Nr. 2 | Februar 2024 | ISSN 2366-7915 | www.doag.org

Red Stack Magazin

DOAG SOUG swiss oracle user group AOUG AUSTRIAN ORACLE USER GROUP

inklusive BUSINESS NEWS

DATENBANKEN

Aus der Praxis

Lift & Shift – geht es auch etwas größer bitte?

Im Interview

Tirthankar Lahiri, Senior VP of Data and In-Memory Technologies for Oracle DB

Business News

Die Magie von APEX

CloudLand

WWW.CLOUDLAND.ORG

Eventpartner: Heise Medien

DAS CLOUD NATIVE FESTIVAL

DAS EVENT DER DEUTSCHSPRACHIGEN CLOUD NATIVE COMMUNITY

18. – 21. JUNI

#CLOUDLAND2024

Christian Trieb
Vorstand Datenbanken, Leiter Datenbank Community

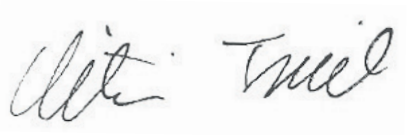
Liebe Mitglieder, liebe Leserinnen und Leser,

mit einem Interview mit Tirthankar Lahiri, ORACLE Senior Vice President, Data and In-Memory Technologies, startet diese Ausgabe zum Thema „Datenbanken“. Daten und insbesondere Daten, die in Datenbanken gespeichert werden, bilden immer noch die Grundlage zahlreicher Applikationen und Anwendungen. Tirthankar Lahiri verantwortet die Entwicklung der Oracle-Datenbank-Maschine, Oracle TimesTen, und NoSQLDB. Insoweit ist er ein kompetenter Ansprechpartner bei Oracle. Dies hat er auch durch seine Keynote während der letztjährigen DOAG 2023 Konferenz + Ausstellung gezeigt. Weitere Artikel beschreiben unterschiedliche Aspekte der Oracle-Datenbank wie den Optimizer, Oracle In-Memory, Sicherheitsaspekte und vieles mehr. Dieses Jahr soll ja die Oracle 23c Datenbank On-Prem verfügbar werden. Daher bietet es sich jetzt schon an, sich mit einigen Features auseinanderzusetzen. Insbesondere mit der Multitenant-Architektur, da die Oracle 23c Datenbank die bisherige Non-Multitenant-Architektur nicht mehr unterstützt.

Die Business News widmet sich dem Thema „Die Magie von APEX“. APEX, dessen Grundlagen in der Oracle-Datenbank liegen, wird hierbei genutzt, um Geschäftslogik schnell abbilden zu können. Dies zeigt, dass der Zusammenhang zwischen Technik und Geschäftsanwendungen sehr eng sein kann. Man kann mit APEX sehr datenbanknahe Logik schnell und gut abbilden.

Auch die DOAG bietet im laufenden Jahr mit unterschiedlichen Veranstaltungen (zum Beispiel: DOAG 2024 Datenbank mit Exaday und der APEX connect) diesen Themen sehr breiten Raum, um sie aus unterschiedlichen Perspektiven zu betrachten.

In diesem Sinne wünsche ich Ihnen einen regen Erkenntnisgewinn beim Lesen der Artikel dieser Ausgabe.
Bleiben Sie gesund und ein gutes erfolgreiches 2024.



Christian Trieb

Ausgabe Nr. 2/2024
auf Abruf!

In-Memory im Oracle DWH – der ultimative Performance Boost?

Ein Blick in die Black-Box: Explainable AI (XAI) erklärt

Einleitung

Datenbank

KI

APEX

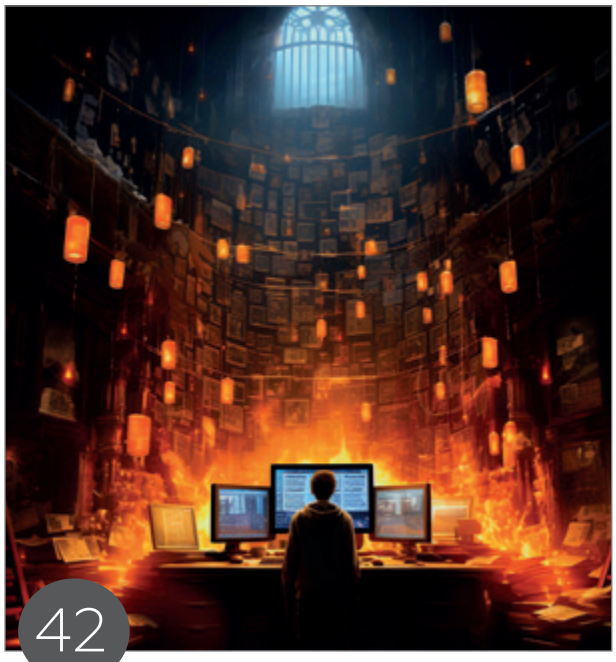
PL/SQL

Cloud

Security

BUSINESS NEWS

Die Magie von APEX



Leitartikel | Wie Oracle APEX zu einem leistungsstarken Enterprise-App-Entwicklungs-Framework wurde

Lift & Shift – geht es auch etwas größer bitte?

Die Magie von APEX – Interview mit Niels de Bruijn

Weg mit dem Speck – Verchlankung der inneren Unternehmensstrukturen mit Oracle APEX

Intern

News

TIMELINE

22. Dezember 2023

Die DOAG Geschäftsstelle schließt und verabschiedet sich bis zum Jahresanfang in die Betriebsferien und blickt auf ein ereignisreiches Jahr 2023 zurück.

11. Januar 2024

Im DevTalk mit Robert Marz und Oliver Lemm dreht sich alles rund um den Erfahrungsaustausch zum Thema „SQLcl."

31. Januar 2024

Beim Regionaltreffen NRW in Dortmund stehen zwei Vorträge auf der Agenda. „Wie geil ist das denn? Die KI macht meine Arbeit!" von Frank Meyfarth und „Decision Model Notation." Es moderiert Axel vom Stein.

9. Februar 2024

In der DB WebSession mit Steffen Nothmann erfahren die Teilnehmenden alles über das Thema „Alles rund um die Oracle-Namensgebung."

APEX connect

by DOAG

22. - 24.04.2024

VAN DER VALK AIRPORTHOTEL DÜSSELDORF

apex.doag.org

„Ein besonders prominenter Analyst bezeichnete das relationale JSON-Dualitätsfeature als eine der wichtigsten Innovationen in der Informatik der letzten zwanzig Jahre."

Martin Meyer, Redaktionsleiter des Red Stack Magazin, sprach mit Tirthankar Lahiri, Senior Vice President of Data and In-Memory Technologies for Oracle Database, über wichtige Features in der Oracle Database 23c, KI-Funktionalität bei Oracle, die Converged Database, die KI-Vektorsuche sowie in Zukunft zu erwartende Entwicklungen in Bezug auf Datenbanken.

Bitte stellen Sie sich unseren Lesern vor. Wer sind Sie und was ist Ihre Aufgabe bei Oracle?

Mein Name ist Tirthankar Lahiri, ich bin SVP of Data and In-Memory Technologies for Oracle Database. Ich leite die Oracle Database „Data Engine"-Organisation, die geschäftskritische Bereiche wie Advanced Compression, Database In-Memory, Database Filesystem, Flashback Time Travel und vieles mehr umfasst. Ebenso leite ich die Oracle TimesTen In-Memory-Datenbank – eine leistungsstarke In-Memory-Datenbank für die Transaktionsverarbeitung – und Oracle NoSQLDB, eine verteilte NoSQL-Datenbank, die auch als Cloud-Service auf OCI angeboten wird.

Was ist das wichtigste Feature in Oracle 23c und warum?

Oracle Database 23c verfügt über etliche herausragende neue Features, die das „App-Simple"-Thema unterstützen: die Möglichkeit, moderne Anwendungen jedes Typs und jeder Größenordnung zu entwickeln, bereitzustellen und auszuführen. Für mich ist das wichtigste Feature eine bahnbrechende Funktionalität, die wir als „JSON Relational Duality" bezeichnen. Dieses Feature löst definitiv das langjährige Problem des „Object Relational Mismatch", das Entwickler seit den frühen Tagen der Entwicklung plagt.

Obwohl relationale Datenbanken sehr leistungsfähig sind, neigen Entwickler dazu, Anwendungen in Bezug auf sprachspezifische hierarchische Objekte zu programmieren, aber relationale Datenbanken stellen Daten in Bezug auf Zeilen und Spalten dar. Dadurch müssen Entwickler ihre Anwendungsobjekte den zugrunde liegenden relationalen Strukturen zuordnen, die in der Datenbank gespeichert sind.

Bestehende Lösungen für dieses Problem umfassen die Verwendung von objektbezogenen Mapping-Tools (Object-Relational Mapping, ORMs) oder die Verwendung von Dokumentdatenbanken, die Daten nativ in hierarchischen Dokumenten speichern.

Keiner der beiden Ansätze funktioniert richtig gut: ORMs führen eine parallele Metadatenschicht ein, führen zu ineffizientem SQL mit mehreren Datenbankzugriffen für ein einzelnes Objekt und sind spezifisch für eine Programmiersprache. Das macht das Teilen desselben Mappings für einen Python-Microservice und einen JavaScript-Microservice schwierig.

Dokumentendatenbanken scheinen das Problem zu lösen, indem sie Anwendungsobjekte direkt in gespeicherte JSON-Dokumente abbilden. Sie leiden jedoch unter dem Problem der Datenduplizierung zwischen Dokumenten, die die gleichen Daten teilen (zum Beispiel dupliziert jedes Auftragsdokument Kundeninformationen über alle von einem Kunden erteilten Aufträge hinweg): Dieses Problem lösen relationale Datenbanken durch Datennormalisierung. Duplizierte Daten sind ineffizient zu speichern, aufwendig zu aktualisieren und können Inkonsistenzen verursachen.

Das relationale JSON-Dualitätsfeature löst dieses Problem auf elegante Weise, indem es eine vollständig konsistente transaktionale „JSON-Dualitätssicht" über die relationalen Daten erstellt, sodass Daten nahtlos als JSON-Dokumente gelesen und geschrieben werden können, während der zugrunde liegende Speicher in normalisierten relationalen Tabellen verbleibt. Beispielsweise ermöglicht eine Dualitätssicht für Bestellungen, dass jede Bestellung als ein einziges JSON-Dokument gelesen und geschrieben werden kann. Alle zugrunde liegenden Daten in der Sicht bleiben aber normalisiert. Es gibt nur eine Zeile in der Datenbank für jeden Kunden, selbst wenn die Kundendaten logisch von verschiedenen Bestelldokumenten geteilt werden. Dadurch entkoppelt die relationale JSON-Dualität das Datenzugriffsmodell vom Datenspeichermodell, wobei die Einfachheit des JSON-Zugriffs sowie die Leistungsfähigkeit des relationalen Speichers erhalten bleiben. Darüber hinaus ist die relationale JSON-Dualität weitaus effizienter als eine aufgesetzte Zuordnungsschicht, da alle Daten für ein Objekt in einem Datenbankzugriff abgerufen oder geschrieben und von der Oracle Database Execution Engine optimiert werden.

Aus diesem Grund haben Datenbankanalysten dieses Feature gelobt und ein besonders prominenter Analyst bezeichnete es als „eine der wichtigsten Innovationen in der Informatik der letzten zwanzig Jahre."

Wie wird sich die KI-Fähigkeit von Oracle auf die tägliche Arbeit von Administratoren, Betreibern und Entwicklern auswirken?

Oracle nähert sich dem Thema KI aus mehreren Richtungen. Erstens verfügt Oracle Database bereits über eine Vielzahl an integrierten Machine-Learning-Modellen, die das Training und die Inferenz von Daten in der Datenbank mit SQL ermöglichen, ohne dass eine Datenübertragung erforderlich ist. Zweitens fügt Oracle für die neue generative KI-Revolution native Unterstützung für die KI-Vektorsuche hinzu: Vektoren sind im Wesentlichen Zahlenarrays zur Darstellung von Merkmalen unstrukturierter Daten wie Text, Video oder Bilder. Die Vektorsuche ermöglicht es der Oracle-Datenbank, eine Ähnlichkeitssuche in Datenbankdaten durchzuführen und diese Daten zu verwenden, um Interaktionen mit Large Language Models (LLMs) wie OpenAI und Cohere zu erweitern. Drittens entwickelt Oracle auch eine vollständige generative KI-Plattform auf OCI, die LLM-Interaktionen, Vektoreinbettungsmodelle und Retrieval-Augmentation umfasst – also alle notwendigen Funktionen einer vollständigen generativen KI-Lösung. All diese Funktionen ermöglichen eine einfachere Verwaltung und einen einfacheren Betrieb für KI, da nicht mehr unterschiedliche Datenbanken und Dienste verwaltet werden müssen. Das vereinfacht das Leben von Entwicklern erheblich, die sich auf ihre spezifischen KI-Probleme konzentrieren können, ohne sich um die Datenintegration kümmern zu müssen.

Wie wird die KI-Funktionalität von Oracle die Art und Weise beeinflussen, wie Benutzer auf Daten zugreifen?

Die KI-Fähigkeiten von Oracle werden es ermöglichen, immer mehr Aspekte moderner Anwendungen zu generieren, sodass diese nicht mehr von Grund auf neu entwickelt werden müssen. Auch wird KI es Entwicklern ermöglichen, viel mehr mit ihren Daten zu interagieren. Die Anweisung „Select AI" von Oracle in Autonomous Database ermöglicht es Entwicklern, SQL aus natürlicher Sprache zu generieren, und es sind weitere KI-gestützte Tools zur Entwicklungsautomatisierung in der Pipeline.

Wie nutzen Entwickler Ihrer Erfahrung nach die Converged Database am häufigsten?

Entwickler verwenden konvergente Datenbanken, um auf mehrere Datenmodelle und Verarbeitungsalgorithmen in einer einzigen SQL-Anweisung zuzugreifen. Anstatt eine separate Dokumentendatenbank, eine Geodaten-Datenbank und eine relationale Datenbank zu benötigen, kann eine Restaurantreservierungs-App einfach mit einer konvergenten Datenbank entwickelt werden. So können die Restaurants in einem bestimmten Gebiet (Geodaten) gefunden, nach Restaurants mit einer bestimmten Küche oder Menüpunkten gesucht oder Merkmale überprüft (Dokument) und die Reservierung für den Benutzer vorgenommen werden (relational). Es ist viel einfacher, ein paar Zeilen SQL zu schreiben, als auf mehrere Datenbanken zugreifen zu müssen!

Was sind die bisher attraktivsten Features von Converged Database?

Viele moderne Apps kombinieren heute relationale Verarbeitung und Dokumentenverarbeitung. Relational wird für die bekannten Teile des Schemas verwendet und JSON für Daten, deren Schema dynamisch ist oder sich weiterentwickelt. Viele IoT-Anwendungen verwenden relationale Attribute für bekannte Felder wie Geräte-ID, Zeitstempel, Ereignistyp, während sie JSON für die Ereignisdetails der Nutzdaten verwenden (die sich bei der Sensor-Telemetrie mit der Sensorversion und sogar mit der Firmware ändern können). Die kombinierte Leistung von JSON-Dokumenten und relationaler Verarbeitung macht diese Anwendungen einfach.

Was ist der nächste Schritt in Bezug auf die Leistungssteigerung, die wir von Oracle erwarten können?

Alles, was ich sagen kann, ist: Bitte bleiben Sie dran! Wir entwickeln ständig Innovationen in Bezug auf Leistung und Skalierbarkeit für Analysen, Transaktionsverarbeitung und jetzt auch für die KI-Vektorsuche. Es erwarten Sie stetige architektonische Verbesserungen in all diesen Bereichen: In-Memory-, Scale-Out-, immer schnellere KI-Suchen.

Welche Trends sehen Sie und welche Entwicklungen erwarten Sie in Zukunft in Bezug auf Datenbanken?

Das ist eine sehr interessante Frage. Wir sehen, wie sich Datenbanken von Datenverarbeitungs- und Speicher-Engines zu Generatoren für komplette Datenlösungen entwickeln. Möglich wird dies durch deklarierten „Intent" der Anwendung. Statt alle Aspekte einer App von Grund auf neu zu entwickeln, beschreiben Entwickler der Datenbank, wie sich die App verhalten soll. Zum Beispiel ist in 23c eine JSON-Dualitätssicht im Wesentlichen eine Erklärung, dass der Entwickler beabsichtigt, auf relationale Tabellen als JSON-Dokument zuzugreifen. Die Funktion „Usage Domains" ermöglicht es Entwicklern auch, ihre beabsichtigte Verwendung von Daten zu erklären, zum Beispiel die Absicht, eine varchar-Zeichenfolge als Kreditkartennummer oder Reisepassnummer zu verwenden. Erklärte Absichten (Intents) in Kombination mit generativer KI werden es ermöglichen, immer mehr Aspekte von Apps zu generieren, um das Leben der Entwickler zu vereinfachen, indem ein Großteil der routinemäßigen und sich wiederholenden Grundlagen für die App-Entwicklung abgeschafft wird.

TIRTHANKAR LAHIRI

Tirthankar Lahiri ist Senior Vice President of Data and In-Memory Technologies for Oracle Database. Er ist verantwortlich für die Data Engine für Oracle Database, einschließlich missionskritischer Bereiche wie Advanced Compression, Database In-Memory, Database Filesystem, Flashback Time Travel, etc. Darüber hinaus verwaltet er die Oracle TimesTen In-Memory-Datenbank und die Oracle NoSQLDB-Produktteams. Tirthankar verfügt über 28 Jahre Erfahrung in der Datenbankindustrie und hat an einer Vielzahl von Bereichen wie Leistung, Skalierbarkeit, Verwaltbarkeit, Caching, In-Memory-Architekturen und entwicklerorientierte Funktionalität gearbeitet. Er hat 61 erteilte und mehrere angemeldete Patente, einen Bachelor of Technology in Informatik vom Indian Institute of Technology und einen Master of Science in Elektrotechnik von der Stanford University.

Modern Database Flavours

Michael Schulze, Opitz Consulting Deutschland

Nahezu jedes Unternehmen betreibt Datenbanksysteme für die Datenhaltung. Die Entwicklung von Datenbanken reicht bis in die 1960er Jahre zurück, als Mainframesysteme noch dominierten. Den Durchbruch im kommerziellen Bereich brachte das relationale Datenbankmodell (SQL-basiert), das viele Jahre Bestand hatte, bis die Datenanforderungen wuchsen und neue Datenbankkonzepte wie NoSQL in den Fokus kamen. Die heutige Datenbanklandschaft ist eine Mischung aus SQL/NoSQL DBMS-Systemen, die On-Premises und in der Cloud betrieben werden. Zudem gibt es neue Entwicklungen wie „NewSQL“. Hier werden beide Ansätze berücksichtigt und auch mit KI-/AI-Methodik ergänzt. Die Datenbanklandschaften verändern sich also. Der folgende Artikel beschäftigt sich deshalb mit dieser Entwicklung und soll einen Überblick über die verschiedenen Flavours in modernen Datenbankumgebungen geben.

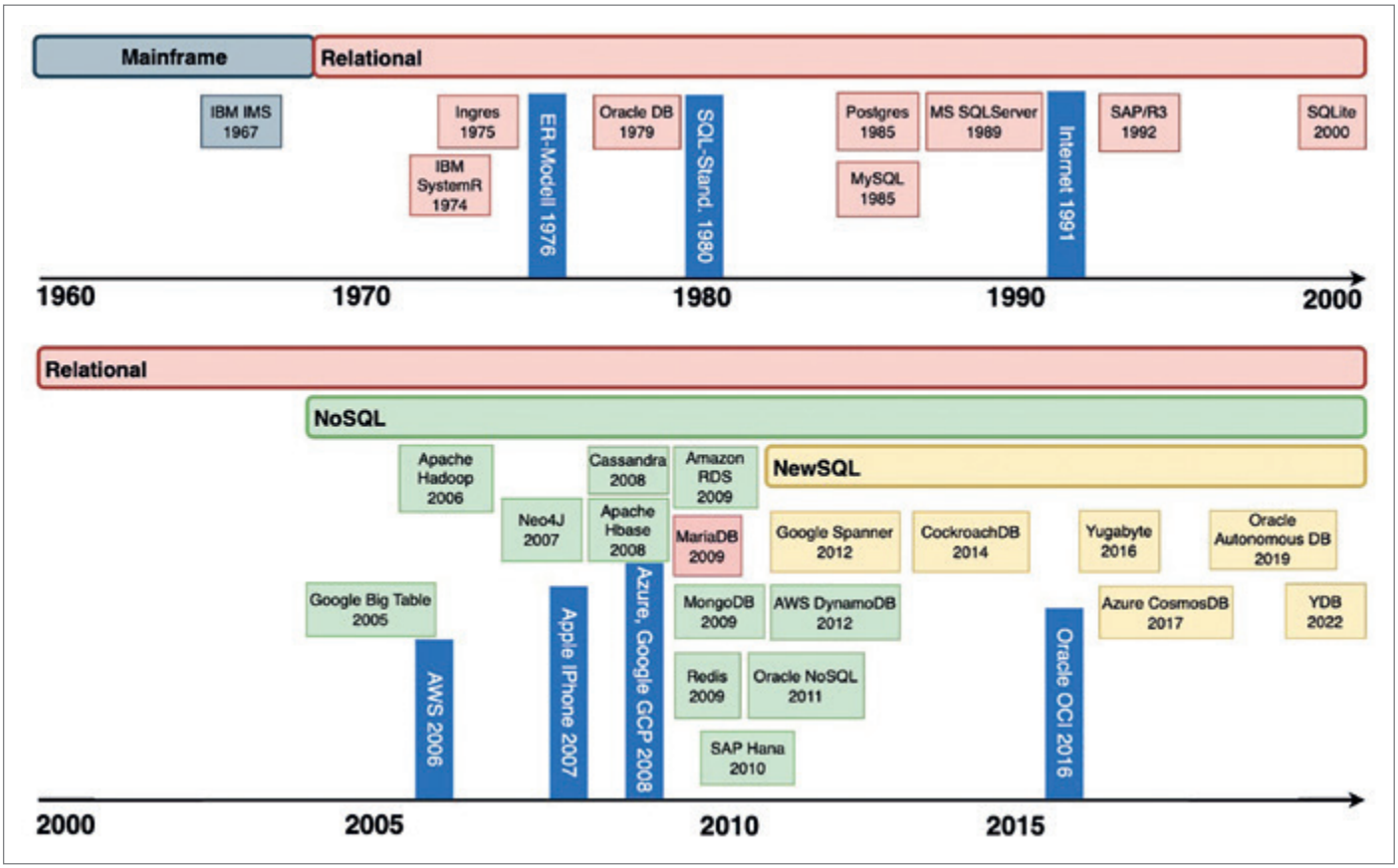


Abbildung 1: Database History 1960 bis heute (Quelle: Michael Schulze)

Geschichte

Erste Datenbanksysteme etablierten sich in den 1960er Jahren im Mainframe-Umfeld. Hier setzte sich zum Beispiel die Firma IBM mit IMS als hierarchisches Datenbanksystem durch. Die Entwicklungen hin zu Client-Server-Umgebungen brachte mit dem relationalen Datenbankmodell, das Anfang der 1970er Jahre seinen Einzug in die IT-Landschaften hielt, neue Lösungen im Datenbankbereich hervor. Diese etablierten sich in den folgenden Jahrzehnten und sind auch heute noch zentraler Bestandteil vieler Datenbankumgebungen. Grundlage für das relationale Datenbankmodell ist das Entity-Relationship-Modell (ER-Modell), das erstmals kontextbezogene Beziehungen und damit Objektabbildungen der realen Welt in Datenbankumgebungen ermöglichte. Das relationale Datenbankmodell wird in zweidimensionalen Tabellen (Spalten/Zeilen) organisiert. Diese können in Relation zu weiteren Tabellen stehen und damit Zusammenhänge definieren. Relationale Datenbanksysteme haben einen Fokus auf Transaktionssicherheit, die insbesondere im Multiuserbetrieb im Online Transaction Processing (OLTP) benötigt werden.

Unternehmen wie beispielsweise Oracle oder IBM etablierten das relationale Datenbankmodell im kommerziellen Enterprise-Bereich im Client-Server-Umfeld. Die verschiedenen relationalen DB-Lösungen führten zur Notwendigkeit einer Standardisierung ihrer Abfragesprache. Dieser Meilenstein wurde 1980 durch die Einführung des SQL-Standards erreicht und führte zur Verbreitung von relationalen Datenbanken. In den 1980er bis 2000er-Jahren wurden weitere Enterprise DBMS-Systeme entwickelt. Zu nennen sind hier: PostgreSQL, Microsoft SQL-Server, SAP/R3 und MySQL, aber auch sehr leichtgewichtige Datenbanken wie SQLite, die sich insbesondere für kleinere Projekte eignen. Ein wichtiger Meilenstein für die weitere Verbreitung von Datenbanksystemen war 1991 die Einführung des Internets. Daraus resultierten erste Cloud-Ausprägungen (1999: Salesforce SaaS) *[1]*. Verschiedene Player entwickelten ab 2003 Datenbanklösungen, die den Fokus auf die Verwaltung von sehr großen Datenbeständen (Bigdata) hatten (zum Beispiel Google BigTable, Apache Hadoop). Diese waren unter anderem auch eine Grundlage für weitere Entwicklungen hin zu eigenständigen Public-Cloudlösungen. So etablierte die Firma Amazon mit Amazon Webservices (AWS) 2006 seine eigene Cloudlösung *[2]*. Dadurch war es nun besser möglich, standortunabhängig zu arbeiten. Die neuen Möglichkeiten brachten jetzt auch zunehmend mobile Endgeräte in den Fokus. Als ein Initiator dafür kann man hier 2007 die Marktvorstellung des ersten Apple iPhones sehen *[3]*. Wachsende Mobilität wirkte sich auch auf die Applikationsanforderungen aus. Verbunden mit dem zunehmenden Einsatz von Sensorik, IoT und vielem mehr, stiegen auch die Datenanforderungen stetig an. Aus dieser Notwendigkeit heraus mussten relationale Datenbanken durch neue Datenbanksysteme ergänzt werden. NoSQL-Datenbanken (Not only SQL) konnten besser zur Bewältigung großer Datenmengen bezüglich Lesegeschwindigkeit, Verteilung und Verfügbarkeit genutzt werden, als das relationale Modell. Sie verbreiteten sich ab 2005 insbesondere im Datawarehouse (DWH)- und BI-Umfeld mit verschiedenen Ausprägungen, auf die später im Artikel noch

```
SQL> -- CREATE
SQL> set feedback on
SQL> CREATE TABLE test(
  2     type VARCHAR2(10),
  3     action VARCHAR2(5),
  4     descr VARCHAR2(10)
  5* );

Table TEST created.

SQL> set feedback off
SQL> INSERT INTO test VALUES ('SQL','C','CREATE');
SQL> INSERT INTO test VALUES ('SQL','R','READ');
SQL> INSERT INTO test VALUES ('SQL','U','UPDATE');
SQL> INSERT INTO test VALUES ('SQL','D','DELETE');
SQL> commit;
SQL> -- READ
SQL> SELECT * FROM test;

TYPE    ACTION    DESCR
------- --------- ---------
SQL     C         CREATE
SQL     R         READ
SQL     U         UPDATE
SQL     D         DELETE
SQL> -- UPDATE
SQL> UPDATE test SET descr = 'update' WHERE action = 'U';
SQL> commit;
SQL> SELECT * FROM test where action = 'U';

TYPE    ACTION    DESCR
------- --------- ---------
SQL     U         update
SQL> -- DELETE
SQL> DELETE FROM test WHERE action = 'D';
SQL> commit;
SQL> SELECT * FROM test where action = 'D';

SQL> set feedback on
SQL> DROP table test;

Table TEST dropped.
```

Abbildung 2: SQLPlus-Output (Quelle: Michael Schulze)

genauer eingegangen wird. Es folgten weitere Public-Cloudlösungen insbesondere 2008: Microsoft Azure, Google Cloud Platform (GCP) und 2016 die Oracle Cloud Infrastructure (OCI) *[4] (siehe Abbildung 1)*. Alle Lösungen brachten auf Grundlage ihrer Infrastruktur neue SQL- und NoSQL-Cloud-Datenbanken mit, die einfache bis verwaltete Services boten. So entstanden auch DB-Services, die Bestandteile von SQL und NoSQL beinhalteten. Zu nennen sind hier etwa die Oracle Autonomous Database, Microsoft Cosmos DB, Google Cloud Spanner und moderne Postgres-Ausprägungen wie Yugabyte und YDB. Diese neuen Lösungen werden auch als „NewSQL" bezeichnet. Zudem hat sich ein Trend der Nutzung von Multicloud-Lösungen etabliert. Hier bieten die Cloudhersteller schon verschiedene Möglichkeiten an. Als ein Beispiel ist der „Oracle Database Service for Azure" zu nennen, der seit 2023 einen verwalteten Datenbank-Service in der Oracle-Cloud für Azure-Kunden über einen direkten Interconnect der beiden Clouds ermöglicht. Zudem lassen sich in jüngster Zeit Entwicklungen hin zu mehr Automatisierung im Datenbankbetrieb durch Nutzung und Integration von KI-/AI-Methoden erkennen.

Nach diesem Ausflug in die Historie und der Entwicklung von Datenbanksystemen möchte ich auf zwei wesentliche Konzepte eingehen und die Fakten und Ausprägungen in Kurzform beschreiben. Ein kleines Beispiel soll zudem die technischen Unterschiede jeder Ausprägung noch besser darstellen.

Relationale Datenbanksysteme (SQL)

Eigenschaften relationaler Datenbanksysteme sind strukturierte Datendefinitionen. Daten werden in zweidimensionalen Tabellen (rowstore) gehalten, diese stehen in Beziehung zueinander und ermöglichen referenzielle Integrität. Ein wesentlicher Bestandteil ist die Structured Query Language (SQL), die als Abfragesprache verwendet wird. Relationale Datenbanken verfolgen das Konzept der Transaktionssicherheit, Konsistenz und Datenintegrität (ACID). Hier spielen die Einhaltung von Faktoren wie Atomarität, Konsistenz, Isolation und Dauerhaftigkeit der Datenhaltung eine zentrale Rolle. Deshalb sind die Systeme sehr gut für den Multiuser-Betrieb mit vielen Transaktionen (Schreib- und Lesezugriffen) geeignet.

Neben „Online Transaction Processing" (OLTP)-Datenbanken mit aktivem Datenbestand und Größenordnungen im GB- bis TB-Bereich gibt es weitere Ausprägungen wie Online Analytical Processing (OLAP). Diese oft großen Datenbanken im Terabyte-/Petabyte-Bereich sind mehrdimensional organisiert und führen mehrere (aktive) Datenbestände durch ETL-Prozesse in passive Datenbestände zusammen, die der Analyse dienen. OLAP-Datenbanken werden deshalb vorrangig im Datawarehouse-/BI-Umfeld eingesetzt. Hier liegt der Fokus mehr auf den lesenden Zugriffen komplexer SQL-Statements. Prominente Hersteller von relationalen Datenbanken sind zum Beispiel Oracle, Microsoft, IBM oder SAP.

Vorteile relationaler SQL-Datenbanken liegen in der ACID-konformen Datenkonsistenz, der flexiblen Datenstrukturie-

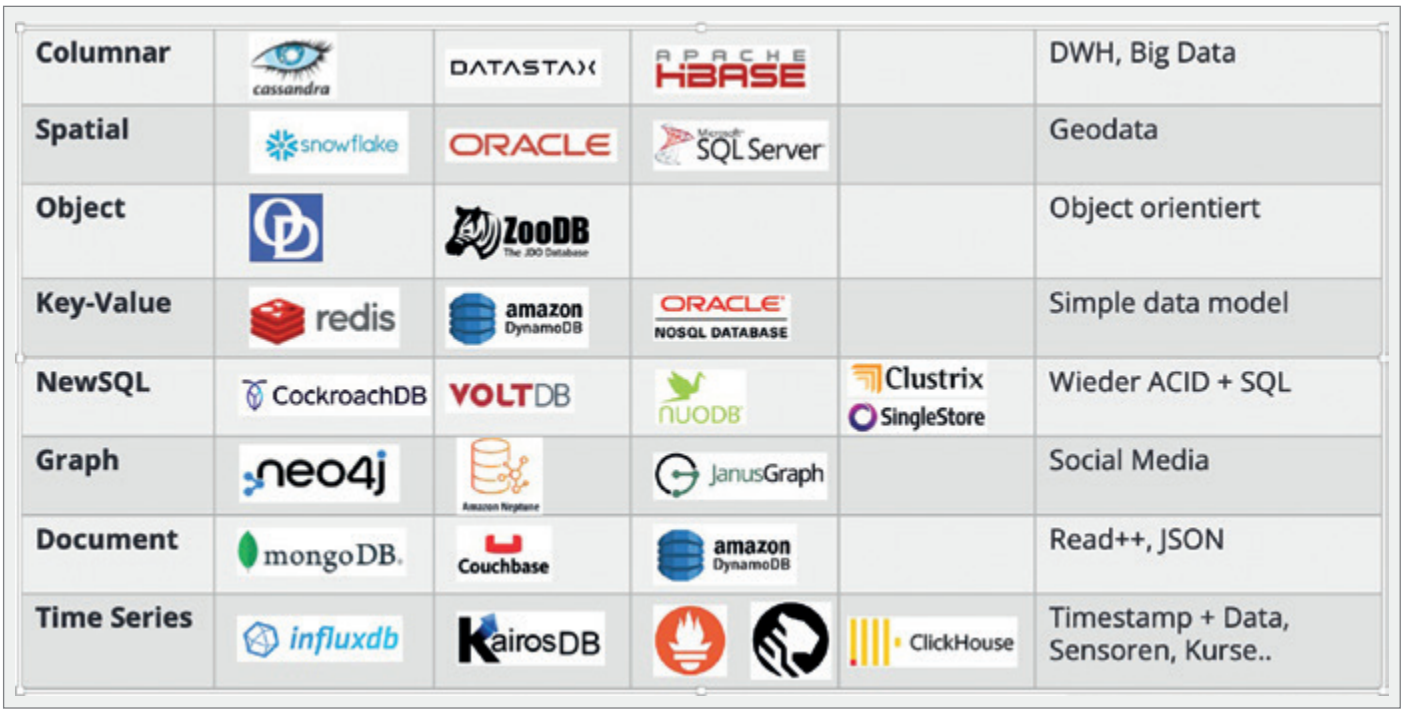


Abbildung 3: Overview NoSQL Databases (Quelle: Michael Schulze)

rung mit komplexen Datenbeziehungen, der Verwendung von SQL und der Datensicherheit. Ein klarer Nachteil, bedingt durch den Fokus auf die Transaktionssicherheit, sind fehlende horizontale Skalierungsmöglichkeiten, höherer Wartungsaufwand sowie Performanceverluste durch wachsende und komplexe Datenbestände mit vielen Objekten.

Um die Unterschiede von SQL/NoSQL noch besser zu veranschaulichen habe ich ein Create-Read-Update-Delete (CRUD)-Beispiel vorbereitet, das elementare Grundlagen beinhaltet und auf beide DB-Konzepte exemplarisch angewendet wird.

SQL-Beispiel

Im folgenden SQL-Beispiel wurde als Usecase eine Oracle-Datenbank und SQLPlus als Schnittstelle genutzt. Das Beispiel ist geläufig und wird daher nur kurz beschrieben *(siehe Abbildung 2)*.

CREATE: Grundlage für alle weiteren SQL-Befehle im Skript ist die Erstellung eines Tabellen-Objekts. Ist dieses vorhanden, können nun Datensätze durch INSERT hinzugefügt werden. Dieser Schritt muss durch ein `commit;` abgeschlossen werden, damit die Transaktion(en) festgeschrieben und die Änderungen für andere Benutzer sichtbar gemacht werden.

READ: Das Auslesen von Daten erfolgt mittels `SELECT`. Damit können Daten durch SQL-Syntax Daten gefiltert werden.

UPDATE: Hierbei wird ein vorhandener Datensatz manipuliert, auch diese Transaktion wird erst für andere sichtbar, wenn ein `commit;` erfolgt.

DELETE: Als letzter Schritt im CRUD-Beispiel erfolgt das Löschen von Daten. Zudem wird das Tabellenobjekt durch `DROP` wieder aus der Datenbank entfernt.

NoSQL-Datenbanksysteme

NoSQL-Datenbanken organisieren die Daten unstrukturiert/teilstrukturiert und haben weniger Abhängigkeiten. Das ermöglicht die Verwaltung sehr großer Datenbestände, horizontale Skalierungsmöglichkeiten (Verteilung auf mehrere Systeme) und dadurch Performancevorteile. Es gibt keine Abhängigkeiten zu Schemaobjekten und auch keine Abfragesprache wie SQL, alle notwendigen Informationen werden direkt mit der entsprechenden Datenbankaktion mitgegeben. NoSQL-Datenbanken verfolgen das BASE-Konzept, das einen starken Fokus auf Verfügbarkeit und Geschwindigkeit setzt. Durch die Anwendung der horizontalen Skalierung müssen die Daten nach Änderung auf die einzelnen Knoten verteilt werden, was einen kleinen Zeitversatz zur Folge hat. Deshalb ist hier keine Transaktionssicherheit gewährleistet, die Daten sind „eventually consistent“ und man muss softwareseitig Wege finden, um deren Konsistenz sicherzustellen. Aus diesem Grund eignen sich NoSQL-Datenbanken auch eher für einfache Datenmodelle mit vielen Daten und mit einem starken Lesefokus, zum Beispiel einem Dashboard. NoSQL-Datenbanken existieren in verschiedenen Ausprägungen wie etwa document, key-value, graph, column-store (für DWH) und vielen weiteren. Bekannte Vertreter/Produkte sind hier beispielsweise MongoDB, Cassandra, Redis, Couchbase und Oracle NoSQL.

Vorteile von NoSQL-Datenbanken liegen klar in der horizontalen Skalierung. Resultierend daraus, haben sie eine hohe Abfrageperformance auch bei wachsenden Datenbeständen, weniger Abhängigkeiten, besserer Verfügbarkeit und Flexibilität. Aufgrund der Skalierung sind NoSQL-Systeme ideal für Cloud-Umgebungen geeignet. Nachteile sind in der Inkonsistenz bei Datenabruf, der Ineffizienz bei komplexen Abfragen und der fehlenden Abfragesprache zu sehen.

Die folgende Übersicht klassifiziert aktuelle NoSQL-Datenbanksysteme hinsichtlich Ausprägung und Anwendungsfall *(siehe Abbildung 3)*.

```
test> db.test.insertMany([
... { type: "NoSQL", action: "C", descr: "CREATE"},
... { type: "NoSQL", action: "R", descr: "READ"},
... { type: "NoSQL", action: "U", descr: "UPDATE"},
... { type: "NoSQL", action: "D", descr: "DELETE"}
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6578abd3e360e063e0d33562"),
    '1': ObjectId("6578abd3e360e063e0d33563"),
    '2': ObjectId("6578abd3e360e063e0d33564"),
    '3': ObjectId("6578abd3e360e063e0d33565")
  }
}
test> db.test.find({ action: "R" })
[
  {
    _id: ObjectId("6578abd3e360e063e0d33563"),
    type: 'NoSQL',
    action: 'R',
    descr: 'READ'
  }
]
test> db.test.updateOne({ action: "U" }, {$set: { action: "update" }})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
test> db.test.deleteMany({ action: "D" })
{ acknowledged: true, deletedCount: 1 }
test> db.test.deleteMany({})
{ acknowledged: true, deletedCount: 3 }
```

Abbildung 4: mongosh-Output (Quelle: Michael Schulze)

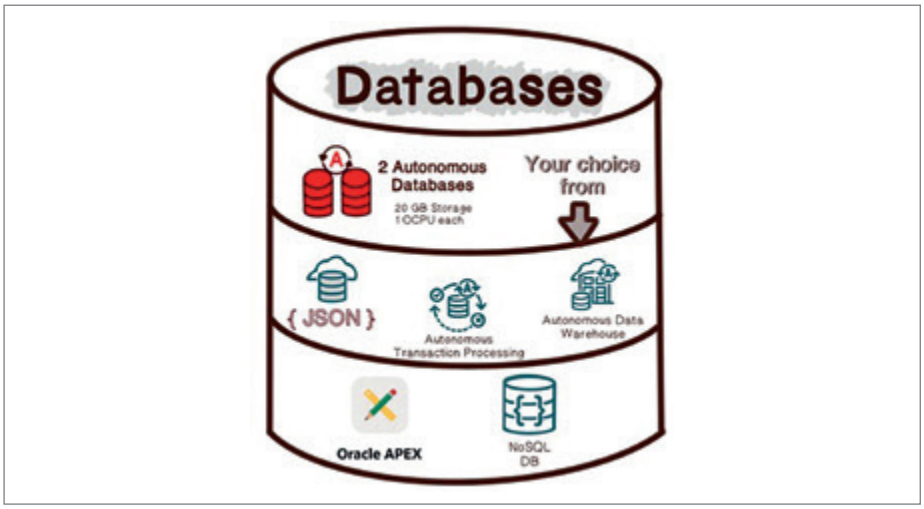


Abbildung 5: Oracle Autonomous Database (Quelle: Michael Schulze)

NoSQL-Beispiel

Es folgt in Analogie das CRUD-Beispiel in einer NoSQL-Variante (dokumentenbasiert). Als Datenbank wurde hier MongoDB (mongosh CLI) *[5]* genutzt *(siehe Abbildung 4)*.

CREATE: Wichtig ist hier zu wissen, dass es hier keine Abhängigkeiten zu Schemaobjekten gibt. Alle notwendigen Informationen werden dem Befehl direkt als JSON-String mitgegeben. `db.test.insertMany` erzeugt die Collection „test“. In dieser Datenstruktur werden die Daten gespeichert. Ein „commit“ oder ähnliches gibt es hier nicht, da NoSQL auf Verfügbarkeit (BASE) und nicht auf Transaktionssicherheit (ACID) konzipiert ist. Mit `db.test.insertOne` können einzelne Daten hinzugefügt werden. Zudem ist es möglich, eine leere Collection mit `db.createCollection("test")` zu erzeugen.

READ: Das Auslesen von Daten erfolgt mittels `db.test.find.` Damit ist eine Datenselektion möglich. Als Ergebnis werden alle passenden Daten und die entsprechende ObjectID zurückgegeben.

UPDATE: Mit `db.test.updateOne` werden Daten in der Collection „test“ aktualisiert. Als Parameter werden das Selektionskriterium und die Änderungswerte angegeben. Ferner gibt es mit `db.<collection>.updateMany` die Möglichkeit, viele Daten en bloc zu ändern.

DELETE: Als letzter Schritt im Beispiel, wird das Löschen von Daten beschrieben. Mit `db.test.deleteOne` und Parameter <ObjectID> werden Daten aus der Collection-Struktur entfernt. Ein Selektionskriterium kann an dieser Stelle nicht angegeben werden. Diese Möglichkeit gibt es aber mit `db.<collection>.deleteMany`.

NewSQL – eine Mischung aus beiden Konzepten

Zielstellungen von „NewSQL“-Lösungen sind JSON-Integration, Schnittstellen zu anderen Datenbanken, das Aufbrechen von monolithischen Strukturen relationaler Datenbanken sowie mögliche horizontale Skalierung im Cloud-Umfeld mit Transaktionssicherheit. Diese Datenbanken etablieren sich seit einiger Zeit.

Als Vertreter sind hier unter anderem die Oracle Autonomous Database *(siehe Abbildung 5)*, Microsoft Cosmos DB, Google Cloud Spanner und moderne Postgres-Ausprägungen wie Yugabyte und YDB zu nennen.

Datenbank-Cloud-Entwicklung und weiterer Ausblick

Aktuelle Datenbankumgebungen müssen immer größer werdenden Daten-

Abbildung 6: Gartner Magic Quadrant for Cloud DBMS-Systems (2023) (Quelle: Gartner)

mengen gerecht werden, skalierbar, verfügbar und kosteneffizient sein. Sie sind insgesamt auch stark von den sich ändernden Anforderungen an die Applikationen abhängig. Man erkennt im Gartner Quadrant (2023) *[6]*, dass bei Cloud-Datenbanklösungen hier zunehmend horizontal skalierbare NoSQL-Lösungen neben relationalen Datenbanken eine große Rolle spielen *(siehe Abbildung 6)*. Der Move-to-Cloud-Prozess ist in vielen Unternehmen schon vollzogen worden, beziehungsweise in naher Zukunft fest geplant. Dem Trend, dass Kunden zunehmend Multicloud-Lösungen flexibel nutzen, folgen auch die Cloud-Anbieter.

Als ein Beispiel ist der „Oracle Database Service for Azure“ zu nennen, der seit 2023 einen verwalteten DB-Service in der Oracle-Cloud für Azure-Kunden über einen direkten Interconnect der beiden Cloudlösungen ermöglicht.

Die weitere Automatisierung von Datenbanken durch Nutzung und Integration von KI-/AI-Methoden wird zukünftig zu schnelleren und effizienteren Datenbanken führen, die sich selbst monitoren und automatisch agieren.

Beispiele dafür sind die „Oracle Autonomous Database“, die schon einen Großteil von Automatismus beinhaltet. Mit der Integration von „select AI“ sind hier weitere Schritte in Richtung KI/AI geplant *[7]*.

Quellen

[1] *https://www.salesforceben.com/salesforce-history/*
[2] *https://techcrunch.com/2016/07/02/andy-jassys-brief-history-of-the-genesis-of-aws/*
[3] *https://www.apple.com/de/newsroom/2007/01/09Apple-Reinvents-the-Phone-with-iPhone/*
[4] *https://www.oracle.com/a/ocom/docs/cloud/oracle-cloud-infrastructure-platform-overview-wp.pdf*
[5] *https://www.mongodb.com/docs/manual/crud/*
[6] *https://www.gartner.com/doc/reprints?id=1-2G077EWH&ct=231221*
[7] *https://blogs.oracle.com/machine-learning/post/introducing-natural-language-to-sql-generation-on-autonomous-database*

Über den Autor

Michael Schulze ist ein Infrastrukturexperte mit mehr als 25 Jahren Berufserfahrung in Unix- und Oracle-Projekten. Sein Spektrum bei Opitz Consulting Deutschland GmbH umfasst als Solution-Architekt die Bewertung, Konzeption und Erstellung von Infrastrukturumgebungen im Datenbank- und Middleware-Umfeld. Ein starker Fokus liegt hier auf Oracle-Produkten wie der Oracle-Datenbank, HA-Lösungen, Engineered Systems, KVM-Virtualisierung, Automatisierungs- und Cloudlösungen. Als Autor für das Red Stack-Magazin und DOAG-Konferenz-Speaker liefert er regelmäßig Beiträge für die Oracle-Community in den verschiedenen Kontexten.

Michael Schulze
michael.schulze@opitz-consulting.com

In-Memory im Oracle DWH – der ultimative Performance Boost?

Stefan Raabe, C.ST Raabe

Bei einer deutschen Behörde, die für den Aufbau mehrerer Data-Warehouse- und Recherche-Umgebungen verantwortlich ist, kommt es immer mal wieder zu Performanceproblemen bei Abfragen des Oracle Analytics Servers auf den Daten dieser Umgebungen. Da diese Langläufer durch unterschiedlichste Konstellationen und Strukturen verursacht werden, ist dann teilweise ein individuelles und aufwändiges Feintuning notwendig. Im Rahmen eines Proof of Concept (PoC) wurde daher untersucht, ob durch den Einsatz der Oracle In-Memory-Option (IM) diese Probleme mit möglichst geringem Aufwand und wenigen standardisierten Vorgehensweisen behoben werden können.

Der Artikel zeigt die hierbei gewonnenen Erfahrungen am realen Beispiel des ausgewählten Data-Warehouse (DWH) mit > 1 Mrd. Fakten. Dabei werden mehrere Szenarien betrachtet, die sich auf die verschiedenen Kompressionsverfahren der IM-Option sowie auf unterschiedliche Vorgehensweisen bei der Auswahl der relevanten Tabellen/Attribute beziehen.

Ausgangsbasis

Bei der DWH-Umgebung, die für den PoC untersucht wurde, handelt es sich um einen klassischen DWH-Aufbau auf einer Oracle 19c Datenbank im Schichtenmodell mit Data Marts in (relationaler) multidimensionaler Struktur. Damit möglichst dynamisch alle Kombinationen an Abfragen auch performante Ergebnisse liefern, sind die Dimensionsschlüssel in den Faktentabellen mit Bitmap-Indizes versehen und es kommen Partitionierung sowie Materialized Views mit Query Rewrite zum Einsatz *(siehe Abbildung 1)*.

Trotzdem gibt es hin und wieder Langläufer bei den Abfragen aus dem Oracle Analytics Server. Insbesondere, da die Anwender unter anderem sehr komplexe Berichte und Dashboards erstellen, bei denen der durch den OAS generierte SQL-Code teils nicht ideal für den Optimizer ist und dieser sich für einen falschen Plan entscheidet. Der generierte SQL-Code lässt sich in diesen Fällen nur schwer beeinflussen und häufig fehlt den Anwendern hierzu auch das Know-how. Daher sollte nach den guten Erfahrungen mit In-Memory-Datenbanken in anderen Projekten geprüft werden, ob Oracle In-Memory hier auch ein Lösungsansatz für diese Probleme sein kann.

Zu beantwortende Fragestellungen

Generell sollte im Rahmen des PoC geprüft werden, ob sich ein Vorschlag zum standardisierten Einsatz von Oracle In-Memory ableiten lässt, der einen idealen Kompromiss zwischen „optimaler Nutzung" und minimalem „Verwaltungsaufwand" abbildet.

Die wichtigste Erkenntnis, die aus dem PoC gewonnen werden sollte, war daher festzustellen, welche Auswirkung es auf die Ausführungsdauer von Abfragen hat, wenn der Data Mart-Layer, oder ausgewählte Teile davon, in den In-Memory Store gelegt werden.

Aber auch der notwendige Ressourcenbedarf sollte bewertet werden, die Dauer zum Befüllen (Populate) des In-Memory Store bei Initial- und Deltaladungen sowie die möglichen Auswirkungen auf die Laufzeit der ETL-Prozesse vom Core- zum Data Mart-Layer.

Geprüfte In-Memory Szenarien

Um zu prüfen, ob es Unterschiede in den Ergebnissen gibt, abhängig von den möglichen In-Memory-Kompressionen oder der Auswahl an Tabellen/Attributen die In-Memory gelegt werden, wurden 7 verschiedene Szenarien getestet. Als Basis dienten hier die 6 Faktentabellen und 24 Dimensionen des Projektes:

1. Alle Tabellen komplett mit Standard-Compression MEMCOMPRESS FOR QUERY LOW
2. Alle Tabellen komplett mit Compression MEMCOMPRESS FOR QUERY HIGH
3. Alle Tabellen komplett mit MEMCOMPRESS FOR CAPACITY LOW
4. Alle Tabellen komplett mit MEMCOMPRESS FOR CAPACITY HIGH
5. Alle Tabellen, exklusive technischer Attribute (für ETL-Steuerung notwendig), mit MEMCOMPRESS FOR QUERY HIGH
6. Alle Tabellen, exklusive technischer Attribute und großer Textspalten, mit MEMCOMPRESS FOR QUERY HIGH
7. Nur (große) Faktentabellen, exklusive technischer Attribute, mit MEMCOMPRESS FOR QUERY HIGH

Hierfür wurde zuerst die In-Memory-Option mit 300GB RAM in der Datenbank aktiviert. Dies erfolgt einfach durch das Setzen des Parameters INMEMORY_SIZE mit Angabe der entsprechenden Größe. Da der In-Memory Column Store ein Bereich der SGA ist, muss diese im Vorfeld entsprechend erweitert werden und es sind eventuell noch Einstellungen zur HugePage-Size entsprechend der RAM-Größe vorzunehmen *[1]*. Detaillierte Informationen hierzu sind im Oracle Whitepaper *[2]* zu finden.

Um entsprechend der beschriebenen Szenarien die Tabellen dann in den In-Memory Column Store zu legen, wurden die Tabellen per ALTER TABLE-Statement angepasst und mit der gewählten Komprimierung versehen. Über die Vergabe einer Priorität wurde außerdem dafür gesorgt, dass zuerst die Faktentabellen und anschließend die Dimensionen direkt geladen wurden *(siehe Listing 1 exemplarisch für Szenario 4)*. In den Szenarien 5-7 wurden zusätzlich die auszuschließenden Attribute mit NO INMEMORY markiert *(siehe Listing 2 exemplarisch für Szenario 6)*.

Ressourcenbedarf

Beim Speicherbedarf für den In-Memory-Store konnten, je nach gewählter Kompression, für die einzelnen Tabellen Kompressionsraten zwischen 1 bis 10 erreicht werden. Ausgehend von der initialen Grö-

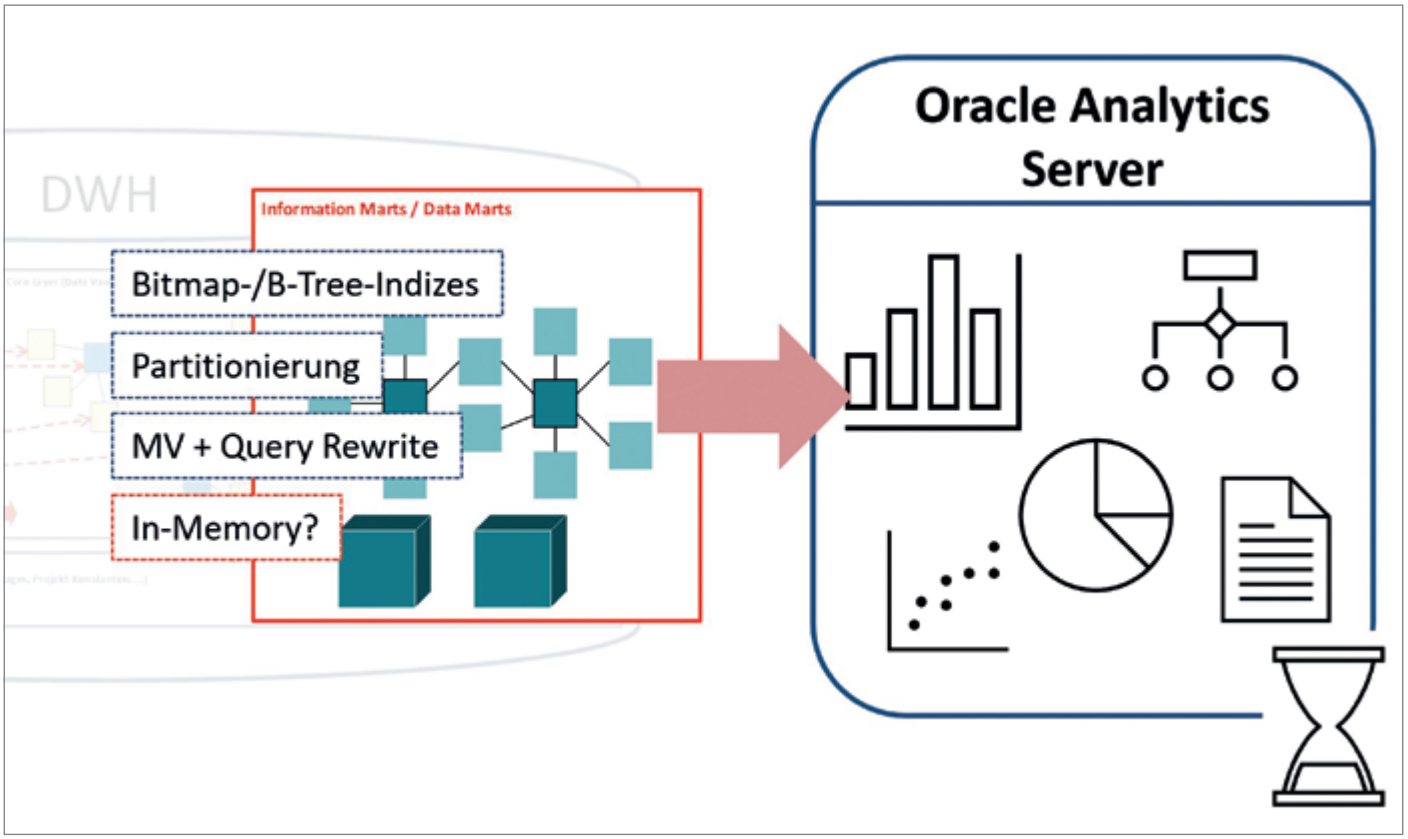


Abbildung 1: Ausgangsbasis (Quelle: Stefan Raabe)

ße sämtlicher Tabellen von circa 515GB auf Platte, war im Durchschnitt mit der Standard Compression nur knapp die Hälfte davon im RAM notwendig und mit der höchsten Kompression (FOR CAPACITY HIGH) sogar nur ungefähr ein Fünftel *(siehe Abbildung 2)*. Durch die Verringerung der gewählten Attribute oder sogar ganzer Tabellen würde sich der benötigte RAM hier noch weiter reduzieren lassen *(siehe hierzu Szenarien 5-7 im Vergleich zu Szenario 2)*.

