTES, USED_BYTES, POPULATE_STATUS
FROM V$INMEMORY_AREA;
```

SQL-Code 6-8: Populationstatus

Oft dauert es ein paar Sekunden, bevor die Daten komplett im In-Memory-Speicher angelegt sind. Erst wenn der *Population-Status* für alle Pools abgeschlossen ist, erscheint das Objekt in der View *V$IM_SEGMENTS*.

6.2 Untersuchung der veränderten Performance

Die Untersuchung der Veränderungen durch die *In-Memory*-Option wird durch Prozeduren durchgeführt. Diese Prozeduren bestehen jeweils aus einem CURSOR, der eine SELECT-Abfrage auf die Aktienwerte verarbeitet. Jede Prozedur wendet innerhalb des CURSORS die

vorherige Prozedur auf einen ausgewählten Zeitpunkt aus. Die Prozedur *Kombination*[208] führt die acht Prozeduren der Indikatoren[209] nacheinander aus. Somit kann eine Prognose für einen Tag erstellt werden. Die Prozedur *JahresProg*[210] filtert den Datensatz nach allen vorhandenen Tagen innerhalb eines Jahres und führt dann die Prozedur Kombination mit jedem vorhandenen Datum des gewählten Jahres aus. Die Prozedur AlleJahre[211] führt die Prozedur JahresProg für alle Unternehmen mit allen vorhandenen Jahren aus. Folglich steigen die Daten, die verarbeitet werden müssen, mit jeder Prozedur an. Somit kann ein Vergleich anhand der unterschiedlichen Menge an Daten erstellt werden. Die folgende Abbildung symbolisiert die steigende Datenmenge durch die verschiedenen Prozeduren:

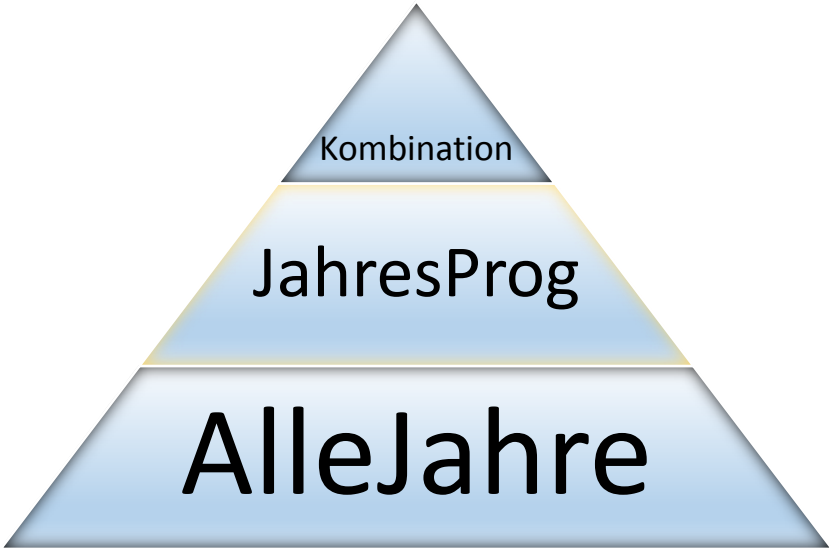

Abbildung 6-5: Aufwand zur Ausführung der Prozeduren

Zu Beginn werden die Ausführungszeiten in Sekunden für die Prozedur Kombination erfasst. Eine zehnmalige Wiederholung der Prozedur mit aktivierter und deaktivierter *In-Memory*-Option schafft die Basis für den Vergleich. Der danach erstellte Durchschnittswert verringert die Schwankungen der Messwerte und verbessert so deren Aussagekraft. Die Durchschnittswerte der beiden Attribute werden subtrahiert und werden in Verhältnis gesetzt.

Prozedur Kombination 27.01.16;ADS.DE **Berechnet acht Indikatoren am 27.01.2016 für das Unternehmen ADS.DE**	
Mit In-Memory	***Ohne In-Memory***
0,015 Sek.	0,032 Sek.
0,016 Sek	0,031 Sek.
0,016 Sek.	0,032 Sek.
0,016 Sek.	0,031 Sek.

[208] *Vgl. Anhang S.83.*
[209] *Vgl. Anhang S.75-82.*
[210] *Vgl. Anhang S.84.*
[211] *Vgl. Anhang S.84.*

0,015 Sek.	0,016 Sek.
0,016 Sek.	0,031 Sek.
< 0,010 Sek.	0,030 Sek.
0,015 Sek.	0,031 Sek.
0,016 Sek.	0,032 Sek.
0,016 Sek.	0,031 Sek.
Durchschnitt = 0,0151	**Durchschnitt = 0,0297**
Mit der aktiven In-Memory Option wird die Prozedur im Durchschnitt 0,0146 Sekunden (197%) schneller ausgeführt.	

Tabelle 6-1: Messwerte für die Prozedur Kombination

Als nächstes wird eine vorgefertigte Prozedur mit einer variablen Anzahl der untersuchten Datenmengen ausgeführt. Dafür wurde die Prozedur *JahresProg* so abgeändert, dass durch die ROWNUM-Funktion die Anzahl der Zeilen eingeschränkt ist. In der folgenden Tabelle werden die Ausführungszeiten der Prozedur *JahresProg* gemessen. Dabei sind die unterschiedlichen Messwerte erfasst, die entstehen, wenn man die Anzahl der Zeilen beschränkt und die *In-Memory*-Option aktiviert oder deaktiviert. Folglich wird innerhalb der Prozedur JahresProg die Prozedur Kombination so oft ausgeführt wie Einträge gefunden worden.

Prozedur JahresProg mit der 10-Einträgen	
Mit In-Memory	***Ohne In-Memory***
0,14 Sek.	0,39 Sek.
0,125 Sek.	0,375 Sek.
0,125 Sek.	0,375 Sek
0,141 Sek.	0,407 Sek.
0,14 Sek.	0,375 Sek.
0,125 Sek.	0,375 Sek.
0,141 Sek.	0,375 Sek.
0,125 Sek.	0,359 Sek.
0,14 Sek.	0,36 Sek.
0,141 Sek.	0,391 Sek.
Durchschnitt = 0,1343 Sek.	**Durchschnitt = 0,3782 Sek.**
Mit der aktiven In-Memory Option wird die Prozedur im Durchschnitt 0,2439 Sekunden (281%) schneller ausgeführt.	

Tabelle 6-2: Messwerte für die Prozedur JahresProg mit 10 Einträgen

Prozedur JahresProg mit der 100 Einträgen	
Mit In-Memory	***Ohne In-Memory***
0,969 Sek.	2,421 Sek.
0,922 Sek.	2,453 Sek.
0,941 Sek.	3,062 Sek.

0,906 Sek.	3,062 Sek.
0,922 Sek.	3,016 Sek.
0,937 Sek.	2,969 Sek.
0,937 Sek.	2,875 Sek.
0,922 Sek.	2,812 Sek.
0,953 Sek.	2,875 Sek.
0,922 Sek.	2,843 Sek.
Durchschnitt = 0,9331 Sek.	**Durchschnitt = 2,8388 Sek.**
Mit der aktiven In-Memory Option wird die Prozedur im Durchschnitt 1,9057 Sekunden (303%) schneller ausgeführt.	

Tabelle 6-3: Messwerte für die Prozedur JahresProg mit 100 Einträgen

In der folgenden Tabelle sind die Ausführungszeiten für die Prozedur JahresProg für das Unternehmen BMW im Jahr 2016 enthalten:

Prozedur JahresProg BMW.DE; 2016 **Berechnet acht Indikatoren für alle in der Tabelle vorhandenen Tage des Jahres 2016 für das Unternehmen BMW.DE**	
Mit In-Memory	***Ohne In-Memory***
0,884 Sek.	2,765 Sek.
0,992 Sek.	2,656 Sek.
0,889 Sek.	2,64 Sek.
0,906 Sek.	2,672 Sek.
0,843 Sek.	2,656 Sek.
0,922 Sek.	2,66 Sek.
0,906 Sek.	2,653 Sek.
0,891 Sek.	2,643 Sek.
0,903 Sek.	2,656 Sek.
0,906 Sek.	2,662 Sek.
Durchschnitt = 0,9042 Sek.	**Durchschnitt = 2,6753 Sek.**
Mit der aktiven In-Memory Option wird die Prozedur im Durchschnitt 1,7711 Sekunden (296%) schneller ausgeführt.	

Tabelle 6-4: Messwerte für die Prozedur JahresProg

In der folgenden Tabelle sind die Messwerte der Prozedur AlleJahre enthalten:

Prozedur AlleJahre **Berechnet acht Indikatoren für alle in der Tabelle vorhandenen Tage für jede Aktien_ID.**[212]

[212] *Alle außer der AKTIEN_ID „GDAXI^“*

Mit In-Memory	***Ohne In-Memory***
1656,96 Sek.	5192,153 Sek.
1854,149 Sek.	6202,459 Sek.
1880,909 Sek.	6286,686 Sek.
1768,818 Sek.	6151,603 Sek.
1874,609 Sek.	4128,25 Sek.
1908,228 Sek.	6205,408 Sek.
Durchschnitt = 1823,9455 Sek.	**Durchschnitt = 5694,4265 Sek.**
Mit der aktiven In-Memory Option wird die Prozedur im Durchschnitt 3870,481 Sekunden (312%) schneller ausgeführt.	

Tabelle 6-5: Messwerte für die Prozedur Rescue

Die Messwerte zeigen eine schnellere Verarbeitung der Prozeduren, wenn die *In-Memory*-Option aktiviert ist. Wenn die Anzahl der Daten jedoch nur im zweistelligen Bereich ist, sind die Unterschiede etwas geringer. Aber dennoch werden die Prozeduren effektiv schneller ausgeführt. Je mehr Daten durch eine Prozedur verarbeitet werden, desto länger ist die Ausführungszeit. Es ist auffällig, dass mit der *In-Memory*-Option die Ausführungszeit dreimal so gering ist. Sogar bei der Prozedur Rescue, wo eine große Datenmenge verarbeitet wird, werden nur knapp 30% der Zeit benötigt.

