

Red Stack

Magazin

DOAG

SÖUG

swiss oracle
user group

AÖUG
AUSTRIAN ORACLE USER GROUP



APEX

Aus der Praxis

Modernisierung einer
Oracle-Datenbank



Im Interview

Knut Göttling, Chief Platform
Architect, enignum.ai

APEX

Maßgeschneiderte KI-
Modelle mit APEX,
PL/SQL, Rust

CLOUD NATIVE FESTIVAL

im Heide Park Soltau

CloudLand
www.cloudland.org



19. – 22.
MÄI
2026

Neueste Trends
& Innovationen
rund um Cloud Native!

Aufregende
Atmosphäre
eines
Freizeitparks!

Ein Treffen mit den
Hyperscalern AWS,
Microsoft Azure
& Google Cloud!

Spannende Vorträge,
interaktive Workshops &
kreatives Networking!

#CLOUDLAND2026





Maik Becker

DOAG Themenverantwortlicher
APEXAusgabe Nr. 2/2026
auf Abruf!

Liebe Mitglieder, liebe Leserinnen und Leser,

ich begrüße euch ganz herzlich zu unserer APEX-Ausgabe des Red Stack Magazins, die in diesem Jahr kurz vor der APEX connect erscheint.

Nach einem Jahr ohne neues APEX-Release blicken wir mit besonderer Spannung auf die kommende Version 26.1. und selten war die Erwartungshaltung so hoch wie aktuell. Zwar hat uns das APEX-Development-Team in den vergangenen Monaten bereits viele Einblicke in kommende Neuerungen gegeben, dennoch wächst die Ungeduld, die neue Version endlich selbst ausprobieren zu können. Insbesondere mit APEXLang erwarten wir erneut einen Game-Changer, der das Potential hat, die Art und Weise der Entwicklung mit APEX nachhaltig zu verändern.

Inhaltlich knüpft diese Ausgabe direkt an die aktuelle Entwicklung an. Mit Carsten Czarski dürfen wir einen weiteren Blick auf das kommende Release werfen, während Sonja Meyer und Martin Bach zeigen, welche Rolle APEXLang künftig für moderne CI/CD-Ansätze in der APEX-Entwicklung spielen wird. Ergänzt wird dies durch eine Vielzahl interessanter Beiträge rund um Oracle APEX von maßgeschneiderten KI-Modellen über echte Praxisbeispiele bis hin zu Best Practices und Anti-Patterns.

Ein besonderes Highlight dieser Ausgabe ist unser Titelinterview mit Knut Göttling. Als Systemarchitekt gibt er aus der Perspektive des Product Development spannende Einblicke in den Einsatz von APEX in einem hochkomplexen, datengetriebenen Umfeld. Sein Beispiel zeigt eindrucksvoll, welches Potential APEX entfalten kann, wenn es konsequent als Plattform für produktive Systeme gedacht und eingesetzt wird.

Auch abseits von APEX zeigt sich in dieser Ausgabe, wie stark Themen rund um Künstliche Intelligenz weiterhin den Diskurs prägen. Sei es im praktischen Einsatz, in organisatorischen Fragestellungen oder bei neuen technologischen Möglichkeiten.

Ich freue mich sehr über die Vielfalt der Beiträge in dieser Ausgabe und möchte mich herzlich bei allen Autorinnen und Autoren bedanken, die ihr Wissen und ihre Erfahrungen mit uns teilen.

Ich wünsche euch viel Freude beim Lesen, spannende neue Erkenntnisse und viele Impulse für eure tägliche Arbeit. Ich hoffe außerdem, möglichst viele von euch auf den bevorstehenden Konferenzen in Soltau persönlich zu treffen.