Im Rahmen der Ressourcenbetrachtung ist außerdem erwähnenswert, dass durch den Einsatz von IM hier im Data-Mart-Layer die bisher angelegten analytischen Indizes entfallen könnten, was im aktuellen Projekt eine Einsparung von knapp 400GB an Festplattenspeicher bedeuten würde (siehe hellgrauer Balken in *Abbildung 2*).

Eine weitere Erkenntnis, die durch den PoC gewonnen werden konnte, ist, dass die standardmäßige Verwendung von 25% des gesamten IM-Speichers für den 64KB Pool, die „metadata area", viel zu hoch ist. Dieser Metadatenspeicher wird verwendet, um unter anderem für Updates das geänderte In-Memory-Abbild temporär vorzuhalten und wird dann sukzessive wieder bereinigt. In den durchgeführten Tests mit täglichen Delta-Ladungen, belegte dieser Speicherbereich nie mehr als maximal 1 Gigabyte. Die initialen crica 75GB (= 25% von 300GB) stellen so eine ziemliche Verschwendung dar und sorgten außerdem dafür, dass nur knapp 225GB für den 1MB-Pool und damit das aktuelle Speicherabbild der Daten zur Verfügung standen. Da für Szenario 1 der Speicher in dem Fall gar nicht ausgereicht hätte, ließ sich glücklicherweise die Größenverteilung über den Underscore-Parameter „_inmemory_64k_percent", nach Empfehlung von Oracle, anpassen.

Dauer des In-Memory Populate

Wie erwartet hatte die gewählte Komprimierung auch Auswirkung auf die Dauer für das initiale Laden der gesamten Daten in den Speicher. Für die gesamte Datenmenge dauerte es in der einfachsten Komprimierung (FOR QUERY LOW) circa 15 Minuten das Speicherabbild aufzu-

```
--> Fakten
ALTER TABLE F_FAKTENTABELLE1 INMEMORY MEMCOMPRESS FOR CAPACITY HIGH PRIORITY CRITICAL;
[…] 5 weitere

--> Dimensionen
ALTER TABLE D_DETAILDIMENSION1 INMEMORY MEMCOMPRESS FOR CAPACITY HIGH PRIORITY MEDIUM;
[…] 23 weitere
```

Listing 1: Laden von Dimensionen und Fakten In-Memory mit höchster Komprimierung

bauen. Für Szenario 2 waren knapp 19 Minuten notwendig und für die beiden CAPACITY-Komprimierungen (Szenario 3 & 4) jeweils ungefähr 25 Minuten. Die Ladung erfolgte hierbei 12x parallel, da dies dem Default des relevanten Parameters INMEMORY_MAX_POPULATE_SERVERS in der vorliegenden CPU-Konfiguration entsprach.

Da für diese initiale Ladung nur in seltenen Fällen ein Neuaufbau des Data Mart-Layers notwendig wäre, fallen diese Zeiten kaum ins Gewicht. Positiv: Für die tägliche Delta-basierte Ladung mit wenigen Millionen geänderten Datensätzen dauerte die Aktualisierung des Speicherabbilds dann nur wenige Sekunden und war kaum messbar. Für die bestehenden ETL-Prozesse ergibt sich daher hier keine wirkliche Verzögerung.

Allerdings gab es auch keine großen Vorteile durch den Einsatz von In-Memory bei den Laufzeiten der Ladestrecken vom Core- in den Data Mart-Layer. Denn selbst bei der täglichen Aktualisierung der Faktentabellen entschied sich der Optimizer für die Schlüssel-Lookups in die Dimensionen, die Unique-Indizes auf den Business Keys zu nutzen. Unabhängig davon, ob die Dimensionen In-Memory lagen, oder nicht. Die einzige Verbesserung der Ladezeiten, die hier zu erwarten ist, entsteht durch die größtenteils wegfallende Index-Wartung der analytischen Indizes. Im getesteten Projekt macht das aber nur einen sehr kleinen Teil der Gesamtlaufzeiten aus.

Tests der Abfrageperformance

Zum Testen der Auswirkungen von In-Memory auf die analytischen Abfragen aus dem Oracle Analytics Server (OAS) wurden verschiedene Dashboards und Einzelabfragen betrachtet. Die durchgeführten Tests wurden aus Vereinfachungsgründen auf Selektionen auf die Größte der Faktentabellen, mit ca. 1,2 Milliarden Datensätzen, sowie die 18 damit

```
--> Fakten
ALTER TABLE F_FAKTENTABELLE1 INMEMORY MEMCOMPRESS FOR QUERY HIGH PRIORITY CRITICAL NO INMEMORY(DWH_GUEL-
TIG_VON,DWH_LADE_ID,DWH_REC_SRC_SYSTEM,DWH_SRC_DELETED);
[…] 5 weitere

--> Dimensionen
ALTER TABLE D_DETAILDIMENSION1 INMEMORY MEMCOMPRESS FOR QUERY HIGH PRIORITY MEDIUM NO INMEMORY(DWH_GUEL-
TIG_VON,DWH_LADE_ID,DWH_REC_SRC_SYSTEM,DAUSPRG_SPALTEN_HASH,NOTIZEN);
ALTER TABLE D_KATEGORIESIERUNGSDIMENSION1 INMEMORY MEMCOMPRESS FOR QUERY HIGH PRIORITY MEDIUM NO INMEMO-
RY(DWH_GUELTIG_VON,DWH_LADE_ID,DWH_REC_SRC_SYSTEM);
[…] 22 weitere
```

Listing 2: Laden von Dimensionen und Fakten In-Memory unter Ausschluss bestimmter Attribute

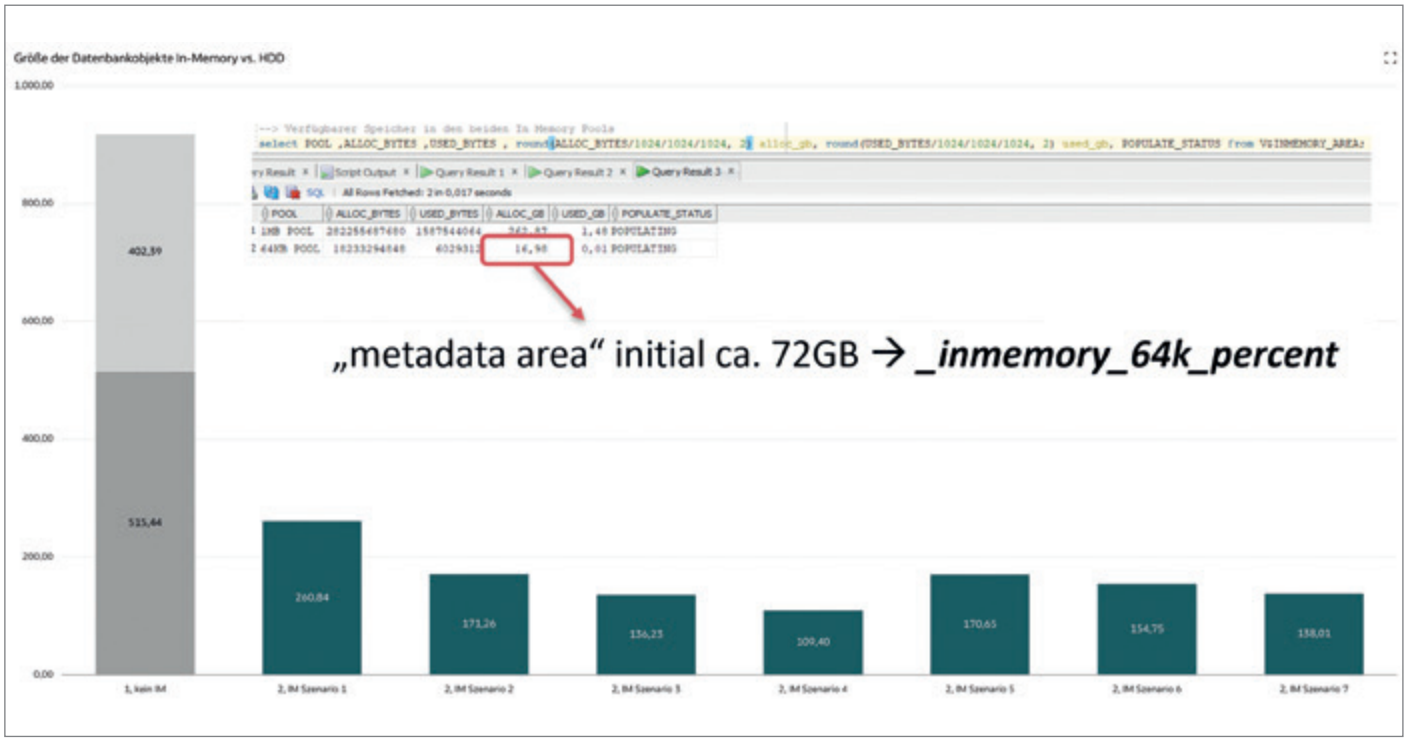


Abbildung 2: Speicherbedarf In-Memory vs. Plattenabbild + Indizes (Quelle: Stefan Raabe)

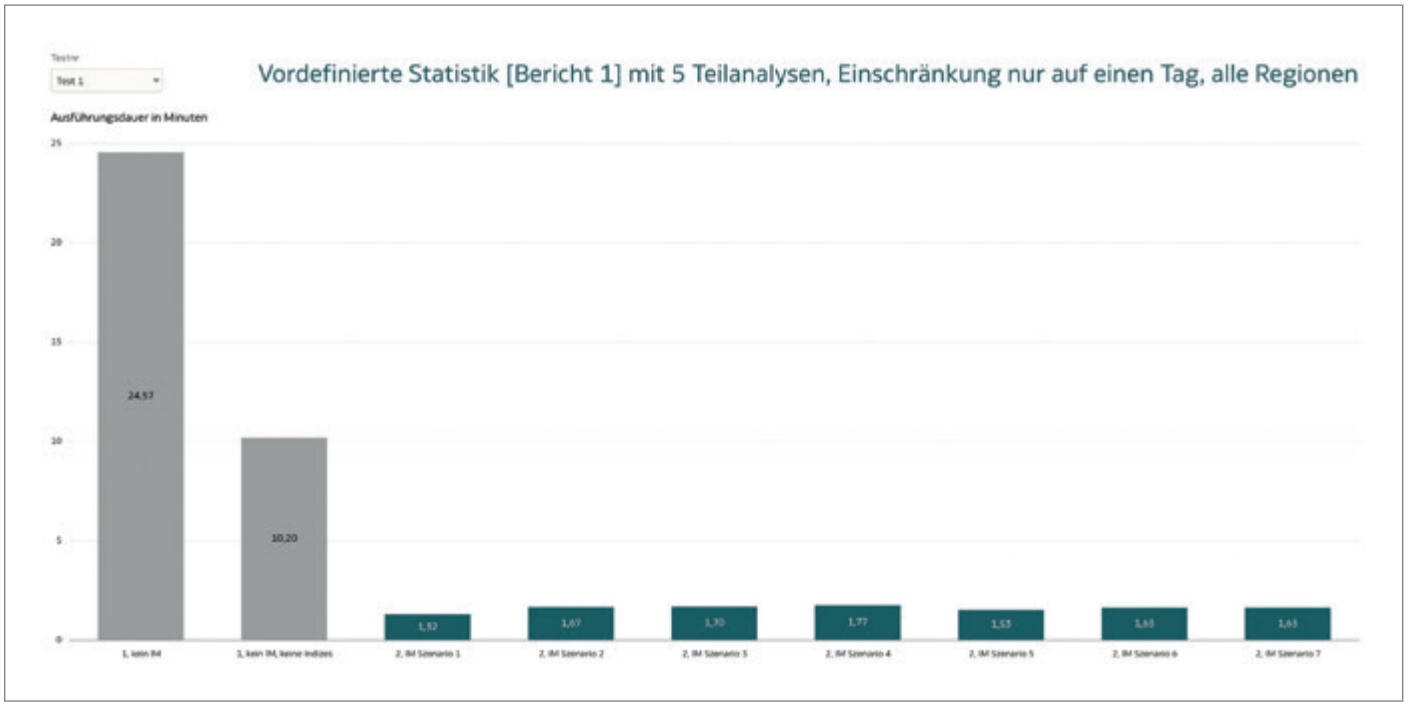


Abbildung 3: Abfragelaufzeiten Fall 1, nach Szenarien (Quelle: Stefan Raabe)

Abbildung 4: Abfragelaufzeiten Fall 2 (Quelle: Stefan Raabe)

verknüpften Dimensionen beschränkt. Vier dieser Konstellationen sollen hier exemplarisch genauer dargestellt werden.

Testfall 1 – Positivbeispiel bei Problemen mit Indizes

Im ersten Fall wurde ein bestehendes Dashboard mit 5 Teilanalysen betrachtet, bei dem es aktuell schon Probleme gab, da sich der Optimizer für eine suboptimale Index-Nutzung entschied, obwohl eine der Teilabfragen die gesamte Datenmenge der größten Faktentabelle scannen musste. Das sorgte dafür, dass im bestehenden Data Mart-Layer, inklusive analytischer Indizes, das Dashboard eine Laufzeit von knapp 25 Minuten hatte. Diese konnte durch das Entfernen der Indizes und der erzwungenen Full Table Scans (FTS), noch ohne In-Memory-Nutzung, auf etwas über 10 Minuten gesenkt werden *(siehe Balken 1 & 2 in Abbildung 3)*.

Durch den Einsatz von In-Memory ließ sich diese Dauer, je nach Szenario, dann weiter auf 1,3 bis 1,8 Minuten reduzieren. Zu sehen ist hier, dass die beste Laufzeit mit der geringsten Komprimierung erreicht wurde und mit höherem

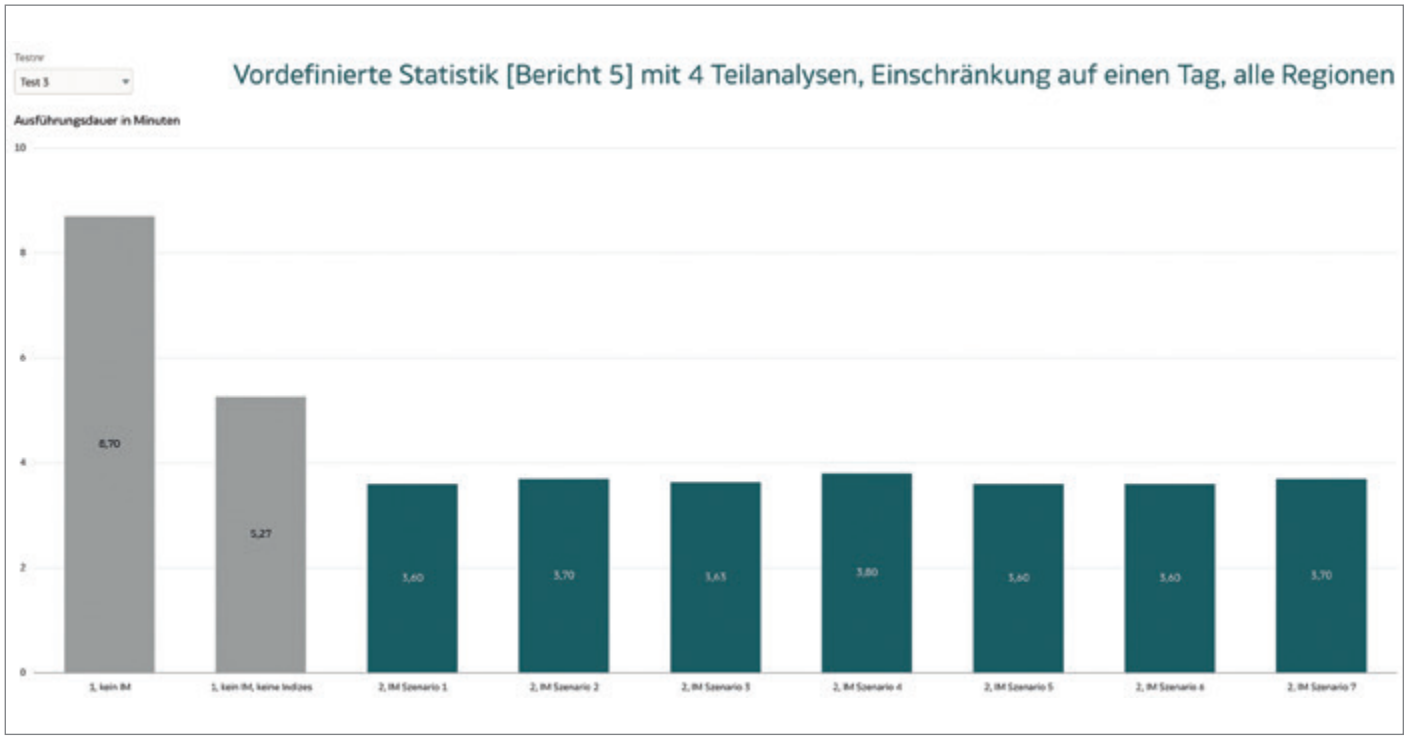
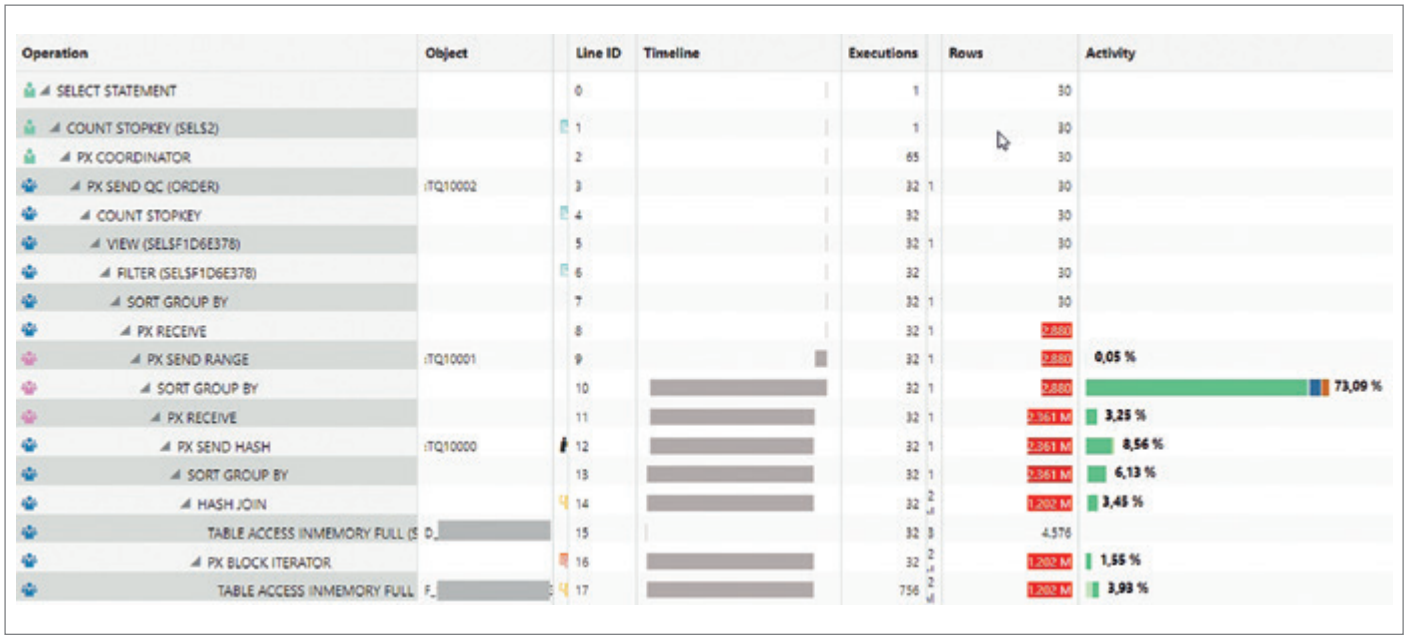


Abbildung 5: Abfragelaufzeiten Fall 3 (Quelle: Stefan Raabe)

```
WITH sawith0 AS (
    SELECT
        t791.region                              AS c2,
        COUNT(DISTINCT t2957.schluessel_dd)       AS c3,
        SUM(t2957.zuschlagbetrag)          AS c4,
        COUNT(DISTINCT
            CASE
                WHEN t2957.zuschlagbetrag > 0 THEN
                    t2957.schluessel_dd
            END
        )                                          AS c5
    FROM
        dwh_dm.d_dimension_regionen         t791,
        dwh_dm.f_faktentabelle    t2957
        WHERE ( t791.ddst_id = t2957.ddst_id )
    GROUP BY t791.region
    HAVING SUM(t2957.betragx) + COUNT(DISTINCT t2957.schluessel_dd) + COUNT(DISTINCT
            CASE WHEN t2957.zuschlagbetrag > 0 THEN t2957.schluessel_dd
            END
        ) + SUM(t2957.zuschlagbetrag) IS NOT NULL
)
SELECT
    d1.c1    AS c1,
    d1.c2    AS c2
FROM
    (SELECT DISTINCT
            0        AS c1,
            d1.c2    AS c2
     FROM sawith0 d1
        ORDER BY c2
    ) d1
WHERE ROWNUM <= 5000001
```

Listing 3: Aus dem OAS generierte Abfrage – Fall 3



Operation	Object	Line ID	Timeline	Executions	Rows	Activity
SELECT STATEMENT		0		1	30	
COUNT STOPKEY (SELS2)		1		1	30	
PX COORDINATOR		2		65	30	
PX SEND QC (ORDER)	:TQ10002	3		32	30	
COUNT STOPKEY		4		32	30	
VIEW (SELSF1D6E378)		5		32	30	
FILTER (SELSF1D6E378)		6		32	30	
SORT GROUP BY		7		32	30	
PX RECEIVE		8		32	2.880	
PX SEND RANGE	:TQ10001	9		32	2.880	0,05 %
SORT GROUP BY		10		32	2.880	73,09 %
PX RECEIVE		11		32	2.361 M	3,25 %
PX SEND HASH	:TQ10000	12		32	2.361 M	8,56 %
SORT GROUP BY		13		32	2.361 M	6,13 %
HASH JOIN		14		32	1.202 M	3,45 %
TABLE ACCESS INMEMORY FULL (S	D_	15		32	4.576	
PX BLOCK ITERATOR		16		32	1.202 M	1,55 %
TABLE ACCESS INMEMORY FULL	F_	17		756	1.202 M	3,93 %

Abbildung 6: Ausführungsplan Fall 3 (Quelle: Stefan Raabe)

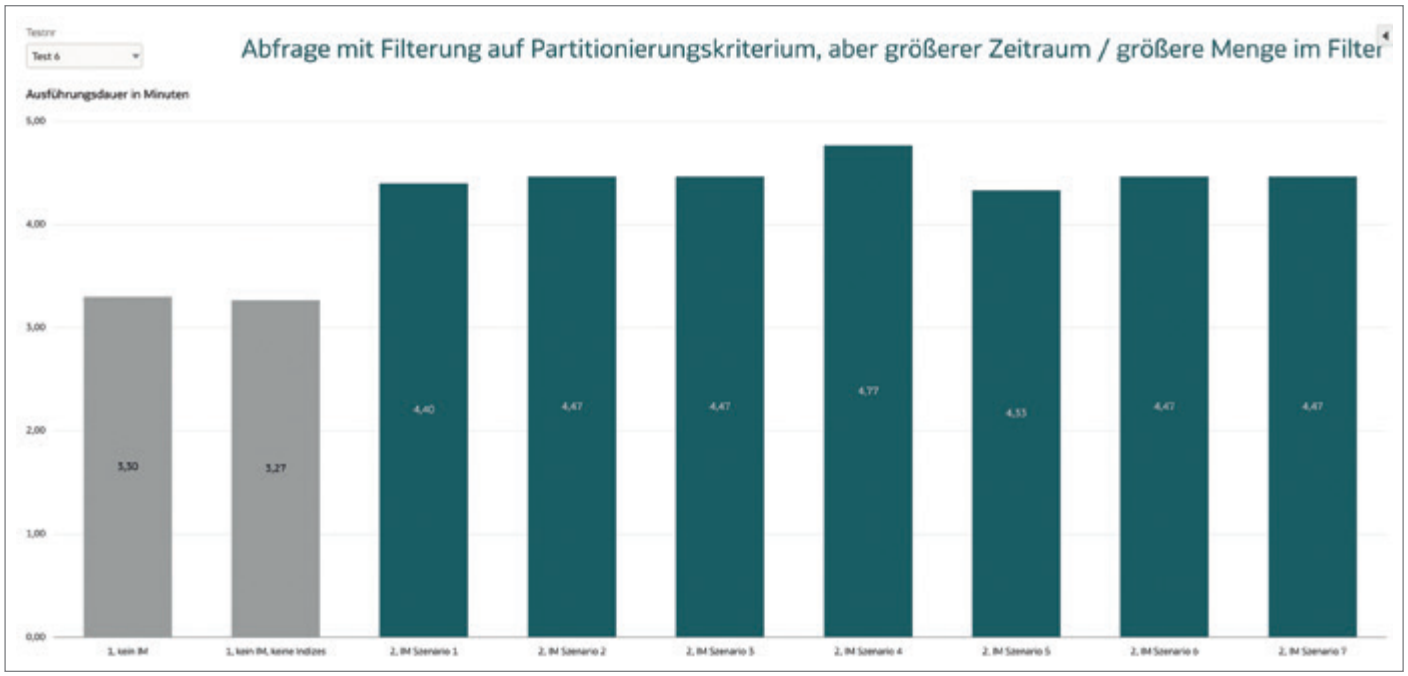


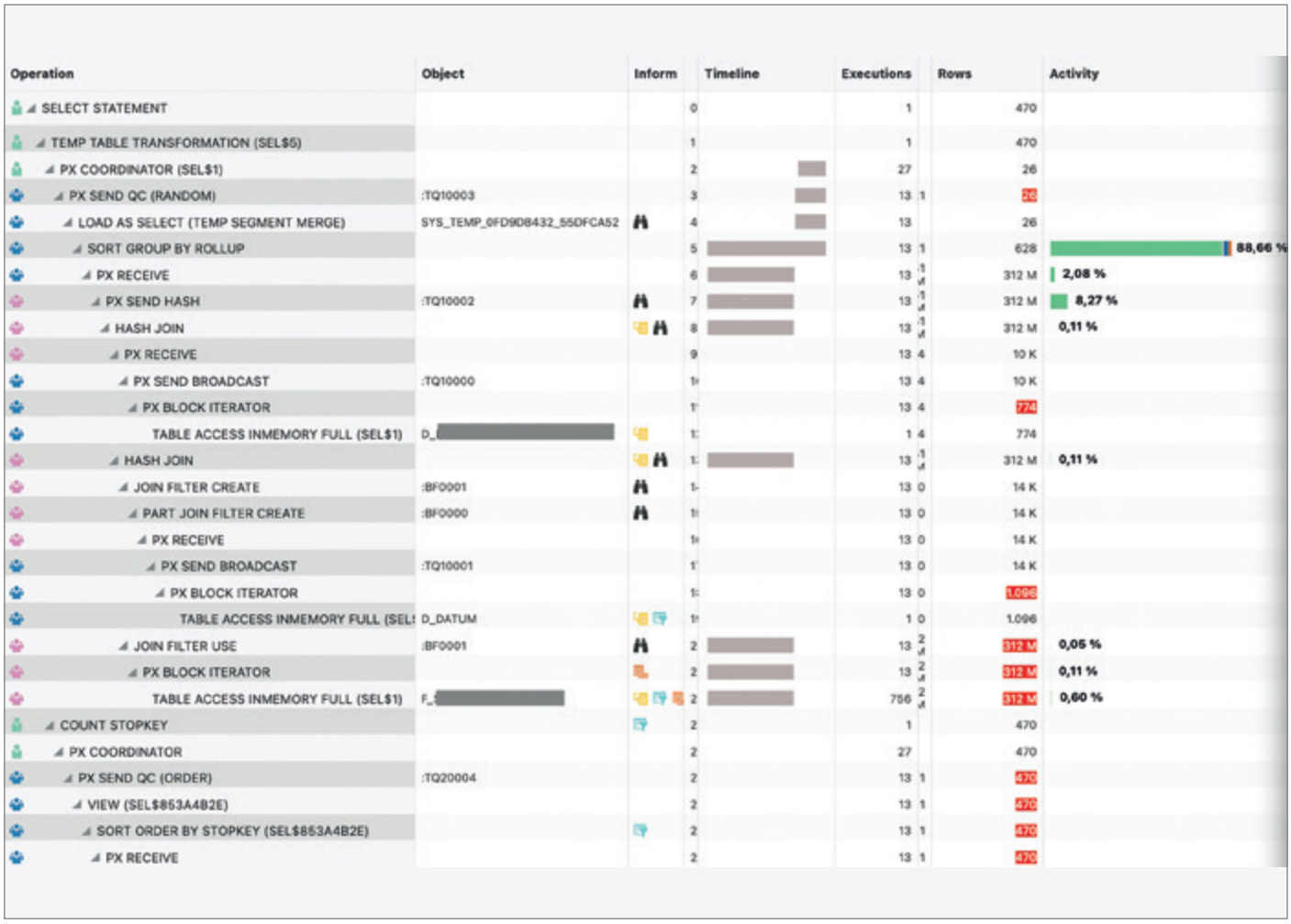
Abbildung 7: Abfragelaufzeiten Fall 4 (Quelle: Stefan Raabe)

Komprimierungsgrad die Laufzeiten leicht ansteigen. Das Ausschließen gewisser Attribute aus dem IM-Store hatte kaum Auswirkung, da die entsprechenden Attribute im vorliegenden Testfall nicht abgefragt wurden. Die Laufzeiten von Szenario 5 und 6 im Vergleich zu Szenario 2 sind daher eher generellen Schwankungen zuzuschreiben. Die Ausführungspläne für diese Szenarien waren jedenfalls identisch. Interessant ist hier, dass auch Szenario 7, bei dem nur die Faktentabellen In-Memory lagen, eine fast identische Laufzeit zu Szenario 2 aufweist. Das liegt daran, dass die hier abgefragten Dimensionen sowieso nur sehr klein waren und der Zugriff auf diese nicht ins Gewicht fiel.

Testfall 2 – weitere Optimierung gegenüber Indexnutzung

Im zweiten Fall wurde eine Einzelabfrage aus dem OAS betrachtet, die durch Einschränkungen auf mehreren Dimensionen und die Verwendung von Bitmap-Indizes und Star Transformation schon bisher recht performant lief. Die Abfrage dauerte im bestehenden DWH rund 11 Sekunden. Durch die testweise Entfernung der Indizes verschlechterte sich die Abfragezeit hier auf über 2 Minuten.

Da hier in dieser Abfrage schon durch Filterung auf den Tabellen die selektierte Datenmenge stark reduziert wurde, ließ sich durch den Einsatz von In-Memory die



Operation	Object	Inform		Timeline	Executions		Rows	Activity
SELECT STATEMENT			0		1		470	
TEMP TABLE TRANSFORMATION (SEL$5)			1		1		470	
PX COORDINATOR (SEL$1)			2		27		26	
PX SEND QC (RANDOM)	:TQ10003		3		13	1	26	
LOAD AS SELECT (TEMP SEGMENT MERGE)	SYS_TEMP_0FD9D8432_55DFCA52		4		13		26	
SORT GROUP BY ROLLUP			5		13	1	628	88,66 %
PX RECEIVE			6		13		312 M	2,08 %
PX SEND HASH	:TQ10002		7		13		312 M	8,27 %
HASH JOIN			8		13		312 M	0,11 %
PX RECEIVE			9		13	4	10 K	
PX SEND BROADCAST	:TQ10000		1		13	4	10 K	
PX BLOCK ITERATOR			1		13	4	774	
TABLE ACCESS INMEMORY FULL (SEL$1)	D_		1		1	4	774	
HASH JOIN			1		13		312 M	0,11 %
JOIN FILTER CREATE	:BF0001		1		13	0	14 K	
PART JOIN FILTER CREATE	:BF0000		1		13	0	14 K	
PX RECEIVE			1		13	0	14 K	
PX SEND BROADCAST	:TQ10001		1		13	0	14 K	
PX BLOCK ITERATOR			1		13	0	1.096	
TABLE ACCESS INMEMORY FULL (SEL	D_DATUM		1		1	0	1.096	
JOIN FILTER USE	:BF0001		2		13		312 M	0,05 %
PX BLOCK ITERATOR			2		13		312 M	0,11 %
TABLE ACCESS INMEMORY FULL (SEL$1)	F_		2		756		312 M	0,60 %
COUNT STOPKEY			2		1		470	
PX COORDINATOR			2		27		470	
PX SEND QC (ORDER)	:TQ20004		2		13	1	470	
VIEW (SEL$853A4B2E)			2		13	1	470	
SORT ORDER BY STOPKEY (SEL$853A4B2E)			2		13	1	470	
PX RECEIVE			2		13	1	470	

Abbildung 8: Ausführungsplan Fall 4 (Quelle: Stefan Raabe)

Performance noch weiter um ungefähr den Faktor 10 auf circa 1 Sekunde reduzieren *(siehe Abbildung 4)*.

Für die unterschiedlichen IM-Szenarien ließen sich hier zwar auch Laufzeitunterschiede zwischen rund 0,5 Sekunden (für Szenario 1) und 1,5 Sekunden (für Szenario 4) messen, für die reelle Nutzung dürfte das aber kaum eine Rolle spielen.

Testfall 3 – Großteil der Zeit außerhalb Datenselektion

Im dritten Fall waren die Performanceverbesserungen weniger signifikant *(siehe Abbildung 5)*. Hier handelte es sich um ein Dashboard mit 4 Teilanalysen, bei denen insbesondere eine der generierten Abfragen *(siehe anonymisiertes Listing 3)* eine Gruppierung aller Daten der Faktentabelle anhand eines Dimensionswertes, ohne weitere Filterung, darstellte.

Hier wurden also die gesamten 1,2 Milliarden Datensätze der Faktentabelle per TABLE ACCESS INMEMORY FULL gelesen und mit der entsprechenden Dimension per Hash Join verknüpft. Die Sortierung und Gruppierung der Daten erfolgten dann aber komplett außerhalb des In-Memory Store in der PGA *(siehe Abbildung 6)*. Der größte Teil der Abfrage-Laufzeit wurde daher mit dem SORT GROUP BY verbracht und dieser Teil des Ausführungsplans war in der In-Memory-Variante identisch mit der Variante ohne In-Memory. Die knapp 30% schnellere Ausführungsdauer für IM wurde also nur durch den schnelleren Full Table Scan erreicht.

Testfall 4 – Probleme mit Grouping Sets

Einen Fall, bei dem die Laufzeiten durch den Einsatz von In-Memory auf den ersten Blick sogar schlechter wurden, konnte durch eine Abfragenkonstellation auch erzeugt werden *(siehe Abbildung 7)*. Hierfür wurde im OAS eine Analyse mit Kennzahlen auf unterschiedlichen Aggregationsstufen erstellt und ein größerer Zeitraum von 3 Jahren selektiert. Das führte dazu, dass ein SQL unter Verwendung von Grouping Sets generiert wurde, mit einem zweifachen Zugriff auf die Unterabfrage aus der WITH-Klausel *(siehe anonymisiertes Listing 4)*.

Für die Ausführung dieser Abfrage erfolgte die Filterung der eingeschränkten drei Jahre zwar direkt schon beim Lesen der Faktentabelle, die Aggregation der verbliebenen ~312 Millionen Datensätze anhand der Dimensionen erfolgte dann aber wieder außerhalb des In-Memory-Store, sodass fast die gesamte Laufzeit für das SORT GROUP BY ROLLUP verwendet wurde *(siehe Abbildung 8)*. Die Ausführungspläne waren hier auch wieder fast identisch zwischen den In-Memory-Varianten und der Variante ohne In-Memory, lediglich

```
WITH
OBICOMMON0 AS (select sum(T2957.BETRAGX) as c1,
     T2871.GRUPPIERUNG1_DE as c2,
     T12214.H1L5_JAHR_ID as c3,
     T12214.H1L2_MONAT_LANG_BEZ_D as c4,
     T12214.H1L2_JAHR_MONAT_ID as c5,
     count(distinct T2957.SCHLUESSEL_DD) as c6,
     count(distinct case  when T2957.ZUSCHLAGBETRAG > 0 then T2957.SCHLUESSEL_DD end ) as c7,
     grouping_id(T2871.GRUPPIERUNG1_DE, T12214.H1L5_JAHR_ID, T12214.H1L2_JAHR_MONAT_ID, T12214.H1L2_MON-
AT_LANG_BEZ_D) as c8
from
     DWH_DM.D_DATUM T12214 ,
     DWH_DM.D_GRUPPIERUNGSKRITERIEN T2871 ,
     DWH_DM.F_STEUERFESTSETZUNG T2957
where  ( T2871.DBESTG_ID = T2957.DBESTG_ID and T2957.DDATU_ID_ENDE = T12214.DDATU_ID and (T12214.H1L5_
JAHR_ID in (2020,2021,2022)) )
group by grouping sets (
     (T12214.H1L2_MONAT_LANG_BEZ_D, T12214.H1L2_JAHR_MONAT_ID, T12214.H1L5_JAHR_ID, T2871.GRUPPIERUNG1_
DE),
     (T12214.H1L2_MONAT_LANG_BEZ_D, T2871.GRUPPIERUNG1_DE))),
SAWITH0 AS (select D1.c1 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c4 as c4,
     D1.c5 as c5,
     D1.c6 as c6,
     D1.c7 as c7
from
     (select D1.c1 as c1,
               D1.c2 as c2,
               D1.c3 as c3,
               D1.c4 as c4,
               D1.c5 as c5,
               D2.c6 as c6,
               D2.c7 as c7,
               ROW_NUMBER() OVER (PARTITION BY D1.c2, D1.c3, D1.c4, D1.c5 ORDER BY D1.c2 ASC, D1.c3 ASC,
D1.c4 ASC, D1.c5 ASC) as c8
          from

                    OBICOMMON0 D1 inner join
                    OBICOMMON0 D2 On D2.c2 = D1.c2 and D2.c4 = D1.c4
          where  ( D1.c8 = 0 and D2.c8 = 6 )
     ) D1
where  ( D1.c8 = 1 ) )
select  D1.c1 as c1, D1.c2 as c2, D1.c3 as c3, D1.c4 as c4, D1.c5 as c5, D1.c6 as c6, D1.c7 as c7, D1.c8
as c8 from ( select distinct 0 as c1,
     D1.c2 as c2,
     D1.c3 as c3,
     D1.c4 as c4,
     D1.c5 as c5,
     D1.c6 as c6,
     D1.c7 as c7,
     D1.c1 as c8
from
     SAWITH0 D1
order by c2, c3, c5 ) D1 where rownum <= 5000001;
```

Listing 4: Aus dem OAS generierte Abfrage Fall 4

die Full Table Scans erfolgten als TABLE ACCESS INMEMORY FULL vs. TABLE ACCESS FULL.

Das nun die Ausführungen über In-Memory langsamer waren, lag daher an einem ganz anderen Effekt. In der für den PoC eingesetzten Datenbank war der Parameter PARALLEL_DEGREE_POLICY auf ADAPTIVE gesetzt, so dass für die Abfragen automatisch der DOP (Parallelisierungsgrad) gesetzt wurde. Und hier hatte sich der Optimizer dafür entschieden bei der In-Memory-Abfrage einen DOP von 13 zu verwenden *(siehe „Executions" in Abbildung 8)* und bei der Variante ohne In-Memory einen DOP von 32. Dadurch war zwar hier der Full Table Scan um einiges langsamer, aber dafür durch den höheren Grad an Parallelisierung das SORT GROUP BY ROLLUP wesentlich schneller. Wenn für beide Varianten der gleiche Grad der Parallelisierung festgesetzt wurde, dann war auch hier die In-Memory-Variante etwas schneller. Ähnlich wie bei Testfall 3.

Grundsätzlich lässt sich jedoch festhalten, dass die Abfrage über Grouping Sets aktuell noch zu einem Plan führt, der nicht ideal für Oracle In-Memory ist. So wäre es beispielsweise durch ein Umschreiben der Abfrage in zwei separate WITH-Klauseln, je eine pro Aggregationslevel, dem Optimizer möglich auch In-Memory Aggregation (IMA) über KEY VECTOR USE und VECTOR GROUP BY zu nutzen *(für Details zu IMA siehe [3])*. Dadurch würde hier die Abfragedauer auf knapp 1 Minute reduziert werden.

Fazit

Der PoC hat gezeigt, dass in analytischen Umgebungen der Einsatz von Oracle In-Memory schon in der einfachsten Ausprägung sehr sinnvoll ist. Es reichte bereits, die Faktentabellen und größeren Dimensionen In-Memory zu legen, um für alle getesteten Abfragen eine bessere Performance zu erreichen. Die Wahl der Komprimierung hatte dabei nur wenig Einfluss auf die Abfragedauer, aber dafür durchaus auf die benötigten Ressourcen.

Der nachträgliche Einsatz in bestehenden Umgebungen ist auch ohne weitere Anpassungen an den Architekturen möglich, lediglich die analytischen Indizes könnten entfernt werden. Dadurch reduziert sich der Aufwand für Detailtunings, was auch ideal in Kombination mit dem Oracle Analytics Server ist, da die hier generierten SQL-Abfragen nicht so einfach beeinflusst werden können.

Mit Oracle 23c soll es weitere Optimierungen für In-Memory geben, unter anderem auch für Group By Aggregationen *(siehe [4], S. 49f.)*. Damit könnten dann hoffnungsweise auch Fälle, wie in Beispiel 3 und 4 gezeigt, weiter verbessert werden.

Quellen

[1] Johannes Ahrens (2016): 7 easy steps to configure HugePages for your Oracle Database Server, *https://www.carajandb.com/en/blog/2016/7-easy-steps-to-configure-hugepages-for-your-oracle-database-server/*, zuletzt abgerufen am 28.02.2023.

[2] Oracle (2019): Oracle Database In-Memory with Oracle Database 19c, *https://www.oracle.com/a/tech/docs/twp-oracle-database-in-memory-19c.pdf*, zuletzt abgerufen am 28.02.2023.

[3] Oracle (2015): Oracle Database In-Memory: In-Memory Aggregation, *https://www.oracle.com/technetwork/database/bi-datawarehousing/inmemory-aggregation-twp-01282015-2412192.pdf*, zuletzt abgerufen am 01.12.2023.

[4] Oracle (2023): Oracle Database New Features – Release 23c, *https://docs.oracle.com/en/database/oracle/oracle-database/23/nfcoa/oracle-database-23c-new-features-guide.pdf*, zuletzt abgerufen am 10.12.2023.

Über den Autor

Stefan Raabe arbeitet als selbstständiger Berater im DWH/BI-Umfeld. Seit über 15 Jahren unterstützt er Kunden aus allen Branchen bei der Konzeption und Umsetzung von Data-Warehouse-Architekturen und Datenintegrationen sowie dem Coaching von internen Mitarbeitenden, vorzugsweise in Projekten mit agilen Vorgehensweisen. Außerdem ist er als Dozent für Oracle-Schulungen und als Referent auf Konferenzen tätig.

Stefan Raabe
consulting@st-raabe.de

Oracle Database 23c – bigger, better, faster, stronger, Teil 1: SQL

Matthias Schulz, Schulz IT Services

Die Oracle-Datenbank 23c ist mit über 300 neuen Features eines der umfangreichsten und spannendsten Datenbank-Releases seit langem. Begleiten Sie mich in dieser Artikelserie auf eine Reise zu den großen Neuheiten und vielen kleinen, aber feinen Verbesserungen.

Teil 1 dieser Artikelserie trägt den Titel „SQL“ und führt durch eine Vielzahl von, zum Teil langerwarteten, Verbesserungen rund um SQL.

Durch die Verfügbarkeit der „Oracle Database 23c free“ *(siehe unten)*, können alle in dieser Artikelserie vorgestellten Features sofort ausprobiert werden.

Oracle 23c – App Simple

Oracle 23c ist die nächste Long-Term-Version der Oracle-Datenbank mit über 300 neuen Features *(siehe Abbildung 1)*. Für alle Nutzer der aktuellen Long-Term-Version Oracle 19c kommen auch noch alle Features aus der Version Oracle 21c hinzu!

Der Titel „App Simple“ gibt die Richtung vor und wird mehr als erfüllt – die Anwendungsentwicklung wird mit Oracle 23c deutlich erleichtert.

Laut Oracle liegen die Schwerpunkte der Version 23c auf

- JSON,
- Graph,
- Microservices,
- und Developer Productivity.

Upgrade

Von den Datenbankversionen 19c und 21c ist ein direktes Upgrade zu Version 23c möglich, ältere Datenbankversionen müssen zunächst auf die Version 19c oder 21c gebracht werden *(siehe Abbildung 2)*.

Oracle 23c free

Bei den bisherigen Versionen der Oracle-Datenbank gab es, irgendwann nach der jeweiligen Veröffentlichung, einmalig

eine kleinere kostenlose Version der Datenbank namens „Oracle XE".

Diese Vorgehensweise wurde komplett überarbeitet und so gibt es bei Oracle 23c, neben dem Beta-Programm, nun auch eine neue kostenlose Version namens „Oracle 23c free", welche von Beginn an verfügbar ist und auf dem aktuellen Stand gehalten wird. Zum Zeitpunkt der Erstellung dieses Artikels wurde bereits die Version 23.3.0.0 der „free"-Version veröffentlicht *(https://www.oracle.com/database/free/)*.

Oracle 23c free – Docker Image

Das jeweils neueste Docker Image der Oracle 23c free kann direkt mit dem folgendem Docker-Befehl geladen werden:

docker pull container-registry.oracle.com/database/free:latest

Teil 1: SQL

Teil eins dieser Artikelserie zu Oracle 23c befasst sich mit den wichtigsten Neuerungen rund um SQL:

- SELECT ohne FROM
- GROUP BY Alias
- Interval-Aggregation
- Table Value Constructor
- Direct Joins für Update und Delete
- Returning-Clause bei Update und Merge
- IF EXISTS – IF NOT EXISTS
- BOOLEAN Datatype
- 4096 Spalten
- Annotations
- SQL Domains
- Schema Level Privileges

SELECT ohne FROM

Um eine Berechnung auszuführen, beispielsweise „heute in zwei Wochen" oder, um einfach eine Zeile aus dem Nichts zu erhalten, musste in Oracle SQL bisher immer die Pseudotabelle DUAL verwendet werden:

```
SELECT SYSDATE + 14 FROM DUAL;
```

Mit Oracle 23c entfällt die Notwendigkeit des „FROM DUAL"-Ausdrucks und man kann nun wie bei PostgreSQL, Microsoft SQL Server und anderen Datenbanken einfach den folgenden Ausdruck verwenden:

```
SELECT SYSDATE + 14;
```

Natürlich ist es nach wie vor möglich „FROM DUAL" zu verwenden, so dass bestehender Code nicht angepasst werden muss.

GROUP BY Alias

Soll in einer Abfrage ein komplexer Term, wie beispielsweise eine längere Case-Anweisung, nicht nur ausgegeben, sondern auch zur Aggregation (GROUP BY) und vielleicht auch noch zur Filterung dieser Aggregation (HAVING) verwendet werden, so musste man den gleichen Term bis zu dreimal schreiben. Nur bei der Sortierung der Ergebniszeilen (ORDER BY) konnte bereits der Alias des Terms verwendet werden.

Mit Oracle 23c kann der Alias des Terms nun auch in der Group-By- und der Having-Clause verwendet werden. Damit werden SQL-Abfragen deutlich kürzer und leichter zu lesen und zu warten *(siehe Listing 1)*.

Interval-Aggregation

Summen und Durchschnittswerte für Zeitintervalle konnten bisher nicht mittels SQL-Aggregation gebildet werden – ab Oracle 23c ist dies nun endlich möglich *(siehe Listing 2)*.

Table Value Constructor

Insert-Befehle waren bis jetzt immer mit einer Menge redundantem Code verbunden. In jeder Zeile wiederholen sich die Tabelle und die Namen der Zielspalten. Zudem verursachen sie jede Menge Overhead, da jeder Befehl einzeln an die Datenbank gesendet und dort geparsed und verarbeitet werden muss.

Mit dem neuen Table-Value-Constructor von Oracle 23c lassen sich Insert-Statements redundanzfrei und mit deutlich weniger Code erstellen und durch die Bündelung vieler Wertetupel auch sehr viel schneller und effizienter verarbeiten *(siehe Listing 3)*.

Der Table-Value-Constructor kann auch dazu verwendet werden, eine Da-

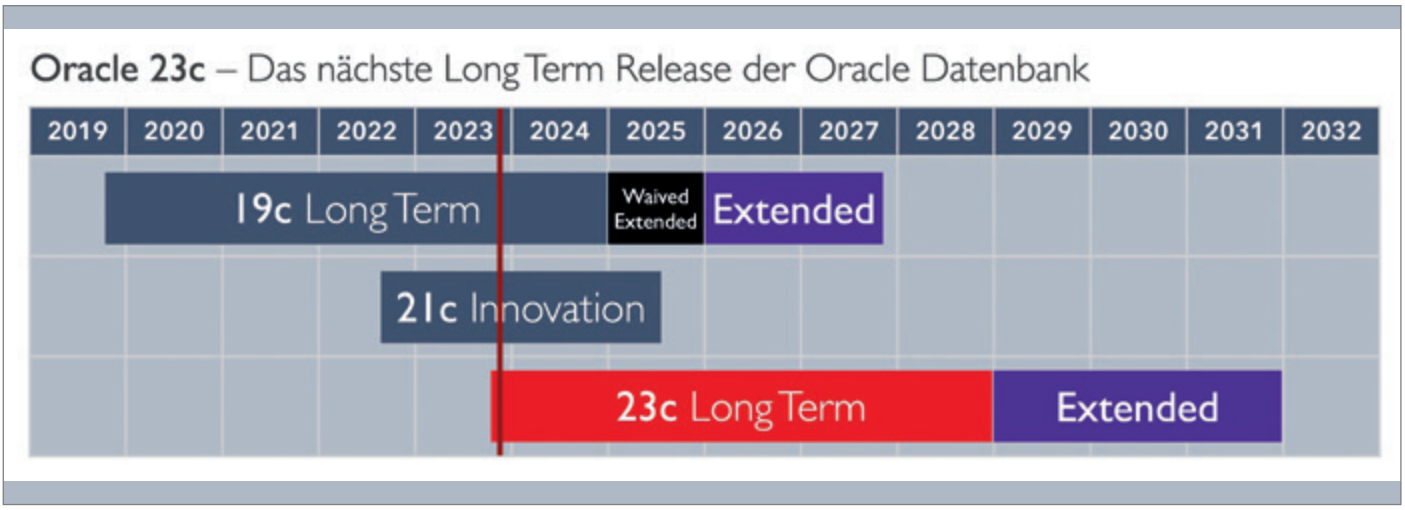


Abbildung 1: Oracle 23c – Long Term Release (Quelle: Matthias Schulz)

```
-- bisher: ------------------------

   SELECT CASE WHEN o.OBJECT_TYPE LIKE 'TABLE%PARTITION' THEN 'TABLE PARTITIONS'
               WHEN o.OBJECT_TYPE LIKE 'INDEX%PARTITION' THEN 'INDEX PARTITIONS'
               ELSE o.OBJECT_TYPE
          END AS PARTTIONS_OR_OBJECTS
     FROM ALL_OBJECTS o
 GROUP BY CASE WHEN o.OBJECT_TYPE LIKE 'TABLE%PARTITION' THEN 'TABLE PARTITIONS'
               WHEN o.OBJECT_TYPE LIKE 'INDEX%PARTITION' THEN 'INDEX PARTITIONS'
               ELSE o.OBJECT_TYPE
          END
   HAVING CASE WHEN o.OBJECT_TYPE LIKE 'TABLE%PARTITION' THEN 'TABLE PARTITIONS'
               WHEN o.OBJECT_TYPE LIKE 'INDEX%PARTITION' THEN 'INDEX PARTITIONS'
               ELSE o.OBJECT_TYPE
          END LIKE '%PARTITIONS'
 ORDER BY PARTTIONS_OR_OBJECTS;

-- ab Oracle 23c: -----------------

   SELECT CASE WHEN o.OBJECT_TYPE LIKE 'TABLE%PARTITION' THEN 'TABLE PARTITIONS'
               WHEN o.OBJECT_TYPE LIKE 'INDEX%PARTITION' THEN 'INDEX PARTITIONS'
               ELSE o.OBJECT_TYPE
          END AS PARTTIONS_OR_OBJECTS
     FROM ALL_OBJECTS o
 GROUP BY PARTTIONS_OR_OBJECTS
   HAVING PARTTIONS_OR_OBJECTS LIKE '%PARTITIONS'
 ORDER BY PARTTIONS_OR_OBJECTS;
```

Listing 1: GROUP BY Alias

```
CREATE TABLE ACTIVITY (
  ID         NUMBER,
  START_TIME TIMESTAMP,
  DURATION   INTERVAL DAY TO SECOND
);

-- bisher: ------------------------

SELECT avg(d.duration) AS AVERAGE_DURATION,
       sum(d.duration) AS TOTAL_DURATION
  FROM ACTIVITY d;

ORA-00932: inconsistent datatypes: expected NUMBER got INTERVAL DAY TO SECOND

-- ab Oracle 23c: -----------------

SELECT avg(d.duration) AS AVERAGE_DURATION,
       sum(d.duration) AS TOTAL_DURATION
  FROM ACTIVITY d;

AVERAGE_DURATION       TOTAL_DURATION
+00 00:23:20.000000    +00 01:10:00.000000
```

Listing 2: Interval-Aggregation

tenmenge zu generieren und erspart somit die altbekannte Kette aus „UNION ALL"-Statements *(siehe Listing 4)*.