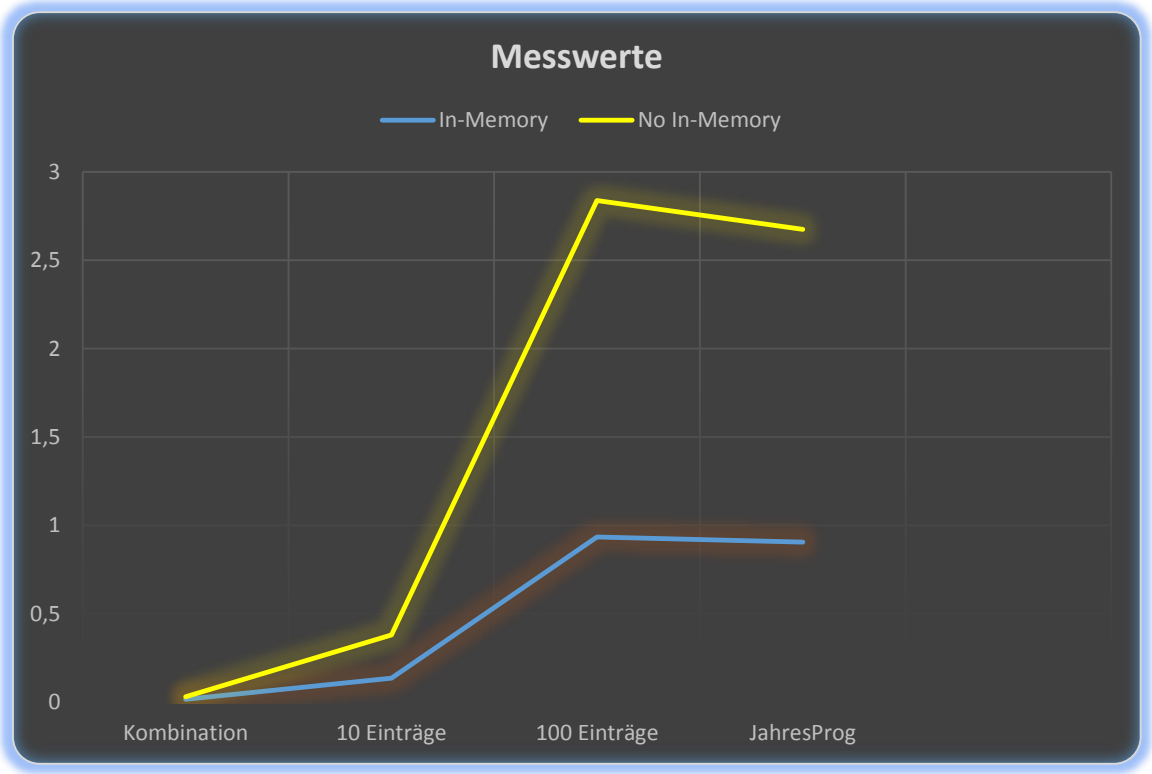

Abbildung 6-6: Messwert der Prozeduren in Sekunden

Der Prozentuale Vergleich in Abbildung 6-7 zeigt, dass alle 5 Tests, außer dem Test der Prozedur Kombination im Bereich von 30% der benötigten Zeit liegen. Als Vergleichswert wurden hier die Ausführungszeiten ohne *In-Memory* mit 100% gesetzt.

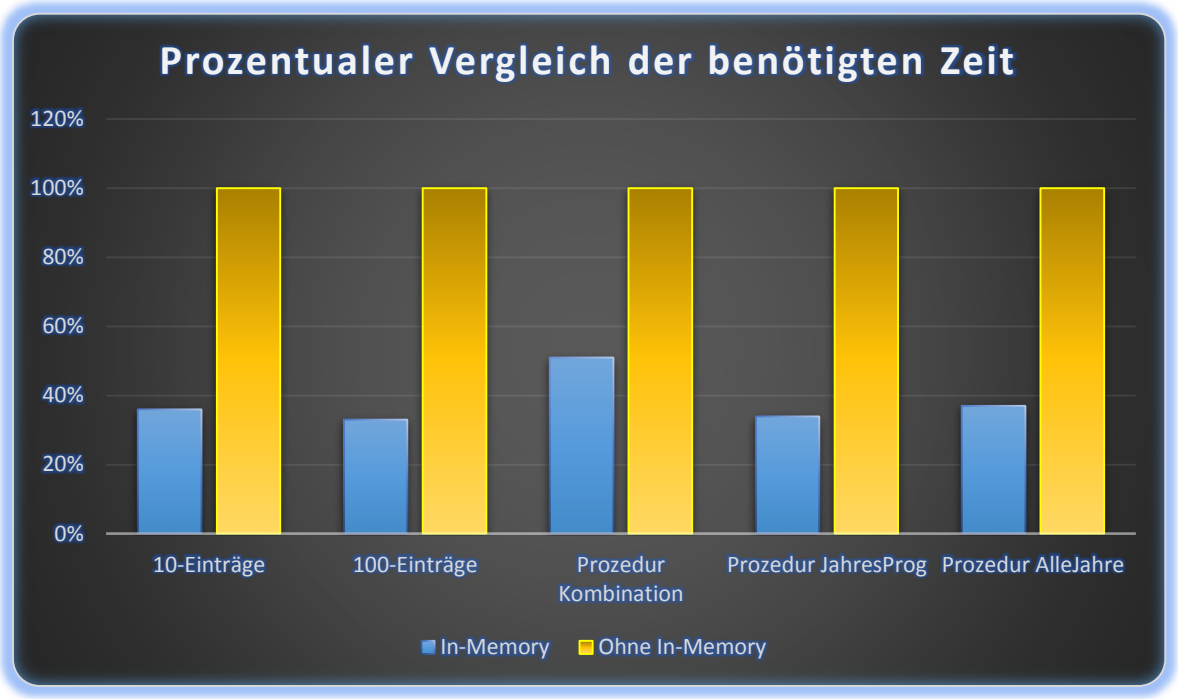

Abbildung 6-7: Prozentualer Vergleich der benötigten Zeit

7. Fazit und Ausblick

Die Oracle-*In-Memory*-Option erscheint auf den ersten Blick leicht zu verstehen zu sein. Es wird der Speicherbereich für die *In-Memory*-Option reserviert und danach werden Objekte mit dem *In-Memory*-Attribut markiert. Aber das *In-Memory*-Konzept ist ein komplexes Konstrukt von dem der Endbenutzer nichts erfährt.

Das Column-Format verändert die Verarbeitung von Suchanfragen auf Tabellen und erschafft so neue Möglichkeiten für die Analyse. Es verhindert das unnötige Lesen von Daten aus nicht-relevanten Spalten und verbessert dadurch die Zugriffszeiten. Neben der Aktivierung der *In-Memory*-Option ist es möglich, die *Population* des *In-Memory*-Column-Storages effektiv zu beeinflussen und wichtige Daten mit einer Priorität zu versehen. Die Priorität bestimmt die Reihenfolge der Daten, die in den Speicher geladen werden, diese ist daher mit Vorsicht festzulegen. Als zusätzliche Verbesserung wird der *In-Memory*-Storage-Index verwendet. Mit diesem Index können Bereiche von Objekten aussortiert werden, um so nur die benötigten Abschnitte lesen zu müssen. Die aussortierten Daten werden dann durch die SIMD-Methode effektiver verarbeitet und verbessern folglich die Zugriffszeiten. Eine weitere Neuerung sind die neuen Kompressionstechniken, die effektiv auf die Daten angepasst werden können. Mit diesen Techniken ist es möglich, das richtige Mittelmaß zwischen Kompressionsrate und Leistung festzulegen. Als negativer Aspekt bleiben der flüchtige Speicher und das Risiko der Einführung und die schlechte Umsetzung von DML-Transaktionen. Denn auch wenn das *In-Memory*-Konzept eine Verbesserung für die Analyse von Daten ist, werden Änderungen an den Daten durch das Spaltenformat schlechter umgesetzt. Für das Problem gibt es aber die Dual-Format-Architektur, die selbst entscheidet, welches Format für eine Aktion verwendet werden soll.

Aus den Messwerten aus Kapitel 6.2 zeigt sich eine deutliche Verbesserung der Zugriffszeit. Die Prozeduren werden im Durchschnitt, wenn die *In-Memory*-Option aktiviert war, dreimal so schnell ausgeführt. Zum Abschluss lässt sich sagen, dass die Vorteile bei dem *In-Memory*-Konzept überwiegen und dieses für eine Performancesteigerung von Analysen und Reportings sorgen kann. Ebenfalls spricht der sinkende Preis von Hauptspeicher für einen Wechsel zum *In-Memory*-Konzept. Gerade im Bereich *Data-Warehouse* sollte es einen großen Vorteil bringen, da die konstant steigende Datenmenge die Datenbank nun weniger ausbremst.

Für den Aktienhandel sollte mit der *In-Memory*-Option eine Echtzeit-Analyse realisiert werden können, da in dieser Arbeit eine große Anzahl an Daten in nur 33% der Zeit analysiert werden konnten. In naher Zukunft wird dieses Konzept voraussichtlich stark verbreitet und noch verbessert werden.

8. Literaturverzeichnis

Legende

„[B_“ : = Bücherquellen

„[ORA_“ : = Offizielle Oracle Dokumente

„[I_“ : = Internetquellen

[B_AB_08]

Achim Bühl (01.06.08) ,

SPSS Version 16: Einführung in die moderne Datenanalyse (Pearson Studium - Scientific Tools),

ISBN-13: 978-3827373328, 896 Seiten.

[B_AF_14]

Andreas Friedrich (06.06.2014),

Volatilität: Chancen und Risiken bei der Investition,

ISBN-13: 978-3842894853, 92 Seiten.

[B_AG_15]

Aleksandra Gruca (17.11.2015),

Man–Machine Interactions 4: 4th International Conference on Man–Machine Interactions,

ISBN-13:978-3319234366, 711 Seiten.

[B_AH_04]

Andrea Held (01.10.04),

Oracle 10g Hochverfügbarkeit. Die ausfallsichere Datenbank mit RAC, Data Guard und Flashback (Edition Oracle),

ISBN-13: 978-3827321633, 648 Seiten

[B_AK_05]

Arun Kumar (01.01.2005),

Easy Oracle Automation: Oracle10g Automatic Storage, Memory and Diagnostic Features (Oracle In-Focus),

ISBN-13: 978-0974599366, 200 Seiten.

[B_BK_00]

Bernhard Keller (01.01.2000),

Aufbau, Struktur und Bewertung von mechanischen Handelssystemen für Aktien- und Futuresmärkte,

ISBN-13: 978-3838670225, 112 Seiten.

[B_BT_11]

Biju Thomas (02.02.2011),

OCA: Oracle Database 11g Administrator Certified Associate Study Guide: Exams1Z0-051 and 1Z0-052,

ISBN: 978-1-118-05942-5, 1152 Seiten.

[B_BV_15]

Bernd Vogel (19.02.2015)

Portfoliostrukturen - Handelsideen für Aktien, Indizes und Fonds,

ISBN-13: 978-3734762864, 120 Seiten.

[B_CA_06]

Christoph Amberger (25.10.2006),

Die größten Geheimnisse des Tradens: Die richtigen Ein- und Ausstiegspunkte für jede Marktlage,

ISBN-13: 978-3898792066, 416 Seiten.

[B_CA_14]

Christian Antognini (02.06.2014),

Troubleshooting Oracle Performance,

ISBN-13: 978-1430257585, 740 Seiten.

[B_CG_08]

Christine Gschoßmann (01.09.2008),

Oracle Backup und Recovery - Das Praxisbuch - Für alle Versionen bis einschließlich 11g (Edition Oracle),

ISBN-13: 978-3827324832, 592 Seiten.

[B_CH_10]

Claus Hilpold, Dieter G. Kaiser (26.03.2010)

Innovative Investmentstrategien: Handelstechniken für eine optimierte Portfoliodiversifikation,

ISBN-13: 978-3834919823, 267 Seiten.

[B_CK_10]

Charles D., II Kirkpatrick, Julie R. Dahlquist (15.11.2010),

Technical Analysis: The Complete Resource for Financial Market Technician,

ISBN-13: 978-0137059447, 671 Seiten.

[B_CL_12]

Clemens Liepert (31.03.2012),

Wegweiser für Investoren zur Analyse des deutschen Aktienmarktes: Führt die Fundamentalanalyse oder die Technische Analyse zur höheren Rendite?,

ISBN-13: 978-3842872066, 164 Seiten.

[B_CR_06]

Claire Rajan (20.05. 2006),

Oracle 10g Database Administrator II: Backup/Recovery & Network Administration,

ISBN-13: 978-1418836641, 627 Seiten.

[B_DB_10]

Donald K Burleson (17.11.2010),

Oracle Tuning (Oracle In-Focus),

ISBN-13: 978-0979795190, 1200 Seiten.

[B_DJ_08]

Derrall Jobmann (01.04.2008),

Die ganze Welt der Technischen Analyse: Erfolgreich mit Indikatoren, Charts & Co.,

ISBN-13: 978-3898794008, 421 Seiten.

[B_DK_13]

Darl Kuhn (16.07.2013),