Maik Becker

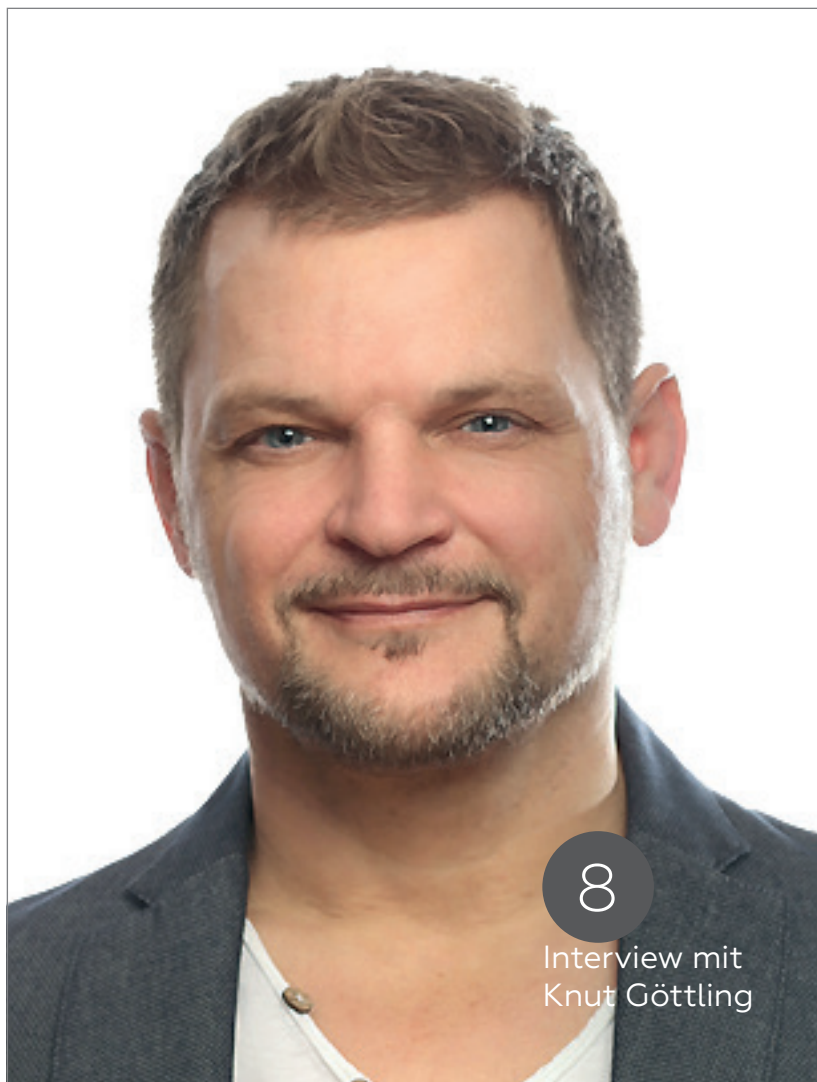
DOAG WEBSESSION

Die DOAG WebSessions* bieten Ihnen in regelmäßigen Abständen spannende Online-Vorträge und -Diskussionen zu einer Vielzahl von Themenbereichen aus den jeweiligen DOAG Communities.

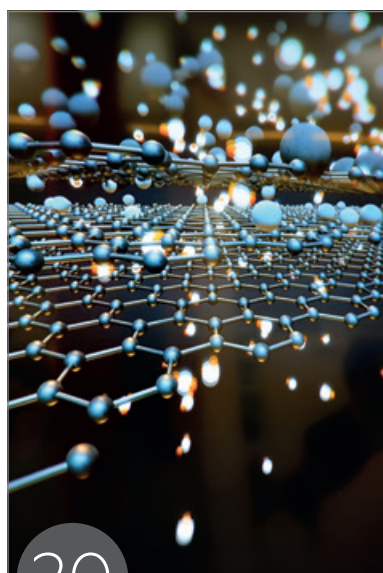
Freuen Sie sich auf WebSessions rund um die Themen Datenbank, Data Analytics und NetSuite oder beteiligen Sie sich bei den DOAG DevTalks an interessanten Gesprächsrunden zu aktuellen Development-Themen!


www.doag.org/go/websessions


*Die Buchung der WebSessions erfolgt ganz einfach über unseren Shop. Mitglieder erhalten im Buchungsprozess automatisch **100 % Rabatt.**



Interview mit
Knut Göttling



Innovative Applikationen
mit Oracle 26ai, APEX
und Graph SQL



Jenseits von ChatGPT:
Maßgeschneiderte
KI-Modelle mit
Oracle APEX, PL/SQL, Rust

Einleitung

- 3 Editorial
- 6 Timeline
- 8 „Wir nutzen KI nicht nur als Marketingbegriff, sondern setzen es als Werkzeug ein“
Interview mit Knut Göttling

APEX

- 12 APEX 26.1. Vollgepackt
Carsten Czarski
- 20 Innovative Applikationen mit Oracle 26ai, APEX und Graph SQL
Dr. Holger Friedrich
- 27 Jenseits von ChatGPT: Maßgeschneiderte KI-Modelle mit Oracle APEX, PL/SQL, Rust
Andreas Maier
- 33 APEX im Git, Klartext im Code – CI/CD mit APEX und ein Hauch Intelligenz
Sonja Meyer, Martin Bach
- 38 Nicht nachmachen! Anti-Patterns in PL/SQL und APEX
Andreas Wismann
- 45 Ende-zu-Ende-Digitalisierung mit APEX-Fachanwendungen im Backend
Benjamin Nebgen
- 48 APEX-Plug-ins – Von den Anfängen bis zu KI
Ronny Weiß
- 56 Wie kommt mein GPS-Track nach APEX?
Wilfried Eigl
- 62 Flows for APEX – BPMN-Prozessmodellierung meets Low-Code
Richard Allen

Datenbank

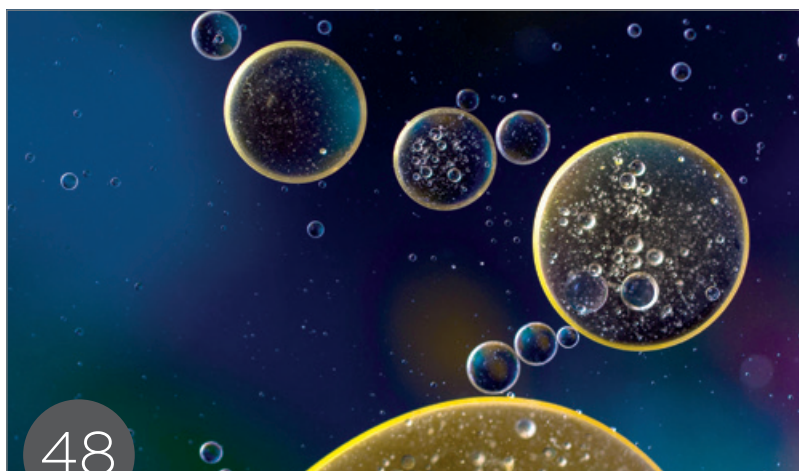
- 66 Modernisierung einer Datenbank
Matthias Jakob
- 72 Pluggable Database Snapshots von der Stange
Henner Hucke

KI

- 77 AI-First-Organisationen:
Der Praxisleitfaden
Dr. Benjamin Linnik
- 83 Private KI mit Application Express
verbinden
Marcel Boermann-Pfeifer

Development

- 93 Mit SQLcl und GitLab-Pipelines den
CI/CD-Prozess optimieren
Maurice Wilhelm



APEX-Plug-ins – Von
den Anfängen bis zu KI



Wie kommt mein GPS-Track
nach APEX?