Last but not least, kann der Table-Value-Constructor auch in Form einer With-Clause zum Einsatz kommen *(siehe Listing 5)*.

Direct Joins für Update und Delete

Während bei Abfragen und Merge-Statements Joins zu anderen Tabellen möglich sind, war dies bei Update- und Delete-Befehle bisher nicht möglich.

Mit den neuen Direct Joins in Oracle 23c können nun auch Update und Delete durch das Hinzufügen einer FROM-Clause mit anderen Tabellen verbunden werden, um Datensätze für das Löschen oder Aktualisieren unmittelbar nutzen zu können *(siehe Listing 6)*.

```
-- bisher: ------------------------

INSERT INTO PERSONS (ID, NAME, CITY) VALUES ('1', 'Bader', 'Berlin');
INSERT INTO PERSONS (ID, NAME, CITY) VALUES ('2', 'Fuchs', 'Frankfurt');
INSERT INTO PERSONS (ID, NAME, CITY) VALUES ('3', 'Mayer', 'München');

-- ab Oracle 23c: -----------------

INSERT INTO PERSONS (ID,   NAME,   CITY)
             VALUES ('1', 'Bader', 'Berlin'),
                    ('2', 'Fuchs', 'Frankfurt'),
                    ('3', 'Mayer', 'München');
```

Listing 3: Table-Value-Constructor – Insert

```
-- bisher: ------------------------

SELECT 1 AS ID, 'Bader' AS NAME, 'Berlin'    AS CITY FROM DUAL UNION ALL
SELECT 2 AS ID, 'Fuchs' AS NAME, 'Frankfurt' AS CITY FROM DUAL UNION ALL
SELECT 3 AS ID, 'Mayer' AS NAME, 'München'   AS CITY FROM DUAL;

-- ab Oracle 23c: -----------------

SELECT t.*
  FROM (VALUES (1, 'Bader', 'Berlin'),
               (2, 'Fuchs', 'Frankfurt'),
               (3, 'Mayer', 'München')
       ) AS t (ID, NAME, CITY);
```

Listing 4: Table-Value-Constructor – Select

```
-- bisher: ------------------------

WITH teilnehmer AS (
    SELECT 1 AS ID, 'Bader' AS NAME, 'Berlin'    AS CITY FROM DUAL UNION ALL
    SELECT 2 AS ID, 'Fuchs' AS NAME, 'Frankfurt' AS CITY FROM DUAL UNION ALL
    SELECT 3 AS ID, 'Mayer' AS NAME, 'München'   AS CITY FROM DUAL
)
SELECT * FROM teilnehmer;

-- ab Oracle 23c: -----------------

WITH teilnehmer(ID, NAME, CITY) AS (
        VALUES (1, 'Bader', 'Berlin'),
               (2, 'Fuchs', 'Frankfurt'),
               (3, 'Mayer', 'München')
)
SELECT * FROM teilnehmer;
```

Listing 5: Table-Value-Constructor – With-Clause

```
UPDATE PERSON p
   SET p.birthday = s.birthday
  FROM STAGE s
 WHERE s.id = p.id;

DELETE PERSON p
  FROM STAGE s
 WHERE s.id = p.id
   AND s.end_of_contract < sysdate;
```

Listing 6: Direct Joins für Update und Delete

Returning-Clause bei Update und Merge

Update- und Merge-Statements haben ab der Version 23c eine Returning-Clause zur Rückgabe von Werten, die auch Bulk-fähig ist.

Bei der Wertrückgabe kann sowohl der bisherige als auch der aktualisierte Wert einer Spalte zurückgegeben werden. Mit „OLD“ wird der Wert vor dem Update zurückgegeben, mit „NEW“ der Wert nach dem Update *(siehe Listing 7)*.

IF EXISTS – IF NOT EXISTS

Soll eine Tabelle nur erstellt werden, falls diese noch nicht existiert, oder nur gelöscht werden, wenn sie noch da ist, hatte man bisher zwei Optionen zur Wahl. Man konnte entweder mit der Fehlermeldung leben oder musste den Create-, beziehungsweise Drop-Befehl, in ein kleines Programm, mit entsprechendem Fehlerhandling, einbauen.

Oracle 23c bringt zwei langersehnte Syntaxerweiterungen für DDL-Befehle:

- IF EXISTS
- IF NOT EXISTS

IF NOT EXISTS erstellt ein Objekt nur dann, wenn es nicht existiert, **IF EXISTS** löscht es nur dann, wenn es vorhanden ist.

Die Antwort der beiden Befehle ist immer die gleiche, egal ob sie nun etwas zu tun hatten oder nicht. Dies vereinfacht die Prozesse im CI/CD-Umfeld, birgt aber auch gewisse Risiken. Weicht die Struktur, der neu zu erstellenden Tabelle gleichen Namens, von der bereits existierenden ab, wird der Create-Table-Befehl trotzdem die erfolgreiche Erstellung der Tabelle melden, ohne jedoch die Unterschiede beachtet zu haben *(siehe Listing 8 und 9)*.

BOOLEAN Datatype

Im Gegensatz zu PL/SQL und den meisten anderen Programmiersprachen, hatte die Oracle-Datenbank nie einen nativen Boolean-Datentyp für Wahrheitswerte in Tabellen und SQL-Befehlen.

Als Workaround wurden meist Text- oder Zahlenspalten mit einer Vielzahl von Vorstellungen verwendet, welcher Wert nun für Wahr (true) und welcher für Falsch (false) steht.

Beispiele für „Wahr“ (true):

- 1, -1
- 'Y', 'y'
- 'Yes', 'YES', 'yes'
- 'True', 'TRUE', 'true'
- ...

Beispiele für „Falsch“" (false):

- 0
- 'N', 'n'
- 'No', 'NO', 'no'
- 'False', 'FALSE', 'false'
- ...

Um den Wildwuchs verwendeter Werte im Zaum zu halten, konnten Check-Constraints verwendet werden, aber auch dies konnte unterschiedliche Implemen-

Abbildung 2: Upgrade-Pfade (Quelle: Matthias Schulz)

```
UPDATE employee
       SET salary = salary +1000
     WHERE id = 416
 RETURNING OLD salary,  NEW salary
      INTO :old_salary, :new_salary;
```

Listing 7: Returning-Clause bei Update

```
-- bisher: ------------------------

CREATE TABLE MYTABLE(ID NUMBER);
--> Table created

CREATE TABLE MYTABLE(ID NUMBER);
--> ORA-00955: name is already used by an existing object

-- ab Oracle 23c: -----------------

CREATE TABLE IF NOT EXISTS MYTABLE(ID NUMBER);
--> Table created

CREATE TABLE IF NOT EXISTS MYTABLE(ID NUMBER);
--> Table created (obwohl nicht passiert ist)
```

Listing 8: Table-Value-Constructor – IF EXISTS

```
-- bisher: ------------------------

DROP TABLE MYTABLE;
--> Table dropped

DROP TABLE MYTABLE;
--> ORA-00942: table or view does not exist

-- ab Oracle 23c: -----------------

DROP TABLE IF EXISTS MYTABLE;
--> Table dropped

DROP TABLE IF EXISTS MYTABLE;
--> Table dropped (obwohl nicht passiert ist)
```

Listing 9: Table-Value-Constructor – IF NOT EXISTS

```
-- bisher: ------------------------

CREATE TABLE MY_TABLE(
  BOOL VARCHAR2(16) CHECK(BOOL IN ('TRUE', 'FALSE'))
);

SELECT *
  FROM MY_TABLE
 WHERE BOOL = 'TRUE';

SELECT CASE WHEN BOOL = 'FALSE' THEN 'Falsch'
            ELSE 'Richtig'
       END
  FROM MY_TABLE;

-- ab Oracle 23c: -----------------

CREATE TABLE MY_TABLE(
  BOOL BOOLEAN
);

SELECT *
  FROM MY_TABLE
 WHERE BOOL;

SELECT CASE WHEN NOT BOOL THEN 'Falsch'
            ELSE 'Richtig'
       END
  FROM MY_TABLE;
```

Listing 10: Boolean Datatype

```
CREATE TABLE T_4096_COLUMNS(
    COL_1    NUMBER,
    COL_2    NUMBER,
    COL_3    NUMBER,
    COL_4    number,
    COL_5    NUMBER,
    ...
    COL_4094 NUMBER,
    COL_4095 NUMBER,
    COL_4096 NUMBER);
```

Listing 11: Tabellen mit 4096 Spalten

tierungen in verschiedenen Datenbanken nicht verhindern. Die Leittragenden waren meist die Anwendungsentwickler, die mit einem Zoo unterschiedlicher Boolean-Werte zurechtkommen mussten.

Da es sich bei allen bisherigen Workarounds um Text- oder Zahlen-Werte handelt, muss bei jeder Verwendung in Case-Statements oder Where-Clauses immer ein Vergleich geschrieben werden, welcher bei einem echten Boolean-Datentyp unnötig wäre.

Mit der Version 23c führt Oracle einen echten Boolean-Datentyp für SQL und Tabellen ein und ermöglicht damit endlich eindeutige Boolean-Werte und Boolean-Aussagen ohne Vergleiche *(siehe Listing 10)*.

Um die Migration bestehender Boolean-Alternativen zu erleichtern, akzeptiert der neue Datentyp eine Vielzahl von Text- und Zahlenwerten als „true“ oder „false“.

4096 Spalten

Bisher konnten Tabellen in Oracle-Datenbanken maximal 1000 Spalten haben, doch ab Version 23c lässt sich diese Grenze auf 4096 Spalten erhöhen.

Gesteuert wird das neue Feature über den Parameter **MAX_COLUMNS**. Mit MAX_COLUMNS = STANDARD bleibt die maximale Anzahl bei 1.000 Spalten, mit MAX_COLUMNS = EXTENDED steigt sie auf 4096 *(siehe Listing 11)*.

Annotations

Die neuen Annotations in Oracle 23c bieten eine erweiterte und umfangreichere Möglichkeit, Objekte in der Datenbank mit Metadaten zu annotieren, als dies mit den bisherigen Comments möglich war.

Objekte wie Tabellen, Tabellenspalten, Views und Indexes können direkt bei der Erstellung oder im Nachgang annotiert werden und die Annotations lassen sich auch jederzeit wieder entfernen *(siehe Listing 12, 13, und 14)*.

Ein großer Vorteil der Annotations ist ihre flexible Key-Value-Syntax und die Möglichkeit, die Annotations aller Objekte über den View USER_ANNOTATIONS_USAGE abrufen zu können *(siehe Listing 15)*.

Die neuen Annotations eröffnen erweiterte Möglichkeiten, die Metadaten

```
CREATE TABLE ACTIVITY (
  ID         NUMBER  PRIMARY KEY
             ANNOTATIONS ("Display_Name" 'Activity ID', Mandatory 'yes'),
  START_TIME TIMESTAMP NOT NULL
             ANNOTATIONS ("Display_Name" 'Start time',  Mandatory 'no')
)
ANNOTATIONS ("Display_Name" 'Activity ', "Display_Type" 'Table');
```

Listing 12: Annotations

```
SELECT * FROM USER_ANNOTATIONS;

ANNOTATION_NAME
DESCRIPTION
Display_Name
Display_Type
```

Listing 13: Annotations – View USER_ANNOTATION

```
SELECT * FROM USER_ANNOTATION_VALUES;

ANNOTATION_NAME     ANNOTATION_VALUE
Display_Name        Activity
Display_Name        Activity ID
Display_Name        Start time
Display_Type        Table
MANDATORY           no
MANDATORY           yes
```

Listing 14: Annotations – View USER_ANNOTATION_VALUES

```
SELECT object_name, object_type, column_name,
       annotation_name, annotation_value
  FROM USER_ANNOTATIONS_USAGE
 ORDER BY annotation_name, annotation_value;

OBJECT_NAME  OBJECT_TYPE  COLUMN_NAME  ANNOTATION_NAME  ANNOTATION_VALUE
ACTIVITY     TABLE        ---          Display_Name     Activity
ACTIVITY     TABLE        ID           Display_Name     Activity ID
ACTIVITY     TABLE        START_TIME   Display_Name     Start time
ACTIVITY     TABLE        ---          Display_Type     Table
ACTIVITY     TABLE        START_TIME   MANDATORY        no
ACTIVITY     TABLE        ID           MANDATORY        yes
```

Listing 15: Annotations – View USER_ANNOTATIONS_USAGE

```
CREATE DOMAIN D_YEAR AS DATE
   CONSTRAINT D_YEAR_CHK CHECK(D_YEAR = trunc(D_YEAR))
      DISPLAY to_char(D_YEAR, 'YYYY')
        ORDER to_char(D_YEAR, 'YYYY')
  ANNOTATIONS (DESCRIPTION 'Domain for year only dates');


CREATE TABLE MOVIE(
    ID     NUMBER PRIMARY KEY,
    TITLE  VARCHAR2(255),
    YEAR   D_YEAR);
```

Listing 16: SQL Domains

aller Datenbankobjekte zentral abzurufen, zu überprüfen und Dokumentationen automatisiert zu erstellen.

SQL Domains

SQL Domains *(siehe Listing 16)* sind wie eine Vorlage für eine Tabellenspalte (Single Column Domain) oder für mehrere Tabellenspalten (Multi Column Domain, Flexible Domain) und können eine beliebige Kombination folgender Elemente umfassen:

- Datatype
- Sequence-Werte
- Constraints
- Formatierung
- Sortierung
- Annotationen
- ...

Der große Vorteil von SQL-Domains besteht darin, dass die Domain nur einmal erstellt werden muss und dann für jede Tabellenspalte dieses Typs verwendet werden kann, ohne dass die eventuell sehr komplexen Details jedes Mal wieder vollständig und korrekt implementiert werden müssen.

Dies spart jede Menge Zeit, reduziert Fehler und erhöht die Qualität und Konformität des gesamten Datenmodells.

Schema Level Privileges

Um anderen Nutzern Zugriff auf die eigenen Tabellen zu geben, gab es bisher nur aufwendige oder riskante Optionen.

1. Der bequeme, aber riskante Weg, der Vergabe von **ANY- Rechten**.
2. Die sichere, aber aufwendige Option, mit einem **Skript**, alle Rechte, für alle betroffenen Objekte, bei jeder Änderung der Datenbank neu zu vergeben.

Die Schema-Level-Privileges von Oracle 23c ermöglichen es, Rechte für einen ganzen Objekttyp eines Schemas an andere User zu vergeben *(siehe Listing 17)*.

Da die Rechte für den Objekttyp eines Schemas vergeben werden, gelten diese Berechtigungen auch für allen neuen Objekte dieses Typs, die nach der Rechtevergabe erstellt werden.

```
-- bisher: ------------------------

-- a) riskante Option:
GRANT SELECT ANY TABLE TO MYUSER;

-- b) aufwendige Option:
BEGIN
    FOR rec IN (SELECT table_name
                  FROM USER_TABLES)
    LOOP
        EXECUTE IMMEDIATE 'GRANT SELECT ON '
                          || rec.table_name
                          || ' TO MYUSER';
    END LOOP;
END;
/
-- ab Oracle 23c: -----------------

GRANT SELECT ANY TABLE ON SCHEMA MYDATA TO MYUSER;

-- weitere Beispiele:

GRANT INSERT ANY TABLE ON SCHEMA MYDATA TO MYUSER;
GRANT UPDATE ANY TABLE ON SCHEMA MYDATA TO MYUSER;
GRANT DELETE ANY TABLE ON SCHEMA MYDATA TO MYUSER;

GRANT SELECT ANY SEQUENCE ON SCHEMA MYDATA TO MYUSER;

GRANT EXECUTE ANY PROCEDURE ON SCHEMA MYDATA TO MYUSER;
```

Listing 17: Schema Level Privileges

Eine pauschale Rechtevergabe mit ANY-Rechten ist zukünftig nicht mehr nötig und die komplexen Berechtigungsskripte, die nach jedem Deployment ausgeführt werden müssen, können entfallen.

Skripte

Die Skripte aus diesem Artikel werden auch auf meinem Blog veröffentlicht:

https://oracledeli.wordpress.com

Über den Autor

Matthias Schulz ist seit 2002 Geschäftsführer der Schulz IT Services GmbH und als Consultant, Trainer und Konferenzsprecher im Datenbankumfeld tätig. Er ist seit 2016 an der Dualen Hochschule Baden-Württemberg Dozent für Informatik (Datenbanken).

Tätigkeitsschwerpunkte sind Data Vault 2.0, High-End-Systeme (Exadata, Exasol), Oracle Cloud und die Oracle-Datenbank (Entwicklung, Tuning, Architektur, ETL, Testing und Java in der Datenbank).

Matthias Schulz
schulz@schulz-it-services.de

Data Mesh: Eine Anleitung für Datenprodukte & Datenverträge

Andreas Buckenhofer, Mercedes-Benz Tech Innovation

Im Verlauf der letzten Jahre sind Container-Architekturen wie Docker auf dem Vormarsch. Sie haben dazu beigetragen, dass monolithische Architekturen als gängiger Standard bei größeren Anwendungen weitestgehend ersetzt wurden. Datenarchitekturen sind heute in den meisten Firmen immer noch monolithisch organisiert. Da diese Art der Architektur im Datenkontext ähnliche Schwachstellen wie im Anwendungskontext hat, werden zunehmend Stimmen laut, die eine Dezentralisierung fordern. Mit Data Mesh wird solch ein dezentraler Ansatz beschrieben.

Das von Zhamak Dehghani beschriebene Konzept „Data Mesh“ basiert auf vier Prinzipien, die im Folgenden kurz erläutert werden.

Domänenorientierung

Domänenorientierung zielt darauf ab, das Eigentum an den gemeinsam genutzten analytischen Daten auf die Geschäftsbereiche zu dezentralisieren, die den Daten am nächsten stehen. Die Datenartefakte werden logisch auf der Grundlage

der Geschäftsdomäne, die sie repräsentieren, zerlegt und ihr Lebenszyklus unabhängig verwaltet. Auf Daten basierende Anwendungen wie Empfehlungssysteme gehören in eine Domäne und nicht zentral entwickelt.

Daten als Produkt

Alle Geschäftsbereiche werden in die Verantwortung genommen, ihre Daten den anderen Domänen als Produkt bereitzustellen. „Daten als Produkt" führt ein Artefakt in die logische Architektur ein, das alle strukturellen Komponenten wie Daten, Code und Richtlinien kontrolliert und kapselt. Data-Engineering-Expertise wird dadurch in die Domänen verlagert.

Self-Service-Datenplattform

Das Prinzip der Schaffung einer Self-Service-Infrastruktur besteht darin, Werkzeuge und benutzerfreundliche Schnittstellen bereitzustellen. Eine Self-Service-Datenplattform soll fachlich orientierte Teams befähigen, den gesamten Lebenszyklus ihrer Datenprodukte selbst zu verwalten und ein zuverlässiges Netz miteinander verbundener Datenprodukte zu managen.

Föderierte Governance

Überall, wo dezentralisierte Dienste eingesetzt werden, müssen unbedingt übergreifende Regeln und Vorschriften eingeführt werden, um deren Betrieb zu steuern.

Hierfür sollte ein Data-Governance-Betriebsmodell verwendet werden. Dieses basiert auf einer föderalen Entscheidungsfindungs- und Verantwortungsstruktur. Das Governance-Modell stützt sich in hohem Maße auf eine automatische Ausführung von Richtlinien auf jeder Ebene für jedes einzelne Datenprodukt.

Data Mesh verleiht Daten den Status von „first-class Citizens" in der technologischen Landschaft. Dies bedeutet, dass Daten mit der gleichen Sorgfalt und Aufmerksamkeit behandelt werden wie Hardware- oder Softwareprodukte. Durch die Einführung von klar definierten Schnittstellen wird die Interoperabilität zwischen den Datenprodukten gewährleistet, was wiederum die Zusammenarbeit und Integration erleichtert. Daten als Produkt zu betrachten, bedeutet vor allem auch Planung, Engineering, Test, Betrieb, und vieles mehr. Dementsprechend müssen Stories durch einen Datenproduktbesitzer (Product Owner) definiert werden. Solche Stories beinhalten unter anderem Datenprodukte und Datenverträge, die nachfolgend aus einer technischen Perspektive beschrieben werden.

Abbildung 1 zeigt eine E2E-Verarbeitung von der Aufnahme der Daten bis zur Erstellung von Datenprodukten, die beispielsweise in einem internen Marktplatz veröffentlicht werden. Die Speicherung der Daten erfolgt in verschiedenen Qualitätsstufen (Bronze – Silber – Gold) in einer Datenbank, beziehungsweise als Parquet/Delta-Dateien in einem Dateisystem. Datenprodukte (DP1, DP2) werden in der Gold-Zone gespeichert.

Der Code für die Verarbeitung der Daten liegt in einem Repository wie zum Beispiel einem Enterprise GitHub. In diesem Repository liegen außerdem die Metadaten für das Datenprodukt. Jedes Datenprodukt (DP1, DP2) wird in einem eigenen Repository verwaltet, insbesondere gehören die folgenden Dateien dazu:

- Datenprodukt: Product.yaml
- Vertrag: Contract.yaml
- Zuordnung Enterprise Modell/Knowledge Graph: Schema.rdf

Die in dem Repository verwalteten Daten können in einem Datenkatalog/Datenmarktplatz publiziert werden. Idealerweise bietet ein Datenkatalog/Datenmarktplatz eine API an, um diese Metadaten zu übertragen.

Abbildung 2 zeigt einen Ausschnitt aus einer Datei contract.yaml. Im Literaturverzeichnis sind [Agile Lab] und [PayPal] als ausführliche Beispiele für Datenprodukte/Datenverträge aufgeführt. Empfehlenswert ist der Beginn mit nur einem Teil der Attribute, um Projekte nicht von der Menge der benötigten Informationen abzuschrecken. Die Attribute können später hinzugefügt werden, sofern diese relevant sind.

Warum werden die Metadaten in einem Code Repository verwaltet und nicht in einer Datenbank oder in einem Datenkatalog? Gründe hierfür sind:

- Toolunabhängige Speicherung der Metadaten
- Nutzung von Werkzeugen, die den Entwicklern vertraut sind (git, Textdateien)
- Möglichkeit zum Upload der Informationen in verschiedene Datenkataloge/Datenmarktplätze

Ein Datenprodukt wird beschrieben durch ID, Namen, Version, Besitzer, und

Datenprodukt	Ladestationen – Car charging station
Zweck	Ladezyklen Fahrzeug, Auslastung/ Nutzung Ladestationen
High-Level Domäne	Fahrzeug
Version	0.9
Status	Draft
Startdatum (1. Datensatz vom)	01.01.2023
Informationsklassifizierung	Confidential
Schema	• Modell: varchar(100) • Startzeit: timestamp • Endzeit: timestamp • Startkapazität: float • ...
Beispieldaten	...
...	

Tabelle 1

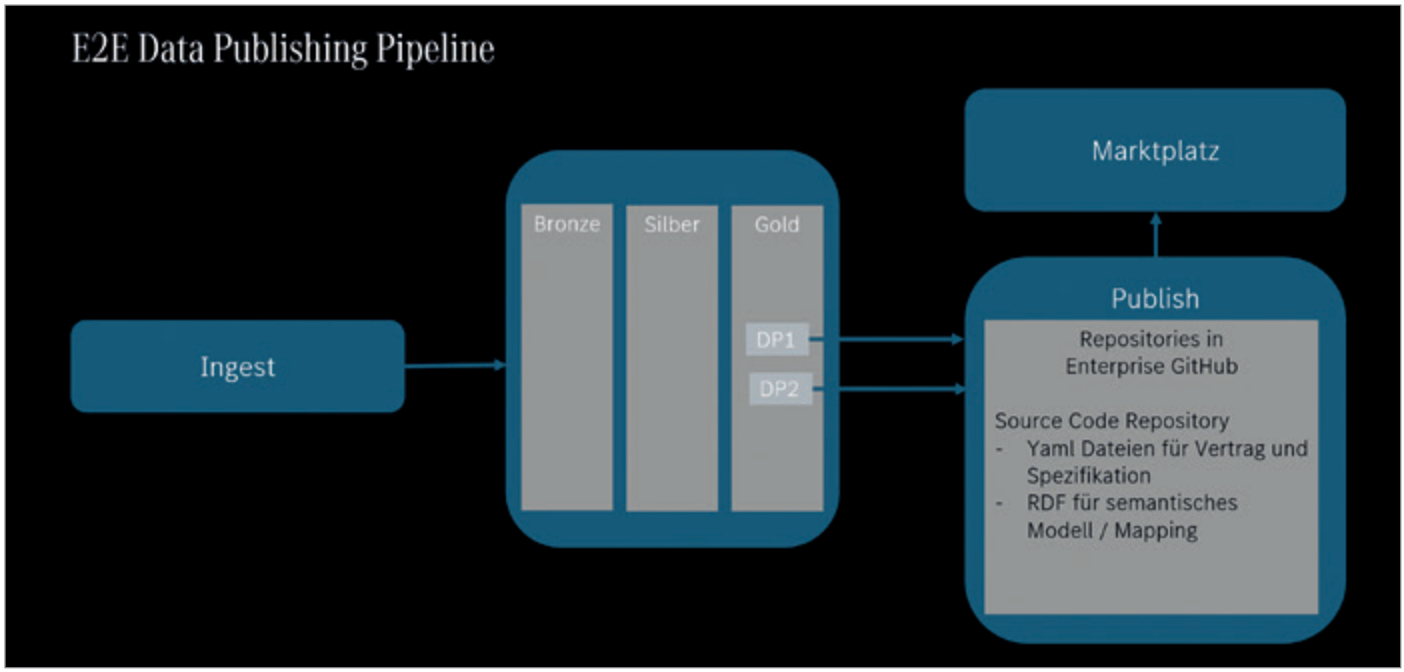


Abbildung 1: E2E Data Publishing Pipeline (Quelle: Andreas Buckenhofer)

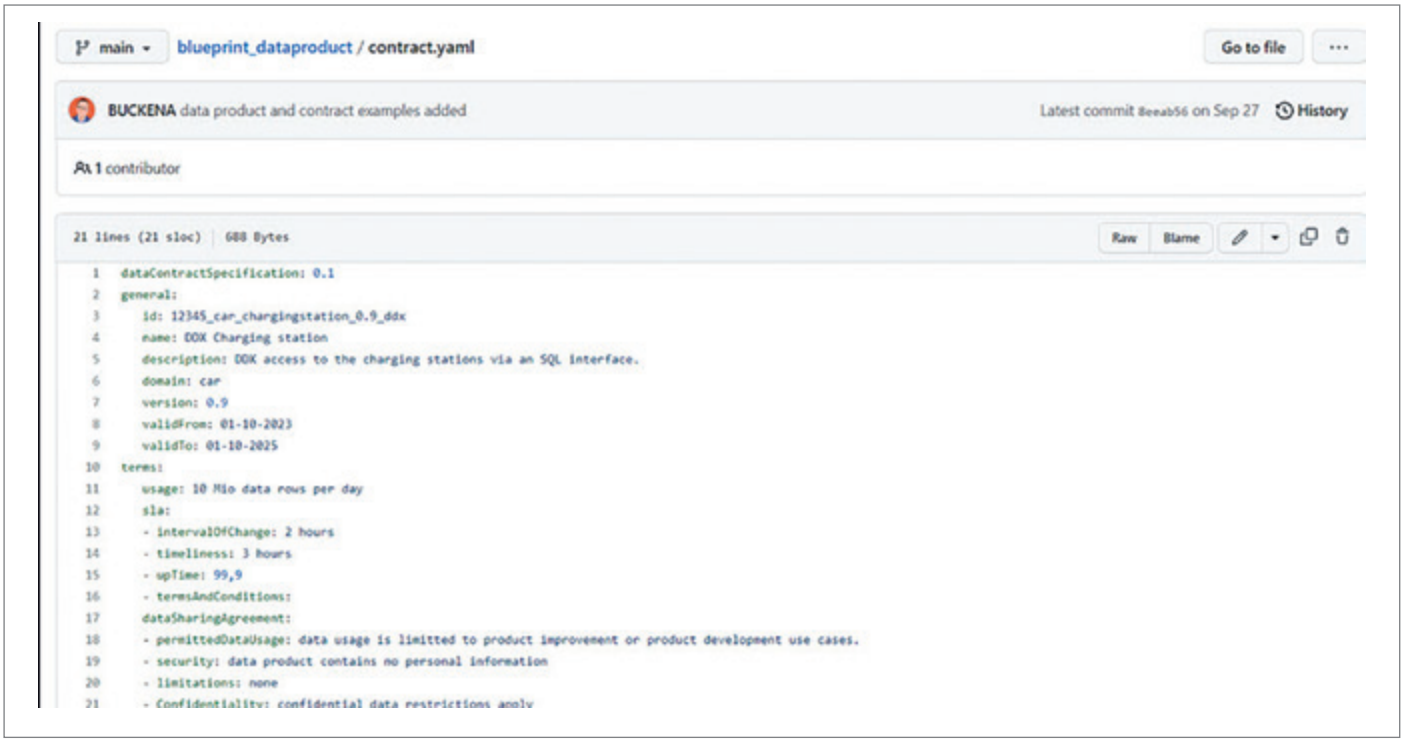


Abbildung 2: contract.yaml (Quelle: Andreas Buckenhofer)

so weiter. Die *Tabelle 1* zeigt einige der Attribute.

Zu dem Datenprodukt sind formelle Nutzungsgrundlagen („Contract“) hinterlegt. Dazu gehören beispielsweise Nutzungsbedingungen, Zugriffsdaten, Zugriffseinschränkungen wie maximal 10 Zugriffe/Minute. Zu einem Datenprodukt kann es mehrere verschiede Verträge geben, beispielsweise verschiede Zugriffsoptionen wie SQL oder API. Die nachfolgende *Tabelle 2* zeigt einige der Attribute.

Außerdem ist zu einem Datenprodukt ein Enterprise-Modell/Knowledge-Graph zugeordnet. In einer weiteren Datei *(schema.rdf)* erfolgt die Zuordnung als RDF-Beschreibung zu Subdomänen.

Data Mesh fördert die Idee, dass Datenprodukte dezentralisiert von Domänen-Teams erstellt und gewartet werden. Diese Herangehensweise soll die Agilität fördern, da Teams unabhängig arbeiten und schneller auf sich ändernde Anforderungen reagieren können. Jedoch birgt die Umsetzung von Data Mesh auch Herausforderungen.

Data Mesh kann zwar die Autonomie stärken, da Teams ihre spezifischen Anforderungen besser verstehen und schnellere Iterationen ermöglichen

können. Dennoch entstehen dabei Fragen zur Konsistenz, Qualität und Sicherheit der Daten. Es erfordert klare Standards und Governance, um sicherzustellen, dass dezentrale Datenprodukte in einer kohärenten Weise zusammenarbeiten.

Die Idee einer „Single Version of Truth" ist ein zentrales Konzept in einem Data Warehouse. Sie besagt, dass es eine einzige, konsistente und zuverlässige Quelle für Daten geben sollte. Im Kontext von Data Mesh könnte dies jedoch herausfordernd sein, da Datenprodukte von verschiedenen Teams erstellt werden, die unterschiedliche Perspektiven und Anforderungen haben. Es wird entscheidend, klare Vereinbarungen über Datenqualität, Semantik und Schnittstellen zu treffen, um sicherzustellen, dass die dezentralen Datenprodukte kohärent und verlässlich sind. Eine Domänen-übergreifende Sicht wie sie durch ein DWH erreicht wird, wird es weiterhin geben. Beispielsweise umfasst eine Supply Chain Datenprodukte über mehrere Domänen wie Zulieferer, Versorgungswege von Teilen/Komponenten, Produktion, Lieferung fertiges Produkt.

Die einzelnen Ideen von Data Mesh sind sicherlich nicht neu. Der starke Fokus auf Dezentralisierung und Unabhängigkeit von Teams führt aktuell zu kontroversen Diskussionen in sozialen Medien. Umso wichtiger sind föderierte Governance und Produktorientierung, um Wildwuchs in den Domänen zu vermeiden. Größere datengetriebene Organisationen profitieren durch

- **Skalierbarkeit und Flexibilität:** eine skalierbare und flexible Datenarchitektur wird ermöglicht, da das Datenmanagement auf verschiedene Domänen verteilt wird und so das Wachstum und die Anpassungsfähigkeit erleichtert.
- **Verbesserte Zusammenarbeit:** Klare Schnittstellen und Interoperabilität zwischen Datenprodukten führen zu einer verbesserten Zusammenarbeit zwischen verschiedenen Teams und Domänen.
- **Effiziente Ressourcennutzung:** Die Self-Service-Datenplattform ermöglicht es, dass fachlich orientierte Teams den Lebenszyklus ihrer Datenprodukte selbst verwalten können, was zu einer effizienteren Ressourcennutzung führt und gleichzeitig die Autonomie der Teams stärkt.

Datenprodukt	Ladestationen – Car charging station
Version	0.9
Status	Draft
Zugriff	SQL, Delta-Tabelle
Zugriffsstring	<...>
Terms	Nutzung erlaubt für Produktverbesserung oder Produktentwicklung
Einschränkungen	
Zugriffsbeschränkungen	• 10 Mio. Datensätze pro Tag (pro Mandant)
Änderungshäufigkeit	2 h
Datenaktualität	8 h
...	

Tabelle 2

Quellen

[Agile Lab] *https://github.com/agile-lab-dev/Data-Product-Specification/tree/main#readme* (besucht am 16.12.2023).

[Dehghani] Zhamak Dehghani: Data Mesh: Delivering Data-Driven Value at Scale, Sebastopol 2022.

[PayPal] *https://github.com/paypal/data-contract-template/blob/main/docs/README.md* (besucht am 16.12.2023).

Über den Autor

Andreas Buckenhofer arbeitet bei Mercedes-Benz Tech Innovation in Ulm als Lead Expert 'Vehicle Data Platforms'. Er verfügt über langjährige Erfahrung in der Entwicklung datenintensiver Produkte. Seine Schwerpunkte sind Datenarchitektur, Datenspeicherung, Datenintegration und Datenqualität.

Andreas Buckenhofer
Andreas.Buckenhofer@mercedes-benz.com

JavaLand

www.javaland.eu

AM NÜRBURGRING

09. – 11.04. 2024

Early Bird bis 15.02.2024

JETZT TICKETS

Präsentiert von: iJUG Verbund · Heise Medien · DOAG

PROGRAMM JETZT ONLINE

SICHERN!

Veranstalter: JavaLand

BUSINESS

NEWS
02/2024

DIE MAGIE VON APEX

Wie Oracle APEX zu einem leistungsstarken Enterprise-App-Entwicklungs-Framework wurde

Ein ganz persönlicher Bericht von Mike Hichwa

Seit meiner Schulzeit in den frühen 1980er Jahren bin ich von der Anwendungsentwicklung fasziniert. Meine High School hatte das Glück, einen Dec VAX PDP 11/750 zu besitzen. Ich nahm ein rotes Buch über die Einführung in BASIC in die Hand und schrieb mein erstes Programm, ein Cribbage-Spiel (ein Kartenspiel). Es gab keine Anleitung, keine Referenzen, keine Lehrer:innen, nur ein Buch und ein VT100.

In den 1990er Jahren war ich geradezu besessen von Oracle Forms. Forms 2.x war okay, aber als Forms 3.0 herauskam, fand ich es genial. Ich arbeitete für Oracle und hatte viele Gelegenheiten, Kunden zu besuchen und ihnen zu helfen, das Beste aus Oracle Forms herauszuholen.

Als sich die Anforderungen der Anwendungen mit grafischen Anzeigen weiterentwickelten, war klar, dass Forms 3.x einige Einschränkungen hatte. Dann kam ein Kollege bei mir vorbei, um mir Mosaic, einen frühen Webbrowser, zu zeigen, war ich wieder wie verwandelt. Es war mir sofort klar, dass die Zukunft der Anwendungsentwicklung mit menschlichen Schnittstellen das Web mit einem „universellen Client", einem Webbrowser, war.

Die Geburtsstunde von APEX

Nach einem weniger als perfekten ersten Versuch namens Oracle WebDB begann die Entwicklung von APEX (zuerst als HTMLDB vermarktet und als „Project Marvel" eingeführt) im Jahr 1999. Das Ziel war es, einen konsistenten, schnellen Weg zum Erstellen von Anwendungen mit minimalem Programmieraufwand bereitzustellen – automatisches „Session State" Management, die Möglichkeit Session-Werte mittels Bindungsvariablen zu referenzieren und deklarative Berichte/Formulare/Diagramme zu ermöglichen.

Einen Code-Generator zu schreiben wäre eine Möglichkeit gewesen, dies alles umzusetzen, aber aus meiner Sicht nicht zielführend. Der Hauptgrund ist die starke Versuchung, den generierten Code zu modifizieren. Wenn der generierte Code modifiziert wird, entsteht eine Abhängigkeit von generierten Artefakten, die unweigerlich dazu führt, dass die Modifikationen irgendwann nicht mehr funktionieren. Ein weiterer Nachteil von generierten Apps ist, dass die Code-Generierung, die Kernlogik, die Generierung von Berichten, die Formularverarbeitung und so weiter dupliziert und es weniger effizient macht, eine große Fülle von Anwendungen auszuführen.

Diese Logik führte zu der Entscheidung, bei der APEX-Implementierung eine Model-Driven-Execution-Engine-Architektur (MDEE) zu verfolgen. In einer MDEE-Architektur wird die Absicht der Anwendung erfasst. Die Absicht ist eine implementierungsunabhängige Sammlung von Metadaten, die definiert, wie die Anwendung funktionieren soll. Die Entwickler:innen geben einfach die gewünschte Funktionalität an, zum Beispiel einen Bericht, ein Formular oder ein Diagramm, das die entsprechenden Funktionen unterstützt. Zur Laufzeit liest die Engine die Metadaten und rendert die Komponente. Wenn sich neue, bessere, sicherere und leistungsfähigere Methoden zum Rendern einer Komponente entwickeln, kann die Engine aktualisiert werden und die Anwendung ist sofort besser.

Große Flexibilität dank Metadaten

Die grundlegende Struktur von APEX ähnelt der von Oracle Forms, mit einer entscheidenden Ausnahme: die Flexibilität der Benutzeroberfläche. Insbesondere wollte ich die Entwickler:innen in die Lage versetzen, Unternehmensanwendungen zu erstellen, die große Freiheiten in der Oberflächen-Gestaltung bieten. Eine konkrete frühe Anforderung bestand darin, die Möglichkeit zu unterstützen, Metadaten deklarativ in einem Assistenten zu sammeln, der häufig als „Flow" bezeichnet wird, um verschiedene Felder und Optionen basierend auf den Entscheidungen der Endbenutzer anzuzeigen. Tatsächlich war der erste interne Name von APEX „Flows".

Von Anfang an „Application Development as a Service"

Etwa im Jahr 2001, nach einer Vorführung für einen Senior Manager bei Oracle, erhielt das Projekt grünes Licht mit einer entscheidenden Änderungsanfrage: „Mache es gehostet." In der heutigen Sprache würde man sagen, mache es zu „Application Development as a Service". APEX war bereits mehrbenutzerfähig, da mehrere Entwickler:innen gleichzeitig eine oder mehrere Apps entwickeln konnten. Das, was fehlte, war eine Möglichkeit, eine Entwicklungsumgebung von einer anderen zu isolieren (heutzutage als „Workspace" bekannt). Um dies zu erreichen, habe ich mir angeschaut, was Oracle E-Business Suite zu dieser Zeit verwendete, nämlich das Hinzufügen einer „security_group_id"-Spalte zu jeder Tabelle. Dies trennte die Metadaten nach Workspace und ermöglichte es, in einer einzelnen APEX-Installation beliebig viele Workspaces zu unterstützen, wobei jeder Workspace beliebig

viele Entwickler:innen und Anwendungen unterstützte. Um die Daten zwischen den Arbeitsbereichen zu isolieren, erhielt jeder Workspace Zugriff auf ein oder mehrere Datenbankschemata.

Ein Registrierungsassistent wurde eingeführt. Da die APEX-Entwicklungsumgebung in sich selbst geschrieben ist, war dies nicht schwer. Als wir dies 2002 intern einführten, mussten Oracle-Mitarbeiter:innen im Intranet einfach zu einer URL gehen und sich selbst einen APEX-Workspace bereitstellen, der einen App Builder zum Erstellen von Anwendungen, einen SQL-Workshop zum Erstellen von Tabellen, einen Datenworkshop zum Importieren von Daten und ein Datenbankschema zum Speichern ihrer Daten enthielt. Dies dauerte in der Regel etwa 30 Sekunden: von der Entscheidung „Ich möchte eine App erstellen" bis zum Vorhandensein alles Notwendigen, um eine Anwendung zu entwickeln. Der Schlüssel zu dieser Einfachheit ist, dass die Entwicklungsumgebung und die Laufzeitumgebung zu 100% browserbasiert sind.

Dies war ein entscheidender Wendepunkt. Bis zum Start dieses internen AppDev-as-a-Service war APEX zwar einfach zu verwenden, erforderte jedoch einen DBA mit Netzwerkkenntnissen und Zugriff auf einen Server, normalerweise eine Sun-Spark-Pizza-Box. Dies bedeutete Stunden an Arbeit, Fachkenntnisse und ein Budget zum Betreiben eines Servers. Jetzt konnten mit dem AppDev-as-a-Service alle, die in ihrem Webbrowser eine URL aufrufen konnten, eine eigene Sandbox erstellen, komplett mit allem, was nötig ist, um in nur 30 Sekunden über einen Browser eine Webanwendung zu erstellen und bereitzustellen. Keine Komplexitäten, kein Drama, einfach zeigen und klicken, einem Assistenten folgen.

Low-Code für alle!

Was als Nächstes geschah, war erstaunlich. Die Anwendungsentwicklung wurde demokratisiert. Unternehmen begannen, ihre Tabellenkalkulationen in Anwendungen umzuwandeln und ihre Anwendungen häufig von Personen mit keinem formellen IT-Know-how weiterentwickeln zu lassen. Es wurde eine Low-Code-Citizen-Developer-Utopie geschaffen. Im Laufe der Zeit haben Zehntausende von Oracle-Mitarbeiter:innen Apps mit APEX erstellt, etwa die Hälfte davon mit nichttechnischen Berufsbezeichnungen. Und alle bei Oracle verwenden täglich APEX-Anwendungen, um ihre Arbeit zu erledigen.

Unsere interne Oracle-APEX-Instanz ist seit etwa 2002 im Betrieb und wurde über die Jahre Hardware-technisch aufgerüstet. Viele Anwendungen darauf, die täglich verwendet werden, sind über zehn Jahre alt. Unsere älteste Anwendung, die noch genutzt wird, ist über 17 Jahre alt und wird immer noch von Tausenden von Personen verwendet. Tatsächlich ist jede jemals auf APEX entwickelte App aufwärtskompatibel zur aktuellen Version.

Eine rasante Entwicklung

Im Jahr 2004 wurde Oracle APEX 1.5 (damals als HTMLDB bezeichnet) veröffentlicht und seitdem wurden 40 weitere Versionen veröffentlicht, die uns heute zu APEX 23.2 gebracht haben. Jede Veröffentlichung hat APEX benutzerfreundlicher und leistungsfähiger gemacht. So wurden „Interactive Reports" in 2008 eingeführt, um Endbenutzer:innen die Auswahl von Spalten, Filtern, das Sortieren und Charting von Daten zu ermöglichen, wobei der Entwickler oder die Entwicklerin dafür lediglich einen Tabellennamen oder eine SQL-Anweisung bereitstellt. 2010 wurden „Dynamic Actions" eingeführt, die deklaratives JavaScript-Event-Handling ermöglichen und effektiv die Notwendigkeit für APEX-Entwickler:innen beseitigen, JavaScript für typische Muster zu programmieren. Eine Plug-in-Architektur wurde ebenfalls 2010 eingeführt. Der „Page Designer" und das „Universal Theme" kamen 2015 dazu. „Faceted Search"-Suche und signifikante Fortschritte bei REST-Webservices gab es 2019, PWA-Unterstützung im Jahr 2022 und Workflow im Jahr 2023.

APEX-Apps bleiben automatisch aktuell

Du denkst vielleicht: „Wer möchte mit APEX-Applikationen arbeiten, die in die Jahre gekommen sind?" Aber APEX-Apps sehen nicht alt aus und verhalten sich auch nicht so, weil sie nicht alt sind. Der von APEX generierte JavaScript-, CSS- und HTML-Code entwickelt sich mit jeder Veröffentlichung weiter und hält die Anwendung durch fortschrittliche Unterstützung für Barrierefreiheit, Sicherheit und führende Funktionalität stets aktuell. Natürlich basiert APEX auf vielen sich ständig weiterentwickelnden JavaScript-Frameworks, von denen einige aus der Mode kommen und durch andere ersetzt werden. Aber APEX kümmert sich um diese Fluktuation. Jede neue Veröffentlichung von APEX bietet ein Upgrade der APEX-Engine, die alle Bytes aktualisiert, die deinen Browser erreichen, einschließlich JavaScript, HTML und CSS. Es ist wichtig zu beachten, dass die Kombination von HTML und Daten auf dem Server stattfindet, was im Browser weniger Unruhe beim Seitenaufbau verursacht, da der Browser das HTML direkt rendert und die App schnell und reibungslos ohne Springen darstellt.

Die Philosophie von APEX besteht darin, alles deklarativ zu halten, was deklarativ sein kann, und kreativen und talentierten Entwickler:innen zu ermöglichen, die Plattform über Plug-ins und andere kreative Methoden zu erweitern. Die Trennung von der Anwendungsdefinition und Implementierung erfordert lediglich ein Upgrade von APEX und gegebenenfalls Plug-ins, um auf dem neuesten Stand zu bleiben.

Das Preis-/Leistungsverhältnis ist einzigartig

Ein weiteres wichtiges Prinzip von APEX besteht darin, es so vielen Entwickler:innen wie möglich zu minimalen Kosten zugänglich zu machen. APEX ist kostenlos auf https://apex.oracle.com für die Evaluierung verfügbar. Dauerhaft kostenlose Versionen sind auf der Oracle Autonomous Database in der Oracle Cloud verfügbar. APEX ist eine kostenfreie Option der Oracle-Datenbank, unabhängig von der Edition einschließlich der kostenfreien XE-Edition. Es gibt keine Zusatzkosten für erweiterte Funktionen. Lizenzmetriken pro Entwickler:in, pro Endanwender:in oder pro Applikation gibt es bei APEX nicht. Wenn du Zugriff auf eine Oracle-Datenbank hast, kannst du APEX in vollem Umfang nutzen, wie es für dich am besten geeignet ist, ohne zusätzliche Kosten über die Oracle-Datenbank hinaus.

Oracle APEX ist in der vollständig von Oracle verwalteten Autonomous Database verfügbar. Eine Variante der Oracle Autonomous Database namens APEX-Service ist ebenfalls verfügbar. Der APEX-Service ist identisch mit der Autonomous Database, kostet jedoch weniger und bietet dafür keinen SQL*Net-Zugriff. Jede Analyse oder faire Bewertung des APEX Low-Code-Frameworks durch Analysten wird zu dem Schluss kommen, dass es bemerkenswert kostengünstig ist.

APEX in der Datenbank: Fluch oder Segen?

Eine der einzigartigen Eigenschaften von Oracle APEX, die einige vielleicht als eigenwillig bezeichnen, ist, dass das Framework in der Datenbank beheimatet ist. Die modellgesteuerte Ausführungs-Engine befindet sich innerhalb der Datenbank. Die APEX-Engine ist in der von Oracle verwendeten Datenbanksprache PL/SQL geschrieben. Aufgrund dessen könnte man zu dem Schluss kommen, dass APEX die Datenbank stärker belastet, weil es innerhalb der Datenbank läuft. Obwohl dies völlig gegen die Intuition zu sein scheint, ist das Gegenteil der Fall. Dies liegt größtenteils daran, dass APEX-Anfragen mit einer einzigen HTTP-Anfrage und Antwort bedient werden, was die Anzahl offener Verbindungen und den Speicherverbrauch dramatisch reduziert.

PL/SQL teilt die gleiche Typenstruktur wie SQL und hat daher keine Abbildungstypen. PL/SQL befindet sich innerhalb der Datenbank, sodass das Ausführen der Vielzahl von SQL-Anweisungen, die für eine moderne Anwendung erforderlich sind, keine Netzwerktraversierung erfordert und nicht mit der damit verbundenen Netzwerklatenz verbunden ist. Eine moderne Unternehmensanwendung führt 10 bis 50 SQL-Anweisungen pro Seite aus, manchmal mehr. Die Vermeidung dieser Anzahl von Rundreisen zwischen Mittelschicht und Datenbank verbessert die Leistung und reduziert den CPU-Verbrauch.

Ein weiterer Vorteil der auf die Oracle-Datenbank ausgerichteten Architektur ist, dass du bei Bedarf an Datenpersistenz kein weiteres bewegliches Teil einführen brauchst. Eine schlanke Architektur bedeutet weniger Verwaltung, ist weniger Fehlerträchtig und ist sicherer. APEX läuft vollständig in der Datenbank, reduziert damit die Komplexität und erhöht gleichzeitig die Leistung. Dazu kommt, dass APEX von den Features der Datenbank stets profitiert. Beispielsweise wird erwartet, dass die Oracle Database 23c, die 2024 verfügbar sein soll, eine JavaScript-MLE-Engine, „Duality Views" und Unterstützung für Vektoren umfasst.

Eine konsolidierte Architektur erleichtert auch Ausfallsicherheit, Katastrophenschutz, Backup, Debugging, IT-Sicherheit und Performance-Tuning. Die Oracle Datenbank läuft in Oracle Cloud Infrastructure (OCI) und anderen hyper-skalierbaren Clouds sowie on-premises. Die gleiche Infrastruktur läuft jedoch auch auf einem Laptop.

Nicht nur Larry ist von APEX überzeugt

2023 hat Oracle öffentlich auf der Oracle CloudWorld und auf anderen Veranstaltungen bekannt gegeben, dass APEX die interne erste Wahl von Oracle für neue Entwicklungsprojekte ist. Die Hauptgründe dafür sind:

- APEX reduziert den Code – jede Codezeile verursacht Kosten – und damit die Kosten für Entwicklung und Wartung.
- APEX reduziert die Entwicklungszeit und ermöglicht schnelles Prototyping und Evolution durch kleinere Entwicklungsteams.
- APEX läuft innerhalb der Oracle Autonomous Database und reduziert damit Betriebskosten und Komplexität.
- APEX ist effizient und skaliert nachweislich auf über 1 Million Nutzer:innen in einer einzigen Anwendung an einem einzigen Tag.
- APEX-Anwendungen entwickeln sich durch ein Upgrade der APEX-Plattform automatisch weiter. In einer autonomen Datenbank geschieht das Upgrade sogar vollautomatisiert.
- APEX hat sich mit über 10 Mio. entwickelten Anwendungen und über 2 Mio. in Betrieb befindlichen Anwendungen bewährt.
- APEX profitiert direkt von den Investitionen von Oracle und der Stärke der Oracle RDBMS-Plattform.

Mehr Apps mit weniger Ressourcen

Die Entwicklung von Anwendungen wird nicht langsamer, sondern immer schneller. Jedes große Unternehmen verfügt über ein umfangreiches und wachsendes Portfolio an Anwendungen. Die Entwicklung von Low-Code-Anwendungen wird immer beliebter. Low-Code wirkt sich auch auf die Art und Weise aus, wie Integratoren Mehrwert liefern und die Entwicklungszeit verkürzen. Mit der Möglichkeit, hybride Teams bestehend aus nicht-professionellen Programmierer:innen mit professionellen Entwickler:innen zusammenarbeiten zu lassen, können noch mehr Anwendungen bereitgestellt werden.

Worauf noch warten?

Oracle APEX ist einzigartig, es hat eine einzigartige Architektur, es hat eine einzigartige Geschichte und eine große, lebendige Community. Oracle erhöht die Investitionen in die Plattform und hat ein einzigartiges Preismodell, das für alle, die bereits die Oracle Datenbank lizenziert haben, keine Zusatzkosten verursacht. Besuche *https://apex.oracle.com*, um mehr zu erfahren und APEX kostenlos zu testen. Das APEX-Team ist sehr stolz auf das Produkt und wir geben unser Bestes, um unsere Kunden zufrieden zu stellen.

Michael Hichwa
michael.hichwa@oracle.com

Mike ist Senior Vice President bei Oracle und verantwortlich für die Entwicklung von Datenbankwerkzeugen sowie die Zugriffsmöglichkeiten für die Oracle-Datenbank. Er ist der ursprüngliche Architekt und Entwickler von Oracle APEX und entwickelt täglich Applikationen damit. Einige von Mikes beliebten öffentlichen Webapplikationen sind ***asktom.oracle.com***, ***livesql.oracle.com***, ***forums.oracle.com*** und ***apex.oracle.com***. Er leitet auch die Treiberentwicklung für JDBC, Python, .NET und andere. Sein Team ist auch verantwortlich für SQL Developer, SQLcl, Oracle REST Data Services und die Entwicklung der Oracle Database Cloud-Konsole. Mike und seine Organisation konzentrieren sich darauf, Entwickler:innen ein Höchstmaß an Produktivität und Leistung zu bieten. Wer wissen möchte, was Mike aktuell so treibt, folgt am besten seinem X-Handle @MikeHichwa.

ā'pěks

(#orclapex)

Die Magie von APEX – Interview mit Niels de Bruijn

Von Katharina Schraft

Business Applikationen kommen in verschiedenen Größen, Ausrichtungen und mit unterschiedlichen Zielsetzungen daher. In dieser Ausgabe der Business News steht APEX im Vordergrund, eine Low-Code-Plattform, mit der es besonders einfach und schnell zu sein scheint, eigene Applikationen zu erstellen – standalone oder integriert mit bestehenden Systemen. Was die Magie von APEX ausmacht, wollte Katharina Schraft, Vorstand Business Solutions, von Niels de Bruijn, APEX-Spezialist und Ehemaliger Vorstand der Development Community, wissen.

Hallo Niels, schön, dass wir heute gemeinsam über das Thema Business Applikationen mit Oracle APEX sprechen können. Was ist deine Rolle in der DOAG und wie bist du mit APEX in Verbindung gekommen?

Nach vier Jahren im Vorstand setze ich mich jetzt als Leitungskraft bei der Development Community dediziert für die jährliche Konferenz APEX connect *[4]* ein. Meine erste Berührung mit Oracle APEX *(Anm. d. Red.: nachfolgend APEX genannt)* ist 20 Jahre her, so lange wie es APEX auch gibt. Anfangs habe ich APEX-Apps für meinen Arbeitgeber entwickelt. Heute sind wir mit 52 APEX-Spezialistinnen und -Spezialisten im deutschsprachigen Raum unterwegs und haben bereits viele hunderte Anwendungen aller Art für unsere Kundinnen und Kunden realisiert.

Was macht für dich die Magie von APEX aus?