Pro Oracle Database 12c Administration,

ISBN: 978-1-4302-5728-8, 756 Seiten.

[B_DS_13]

Daniel Schütz (11.09.2013)

Trading für Einsteiger – simplified - Erfolgreich zum ersten Trade,

ISBN 978-3-89879-643-9, 272 Seiten.

[B_EP_16]

Ed Ponsi (27.07.2016),

Technical Analysis and Chart Interpretations: A Comprehensive Guide to Understanding Established Trading Tactics for Ultimate Profit (Wiley Trading Series),

ISBN-13: 978-1119048336, 384 Seiten.

[B_ET_09]

Emilio Tomasini (14.09.2009)

Trading Systems: A New Approach to System Development and Portfolio Optimisation,

ISBN-13: 978-1905641796, 256 Seiten.

[B_GB_05]

Gregor Bauer (01.01.2005),

Praxisratgeber Trading Die Methodik des erfolgreichen Handelns - Schritt für Schritt,

ISBN: 978-3-89879-721-4, 360 Seiten.

[B_GI_02]

Geoff Ingram (Oktober 2002)

High-Performance Oracle: Proven Methods for Achieving Optimum Performance and Availability,

ISBN: 978-0-471-43034-6, 720 Seiten.

[B_GP_12]

Gregor Peters (30.04.2012),

Der Trend ist mein Freund: Handbuch zur Analyse und Strategie,

ISBN-13: 978-3844817058, 140 Seiten.

[B_HA_06]

Heiko Aschoff (31.07.2006)

Die Investmentstrategien der Profis,

ISBN-13: 978-3898792141, 320 Seiten.

[B_JM_06]

John J. Murphy (01.04.2006)

Technische Analyse der Finanzmärkte - Grundlagen, Strategien, Methoden, Anwendungen,

ISBN 978-3-89879-062-8, 650 Seiten.

[B_JS_05]

Jack D. Schwager (30.05.2005),

Technische Analyse: Schwager on Futures,

ISBN-13: 978-3932114038, 864 Seiten.

[B_KL_09]

Kenneth C. Laudon / Jane P. Laudon / Detlef Schoder (01.11.2009),

Wirtschaftsinformatik - Eine Einführung,

ISBN: 978-3-8632-6578-6, 1172 Seiten.

[B_LF_08]

Lutz Fröhlich (08.12.2008),

Oracle 11g: Das umfassende Handbuch,

ISBN-13: 978-3826659133, 784 Seiten.

[B_LK_16]

Ladis Konecny (14.05.2016),

Aktien und Börse: das einzige Buch, das du brauchst,

ISBN-13: 978-3844815481, 328 Seiten.

[B_LS_13]

Leonard Schulz (22.01.2013),

Vergleich von Fundamentalanalyse und technischer Analyse im Rahmen der Aktienkursprognose,

ISBN-13: 9783842845855, 47 Seiten.

[B_MA_02]

Michaela Amtmann (16.05.2002),

Formen der privaten Pensionsvorsorge,

ISBN 978-3-8386-5428-7, 120 Seiten.

[B_MA_13]

Marek Adar (30. 12.2013),

Das große Oracle Datenbank-Einsteigerbuch,

ISBN-13: 978-3732288694, 700 Seiten.

[B_MA_15]

Marek Adar (30.06.2015),

Ein strukturierter Einstieg in die Oracle-Datenbankadministration,

ISBN-13: 978-3842373723, 640 Seiten.

[B_MB_15]

Martin Bach (28.08.2015),

Expert Oracle Exadata,

ISBN-13: 978-1430262411, 672 Seiten.

[B_MB_16]

Martin Bösch (04.07.2016)

Finanzwirtschaft: Investition, Finanzierung, Finanzmärkte und Steuerung,

ISBN-13: 978-3800652501, 570 Seiten.

[B_MG_13]

Mahmoud Gindiyeh (15.12.2013

Anwendung wahrscheinlichkeitstheoretischer Methoden in der linguistischen Informationsverarbeitung,

ISBN-13: 978-3832535735, 290 Seiten.

[B_MH_16]

Mirko Hotzy (11.07.2016),

Der Oracle DBA Handbuch für die Administration der Oracle Datenbank 12c,

ISBN-13: 978-3446443440, 831 Seiten.

[B_ML_11]

Marcus Langanke (16.12.2011),

Powerstart im Devisenhandel mit Metatrader 4: Technische Indikatoren und Handelsansätze,

ISBN-13: 978-3844865998, 280 Seiten.

[B_MM_07]

Martin Michalky (31.08.2007),

Das große Buch der Börse,

ISBN-13: 978-3898792653, 1180 Seiten.

[B_MP_05]

Michael Proffe (31.10.2005),

Die besten Trendfolgestrategien - simplified: Machen Sie den Trend zu Ihrem Freund und schwimmen Sie mit dem Strom,

ISBN-13: 978-3898791649, 102 Seiten.

[B_MV_03]

Murali Vallath (22.10.2003),

Oracle Real Application Clusters,

ISBN-13: 978-1555582883, 804 Seiten.

[B_MV_14]

Murali Vallath (07.10.2014),

Expert Oracle RAC Performance Diagnostics and Tuning,

ISBN-13: 978-1430267096, 712 Seiten.

[B_MW_07]

Wolf-Gert Matthäus (22.12.2007),

Statistik mit Excel - Beschreibende Statistik für jedermann,

EAN 9783835192119, 215 Seiten.

[B_NF_09]

Nicolas Fritz (29.10.2009),

SIMD Code Generation in Data-Parallel Programming,

ISBN-13: 978-3869312408, 176 Seiten.

[B_OP_06]

Oliver Paesler (12.12.2006),

Technische Indikatoren – simplified,

ISBN-13: 978-3898792486, 224 Seiten.

[B_PD_14]

Pierre M. Daeubner (13.06.2014),

Die besten Tradingstrategien - So schlagen Sie konstant den Markt. Inkl. Money-Management und CFD-Trading-Strategien,

ISBN: 978-3-89879-559-3, 288 Seiten.

[B_PT_12]

Patrick René Thom (01.07.2012),

Vorhersage kurzfristiger Aktienkursrenditen: Entwicklung eines maschinellen Lernverfahrens,

ISBN-13: 978-3844101737, 262 Seiten.

[B_PZ_00]

Peter Zoefel, Achim Buehl, Peter Zöfel (01.11.2000),

Statistik verstehen: Ein Begleitbuch zur computerunterstützten Anwendung (Sonstige Bücher AW),

ISBN-13: 978-3827316905, 336 Seiten.-.

[B_RF_14]

Robert G. Freeman **(**21.10.2014),

OCP: Oracle Database 12c Administrator Certified Professional Study Guide: Exam 1Z0-063,

ISBN-13: 978-1118644072, 768 Seiten.

[B_RP_15]

Richard Pfadenhauer (17.04.2015),

Trading mit Hebelprodukten In 5 Schritten zum erfolgreichen Trader,

ISBN: 978-3-89879-861-7, 208 Seiten.

[B_RR_06]

Rene Rose (30.09.06)

Enzyklopädie der Technischen Indikatoren: Trading-chancen profitabel Nutzen,

ISBN-13: 978-3898791045, 768 Seiten.

[B_SD_13]

Sean Dillon (11.11.2013),

Beginning Oracle Programming,

ISBN: 9781430253709, 1104 Seiten.

[B_SG_07]

Sam R Alapati , Charles Kim (14.11.2007),

Oracle Database 11g - New Features for DBAs and Developers,

ISBN13: 978-1-59059-910-5, 602 Seiten.

[B_SG_16]

Saurabh K. Gupta (15.02.2016),

Advanced Oracle PL/SQL Developer's Guide - Second Edition,

ISBN-13: 978-1785284809, 428 Seiten.

[B_SK_13]

Sigrid Körbler (31.10.13),

Parallel Computing - Systemarchitekturen und Methoden der Programmierung,

ISBN-13: 978-3640123629, 80 Seiten.

[B_SR_03]

Sam R Alapati (13-04.2003),

Expert Oracle9i Database Administration,

ISBN13: 978-1-59059-022-5, 1248 Seiten.

[B_SH_11]

Sunny J. Harris, Bill Cruz (01.04.2011),

Tradestation Made Easy!: Using Easylanguage to Build Profits with the World's Most Popular Trading Software,

ISBN:978-0-471-35353-9, 744 Seiten.

[B_TK_02]

Thomas Kreuzhuber (06.03.2002),

Vier wichtige Aktienanalyse-Indikatoren in der Chartanalyse,

ISBN:978-3-8386-5189-7, 108 Seiten.

[B_TK_09]

Tobias Heckmann (01.03.2009),