AI-First-Organisationen:
Der Praxisleitfaden



Private KI mit Application
Express verbinden



Pluggable Database Snapshots
von der Stange

News

- 71 Oracle Datenbanken Monthly News
- 76 DOAG Datenbank Kolumne

Intern

- 52 DOAG Kalender
- 100 Best of DOAG Online
- 101 Neue Mitglieder + Termine
- 102 Impressum + Inserenten

— TIMELINE —

13. Februar 2026

Die DB WebSession „101 Data Governance: Vom Regelwerk zur Befähigung“ mit Matthias Jung findet statt. Es moderiert Christian Trieb.

19. Februar 2026

Der DOAG DevTalk „ORDS als Schnittstelle“ mit Robert Marz und Oliver Lemm steht auf der Agenda.

25. Februar 2026

Im DOAG DBTalk geht es diesmal um das Thema „PostgreSQL hochverfügbar betreiben“. Mit dabei sind Borys Neselevskyi und Raphael Salguero.

5. März 2026

Mike Dietrich und Sabine Heimsath laden ein zum DOAG DevTalk „Upgrade nach Oracle AI Database 26ai – und welche Features sollte man unbedingt ausprobieren?“

10. bis 12. März 2026

Die JavaLand 2026 lädt vom 10. bis 12. März erstmals in Deutschlands größten und meistbesuchten Freizeitpark, den Europa-Park in Rust, ein. Die Besucherinnen und Besucher genießen ein Vortragsprogramm mit fast 130 Vorträgen zu 13 verschiedenen Themenbereichen, Community-Aktivitäten, Workshops, eine Ausstellung und den beliebten OpenPark-Abendevent mit spektakulären Attraktionen.

12. und 13. März 2026

Das neue Event DevLand findet erstmals statt. Unsere digitale Erlebniswelt verbindet Fachwissen, Hands-on-Erfahrungen und Community-Spirit mit innovativen Formaten in einem einzigartigen Setting aus fünf Themenwelten. Im Europa-Park in Rust treffen sich Developer, Architektinnen und Architekten, Entscheiderinnen und Entscheider sowie Vordenkerinnen und Vordenker, um nicht nur über Code und Software zu sprechen, sondern über KI, Architektur, Cloud, Security, Daten, Organisation und die Frage, wie wir die Zukunft der Technologie gestalten.

13. März 2026

In der DB WebSession mit Christian Pfundtner erfahren die Teilnehmenden alles zum Thema „Oracle Datenbank Lizenzierung SE2 versus EE, On-Premise versus Cloud, Pitfalls“.

19. März 2026

Carsten Czarski und Florian Graßhoff laden zum DOAG DevTalk. Es geht um das Thema „APEX-Upgrade“.

25. März 2026

Zum Regionaltreffen München/Südbayern trifft sich die Community in München. Die Vorträge „Nutzung von Oracle GenAI Agent und APEX für einen eigenen Chatbot“ mit Referentin Dr. Gudrun Pabst und „Transaktionen neu gedacht: Features ab 23ai effektiv in APEX nutzen“ mit David Dürer stehen auf der Agenda. Es moderiert Andreas Ströbel.





25. März 2026

Zum DOAG DBTalk „Oracle Datenbank Audit – Nutzen versus Aufwand?“ laden Christian Pfundtner und Bruno Cirone.

25. März 2026

Das Regionaltreffen Osnabrück/Bielefeld/Münster mit Andreas Kother findet in Paderborn statt. Matthias Jung spricht in seinem Vortrag über „Die verborgene Grammatik der Daten – eine Bilderreise durch Governance, Literacy und Culture“ und Markus Klenke über „Oracle-Forms- Modernisierung – Warum Abwarten teuer wird“.

2. April 2026

Der DOAG DevTalk mit Matthias Schulz und Christian Schwitalla widmet sich dem Thema „Modernes SQL – MODEL Klausel“.

10. April 2026

Unter dem Titel „Oracle auf Windows – immer gut für Überraschungen“ behandelt Elke Fritsch in einer DB WebSession eine Auswahl von Problemfällen und ihre Lösungen oder Workarounds.

10. und 11. April 2026

Das DOAG Führungskräfteforum und die DOAG Delegiertenversammlung finden in Berlin statt.

16. April 2026

Der DOAG DevTalk „SQLcl“ mit Referent Moritz Klein und Moderator Oliver Lemm steht an.

30. April 2026

Zum DOAG DevTalk „CI & CD mit SQLcl Projects“ lädt Oliver Lemm die Referentin Patrizia Regenbergl ein.

4. Mai 2026

Ein weiteres Regionaltreffen München/Südbayern bietet in München mit „APEX-Bordmittel nutzen, außer ...“ von Steffen Clauß und „Low Code mit APEX 2026 – Der Stand der Dinge“ von Carsten Czarski zwei Vorträge. Es moderiert Andreas Ströbel.