APEX nimmt einem viel Arbeit bei der Entwicklung von Webapplikationen ab, dadurch kann die Zusammenarbeit mit dem Fachbereich wieder im Fokus stehen. Dazu kommt die starke APEX Community. Viele Entwicklerinnen und Entwickler sind sehr leidenschaftlich dabei und teilen gerne ihr Know-how. Diese Magie kann man beispielsweise bei der APEX connect 2024 auch selbst erleben. Dazu sage ich später noch etwas.

Wie kann APEX sich bei den vielen Low-Code-Plattformen am Markt behaupten?

Auf der Seite Low-Code Vendors *[6]* haben wir neben APEX bereits 150 weitere Plattformen gelistet. Ein Vergleich damit würde den Rahmen dieses Interviews sprengen. Generell kann man jedoch festhalten, dass APEX mit den Top-Spielern am Markt mithalten kann. Insbesondere ist der Preis unschlagbar: Meist sind die Anzahl Benutzerinnen und Benutzer und/oder die Zahl der Applikationen zu lizenzieren. Dagegen zahlt man für den APEX Cloud Service aktuell 360 US-Dollar pro Monat – und dann gibt es diese Limitierungen nicht. Inhaltlich ist hervorzuheben, dass APEX mit den (deklarativen) Integrationsmöglichkeiten sehr weit fortgeschritten ist. Die Daten müssen nicht zwanghaft in der Oracle Datenbank liegen. Die Herausforderung bei APEX liegt aus meiner Sicht eher bei der Aufklärung in den oben genannten Punkten, denn am Commitment von Oracle kann es nicht liegen: Larry Ellison setzt komplett auf APEX, sowohl intern als auch für Applications (SaaS). Hinzu kommt, dass zurzeit alle Applikationen beim aufgekauften Unternehmen Cerner (Gesundheitswesen) auf APEX umgestellt werden.

Wie vielfältig siehst du die Einsatzmöglichkeiten?

Meist wird APEX für Lösungen im Intranet oder Extranet eingesetzt. Solange man eine Webapplikation anstrebt, ist alles mit APEX realisierbar. Die Frage ist immer: Geht dies „per Klick" (no-code), muss ich hierfür eine SQL-Abfrage schreiben oder ein Plug-in einsetzen (low-code) oder muss ich es selbst mittels PL/SQL oder JavaScript realisieren (full-code)? Umso mehr man deklarativ machen kann, umso höher ist die Produktivität und Qualität. In jedem Projekt von uns ist zwar Custom Code enthalten, aber der Anteil reduziert sich, umso mehr Features durch APEX bereitgestellt werden.

Fachlich gesehen, sind die Einsatzmöglichkeiten von APEX quasi unendlich. So haben wir bei der MT GmbH bereits 30 Apps produktiv laufen, unter anderem eine Fuhrparkverwaltung, ein Pipeline Reporting Tool mit Schnittstelle zu MS Dynamics, eine Restbudget-App, eine App zur Meldung von Projekten und eine Urlaubsverwaltungs-App inklusive Schnittstelle zu SAP. Nicht nur uns hilft APEX ungemein, sondern beispielsweise auch Union Investment, wo seit Jahren erfolgreich eine APEX-App für die Verwaltung von Fonds-

stammdaten, Benchmarks und Anlagerestriktionen für alle Mitarbeitenden betrieben wird. Für weitere Praxisbeispiele möchte ich auf mein Webinar *[7]* von September 2023 verweisen.

Gibt es etwas, das APEX in dem Umfeld nicht kann? Und wie wird es sich in dem Kontext der Business Solutions weiterentwickeln?

Was APEX „out of the box", also ohne eigene Kodierung, zurzeit nicht hat, ist KI-Unterstützung, Offlinefähigkeit, Testautomatisierung, Custom Reports in Word/Excel, Deployment-Automatisierung und Prozessmodellierung auf Basis von BPMN 2.0. Die gute Nachricht ist jedoch, dass für diese Bereiche bereits erprobte Lösungen von Dritten existieren oder das APEX-Team dabei ist, die Funktionalität bereit zu stellen *[3] [2] [8] [9] [10] [11]*.

APEX stellt eine sinnvolle Ergänzung zu Business Solutions dar. App-Hersteller wie SAP, Microsoft, Oracle, Salesforce & Co. bieten eine REST-Schnittstelle an. Auf der Basis können mittels APEX deklarativ intuitive Webapplikationen bereitgestellt werden. Wie einfach dies geht, wird in einem APEX Office Hours Video *[5]* gezeigt. Auch das Beispiel mit Oracle E-Business Suite bei Interseroh in dieser Ausgabe *[Anm. d. Red.: siehe Artikel ab S. 50]* stellt dies unter Beweis.

Wo und in welchem Rahmen kann man mehr über APEX und die Einsatzszenarien im Business Umfeld herausfinden?

Da ist natürlich der Besuch der APEX-connect-Konferenz vom 22. bis zum 24. April 2024 in Düsseldorf zu empfehlen. Auf der Jubiläumsausgabe – es ist die 10. APEX-Konferenz – werden am ersten Tag APEX-Lösungen vorgestellt, und parallel findet ein APEX-Bootcamp für Anfängerinnen und Anfänger statt. Am zweiten und dritten Tag gibt es wieder viele Vorträge zur Vertiefung in die Themen APEX, PL/SQL und JavaScript. Bis zur Konferenz lohnt es sich auch, einen Blick auf meinem Blogpost *[1]* zu werfen, in dem ich Anregungen für APEX-Neulinge gebe.

Niels, vielen Dank für das Interview!

Quellen

[1] Getting up to speed with Oracle Application Express (APEX), Niels de Bruijn 2023: *https://nielsdebr.blogspot.com/2022/10/up-to-speed-with-apex.html*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[2] Prozessmodellierung mittels Flows for APEX: *https://flowsforapex.org*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[3] Oberflächentests mittels LCT: *https://lct.software*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[4] APEX connect: *https://apex.doag.org*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[5] APEX Office Hours: *https://www.youtube.com/watch?v=5cmT8d-FJkz0&t=2779s*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[6] Low-Code Vendors: *https://lowcode.mt-itsolutions.com*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[7] Low-Code Apps entwickelt mit Leidenschaft und Oracle APEX: *https://www.youtube.com/watch?v=QStf2veRPr8*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[8] Offlinefähige APEX-Anwendungen: *https://apex.mt-itsolutions.com/offline-capability*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[9] APEX Statement of Direction: *https://apex.oracle.com/sod*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[10]: APEX Office Print: *https://apex.mt-itsolutions.com/aop*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

[11]: Schritt für Schritt zur DevOps-Kultur: *https://apex.mt-itsolutions.com/devops*, letzter Zugriff am 14.12.2023 um 10:00 Uhr.

Niels de Bruijn
niels.debruijn@mt-itsolutions.com

Niels de Bruijn arbeitet seit 2003 für die MT GmbH, ein mittelständisches IT-Dienstleistungsunternehmen mit Hauptsitz in Ratingen. Als Bereichsleiter verantwortet er die Applikationsentwicklung mittels Low-Code-Plattformen, angefangen bei der Plattformauswahl, über die Implementierung bis hin zur Wartung. Sein Wissen teilt er auf IT-Konferenzen sowie über Beiträge als Mitglied in den Vereinen DOAG e.V. und Low-Code Association e.V.

Katharina Schraft
katharina.schraft@doag.org

Katharina Schraft ist Vorstand der Business Solutions der DOAG. Sie ist seit 2010 bei der PROMATIS Unternehmensgruppe am Hauptsitz in Ettlingen (TechnologieRegion Karlsruhe) tätig und hat ihren Schwerpunkt in der ERP-Beratung mit Fokus auf Finance. Seit 2018 ist sie Vice President und Leiterin der Business Unit NetSuite bei PROMATIS und berät Kunden unter anderem in strategischen Buchhaltungs- und Compliance-Fragestellungen.

APEX connect

by DOAG

on demand

APEX 2023 VERPASST?

Jetzt On-demand-Ticket buchen und
Vortragsaufzeichnungen anschauen!

Alle Angebote im
On-demand-Ticket-Shop

apex.doag.org DOAG

Verpackungslizenzierung 2.0 mit EBS und APEX

Eva Reil, Ann-Kathrin Denker, Simon Grossmann und Johannes Michler

Die Interseroh+ ist das duale System des Kreislaufdienstleisters Interzero. Über die Verpackungslizenzierung hinaus bietet das duale System Interseroh+ den kompletten Wertstoffkreislauf aus einer Hand – vom recyclinggerechten Verpackungsdesign bis hin zu hochwertigen Rezyklaten für eine nachhaltige Produktion. Unternehmen, die sich für eine strategische Partnerschaft mit der Recycling-Allianz Interseroh+ entscheiden, erfüllen damit nicht nur ihre Pflichten aus dem nationalen Verpackungsgesetz, sondern auch im europäischen Kontext. Das Angebot der Interseroh+ richtet sich daher an alle Unternehmen, die nicht nur ihre gesetzliche Pflicht zur Verpackungslizenzierung erfüllen, sondern selbst Verantwortung für das Schließen von Recycling- und Rohstoffkreisläufen übernehmen wollen.

Historie der Oracle E-Business Suite bei Interseroh+

In Deutschland fallen jährlich circa 2,6 Millionen Tonnen Verpackungen an *[1]*. Die Verpackungen werden getrennt nach Leichtverpackungen (LVP; u. a. gelber Sack, gelbe Tonne), Glas (u. a. Glascontainer) und PPK (u. a. Papiertonne) gesammelt. Die Organisation der bundesweiten Entsorgung, Sortierung und Verwertung gebrauchter Verkaufsverpackungen an privaten Haushalten wird über sogenannte duale Systeme sichergestellt.

Den Rahmen dazu bildet das Verpackungsgesetz (VerpackG), in dem unter anderem Zulassungskriterien für den Systembetrieb eines dualen Systems und einzuhaltende Verwertungsquoten des gesammelten Materials vorgeschrieben werden.

Die Finanzierung erfolgt über die Beteiligungsentgelte der Produkthersteller, die erstmals mit Ware befüllte Verpackungen in Deutschland in Verkehr bringen. Sie sind gesetzlich dazu verpflichtet, ihrer Produktverantwortung nachzukommen und ihre in Umlauf gebrachten Verkaufsverpackungen an einem dualen System zu beteiligen. Die Leistungen der dualen Systeme werden über die sogenannten Lizenzentgelte finanziert. Diese richten sich nach dem verwendeten Verpackungsmaterial und der Menge.

Bei der Interseroh+ wird die Geschäftsabwicklung intern in die Bereiche Betrieb und Vertrieb unterteilt. Der Betrieb stellt die Aufrechterhaltung der Zulassungskriterien zum Systembetrieb eines dualen Systems sicher (unter anderem Sicherstellung eines flächendeckenden Entsorgungs-, Sortier- und Verwertungsnetzwerks), während sich der Vertrieb um die Lizenzierung der Verkaufsverpackungen im dualen System der Interseroh+ kümmert. Hierbei war die Interseroh+ schon immer hochgradig digitalisiert. Dies zeigt sich daran, dass es sich bei den Betriebs- und Vertriebsprozessen jeweils um sehr „virtuelle" Prozesse handelt. Interseroh+ ist ein rein digitales duales System, welches über „keine eigenen Müllautos" verfügt.

Damit Interseroh+ hierfür zielgruppengerechten und effizienten Service bieten kann, hat sich der Dienstleister vor etwa 15 Jahren dazu entschlossen, die IT-Anforderungen des Betriebs und Vertriebs eines dualen Systems mittels der Oracle E-Business Suite (EBS) umzusetzen. Hierbei wurde auf den Prozessen und Objekten, welche bereits mit dem ERP-System mitgeliefert werden, aufgebaut. Objekte wie Auftrag, Rechnung oder Bestellung und deren Prozesse sind bereits im Standard implementiert und wurden für die Umsetzung der Anforderungen schon in der ersten Implementierung verwendet. Über die Jahre wurden viele Schnittstellen und Portale an die E-Business Suite, in der alle Daten zentral zusammenlaufen, angeschlossen.

Diese historisch gewachsene Systemlandschaft sollte im Rahmen des Projekts „PUSH.IT" rundum erneuert werden – sowohl um eine noch höhere Automatisierung zu erreichen als auch, um leichter in der Lage zu sein, zeitgemäße Portale und moderne Schnittstellen anzubieten. Dafür sollte aufbauend auf dem soliden Fundament des EBS-Standards die Orchestrierung dieser – branchenbedingt sehr speziellen – Prozesse mittels Oracle APEX erfolgen.

Fachliche Anpassungen an die Prozesse

Vor der IT-seitigen Umsetzung wurden die neuen Soll-Prozesse in den beiden Fachbereichen Betrieb und Vertrieb als auch in die Ziele der neuen Lösung aufgenommen.

Neben der strategischen Anforderung an die Portale sollten darüber hinaus auch die Verträge im Vertrieb digitalisiert und standardisiert werden *(siehe Abbildung 1)*. In der alten Lösung gab es zahlreiche Kunden- und Vertragsindividuallösungen, so dass teilweise neben den automatisierten Prozessen auch manuelle Tätigkeiten auszuführen waren. Hiervon wollte man sich mit der neuen Lösung verabschieden. Ziel war es, die Kunden anhand festgelegter Kriterien in definierte Kundencluster einzugliedern. Dadurch konnten die Prozesse vereinfacht, standardisiert und weitergehend automatisiert werden.

Als Basis für die Modellierung der neuen IT-Systemlandschaft *(siehe Abbildung 2)*, wurden die bestehenden Prozesse und System herangezogen. Das Schaubild zeigt mittig in Rot die Oracle E-Business Suite, welche die mitgelieferten Standardmodule enthält. In Grün sind die externen und internen APEX-Portale zu sehen. Rechts befindet sich in Rot die Oracle BI Suite, welche für Datenauswertungen verwendet wird. Darüber hinaus sind noch weitere Umsysteme, wie SharePoint, Microsoft CRM für Kunden- und Vertragsakten sowie SAP für Hauptbuch und Cash-Management abgebildet.

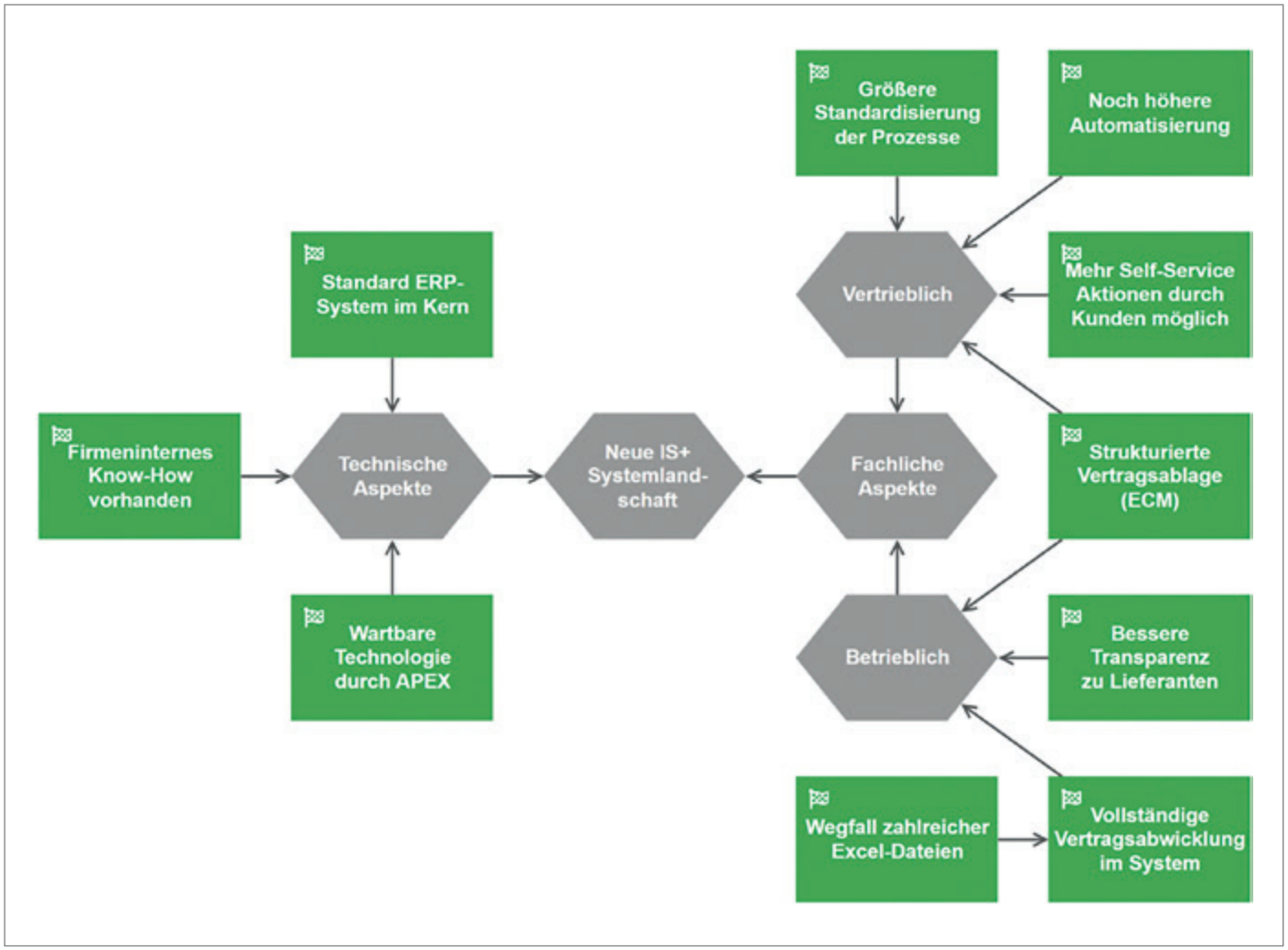


Abbildung 1: Ziele der neuen Prozesse im betrieblichen und vertrieblichen Bereich (© PROMATIS)

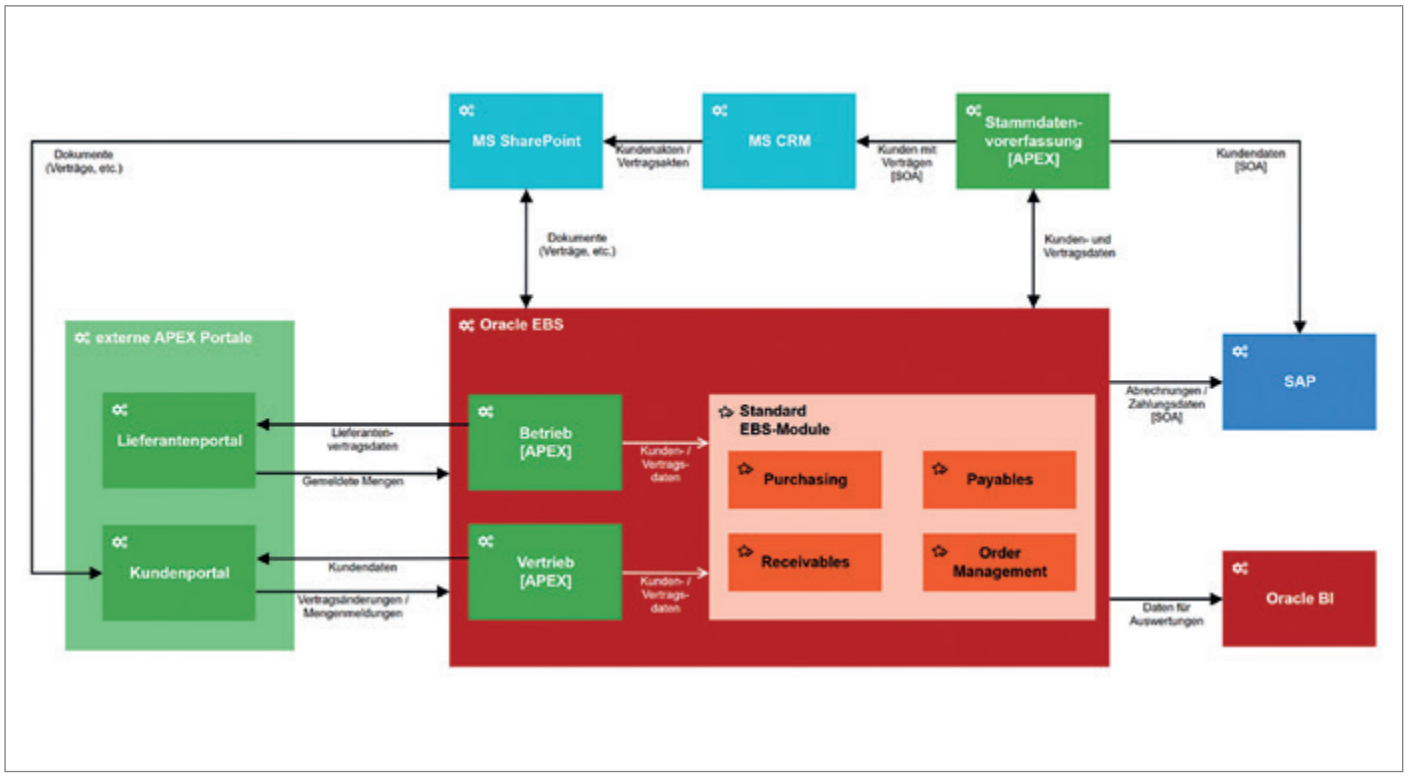


Abbildung 2: Neue IT-Systemlandschaft der Interseroh+ (© PROMATIS)

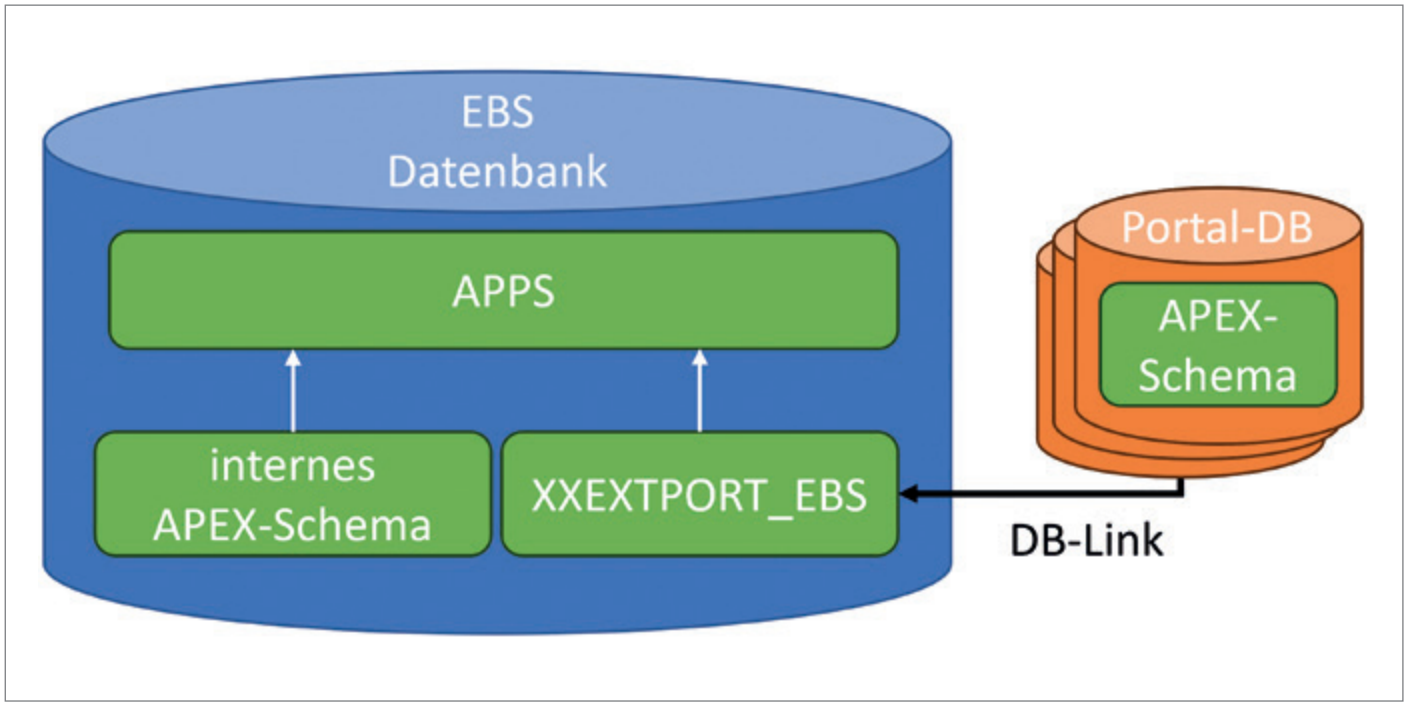


Abbildung 3: Architektur der externen und internen APEX-Portale (© PROMATIS)

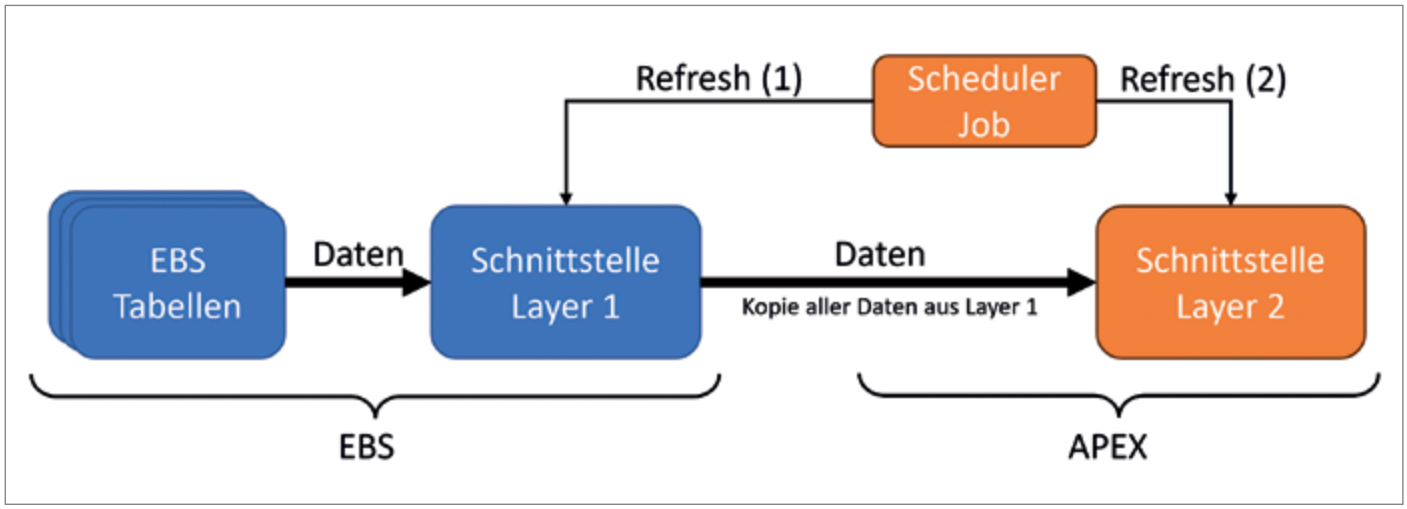


Abbildung 4: Lesende Schnittstelle zur Übertragung der Daten von der Oracle E-Business Suite an die APEX-Portale (© PROMATIS)

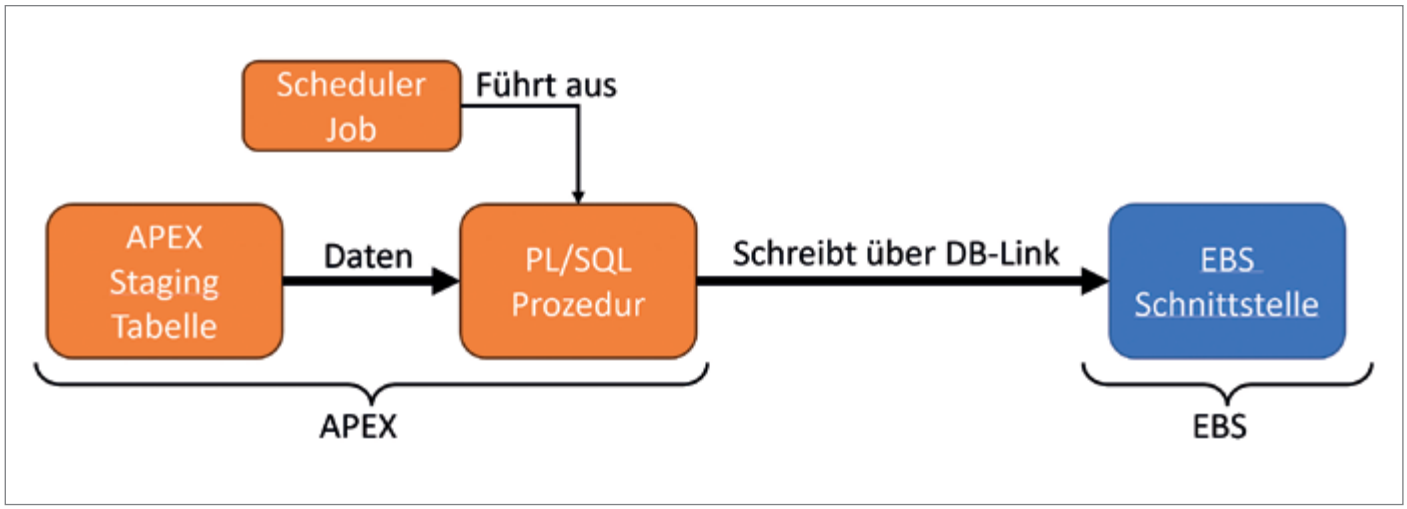


Abbildung 5: Schreibende Schnittstelle zur Übertragung der Daten von APEX an die Oracle E-Business Suite (© PROMATIS)

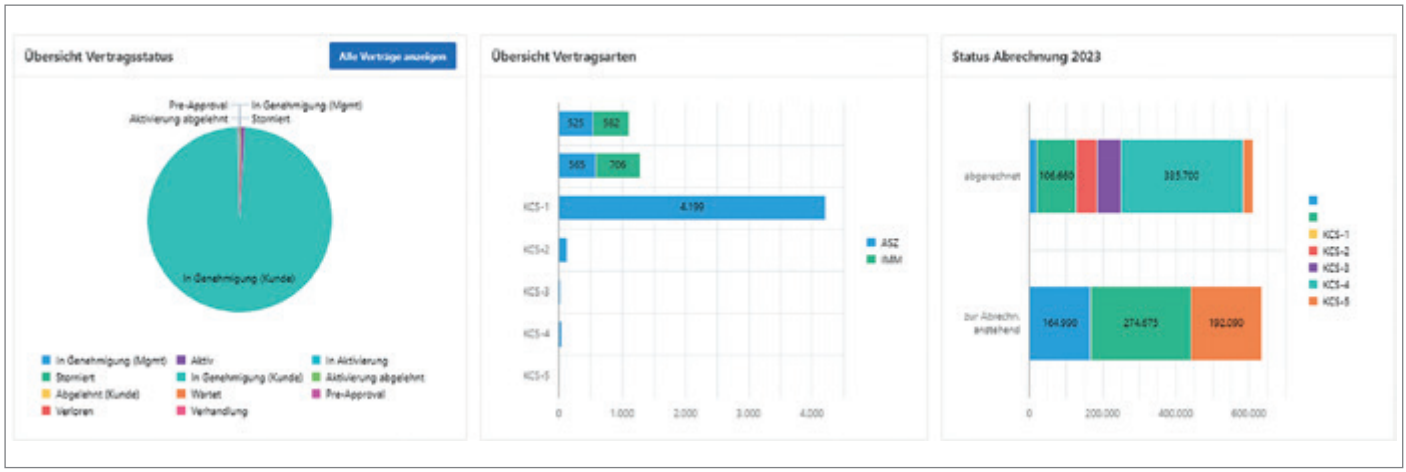


Abbildung 6: Beispiel eines einfachen Dashboards in einem der internen Portale (© PROMATIS)

Kunden- und Vertragsdaten sollten weiterhin in Microsoft CRM und die dazugehörigen Dokumente in SharePoint abgelegt werden. Das Hauptbuch und Cash-Management werden weiterhin in SAP abgewickelt. Die Oracle E-Business Suite dient nach wie vor als zentrales System, welches Daten und Prozesse für alle Umsysteme zur Verfügung stellt.

Auf Grund dieser Anforderung hat sich die Interseroh+ dazu entschieden, ihre IT-Landschaft rund um die Oracle E-Business Suite neu zu denken, die in der Oracle EBS für den Betrieb und Vertrieb implementierten Prozesse von Grund auf zu erneuern sowie die internen und externen Portale mit Oracle APEX umzusetzen.

Die Hauptanforderung war es, hierbei eine einheitliche und datenbanknahe Portalinfrastruktur über verschiedene Portale hinweg zu realisieren. Dies konnte mit Oracle APEX, ein leichtgewichtiges Framework zur Erstellung von Webapplikationen von Oracle, umgesetzt werden. Es wird für alle neuen Portale – intern wie extern – bei der Interseroh+ eingesetzt. Durch diese strategische Entscheidung können neue Portale schnell implementiert und einfach an die Oracle E-Business Suite angebunden werden. Hierbei bietet Oracle APEX die Möglichkeit, viele vorgefertigte Prozesse und Elemente, wie Formulare und Reports, in die Webseiten einzubinden.

Technische Umsetzung

Bei der technischen Umsetzung werden viele Prozesse und Masken in Portale ausgelagert, die sich in zwei Klassen unterteilen lassen – externe Portale und interne Portale *(siehe Abbildung 3)*: In der EBS-Datenbank befindet sich das interne APEX (links unten) in einem separaten Schema und greift von dort aus auf die Daten der EBS zu. Das externe APEX (rechts) befindet sich in einer eigenen Datenbank und greift über einen Datenbanklink auf die Daten der EBS zu. Beide Portale wurden mit Oracle APEX implementiert und über Schnittstellen an die Oracle E-Business Suite angebunden.

Externe Portale

Externe Portale sind Portale, welche für die externen Partner (Kunden und Lieferanten) bestimmt sind. Diese Portale sollen nicht nur einfach zu bedienen, sondern auch optisch ansprechend sein sowie die Farben und das Design der Interseroh+ widerspiegeln. Ein Bespiel für ein externes Portal ist das neue Kundenportal der Interseroh+. Hier können Kunden Verträge rund um die Verpackungslizenzierung in Deutschland abschließen und verwalten.

Die Infrastruktur der externen Portale ist physisch von den internen Portalen und der Oracle E-Business Suite getrennt *(siehe Abbildung 3)*. Die Portale werden auf einer eigens dafür in der Oracle Cloud zur Verfügung gestellten Oracle-Datenbank betrieben. Der Datenaustausch zwischen den externen Portalen und der Oracle E-Business Suite erfolgt direkt zwischen den beiden Datenbanken über einen Datenbanklink.

Ziel der externen Portale war unter anderem auch eine vollständige Verfügbarkeit zu gewährleisten, wenn das ERP-System der E-Business Suite zum Beispiel auf Grund von geplanten oder ungeplanten Wartungsarbeiten nicht zur Verfügung steht. Dies wurde durch die Implementierung einer mehrstufigen Schnittstelle zwischen den beiden Datenbanken erreicht.

Die Schnittstelle, welche Daten aus der Oracle E-Business Suite extrahiert und in den externen Portalen zur Verfügung stellt, besteht aus zwei Schichten von Materialized Views, welche die Daten zwischenspeichern *(siehe Abbildung 4)*. Die erste Schicht befindet sich dabei noch innerhalb der EBS-Datenbank. Hier werden die Daten, welche von den externen Portalen benötigt werden, vorselektiert, aggregiert und gefiltert. Anschließend werden die Daten durch die zweite Schicht abgeholt und zwischengespeichert. Die zweite, sich in der Datenbank der externen Portale befindende Schicht, sorgt dafür, dass die Daten aus der EBS zur Verfügung gestellt werden, auch wenn diese nicht erreichbar ist.

Für die Gegenrichtung zum Übertragen der Daten an die Oracle E-Business Suite, werden die Daten in einer Tabelle in der Datenbank der externen Portale zwischengespeichert *(siehe Abbildung 5)*. Ein eingeplanter Prozess versucht regelmäßig die Daten an die Oracle E-Business Suite zu übertragen und merkt sich dabei, welche Datensätze erfolgreich übertragen wurden und welche auf Grund von Fehlern bei der Übertragung – beispielsweise, weil die EBS nicht erreichbar war – einen erneuten Versuch benötigen.

Durch diese Architektur der Schnittstellen lässt sich sicherstellen, dass die externen Portale verfügbar bleiben, auch wenn an der Oracle E-Business Suite Wartungsarbeiten durchgeführt werden.

Interne Portale

Mit Hilfe von internen Portalen wickeln die Mitarbeitenden der Interseroh+ das Vertragsmanagement und die Überwa-

chung der Prozesse ab. Diese Portale wurden ebenfalls mittels Oracle APEX implementiert. Im Gegensatz zu den externen Portalen, spielt bei den internen Portalen das grafische Design eine untergeordnete Rolle. Hier wurde mehr Wert auf die angezeigten Daten gelegt. Ein weiterer Unterschied der internen Portale zu den externen Portalen, ist die Architektur *(siehe Abbildung 3)*.

Anders als die externen Portale, werden die internen Portale direkt in der Datenbank der Oracle E-Business Suite implementiert. Dies gibt den internen Portalen die Möglichkeit, direkt lesend und schreibend auf die Daten und Prozesse der Oracle EBS – wie beispielsweise Kundendaten oder Rechnungsprozesse – zuzugreifen. So können zum Beispiel direkt aus den Portalen heraus Abrechnungen gestartet oder deren Status eingesehen werden.

Mittels der in Oracle APEX verfügbaren Charts *(siehe Abbildung 6)*, konnten übersichtliche und nützliche Dashboards erstellt werden, die die Vertragszahlen für die Mitarbeitenden und das Management aufarbeiten und interaktiv zur Verfügung stellen. Darüber hinaus wurden mittels APEX Interactive Reports diverse Auswertungen implementiert, um die Daten aller Verträge genauer einsehen zu können. Dies ermöglicht den Mitarbeitenden sehr leicht einen detaillierten Zugriff auf alle relevanten Vertragsdaten zu erhalten.

Zusammen mit den vielen Möglichkeiten, Daten in Oracle APEX aufzubereiten und Formulare zu implementieren, konnte das gesamte Vertragsmanagement des Interseroh+ Vertriebs auf Oracle APEX umgestellt werden.

Fazit

Das duale System Interseroh+ hat es durch die strategische Neuausrichtung ihrer ERP-Lösung sowie den internen und externen Portalen geschafft, die Prozesse im Betrieb und Vertrieb zu revolutionieren. Mit den neuen Lösungen konnten viele neue Standards implementiert und Arbeitsabläufe besser automatisiert werden. Durch die vielen Möglichkeiten in Oracle APEX, Daten aufzubereiten und anzuzeigen, hat es der Interseroh+ Vertrieb geschafft, alle vertraglichen Kennzahlen und Prozessschritte in ihren neuen Portalen abzubilden. Dieser Erfolg stellt einen neuen Meilenstein in der Digitalisierung des Betriebs und Vertriebs der Interseroh+ dar.

Referenzen

[1] Interseroh (Hrsg.) (2018): Duales System Interseroh – Verkaufsverpackungen lizenzieren und optimal verwerten. Interseroh, Köln. *https://www.interzero.de/fileadmin/PDF/Broschueren_und_Informationsmaterial/190912_RZ_Neugestaltung_DSI_Broschuere_2018.pdf*, letzter Zugriff am 12.12.2023 um 15:23 Uhr.

Eva Reil
eva.reil@interzero.de

Eva Reil ist Head of Application Management bei Interzero Business Solutions und verantwortlich für die Systemarchitektur, die Integration der Geschäftsprozesse und den täglichen Betrieb im Bereich der Oracle E-Business Suite, einschließlich Oracle SOA Suite, OCI und APEX. In den 6 Jahren, in denen Eva bei Interzero tätig war, hat sie mehrere Projekte geleitet, darunter die Migration der E-Business Suite in die Oracle Cloud. Vor ihrer Tätigkeitbei bei Interzero Business Solutions war Eva über 15 Jahre lang als unabhängige Beraterin tätig und spezialisierte sich aufOracle-E-Business-Suite-Implementierungen und -Upgrades, Customizations und Datenmigrationen für globale Konzerne.

Johannes Michler
johannes.michler@promatis.de

Johannes Michler ist Senior Principal Consultant, Systemarchitekt und Projektleiter für die PROMATIS Gruppe mit Fokus auf serviceorientierte Architekturen (SOA), Web-Portale (insbesondere mit ADF und APEX) sowie Prozessautomatisierung. Als Mitglied im Management Board bekleidet er die Funktion „Executive Vice President – Head of Platforms & Development." Seit 2010 ist er für die DOAG als Referent und Autor mit wissenschaftlichen und anwendungsnahen Beiträgen aktiv. Er nimmt als Referent an zahlreiche Veranstaltungen der Oracle Community (IOUG & OATUG) teil und ist als „ACE Director" Teil der Oracle-ACE-Community.

Ann-Kathrin Denker
ann-kathrin.denker@interseroh.com

Ann-Kathrin Denker ist Leiterin Vertrags- und Systemmanagement beim dualen System Interseroh+. Sie ist seit mehr als 12 Jahren für die Interzero-Gruppe tätig und hat während dieser Zeit vorrangig strategisch relevante Themen als Projektleiterin verantwortet. Dazu zählt auch das IT-Projekt PUSH.IT zur Einführung der neuen IT-Systemlandschaft im dualen System.

Simon Grossmann
simon.grossmann@promatis.de

Simon Grossmann ist seit 2015 bei der PROMATIS Gruppe und besitzt als technischer Consultant mehrjährige Erfahrung in der Java-Anwendungsentwicklung und der Realisierung prozessorientierter Informationssysteme mit Oracle-Komponenten sowie fundierte Kenntnisse im Einsatz moderner Web-Technologien (APEX, JEE) in Verbindung mit leistungsfähigen Software-Entwicklungsumgebungen, Application-Servern und Service-orientierten Architekturen (SOA).

Weg mit dem Speck – Verschlankung der inneren Unternehmensstrukturen mit Oracle APEX

Yves Chassein, PROMATIS GmbH

Die Digitalisierung ist nicht nur ein Trend ist, sondern bereits in der Unternehmensrealität angekommen. In dem Zuge stellen sich Unternehmen „großen" Herausforderungen, wie beispielsweise die Einhaltung der Lieferkettensorgfaltspflicht oder auch dem Anspruch an konzernumfassende, homogene und transparente Finanzdaten. Dabei zeigen gerade die kleinen, aber feinen Innovationen, welche Vorteile die Digitalisierung für Unternehmen bietet. PROMATIS ist Experte für komplexe und große Business-Lösungen, immer mit dem Ziel, den Kunden die perfekte Anwendung bereitzustellen. Das umfasst sowohl Applikationen als auch die Bereitstellung technischer Komponenten, die dann zu einer ganzheitlichen und vor allem auf die Bedürfnisse des Unternehmens zugeschnittenen Lösung führen. Diese Praxisorientierung ist fest in der Organisation verankert und basiert auf der Sensibilität für neue Herausforderungen sowie einer Kreativität in der Realisierung. Und so entstanden mit Hilfe von Oracle APEX neue Lösungen, die für eine Verschlankung, eine Vereinfachung innerhalb der Strukturen sorgen. Im Folgenden werden zwei dieser Applikationen vorgestellt.

Was ist Oracle Application Express?

Oracle Application Express (APEX) ist eine Low-Code-Plattform, die mit jeder Oracle Datenbank verfügbar ist. Sie eignet sich zur Entwicklung und Bereitstellung von datenbanknahen Web-Applikationen. Neben „Stand-Alone"-Anwendungen können auch Erweiterungen für die Oracle E-Business Suite sowie für Oracle Fusion erstellt werden.

Warum Oracle Application Express?

Als Oracle Partner haben wir die Vorzüge von APEX schon in zahlreichen Kundenprojekten kennenlernen dürfen. Deswegen war auch die Entscheidung nicht schwer, unsere internen Applikationen mit Hilfe von APEX zu entwickeln. So hat APEX in unserer eigenen Systemlandschaft über die letzten Jahre hinweg immer mehr an Stellenwert gewonnen. Aus diesem Grund haben wir unsere digitale Lohnabrechnung sowie den Abwesenheitsmanager mit APEX erstellt.

Digitale Lohnabrechnung

Wer kennt sie nicht, die Lohnabrechnung, ein Dokument, auf das wir jeden Monat sehnlichst warten. Viele Informationen sind darauf erfasst, streng nach den Vorgaben des §108 der Gewerbeordnung (GewO), die auch fordert, dass die Abrechnung in „Textform" dem Arbeitnehmenden zur Verfügung gestellt werden muss. Das bedeutet aber nicht, dass Papierform zwingend erforderlich ist, was somit die Tür zum digitalen Gehaltszettel öffnet. Doch der Weg hierzu ist nicht so einfach wie es scheint. Zwar nutzt ein Großteil der Unternehmen eine HR-Software für den Bereich der Personalverwaltung, doch der nächste Schritt fehlt oftmals – bis jetzt, denn die Digitalisierung „digitalisiert" unaufhaltsam. So verzeichnete DATEV, einer der größten europäischen Dienstleister im Bereich der Steuerberater, innerhalb der letzten drei Jahre eine Vervierfachung der rein elektronischen Zustellung der Entgeltdokumente. Die Vorteile sind allen ersichtlich. Das Ausdrucken, Kuvertieren und Versenden der Abrechnungen entfallen, was Zeit, Kosten und Ressourcen einspart. Bei der elektronischen Zustellung müssen jedoch einige Punkte hinsichtlich eindeutiger Zuordnung und Gewährleistung der Datensicherheit berücksichtigt werden.

Dieser Herausforderung stellen wir uns auch täglich. Dank einer etablierten HR-Lösung erfolgt die gesamte Personalverwaltung digital. Doch der Medienbruch zum Gehaltszettel aus Papier bestand bis dato. Bis zur Einführung der APEX-Lösung wurde der Gehaltszettel ausgedruckt, kuvertiert und verteilt *(siehe Abbildung 1)*.

Um hier Abhilfe zu schaffen, wurde eine APEX-Anwendung entwickelt. Diese stellt ein internes Portal – das den hohen Sicherheitsstandards der PROMATIS entspricht – bereit und bietet für jeden Mitarbeitenden einen persönlichen Bereich, der alle relevanten Dokumente komfortabel auflistet und einfach zu nutzen ist. Hier werden unter anderem die Gehaltsabrechnungen bereitgestellt. Darüber hinaus wird jeder Mitarbeitende via E-Mail über einen neuen Eintrag informiert *(siehe Abbildung 2)*.

Kernstück dieser Anwendung ist das Importieren der Gehaltsabrechnung. Hierzu wurde beim Export aus dem Lohn- und Gehaltssystem ein definiertes Format genutzt, das neben dem Dateityp („Lohnabrechnung") den Vor- und Nachnamen des Mitarbeitenden enthält, dem die Lohnabrechnung zugeordnet werden muss. Der Dateiname sieht dann wie folgt aus: *Lohnabrechnung_2021_September_1111_Mustermann_Max.pdf*

Beim Import kann der Aufbau des Namenschemas durch einen Mitarbeitenden der Personalverwaltung angepasst werden. Dies wird ebenfalls in der APEX-Anwendung konfiguriert *(siehe Abbildung 3)*.

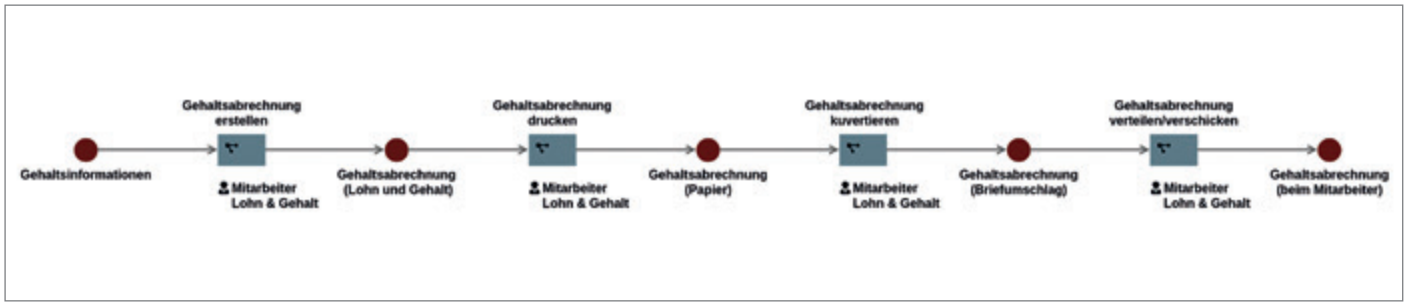


Abbildung 1: Ablauf der Lohnabrechnung vor der Digitalisierung (Quelle: PROMATIS)

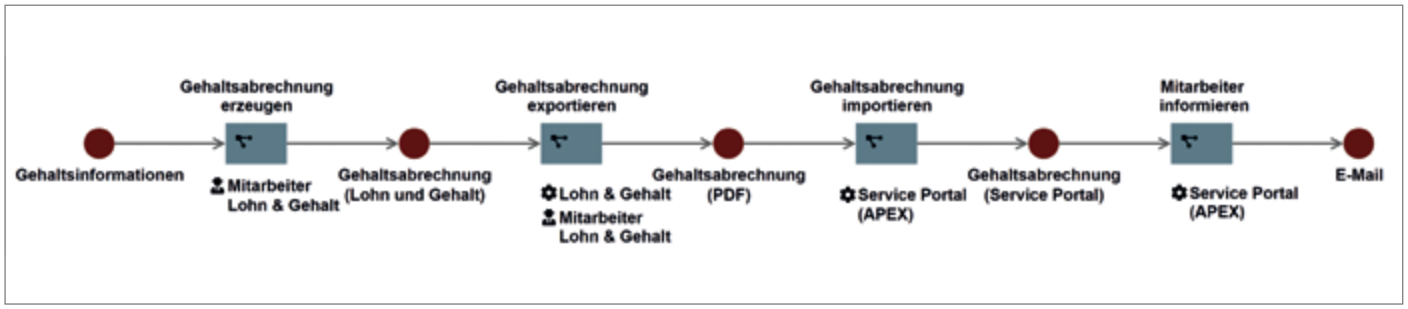


Abbildung 2: Ablauf der Lohnabrechnung nach der Digitalisierung (Quelle: PROMATIS)

Abbildung 3: Upload Definition für das Importieren von Gehaltsabrechnungen (Quelle: PROMATIS)

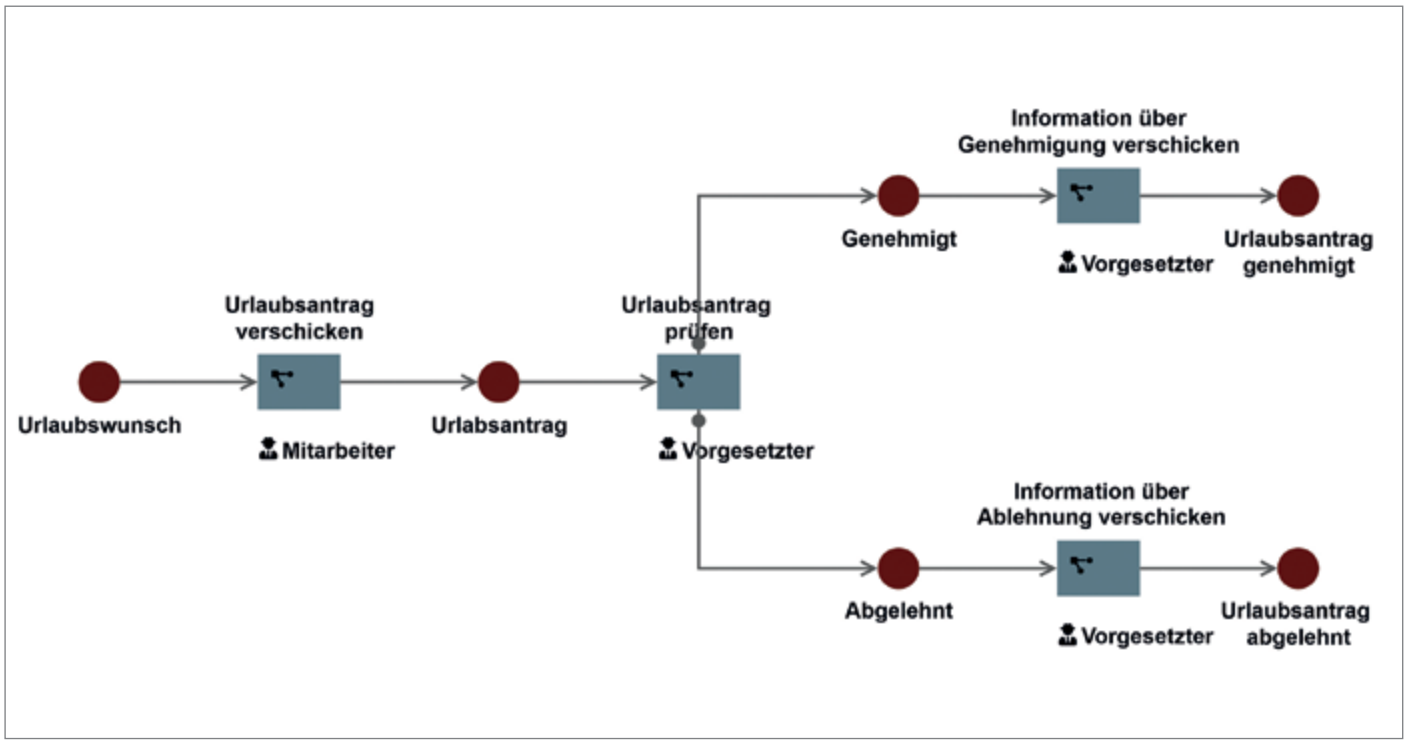


Abbildung 4: Urlaubsantragsprozess vor der Digitalisierung (Quelle: PROMATIS)

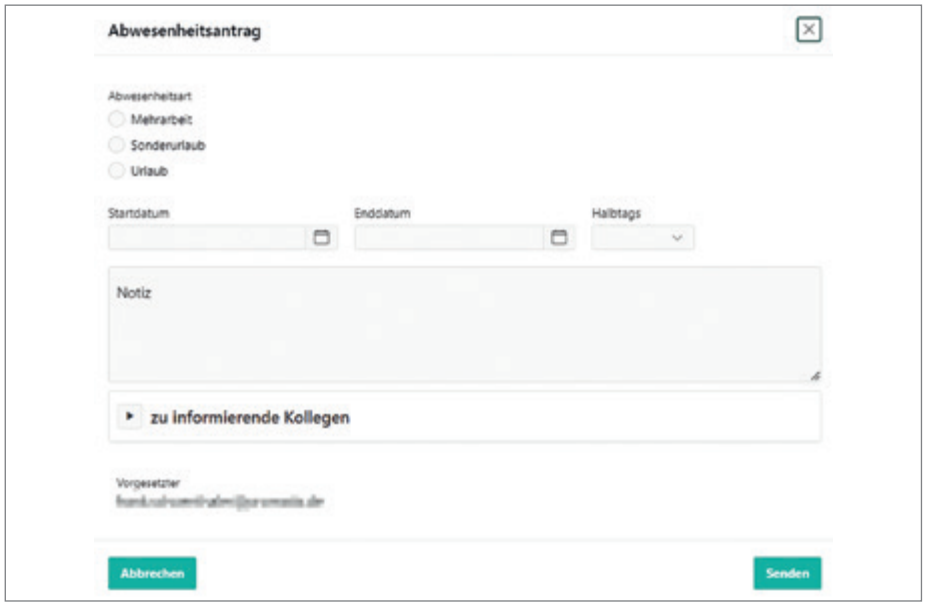


Abbildung 5: Maske zur Erfassung von Urlaubsträgen (Quelle: PROMATIS)

Abbildung 6: Urlaubsantrag aus APEX heraus (Quelle: PROMATIS)

Neben dem Namensschema können hier auch Besitzer sowie Ansprechpartner hinterlegt werden, die für die Gehaltsabrechnung zuständig sind. Diese Informationen kann der Mitarbeitende dann zusammen mit dem PDF in seiner persönlichen Ansicht einsehen. Die Zuordnung zum jeweiligen Mitarbeitenden wird mit Hilfe von Logik in PL/SQL abgebildet. Das Einlesen der Daten wird ebenfalls mit Hilfe von Standard-Boardmittel der Oracle Datenbank erreicht. Auf der einen Seite werden die zu verarbeitenden Daten mit Hilfe einer „External Table" mit „Preprocessor" erreicht. *Listing 1* zeigt die „Create Table"-Anweisung zur Erzeugung der Tabelle, die alle Dateien enthält, die importiert werden sollen.