Markttechnische Handelssysteme, quantitative Kursmuster und saisonale Kursanomalien: Eine empirische Untersuchung am deutschen Aktienmarkt (Finanzierung, Kapitalmarkt und Banken),

ISBN-13: 978-3899367881, 200 Seiten.

[B_TK_10]

Tobias Killer (21.01.2010)

Hallo Herr Killer: Versicherungen sind mehr als nur ein Fuzzi!!,

ISBN-13: 978-3941412149, 184 Seiten.

[B_TK_14]

Thomas Kyte (5.11.2014),

Expert Oracle Database Architecture,

ISBN-13: 978-1430262985, 824 Seiten.

[B_TL_13]

Tristan Leichsenring (16.08.2013),

Technische Indikatoren im Test: Eine Untersuchung am Dax,

ISBN-13: 978-3842884595, 100 Seiten.

[B_TR_06]

Thomas Renner (07.08.2006),

Die Problematik der Aktienauswahl bei privaten Anlegern,

ISBN 978-3-8386-9746-8, 92 Seiten.

[B_TV_10]

Thomas Vittner (20.09.2010),

Die Tradingakademie,

ISBN: 978-3-89879-567-8, 351 Seiten.

[B_UG_04]

Uwe Gresser(19.03.2004),

Trading mit Cfds: Methoden, Ansätze, Strategien,

ISBN-13: 978-3898797405, 220 Seiten.

[B_UT_06]

Ulrich Tebert (07.12.2006),

Die Eignung gängiger Indikatoren der Technischen Analyse zur Performancesteigerung,

ISBN-13: 9783836600170, 86 Seiten.

[ORA_BR_16]

Bert Rich (01.03.2016),

Oracle Database Reference, 12c Release 1 (12.1),

E41527-22, 2192 Seiten.

[ORA_KR_09]

Kathy Rich (April, 2009**),**

Oracle Database Reference, 10g Release 2 (10.2),

B14237-04, 948 Seiten.

[ORA_LA_15]

Lance Ashdown, Tom Kyte (November, 2015),

Oracle Database Concepts, 12c Release 1 (12.1),

E41396-13, 646 Seiten.

[ORA_MC_05]

Michele Cyran (Oktober, 2005),

Oracle Database Concepts, 10g Release 2 (10.2),

B14220-02, 542 Seiten.

[ORA_MC_15]

Maria Colgan (Juli, 2015)

White Paper Oracle Database In-Memory,

29 Seiten.

Abgerufen von http://www.oracle.com/technetwork/database/in-memory/overview/twp-oracle-database-*In-Memory*-2245633.html am 02.07.16.

[ORA_MB_16]

Mary Beth Roeser (Januar, 2016),

Oracle Database SQL Language Reference, 12c Release 1 (12.1),

E41329-20, 1912 Seiten.

[ORA_RB_16]

Rajesh Bhatiya, Immanuel Chan, Lance Ashdown (Februar, 2016),

Oracle Database Performance Tuning Guide, 12c Release 1 (12.1),

E49058-07, 340 Seiten.

[ORA_RR_07]

Ravi Rajamani (Juni, 2007)

Oracle Database 11g: SecureFiles – An Oracle White Paper,

Abgerufen von

http://www.oracle.com/technetwork/database/options/compression/overview/securefiles-131281.pdf am 28.07.2016.

[ORA_RU_14]

Randy Urbano (August, 2014),
Oracle Database Administrator's Guide, 12c Release 1 (12.1),
E41484-10, 1406 Seiten.

[ORA_RU_16]

Randy Urbano (01.07.2016),
Oracle Database Administrator's Guide, 12c Release 1 (12.1),
E41484-11, 1662 Seiten.

[ORA_TK_15]

Tom Kyte, Lance Ashdown (Mai, 2015**)**
Oracle Database Concepts, 11g Release 2 (11.2),
E40540-04, 472 Seiten.

[ORA_TL_15]

Tirthankar Lahiri (März, 2015),
White Paper When To Use Oracle Database In-Memory,
11 Seiten. Abgerufen von http://www.oracle.com/technetwork/database/in-memory/overview/twp-dbim-usage-2441076.html am 12.07.2016

[I_AR_15]

Andy Rivenes (24.04.2015),
In-Memory Priority,
Abgerufen am 04.08.2016 von
https://blogs.oracle.com/In-Memory/entry/in_memory_priority.

[I_AR_16]

Andy Rivenes (20.02.2016),
What is an In-Memory Compression Unit (IMCU)?,
Abgerufen am 02.07.2016 von
https://blogs.oracle.com/In-Memory/entry/what_is_an_in_memory.

[I_CL_11]

Chris Lomont(21.06.2011),
Introduction to Intel® Advanced Vector Extensions,
Abgerufen am 16.07.2016 von
https://software.intel.com/en-us/articles/introduction-to-intel-advanced-vector-extensions.

[I_CL_16]

Claus Lampert(o.J.),

„Der Momentum – Indikator“,

Abgerufen am 27.07.2016 von http://www.charttec.de/html/indikator_momentum.php.

[I_DB_15]

Don Burleson (13.01.2015),

inmemory_clause_default Tips,

Abgerufen am 28.07.2016 von

http://www.dba-oracle.com/t_inmemory_clause_default.htm.

[I_DB2_15]

Don Burleson (13.01.2015),

optimizer_inmemory_aware Tips,

Abgerufen am 28.07.2016 von

http://www.dba-oracle.com/t_optimizer_inmemory_aware.htm.

[I_JD_15]

Julian Dontcheff (14.01.2015),

The 7 Initialization Parameters Related to the IM Column Store,

Abgerufen am 08.08.2016 von

https://juliandontcheff.wordpress.com/2015/01/14/the-7-initialization-parameters-related-to-the-im-column-store/.

[I_FH_11]

Dr. Frank Haney (DOAG-Konferenz 16.11.2011),

Abgerufen am 16.07.2016 von

http://www.doag.org/formes/servlet/DocNavi?action=getFile&did=3362425&key=

[I_MC_14]

Maria Colgan (25.06.2014),

Oracle Database In-Memory Population,

Abgerufen am 02.08.2016 von

https://blogs.oracle.com/In-Memory/entry/more_q_a_on_oracle.

[I_MCO_14]

Maria Colgan (17.08.2014),

Getting started with Oracle Database In-Memory Part III - Querying The IM Column Store, Abgerufen von

https://blogs.oracle.com/In-Memory/entry/getting_started_with_oracle_database2.

[I_MC_16]

Maria Colgan (12.02.2016),
„How do I limit the amount of memory each PDB can use in the IM column store?“,
Abgerufen am 25.07.2016 von https://blogs.oracle.com/In-Memory/entry/questions_you_asked_how_do.

[I_MRM_2016]

Mahesh Reddy M (2016),
How do i configure Oracle database 12c In-memory Option,
Abgerufen am 27.07.2016 von
http://en.community.dell.com/techcenter/enterprise-solutions/w/oracle_solutions/11379.how-do-i-configure-oracle-database-12c-in-memory-option.

[I_ORA1]

o.V. (16.5.2014),
Compression Advisor,
Abgerufen am 16.07.2016 von
http://www.oracle.com/technetwork/database/options/compression/compression-advisor-095705.html

[I_OV_16]

o.V. (Juni, 2016),
Query Acceleration of Oracle Database 12c In-Memory using Software on Chip Technology with Fujitsu M10 SPARC Servers,
Abgerufen am 05.08.2016 von http://www.oracle.com/technetwork/database/in-memory/overview/twp-inmem-on-m10-en-2714932.html

[I_RB_14]

Rene Berteit (17.01.2017)
Indikatoren-Knowhow: Aroon Up/Down,
Abgerufen am 27.07.2016 von http://www.godmode-trader.de/know-how/indikatoren-knowhow-aroon-updown,3635454.

[I_US_11]

Ulrike Schwinn (Juli, 2011),
Speicherplatzeinsparung durch Komprimierung und der Compression Advisor dbms_compression, Abgerufen am 15.07.2016 von
http://www.oracle.com/webfolder/technetwork/de/community/dbadmin/tipps/compadv/index.html.

[I_YA_16]

Yahoo! Inc (2016),

Bestandteile – DAX,

Erstmalig abgerufen von https://de.finance.yahoo.com/q/cp?s=%5EGDAXI am 15.04.2016.

9. Anhang

SQL-Prozeduren der Indikatoren

MACD