8. Mai 2026

Christian Trieb lädt Christian Pfundtner ein, in der DB WebSession über das Thema „Datenbank Partitioning by Example“ zu sprechen.



„Wir nutzen KI nicht nur als Marketingbegriff, sondern setzen es als Werkzeug ein.“

Maik Becker unterhielt sich mit Knut Göttling, Chief Platform Architect bei enigum.ai, über die Plattform APEX im Zusammenhang mit der Plattform Enigum AI, Geodaten, Künstliche Intelligenz im Entwicklungsalltag, den Einsatz von Komponenten und Funktionen aus dem Oracle-Stack, Trends in der APEX- und der Softwareentwicklung sowie die Bedeutung der Oracle- und APEX-Community und der DOAG.

Hallo Knut, stell dich doch bitte einmal unseren Lesern vor: Wie war dein bisheriger Werdegang und womit beschäftigst du dich heute?

Moin, mein Name ist Knut Göttling. Ich habe bis 1997 Elektronik an der FH Furtwangen studiert und bin im Zuge meiner Arbeiten immer mehr von der Hardwareentwicklung zur Softwareentwicklung gerutscht. Dabei ging es immer um eine zentrale Datenbank

und ein Backend in einer beliebigen Sprache mit mehr oder weniger Frontend davor.

Nach meinem Umzug nach Hamburg 2001 habe ich mich vollständig auf die Entwicklung mit PL/SQL in Oracle-Datenbanken konzentriert. Besonders prägend waren rund zehn Jahre bei einer Hamburger Privatbank, bei der die komplette Geschäftslogik in der Datenbank umgesetzt war. In dieser Zeit habe ich ein sehr tiefes Verständnis für datenbank-

zentrierte Architekturen entwickelt, weshalb auch in den folgenden Jahren die Datenbank meist der zentrale Bestandteil meiner Arbeit blieb. 2018 habe ich bei Kaiserwetter Energy Asset Management die Leitung der Softwareentwicklung für die Plattform „Aristoteles“ übernommen — ein System zur Analyse und Optimierung von Wind- und Solarparks auf Basis von SAP HANA. Dort ging es um sehr große Datenmengen und Echtzeit-Auswertungen.

Seit 2022 arbeite ich als Systemarchitekt an der Plattform „Enignum“, einer KI-basierten Lösung zur Stärkung der Resilienz von Energiesystemen.

Wie hat sich dein Weg in Richtung Oracle-Datenbank entwickelt und wie kam es schließlich dazu, dass du dich intensiver mit APEX beschäftigst?

Während meiner Zeit bei der Berenberg Bank wurde ausschließlich Oracle eingesetzt, die Business-Logik lag komplett in PL/SQL. APEX war zwar immer wieder Thema, wurde aber damals nicht eingesetzt. Unter anderem, weil es noch nicht die heutigen Möglichkeiten hatte. Auch in späteren Projekten spielte APEX zunächst keine Rolle, da ein Umstieg oft zu aufwendig gewesen wäre. Entsprechend lag mein Fokus weiterhin auf SQL und PL/SQL.

Erst als ich selbst für Architektur und Technologieauswahl verantwortlich war und freie Hand hatte, habe ich mich intensiver mit Alternativen beschäftigt. Ursprünglich plante ich ein klassisches Setup mit separatem Frontend (zum Beispiel React oder Vue). Als ich Aufwand, Kosten und Zeitrahmen genauer betrachtet habe und ein konkreter Go-Live-Termin näher rückte, habe ich mir APEX noch einmal genauer angesehen.

Am Ende war entscheidend, dass APEX viele benötigte Funktionen bereits mitbringt, die Datenanbindung sehr einfach ist und ich deutlich schneller ans Ziel komme. Nach dem Hinweis an meinen CEO, dass ich zwar nicht jedes Pixel kontrollieren kann, aber dafür massiv an Geschwindigkeit gewinne, war die Entscheidung schnell getroffen.

In welchem Kontext setzt du APEX aktuell ein und welche Anforderungen stellt eure Anwendung dabei an die Plattform?

Wir setzen APEX ausschließlich für unsere Plattform Enignum ein. Da wir sehr stark auf Styling achten und dafür eigene Designer beschäftigen, war mir die Möglichkeit des Einsatzes von CSS sehr wichtig. Das lässt sich mit APEX gut umsetzen.

Unser Frontend ist zur Darstellung und Verwaltung von Geo-Objekten gedacht. Im ersten Ansatz habe ich dazu die Map-Region eingesetzt. Die Map-Region ist leider recht langsam und hat mit der Darstellung von sehr vielen Geo-Objekten natürlich ihre liebe Mühe, weshalb wir für die Zukunft über den Einsatz von MapLibre nachdenken. Man muss dabei auch sehen, dass die Map-Region nie für solche Datenmengen gebaut wurde. Carsten Czarski (Architect im Oracle APEX Development Team) nennt mich auch deshalb immer wieder den „Load-Tester“ der Map-Region.

Zur Darstellung von Kennzahlen, Reports und Geschäftsdaten ist APEX natürlich prädestiniert und sehr einfach anzuwenden.

Kannst du unseren Lesern erläutern, worum es bei der Plattform Enignum AI fachlich geht und welchen Nutzen die Anwendung bietet?