Auf der anderen Seite wird dann per UTL_FILE auf die Dateien zugegriffen und diese als BLOB in die Datenbanktabelle geladen. Der Versand der E-Mail wird abschließend mit Hilfe der E-Mail-Templates und dem PL/SQL Package APEX_MAIL durchgeführt. Diese Innovation fand in allen Bereichen große Zustimmung und Akzeptanz, denn der digitale Gehaltszettel ist nun von überall und jederzeit verfügbar, sogar mit der gesamten Historie der Abrechnungen. Doch auch die Personalverwaltung ist mehr als zufrieden, denn der monatliche Abrechnungs-Marathon gehört nun der Vergangenheit an.

Abwesenheitsmanager

Neben der Lohnabrechnung stellt das interne Portal auch eine Funktion zur Beantragung und Genehmigung von Urlaub zur Verfügung. Früher wurden diverse E-Mails zwischen Mitarbeitenden und Vorgesetzten ausgetauscht, bis daraus schließlich ein genehmigter Urlaub resultierte.

Bevor der Urlaubsprozess mit Hilfe von APEX digitalisiert wurde, schickte der Mitarbeitende im ersten Schritt eine E-Mail an seinen Vorgesetzten mit dem Wunsch nach

Abbildung 7: Genehmigungsmaske für Urlaubsanträge (Quelle: PROMATIS)

Abwesenheitsart	Startdatum	Enddatum	Halbtags	Notiz	Status	Ablehnungsgrund	Zuletzt geändert am
Urlaub	01.12.2023	07.12.2023			Beantragt		01.12.2023
Urlaub	10.11.2023	22.12.2023			Abgelehnt		10.11.2023
Urlaub	10.11.2023	17.11.2023			Genehmigt		10.11.2023
Urlaub	09.11.2023	10.11.2023			Gesamtstorniert		08.11.2023
Urlaub	09.11.2023	10.11.2023			Gesamtstorniert		08.11.2023
Urlaub	08.11.2023	09.11.2023			Gesamtstorniert		07.11.2023
Urlaub	29.09.2023	05.11.2023			Teilstorniert		07.11.2023
Urlaub	22.09.2023	30.09.2023			Genehmigt		22.09.2023
Urlaub	22.09.2023	29.09.2023			Teilstorniert		22.09.2023
Urlaub	21.09.2023	02.11.2023			Teilstorniert		21.09.2023
Urlaub	21.09.2023	30.09.2023			Teilstorniert		21.09.2023

Abbildung 8: Urlaubsantragsübersicht in APEX (Quelle: PROMATIS)

```
CREATE TABLE "PROM_SVC_PORTAL"."MASS_UPLOAD_FILES"
 (    "FILE_PERMISSIONS" VARCHAR2(20 BYTE),
  "FILE_HARDLINKS" VARCHAR2(10 BYTE),
  "FILE_OWNER" VARCHAR2(20 BYTE),
  "FILE_GROUP" VARCHAR2(20 BYTE),
  "FILE_SIZE" VARCHAR2(50 BYTE),
  "FILE_DATE_MONTH" VARCHAR2(20 BYTE),
  "FILE_DATE_DAY" VARCHAR2(20 BYTE),
  "FILE_DATE_TIME" VARCHAR2(20 BYTE),
  "FILE_NAME" VARCHAR2(255 BYTE)
 )
 ORGANIZATION EXTERNAL
  ( TYPE ORACLE_LOADER
    DEFAULT DIRECTORY "MASS_UPLOADS_SCRIPTS_DIR"
    ACCESS PARAMETERS
    ( RECORDS DELIMITED BY NEWLINE
          NOLOGFILE
          NOBADFILE
          PREPROCESSOR MASS_UPLOADS_SCRIPTS_DIR: 'list_files.sh'
          FIELDS TERMINATED BY ','
     )
    LOCATION
     ( 'location.txt'
     )
  )
 REJECT LIMIT UNLIMITED ;
```

Listing 1: „External Table" mit „Preprocessor"

Urlaub. Dieser Antrag wurde durch den Vorgesetzten geprüft und im positiven Fall mit einer Rückmeldung an den Mitarbeitenden sowie die Urlaubsverantwortlichen gesendet. Im negativen Fall wurde nur der Mitarbeitende über die Ablehnung unterrichtet. Der Prozess bestand aus vielen manuellen Schritten, die zudem auch noch diverse Freiheitsgrade enthielten, wie beispielweise die E-Mail des Urlaubsantrags, die an den Vorgesetzten geschickt wurde. Der alte Prozess ist in ist in *Abbildung 4* dargestellt. Mit Hilfe von APEX wurde eine Maske für die Erfassung von Urlaubsanträgen implementiert. Diese Maske ermöglichte eine Reduzierung der Freiheitsgrade zur Erfassung auf lediglich ein Notizfeld *(siehe Abbildung 5)*.

Nach Erfassung wird eine E-Mail an den Vorgesetzten mit allen Informationen versendet. Diese E-Mail wird direkt mit APEX und der eingebauten E-Mailfunktionalität verschickt und ist in *Abbildung 6* zu sehen.

Im Hintergrund wird durch die Eingabemaske ein „Task" im APEX Task Framework erstellt. Dieser ist die Klammer für den Urlaubsantrag und kann vom Vorgesetzen im nächsten Schritt genehmigt oder abgelehnt werden. Hierzu wird der Urlaubsantrag durch den Vorgesetzten geöffnet und entweder genehmigt oder abgelehnt *(vgl. Abbildung 7)*. In beiden Fällen wird eine E-Mail mit der Entscheidung verschickt.

Der Prozess selbst sieht dem „alten" Prozess immer noch ähnlich aus. Es wurden aber die Freiheitsgrade reduziert sowie eine Nachvollziehbarkeit geschaffen. Jedem Mitarbeitenden wird nun ein Bericht über die persönlich eingereichten Urlaubsanträge zur Verfügung gestellt und jeder Vorgesetzte hat neben diesem Bericht über die persönlichen Anträge auch eine Übersicht über die Anträge seiner Mitarbeitenden *(siehe Abbildung 8)*.

Mittels der Überführung des Urlaubsprozesses von E-Mail in APEX konnte der Prozess intuitiver und weniger fehleranfällig gestaltet werden. Durch die Reports hat man als Mitarbeitender sowie Vorgesetzter immer eine Übersicht aller offenen und bearbeiteten Anträge.

Fazit

APEX als Low-Code-Plattform eignet sich sehr gut zur Digitalisierung der Prozesse rund um den Mitarbeiteralltag. Egal, ob es die digitale Lohnabrechnung, der Urlaubsantrag oder sogar die Krankmeldung ist, mit Hilfe von APEX ist es möglich, diese Prozesse mitarbeiterfreundlicher zu gestalten, ohne eine große Suite, wie beispielsweise Oracle Fusion, einführen zu müssen. Neben dem Betrieb als isolierte Anwendung, können solche APEX-Anwendungen ebenso mit Dritt-Systemen über REST-Schnittstellen kommunizieren sowie Daten beziehen und auch bereitstellen. Mit all diesen Funktionen und Möglichkeiten ist APEX der perfekte Begleiter der Unternehmensdigitalisierung.

Yves Chassein
yves.chassein@promatis.de

Yves Chassein ist Vice President und Mitglied des Management Boards der PROMATIS Gruppe sowie Leiter des internen APEX-Entwicklungsteams. Er besitzt mehrjährige Erfahrung in den Bereichen Geschäftsprozesse, prozessorientierte Informationssysteme sowie Anwendungsentwicklung mit APEX, SQL, PL/SQL und Java. Chassein ist Spezialist in der Realisierung komplexer transaktionaler und analytischer Zusatzfunktionen für Oracle-Applikationen und führt die Planung und Umsetzung SOA-basierter Integrationskonzepte On-Premises und in der Cloud durch.

Der Oracle Optimizer – einfach erklärt

Klaus Reimers, Ordix

Der Optimizer ist das Kernstück einer jeden Datenbank. Das gilt auch für die Datenbank von Oracle. Die Aufgabe des Oracle-Optimizers besteht darin, einen effizienten Weg zum Ausführen von SQL-Anweisungen zu finden. Für jede auszuführende SQL-Anweisung erstellt der Optimizer Ausführungspläne. Im Folgenden wird erklärt, wie man zu einem Ausführungsplan kommt und wie ein Optimizer vorgeht.

Der Weg zu Ausführungsplan

Alle Wege führen nach Rom. So gibt es auch viele Möglichkeiten, um zu einem Ausführungsplan zu kommen *(siehe Abbildung 1)*.

Das geforderte SQL landet im Shared Pool (Library Cache – SQL Area). Vor der Ablage können Literale automatisch in Bind Variable verwandelt werden (Cursor Sharing). Der normale Ablauf ist dann der Weg durch den Optimizer, der den Ausführungsplan erzeugt. Dabei werden Parameter aus dem Parameterfile (spfile) sowie Statistiken aus der Datenbank gelesen und als Input für den Optimizer verwendet. Sollte der Plan nicht stabil sein, so kann man diesen über eine Stored Outline, eine SQL Baseline oder ein SQL Profile stabilisieren.

Parsen

Zunächst wird das SQL in der SQL Area in drei Schritten geparst.

- Syntaxcheck (formal korrektes SQL)
- Semantikcheck (inhaltlich korrektes SQL)
- Ausführungsplan Erzeugung

Ein Fehler im Syntaxcheck sollte in der Produktion nicht auftreten *(siehe Listing 1)*.

Im semantischen Check werden die Inhalte des SQL geprüft. Gibt es die definierten Tabellen (Mengen)? Sind die jeweiligen Spalten vorhanden? Gibt es Check-Constraints, die die Ausführung verhindern *(siehe Listing 2)*?

Ideal ist in den meisten Fällen eine Verwendung von Bind-Variablen, um einen Softparse anstelle eines Hardparse durchführen zu können *(siehe Listing 3)*.

In diesem Beispiel sieht man, dass das mittlere SQL mehrfach bei unterschiedlichen Werten ausgeführt worden ist.

Dynamic Sampling

Oracle erwartet für den Optimizer als Input Statistiken. Sind keine Statistiken vorhanden, so wird eine Schätzung durchgeführt.

In diesem Beispiel wurden 100.000 Sätze gelesen, laut Schätzung sollten es 98.842 Sätze sein *(siehe Listing 4)*.

Statistiken

Statistiken werden über das Package *dbms_stats* erzeugt *(siehe Listing 5)*.

Danach sind die erwarteten Sätze in dem in *Listing 6* dargestellten Beispiel (Gleichverteilung) korrekt.

Es wurden 90.000 Sätze gelesen und 90.001 Sätze geschätzt.

Die Statistiken sind in diversen Views zu finden.

- Tabellen
  - dba_tables
  - dba_tab_statistics
- Spalten
  - dba_tab_columns
  - dba_tab_col_statistics
  - dba_histograms
- Indizes
  - dba_indexes
  - dba_ind_statistics

Die einzelnen Inhalte und wichtigen Spalten dieser Views hier zu beschreiben, würde den Rahmen dieses Artikels sprengen.

Statistiken sind immer veraltet. Die Berechnung des Optimizers basiert also auf dem Stand der Erzeugung. Veränderungen an den Quantitäten der beteiligten Mengen werden nicht berücksichtigt.

Vorgehen des Optimizers

Der Optimizer versucht alle möglichen Pläne zu durchdenken und dann den vermeintlich besten Plan in Ausführung zu bringen. Jeder Plan wird mit Kosten ausgewiesen und der günstigste Plan wird genommen. Je mehr Mengen an dem SQL beteiligt sind, desto mehr mögliche Ausführungspläne gibt es, die der Optimizer durchdenken muss. Im in *Abbildung 2* gezeigten Beispiel werden zwei Mengen mit jeweils einem Index auf der Join-Bedingung gejoint. In diesem trivialen Fall muss der Optimizer schon 24 mögliche Pläne durchdenken. Da der Denkprozess bei komplexen SQLs zu lange dauern würde, bricht der Optimizer nach einer von uns (offiziell) nicht zu beeinflussenden Zeit ab. Das bedeutet also, je komplexer ein SQL ist, desto weniger der möglichen Pläne kann der Optimizer durchdenken.

Preferences

Die Statistiken werden über das Package dbms_stats erzeugt. Dabei gibt es viele Stellschrauben, mit denen die Erstellung der Statistiken beeinflusst werden kann. Dabei werden GLOBAL_PREFS, DATABASE_PREFS, SCHEMA_PREFS und TABLE_PREFS unterschieden. Auf vier Ebenen können somit die Inputs der Statistikerzeugung beeinflusst werden. Eine komplette Auflistung der Möglichkeiten würde den Umfang des Artikels sprengen, hier aber einige Beispiele.

Die hier ausgegebenen Werte werden jeweils über *dbms_stats.get_prefs* ausgegeben *(siehe Listing 7)*.

Mit `Options` wird definiert, mit welchem Verfahren das Package *dbms_stats* aufgerufen wird. `Estimate_Percent` bestimmt die Größe der Stichprobe, `Cascade` definiert, ob auch Indizes neue

```
select * from doag2023_small wher a = 4711;
select * from doag2023_small wher a = 4711
                                      *
ERROR at line 1:
ORA-00933: SQL command not properly ended
```

Listing 1: Fehler im Syntaxcheck

```
select * from doag2023_small where c = 4711;
select * from doag2023_small where c = 4711
                                       *
ERROR at line 1:
ORA-00904: "C": invalid identifier
```

Listing 2: Fehler im Semantikcheck

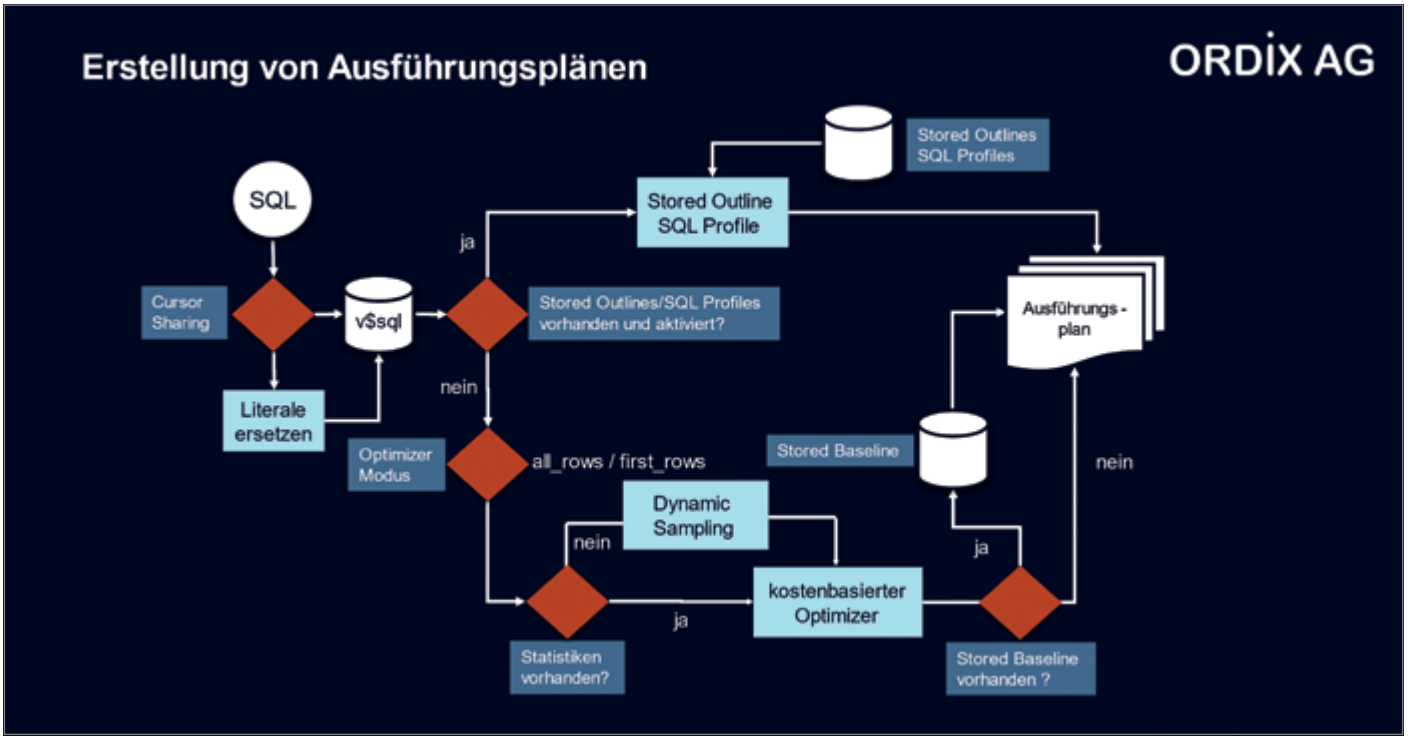


Abbildung 1: Wege zu einem Ausführungsplan (Quelle: ORDIX AG)

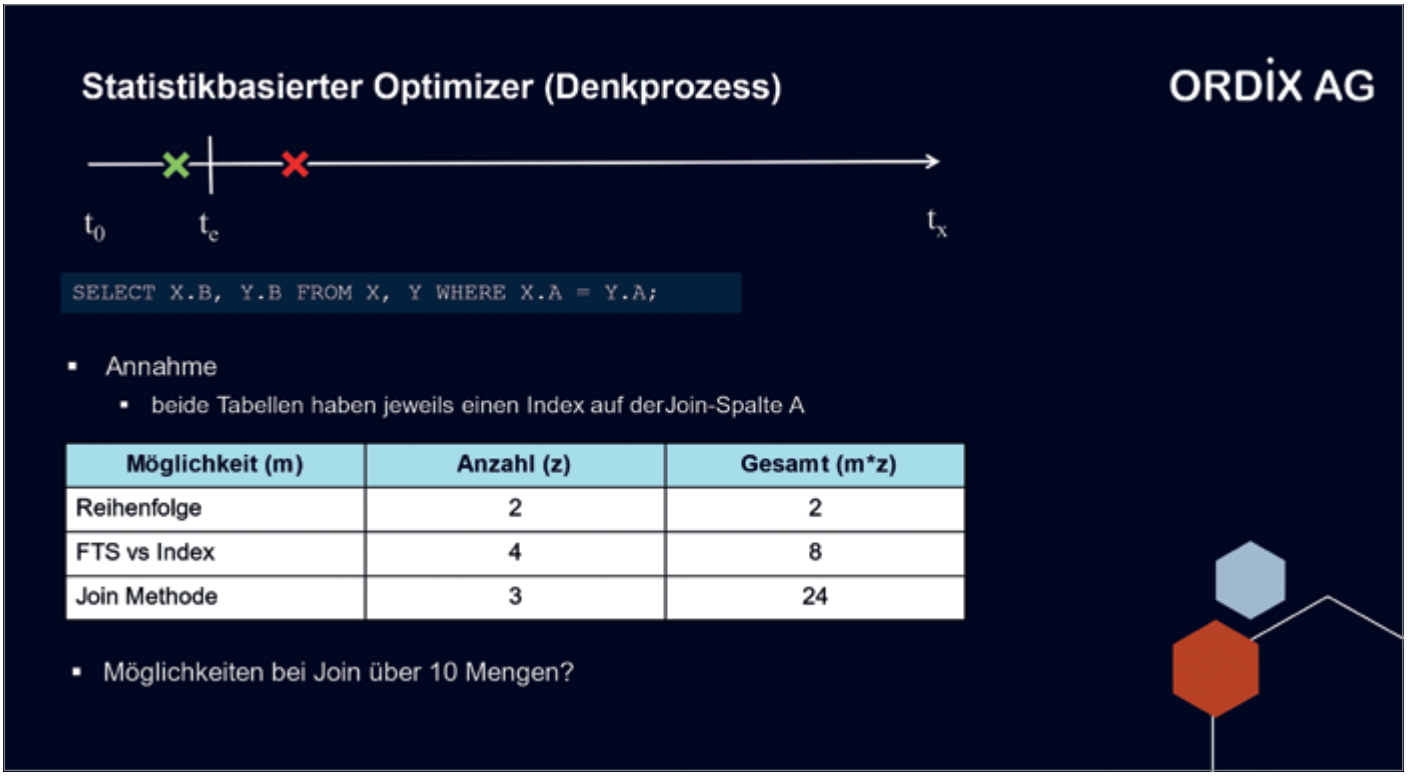


Abbildung 2: Denkprozess des Optimizers (Quelle: ORDIX AG)

```
EXECUTIONS SQL_ID                                 SQL_TEXT
------------------ ------------------------------------ ---------------
         1 7gh0k19b7w3ts   select a from doag2023_small where a = 4711
         2 d9m55a6tt11ud   select a from doag2023_small where a = :var_a
         1 4ghc1y1ag58z9   select a from doag2023_small where a = 4712
```

Listing 3: Hardparse/Softparse

```
select * from doag2023_big big where a > 0;
-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              | 98842 |    48M|  1947   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| DOAG2023_BIG | 98842 |    48M|  1947   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Listing 4: Dynamic Sampling

```
exec dbms_stats.gather_schema_stats(user);
```

Listing 5: Erzeugung der Statistiken für ein Schema

```
select * from doag2023_big big where a > 10000;
-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              | 90001 |    43M|  1947   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| DOAG2023_BIG | 90001 |    43M|  1947   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Listing 6: ohne Dynamic Sampling

```
OPTIONS    ESTIMATE_PERCENT             CASCADE                   METHOD_OPT
---------- ---------------------------- ------------------------- -------------------------
GATHER     DBMS_STATS.AUTO_SAMPLE_SIZE  DBMS_STATS.AUTO_CASCADE   FOR ALL COLUMNS SIZE AUTO
```

Listing 7: einige Preferences

```
select * from doag2023_big where a > 50000;
-------------------------------------------------------------------------------
| Id  | Operation         | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT  |              | 49413 |    23M|  1947   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| DOAG2023_BIG | 49413 |    23M|  1947   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Listing 8: Zugriff mit optimizer_index_cost_adj=100

```
select * from doag2023_big where a > 50000;
-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              | 49413 |    23M|  1816   (1)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DOAG2023_BIG | 49413 |    23M|  1816   (1)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | I_BIG        | 49413 |       |    50   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Listing 9: Zugriff mit optimizer_index_cost_adj=50

```
select * from doag2023_big where a > 0;
-------------------------------------------------------------------------------
| Id  | Operation          | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------
|   0 | SELECT STATEMENT   |              |   100K|    48M|  1947   (1)| 00:00:01 |
|*  1 |  TABLE ACCESS FULL| DOAG2023_BIG |   100K|    48M|  1947   (1)| 00:00:01 |
-------------------------------------------------------------------------------
```

Listing 10: Zugriff mit optimizer_mode=all_rows

```
select * from doag2023_big where a > 0;
-----------------------------------------------------------------------------------------------
| Id  | Operation                           | Name         | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                    |              |    10 |  5060 |     3   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID BATCHED| DOAG2023_BIG |    10 |  5060 |     3   (0)| 00:00:01 |
|*  2 |   INDEX RANGE SCAN                  | I_BIG        |       |       |     2   (0)| 00:00:01 |
-----------------------------------------------------------------------------------------------
```

Listing 11: Zugriff mit optimizer_mode=fist_rows_10

Statistiken bekommen sollen, wenn die Basistabelle eine neue Statistik erhält und über `Method_Opt` wird die Art der Histogrammerzeugung definiert. Alle Preferences haben von Oracle definierte Vorgaben und in vielen Fällen wird Oracle die interne Vorgehensweise überlassen (AUTO). Preferences können und sollten sehr individuell, passend zum jeweiligen Anwendungsfall angepasst werden.

Parameter

Die Parameter sind der zweite Input für den Optimizer und können den Ausführungsplan sehr stark beeinflussen. Es gibt etwa 20 Parameter aus der Parameterdatei (spfile), die den Optimizer beeinflussen können. Zusätzlich gibt es etwa 100 weitere nicht dokumentierte Parameter (Underscore Parameter). Auch hier gebe ich wegen der Komplexität nur zwei Beispiele.

Der Parameter `optimizer_index_cost_adj` definiert, wie „teuer" ein Indexzugriff im Rechenmodell des Optimizers sein soll. Der Standard ist 100 *(siehe Listing 8)*.

Der Ausführungsplan hat sich durch eine Minimierung des Wertes im Parameter vom Full Table Scan zum Index-Zugriff verändert *(siehe Listing 9)*.

Der Parameter `optimizer_mode` definiert, welcher interne Optimizer verwendet werden soll. Der Standard ist *all_rows (siehe Listing 10)*.

Auch hier ist zu erkennen, dass es bei einer Veränderung der Parameter zu komplett unterschiedlichen Ausführungsplänen kommen kann *(siehe Listing 11)*.

Stabilisierung von Ausführungsplänen

Ist ein SQL sehr komplex, so kann der Optimizer zu einem falschen Ergebnis kommen oder aber der Plan kann immer wieder kippen. Hier hilft dann häufig nur die Möglichkeit, den Ausführungsplan zu stabilisieren. Je nach vorhandener Edition und Option stehen unterschiedliche Möglichkeiten zur Verfügung. Stored Outlines sind seit Oracle 10g nicht mehr im Support, funktionieren aber immer noch ohne Probleme. Diese sind Teil der SE2. Bei der Verwendung von SQL Profiles wird die EE mit den Optionen Diagnostic Pack und Tuning Pack benötigt. SQL Baselines sind je nach Einsatz Teil der SE2 oder der EE mit den Optionen.

Weitere Informationen zum Thema Optimizer oder auch anderen Themen rund um Oracle finden Sie in unserem ORDIX Blog. *(https://blog.ordix.de/)*

Quellen

[1] PL/SQL Packages and Types Reference *https://docs.oracle.com/en/database/oracle/oracle-database/19/arpls/index.html#Oracle%C2%AE-Database*

Über den Autor

Klaus Reimers verfügt über langjährige Erfahrung als IT-Berater. Seit 1999 ist er bei der ORDIX AG beschäftigt. Als Principal Consultant ist Klaus Reimers mit dem Schwerpunkt Beratung und als ausgewiesener Experte im Datenbankumfeld unterwegs. Durch zahlreiche Vorträge auf der DOAG Konferenz- und Ausstellung ist er vielen Oracle-Nutzern bekannt. Zusätzlich zu seiner Beratertätigkeit ist Klaus Reimers auch als Referent/Trainer im ORDIX-Seminarzentrum tätig.

Klaus Reimers
kr@ordix.de

JavaLand

on demand

JavaLand 2023 verpasst?

Jetzt On-demand-Ticket buchen und Vortragsaufzeichnungen anschauen!

Alle Angebote im On-demand-Ticket-Shop

Präsentiert von: iJUG Verbund · Heise Medien · DOAG

Veranstalter: JavaLand

Ein Blick in die Black-Box: Explainable AI (XAI) erklärt

Verena Barth, Wüntsch & Barth, Tobias Goerke, viadee Unternehmensberatung

Machine-Learning-Modelle (ML-Modelle) finden in einem enormen Tempo Einzug in unseren Alltag und werden für Prognosen, datengetriebene Entscheidungen oder Generierung von Inhalten genutzt. Um die gewünschte Genauigkeit der Vorhersagen zu erreichen, werden statt menschenverständlicher Entscheidungssysteme oftmals komplexe Verfahren wie tiefe neuronale Netze (DNNs) verwendet, deren Entscheidungen aufgrund ihrer inhärenten Komplexität unverständlich und nicht nachvollziehbar sind. Der Bereich der Explainable AI (XAI) versucht, das Problem fehlender Transparenz von ML-Modellen zu adressieren und die Ergebnisse für den Menschen verständlich zu machen. In diesem Artikel wird das Potenzial von XAI erschlossen und einige exemplarische XAI-Methoden kurz vorgestellt. Im Anschluss werden Kriterien aufgeführt, die bei der Auswahl einer passenden XAI-Methode zu beachten sind. Abschließend werden allgemeine Handlungsempfehlungen für die Sicherstellung interpretierbarer ML-Modelle innerhalb eines Entwicklungs- und Deployment-Workflows gegeben.

Warum XAI? Die Black-Box-Problematik

Für komplexe Aufgaben oder zur Verarbeitung vieldimensionaler Eingabeparameter wie Bild- oder Textdaten, kommen häufig Black-Box-ML-Modelle zum Einsatz. Im Gegensatz zu White-Box-Modellen, wie zum Beispiel linearen/logistischen Regressionen oder kurzen Entscheidungsbäumen, besitzen sie meist einen effizienteren Lernalgorithmus oder erreichen eine höhere Genauigkeit der Vorhersagen durch das Finden komplexerer Entscheidungsgrenzen. Solche Black-Boxes, wie unter anderem Random Forrests, Support Vector Machines (SVMs) und DNNs, sind im Gegensatz zu intrinsisch interpretierbaren White-Box-Modellen aufgrund ihrer Komplexität nicht verständlich und ihre Entscheidungen daher nicht nachvollziehbar.

Trotz ihrer beeindruckenden Leistung zeigen viele Beispiele leider immer wieder die Unvollkommenheit bereits eingesetzter KI-Systeme auf. Diese reichen von geschlechterspezifischen Stereotypen bei der Verarbeitung natürlicher Sprache *[6]*, über die Benachteiligung von Frauen beim automatisierten Einstellungsprozess bei Amazon *[7]* bis hin zu rassistischen Tendenzen bei einem in den USA eingesetzten Algorithmus, der das Strafmaß durch die Vorhersage der Wahrscheinlichkeit einer erneuten Straftat bestimmt und dunkelhäutige Menschen benachteiligt *[2]*. Es existieren zahlreiche Sammlungen dokumentierter Fehler von in der realen Welt eingesetzten KI-Systemen; die AI Incident Database (AIID) hat bereits über 1200 Einträge riskanter Zwischenfälle erfasst *[17]*.

Diese Voreingenommenheit oder Diskriminierung des Modells kann vielseitige Ursachen haben wie unangemessene Evaluationsmetriken des Modells, falsche Annahmen über die Daten oder die Verzerrungen (Bias) in ihnen, die oft historische und sozio-technische Probleme der Gesellschaft widerspiegeln. *[14]*

Regierungen erkennen diese Problematik und fordern Transparenz in KI-Systemen. So stellt die Datenschutzgrundverordnung (DSGVO) Anforderungen an Systeme mit automatisierten Entscheidungsfindungen und der voraussichtlich 2026 inkrafttretende Artificial Intelligence Act (AI Act) ordnet KI-Anwendungen in Risikoklassen ein.

Angesichts bestehender Probleme und der zunehmenden produktiven Verwendung von ML-Modellen ist es dringend erforderlich, das Problem der mangelnden Transparenz und Nachvollziehbarkeit anzugehen.

Was ist Explainable AI und warum sollte man es anwenden?

Hier kommt Explainable AI ins Spiel: Dieses Forschungsfeld versucht die Interpretierbarkeit von und das Vertrauen in ML-Modelle zu fördern, ohne ihre (Lern-) Leistung einzuschränken. Eine Anwendung von XAI ist notwendig, da eine Erklärung, beziehungsweise Rechtfertigung, der Verhaltensweisen und Entscheidungen für die Nachvollziehbarkeit, Fairness und Sicherheit der Modelle essenziell ist.

Der Erhalt von Erklärungen der Vorhersagen ist zudem hilfreich bei der Fehlerdiagnose im Modell und bei der Identifikation von Verzerrungen in den Daten. Dadurch wird das Modell verbessert und robuster gegenüber Schwachstellen und Angriffen gemacht; außerdem unterstützt es bei der Erkenntnisgewinnung in der Problemdomäne. Damit (personenbezogene) Entscheidungen gerechtfertigt werden können, ist eine Nachvollziehbarkeit je nach Land und Domäne gesetzlich vorgeschrieben.

Erklärbarkeit und Rechtfertigung der Entscheidungen werden umso wichtiger, je komplexer das ML-Modell ist oder je kritischer der Anwendungskontext. Da transparente Modelle intrinsisch interpretierbar sind, können sie als „Explainable AI“ verstanden werden. Nachfolgend fokussieren wir uns auf XAI-Methoden für (Black-Box)-ML-Modelle beliebiger Komplexität, die nachträglich, post-hoc angewendet werden.

Taxonomie der XAI-Methoden

XAI-Methoden unterscheiden sich anhand der Modellart, auf die sie angewendet werden können, und anhand des resultierenden Erklärungsumfangs und -formats:

Es gibt modellspezifische und modellagnostische XAI-Methoden. Modellspezifische sind auf eine bestimmte Modellart beschränkt, während modellagnostische post-hoc, das heißt nach dem Training, auf jeden Modelltypen angewendet werden können. Da modellagnostische Methoden keinen Zugriff auf die Modellinterna haben, liefern sie Erklärungen nur anhand einer Analyse der Ein- und Ausgaben. *[16]*

XAI-Methoden können zudem nach dem Umfang der erhaltenen Erklärung klassifiziert werden, wobei zwischen globalen und lokalen Methoden unterschieden wird. Lokale Methoden erklären die Gründe einer Modellentscheidung spezifisch für eine einzelne Dateninstanz, die nicht auf eine globale Skala generalisiert werden können. Globale Erklärungen zielen dahingegen auf ein Verständnis des Verhaltens des Gesamtsystems ab.

XAI-Methoden lassen sich bezüglich ihrer Erklärungsansätze grob in vier Kategorien einteilen *[1]*:

1. Visuelle Erklärungen vereinfachen das Verständnis eines Modells durch Visualisierung und eignen sich dadurch auch für Menschen ohne Expertenwissen.
2. Eine Erklärung durch Feature-Relevanz quantifiziert den Einfluss jedes Eingabe-Features auf die Modellvorhersage, indem die Auswirkung der Änderung seines jeweiligen Werts auf das Vorhersageergebnis oder die Modell-Leistung (z. B. die Vorhersagegenauigkeit) beobachtet wird.
3. Bei einer Erklärung durch Wissensextraktion/Vereinfachung werden entweder Regeln für den Entscheidungsprozess unter der Verwendung der Ein- und Ausgaben konstruiert oder das Black-Box-Modell mit einem transparenten, interpretierbaren Modell approximiert.
4. Für eine beispielbasierte Erklärung und den Erhalt eines besseren Modell-Verständnisses werden repräsentative Dateninstanzen des Trainingsdatensatzes ausgewählt. Diese zeigen die inneren Beziehungen und Korrelationen des Modells auf.

Wie funktionieren XAI-Methoden?

XAI-Methoden operieren grundsätzlich mit einem von zwei Ansätzen:

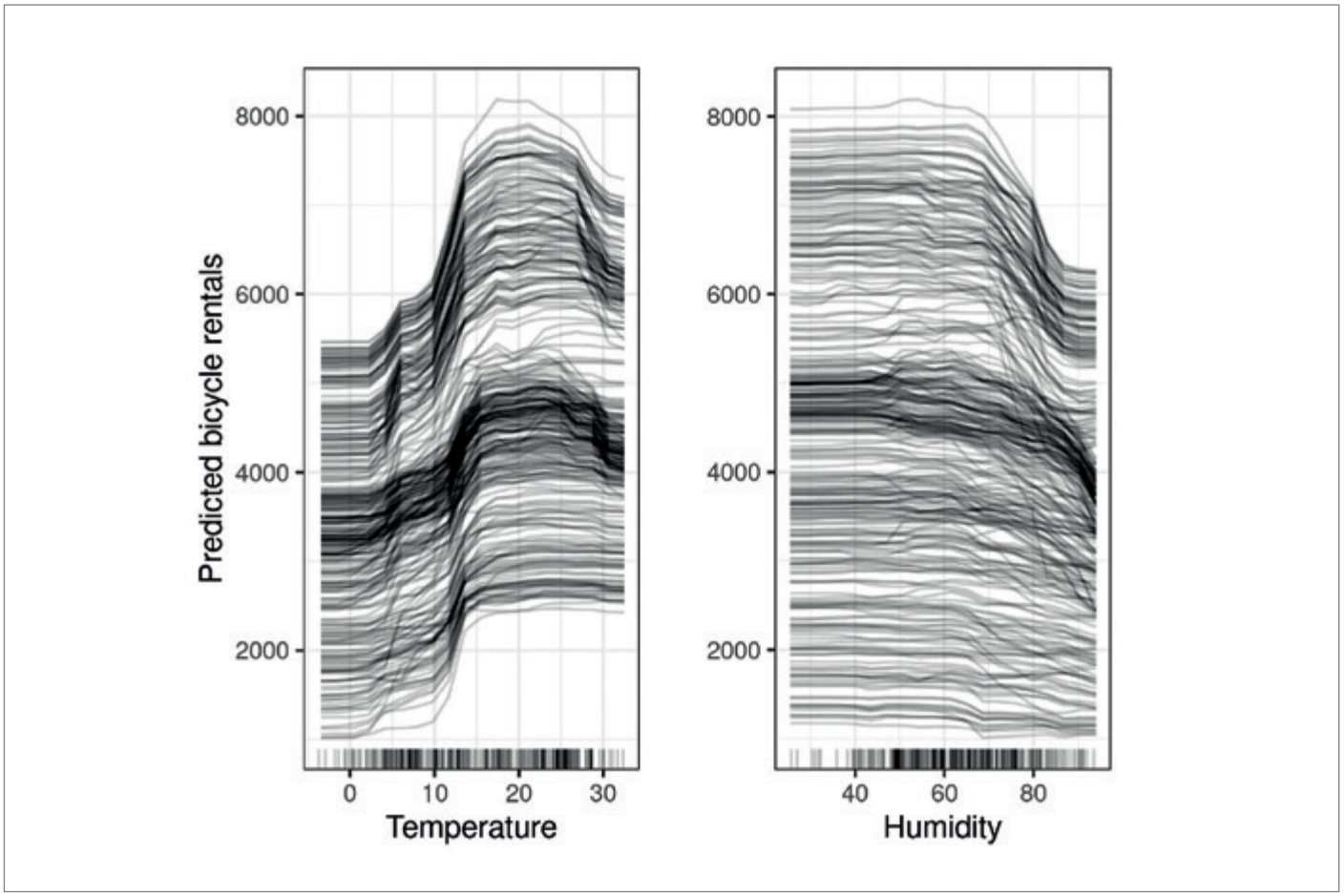


Abbildung 1: ICE der Features „Temperature“ und „Humidity“ am Beispiel des Bike Rental Datasets [16]

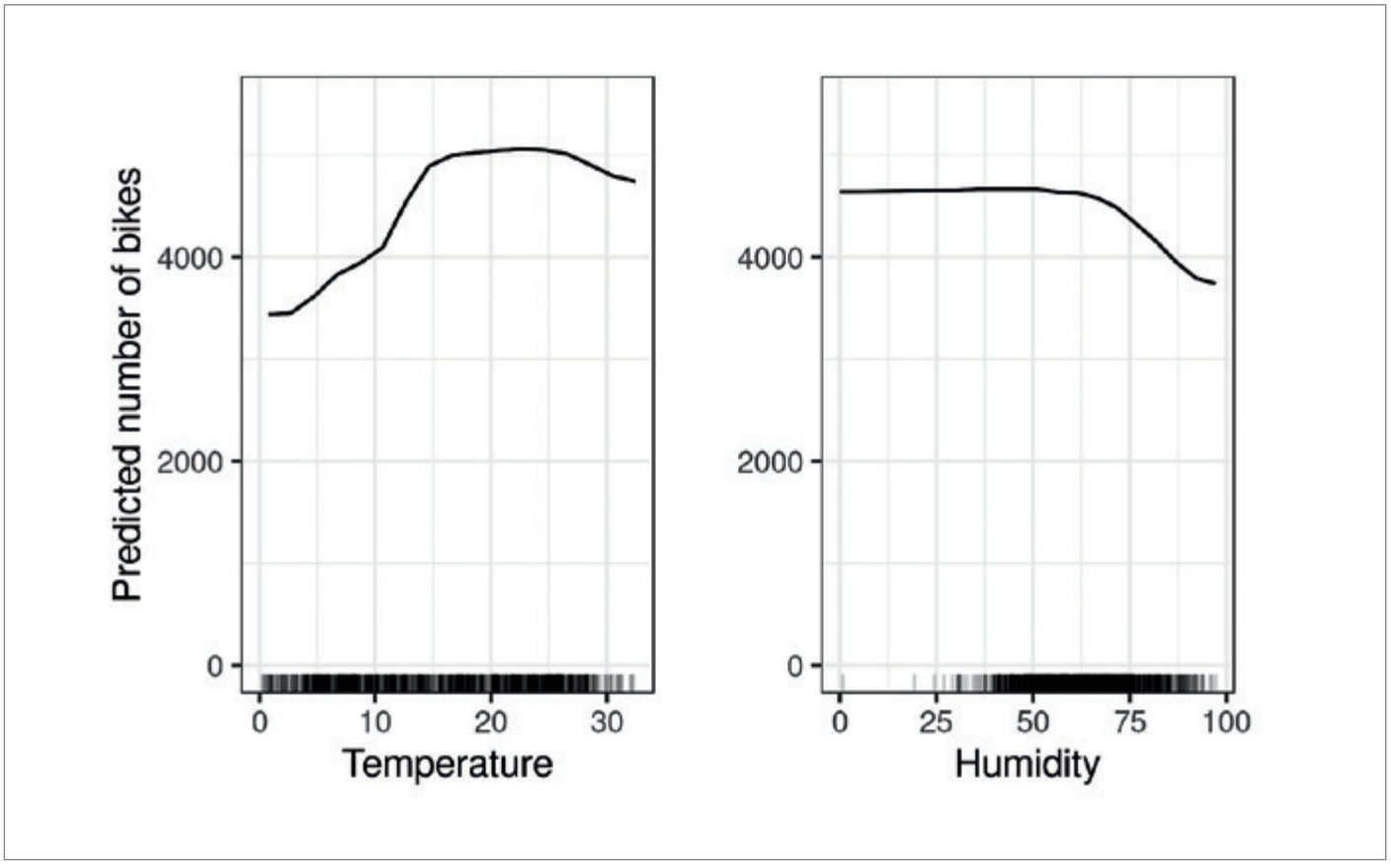


Abbildung 2: PDP der Features „Temperature“ und „Humidity“ am Beispiel des Bike Rental Datasets [16]

Gradientenbasierte Methoden verwenden eine Abwandlung des Backpropagation-Algorithmus und den Gradienten der Ausgabe in Bezug auf die Eingabe, um die wichtigsten Merkmale der Eingabe zu ermitteln. Dabei unterscheiden sie sich hauptsächlich durch die Art der Berechnung des Gradienten und werden vor allem für XAI bei Bilddaten in Form von Saliency Maps verwendet.

Perturbationsbasierte Methoden erklären die Vorhersage eines Modells durch verschiedene durch Perturbationen erzeugte Variationen des Eingabe-Feature-Raums (Störungsmodell). Einfach gesagt: Sie verändern die Modelleingabe und schauen, wie sich das auf die Vorhersage auswirkt. Da sie für die Erklärungsgenerierung keinen Zugriff auf interne Informationen des Modells benötigen, sind sie modellagnostisch und eignen sich für Black-Box-Modelle.

Um modellagnostisch zu bleiben, konzentrieren wir uns im Folgenden auf perturbationsbasierte Methoden.

XAI erklärt

Anhand des *Partial Dependence Plot (PDP)* und der Individual Conditional Expectation (ICE) wird die Funktionsweise perturbationsbasierter Methoden erklärt. Anschließend werden die beliebten, exemplarisch ausgewählten Methoden *Shapley Additive Explanations (SHAP)* und *Anchors* kurz vorgestellt und gezeigt, sie dazu beitragen, die „Black-Box-Natur" von KI-Modellen aufzubrechen und Einblicke in die Entscheidungsfindung zu gewähren. Sie verdeutlichen außerdem Probleme, unter welchen die Qualität der XAI-Erklärungen öfters leidet.

Die für tabellarische Daten anwendbaren XAI-Methoden Partial Dependence Plot und Individual Conditional Expectation erklären beide visuell die Entscheidung des Modells anhand der marginalen Einflüsse einzelner Eingabe-Features und beantworten somit die Frage: „Was ist der Zusammenhang zwischen dem betrachteten, unabhängigen Feature und der Vorhersage?"

PDP visualisiert den globalen Einfluss eines bestimmten Features (am Beispiel des Bike Rental Datasets *[8]*: *„Welchen durchschnittlichen Einfluss hat die Temperatur auf die Anzahl der vermieteten Fahrräder?"*), während sich ICE auf eine spezifische Dateninstanz fokussiert („Welchen Einfluss hatte die Temperatur vorgestern auf die Anzahl der vermieteten Fahrräder des Tages?"). Beide Methoden generieren Vorhersagen, indem sie den Wert des Features „Temperatur" verändern während alle anderen Feature-Werte unverändert bleiben. So wird der marginale Effekt isoliert und man erhält Aufschluss darüber, wie sich das Feature auf die Vorhersage auswirkt.

Der ICE-Plot *(siehe Abbildung 1)* zeigt ähnliche Auswirkungen der Temperatur auf die Fahrradvermietungen über verschiedene Tage, während der globale PDP-Plot *(siehe Abbildung 2)* den durchschnittlichen Effekt für alle Instanzen darstellt und den Trend von ICE bestätigt: Wenn die Temperatur über 15 Grad ist, werden mehr Fahrräder vermietet, wenn die Luftfeuchtigkeit hoch ist, nimmt die Anzahl an Vermietungen ab.

Da der Feature-Effekt isoliert betrachtet wird, sollten die Eingabe-Features nicht zu stark korrelieren, da andernfalls seltene, unrealistische (oder sogar unmögliche) Dateninstanzen bei der Erklärungserzeugung berücksichtigt werden könnten. Das mindert die Qualität der resultierenden Erklärung oder verfälscht diese. Ein Beispiel hierfür: Sollte die Dateninstanz eine Person darstellen, würde die Veränderung des Features „Gewicht" auf „41 kg" eine unmögliche Perturbation ergeben, wenn das Feature „Körpergröße" den Wert „200 cm" behält.

Eine weitere populäre XAI-Methode ist Shapley Additive Explanations, die auf einem häufig zitierten Paper *[13]* basiert und viele verschiedene Erklärungsformate bietet. SHAP beantwortet die Frage, welchen Beitrag der Wert eines oder mehrerer Features zur Vorhersage verglichen mit der durchschnittlichen Vorhersage liefert, zum Beispiel: *„Inwieweit wurde meine Vorhersage für die Kredithöhe durch die Tatsache beeinflusst, dass ich vier Bankkonten habe, statt nur der durchschnittlichen Anzahl von zwei Konten?"*

Die Methode basiert auf der spieltheoretischen Idee, dass das Vorhersageergebnis fair unter allen Features aufgeteilt wird und, dass für die Bestimmung der Wichtigkeit eines einzelnen Features (Shapley Werte), alle Feature-Kombinationen berücksichtigt werden sollten. SHAP generiert eine Vielzahl an Plots: So lassen sich sowohl fallspezifische als auch globale Feature Importance Plots (mit der durchschnittlichen Wichtigkeit einzelner Features) erstellen, indem die Shapley Werte einzelner Dateninstanzen kombiniert werden. Die *Abbildungen 3 und 4* zeigen Plots am Beispiel des Adult Census Income Datasets *[4]*, das Personen basierend auf ihren Charakteristika klassifiziert, ob sie mehr oder weniger als 50.000 US-Dollar pro Jahr verdienen. In *Abbildung 3* sind die für die Vorhersage wichtigsten Features aufgeführt (Feature Importance Plot) und man sieht, dass sie für die Klassifikation beider Klassen etwa gleich ausschlaggebend sind.

(Globale) Dependency Plots wie in *Abbildung 4* zeigen, welchen Einfluss der Feature Wert auf die Ausgabe des Modells hat. Sie sind ähnlich wie PDPs, wobei sie nur den möglichen Eingaberaum berücksichtigen (aber Korrelationen in den Eingabedaten trotzdem ignorieren). Durch die vertikale Varianz des Streudiagramms vermitteln sie außerdem die Größe der vorherrschenden Interaktionseffekte der Features. Eine Interpretation dieser visuellen Erklärung ist für Personen ohne datenwissenschaftliche Kenntnisse oftmals nicht trivial, da sie viele Informationen beinhaltet (wie zum Beispiel die Feature Werte, die Varianz der Datenpunkte eines Feature und die Wichtigkeit des Features auf die Vorhersage „Einkommen > 50.000").

Anchors *[18]* ist eine weitere beliebte perturbationsbasierte XAI-Methode, welche die kritischsten Bedingungen („Anchors") identifiziert, die erfüllt sein müssen, damit die Vorhersage eines Modells gültig ist. Sie zerlegt komplexe Modellentscheidungen in leicht verständliche Regeln, die auf möglichst viele Instanzen des Datensatzes zutreffen („Coverage") und dabei eine möglichst hohe Wahrscheinlichkeit aufweisen, dass ihre Erfüllung zu der prognostizierten Modellentscheidung führt („Precision").

„Education = Bachelors AND Relationship = Husband AND Occupation = Sales. Precision: 0.95, Coverage 0.02": Dieses Anchors-Beispiel einer Dateninstanz des Income-Datensatzes mit positiver Vorhersage (> 50 000 Einkommen) bedeutet, dass Personen, auf die diese Regel zutrifft, zu 95% auch eine positive Vorhersage erhalten werden. Leider ist diese Regel sehr spezifisch und trifft mit ei-

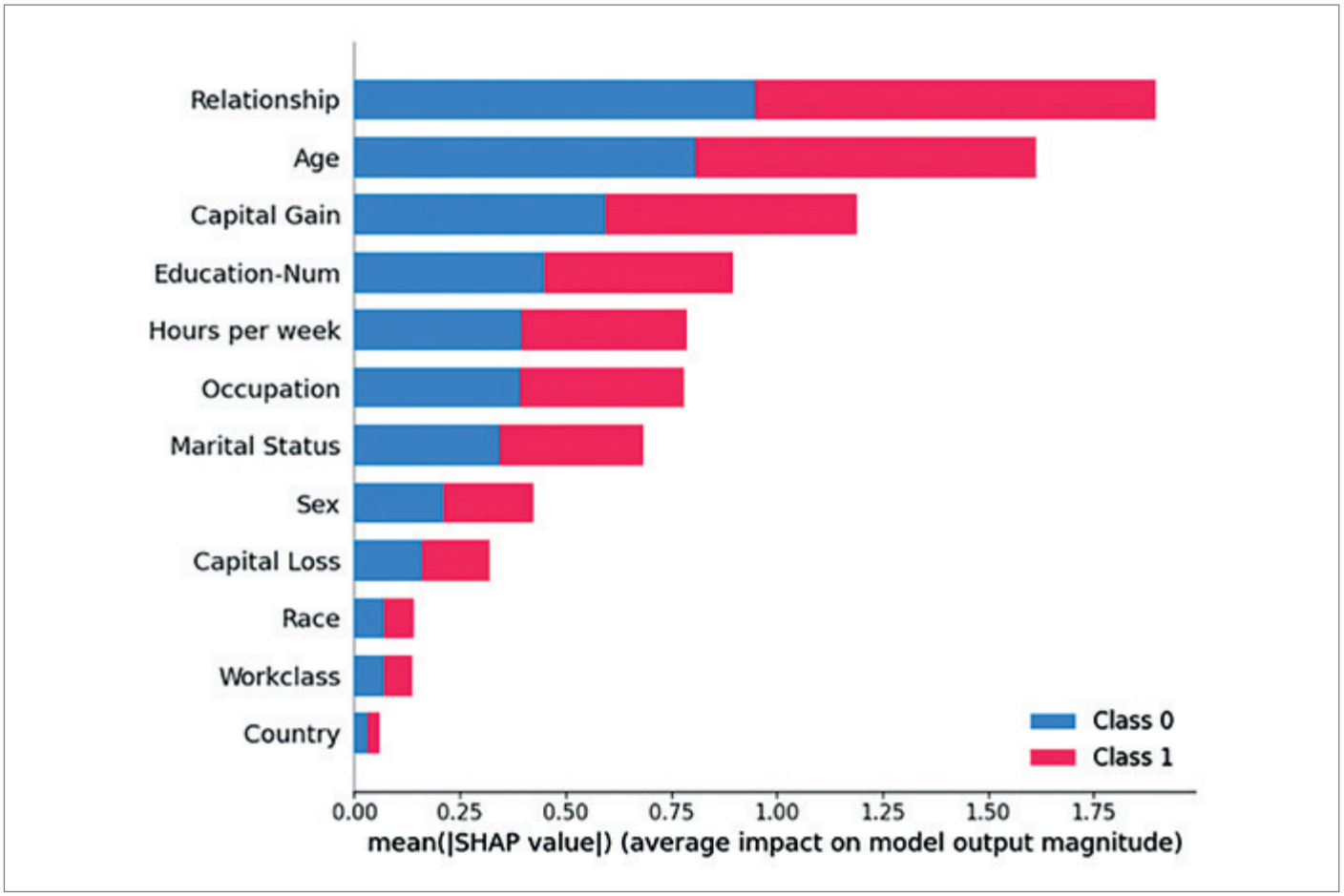


Abbildung 3: Feature Importance Plot am Beispiel des Adult Census Income Datasets (Quelle: Verena Barth)

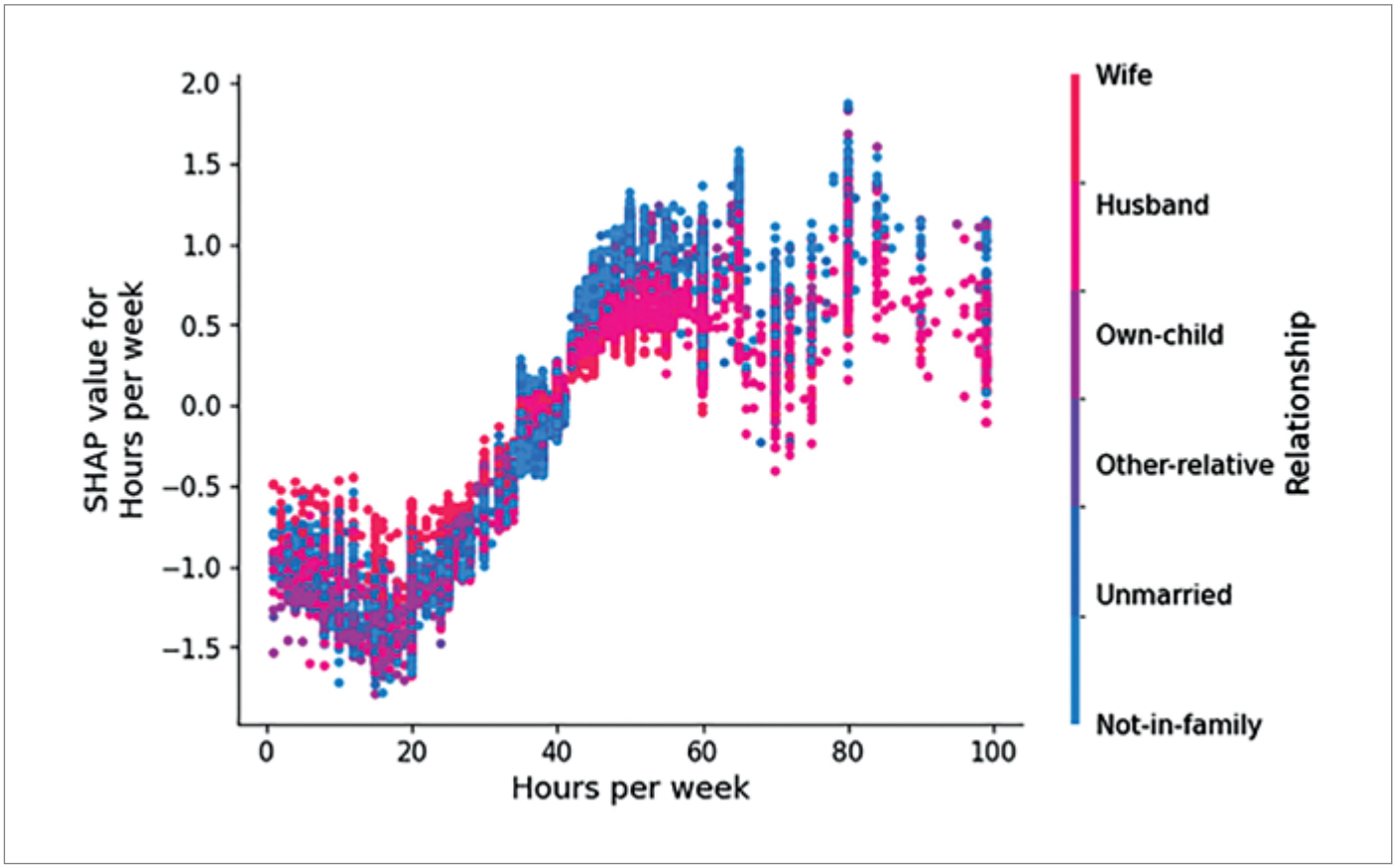


Abbildung 4: Dependence Plot am Beispiel des Adult Census Income Datasets (Quelle: Verena Barth)

ner Coverage von 0.02 nur auf 2% des Datensatzes zu. Eine allgemeingültigere Regel mit einer Coverage von 0.11 und einer trotzdem recht hohen Precision von 0.86 wäre: *„Education = Bachelors AND Relationship = Husband“*.