```
CREATE OR REPLACE PROCEDURE MACD
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT number,
  RES OUT NUMBER
)
AS
  EMA number:=0; -- Variable für EMA-Wert
  EMA0 number:=0; -- EMA-Startwert
  GF number:=2/(1+AnzahlT); --Glättungsfaktor
  V_CLOSE AKTIEN%ROWTYPE;

CURSOR AKT_CUR
  IS
    SELECT * FROM AKTIEN
    WHERE AKTIEN_ID = FIRMA
    AND DATUM <= DATUM_A-1
    AND ROWNUM <= AnzahlT
    ORDER BY DATUM ASC;
  BEGIN
      OPEN AKT_CUR;
      FETCH AKT_CUR INTO V_CLOSE;
            IF V_CLOSE.CLOSE_S > 0
             THEN
              EMA0:=V_CLOSE.CLOSE_S;
             ELSE
              DBMS_output.put_line('MACD ERROR');
            END IF;
            EMA:= ((V_CLOSE.CLOSE_S - EMA0) * GF )+ EMA0;
      LOOP
            FETCH AKT_CUR INTO V_CLOSE;
            EXIT WHEN AKT_CUR%NOTFOUND;
            IF AKT_CUR%ROWCOUNT >'2' AND AKT_CUR%ROWCOUNT<AnzahlT
             THEN
              EMA:= ((V_CLOSE.CLOSE_S -EMA) * GF )+ EMA;
              ELSE
              EXIT WHEN AKT_CUR%NOTFOUND;
            END IF;
      END LOOP;
    RES:=EMA;
  CLOSE AKT_CUR;
END;
/
```

Momentum