Die Plattform Enignum AI aggregiert Geo-Informationen aus vielen verschiedenen, öffentlichen wie auch nicht öffentlichen Quellen. Wir reden hierbei über rund 180 Millionen Objekte. Es handelt sich dabei zum Beispiel um Ladestationen, Kreis-, Länder- und Staatsgrenzen, Grundstücke, Häuserumrisse, Übertragungs- und Verteilnetze, Netzanschluss- und Übergabepunkte, Umspannstationen, erneuerbare und konventionelle Energieanlagen sowie Speicher und viele andere Dinge, die in diesem Zusammenhang eine Rolle spielen und die zur Analyse und Bewertung der Resilienz von Energiesystemen für Unternehmen, Städte, Regionen oder ganzer Länder herangezogen werden.

Diese Daten werden auf Karten visualisiert und mit Verbrauchs- und Prognosedaten kombiniert. Ziel ist es, Aussagen über Versorgungssicherheit und Resilienz von Energiesystemen für Unternehmen, Städte oder ganze Regionen zu ermöglichen.

Ein wesentlicher Bestandteil ist der Einsatz von KI, die etwa zur Analyse und Erkennung von Infrastruktur eingesetzt wird, für deren Informationen wir nicht auf öffentliche Schnittstellen zurückgreifen können.

Welche konkreten Faktoren waren für dich ausschlaggebend bei der Entscheidung für Oracle APEX und welche Eigenschaften sind dir dabei besonders wichtig?

Bei APEX waren für mich genau zwei Dinge extrem wichtig, um nicht zu sagen entscheidend. Zuerst natürlich die Entwicklungsgeschwindigkeit. Viele notwendige Komponenten wie beispielsweise Authentifizierung sind bereits integriert und lassen sich gut skalieren. Dadurch reduziert sich der Gesamtaufwand erheblich.

Als zweites ist für mich die Nähe zu den Daten wichtig. So kann ich als datenzentrischer Entwickler Informationen sehr schnell sichtbar machen, ohne auf zusätzliche Frontend-Spezialisten für React, Vue oder andere Technologien angewiesen zu sein.

Außerdem ist natürlich noch zu nennen, dass sich eine neue Version des Frontends sehr einfach und schnell in Produktion ausrollen lässt.

Welche genaue Rolle spielen Geodaten in Enignum und warum sind sie für eure Architektur so zentral?

Geodaten sind in Enignum kein Nebenaspekt, sondern das tragende Strukturprinzip des gesamten Systems. Sie stellen ein sehr starkes Ordnungsprinzip dar.

„Alles befindet sich irgendwo und hat Nachbarn!“

In Enignum fungieren Geodaten als Integrationsschicht für Daten aus verschiedenen Datenquellen, die semantisch völlig unterschiedlich sind. Wir schaffen in Enignum hierfür einen gemeinsamen Kontext. Zum Beispiel: Eine Leitung verläuft durch ein bestimmtes Gebiet, eine PV-Anlage steht auf einem bestimmten Gebäude, ein Windpark liegt in einer bestimmten Genehmigungszone, ein Umspannwerk versorgt eine bestimmte Region und so weiter.

Gerade im Energiesektor schaffen Geodaten Entscheidungsfähigkeit, weil Informationen über Machbarkeit und Wirtschaftlichkeit vorliegen. Wir können dadurch Fragen beantworten wie beispielsweise:

- Welche Einspeiser liegen im Einzugsgebiet einer Leitung?
- Welche Netzausbaumaßnahmen betreffen welche Kommunen?
- Wo entstehen Engpässe?
- Welche Flächen sind regulatorisch zulässig?

Ohne Geodaten wären das alles isolierte Tabellen und Informationen. Geodaten helfen also, daraus entscheidungsrelevante Zusammenhänge zu schaffen.

Welche weiteren Komponenten und Funktionen aus dem Oracle-Stack nutzt du in eurer Anwendung neben APEX?

Da wir mit der Version 26ai arbeiten, haben wir natürlich die Möglichkeit, viele Werkzeuge aus der Datenbank einzusetzen, weil es sich mittlerweile um eine Converged Database handelt, wie es von Oracle selbst benannt wird.

Zur Vermessung und Analyse der Geo-Informationen setzen wir die Spatial-Packages von Oracle ein. Das heißt, die Berechnung von Flächen, Längen, das Zusammenführen von Arealen zu einem Objekt, die Ermittlung des Abstandes zum Nachbarn und die Berechnung von geografischen Schwerpunkten lösen wir über Spatial-Funktionalität, die wiederum kostenfrei in der 26ai zur Verfügung steht.

Der Import der großen Datenmengen wird bei uns über die JSON- und XML-Tables erledigt, da sie so sehr einfach aus ihren Transportformaten in Tabellenformate umgewandelt werden können.

Parallel-DML ist vor allem beim Import der oftmals Millionen Datensätze wichtig, da wir sonst keine Möglichkeit hätten, alles tages- beziehungsweise monatsaktuell zu halten.

Materialized Views, Function Based Indexes, JSON-Duality-Views und viele andere Features aus dem Oracle-Werkzeugkasten kommen bei uns natürlich auch zum Einsatz. Über letzteres realisieren wir alle notwendigen REST-Schnittstellen zu angrenzenden Systemen.

Die Infrastruktur betreiben wir vollständig in OCI, inklusive Netzwerk, Load Balancing, Security und Dynamic Routing Gateways. Robert Melzer vom Oracle Support war mir dabei ein sehr guter Lehrer und eine große Hilfe. Danke dafür, Robert!