Worauf muss bei der Methodenauswahl geachtet werden?

Bei der Auswahl von XAI-Methoden sind einige wichtige Überlegungen zu berücksichtigen. Nach wie vor gibt es keine eindeutige Empfehlung, welche Methode(n) in welchem Kontext am besten geeignet ist/sind.

XAI-Methoden können je nach den Erklärungszielen verschiedener Benutzergruppen wie Stakeholdern, Experten und Nutzern ohne Fachwissen, zum Beispiel in Bezug auf Systemdebugging und -validierung, rechtlichen oder ethischen Anforderungen sowie der Vertrauensbildung zugeordnet werden *[10]*. Die Qualität der resultierenden Erklärungen variiert jedoch je nach Anwendungskontext, da sie kontextabhängig und durch individuelle Überzeugungen und kognitive Verzerrungen stark subjektiv ist, somit ist eine objektive Beurteilung schwierig *[15]*.

Die Güte einer Methode kann daher anhand von Eigenschaften des Modell-, Daten- und Nutzungskontexts definiert werden, die

- die Anwendung der XAI-Methode erschweren oder unmöglich machen,
- aufgrund der algorithmischen Beschaffenheit der XAI-Methode einen negativen, verfälschenden Einfluss auf ein solides, kohärentes und vernünftiges Erklärungsergebnis haben,
- die Interpretierbarkeit der Erklärung mindern oder verkomplizieren. *[3]*

Dadurch lassen sich Ausschlusskriterien ableiten, die eine Anwendung der Methode für ein gewisses Modell unmöglich machen (wie zum Beispiel die Art der ML-Aufgabe), und eignungsbeeinflussende Kriterien, die einen negativen Einfluss haben *(siehe Abbildung 5)*.

Ausschlusskriterien sind zum Beispiel die Art der ML-Aufgabe (manche Methoden sind nur für Klassifikationen geeignet) oder die Zugriffsmöglichkeit auf die Daten, die Klassenwahrscheinlichkeiten, die Labels oder Vorverarbeitungsschritte der Daten des zu erklärenden Modells (um beispielsweise Kategorie-Kodierungen nachvollziehen zu können).

Wie anhand von PDP verdeutlicht, muss man bei allen perturbationsbasierten Verfahren zudem darauf achten, dass die Korrelationen unter den Eingabefeatures nicht zu hoch sind, um verfälschte Erklärungen zu vermeiden.

Manche Methoden, zum Beispiel Anchors, verwenden für die Interpretierbarkeit der resultierenden „Wenn-Dann“-Regeln bei kontinuierlichen Features interne Diskretisierungsverfahren (Binning). Die Auswahl eines guten Verfahrens ist schwierig und stark von den Daten abhängig *[9]*; bei Anchors, das generalisiert anwendbar ist, ist es nicht auf die Daten abgestimmt. Bei sehr schiefen, beziehungsweise ungleichmäßigen Verteilungen können dadurch Entscheidungsgrenzen verschwimmen (Informationsverlust), was die Güte der Methode mindert *[16]*. Die Grenzen können dann entweder sehr nah beieinander liegen (zum Beispiel 29 <= „Age“ < 32) oder viel zu weit sein („Age“ > 48), sodass entstehende Erklärungen entweder zu spezifisch oder unpräzise sind.

Außerdem gibt es weitere weiche Kriterien wie Performance-Präferenzen der XAI-Methode, die abhängig von der Häufigkeit und der Performance der Modell-Aufrufe sowie der Anzahl der Features (je mehr Features, desto höher Berechnungsaufwand) ist. Die Anwendung vieler Methoden erzeugt außerdem einen gewissen Einarbeitungs- und Vorbereitungsaufwand.

Zusätzlich zu diesen sich auf die Anwendbarkeit der Methoden beziehenden Faktoren sollte immer auf den Kontext sowie das Daten- oder Domänenwissen des Rezipienten der Erklärung geachtet werden. Zudem sollten Erklärungsbedürfnisse und -ziele erfüllt und sichergestellt werden, sodass die Abbildungen nicht missinterpretiert werden.

Diese Kriterien können bei der Auswahl einer XAI-Methode helfen, aber …

Wie setzt man XAI erfolgreich ein?

Veröffentlichte Richtlinien zur Förderung der Transparenz von KI-Systemen weisen oft nur auf die Relevanz der Interpretierbarkeit hin, unterstützen allerdings nicht bei der konkreten Anwendung von XAI. Nachfolgend sind einige Richtlinien für eine Integration transparenzfördernder Maßnahmen in den Entwicklungszyklus zusammengefasst.

Sofern White-Box-Modelle die ML-Aufgabe mit einer guten Leistung meistern können, sind sie in jedem Fall Black-Box-Modellen vorzuziehen *[12]*. Falls nur ein komplexes Black-Box-Modell infrage kommt, empfehlen der XAI-Forscher Belle *(vgl. S. 15-17 [5])* und die Verantwortliche für Responsible AI Kangur *[11]* in ihren Richtlinien:

- Visualisierungstechniken sollten bereits bei der Datenexploration zur Vermeidung von verzerrten Daten angewendet werden. Diese sollten mindestens für alle wichtigen Features mit Diskriminierungspotenzial angewendet werden.
- Mit einem „local first“-Erklärungsansatz sollten anschließend lokale Methoden am besten auf richtig und falsch vorhergesagte Dateninstanzen angewendet werden, um zu sehen, wie sich kleine Veränderungen auf das Ergebnis auswirken. Gegebenenfalls kann so das Feature Engineering verbessert werden.
- Statt einer ausschließlichen Performancebetrachtung sollten auch globale Feature-Importance-Methoden angewendet werden, um die Vertrauenswürdigkeit des Modells sicherzustellen.
- Auch wenn das Modell in Betrieb ist, sollten die Eingabedaten kontinuierlich auf Änderungen geprüft und überwacht werden, um neuere Entwicklungen widerzuspiegeln, das Modell zu aktualisieren und somit Vertrauen in die Robustheit des Modells zu gewährleisten.

Anhand der Werke von Belle *(vgl. S. 16 [5])*, Kangur *[11]* und Leslie *(vgl. S. 45-46 [12])*, der momentan führenden Experten hinsichtlich XIA-Methoden, kann geschlossen werden, dass ein allgemeiner Konsens darüber besteht, dass eine Kombination mehrerer Erklärungstechniken förderlich für den Erhalt eines besseren Gesamtbilds des Modells und einer vollständigeren Erklärung sei.

Neben der Anwendung von XAI gibt es auch organisatorische Maßnahmen,

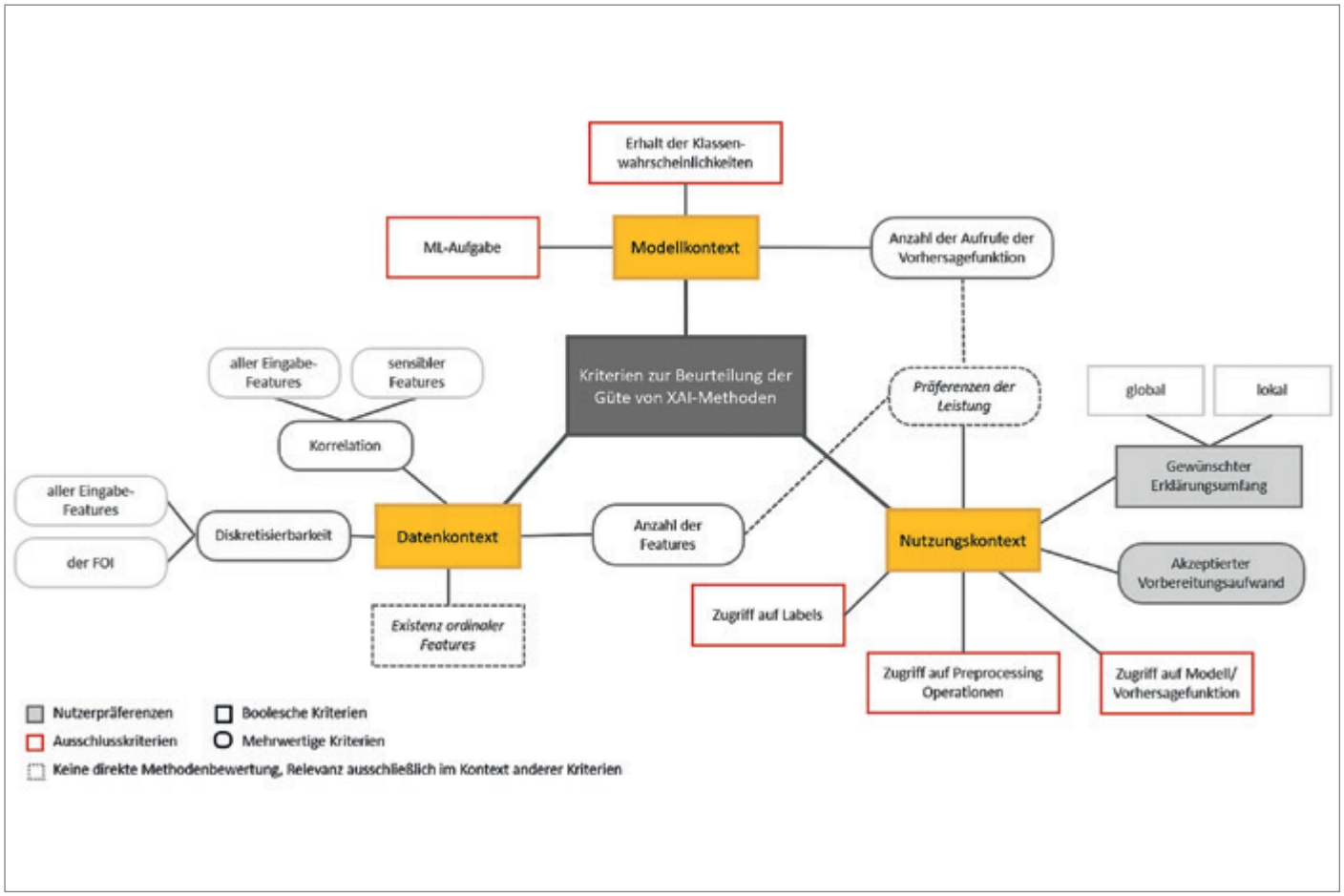


Abbildung 5: Übersicht der für die Beurteilung der Eignung einer XAI-Methode identifizierten Kriterien des Modell-, Daten- und Nutzungskontexts [3]

die zur ante-hoc-Vermeidung von Diskriminierung beitragen: Generell sollte schon bei der Teamzusammenstellung auf Diversität Wert gelegt werden, mit verschiedenen Personengruppen und Experten verschiedener Bereiche. Sie sollten ein gemeinsames Verständnis von Ziel, Kontext, den domänenspezifischen Anforderungen und Auswirkungen der ML-Aufgabe haben.

Hinsichtlich einer Gewährleistung der Nachvollziehbarkeit empfiehlt sich außerdem die Beachtung von MLOps-Praktiken, zum Beispiel der Dokumentation und Archivierung von (Meta-) Informationen von Datensätzen oder ML-Modellen in Model/Dataset Sheets. Es gibt diverse Tools/Plattformen, die Entwicklerinnen und Entwickler dabei unterstützen.

Zusammengefasst fördert eine Anwendung von XAI die essenzielle Interpretierbarkeit von ML-Modellen. Denn nur wenn Menschen maschinellen Entscheidern vertrauen, werden sie diese nutzen können. Mit Beachtung der oben genannten Richtlinien zur Anwendung von XAI und der aufgeführten Eigenschaften des Modell-, Daten- und Nutzungskontexts zur Auswahl konkreter Methoden kann das Problem mangelnder Transparenz und Nachvollziehbarkeit aktiv angegangen und somit potenzielle Ungleichbehandlung vermieden werden.

Quellen

[1] Adadi, A. & Berrada, M. (2018), Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI) in IEEE Access vol. 6, pp. 52138–52160.

[2] Angwin, J., Larson, J., Kirchner, L. & Mattu, S. (2016): Machine Bias: There's software used across the country to predict future criminals. And it's biased against blacks., [Online], Verfügbar unter *https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing* (Zugriff am: 12.12.2023).

[3] Barth, V. (2021): Konzeption und Implementierung eines Empfehlungssystems für die Auswahl und Anwendung von XAI-Methoden, [Online], verfügbar unter *https://github.com/vereba/xai_recommender/blob/main/Masterarbeit_Verena_Barth.pdf* (Zugriff am 12.12.2023).

[4] Becker, B. & Kohavi, R. (1996): Adult. UCI Machine Learning Repository. *https://doi.org/10.24432/C5XW20*.

[5] Belle, V. & Papantonis, I. (2021): Principles and Practice of Explainable Machine Learning in Frontiers in big Data vol. 4, 688969.

[6] Bolukbasi, T., Chang, K.-W., Zou, J. Y., Saligrama, V. & Kalai, A. T. (2016): Man is to computer programmer as woman is to homemaker? Neural Information Processing Systems (2016).

[7] Dastin, J. (2018): Amazon scraps secret AI recruiting tool that showed bias against women, [Online], Verfügbar unter *https://www.reuters.com/article/us-amazon-com-jobs-automation-insight/amazon-scraps-secret-ai-recruiting-tool-that-showed-bias-against-women-idUSKCN1MK08G* (Zugriff am: 12.12.2023).

[8] Fanaee-T,H. (2013): Bike Sharing Dataset. UCI Machine Learning Repository. *https://doi.org/10.24432/C5W894*.

[9] Garcia, S., Luengo, J., Sáez, J. A., López, V. & Herrera, F. (2013): A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning in IEEE Transactions on Knowledge and Data Engineering vol. 25(4), pp. 734–750.

[10] Hashemi, M. (2023): Who wants what and how: a Mapping Function for Ex-

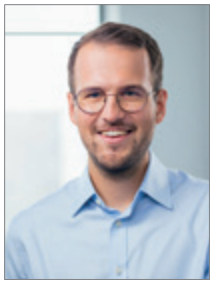
plainable Artificial Intelligence. ArXiv abs/2302.03180
[11] Kangur, A. (2020): Explainable AI in practice: How committing to transparency made us deliver better AI products, [Online], Verfügbar unter *https://towardsdatascience.com/explainableai-in-practice-6d82b77bf1a7* (Zugriff am: 12.12.2023).
[12] Leslie, D. (2019): Understanding artificial intelligence ethics and safety: A guide for the responsible design and implementation of AI systems in the public sector, SSRN 3403301.
[13] Lundberg, S. M. & Lee, S.-I. (2017): A Unified Approach to Interpreting Model Predictions, arXiv preprint arXiv:1705.07874.
[14] Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K. & Galstyan, A. (2019): A survey on bias and fairness in machine learning, arXiv preprint arXiv:1908.09635
[15] Miller, T. (2019): Explanation in Artificial Intelligence: Insights from the Social Sciences in Artificial intelligence vol. 267, pp. 1–38
[16] Molnar, C. (2019): Interpretable Machine Learning. Verfügbar unter *https://christophm.github.io/interpretable-ml-book/*.
[17] Partnership on AI (2021): Artificial Intelligence Incident Database, [Online], Verfügbar unter *https://incidentdatabase.ai* (Zugriff am: 12.12.2023).
[18] Riberio, M. T., Singh, S. & Guestrin, C. (2018): Anchors: High precision model-agnostic explanations, Thirty-Second AAAI Conference on Artificial Intelligence.

Über die Autoren

Verena Barth war als IT-Beraterin bei der viadee mit dem Fokus auf Machine Learning und MLOps tätig. Sie setzt sich leidenschaftlich für die Verständlichkeit von komplexen ML-Systemen ein, um eine nachvollziehbare und gewissenhafte Anwendung zu ermöglichen. 2024 gründet sie ein Unternehmen, das mithilfe von LLMs professionelles und personalisiertes Business Coaching skalierbar anbietet.

Tobias Goerke ist Berater bei der viadee IT-Unternehmensberatung. Als Data Scientist liegen seine Schwerpunkte in der Einführung künstlich intelligenter Systeme und der Erforschung verschiedener Verfahren des erklärbaren Maschinenlernens.

Verena Barth
verena.barth95@web.de



Tobias Goerke
Tobias.Goerke@viadee.de


ORDEM E PROGRESSO

Mehrsprachige Anwendungen mit APEX – ganz einfach?

Dr. Gudrun Pabst, Muniqsoft Consulting

APEX bietet die Möglichkeit, Applikationen zu übersetzen und den Anwendern mehrsprachig zur Verfügung zu stellen. Dabei wird einfach eine weitere Sprache für die Applikation im Übersetzungsbereich eingetragen, die Texte der Applikation werden in das Übersetzungsrepository geladen und wahlweise dort oder über den Export und Import einer XLIF-Datei übersetzt. Danach wird die übersetzte Applikation veröffentlicht und schon ist die Applikation zweisprachig. Aber ist dies wirklich so einfach? Wie werden Wertelisten für Stammdaten übersetzt, die auf Datenbanktabellen basieren? Wie können für diese Stammdaten Texte in mehreren Sprachen durch Endanwender erfasst werden? Wie sieht ein geeignetes Vorgehen bei Meldungstexten in Dynamic Actions aus? Dieser Artikel erläutert den Ansatz, den wir in unseren APEX-Projekten für die Umsetzung gewählt haben, um den Anwendern sowohl die Oberfläche als auch die Stammdaten mehrsprachig zur Verfügung zu stellen, ohne auf ein relationales Datenmodell zu verzichten.

Vorarbeiten

Bevor wir übersetzen können, müssen wir erst die Anwendung erstellen, die übersetzt werden soll. Beim Anlegen dieser Anwendung wird die Sprache der Applikation abgefragt. Dies ist die Sprache, in der die Anwendung entwickelt wird und von der aus in alle anderen benötigten Sprachen übersetzt wird *(siehe Abbildung 1)*.

Dabei ist zu beachten, dass APEX statische Texte für erstellte Objekte (zum Beispiel Button-Labels) **nicht** in der Sprache der Anwendung generiert, **sondern** in der Sprache der Entwicklungsoberfläche zur Zeit der Erstellung!

In der erstellten Applikation können wir anschließend unter „Shared Components“ → „Globalization Attributes“ auswählen, wie APEX bestimmt, in welcher Sprache die Anwendung ausgeführt wird *(siehe Abbildung 2)*.

Für übersetzte Applikationen kann festgelegt werden, dass die aktuelle Sprache

- aus den Browser-Einstellungen der Anwender,
- aus Präferenzen, die in Applikationseinstellungen oder einem Item gespeichert sind, oder
- aus der Session-Einstellung (gesetzt über `APEX_UTIL.SET_SESSION_LANG` oder `P_LANG` in der URL)

ermittelt wird.

Ablauf des Übersetzungsvorgangs

Öffnet man im Application Builder unter „Shared Components“ im Abschnitt „Globalization“ den Link „Application Translations“, zeigt die Zielseite den Ablauf des Übersetzungsvorgangs *(siehe Abbildung 3)*.

Die wichtigsten Punkte des Übersetzungsvorgangs sind hier kurz zusammengefasst, werden aber in den folgenden Abschnitten noch genauer erklärt:

- Zunächst muss jede gewünschte Zielsprache eingetragen und dadurch eine übersetzte (versteckte) Applikation definiert werden.

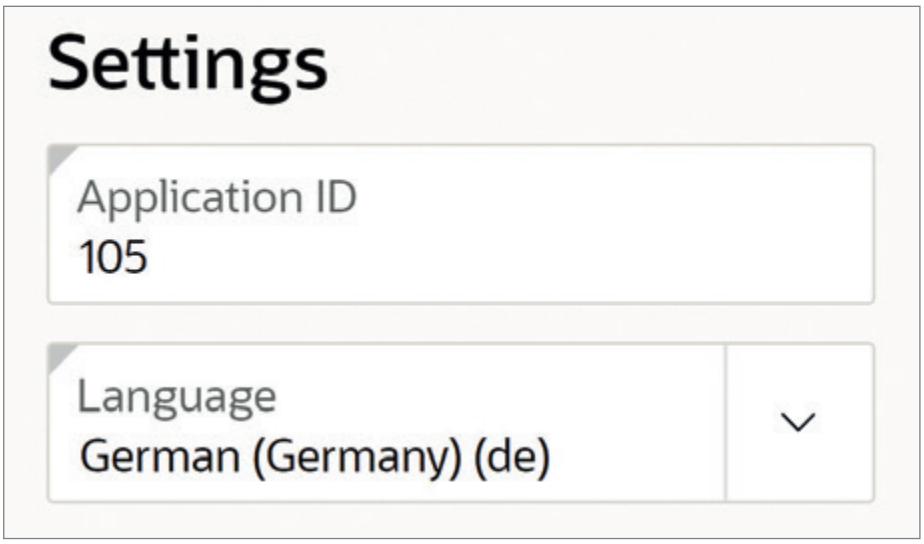


Abbildung 1: Quellsprache (Quelle: Dr. Gudrun Pabst)

- Anschließend werden im „Seed"-Schritt alle übersetzbaren Texte aus der Originalanwendung extrahiert und den übersetzten Applikationen im „Translation Repository" zugeordnet.
- Der dritte und der fünfte Schritt sind nur bei Verwendung eines Übersetzungswerkzeugs nötig, das das XLIFF-Format benutzt. Hier werden alle Texte für jede Zielsprache in eine XLIFF-Datei heruntergeladen und nach dem Übersetzen wieder in das Translation Repository hochgeladen.
- Im vierten Schritt findet die Übersetzung statt – entweder in den heruntergeladenen XLIFF-Dateien oder direkt im Translation Repository.
- Durch „Publish" werden beim ersten Mal die übersetzten Anwendungen als versteckte APEX-Applikationen erzeugt und bei jeder Durchführung mit den aktuell im Translation Repository hinterlegten übersetzten Texten versehen.

Schritt 1: Definition einer Zielsprache

Über den Link im ersten Schritt des Ablaufs öffnet sich die Seite der festgelegten Zielsprachen. Hier wird über den Create-Button eine neue Zielsprache definiert *(siehe Abbildung 4)*.

Jede Zielsprache erfordert eine neue Applikation mit eigener ID. Diese ID darf nicht auf 0 enden!

Bei der Entwicklung einer Anwendung liefert APEX automatisch Texte für die Bedienelemente von Interactive Reports und Interactive Grids sowie für interne Meldungen. Diese Texte stehen in mehreren Sprachen zur Verfügung. Damit sie in unseren Anwendungen genutzt werden können, müssen die entsprechenden Sprachdateien installiert sein.

Auch in den übersetzten Anwendungen werden diese Texte in der gewählten Zielsprache verwendet, wenn diese eine der unterstützten Sprachen ist. Für alle anderen Zielsprachen müssen zusätzlich zur Übersetzung der Anwendung ebenso diese APEX-Texte übersetzt werden. Hierfür steht im unteren Bereich der „Translate"-Seite der Punkt „Text Messages" zur Verfügung *(siehe Abbildung 5)*.

Alle gewünschten Messages müssen in allen benötigten Sprachen selbst eingetragen werden. Die eingetragenen Meldungen sind Objekte der Anwendung, nicht des Workspace, das heißt, für andere Anwendungen im Workspace müssen die Meldungen wieder eingegeben werden.

Die Listen der Schlüsselwerte für die Meldungstexte sind in der Dokumentation unter „Translating Messages" zu finden.

Dieser Bereich kann auch für eigene Meldungen verwendet werden. Dabei sollten die eigenen Messages mit einem eigenen Kürzel starten, das sich von den von APEX verwendeten unterscheidet. In PL/SQL kann über `apex_lang.message` auf die Einträge zugegriffen werden, in JavaScript über `apex.lang` .

Schritt 2: Seed

Der Aufruf dieses Punktes zeigt die Liste aller zuvor definierten Zielsprachen und -applikationen. Beim Start des „Seed"-Prozesses werden aus der Quellapplikation alle übersetzbaren Texte ausgelesen und für alle angehakten Anwendungen Einträge im Translation Repository gemacht. Als übersetzter Text wird dabei der ausgelesene Text eingetragen.

Schritt 3: Download des XLIFF (optional)

Bei Verwendung eines Übersetzungswerkzeugs, das das XLIFF-Format unterstützt, werden nach der Befüllung des Translation Repository für jede Sprache die XLIFF-Dateien heruntergeladen *(siehe Abbildung 6)*.

Bei der Auswahl *„Only those elements requiring translation"* ist zu beachten, dass diese Elemente alle Objekte umfassen, für die der Quelltext und der übersetzte Text identisch ist. Wird in der deutschen und in der englischen Anwendung das Label „Name" für ein Item verwendet, taucht diese Beschriftung immer im XLIFF-File auf.

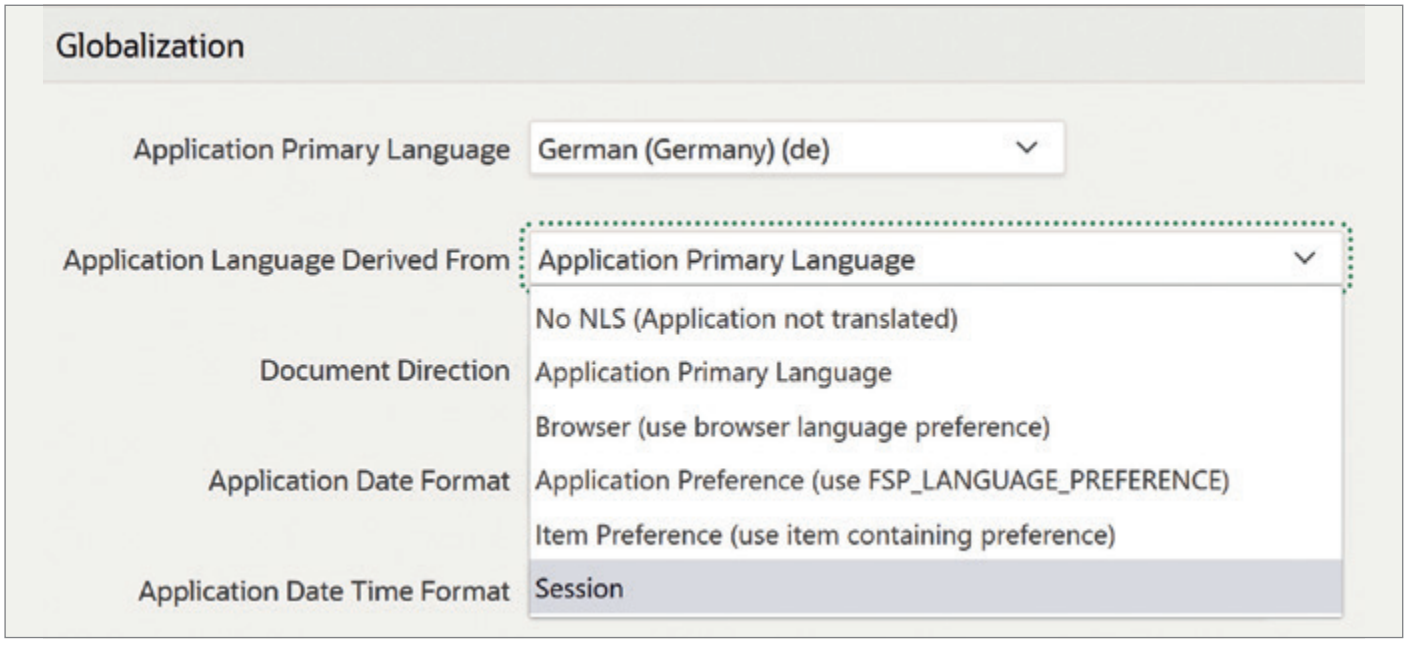


Abbildung 2: Ermittlung der Ausführungssprache (Quelle: Dr. Gudrun Pabst)

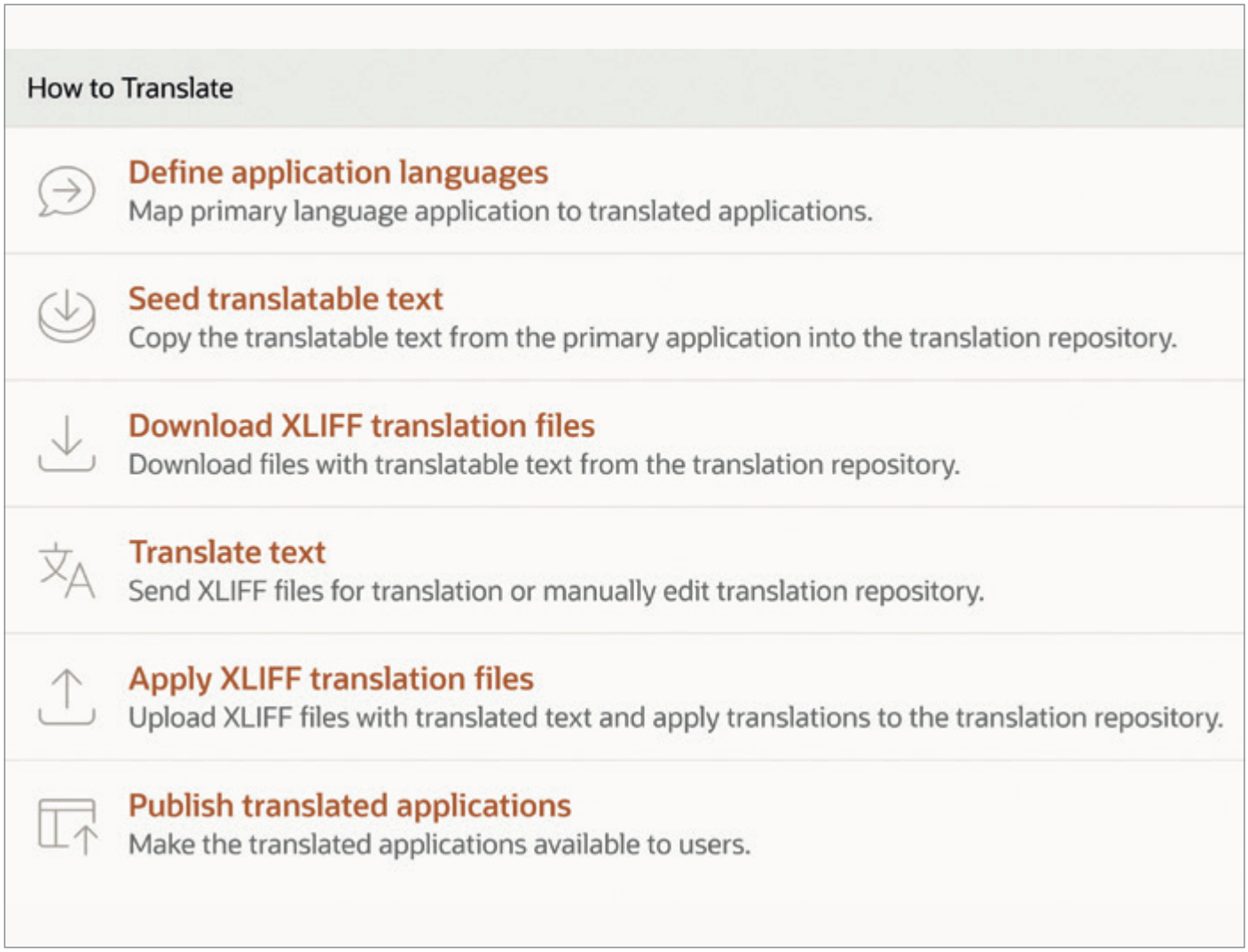


Abbildung 3: Ablauf der Übersetzung (Quelle: Dr. Gudrun Pabst)

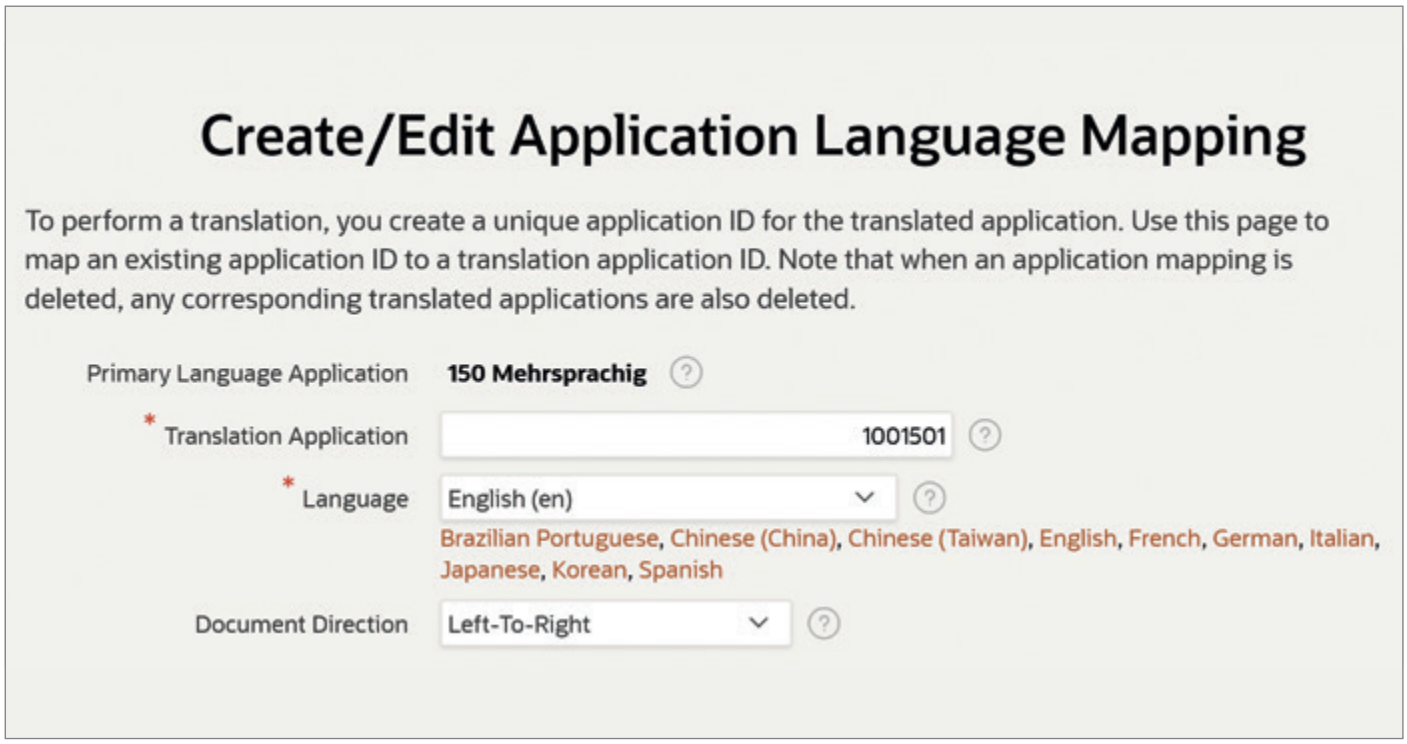


Abbildung 4: Definition einer Zielsprache (Quelle: Dr. Gudrun Pabst)

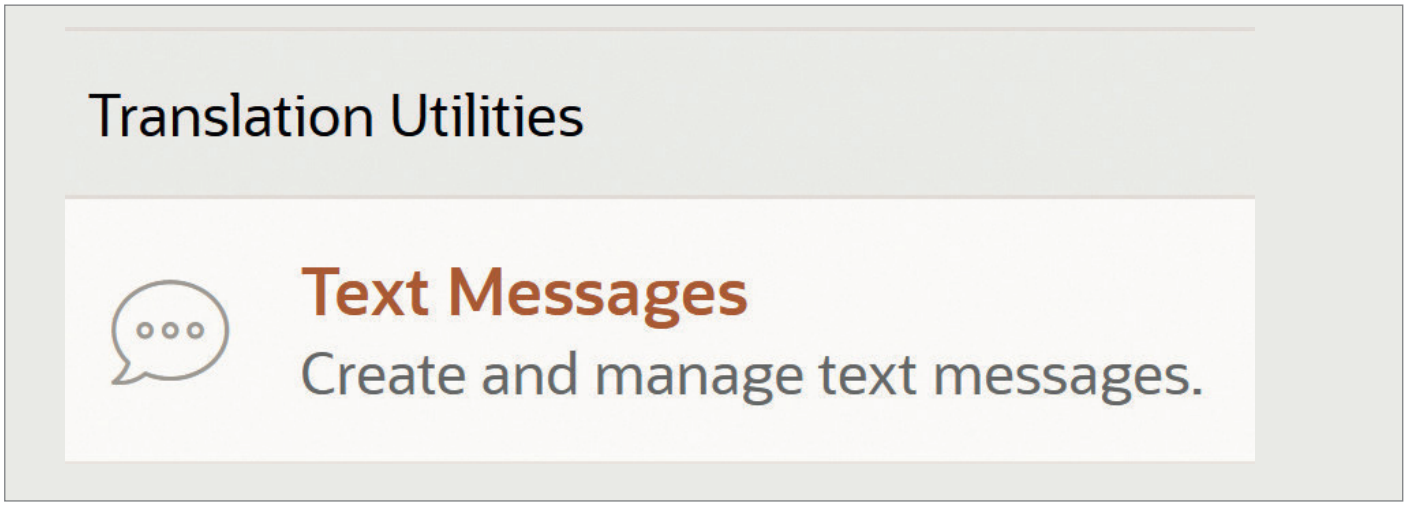


Abbildung 5: Text Messages (Quelle: Dr. Gudrun Pabst)

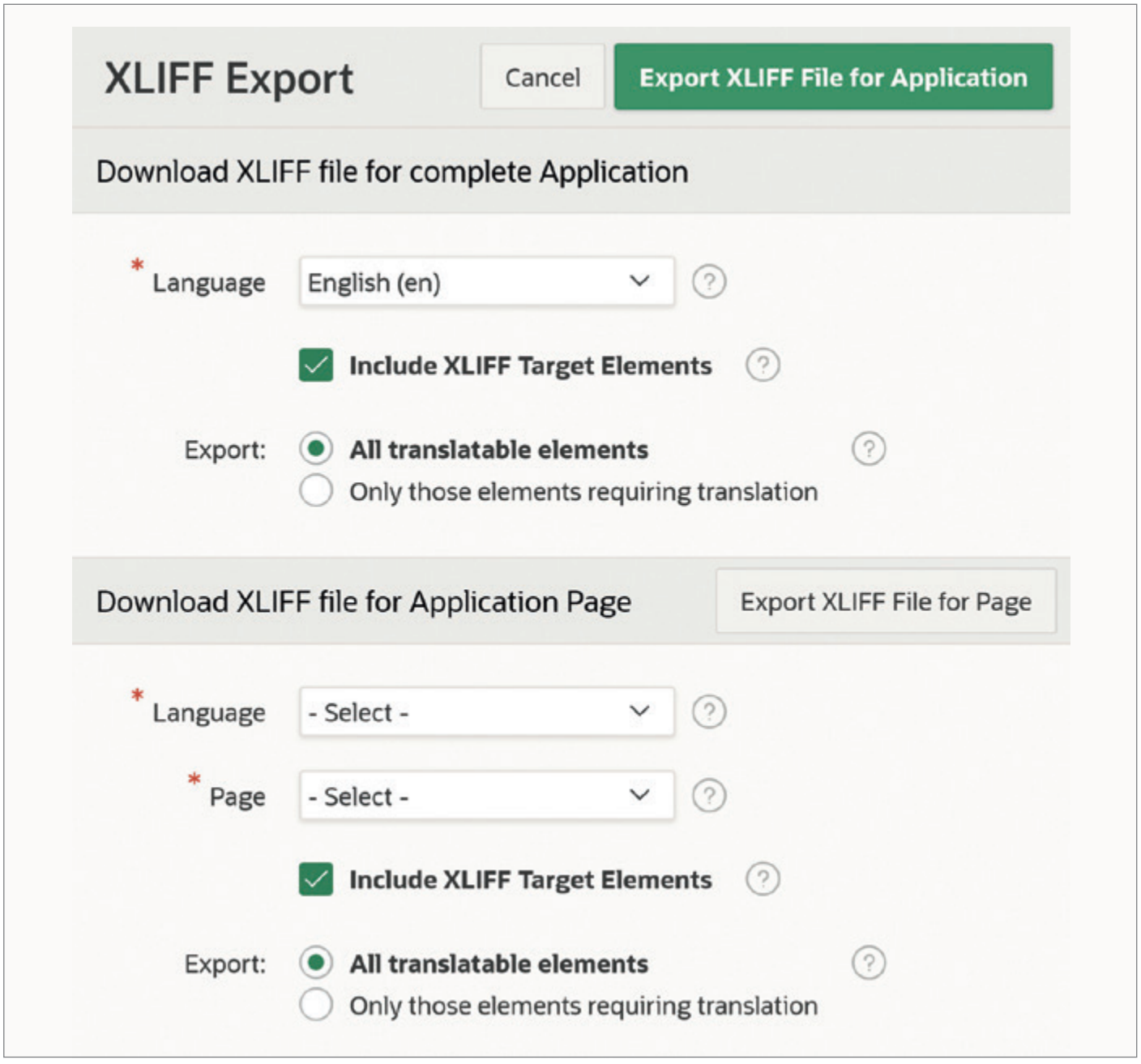


Abbildung 6: XLIFF-Export (Quelle: Dr. Gudrun Pabst)

Edit	Translated Application	Page	Language	Translate From	Translate To	Column Description	Translated
	1001501	1	en	Startseite	Startseite	Page Name.	No
	1001501	1	en	Mehrsprachig	Mehrsprachig	Page Title.	No
	1001501	10	en	Kategorien_IG	Kategorien_IG	Region name	No
	1001501	10	en	Kategorien	Kategorien	Page Name.	No
	1001501	10	en	Kategorien	Kategorien	Page Title.	No
	1001501	10	en	Kategorie-Namen gespeichert.	Kategorie-Namen gespeichert.	Page Process Success Message.	No
	1001501	10	en	Kategorien	Kategorien	Region name	No
	1001501	10	en	Fehler beim Speichern der Namen: #SQLERRM#	Fehler beim Speichern der Namen: #SQLERRM#	Page Process ErrMsg.	No

Abbildung 7: Translation Repository (Quelle: Dr. Gudrun Pabst)

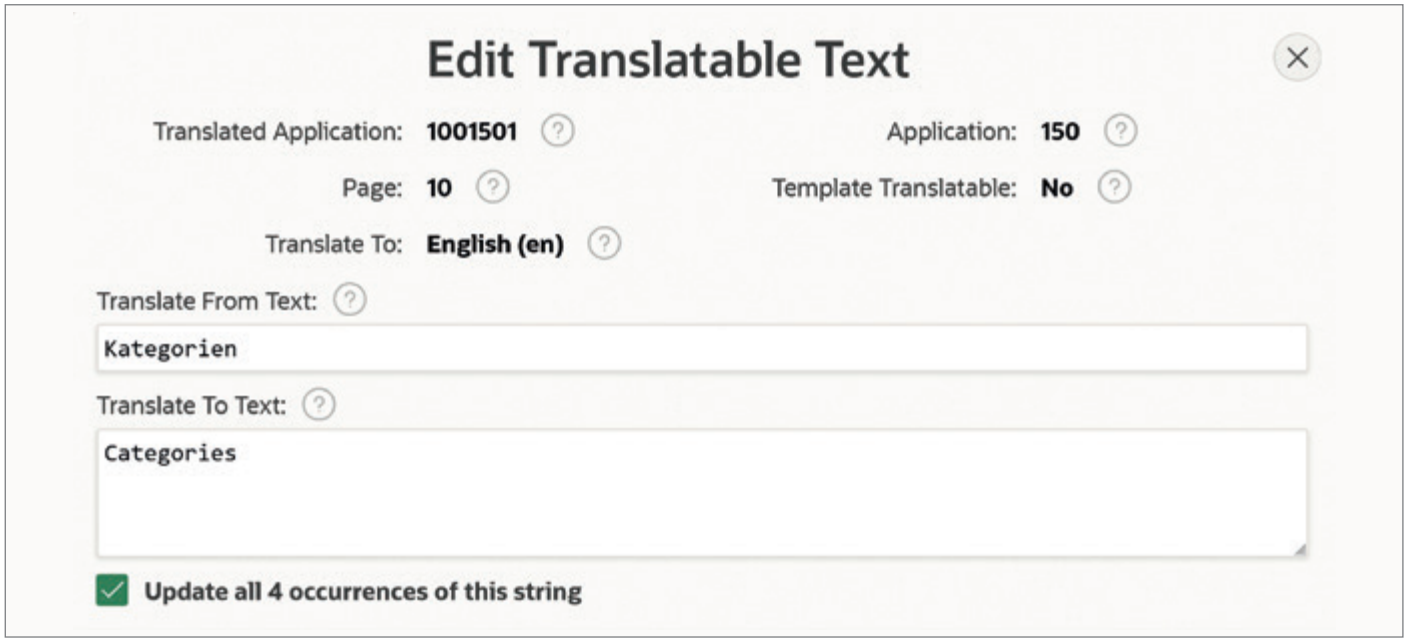


Abbildung 8: Manuelle Übersetzung (Quelle: Dr. Gudrun Pabst)

	Code	Name	Category	Condition	Price	Quantity
	BILD_002	Bild (abstrakt)	Dekoration	neu	10.00	15
	STUHL_012	Bürostuhl (hohe Lehne)	Möbel	gebraucht	50.00	10

1 - 2

Abbildung 9: Nicht übersetzte Schlüsselwerte (Quelle: Dr. Gudrun Pabst)

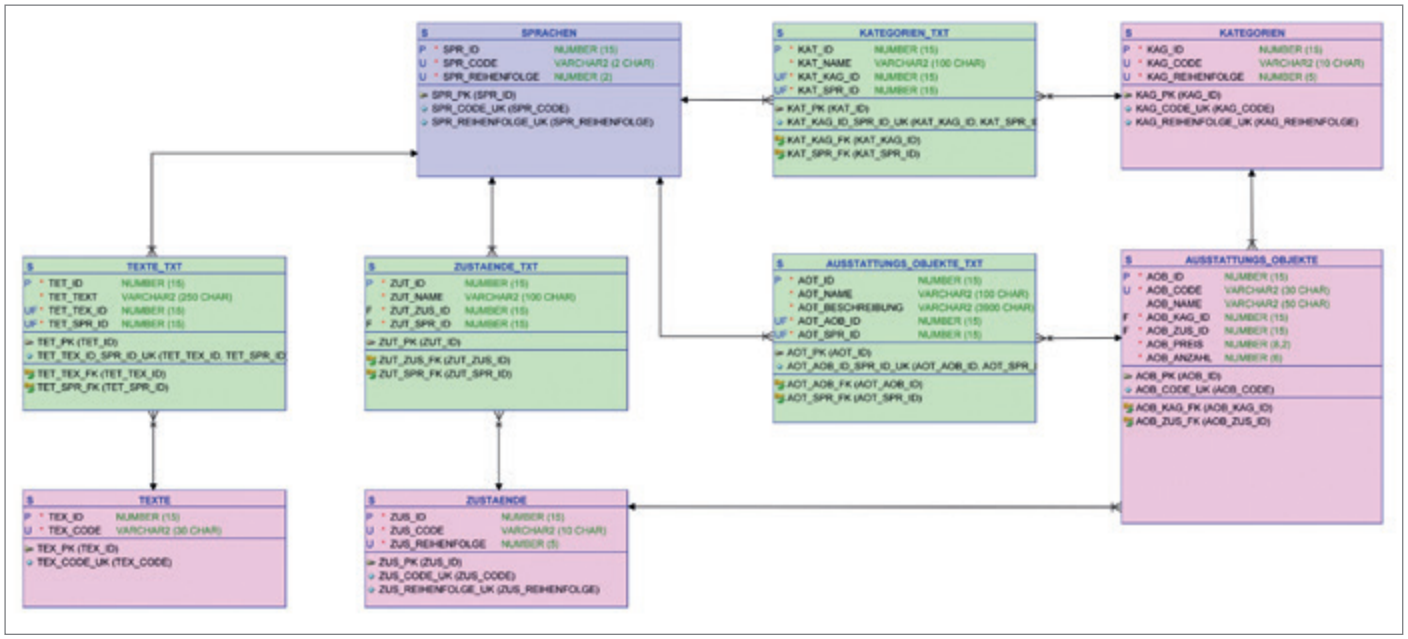


Abbildung 10: Datenmodell (Quelle: Dr. Gudrun Pabst)

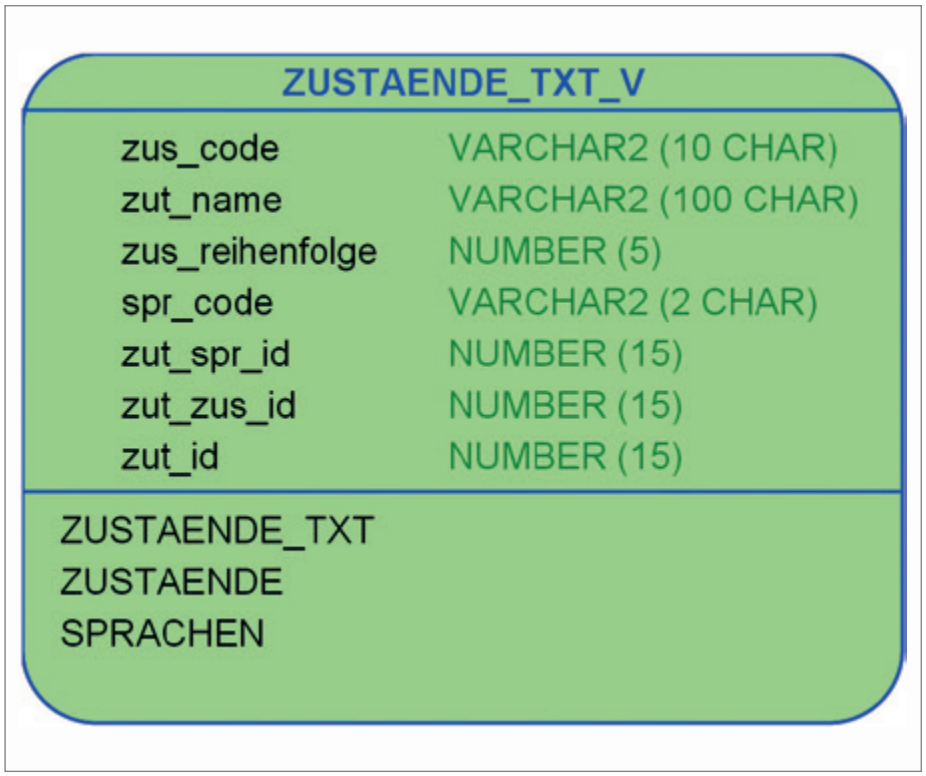


Abbildung 11: Sprachview (Quelle: Dr. Gudrun Pabst)

Schritt 4: Übersetzen

Im nächsten Schritt werden die extrahierten Texte der Anwendung in die Zielsprachen übersetzt. Dies kann entweder außerhalb in einem Übersetzungstool erfolgen oder durch Bearbeitung der Einträge im Übersetzungsrepository.

Für die manuelle Bearbeitung wird im unteren Bereich der „Translate"-Seite der Link auf das Translation Repository aufgerufen. Dieses listet alle Texte für alle definierten Zielsprachen zusammen mit einem Bearbeitungsbutton in einem Interactive Report auf *(siehe Abbildung 7)*.

Über den Edit-Link öffnet sich die Bearbeitungsseite *(siehe Abbildung 8)*.

Hier kann für den aktuellen Eintrag die Übersetzung angegeben werden. Kommt der Text mehrfach vor, können gleichzeitig alle mit derselben Übersetzung versehen werden.

Schritt 5: Upload des übersetzten XLIFF (optional)

Im Upload-Dialog für die XLIFF-Dateien können bis zu 10 Dateien (das heißt Übersetzungen für 10 Sprachen) hochgeladen werden. Nach dem Hochladen wird für jede Datei angegeben, zu welcher Zielanwendung und -sprache sie mittels Apply angewendet wird.