```
CREATE OR REPLACE PROCEDURE MOM
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT IN number,
  Momentum_Z OUT number
)
AS
  y number:=0; -- Schlusskurs zum Startwert
  t number:=0; -- Schlusskurs des letzten Wertes in der Periode
  RES number:=0; -- Ergebis
  Momentum Aktien%ROWTYPE;
  CLOSE_T AKTIEN.CLOSE_S%TYPE;
  DATUM_C AKTIEN.DATUM%TYPE;

 CURSOR MOM_CUR
    IS
        SELECT *
        FROM AKTIEN
        WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <=
AnzahlT+1
        ORDER BY DATUM DESC;
  BEGIN
    OPEN MOM_CUR;
    LOOP
      FETCH MOM_CUR INTO  Momentum;
      EXIT WHEN MOM_CUR%NOTFOUND;
      IF MOM_CUR%ROWCOUNT ='1'
       THEN
        y:=Momentum.CLOSE_S;
      ELSIf MOM_CUR%ROWCOUNT = AnzahlT
       THEN
        t:=Momentum.CLOSE_S;
      END IF;
      EXIT WHEN MOM_CUR%NOTFOUND;
    END LOOP;

   IF y > 0 AND t > 0
    THEN
     RES:=(y/t)*100;
    ELSE
     DBMS_OUTPUT.PUT_LINE('MOMENTUM ERROR');
   END IF;

   Momentum_Z:=RES;
  CLOSE MOM_CUR;
END;
/
```

Average True Range

```
CREATE OR REPLACE PROCEDURE ATR
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT IN number,
  ATRa OUT number

)
AS

  y number:=0; -- Schlusswert
  t number:=0; -- Höchster Wert
  z number:=0; -- Niedrigster Wert
  SP1 number:=0; -- Spanne 1
  SP2 number:=0; -- Spanne 2
  SP3 number:=0; -- Spanne 3

  V_WPR AKTIEN%ROWTYPE;