Natürlich verwenden wir auch Buckets zur Speicherung und für den Im- und Export von großen Datendateien. Die Anbindung und der Zugang dazu aus der Datenbank heraus waren auch erstaunlich einfach. Auch hier werden wieder die PL/SQL-Packages von Oracle kostenfrei mitgeliefert.

Für welche Größenordnung an Nutzern ist die Anwendung ausgelegt und in welchem Modell wird sie betrieben?

Ursprünglich war Enignum AI als Freemium Modell geplant. Aufgrund der Sensibilität der Daten in Bezug auf kritische Infrastruktur wird die Plattform jedoch als SaaS-Lösung betrieben. Die Benutzerzahlen werden dann bei ein paar Hundert bis Tausend Anwendern liegen.

Wie setzt ihr Künstliche Intelligenz konkret in eurer Anwendung und in eurem Entwicklungsalltag ein?

Wir nutzen KI nicht nur als Marketingbegriff, sondern setzen es als Werkzeug ein. Sie hilft uns dort, wo klassische, regelbasierte Logik an ihre Grenzen stößt. Etwa bei Mustererkennung oder unscharfen Datenlagen. Als Beispiel kann man das Clustering nennen. In welcher Region gibt es besonders viele erneuerbare Energien? Wie entwickelt sich die Einspeiseleistung einer Region? Welche Ausbauggebiete zeigen eine überdurchschnittliche Dynamik? Wie kann ich die elektrische Infrastruktur eines Gebiets ermitteln, wenn es dafür keine öffentlich verfügbaren Informationen gibt? KI hilft uns, diese Fragen zu beantworten.

Zusätzlich ist KI mittlerweile auch ein extrem guter und ideenreicher Assistent bei der Entwicklung, die ich täglich einsetze. Kleine Code-Snippets kann sie sehr schnell und oftmals sehr gut erstellen. Und das ohne viel Suchen und „Murren und Knurren“.

Welche Rolle spielen für dich die Oracle- und APEX-Community sowie die DOAG in deinem beruflichen Alltag?

Eine sehr wichtige. Durch Konferenzen wie die APEX connect, die DOAG Konferenz und Ausstellung in Nürnberg, die Cloud World in Las Vegas oder auch die DOAG-Regionalsessions bekomme ich nicht nur Einblicke in neue Entwicklungen, sondern vor allem direkten Austausch mit Menschen, die täglich mit denselben Technologien arbeiten. Viele Fragen lassen sich dort schneller und fundierter klären als über jede Dokumentation.

Für mich ist die Community kein reiner Informationskanal, sondern ein fachliches Netzwerk. Gerade bei komplexen Themen hilft es enorm, unterschiedliche Sichtweisen zu hören und Lösungen gemeinsam zu diskutieren. Viele Ideen für Enignum sind im Gespräch entstanden, nicht isoliert am Schreibtisch.

Als Oracle ACE sehe ich mich dabei nicht nur als Teilnehmer, sondern auch als aktiven Teil dieser Community. Wissen weiterzugeben, Vorträge zu halten und Erfahrungen zu teilen gehört für mich selbstverständlich dazu. Der Austausch funktioniert in beide Richtungen – man lernt genauso viel, wie man selbst einbringt.

Die DOAG hat für mich eine besondere Bedeutung, weil sie Praxis, Architektur, Strategie und Produktentwicklung zusammenbringt. Man bekommt technische Tiefe, ehrliches Feedback und einen guten Eindruck davon, wohin sich die Oracle-Welt entwickelt.

Ohne die Community wäre meine fachliche Entwicklung deutlich langsamer verlaufen. Sie hilft mir, auf dem aktuellen Stand zu bleiben, Lösungen besser zu machen und neue Ideen schneller umzusetzen.

Welche Trends beobachtest du aktuell bei Oracle APEX und in der Softwareentwicklung insgesamt – und wohin geht aus deiner Sicht die Entwicklung?

Ich sehe bei Oracle APEX und in der Softwareentwicklung insgesamt einen klaren Trend zu höherer Produktivität bei gleichzeitig steigender Komplexität im Hintergrund. Low-Code-Plattformen wie APEX werden dabei nicht weniger technisch, sondern ermöglichen es, sich stärker auf Geschäftslogik und Datenmo-

delle zu konzentrieren statt auf Boilerplate-Code. Gleichzeitig werden Anwendungen modularer und API-getriebener, sodass APEX häufig als Integrations- und Frontend-Schicht in einer größeren Architektur dient.

Ein weiterer zentraler Trend ist die stärkere Verzahnung mit Cloud-Infrastrukturen. Themen wie Autonomous Database, skalierbare Services und DevOps-Pipelines sind heute Standard und kein Spezialfall mehr. Parallel dazu wird KI zunehmend zum Entwicklungswerkzeug – etwa bei Code-Generierung, Analyse oder intelligenten Funktionen innerhalb von Anwendungen.

Ich erwarte, dass datengetriebene Anwendungen weiter an Bedeutung gewinnen und Features wie JSON, Spatial oder Graph noch stärker genutzt werden. APEX profitiert davon, weil es sehr nah an der Datenbank arbeitet und komplexe Datenmodelle direkt nutzbar macht. Insgesamt sehe ich die Zukunft in modularen, cloud-nativen und KI-unterstützten Systemen, bei denen Geschwindigkeit und fachliche Qualität gleichermaßen wichtig sind.