Schritt 6: Publish

Der Aufruf dieses Punktes zeigt die Liste aller zuvor definierten Zielsprachen und -applikationen. Beim Start des „Publish"-Prozesses werden aus dem aktuellen Stand der Quellapplikation und allen Texten im Translation Repository alle angehakten Anwendungen erstellt.

Wurden seit dem Seed-Prozess an der Quellapplikation Änderungen gemacht, die Texte beinhalten, werden diese Texte unübersetzt in die Zielanwendungen übertragen.

Deployment

Um die Anwendung mit allen Übersetzungen aus der Entwicklungsumgebung

in andere Umgebungen zu übertragen, wird sie mit der aktivierten Einstellung „Export Translations“ exportiert. Nach dem Import in die Zielumgebung sind alle übersetzten Texte im Translation Repository, jedoch noch nicht veröffentlicht. Alle Zielanwendungen müssen mittels „Publish“ in der Zielumgebung zur Verfügung gestellt werden.

Beim Export und Import der Anwendung werden auch alle angelegten Meldungstexte unabhängig von der Sprache exportiert und importiert, selbst wenn die Applikation keine Definition für die Sprache des Meldungstexts enthält.

Ergebnis der Übersetzung

Wenn für alle Texte, die im Translation Repository vorhanden sind, Übersetzungen eingetragen wurden, sieht die Anwendung in der Zielsprache so aus:

- Eingebaute APEX-Objekte, wie Löschabfragen oder Beschriftungen der Menüs von Interactive Reports und Interactive Grids, sind übersetzt.
- Labels, Region Titles und viele andere Attribute, sogar die Region Source, sind übersetzt.
- Der Code einer JavaScript-Action in einer Dynamic Action ist übersetzt, aber nicht JavaScript-Code im Page-Attribut „Function and Global Variable Declaration“.
- Für einige Attribute wie die Source für Display-Only-Items gibt es keine Möglichkeit zum Übersetzen im Translation Repository.
- Texte in LOVs aus der Datenbank, insbesondere Texte der Anwender, sind nicht übersetzt *(siehe Abbildung 9)*.

Datenmodell für die Übersetzung der weiteren Objekte

Sowohl die Wertelisten als auch die Display-Only-Items mit statischem Inhalt lassen sich einfach übersetzen, wenn man ein geeignetes Datenmodell verwendet und die benötigten Texte aus der Datenbank ermittelt.

Für Wartbarkeit und Flexibilität werden die Sprachen in einer Tabelle hinterlegt. Jede Sprache wird über ihren Sprachcode identifiziert, dieser entspricht dem von APEX verwendeten Sprachcode (gegebenenfalls mit Lokalisierung).

Die Schlüsseltabellen enthalten nur noch die Schlüssel und gegebenenfalls sprachunabhängige Sortierungen, für die Texte zu den Schlüsseln verwenden wir Zuordnungstabellen zwischen der Sprachentabellen und der jeweiligen Schlüsseltabelle *(siehe Abbildung 10)*.

Um die Verwendung in Abfragen zu erleichtern, wird zu jeder Schlüsseltabelle mit Sprachtabelle eine View erstellt, die den Sprachcode, die Spalten der Schlüsseltabelle und die Spalten für die Texte aus der Sprachtabelle enthält *(siehe Abbildung 11)*.

Verwaltung in APEX

In APEX legen wir einige Application Items für die IDs der Sprachen an:

- Items `A_SPR_ID_<XX>` für die ID für jeden Sprachcode xx
- Items `A_SPR_ID` und `A_SPR_CODE` für die aktuelle Sprache

Diese Items werden initial beim Start der Anwendung gefüllt.

Ist ein Umschalten der Sprache möglich (abhängig von der Einstellung „Application Language Derived From“), müssen beim Setzen der neuen Sprache zusätzlich die Application Items für die aktuelle Sprache umgestellt werden.

Die Abfragen von LOVs und Regions werden auf die Views umgestellt und mit einer Where-Bedingung versehen, die die Daten auf die Sprache aus dem Application Item `A_SPR_ID` einschränkt.

Für die Eingabe der Sprachtexte in APEX können wir bei den Schlüsseltabellen ein Interactive Grid zur Verfügung stellen. Dieses hat als Quelle eine Abfrage über die Basistabelle der Schlüsselwerte sowie über die Texte zu den Schlüsselwerten in allen Sprachen *(siehe Abbildung 12)*.

Die Spalten der Sprachtexte werden auf „Query Only“ gestellt.

Für das Speichern der Daten der Basistabelle wird der Standardprozess des Interactive Grid auf der Basistabelle verwendet, für das Speichern der Sprachtexte wird ein Prozess angelegt, der diese Texte mittels Merge mit gegebenenfalls bereits vorhandenen Sprachtexten zum Basisdatensatz zusammenführt.

Bei Tabellen mit mehr Spalten ist es sinnvoll, den Anwendern ein Einzelsatzformular zur Verfügung zu stellen. Das Vorgehen ist dasselbe wie beim Interactive Grid:

Die Quelle der Daten ist eine Abfrage, die zu den Basisdaten die Sprachtexte in

SQL Query

```
select kag.kag_id           ,
       kag.kag_code         ,
       kag.kag_reihenfolge  ,
       kad.kat_name         kat_name_de ,
       kae.kat_name         kat_name_en
  from kategorien           kag ,
       kategorien_txt_v     kad ,
       kategorien_txt_v     kae
 where kad.kat_kag_id = kag.kag_id
   and kad.spr_code   = 'de'
   and kae.kat_kag_id = kag.kag_id
   and kae.spr_code   = 'en'
```

Abbildung 12: SQL-Query des Interactive Grid (Quelle: Dr. Gudrun Pabst)

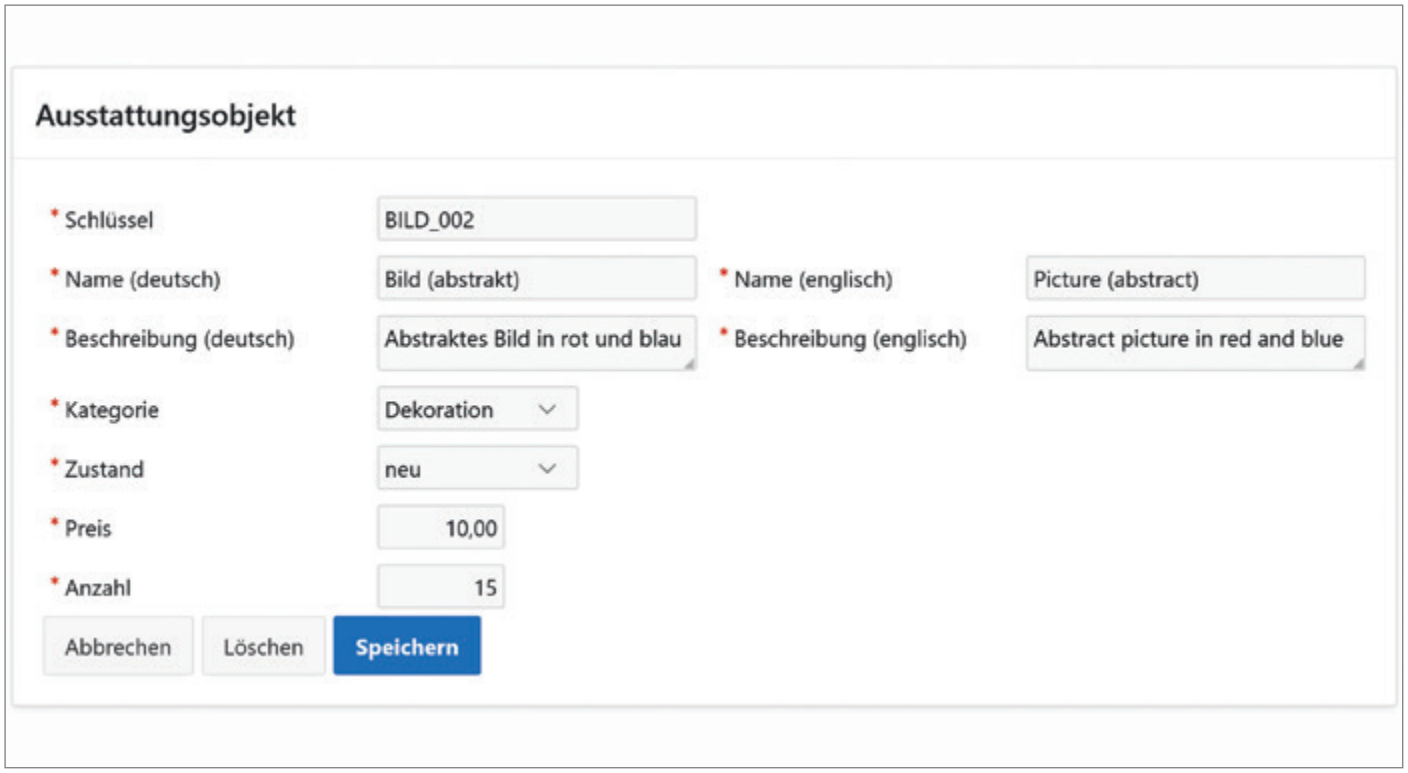


Abbildung 13: Formular mit Eingabe der Texte in zwei Sprachen (Quelle: Dr. Gudrun Pabst)

allen verwalteten Sprachen ermittelt *(siehe Abbildung 13)*.

Die Items der Sprachtexte werden auf „Query Only" gestellt. Das Ziel für das Speichern der Basisdaten ist die Tabelle, für die Sprachtexte wird ein Prozess mit Merge-Logik angelegt.

Bei beiden Varianten wird über eine NOT-NULL-Validierung sichergestellt, dass für alle Sprachen die benötigten Texte hinterlegt sind.

Zusammenfassung

Die Mehrsprachigkeit in APEX funktioniert auf Basis von versteckten Schattenapplikationen je gewünschter weiterer Sprache. Diese Applikationen werden definiert, dann wird das Translation Repository mit den zu übersetzenden Texten gefüllt, die Übersetzung durchgeführt und abschließend die Schattenapplikationen mittels „Publish" erstellt. Die Schattenapplikationen werden nicht automatisch neu erstellt, so dass nach jedem Entwicklungsschritt, auch wenn keine neuen Texte dazugekommen sind, mindestens der „Publish"-Vorgang nochmals durchgeführt werden muss.

Nicht alle Felder, in die Texte eingegeben werden können, werden in das Translation Repository übertragen. Texte für Wertelisten, insbesondere für Listen, die von den Anwendern gepflegt werden, sind von der APEX-Übersetzung nicht betroffen. Für beide Fälle ist es sinnvoll, im Datenmodell die Daten eines Objekts in sprachunabhängige und sprachabhängige Anteile aufzuteilen und entsprechende Tabellen anzulegen. In APEX sind dann geeignete Eingabeseiten einzubauen.

Eine große Herausforderung ist es, bei der Entwicklung immer die Mehrsprachigkeit zu beachten.

Dr. Gudrun Pabst
gudrun.pabst@muniqsoft-consulting.de

PL/SQL-Performancesteigerung durch optimale Datentypen und native Kompilierung

Jan Gorkow, SD&C Solutions Development & Consulting

Die Oracle-Datenbank bringt mit PL/SQL eine sehr mächtige und einfach zu erlernende Programmiersprache mit, deren Potential im Hinblick auf die Implementierung von Business-Logik oftmals unterschätzt wird. Insbesondere in Kombination mit APEX als Low Code Rapid Development Framework für Webanwendungen ergibt die datennahe Umsetzung vom Applikationslogik in PL/SQL sehr viel Sinn. Hierbei haben insbesondere die Wahl geeigneter numerischer Datentypen sowie des Optimierungslevels und der Kompilierungsmethode gerade bei rechenintensiven Anwendungen großen Einfluss auf die Performance, beziehungsweise die Laufzeiten, und damit am Ende auch auf die User Experience.

Motivation

Wohin mit der Business-Logik? In die Datenbank oder genau das nicht? Diese grundsätzliche Frage der Softwarearchitektur, an der sich viele Geister scheiden, wird immer wieder gern und teilweise hitzig diskutiert.

Betrachtet man gängige Drei-Schicht-Architekturen von Software-Systemen, so ergibt sich nach langjähriger Erfahrung des Autors nicht selten folgendes

Bild, stark vereinfacht dargestellt in *Abbildung 1*.

- Die Datenbank dient als reiner Persistenz-Layer und enthält maximal die technisch erforderliche Logik zur Sicherstellung der Datenkonsistenz, wird also relativ „dumm" gehalten.
- Die Business-Logik wird außerhalb der Datenbank realisiert und in Applikationsservern gehostet. Die erforderliche Skalierung wird dadurch erreicht, dass beliebig viele Applikationsserver-Instanzen über eine Virtualisierungsplattform, wie zum Beispiel Kubernetes, parallel ausgeführt werden.
- Die Schnittstelle zu den Anwendern bildet ein Browser-basiertes Web Frontend.

Dem Autor blutet bei solchen Architekturen oftmals das Herz, und das nicht nur, weil viele großartige und durch die Kunden mitbezahlte Features der Oracle-Datenbank hierdurch gar nicht zum Einsatz kommen können. Es ist darüber hinaus insbesondere die hohe Komplexität des Gesamtsystems, die zu einem nicht unerheblichen Overhead vor allem für die Kommunikation zwischen Applikationslogik und DB führt. Und Overhead geht grundsätzlich zu Lasten der Performance! Ein genauerer Blick unter die Haube macht deutlich, dass oftmals für simpelste Berechnungen eine Vielzahl von Datenbankzugriffen notwendig sind und für jeden einzelnen Zugriff müssen alle Protokollschichten auf Seiten des Clients (Applikation) sowie auch des Servers (DB) durchlaufen werden. Selbst wenn einzelne Zugriffe im Millisekunden-Bereich beantwortet werden, summieren sich so die vielfach Millionen Kleinstzugriffe zu erheblichen Wartezeiten. Nicht selten muss nur aufgrund dieser Architektur auf Applikationsebene stark skaliert werden, um dem erwarteten Workload gerecht zu werden. Ein weiterer wesentlicher Punkt ist, dass oftmals auf Applikations- und Frontend-Ebene recht schnelllebige Frameworks zum Einsatz kommen, die bei Aktualisierungen keine volle Abwärtskompatibilität aufweisen. Dies führt dazu, dass bei entsprechenden Migrationen nicht selten Anwendungsteile neu geschrieben werden müssen, nur um den bestehenden Funktionsumfang beizubehalten.

Dem gegenüber steht die Alternative des SmartDB-Ansatzes. Dieser folgt dem Grundgedanken, dass die Oracle-Datenbank selbst bereits all das mitbringt, was für die Abbildung von Business-Applikationen erforderlich ist: Sie ist ohne Zweifel hochgradig skalierbar, die Persistierung ist obligatorisch, die Business-Logik ist in PL/SQL unter Nutzung einer Vielzahl integrierter Features der Datenbank abbildbar und nicht zuletzt, lässt sich auch eine moderne Präsentationsschicht mittels APEX auf Basis der Oracle-Datenbank umsetzen *(siehe Abbildung 2)*. Insgesamt führt dies zu einer erheblichen Verringerung der Komplexität des Gesamtsystems. Allein durch den Wegfall der angeführten Netz-Kommunikation erübrigt sich schon ein Teil der ansonsten erforderlichen Hardware-Ressourcen. Und nicht zuletzt sorgt Oracle als Hersteller für eine starke Abwärtskompatibilität, wodurch selbst Jahrzehnte alte PL/SQL-Programme in aktuellen Versionen der Oracle DB noch unverändert ihren Dienst tun, wie am ersten Tag. Das ist Investitionssicherheit und all dies zusammen führt nun zur zentralen Fragestellung dieses Artikels:

Wie kann insbesondere rechenintensive Business-Logik performanceoptimiert in PL/SQL abgebildet werden?

Die PL/SQL-Kompilierungsmethode: Interpreted vs. Native

PL/SQL-Code kann auf zwei verschiedene Arten kompiliert werden, was über den Session-Parameter PLSQL_CODE_TYPE gesteuert wird.

- Bei der interpretierten Kompilierung (PLSQL_CODE_TYPE = INTERPRETED), welche den Standard darstellt, wird der geschriebene Source Code in einen systemunabhängigen Bytecode übersetzet, welcher erst zur Laufzeit durch die PL/SQL Interpreter Engine in für die CPU verständliche Anweisungen umgesetzt wird. Dies führt zu einer langsameren Ausführung im vergleich zu nativ kompiliertem PL/SQL-Code. Dafür ist die Kompilierung schneller und im Hinblick auf Debugging ist dieser Modus obligatorisch.
- Bei der nativen Kompilierung (PLSQL_CODE_TYPE = NATIVE) wird der Source Code direkt in C-Code übersetzt. Dieser Vorgang ist zwar aufwendiger, führt jedoch zur Laufzeit zum Wegfall der Interpreter Engine und somit zu einer schnelleren Ausführung. Dieser Vorteil wird mit dem Wegfall der Debugging-Möglichkeiten sowie einem höheren Speicherverbrauch zur Laufzeit erkauft.

Grundsätzlich lohnt es sich, die native Kompilierung schon während der Entwicklungszeit und spätestens bei Performancetests einmal auszuprobieren. Hierdurch können erhebliche Performancesteigerungen erreicht werden, ohne auch nur eine Zeile Code anpassen zu müssen.

Frei nach dem Motto, was für den eigenen Code gut ist, kann für Built-in Code nicht schlecht sein, bietet Oracle die Möglichkeit an, auch mitgelieferten PL/SQL-Code nativ zu kompilieren. Dies kann über das mitgelieferte Oracle-Skript *dbmsupgnv.sql* erreicht werden. Eine Rückänderung auf interpretierte Kompilierung ist über das Skript *dbmsupgin.sq*l möglich. Nach Ansicht des Autors sollte diese Möglichkeit jedoch nur dann erwogen werden, wenn zuvor alle anderen Optimierungsmaßnahmen am eigenen Code ausgeschöpft wurden, um die allerletzten paar Prozent an Laufzeitverkürzung noch aus dem System herauszukitzeln. Wenn man sich für diesen Weg entscheidet, sollte für die Built-in- Packages jedoch auf jeden Fall das Optimierungslevel 2 beibehalten werden. Hierzu mehr im folgenden Abschnitt.

Der PL/SQL-Optimizer

Neben der Kompilierungsmethode kann der PL/SQL-Optimizer einen nicht unerheblichen Einfluss auf die Laufzeit haben. Dieser wurde mit Oracle 10g eingeführt und wird über den Session- beziehungsweise System-Parameter PLSQL_OPTIMIZE_LEVEL gesteuert. Folgende Werte sind hierbei möglich:

- 0: Dieser Wert deaktiviert de facto den PL/SQL-Optimizer, was ein Kompilierungsverhalten wie in der Oracle-Version 9i bewirkt. So gibt es in diesem

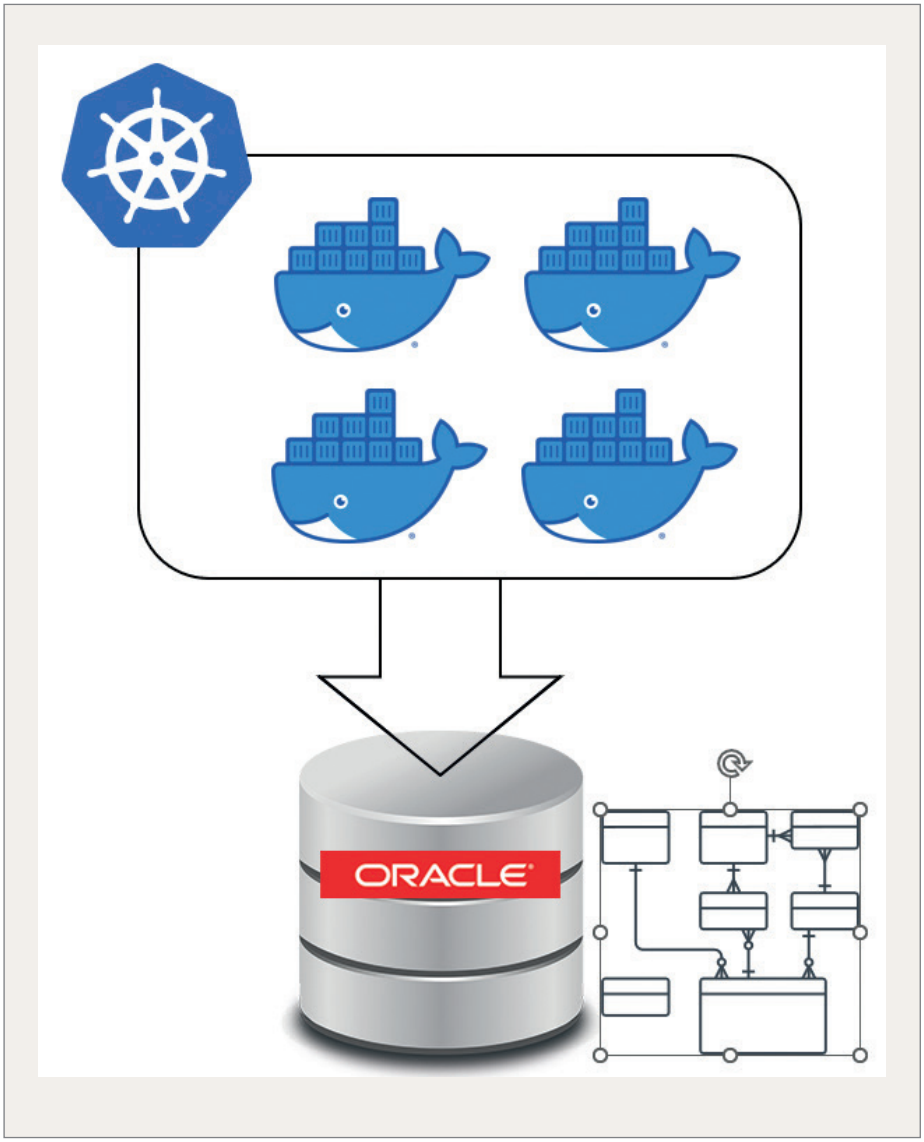


Abbildung 1: Angetroffene Systemarchitektur (Quelle: Jan Gorkow)

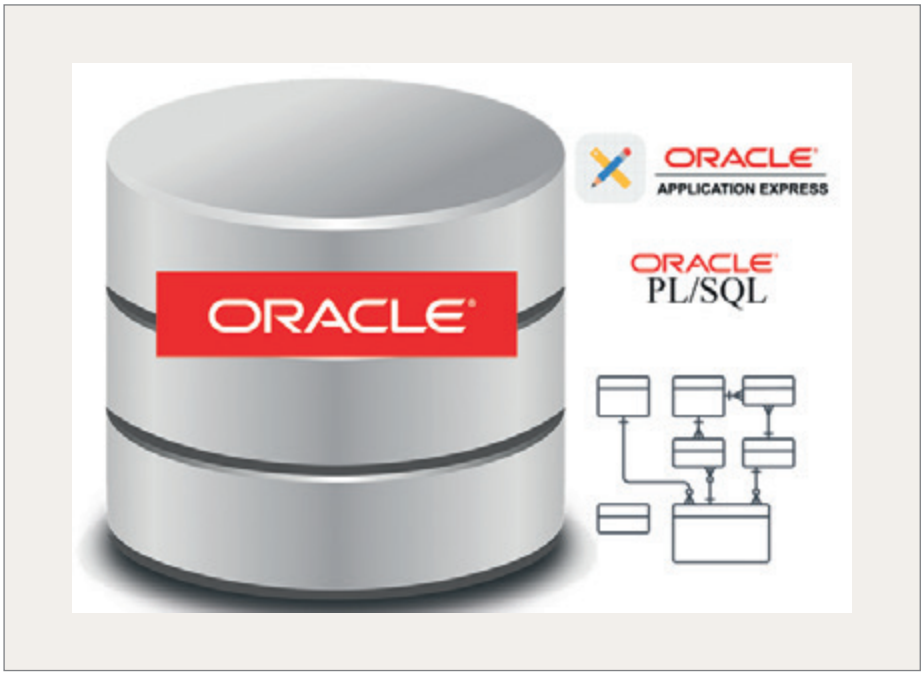


Abbildung 2: SmartDB-Ansatz (Quelle: Jan Gorkow)

Fall beispielsweise keine semantische Identität von BINARY_INTEGER und PLS_INTEGER.

- 1: Im Optimierungslevel 1 werden unnötige Berechnungen und Exceptions eliminiert, jedoch die Anweisungsreihenfolge grundsätzlich beibehalten.
- 2: Dies ist der Default seit der Einführung des PL/SQL-Optimizers mit Oracle 10g. Neben diversen Code-Transformationen, zum Beispiel der Änderung der Reihenfolge von Bedingungen sowie Redundanz- und Wiederholungsvermeidung, wie in *Abbildung 4* veranschaulicht, wird hierdurch manuelles Inlining lokaler Routinen durch das INLINE Pragma (PRAGMA INLINE (subprogram, 'YES')) ermöglicht.
- 3: Dieser höchste Level führt neben weiteren Code-Transformationen ein automatisches Inlining lokaler Routinen durch. Dies kann wiederum über das INLINE Pragma (PRAGMA INLINE (subprogram, 'NO')) unterbunden werden.

Insbesondere das Inlining von vielfach aufgerufenen Hilfsroutinen führt zu Performanceverbesserungen, da sich die eigentlichen Methodenaufrufe und damit der jeweils einhergehende Auf- und Abbau des Call Stacks erübrigen. Allerdings führt dies aufgrund des redundanten Codes zwingend zu einem größeren Kompilat und damit zu einem höheren Speicherbedarf während der Ausführung, was nach Ansicht und Erfahrung des Autors jedoch heutzutage vernachlässigt werden kann.

Aber es ist Vorsicht geboten: Insbesondere das automatische Inlining in Verbindung mit den ebenfalls automatisch vom Optimizer vorgenommenen Code-Transformationen beziehungsweise Eliminierungen kann in seltenen Fällen zu einem veränderten Verhalten führen. Ein Beispiel hierfür ist in den *Listings 1 und 2* angeführt, so getestet auf der aktuellen Oracle DB Version 21.12.

Bei der Kompilierung mit PLSQL_OPTIMIZE_LEVEL=2 wird der String abc korrekt als keine Zahl (Returnwert 0) erkannt. Wird die gleiche Routine hingegen mit PLSQL_OPTIMIZE_LEVEL=3 kompiliert, so wird der String abc fälschlicherweise als Zahl (Returnwert 1) erkannt. Dies mag wie ein Bug aussehen, ist es aber tatsächlich nicht. Der PL/SQL-Opti-

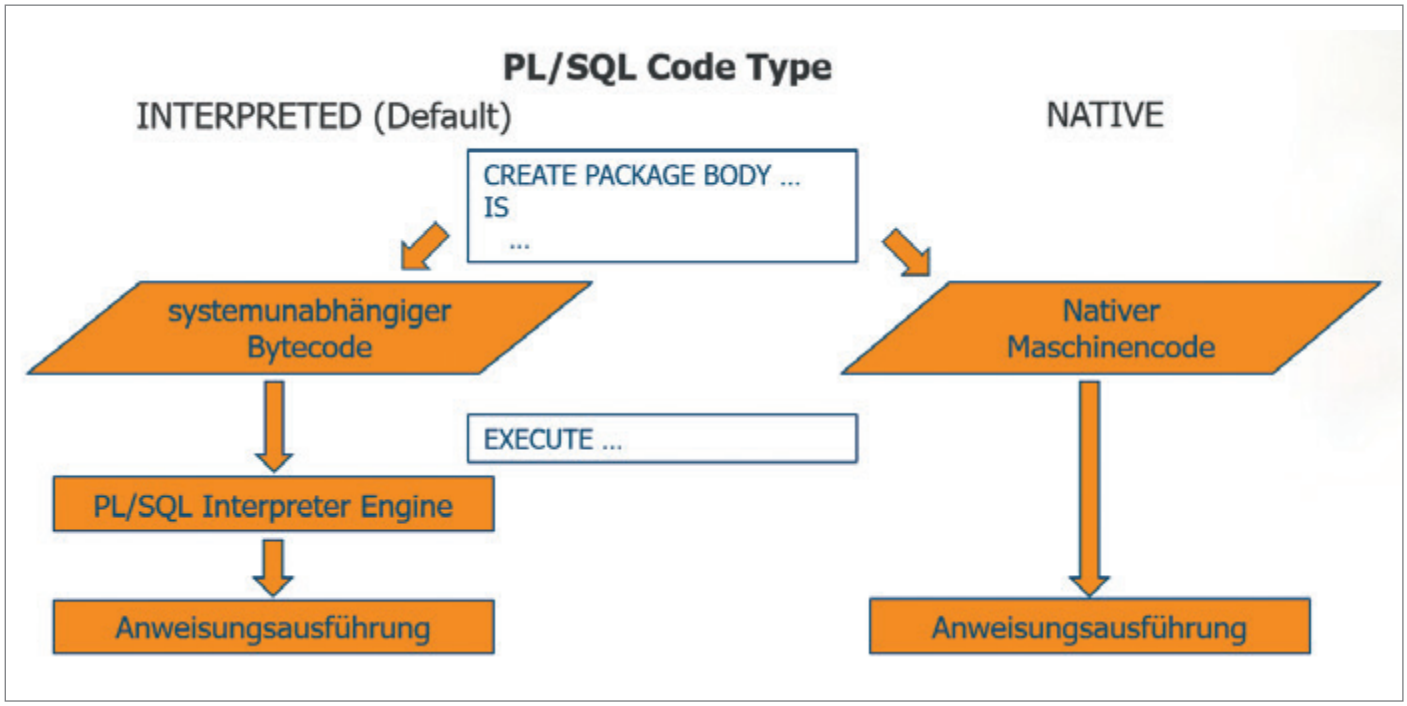


Abbildung 3: PL/SQL Code Type – interpreted vs. Native (Quelle: Jan Gorkow)

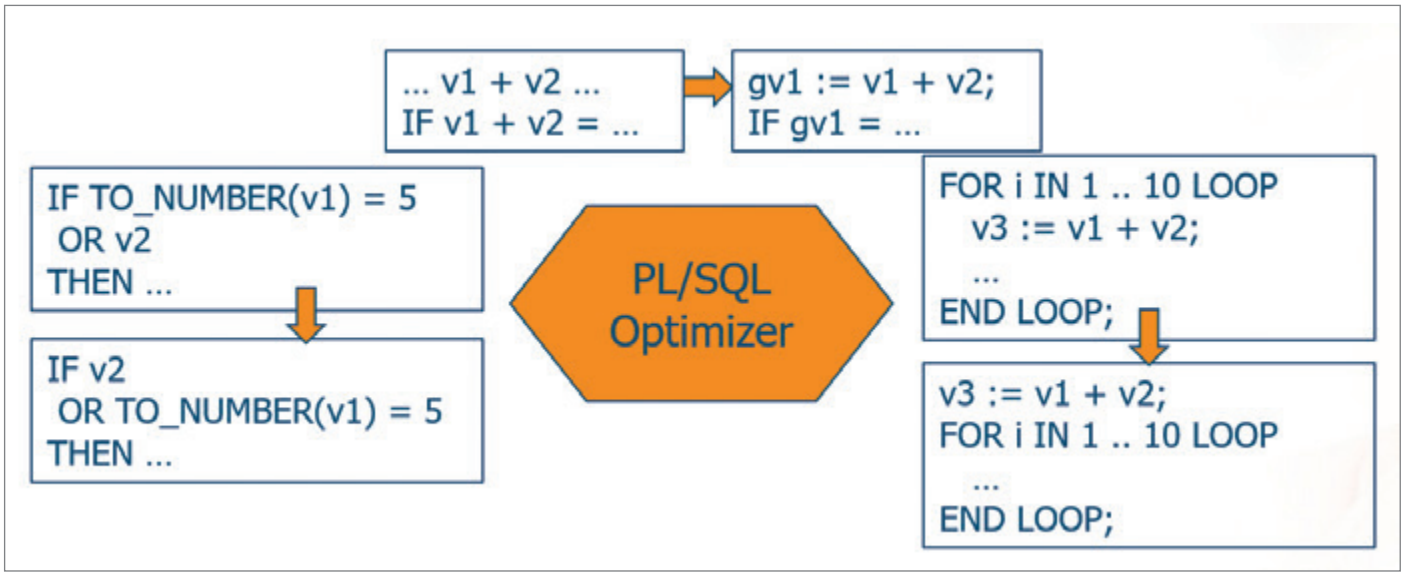


Abbildung 4: PL/SQL-Optimizer – Beispiele für Code-Transformationen (Quelle: Jan Gorkow)

mizer eliminiert bei Level 3 lediglich die entscheidende Codezeile v_test := TO_NUMBER (is_number.numc);, da die Variable *v_test* im weiteren Programmverlauf nicht mehr verwendet wird. Infolgedessen wird keine Exception mehr ausgelöst und somit der Returnwert nicht auf 0 geändert. Eine Lösungsvariante für eine entsprechende Anpassung des Codes ist in *Listing 3* dargestellt.

Hier wird die entscheidende Codezeile nicht eliminiert, da die Variable *v_test* im weiteren Programmverlauf noch benutzt wird. Noch besser als diese Variante ist die mit Oracle 12c eingeführte ON CONVERSION ERROR-Klausel, auf welche an dieser Stelle aber nicht weiter eingegangen werden soll.

Leider gibt es bis dato keine Möglichkeit, sich den aus dem PL/SQL-Optimizer resultierenden Code anzuschauen. Daher sollten die folgenden Empfehlungen des Autors beachtet werden:

- Vor dem Kompilieren sollten sämtliche Compiler Warnings mittels ALTER SESSION SET PLSQL_WARNUNG='ENABLE:ALL' aktiviert werden. Hierdurch erhält der Entwickler eine Reihe von Informationen über die Aktivitäten des PL/SQL-Optimizers, zum Beispiel welche Routinen wo geinlined werden und welche Routinen aufgrund vollständigen Inlinings sämtlicher Aufrufe eliminiert werden.
- PLSQL_OPTIMIZE_LEVEL = 3 sollte nur für selbst geschriebenen Code eingesetzt werden und dieser sollte gut und umfassend auf das gewünschte und erwartete Verhalten hin getestet werden. Bei Abweichun-

```
SET SERVEROUTPUT ON

ALTER SESSION SET PLSQL_CODE_TYPE = native;
ALTER SESSION SET PLSQL_OPTIMIZE_LEVEL = 2;
ALTER SESSION SET PLSQL_WARNINGS = 'ENABLE:ALL';

CREATE OR REPLACE PROCEDURE optimizer_test
   AUTHID DEFINER
IS
   FUNCTION is_number (numc IN VARCHAR2)
      RETURN NUMBER
   IS
      v_test     NUMBER;
      v_result   NUMBER;
   BEGIN
      BEGIN
         v_test := TO_NUMBER (is_number.numc);
         v_result := 1;
      EXCEPTION
         WHEN OTHERS
         THEN
            v_result := 0;
      END;

      RETURN v_result;
   END is_number;
BEGIN
   DBMS_OUTPUT.put_line ('is_number 123: ' || is_number (numc => '123'));
   DBMS_OUTPUT.put_line ('is_number abc: ' || is_number (numc => 'abc'));
END optimizer_test;
/
SHOW ERRORS

EXECUTE optimizer_test();
is_number 123: 1
is_number abc: 0
```

Listing 1: Korrektes Verhalten bei PLSQL_OPTIMIZE_LEVEL = 2

```
ALTER SESSION SET PLSQL_OPTIMIZE_LEVEL = 3;

ALTER PROCEDURE optimizer_test COMPILE;
SHOW ERRORS

EXECUTE optimizer_test();
is_number 123: 1
is_number abc: 1
```

Listing 2: Fehlerhaftes Verhalten bei PLSQL_OPTIMIZE_LEVEL = 3

gen kann der Code in der Regel so umgeschrieben werden, dass das gewünschte Verhalten auch bei höchstem Optimierungslevel erreicht wird.
- Built-in Packages sollten niemals mit einem anderen Level als 2 kompiliert werden, da dies unter Umständen, wie am Beispiel gesehen, zu einem ungewünschten Verhalten führen kann und sich hier Codeanpassungen per se verbieten. Oftmals wären diese ohnehin nicht ohne weiteres möglich, da der Code hier nicht selten ausschließlich in gewrappter Form vorliegt.

Ein kleiner Exkurs zu Rekursionen

Wenngleich die Warnungen des PL/SQL Compilers hier und da einmal anderer Meinung sind, können rekursive Aufrufe verständlicherweise nicht geinlined werden, da Oracle zum Kompilierungszeitpunkt keine Abschätzung über eine maximale Rekursionstiefe treffen kann. Daher findet zur Laufzeit bei rekursiven Aufrufen zwingend ein vielfacher Auf- und Abbau des Call Stacks statt, was wiederum etwas Zeit kostet. Oftmals kann derartiger Code aber auch in Schleifenform implementiert werden. In solchen Fällen muss entwicklungsseitig die Abwägung zwischen Eleganz und Performance des Codes getroffen werden.

Numerische PL/SQL – Datentypen

Oracle bringt eine ganze Reihe numerischer Datentypen für PL/SQL mit, die sich grundsätzlich in zwei Gruppen einteilen lassen:

- Library-basierte Arithmetik: Hierzu zählen NUMER und INTEGER, deren Rechenlogiken über entsprechenden Programmbibliotheken abgebildet werden. Hierdurch wird eine extreme Genauigkeit bei gleichzeitig größtmöglichem Datenbereich zu Lasten der Performance erreicht.
- CPU-basierte Arithmetik: Hierzu zählen unter anderem PLS_INTEGER (32 Bit-Integer), BINARY_FLOAT (32 Bit-Fließkommawert) und BINARY_DOUBLE (64 Bit-Fließkommawert). Der Datenbereich sowie die Genauigkeit dieser Typen sind begrenzt. Dafür werden Berechnungen direkt von der CPU ausgeführt, ohne vorherige Aufbereitung durch eine Library, was zu einer erheblich höheren Performance führt, insbesondere in Kombination mit nativer Kompilierung.

Mit Oracle 11g wurden 3 weitere CPU-arithmetische Datentypen eingeführt:
- SIMPLE_INTEGER
- SIMPLE_FLOAT
- SIMPLE_DOUBLE

Diese basieren jeweils auf ihrem PLS_ beziehungsweise BINARY_-Pendant und unterscheiden sich von diesen lediglich durch ein NOT NULL-Constraint, weshalb entsprechenden Variablen schon bei der Deklaration zwingend ein Wert zugewiesen werden muss. Da hierdurch jedoch entsprechende Prüfungen auf NULL / NOT NULL zur Laufzeit entfallen können, sind Berechnungen auf Basis der SIMPLE_-Typen noch einmal geringfügig schneller.

Wie immer gilt: Kein Vorteil ohne Nachteil! So gibt es auch im Hinblick auf die CPU- Arithmetik-basierten Typen Einschränkungen, derer sich jeder Entwickler bewusst sein muss:

- Keine Overflow-/Underflow-Exceptions: Addiert man beispielsweise zum höchstmöglichen Wert eines SIMPLE_INTEGERs noch 1 dazu, so wechselt das Vorzeichen ins Negative und man erhält den kleinstmöglichen Wert, wie in *Listing 4* zu sehen ist.
- Validierungsprüfungen per Konstanten: Insbesondere im Hinblick auf die Fließkommatypen muss man sich mit den verfügbaren Konstanten zur Validierungsprüfung, zum Beispiel BINARY_DOUBLE_NAN oder BINARY_DOUBLE_MAX_NORMAL, vertraut machen und diese nutzen.
- Fließkommatypen unterliegen grundsätzlich einer begrenzten Genauigkeit

Oracle Datenbanken Monthly News

Auf dem deutschsprachigen Oracle-Blog ist die Januar-Ausgabe der News-Serie erschienen.

DOAG Online

Es ist wieder so weit: die neue Ausgabe ist online! Das sechsköpfige Redaktionsteam von Oracle Deutschland hat wieder Neuigkeiten rund um die Oracle-Datenbank für On-Premises und Cloud-Installation zusammengestellt.

Alles wird wieder in einem Video präsentiert.

In der aktuellen Ausgabe wird wieder ein zusätzliches Quick Link Posting (in Englisch) zur Verfügung gestellt, um einen schnellen Zugriff auf die zugehörigen Links zu gewährleisten.

https://www.doag.org/de/home/news/oracle-datenbanken-monthly-news-30/

```
CREATE OR REPLACE PROCEDURE optimizer_test
   AUTHID DEFINER
IS
   FUNCTION is_number (numc IN VARCHAR2)
      RETURN NUMBER
   IS
      v_test   NUMBER;
   BEGIN
      BEGIN
         v_test := TO_NUMBER (is_number.numc);
      EXCEPTION
         WHEN OTHERS
         THEN
            v_test := NULL;
      END;

      RETURN CASE WHEN v_test IS NOT NULL THEN 1 ELSE 0 END;
   END is_number;
BEGIN
   DBMS_OUTPUT.put_line ('is_number 123: ' || is_number (numc => '123'));
   DBMS_OUTPUT.put_line ('is_number abc: ' || is_number (numc => 'abc'));
END optimizer_test;
/

SHOW ERRORS

EXECUTE optimizer_test();
is_number 123: 1
is_number abc: 0
```

Listing 3: Angepasster Code mit korrektem Verhalten bei PLSQL_OPTIMIZE_LEVEL = 3

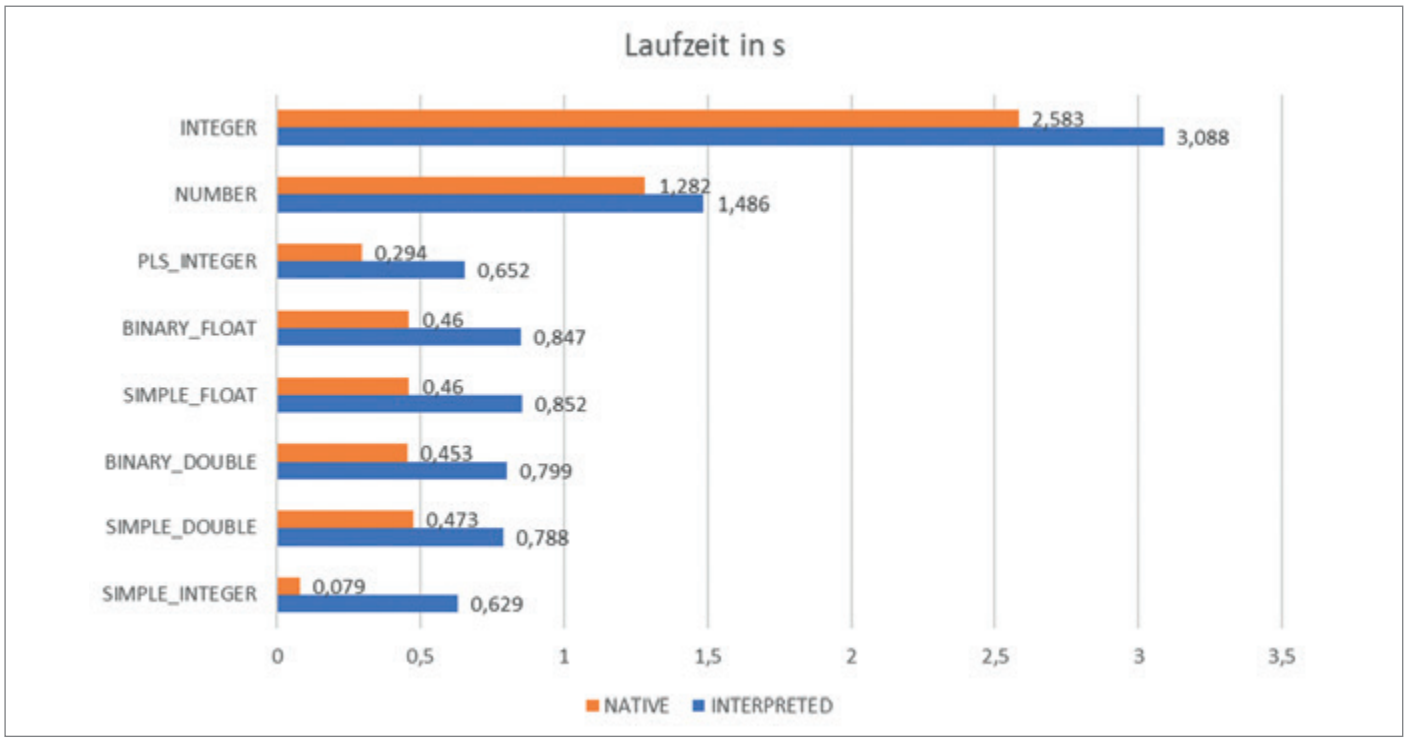


Abbildung 5: Ergebnisse Performancevergleich (Quelle: Jan Gorkow)

```
DECLARE
   v1   SIMPLE_DOUBLE := 0.1;
   v2   SIMPLE_DOUBLE := 0.2;
   v3   SIMPLE_DOUBLE := 0.3;
BEGIN
   DBMS_OUTPUT.put_line (
      'v1 + v2 ' || CASE WHEN v1 + v2 = v3 THEN '=' ELSE '!=' END || ' v3');
   DBMS_OUTPUT.put_line (v1);
   DBMS_OUTPUT.put_line (v2);
   DBMS_OUTPUT.put_line (v3);
END;
/
v1 + v2 != v3
1,0000000000000001E-001
2,0000000000000001E-001
2,9999999999999999E-001
```

Listing 5: Abbildungsungenauigkeiten von Fließkommatypen

```
--ALTER SESSION SET PLSQL_CODE_TYPE = interpreted;
ALTER SESSION SET PLSQL_CODE_TYPE = native;

CREATE OR REPLACE PROCEDURE numeric_performance
   AUTHID DEFINER
IS
   v_value     SIMPLE_INTEGER := 0; --Datentyp zum Testen
   v_step      SIMPLE_INTEGER := 1; --Datentyp zum Testen
   v_runtime   INTERVAL DAY TO SECOND;
   v_start     TIMESTAMP WITH TIME ZONE := CURRENT_TIMESTAMP;
BEGIN
   FOR i IN 1 .. 100000000
   LOOP
      v_value := v_value + v_step;
   END LOOP;

   v_runtime := CURRENT_TIMESTAMP - v_start;

   DBMS_OUTPUT.put_line (
         TO_CHAR (v_value)
      || ' adds in '
      || TO_CHAR (
               EXTRACT (MINUTE FROM v_runtime) * 60
            + EXTRACT (SECOND FROM v_runtime),
            '0D999')
      || ' seconds');
END numeric_performance;
/

EXECUTE numeric_performance();
```

Listing 6: Template für Datentypen – Performance Testcase

Dies führt unter anderem zu Ergebnisrundungen bei der Addition sehr großer mit sehr kleinen Zahlen, zum Beispiel 1015 + 0,01 = 1015. Darüber hinaus können sie dezimale Werte nicht exakt abbilden, was beispielsweise zu vermeintlich folgender Aussage führen kann: 0,1 + 0,2 != 0,3, wie in *Listing 5* gezeigt.

```
DECLARE
   v1   SIMPLE_INTEGER := 2147483647;
BEGIN
   DBMS_OUTPUT.put_line (v1);
   v1 := v1 + 1;
   DBMS_OUTPUT.put_line (v1);
END;
/
2147483647
-2147483648
```

Listing 4: Numerischer Überlauf ohne Exception

Performancevergleich

Doch wie groß kann der Einfluss der Kompilierungsmethode und die Wahl der numerischen Datentypen nun im Extremfall tatsächlich sein? Um dies zu verdeutlichen, sollen für jeden betrachteten Datentyp jeweils 100.000.000 Typ-reine Additionen von 1 zu einem laufenden Wert berechnet werden *(siehe Listing 6)*. Das Ergebnis ist in *Abbildung 5* grafisch dargestellt. Bemerkenswert ist tatsächlich die Spannbreite zwischen den Extremfällen der interpretiert kompilierten Logik unter Verwendung des INTEGER (3,088s) im Vergleich zur nativ kompilierten Logik unter Verwendung des SIMPLE_INTEGER (0,079s), da dies bei identischem Ergebnis ein Unterschied um einen Faktor von 39 ausmacht!

Als Testsystem für diesen Vergleich diente folgender Laptop:

- AMD Ryzen 7 PRO 5850U CPU mit 1,9 GHz Basis und 4,4 GHz maximaler Taktfrequenz
- Oracle 21.12.0.0.0 EE
- Windows 11 Enterprise

Fazit und Empfehlungen

Grundsätzlich lohnt sich der Einsatz der nativen Kompilierung für den eigenen PL/SQL-Code. Ohne eine Zeile Code anpassen zu müssen, kann so die Laufzeit oftmals schon erheblich und risikoarm verringert werden.

Auch mit dem automatischen Inlining via PLSQL_OPTIMIZE_LEVEL=3 hat der Autor bereits in vielen Projekten sehr gute Erfahrungen gemacht. Wichtig ist in diesem Zusammenhang aber, die Implementierungen gut zu testen, um böse Überraschungen, die durch Inlining in Kombination mit Code-Eliminierungen, wie gesehen, auftreten können, zu vermeiden.

Im Hinblick auf die Wahl der Datentypen sollten, wo immer möglich, CPU-Arithmetik- basierte Typen verwendet werden. Diese sollten immer dann in Betracht gezogen werden, wenn mit den angeführten Punkten zur Genauigkeit, zu den Wertebereichen und gegebenenfalls auch zum NOT NULL-Verhalten gelebt, beziehungsweise entsprechend damit umgegangen werden kann. Ansonsten gilt: Bei maximal erforderlichem Wertebereich und Genauigkeit führt kein Weg an dem altbekannten NUMBER-Datentyp vorbei.

Wenngleich sicher nicht jede Business-Anforderung derartig rechenintensiv ist, dass die hier betrachteten Punkte wie im Beispiel eine Performanceverbesserung um das fast 40-fache bewirken könnten, sei abschließend noch eines angemerkt: Selbst wenn Sie in Ihren Projekten aktuell keine Performanceprobleme oder etwaige Anforderungen haben, gilt es zu bedenken, dass die hier angeführten Punkte nicht nur die Performance im Sinne der Laufzeit senken, sondern im gleichen Maße auch die Systemlast verringern. Und hier kommen wir dann plötzlich auch zu Themen wie Skalierbarkeit (zum Beispiel parallele und/oder schnellere Verarbeitung von mehr Benutzer-Requests als bislang) und Nachhaltigkeit (durch geringeren Ressourceneinsatz).

Über den Autor

Jan Gorkow ist gelernter Fachinformatiker Anwendungsentwicklung und hat sich bereits sehr früh in seiner Ausbildung auf Oracle-Datenbankentwicklung, seinerzeit noch unter Oracle 8i, spezialisiert. Seit 2011 ist er als Senior Managing Consultant für die SD&C Solutions Development & Consulting GmbH tätig, welche seit Ende 2021 zur Datacider Gruppe gehört. Mit seinem Team von Oracle-Datenbank- und APEX-Spezialisten konzipiert und realisiert er für Kunden hoch-performante DB-Anwendungen, um das in den Daten steckende Potential zu erschließen und optimale Business-Entscheidungen zu ermöglichen.

Jan Gorkow
jan.gorkow@sd-c.de

Lift & Shift – geht es auch etwas größer bitte?

DI Kurt Rahstorfer (solicon IT), DI Christian Ropposch (Andritz)

Die Andritz AG betreibt und entwickelt für interne Zwecke die „Sales Application Services“ (SAS) basierend auf Oracle APEX. Derzeit werden circa 70 Applikationen weltweit eingesetzt. Neben der Oracle-Datenbank sind auch BI-Publisher (Angebotsgenerierung), SAP-BO (Management-Reports, Dashboards), Jenkins (CI/CD) und etliche zusätzliche Services (als Docker-Container) im Einsatz. Die gesamte SAS-Landschaft sollte in die Oracle Cloud verschoben werden. Nachdem SAS innerhalb der Andritz AG eine zentrale Rolle spielt, waren bei dieser Migration auch viele Schnittstellen zu anderen Kernsystemen zu berücksichtigen. Dieser Artikel skizziert den Projektverlauf samt aktuellem Status und einiger „Lessons Learned“. Im zweiten Teil gehen wir auf einige technische Aspekte ein, die uns im Projektverlauf beschäftigt haben und eventuell auch bei anderen Cloud-Migrationen relevant sind.

Projektverlauf

Was waren die „Knackpunkte“ (org.) beim Projekt wie es bisher gelaufen ist? An welchen Punkten hätten wir anders agieren müssen? Zwei spannende Fragen, die wir hier versuchen, zu klären.

- Frühzeitig ALLE Stakeholder einbeziehen und das Projekt, beziehungsweise die Projektplanung, dem Management vorstellen, so dass allen Beteiligten klar ist, worum es im Projekt geht und wie die Ziele lauten. Zusätzlich ist es auch wichtig, dass man in Kontakt bleibt und sich bezüglich des Projektstatus austauscht.
- Frühzeitig klären, wie benötigte Ressourcen verfügbar sind und sicherstellen, dass dieser Bedarf auch in die Planung des jeweiligen Teams einfließt. Auch hier ist es essenziell, sich immer wieder mal abzustimmen, ob der Plan noch hält. Ansonsten kommt es zu Überraschungen, wenn die Ressource dann doch nicht verfügbar ist, weil das Gegenüber die Planungsänderung nicht oder zu spät kommuniziert hat.

Beide Aspekte sind besonders bei international aufgestellten Organisationen spannend. Wenn man im gleichen Stockwerk oder Gebäude arbeitet, ist das eine Sache. Wenn man aber ein Thema hat, bei dem Stakeholder und Ressourcen weltweit verteilt sind, ist das eine ganz andere Geschichte.

Viele dieser typischen Aktivitäten haben wir adressiert, aber ein paar Punkte erkennen wir jetzt als „suboptimal“:

- Stakeholder: Da waren wir sehr fokussiert auf die direkten Ansprechpartner. Andere Akteure wurden zwar informiert, aber nicht „im Loop“ gehalten. Dies war mit ein Grund, warum ab Frühling 2023 plötzlich wieder eine Cloud-Grundsatz-Diskussion ausgebrochen ist: Das Infrastruktur-Team war nicht ideal eingebunden. Als Konsequenz mussten wir im April nochmal erklären, warum (damals) Azure keine ideale Lösung war.
- Ressourcen: Die Annahme, dass wir auf Ressourcen kurzfristig Zugriff haben, weil sie „eh“ Teil des SAS-Teams sind, hat sich so nicht bewahrheitet. Das SAS-Team als Ganzes hat auch eine abgestimmte Planung mit dem Business bezüglich Terminen für neue Features. Diese Versprechen kann man dann schwer brechen …

Oracle@OCI vs ODSA vs Oracle@Azure

Wie oben schon erwähnt, gab es ab April 2023 die Diskussion, wie weit man den Betrieb der SAS-Infrastruktur nach Azure verlagern könnte. Eine mögliche Lösung war damals ODSA (Oracle Database Service for Azure). Damit ergab sich für SAS eine geteilte Architektur *(siehe Abbildung 1 "Die Architektur-Variante mit ODSA“)*. In diesem Szenario würden nur die Autonomous Databases (ADB) samt Load-Balancer in der OCI bleiben. Der „custom“ ORDS hat ein Fragezeichen, weil er durch das Standard-Setup der ADB abgelöst werden soll. Bei BI-Publisher steht ein Fragezeichen, weil hier noch finale Tests ausständig sind, ob in Azure die vorhandenen Lizenzen ausreichen (Thematik OCPU in OCI vs vCPU in Azure).