CURSOR WPR_CUR
    IS
        SELECT *
        FROM AKTIEN
        WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <=
AnzahlT
        ORDER BY DATUM DESC;
  BEGIN
  OPEN WPR_CUR;
      FETCH WPR_CUR INTO V_WPR;
        y:= V_WPR.CLOSE_S;
        t:= V_WPR.HIGH_S;
        z:= V_WPR.LOW_S;
      LOOP
        FETCH WPR_CUR INTO V_WPR;
            IF WPR_CUR%ROWCOUNT >='2' and  WPR_CUR%NOTFOUND
             THEN
                SP1:=abs(t-z);
                SP2:=abs(t-V_WPR.CLOSE_S);
                SP3:=abs(z-V_WPR.CLOSE_S);
            END IF;
            EXIT WHEN WPR_CUR%NOTFOUND;
      END LOOP;
            IF SP1 > SP2 AND SP1 > SP3
             THEN
                ATRa:=SP1;
            ELSIF SP2 > SP1 AND SP2 > SP3
             THEN
                ATRa:=SP2;
            ELSE ATRa:=SP3;
            END IF;
  CLOSE WPR_CUR;
END;
/
```

Aroon

```
CREATE OR REPLACE PROCEDURE Aroon
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT IN number,
  AroonUP OUT number,
  AroonDown OUT number
)
AS
  y number:=0; -- Schlusskurs Startdatum
  t number:=0; -- Anzahl bereits gezählter Tage
  V_ARO AKTIEN%ROWTYPE;
  AroonUPx number:=0;
  AroonDownx number:=0;
CURSOR Aro_CUR
    IS
        SELECT *
        FROM AKTIEN
        WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <=
AnzahlT
        ORDER BY DATUM DESC;
BEGIN
  OPEN Aro_Cur;
      FETCH Aro_Cur INTO V_ARO;
      y:= V_ARO.CLOSE_S;
    LOOP
      FETCH Aro_Cur INTO V_ARO;
      EXIT WHEN Aro_CUR%NOTFOUND;
      IF y < V_ARO.Close_S AND AroonUPx=0
       THEN
        t:=Aro_Cur%ROWCOUNT;
        AroonUPx:=100*((AnzahlT-t)/AnzahlT);
      ELSIF y > V_ARO.Close_S AND AroonDownx=0
       THEN
        t:=Aro_Cur%ROWCOUNT;
        AroonDownx:=100*((AnzahlT-t)/AnzahlT);
      END IF;
      EXIT WHEN Aro_Cur%NOTFOUND;
      AroonDown:=AroonDownx;
      AroonUp:=AroonUpx;
    END LOOP;
CLOSE Aro_Cur;
END;
/
```

Bollinger Band

```
CREATE OR REPLACE PROCEDURE BB
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  BB_UP out number,
  BB_DOWN OUT NUMBER
)
AS

  CLOSE_T AKTIEN.CLOSE_S%TYPE;
  DATUM_C AKTIEN.DATUM%TYPE;
  MWx Number; --Mittelwert
  StdDevx Number; --Standardabweichung
  V_BB AKTIEN%ROWTYPE;

CURSOR BB_CUR
    IS
        AVG(CLOSE_S) OVER (ORDER BY DATUM) "MWx",
        SELECT DATUM, CLOSE_S,
        STDDEV(CLOSE_S) OVER (ORDER BY DATUM) "StdDevx"
        FROM AKTIEN
        WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <= 20
        ORDER BY DATUM ASC;
BEGIN
  OPEN BB_CUR;
    LOOP
      FETCH BB_CUR INTO DATUM_C,CLOSE_T,MWx,StdDEVx;
          EXIT WHEN BB_CUR%NOTFOUND;
    END LOOP;
      IF MWx > 0 AND StdDEVx > 0 AND CLOSE_T > 0
        THEN
            BB_UP:=MWx+(2*StdDEVx);
            BB_DOWN:=MWx-(2*StdDEVx);
        ELSE DBMS_OUTPUT.put_line('BB ERROR');
      END IF;
  CLOSE BB_CUR;
END;
/
```

Relative Strength Index

```
CREATE OR REPLACE PROCEDURE RSI
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT IN number,
  RSIx OUT NUMBER
)
AS

  x number:=0;
  y number:=0;
  t number:=0;
  z number:=0;
  pos number:=0;
  neg number:=0;
  V_CLOSE AKTIEN%ROWTYPE;

CURSOR RSI_CUR
    IS
      SELECT * FROM AKTIEN
      WHERE AKTIEN_ID = FIRMA
      AND DATUM <= DATUM_A-1
      AND ROWNUM <= AnzahlT
      ORDER BY DATUM DESC;
  BEGIN

OPEN RSI_CUR;

    FETCH RSI_CUR INTO V_CLOSE;

    x:=V_CLOSE.CLOSE_S; -- Startwert

     LOOP
      FETCH RSI_CUR INTO V_CLOSE;
      EXIT WHEN RSI_CUR%NOTFOUND;
    IF RSI_CUR%ROWCOUNT >'1' AND RSI_CUR%ROWCOUNT <= AnzahlT THEN
      y:=V_CLOSE.CLOSE_S -x; -- Differenz zum nächsten Schlusswert

        IF y > '0' THEN
        pos:= pos + y;
        ELSIF y < '0' THEN
        neg:= neg + y;
    END IF;
         x:=V_CLOSE.CLOSE_S; -- X Wird auf den nächsten Schlusswert
gesetzt

    EXIT WHEN RSI_CUR%ROWCOUNT = AnzahlT;
    END LOOP;
    pos:=pos/AnzahlT; -- Durschschnittliche Positive Änderung
    neg:=(neg*(-1))/AnzahlT; -- Durschschnittliche Negative Änderung
    IF pos > 0 AND neg > 0 THEN
    RSIx:= 100-(100/(1+(pos/neg)));
    ELSE DBMS_output.put_line('RSI ERROR');

CLOSE RSI_CUR;
END;
```

Williams Percent Range

```
CREATE OR REPLACE PROCEDURE WPR
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT IN number,
 WPRx OUT Number
)
AS

  y number:=0;
  t number:=0;
  CLOSE_T AKTIEN.CLOSE_S%TYPE;
  DATUM_C AKTIEN.DATUM%TYPE;
  Maxi VARCHAR2(512);
  Mini VARCHAR2(512);


  V_BB AKTIEN%ROWTYPE;

CURSOR WPR_CUR
    IS
    SELECT DATUM, CLOSE_S,
    MAX(HIGH_S) OVER (ORDER BY DATUM) "Maxi",
    Min(Low_S) OVER (ORDER BY DATUM) "Mini"
    FROM AKTIEN
    WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <=
AnzahlT
    ORDER BY DATUM ASC;
  BEGIN
  OPEN WPR_CUR;
    LOOP
      FETCH WPR_CUR INTO DATUM_C,CLOSE_T,Maxi,Mini;
EXIT WHEN WPR_CUR%NOTFOUND;

          END LOOP;
          IF MAXI > 0 AND MINI > 0 AND CLOSE_T > 0 AND MAXI > MINI THEN
     WPRx:=((Maxi-Close_T)/(Maxi-Mini))*100;
ELSE DBMS_OUTPUT.PUT_LINE('WPR ERROR');
END IF;

  CLOSE WPR_CUR;
END;
/
```

On-Balance-Volume

```
CREATE OR REPLACE PROCEDURE OBV
(
  DATUM_A IN DATE,
  Firma IN VARCHAR2,
  AnzahlT number,
  proz OUT NUMBER
)
AS
  x number:=0;
  y number:=0;
  t number:=0;
  up number:=0;
  down number:=0;
  neutral number:=0;
  obvx number:=0;
  V_OBV AKTIEN%ROWTYPE;