Letzte Frage: Wo siehst du aktuell noch Verbesserungspotenzial bei APEX und an welchen Stellen sollte Oracle aus deiner Sicht nachschärfen?

Verbesserungspotenzial sehe ich vor allem im Bereich Lifecycle Management und DevOps-Integration. Export und Import funktionieren zwar, aber Themen wie CI/CD und Versionskontrolle könnten noch stärker integriert werden.

Für professionelle Teams ist das inzwischen Standardanforderung. Ich glaube, gehört zu haben, dass das APEX-Team bei Oracle bereits an diesem Thema arbeitet.

Auch die KI-Integration direkt in die Plattform bietet großes Potential – etwa intelligente Assistenten für Datenmodellierung, Performance-Analyse oder UI-Optimierung. Vieles ist bereits möglich, aber noch nicht systematisch eingebettet.

Beim Debugging von Seiten ist mir aufgefallen, dass die Logs in größeren Anwendungen nicht immer einfach auszuwerten sind. Hier würde ich mir eine Gruppierung in der Darstellung wünschen, um effizienter in der Fehleranalyse zu werden.

Ich persönlich würde mir noch wünschen, dass die Map-Region eine Schwester bekommt, die zur performanten Anzeige großer Datenmengen geeignet ist. Die Map-Region kann pro-

blemlos ein paar Hundert Objekte darstellen. Sobald sie aber ein paar Tausend bis Millionen Objekte anzeigen sollen, ist sie nicht mehr benutzbar. Da hilft dann nur noch die Verwendung von Vector Tiles. Karin Patenge von Oracle, die mir bei meinen Fragen zu Oracle Spatial sehr weitergeholfen hat, hat dazu einen sehr guten und lesenswerten Post (<https://medium.com/@kpatenge/how-to-create-vector-tiles-from-spatial-data-managed-in-the-oracle-database-db662f2b5544>) geschrieben. Vielleicht könnte noch ein Knöpfchen integriert werden, das es erlaubt, eine Karte ganzseitig darzustellen.

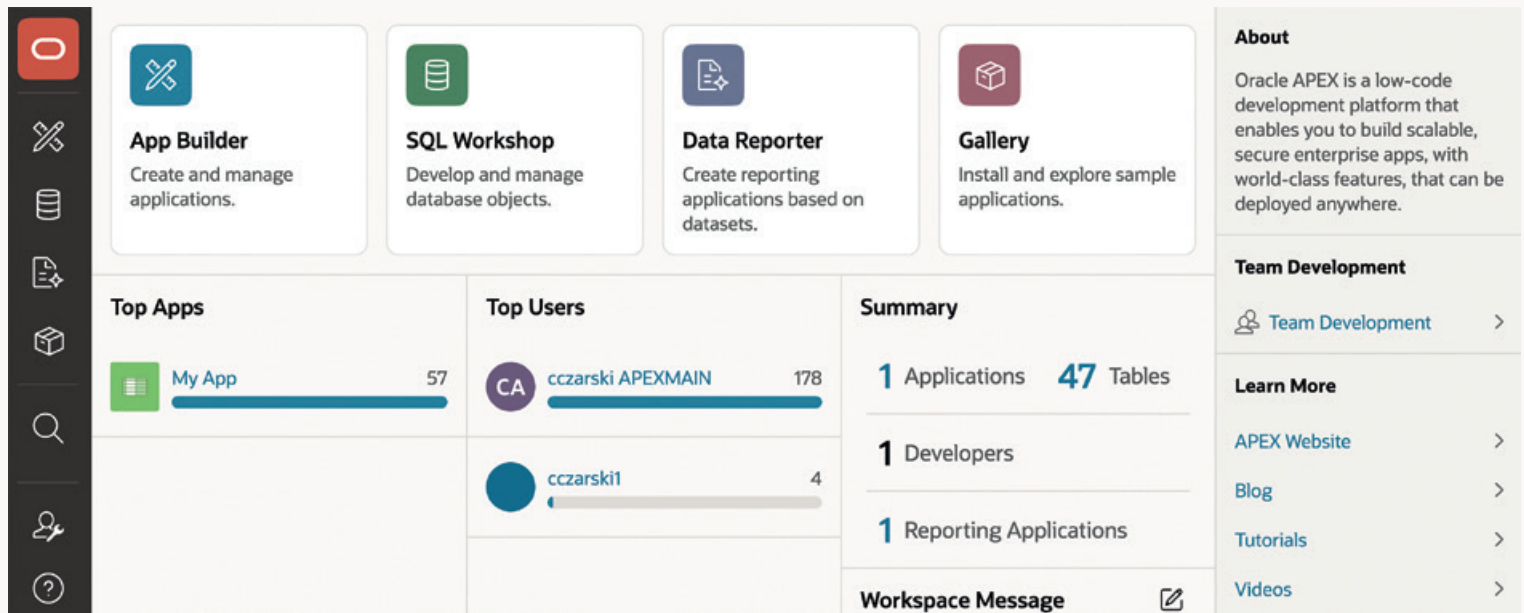
Schlussendlich muss ich noch sagen, dass die Entwicklung der Plattform Enignum AI als One-Man-Show ohne die Verwendung von APEX in einer so kurzen Zeit nicht mal ansatzweise möglich gewesen wäre. Hier zeigen sich deutlich die Vorteile bei der Verwendung eines Low-Code Frameworks.