Im Rahmen der Oracle Cloud World gab es dann die Ankündigung eines neuen Service „Oracle@Azure“. Oracle Exadata-Systeme laufen direkt in den Azure- Datacentern. In der ersten Iteration ab Januar 2024 (zumindest laut Ankündigung) werden damit in Frankfurt dedicated Exadata-Setups zur Verfügung stehen. Dieses Szenario ist für die Andritz AG interessant, weil es zur Azure-Strategie passt und es außer der SAS-Umgebung noch weitere Oracle-Datenbanken gibt, die früher oder später in die Cloud sollen. Die Architektur ändert sich damit wieder *(siehe Abbildung 2 „Komplett in Azure mit Oracle@Azure“)*.

Derzeit werden noch beide Varianten diskutiert.

Technische "Highlights" während der Migration Export/Import mit Datapump

Auch 2,5 TB sind mittlerweile schnell migriert, wenn man verfügbare Ressourcen entsprechend nutzt. Diese Variante war schnell genug für uns (Export bzw. Import erfolgten mit Parameter PARALLEL = 6), deshalb gab es auch keine weiteren Tests mit anderen Strategien. Der letzte Import hat länger als erwartet gedauert. Es scheint, dass die ADB mittlerweile schon während des Imports mit der Erstellung zusätzlicher Indizes startet. Jedenfalls hatten wir diesmal gleich nach Abschluss des Imports circa 1500 Indizes mehr als im eigenen Datacenter. Dieses „neue“ Verhalten werden wir nochmal prüfen müssen, bevor der Go-Live passiert.

Beim Export macht es Sinn, dass man die Dump-Files verschlüsselt. Unser Parameter-File hatte folgenden Inhalt *(siehe Listing 1)*.

Der Parameter „encryption_pwd_prompt“ ermöglicht die Verschlüsselung. Beim Import wird dieses Passwort wieder benötigt.
Interessant ist auch der OCI-Upload Schritt *(siehe Listing 2)*.

Bei einer „bulk-upload“-Operation hat man auch die Möglichkeit, die Parallelität dieser Operation einzustellen. Dazu dient der Parameter „--parallel-upload-count“. Wir sind beim Standardwert (= 10) geblieben. Wenn man eine FastConnect-Anbindung hat, können höhere Werte den Durchsatz nochmal erhöhen.

```
directory=DUMPDIR
parallel=6

job_name=expdp_sas
dumpfile=sas_dump%u.dmp
logfile=sas_dump.log
consistent=y
encryption_pwd_prompt=yes

# Liste mit Applikationsschemas
SCHEMAS= ...
```

Listing 1: Beispiel Datapump-Parameterfile

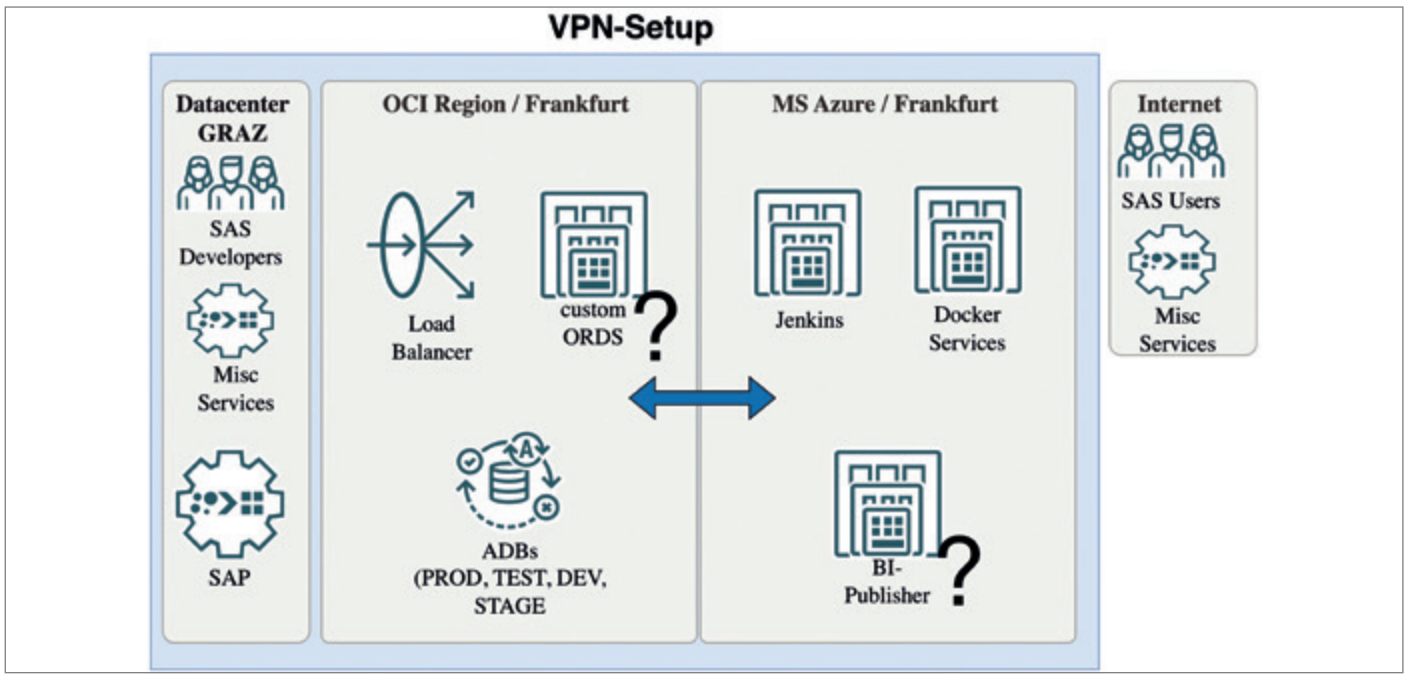


Abbildung 1: Die Architektur-Variante mit ODSA (Quelle: Kurt Rahstorfer und Christian Ropposch)

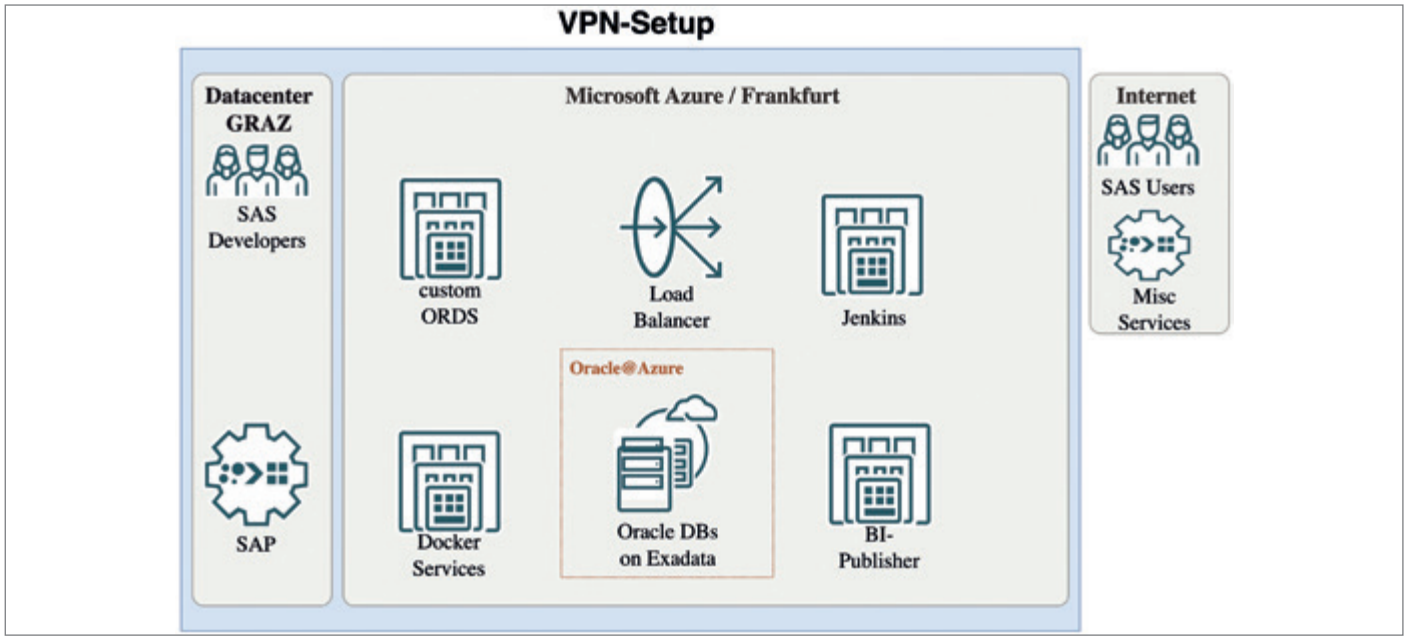


Abbildung 2: Komplett in Azure mit Oracle@Azure (Quelle: Kurt Rahstorfer und Christian Ropposch)

Abbildung 3: Patchlevel-Infos im Oracle Tenant Azure (Quelle: Kurt Rahstorfer und Christian Ropposch)

```
./bin/oci os object bulk-upload -bn SAS-bucket --src-dir /ORANFS/oracle/dumps/GRZPRD03/poc --overwrite
```

Listing 2: Bulk Upload per Commandline

```
BEGIN DBMS_CLOUD.CREATE_CREDENTIAL(
  credential_name => 'KR'
 ,username => 'oracleidentitycloudservice/kr@solicon-it.com'
 ,password => '<Name des auth-token>' );
END;
/
```

Listing 3: DBMS-Credential einstellen

```
directory=data_pump_dir
credential=KR
dumpfile=https://objectstorage.eu-frankfurt-1.oraclecloud.com/n/XXXXXXXXXXXX/b/SAS-bucket/o/sas_dump%u.dmp
parallel=6
```

Listing 4: Datapump-Import Parameterfile

```
-- Diese Grants funktionieren (Beispiele)
grant select on v$session to kurt;
grant select on dba_objects to kurt;

-- Diese Grants funktionieren nicht
grant select on ORDS_METADATA.ORDS_OBJECTS to kurt;
ERROR at line 1:
ORA-01031: insufficient privileges

grant execute on APEX_INSTANCE_ADMIN to kurt;
ERROR at line 1:
ORA-01031: insufficient privileges

grant select on sys.aud$ to kurt;
ERROR at line 1:
ORA-00942: table or view does not exist
```

Listing 5: Vergabe von Grants in der ADB

```
grant APEX_ADMINISTRATOR_ROLE to kurt;

create or replace procedure kurt.apex_privs_test authid current_user
is
  v VARCHAR2(100);
BEGIN
  EXECUTE IMMEDIATE
  'BEGIN :1 := APEX_INSTANCE_ADMIN.get_parameter(''UPGRADE_STATUS''); END;'
  USING OUT v;
  dbms_output.put_line('v = ' || v);
END;
/
```

Listing 6: Ein möglicher Workaround bei fehlenden Grants

Der Import in die ADB funktioniert wie gewohnt. Wir haben dazu einfach eine Linux-VM im Oracle Tenant eingerichtet. Diese Maschine steht jetzt den DBAs zur Verfügung. Falls man in der Datenbank Profile verwendet, muss man sie vorab auch in der ADB anlegen. Allgemein hat es Sinn, zu prüfen, ob der Wert des Parameters PASSWORD_LIFE_TIME im DEFAULT_PROFILE passt. Es gibt noch einen „Stolperstein", damit man den Import starten kann. Die ADB braucht Zugriffsrechte auf den Object-Store. Der DBA benötigt im Tenant ein eigenes Auth-Token für diese Operation. Dieses Token erzeugt man am einfachsten in der Web-GUI. Danach muss man der ADB mitteilen, dass dieses Token verfügbar ist. Beispielsweise in Sql/Plus als ADMIN *(siehe Listing 3)*. Wichtig:

- username = Ein qualifizierter User- oder Servicename im Oracle Tenant.
- password = Der Name des Auth-Tokens, welches man zuvor erzeugt hat.

Die Oracle-Dokumentation hat hier alle Details dazu. *(siehe https://docs.oracle.com/en-us/iaas/autonomous-database-serverless/doc/dbms-cloud-subprograms.html#GUID-742FC365-AA09-48A8-922C-1987795CF36A (Abschnitt „Usage Notes"))*.

Wenn diese Punkte erledigt sind, funktioniert auch der Import. *Listing 4* zeigt das Parameterfile für den Import.

Grants auf Triple O's („Oracle Owned Objects" – „OOOs")

Kurz gesagt: Das geht einfach nicht immer!

Bei einer Oracle ADB gibt es eine Art „Gewaltentrennung". Zugriff auf das OS und auf die CDB sind nicht möglich und Zugriff auf Oracle Schemas (= alles mit ORACLE_MAINTAINED='Y' beziehungsweise CLOUD_MAINTAINED='YES' in der Tabelle DBA_USERS / DBA_OBJECTS) ist nur in den vorgegebenen Grenzen möglich. Als User ADMIN ist die definitive Prüfung möglich. Wenn ADMIN einen Object Grant erteilen darf, ist alles gut. Wenn nicht, dann gibt es keine Möglichkeit diese Berechtigung direkt zu erhalten.

Beispiele *(siehe Listing 5 – Die Statements wurden als ADMIN ausgeführt. Der User „kurt" hat nur minimale Berechtigungen.)*.

Anhand der Fehlermeldungen sieht man gut, wo ADMIN keinen Grant vergeben darf. Auf interne Strukturen von SYS (hier AUD$) hat man gar keinen Zugriff. ADMIN kennt diese Objekte nicht. Diese Beispiele zeigen, dass die Verwendung von „OOOs" in eigenen Applikationen nicht unbedingt ein Showstopper sein muss. Für viele Objekte kann ADMIN (auf einer ADB braucht es für Grants auf SYS-Objekte nicht den SYS-User) die gewünschten Berechtigungen erteilen, so dass der PL/SQL-Code wie gewohnt funktioniert.

Anbei ein kurzes Beispiel *(siehe Listing 6)*, wie in PL/SQL ein Workaround mit dynamischem PL/SQL für den Fall „APEX_INSTANCE_ADMIN" funktionieren könnte.
Kompilieren kann man diese Prozedur immer. Der jeweilige User kann sie aber nur dann ausführen, wenn die Rolle APEX_ADMINISTRATOR_ROLE verfügbar ist. Ob diese Art von Workaround eine mögliche Variante ist, wird man in der konkreten Situation entscheiden müssen. Der Zugriff auf interne SYS-Strukturen (zum Beispiel AUD$) ist momentan nicht möglich.

Flashback

Im SAS-Entwicklungsteam sind „continuous integration & continuous deployment" (CI/CD) schon seit Jahren gelebte Praxis. Dazu gehört auch, dass man eine Umgebung (STAGE) hat, die man beliebig (und schnell!) zurücksetzen kann. Oracle Flashback hat sich hier bewährt. Das DEV-Team kann selbst auf zuvor definierte „guaranteed Restorepoints" zurückspringen, um etwa Deployments nach Fehlern zu wiederholen. Das geht so nicht mehr. Für „guaranteed Restorepoints" benötigt man den SYS-User. Darauf hat man in einer ADB keinen Zugriff. Es gibt aber die Möglichkeit, einen Restore durchzuführen. Damit kann man bis zu 60 Tage zurückspringen. Die Aussicht, mehrmals täglich einen Restore einer circa 700 GB großen DB anzustoßen, hat uns nicht wirklich begeistert. Aber bei ersten Tests hat sich herausgestellt, dass die ADB hinter den Kulissen sehr wohl mit Flashback arbeitet, wenn es möglich ist. Ansonsten wären Restore-Zeiten von unter 5 Minuten kaum zu erklären.

Patching bei ABDs

„Es ist Montagmorgen und meine ADB läuft mit Oracle 19.21, obwohl das Release Update erst in 4 Wochen kommt. Was zur Hölle …" So oder so ähnlich mag ein DBA reagieren, wenn er von einem ADB-Wartungsfenster erstmalig überrascht wurde. Es ist in der Tat so, dass RUs auf (shared) ADBs circa 4 Wochen früher eingespielt werden, bevor sie an den quarterly Release-Terminen allgemein verfügbar sind. Auf einer shared ADB hat man keine Möglichkeit, diese RU-Termine zu beeinflussen. Bei einem „dedicated"-Setup kann man RU-Termine eine Zeit lang (bis zu 2 Quartalen) hinauszögern.

Abbildung 3 („Patchlevel-Infos im Oracle Tenant Azure") zeigt eine ADB mit Patch-Level „Regular". Beim Erstellen der ADB gibt es auch die Möglichkeit, Patch-Level „Early" auszuwählen. Auf derartigen ADBs kommen die RUs nochmal circa 7-10 Tage früher. Es kann zum Beispiel Sinn machen, die PROD-Umgebung auf „Regular" einzustellen, aber für DEV und TEST „Early" auszuwählen. So hat man zumindest noch ein kurzes Zeitfenster, falls ein RU „Auffälligkeiten" zeigt. Die Einstellung „Regular" (default) oder „Early" muss beim Erstellen der DB passieren. Eine nachträgliche Änderung ist nicht möglich.

Ähnliches gilt für Oracle APEX. Allerdings kann man hier ein Upgrade etwas (bis zu 90 Tagen) hinauszögern. In der Oracle-Dokumentation *(siehe https://docs.oracle.com/en/cloud/paas/autonomous-database/serverless/adbsb/apex-apply-defer-updates.html#GUID-1DE968CA-A438-444F-B00C-396012397F0A - beziehungsweise mit der Suche nach „Defer Oracle APEX Upgrades")* ist zu finden, wie man das einstellt.

Sachen, die so in ADBs nicht mehr funktionieren: XMLDB Webservices

Seit der Version 11g kann Oracle mittels XMLDB Webservice-Funktionalität bereitstellen. Mittlerweile wurden in SAS viele dieser „legacy" Services durch REST abgelöst. Bei ersten Tests zeigte sich, dass die noch vorhandenen Webservices in einer ADB nicht mehr funktionieren. Die Oracle-Dokumentation

ist dazu recht klar. Die XMLDB Features sind in einer ADB nur mehr recht eingeschränkt verfügbar. *(siehe auch: https://docs.oracle.com/en-us/iaas/autonomous-database-serverless/doc/autonomous-xml-db.html)*

Ein möglicher Workaround ist, dass man die Oracle Integration Cloud für diese Services einsetzt.

Endpunkte mit „selfsigned"-Zertifikaten

Der Zugriff auf externe Ressourcen von der ADB aus funktioniert sehr gut. Die ADB hat ein internes Wallet mit Root-Zertifikaten von (mehr oder weniger) allen großen Anbietern. Man muss sich nur mehr darum kümmern, dass die ACLs (und Firewalls) entsprechend eingestellt sind und schon klappt der Zugriff auf externe Webservices. Auf das interne Wallet der ADB hat man keinen Zugriff. Es besteht keine Möglichkeit „custom" Wallets zu definieren, deshalb kann man auch nicht mit „selfsigned"-Zertifikaten arbeiten.

Eine mögliche Lösung ist es, einen Proxy (mit einem gültigen Zertifikat) einzurichten, der Anfragen an derartige Ziele routet.

mTLS/TLS

Mittlerweile funktionieren die meisten (aktuellen) Applikationen mit mTLS einwandfrei. Wir sind dann aber doch auf TLS gewechselt, weil es Probleme bei alten (JAVA)-Applikationen gab. Eine spezielle Situation mit einer JAVA-basierten SAP-Integration bereitete uns einiges an Kopfzerbrechen. Da hat auch TLS nicht funktioniert. Nach Recherchen stellte sich heraus, dass dort die maximale Länge für Cipher-Keys („javax.crypto.Cipher.getMaxAllowedKeyLength") auf 128 Bit eingestellt war. Das ist für die Verbindung mit einer ADB zu wenig!

Landing-Zone Blueprints als „Best Practice"

Oracle bietet seit einiger Zeit fertig vorkonfigurierte Landing Zones für die OCI an *(siehe https://github.com/oracle-devrel/technology-engineering/blob/main/landing-zones/README.md)*. Momentan sind zwei Ausprägungen verfügbar:

- CIS LANDING ZONE (CIS LZ)
- ORACLE ENTERPRISE LANDING ZONE (OELZ)

Für eine ausführliche Betrachtung fehlt hier der Platz, aber wenn man ein neues OCI-Migrations-Projekt startet, lohnt es sich, einen Blick auf diese Blueprints zu werfen. Bei kleineren Setups kann die CIS LZ eventuell direkt übernommen werden. Jedenfalls hat man damit auch viele Operations-, Security- und Compliance-Themen adressiert.

Fazit

„Lift & Shift" war ein häufig verwendetes Schlagwort in den letzten Jahren, um hervorzuheben, wie einfach man Datenbanken in die Oracle Cloud bewegen kann. Auf technischer Ebene funktioniert das sehr gut. Auf die eine oder andere Überraschung sollte man sich trotzdem gefasst machen, speziell wenn man viel Applikationslogik in PL/SQL hat. Eine Applikation auf Enterprise-Level ist meist mit einer Vielzahl von anderen Applikationen integriert. Auch wenn „Lift & Shift" schnell von der Hand geht, hat man mit den ganzen Applikations- und Integrationstests ein großes Migrationsprojekt!

Über die Autoren

DI Kurt Rahstorfer, CTO der solicon IT GmbH, schätzt die Oracle-Datenbank seit Version 7.3. und beschäftigt sich mit unterschiedlichsten Data-Management- und Data-Engineering-Themen im Kontext „Cloud" und „Big Data". Motto: „It's all about information!"

DI Christian Ropposch ist in der Andritz-Gruppe im Bereich der Sales Application Services für die Sales Platform (build on Oracle APEX) zuständig. Er beschäftigt sich seit über 25 Jahren mit Oracle-Datenbanken und seit mehr als 10 Jahren auch mit Oracle APEX.

DI Kurt Rahstorfer
kurt.rahstorfer@solicon-it.com

DI Christian Ropposch
christian.ropposch@andritz.com

Compliance im Unternehmen – was ist wirklich wichtig?

Sandra Leist, Werk3 für Datenschutz

„Compliance“ – ein Wort, das uns Unternehmer und Unternehmerinnen täglich begegnet. Doch was bedeutet es? Es geht darum, dass die rechtlichen und unternehmerischen Vorgaben eingehalten werden. Weiterhin geht es darum, die möglichen Risiken zu kennen und vorher durch geeignete Maßnahmen zu minimieren. [1] Die Datenschutzgrundverordnung gibt seit Mai 2018 verschiedene Regelungen vor, die bei der Verarbeitung von Daten beachtet werden müssen. Doch welche Regeln sind das? Was muss und was sollte tatsächlich bedacht werden? Wie sieht die Umsetzung im Alltag aus?

In zehn einzelnen Schritten stelle ich Ihnen die wichtigsten Vorgaben dar, die jedes Unternehmen betreffen, wenn Sie Daten verarbeiten, in denen immer ein Mensch, also eine natürliche Person, identifiziert werden kann. Die DSGVO spricht von personenbezogenen Daten. Dazu gehören die persönlichen Merkmale wie der Name, das Geburtsdatum oder das Foto. Aber es gehören auch die Konto-, die Fahrgestell- oder Personalnummer dazu. Weiterhin kann man durch eine IP-Adresse, eine Gerätenummer oder Besitztümer einen Menschen identifizieren. Eine besondere Kategorie stellen die sensiblen Daten dar. Dazu gehören die Gesundheitsdaten, biometrische und genetische Daten. Die Liste kann vielfältig fortgeführt werden. Sie merken schon – das Thema Daten ist komplex. Dazu habe ich eine Checkliste erstellt, die die wichtigsten Inhalte der Datenschutzgrundverordnung *[2]* in Verbindung mit dem Bundesdatenschutzgesetz *[3]* abbildet.

Bestellung eines Datenschutzbeauftragten

Ab einer Größe von 20 Mitarbeitenden, die regelmäßig mit der Verarbeitung von personenbezogenen Daten beschäftigt

sind, wird ein Datenschutzbeauftragter offiziell benötigt. Dieser wird der zuständigen Aufsichtsbehörde des Bundeslandes benannt. Das kann ein interner Mitarbeitender sein, wozu auch der IT-Administrator gehört, jedoch keine Person mit Führungsaufgaben. Oder es übernimmt ein externes Unternehmen. Bei der Verarbeitung von besonders sensiblen Daten oder sehr hochwertigen Gütern, wird ein Datenschutzbeauftragter empfohlen. An der Stelle ist zu erwähnen, dass alle Anforderungen, die jetzt folgen auch umgesetzt werden müssen, selbst wenn kein Datenschutzbeauftragter bestellt wurde.

Unternehmen mit den Mitarbeitenden

Der Schutz der Daten ist nur so gut, wie die Mitarbeitenden, die die Arbeit im Alltag umsetzen, die Mechanismen kennen. Deswegen wird in dem Bereich mit einer Verschwiegenheitserklärung, einer regelmäßigen Sensibilisierung (Schulungen), verschiedenen Berechtigungen und Aufgabenverteilungen nach dem „Need-to-Know-Prinzip" gearbeitet. Dazu erhält das mobile Arbeiten immer mehr an Beliebtheit und Akzeptanz. Zum Schutz der Mitarbeitenden und der Daten werden Richtlinien entworfen, die einen sicheren Umgang möglich machen.

Weitere Richtlinien können sein: Passwort einrichten, Bring your own device, IT-Sicherheit.

IT- und Informationssicherheit

Die DSGVO gilt für jedes Stück Papier und jede Datei in den technischen Geräten. Dazu zählt die ausgedruckte Bewerbung, die Notizen beim Telefonat, der Drucker, der Schredder, Passwörter und jede Datensicherung, ob im Netzwerk oder in einer Cloud. Hier gilt es entsprechende Zutritts- und Zugriffsregelungen schriftlich festzulegen, Technik und Arbeitsabläufe vor Ort zu überprüfen und die Datenverarbeitung bis zum letzten Vernichten und Löschen zu durchdenken. Dies wird in den sogenannten technischen und organisatorischen Maßnahmen niedergeschrieben.

Dokumentationen

Eine weitere Dokumentation, nach der die Datenschutz-Aufsichtsbehörde fragt, ist das Verzeichnis von Verarbeitungstätigkeiten. Hier werden alle Prozesse, in denen Daten verarbeitet werden, aufgeschlüsselt. Dazu kommt die passende Rechtsgrundlage, der Zweck und die Löschfristen. Als Beispiel ist die Buchhaltung, der eigentliche Geschäftszweck oder der Bereich Marketing zu erwähnen. Daraus ergeben sich weitere Konzepte, wie das Lösch-, Schulungs- und Berechtigungskonzept.

Informationen

Die Informationspflicht wird seitens des Verantwortlichen mit den sogenannten Datenschutzhinweisen erfüllt. Dies kennen Sie bereits von den Websites, die neben dem Impressum eine eigene Seite vorhalten, in der die Datenverarbeitung online beschrieben wird. In Artikel 13 und 14 DSGVO *[4]* finden Sie dazu die erforderlichen Schwerpunkte, die erwähnt werden müssen. Dazu kommen alle Tools, Social-Media-Seiten und Tracking-Verfahren, die tatsächlich im Einsatz sind. Hier gilt nicht: „Viel hilft viel!", sondern schreiben Sie nur die Verarbeitungen rein, die genutzt werden.

Diese Datenschutzhinweise sind immer dann Pflicht, wenn Sie als Verantwortlicher Daten von Kunden, Bewerbern, Mitarbeitenden oder Interessenten das erste Mal aufnehmen. Dazu zählen zum Beispiel der Eingang einer E-Mail, die Ausschreibung einer freien Stelle oder das Abschließen eines Vertrages vor Ort im Laden. Ich empfehle Ihnen, eine zentrale Stelle digital vorzuhalten, zum Beispiel die Website und ein Exemplar in Papierform. Weisen Sie dazu in allen Dokumenten und in Ihrer E-Mail auf die Datenschutzhinweise hin.

6. Kooperationspartner

Die Website benötigt einen Hosting-Anbieter, die IT benötigt einen Administrator und die Büroräume eine externe Reinigungsfirma. Ihnen fallen bestimmt weitere Partner ein, mit denen Sie zusammenarbeiten und die für Sie, die Daten Ihrer Kunden verarbeiten. Dann benötigen Sie eine sogenannte Auftragsverarbeitungsvereinbarung. Hier ein kleines Beispiel: Der Website-Besucher hinterlässt Daten wie die IP-Adresse. Wenn Sie weitere Analyse-Tools nutzen, werden viele weitere Daten erhoben. Der Hosting-Anbieter hat technisch gesehen Zugriff auf diese Daten, die ihm nicht gehören. Denn Sie bleiben weiterhin verantwortlich für die Verarbeitung dieser Daten. Die Auftragsverarbeitungsvereinbarung legt Pflichten fest, die der Hosting-Anbieter einhalten muss, damit ein konformer Umgang mit den Daten und eine Absicherung dieser gewährleistet wird.

7. Betroffene

Der Begriff der Betroffenen beinhaltet alle Kunden, Bewerber, Mitarbeitenden, Ehemalige oder Kooperationspartner. Dazu gibt es verschiedene Betroffenenrechte, die geltend gemacht werden können. Die bekanntesten Rechte sind aktuell das Auskunfts- und Löschersuchen oder das Beschwerderecht. Beim Auskunftsersuchen geht es darum, dass der Betroffene die Anfrage stellt, welche Daten Sie im Unternehmen verarbeiten, zu welchem Zweck dies erfolgt und welcher Herkunft die Daten sind. Damit wird oft das Löschersuchen gekoppelt. Es erfolgt demnach ein Antrag, die Daten zu löschen. Für Sie als Unternehmer ist wichtig, dass Sie innerhalb der Frist von vier Wochen antworten. Diese Antwort benötigt bestimmte Inhalte, daher empfehle ich hier immer ein beratendes Datenschutzunternehmen hinzuzuziehen. Dem Antrag auf Löschen kann oft nur bedingt entgegengekommen werden, denn gesetzliche Aufbewahrungsfristen geben klare Vorgaben, an die Sie sich halten müssen. Dazu gehören zum Beispiel steuerrechtliche Pflichten nach § 147 AO *[5]* und handelsrechtliche Pflichten nach § 257 HGB *[6]*. Das Beschwerderecht ist ein Instrument, durch das der Betroffene bei der Datenschutz- Aufsichtsbehörde sein Anliegen darstellen kann. Ein bekannter Fall war im letzten Jahr die Nutzung von Schriftarten auf der Website, ohne die Einwilligung einzuholen,

Daten in ein Nicht-EU-Land weiterleiten zu dürfen. Ein weiteres Beispiel ist die Frage von abgelehnten Bewerbern, die nach sechs Monaten nachfragen und erfahren, dass ihre Daten noch nicht gelöscht sind. Hier könnte ich noch viele Beispiele aufzählen. Wichtig für Sie ist – wenn eine Beschwerde erfolgt, dann schreibt Sie die Aufsichtsbehörde an und Sie müssen Stellung dazu nehmen. Oft sind diese Anschreiben gefüllt mit der Vorgabe, bereits hier erwähnte Dokumentationen mitzusenden. Meine Empfehlung an der Stelle ist, dem fristgerecht nachzukommen.

8. Risikoeinschätzung

Nutzen Sie in Ihrem Unternehmen die Videoüberwachung? Wollen Sie mit Ländern zusammenarbeiten, die ein anderes Datenschutzniveau haben? Arbeiten Sie mit dem sogenannten „Scoring-Verfahren"? Arbeiten Sie mit großen Mengen an Daten? Dies sind nur einige Beispiele, wann Sie diese Risikoeinschätzung durchführen müssen. Die niedersächsische Datenschutzbehörde hat dazu eine Liste herausgegeben *[7]*, die weitere Beispiele für eine sogenannte Datenschutzfolgeabschätzung benennt. Wichtig an der Stelle zu erwähnen ist, dass die kurz DSFA genannte Abschätzung vor dem Einrichten der Datenverarbeitungen durchdacht und vollzogen wird.

Weiterhin werden Sie eine Risikoeinschätzung benötigen, wenn es um einen Datenschutzvorfall, -verstoß geht. Der Bildschirm wird schwarz, eine Nachricht geht auf mit der Nachricht, dass Ihre Daten verschlüsselt wurden und Sie bezahlen sollen für das Entschlüsseln. Ein Mitarbeitender versendet eine E-Mail mit allen Adressen im CC und nicht im BCC. Der Laptop wird auf dem Nachhauseweg aus dem Auto gestohlen, während Sie einkaufen. In jedem dieser Fälle beginnt ab Bekanntwerden eine Frist von 72 Stunden. In dieser Zeit müssen Sie, am besten mit ihrem Datenschutzbeauftragten, das Ausmaß des Vorfalls, die Menge der Daten, den Schaden der Betroffenen definieren und entscheiden, ob eine Meldung an die Aufsichtsbehörde erfolgt und danach, ob zusätzlich die Betroffenen informiert und welche Maßnahmen eingeleitet werden müssen. Die Frist ist bindend.

Abbildung 1: Checkliste (Quelle: Sandra Leist)

9. Rechenschaftspflicht

Eine weiteres Schriftstück ist der sogenannte Rechenschaftsbericht. Dieser Bericht stellt die Zusammenfassung des Geschäftsjahres zum Thema Datenschutz dar. In diesem Bericht werden alle Vorfälle, Betroffenenanfragen, Schulungen, Dokumentationen und weitere Vorkommnisse aufgelistet. Selbst wenn es keine Vorfälle gab, wird dies im Bericht vermerkt. Die Aufsichtsbehörde fragt nach diesem Bericht, da dieser einen zusammenfassenden Einblick ermöglicht, wie Sie mit dem Thema „Compliance im Unternehmen" umgehen. Nun steht der Bericht für 2023 an.

Haben Sie einen zentralen Ordner im Unternehmen, in dem alle Vorkommnisse dokumentiert wurden oder müssen Sie suchen? Eine kleine Empfehlung von mir – entwerfen Sie auch hier einmal einen Plan, wie diese Informationen gesammelt werden.

10. Notfallplan

In meinem Berufsalltag merke ich oft, dass sich Unternehmer nicht mit den Fällen „Was wäre, wenn…" beschäftigen wollen. Doch das ist fatal. Gibt es in Ihrem Unternehmen eine Stellvertreterregelung? Weiß eine Vertrauensperson,

wo die Passwörter liegen, damit Gelder überwiesen werden können, wenn Sie es nicht machen können? Wer ist Ansprechpartner bei einem Elektroschaden? Wie schnell sind die Daten nach einem Verlust wieder hergestellt? Die Liste der möglichen Notfälle ist lang. Naturschäden, Stromausfall, Zerstörung von Hardware oder Ihr Fehlen als Geschäftsführung – Sie benötigen einen Plan B, schriftlich festgelegt und vertrauensvoll abgelegt *(siehe Abbildung 1)*. Im Safe, beim Notar... Ihnen fällt bestimmt ein passender Ort ein.

Compliance – ist machbar!

Zehn Schritte auf dem Weg zur Sicherheit. Mit diesen Schritten schaffen Sie Sicherheit für Ihre Kunden, für Ihre Mitarbeitenden, für Sie selbst. Sicherheit schafft Vertrauen und Vertrauen macht es möglich, gemeinsam die Zukunft zu gestalten. Gehen Sie die zehn Schritte durch, einer nach dem anderen. Wenn Sie sich auf den Weg machen, werden Ihre Mitarbeitenden mitgehen. Vielleicht klingen diese zehn Schritte nach viel Arbeit, aber ich verspreche Ihnen, es wird weniger, leichter und umsetzbarer.

Quellen

[1] Vgl. *https://www.juraforum.de/lexikon/compliance*, 18.12.2023.
[2] Vgl. *https://dsgvo-gesetz.de*, 18.12.2023.
[3] Vgl. *https://www.gesetze-im-internet.de/bdsg_2018/index.html*, 18.12.2023.
[4] Vgl. *https://dsgvo-gesetz.de/art-13-dsgvo/*, 18.12.2023.
[5] Vgl. *https://www.juraforum.de/gesetze/hgb/257-aufbewahrung-von-unterlagen-aufbewahrungsfristen*, 18.12.2023.
[6] *Vgl. https://www.juraforum.de/gesetze/ao/147-ordnungsvorschriften-fuer-dieaufbewahrung-von-unterlagen*, 18.12.2023.
[7] *https://www.lfd.niedersachsen.de/startseite/datenschutzrecht/ds_gvo/liste_von_verarbeitungsvorgangen_nach_art_35_abs_4_ds_gvo/muss-listen-zur-datenschutzfolgenabschatzung-179663.html*, 18.12.2023.

Über die Autorin

Meine Vision ist es, gemeinsam die Zukunft zu gestalten. Ich bin die Geschäftsführerin der Werk3 für Datenschutz GmbH. Wir beraten und begleiten Unternehmen, in den eben genannten und vielen weiteren Schwerpunkten der Datenschutzgrundverordnung. Unsere Aufgabe ist es, die Menschen und das Unternehmen kennenzulernen, um dann gemeinsam das Datenschutzkonzept zu erarbeiten. Wir arbeiten eng mit den IT-Administratoren und dem Qualitätsmanagement zusammen. Wir entwickeln leicht umsetzbare, durch für Sie maßgeschneiderte Lösungen. Wir geben Ihnen Sicherheit durch Kompetenz und Qualität, die immer auf der Basis der aktuellen Rechtsprechung beruht. Wir arbeiten vorausschauend durch den Einsatz moderner Technologien. Zukünftig wird das Thema „künstliche Intelligenz" weiterwachsen. Durch unsere individuelle Beratung und Begleitung können wir KI im Unternehmen absichern und somit Prozesse und Mitarbeitende binden. Mit viel Begeisterung und Expertise sind wir die Problemlöser in Sachen Datenschutz. Mein Name ist Sandra Leist. Ich bin überzeugt, der anwendbare Datenschutz schafft Vertrauen, bietet Sicherheit und ist richtungsweisend für Ihre Zukunft.

Sandra Leist
mail@datenschutz-werk.de

Analyse der CIS-Report-Empfehlungen zu GLOBAL_NAMES

Christian Pfundtner, DB Masters

Aktuell ist das Thema Datenbank-Security sehr wichtig und dringend geworden, nachdem es in den letzten 20 Jahren kaum von Interesse war. Viele Anbieter nutzen die CIS-Benchmarks (tm) – beispielsweise für die Oracle-Datenbank 19c – als Quelle für Ihre Checks und Empfehlungen. Im CIS-Benchmark steht unter anderem [1]: "This CIS Benchmark was created using a consensus review process comprised of a global community of subject matter experts. The process combines real world experience with data-based information to create technology specific guidance to assist users to secure their environments. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal." Was nicht zu finden ist, ist der Hinweis, dass die Umsetzung der Empfehlungen ohne weiteres Hinterfragen dazu führen kann, dass die Applikation nicht mehr funktioniert! In diesem Artikel schauen wir uns die Empfehlung und die Auswirkungen der Umsetzung genauer an.

Zuerst werfen wir einen Blick auf den Instance-Parameter global_names. Die folgenden Zeilen sind vom CIS-Benchmark übernommen *[1]*:

Description:

The global_names setting requires that the name of a database link matches that of the remote database it will connect to. This setting should have a value of TRUE.

Rationale:

Not requiring database connections to match the domain that is being called remotely could allow unauthorized domain sources to potentially connect via brute-force tactics.

Diese simple Änderung eines Instanz-Parameters kann Applikationen, die Datenbank-Links nutzen, **funktionsunfähig** machen. Abgesehen davon, ist die Begründung mehr als nur schwach. Warum soll sich ein Angreifer die Mühe machen, eine Datenbank und DB-Links anzulegen, nur um brute-force-Attacken auf Benutzer und Passwörter durchzuführen? Das geht doch viel einfacher und mit weniger Aufwand mit jedem beliebigen Oracle Client!

Welche Auswirkung hat diese Änderung?

Damit die Auswirkungen besser verständlich sind, muss man die Bestandteile des GLOBAL_NAMES kennen und verstehen.

Die Verwirrung wird noch größer, da es einerseits den Instance-Parameter global_names und andererseits den GLOBAL_NAME der Datenbank gibt. Der Instance-Parameter global_names kann nur die Werte TRUE oder FALSE enthalten.

Der GLOBAL_NAME setzt sich aus dem Namen der Datenbank (db_name) und der Datenbank Domain (db_domain) zusammen. Der DBCA schlägt die DNS-Domain als db_domain vor, was aber nicht verpflichtend ist und nur eine der Möglichkeiten ist.

In der Praxis wird die db_domain auch gerne genutzt, um die verschiedenen Umgebungen (PROD, DEV, TEST, QA, ...) zu unterscheiden, sofern man dies nicht mittels db_name umsetzt.

Was der CIS-Benchmark und alle uns bekannten Security Scanner verschweigen: wird GLOBAL_NAMES auf TRUE gesetzt, müssen folgende Punkte beachtet werden:

- Mittels "ALTER DATABASE RENAME GLOBAL_NAME TO '<db_name>.<db_domain>';" muss der Globale Name der Datenbank korrigiert werden. Erfolgt dieser Schritt nicht, ist kein Connect via Datenbank-Link auf diese Datenbank möglich.
- Es müssen alle Datenbanken, die über DB-Links miteinander verknüpft sind, gleichzeitig umgestellt werden. Ein Connect zwischen Datenbanken mit unterschiedlichen Einstellungen ist nicht möglich.
- Bei allen Datenbank-Links wird automatisch die DB_DOMAIN angehängt. Da diese mit Sicherheit nicht entsprechend bedacht wurde, würde voraussichtlich kein einziger DB-Link mehr funktionieren.
- Bei der Nutzung eines DB-Links wird überprüft, ob der DB-Link-Name mit dem GLOBAL_NAME der Zieldatenbank übereinstimmt. Dies wird in der Praxis meist nicht der Fall sein!
- Wegen dieser Anforderungen darf es pro Ziel-Datenbank nur einen DB-Link geben.
- Somit müssen voraussichtlich alle DB-Links gelöscht und mit geänderten Namen angelegt werden. Da die Namen der DB-Links auch in vielen Datenbankobjekten (Views, Synonyme, PL/SQL und Java Code) vorkommen können, müssen diese dort meist ebenfalls geändert werden.

Somit hat die Änderung von GLOBAL_NAMES eine massive Auswirkung auf den Betrieb von Applikationen, die Datenbank-Links benötigen. Das kann von einer einfachen Betriebsstörung bis zum Stillstand der Anwendung reichen.

Das Fatale an genau dieser Empfehlung ist, dass sie keinerlei Rücksicht darauf nimmt, ob Datenbank-Links genutzt werden oder nicht. Es wird lediglich der aktuelle Wert von global_names abgefragt. Nutzt man keine Datenbank-Links, führt die Änderung des Parameters selbstverständlich zu keinen weiteren Problemen, da nur bestehende DB-Links betroffen sind. Die Änderung bewirkt also keinerlei Verbesserung der aktuellen Datenbank-Security und ist somit sinnlos.

In einer realen Datenbank-Umgebung muss man jedoch noch viele weitere Punkte beachten, bevor man den Parameter ändern darf:

- Nutzt man Datenbank-Links?
- Welche anderen Datenbanken werden mit diesen Links angesprochen? Haben diese ihrerseits weiterführende Links (es müssen alle Datenbanken gleichzeitig umgestellt werden)?
- In welchen Datenbank-Objekten werden die DB-Links referenziert (Views, Synonyme, PL/SQL und Java Code)?
- In welchen SQL-Statements werden die DB-Links genutzt?

Wie kann man die aktuellen Einstellungen sowie die vorhandenen Datenbanklinks prüfen?

Ein einfaches SQL-Skript zum Prüfen der relevanten Einstellungen sieht wie folgt aus *(siehe Listing 1)*.

Mit diesem Skript kann man auf einen Blick feststellen, wie die Konfiguration aktuell aussieht.

Wenn man nun die Einstellung von global_names einfach auf TRUE ändert, wird der Zugriff auf Datenbanklinks voraussichtlich mit folgender Fehlermeldung enden:

```
02085, 00000, "database link %s connects to %s"
// *Cause: a database link connected to a database with a different name.
// The connection is rejected.
// *Action: create a database link with the same name as the database it
// connects to, or set global_names=false.
```

Durch Anpassungen von db_domain, GLOBAL_NAMES, Connect Strings und der Neuanlage von Datenbank-Links kommt man einige Schritte weiter. Allerdings wird man möglicherweise die vorhandenen Datenbank-Links nicht mehr los, weil beim Versuch diese zu droppen, folgender Fehler auftrifft:

```
02024, 00000, "database link not found"
// *Cause: Database link to be dropped is not found in dictionary
// *Action: Correct the database link name
```

```
set pages 50 lines 160 tab off
col GLOBAL_NAME for a12
col NAME for a20
col VALUE for a20
col owner for a16
col db_link for a16
col username for a16
col host for a16

SELECT * FROM global_name;

SELECT con_id, name, value
  FROM V$SYSTEM_PARAMETER
 WHERE name in ('db_name','db_domain','global_names');

SELECT owner, db_link, username, host
  FROM dba_db_links;
```

Listing 1: Skript "check_env.sql"

Nutzt man Datenbank-Links, die auf die eigene Datenbank zeigen sollen, muss man noch den folgenden Fehler umgehen:
02082, 00000, "a loopback database link must have a connection qualifier"
*// *Cause: An attempt was made to create a database link with the same name*
// as the current database.
*// *Action: a loopback database link needs a trailing qualifier, for example*
// MYDB.EXAMPLE.COM@INST1 - the '@INST1' is the qualifier

Ein weiteres Problem ist, dass man pro Ziel-Datenbank (ohne zusätzliche Qualifier) nur einen Datenbank-Link anlegen kann. Diese Einschränkung kann zu massiven Änderungen in der Applikation führen.

Sobald dann die Datenbank-Links wieder funktionieren, beginnt erst die Arbeit der Entwickler. Diese müssen jegliche Verwendung von DB-Links in der Applikation anpassen:

- SYNONYME
- VIEWS
- MViews
- stored (und wrapped) PL/SQL Code
- stored Java
- stored JavaScript Code (ab Oracle 23c)
- SQL-Statements, die DB-Links nutzen

Sie sehen also, dass diese simple Empfehlung gravierende Folgen haben kann.

In meinem Blog *[2]* gibt es eine ausführlichere Version dieses Artikels, der auch Beispiele enthält, welche Fehler und Probleme in „realen" Umgebungen auftreten können und wie man sie beheben kann.

Fazit/Management Summary

Wenn man, wie viele Security-Scanner empfehlen, einfach den Instance-Parameter global_names auf TRUE setzt und Datenbank-Links verwendet, läuft man Gefahr, dass die über den Link angeschlossenen Datenbanken ebenfalls betroffen sind und Applikationen nicht mehr funktionieren. Eine solche Änderung bedeutet massive Aufwände für DBAs und Entwickler und muss als eigenes Projekt geplant und umgesetzt werden, um den laufenden Betrieb der Applikationen sicherstellen zu können.

Quellen

[1] CIS – Center for Internet Security: Oracle Datenbank 19c Benchmark *https://www.cisecurity.org/benchmark/oracle_database*
[2] *https://www.database-blog.at/2023/10/18/analyse-cis-benchmark-report-empfehlungen-zu-global_names/3*

Über den Autor

Christian Pfundtner ist seit über 30 Jahren im Oracle-Datenbank-Umfeld tätig. Seine Schwerpunkte liegen in den Bereichen Hochverfügbarkeit, Performance Tuning und Security. Er gehört zu den ersten vier Oracle Certified Masters in Europa und hat beginnend mit den Oracle-Datenbank-Zertifizierungen (ab Oracle 7.3) alle Zertifizierungen erlangt. Seit vielen Jahren gehört er auch zu der Oracle ACE Community. Seit 2010 ist er Eigentümer und Geschäftsführer der DB Masters GmbH.

Christian Pfundtner
cp@dbmasters.at

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News Dezember 2023/Januar 2024

DOAG.tv mit Vince Ebert und Björn Bröhl über den Umgang mit KI

Der Vorstandsvorsitzende bat den Wissenschaftskabarettisten vor seiner Keynote auf der DOAG 2023 Konferenz + Ausstellung zum Gespräch.

Interview mit DOAG Vorstand KI Oliver Szymanski

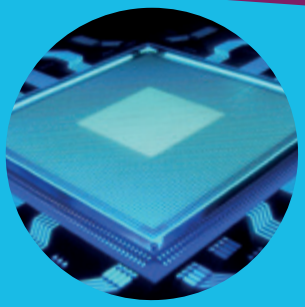
Oliver Szymanski ist seit April 2023 DOAG Vorstand KI und Leiter der neuen KI Community, die im November mit der neuen Konferenz KI Navigator gleich ein Ausrufezeichen gesetzt hat.

Der neue DOAG Vorstand Frank Prechtel im Interview

Im April 2023 wurde Frank Prechtel als neuer DOAG Vorstand Querschnittsgruppen gewählt und hat somit die Nachfolge des langjährigen Vorstands Michael Paege angetreten.

DOAG.tv mit Neil Chandler und Sabine Heimsath über Exadata

"Exadata ist faszinierend", "Exadata hat eine Magie" – und warum dies so ist, wird aus dem auf Englisch geführten "DOAG@Talk" mehr als deutlich.



DOAG Datenbank Kolumne: Oracle Database Appliance X10

Am 20. September 2023 hat Oracle die neue Hardwaregeneration der Oracle Database Appliance (ODA) vorgestellt.

Devs on Tape wird 50: Ein Meilenstein für die Macherinnen und Macher

Mit der 50. Folge feierte der Entwickler-Podcast der DOAG sein erstes großes Jubiläum. Wir gratulieren Carolin Krützmann und Kai Donato.

Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Bernd Weiler
- Rebecca Marburger
- Rudolf Jansen
- Axel Seuthe

Korporative Mitglieder:

- SII Technologies GmbH, Repräsentantin: Agnes Schiele

Termine

April 04

09. - 11.04.2024
JavaLand 2024
Zwei ereignisreiche Konferenztage mit anschließendem Schulungstag rund um das Java-Ökosystem
Nürburgring, Nürburg

22. - 24.04.2024
APEX connect 2024
Konferenz mit zahlreichen Vorträgen und Workshops zu den Themen APEX, JavaScript und PL/SQL
Düsseldorf

25. - 27.04.2024
DOAG Leitungskräfteforum und Delegiertenversammlung
Berlin

Mai 05

15. - 16.05.2024
DOAG 2024 Datenbank mit Exaday
Konferenz rund um die Oracle Database und Engineered Systems
Düsseldorf

Juni 06

18 - 21.06.2024
CloudLand 2024 - Das Cloud Native Festival
Community-Veranstaltung rund um die Themen Cloud- und Container-Technologien, Continuous Delivery, Microservices und DevOps
Brühl (Phantasialand)

Impressum

Red Stack Magazin inkl. Business News wird gemeinsam herausgegeben von den Oracle-Anwendergruppen DOAG Deutsche ORACLE-Anwendergruppe e.V. (Deutschland, Tempelhofer Weg 64, 12347 Berlin, *www.doag.org*), AOUG Austrian Oracle User Group (Österreich, Lassallestraße 7a, 1020 Wien, *www.aoug.at*) und SOUG Swiss Oracle User Group (Schweiz, Dornacherstraße 192, 4053 Basel, *www.soug.ch*).

Red Stack Magazin inkl. Business News ist das User-Magazin rund um die Produkte der Oracle Corp., USA, im Raum Deutschland, Österreich und Schweiz. Es ist unabhängig von Oracle und vertritt weder direkt noch indirekt deren wirtschaftliche Interessen. Vielmehr vertritt es die Interessen der Anwender an den Themen rund um die Oracle-Produkte, fördert den Wissensaustausch zwischen den Lesern und informiert über neue Produkte und Technologien.

Red Stack Magazin inkl. Business News wird verlegt von der DOAG Dienstleistungen GmbH,Tempelhofer Weg 64, 12347 Berlin, Deutschland, gesetzlich vertreten durch den Geschäftsführer Fried Saacke, deren Unternehmensgegenstand Vereinsmanagement, Veranstaltungsorganisation und Publishing ist.

Die DOAG Deutsche ORACLE-Anwendergruppe e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Björn Bröhl. Die DOAG Deutsche ORACLE-Anwendergruppe e.V. informiert kompetent über alle Oracle-Themen, setzt sich für die Interessen der Mitglieder ein und führt einen konstruktiv-kritischen Dialog mit Oracle.

Redaktion:
Sitz: DOAG Dienstleistungen GmbH
(Anschrift s.o.)
ViSdP: Fried Saacke
Redaktionsleitung Red Stack Magazin:
Martin Meyer
Redaktionsleitung Business News:
Marcos López
Kontakt: redaktion@doag.org

Weitere Redakteure (in alphabetischer Reihenfolge): Verena Barth, Andreas Buckenhofer, Niels de Bruijn, Yves Chassein, Ann-Kathrin Denker, Tobias Goerke, Jan Gorkow, Simon Grossmann, Michael Hichwa. Tirthankar Lahiri, Sandra Leist, Johannes Michler. Dr. Gudrun Pabst, Christian Pfundtner, Stefan Raabe, Kurt Rahstorfer, Klaus Reimers, Eva Reil, Christian Ropposch, Katharina Schraft, Michael Schulze, Christian Trieb.

Titel, Gestaltung und Satz:
Diana Tkach
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:
Titel: © usertrmk | *www.freepik.com*
S. 6: © brooke-cagle | *www.unsplash.com*
S. 8: © Yeskay1211 | *www.pixabay.com*
S. 12: © Piro4d | *www.pixabay.com*
S. 18: © WikiImages | *www.pixabay.com*
S. 36: © Hans | *www.pixabay.com*
Titel S. 42: © Vector_Corp | *www.freepik.com*
S. 46 © rawpixel.com | *www.freepik.com*
S. 50: © 2094600 | *www.freepik.com*
S. 56: @ pikisuperstar | *www.freepik.com*
S.62: © Marisa04 | *www.pixabay.com*
S. 68: © Mohammad_usman | *www.pixabay.com*
S. 76: © AbsolutVision | *www.pixabay.com*
S. 86: © Ralphs_Fotos | *www.pixabay.com*
S. 95: © 12019 | *www.pixabay.com*
S. 101: © uniquedesign52 | *www.pixabay.com*
S. 105: © 3974931 | *www.pixabay.com*

Anzeigen:
sponsoring@doag.org

Mediadaten und Preise:
www.doag.org/go/mediadaten

Druck:
WIRmachenDRUCK GmbH,
www.wir-machen-druck.de

Inserentenverzeichnis

DOAG

DOAG Datenbank mit Exaday 2023

ON DEMAND

DATENBANK 2023 VERPASST?

Jetzt On-demand-Ticket buchen und Vortragsaufzeichnungen anschauen!

ALLE ANGEBOTE IM TICKETSHOP

DOAG

DOAG 2024 Datenbank

mit Exaday

15. und 16. Mai in Düsseldorf

datenbank.doag.org