CURSOR OBV_CUR
    IS
    SELECT *
    FROM AKTIEN
    WHERE DATUM <= DATUM_A-1 AND AKTIEN_ID = FIRMA  AND ROWNUM <=
AnzahlT AND Volume_S !='0'
    ORDER BY DATUM ASC;
  BEGIN
  OPEN OBV_CUR;
      FETCH OBV_CUR INTO V_OBV;
         y:= V_OBV.CLOSE_S;
         x:= V_OBV.Volume_S;
LOOP
      FETCH OBV_CUR INTO V_OBV;
         EXIT WHEN OBV_CUR%NOTFOUND;
        IF y > V_OBV.Close_S THEN
         up:= up+1;
         t:=OBV_CUR%ROWCOUNT;
         obvx:= obvx + V_OBV.Volume_S;
         y:=V_OBV.Close_S;
        ELSIF y < V_OBV.Close_S THEN
         down:=down+1;
         t:=OBV_CUR%ROWCOUNT;
         obvx:= obvx - V_OBV.Volume_S;
         y:=V_OBV.Close_S;
        ELSIF y = V_OBV.Close_S THEN
         t:=OBV_CUR%ROWCOUNT;
         obvx:= obvx *1;
         y:=V_OBV.Close_S;
          neutral:=neutral+1;
        END IF;
         EXIT WHEN OBV_CUR%NOTFOUND;

END LOOP;
 IF x= 0 THEN
   DBMS_OUTPUT.put_line('Startwer ist nicht vorhanden');
 ELSIF obvx > 0 THEN
  proz:= (x * 100)/obvx;
 ELSE DBMS_OUTPUT.PUT_LINE('OBV ERROR');
 END IF;
CLOSE OBV_CUR;
END;
/
```

Kombination mit Punktesystem

```
CREATE OR REPLACE PROCEDURE Kombination
(  DATUM_Ax IN DATE,  Firma IN VARCHAR2)
AS
x number:=0; y number:=0; t number:=0; BB_UP number;
BB_DOWN Number; RSIx number; resu number; resu2 number; MACDw Number;
momen number; up number; down number; wpr_y number; prozen number;
atr_y number; V_DAY number; V_DAYB number; RSIV number; MACDV number;
MOMV number; AROV number; OBVV number; WPRV number; allresult number;
Datum_A date :=DATUM_Ax;

BEGIN

BB(DATUM_A,Firma,BB_UP,BB_DOWN);
DBMS_output.put_line('Max. Steigung: ' || BB_UP);
DBMS_output.put_line('Max. Gef鋏e: ' || BB_DOWN);

RSI(DATUM_A,Firma,'7',RSIx);
IF RSIx = '50' THEN RSIV :='0'; ELSIF RSIx < '50' THEN RSIV :='-1';
ELSIF RSIx < '30' THEN RSIV :='-2';ELSIF RSIx > '50' THEN RSIV :='1';
      ELSIF RSIx > '70' THEN RSIV :='2'; END IF;
DBMS_output.put_line('RSI: ' || RSIx);

MACD(DATUM_A,Firma,26,resu);
MACD(DATUM_A,Firma,12,resu2);
MACDw := RESU2-RESU;
DBMS_OUTPUT.put_line('MACD:' || TO_CHAR(MACDw,'0.9999999'));
IF MACDw > 0 THEN MACDV:='1'; ELSIF MACDw < 0 THEN MACDV:='-1';
ELSIF MACDw = 0 THEN MACDV:='0';END IF;

MOM(DATUM_A,Firma,10,momen);
DBMS_OUTPUT.put_line('Rate of Change:' || momen);
IF momen = '50' THEN RSIV :='0';ELSIF momen < '50' THEN MOMV :='-1';
ELSIF momen < '30' THEN MOMV :='-2';ELSIF momen > '50' THEN MOMV :='1';
ELSIF momen > '70' THEN MOMV :='2';END IF;

Aroon(DATUM_A,Firma,14,up,down);
IF up > down THEN AROV:='1';ELSIF down > up THEN AROV:='-1';END IF;
DBMS_OUTPUT.put_line('UP:' || up);DBMS_OUTPUT.put_line('DOWN:' ||down);

WPR(DATUM_A,Firma,14,wpr_y);DBMS_OUTPUT.put_line('WPR: ' || wpr_y);
IF wpr_y = '50' THEN WPRV :='0';ELSIF wpr_y < '50' THEN WPRV :='-1';
ELSIF wpr_y < '30' THEN WPRV :='-2';ELSIF wpr_y > '50' THEN WPRV :='1';
ELSIF wpr_y > '70' THEN WPRV :='2';END IF;

OBV(DATUM_A,Firma,14,prozen);
IF prozen > 0 THEN OBVV:='1';ELSIF prozen < 0 THEN OBVV:='-1';
END IF;
DBMS_OUTPUT.put_line('OBV Įderung:' || TO_CHAR(prozen,'00.99'));

ATR(DATUM_A,Firma,2,atr_y);
DBMS_OUTPUT.put_line('ATR:' || TO_CHAR(atr_y,'0.99'));
allresult:= RSIV+MACDV+MOMV+AROV + OBVV+  WPRV ;
  DBMS_OUTPUT.put_line('Punkte:' ||allresult);
END;
/
```

Prozedur JahresProg

```
CREATE OR REPLACE PROCEDURE JahresProg
( JAHR in VARCHAR,
  FIRMA IN VARCHAR
  )
AS
  V_AKTIEN AKTIEN%ROWTYPE;  x VARCHAR(512);  y DATE;
CURSOR ALL_CUR
    IS
    SELECT Datum, Aktien_ID
    FROM AKTIEN
    WHERE to_char (datum, 'yyyy')= JAHR AND AKTIEN_ID= FIRMA
    ORDER BY Datum ASC;
BEGIN
  OPEN ALL_CUR;
   LOOP
    FETCH ALL_CUR INTO x, V_AKTIEN.AKTIEN_ID;
    EXIT WHEN ALL_CUR%NOTFOUND;
      DBMS_output.put_line(x|| V_AKTIEN.AKTIEN_ID);
        Kombination(x, V_AKTIEN.AKTIEN_ID);
    END LOOP;
  CLOSE ALL_CUR;
END;
/
```

Prozedur AlleJahre

```
CREATE OR REPLACE PROCEDURE AlleJahre
(
RESULT OUT VARCHAR
)
AS
  V_AKTIEN AKTIEN%ROWTYPE;
x VARCHAR(512);
y VARCHAR(512);
CURSOR RES_CUR
  IS
  SELECT DISTINCT AKTIEN_ID, (TO_CHAR (DATUM, 'yyyy'))
  FROM AKTIEN
  WHERE AKTIEN_ID <> '^GDAXI'
  GROUP BY AKTIEN_ID, TO_CHAR(DATUM, 'yyyy')
  ORDER BY AKTIEN_ID, TO_CHAR(DATUM, 'yyyy');
 BEGIN
  OPEN RES_CUR;
      LOOP
      FETCH RES_CUR INTO y,x;
      EXIT WHEN RES_CUR%NOTFOUND;
          DBMS_output.put_line(x|| y);
      JahresProg(x, y);
             END LOOP;
  CLOSE RES_CUR;

END;
/
```