Insgesamt bin ich in den letzten Jahren zu einem überzeugten Anhänger von APEX geworden. Danke an das gesamte APEX-Team bei Oracle, speziell an „APEX-Carsten“ und Christian Rokkita sowie alle anderen hilfsbereiten Menschen in der Community. Ich bin wirklich sehr beeindruckt von der tollen Unterstützung.



KNUT GÖTTLING

Knut Göttling ist Chief Platform Architect bei enignum.ai und arbeitet seit 2022 an der Plattform „Enignum“, einer KI-basierten Lösung zur Stärkung der Resilienz von Energiesystemen. Er verfügt über langjährige Erfahrung in der Entwicklung skalierbarer Anwendungen mit Fokus auf Oracle-Datenbanken und Geodatenverarbeitung. Sein Schwerpunkt liegt auf der effizienten Verarbeitung und Analyse großer Datenmengen, insbesondere im Umfeld von Geodaten und Energiedaten. Dabei verbindet er tiefes Datenbank-Know-how mit modernen Ansätzen zur Performance-Optimierung und Parallelisierung.



APEX 26.1. Vollgepackt

Carsten Czarski, Oracle Global Services

Nach etwas längerer Entwicklungszeit steht die neue APEX-Version 26.1 vor der Tür. Im Mittelpunkt sind die schon auf der DOAG-Konferenz 2025 vorgestellte Domain Specific Language **APEXlang** und das omni-präsente Thema Künstliche Intelligenz (KI). Aber APEX 26.1 enthält jede Menge neuer Features darüber hinaus: in nahezu allen Bereichen finden sich viele Verbesserungen im Detail. Dieser Artikel gibt einen Überblick über die wichtigsten Neuerungen.

Sprechen Sie ... APEXlang?

APEX 26.1 führt die lange erwartete Domain Specific Language (DSL) **APEXlang** ein. APEXlang ist eine quelloffene Sprache zur Beschreibung von APEX-Anwendungen. Wenn man eine APEX-Anwendung exportiert, wird es künftig die Wahl zwischen dem traditionellen SQL- und dem neuen APEXlang-Format geben (siehe Abbildung 1).

APEXlang beseitigt die bekannten Unzulänglichkeiten des SQL-Formats:

- APEXlang ist offen sowie dokumentiert und das Editieren einer APEXlang-Datei mit einem Code-Editor ist, im Gegensatz zum SQL-Format, *ausdrücklich* unterstützt. *Abbildung 2* zeigt APEXlang in VSCode.

- APEXlang-Dateien enthalten keine „*internal IDs*“, welche beim Versionieren der SQL-Exportdateien immer wieder Probleme bereiteten.
- APEXlang-Dateien sind wesentlich besser lesbar: Attribut-Namen entsprechen den Namen im Page Designer oder auf den Shared-Components-Seiten im APEX Application Builder – und nicht mehr den internen Tabellenspalten der APEX-Engine.

Die **SQL Developer Extension für VS-Code** wird APEXlang aus dem Stand unterstützen. Das beinhaltet nicht nur einfaches Syntax Highlighting, sondern auch *Code Diagnostics*, also das Markieren falscher Namen oder Werte. Darüber hinaus können APEX-Anwendungen direkt in die Datenbank

importiert oder aus dieser exportiert werden.

Das Importieren einer Anwendung im APEXlang-Format wird mit SQLcl, VS-Code und (später) auch mit dem APEX Application Builder möglich sein. *Listing 1* zeigt, wie ein APEXlang-Import mit SQLcl aussieht.

APEXlang-Dateien werden sich ganz normal in Git, SVN oder andere Code-Repositorys einchecken lassen. *Diff* und *Merge* wird, im Vergleich zum traditionellen SQL-Format, wesentlich einfacher sein, allein schon, weil all die internen IDs wegfallen.

Das Wegfallen der internen IDs hat allerdings eine Konsequenz: APEX-Komponenten referenzieren sich untereinander, beispielsweise referenziert ein Page-Prozess einen Button, oder eine Region re-

ferenziert ein Template. Dies erfolgt nun mit lesbaren *Static IDs* (für viele Komponenten gibt es diese schon länger, wenn es möglich ist, die Komponente mit einem PL/SQL-Package zu nutzen). Mit APEXlang bekommt nahezu jede Komponente eine **Static ID** (siehe *Abbildung 3*).

Wichtig ist, dass diese neue Static ID dem Referenzieren der Komponente (in *Abbildung 3* dem Button) in APEXlang-Datenteilen dient. Beim Erstellen einer Komponente wird APEX immer eine gut lesbare Static ID generieren, so dass man hier keine Mehrarbeit hat. Möchte man die Static ID selbst setzen, so sollte Wert auf eine „sprechende“ ID gelegt und diese danach am besten nicht mehr geändert werden (daher das kleine Schloss rechts neben dem Eingabefeld, was unbeabsichtigtes Ändern verhindert).

Davon zu unterscheiden ist die **HTML DOM ID**, welche das „id“ Attribut im HTML-DOM festlegt. In früheren APEX-Versionen wurde *dieses* Attribut **Static ID** genannt, so dass man sich als APEX-Entwickler hier ein wenig umgewöhnen muss.

Gestatten? Mein Name ist Agent, APEX AI Agent.

Unterstützung für Künstliche Intelligenz (KI) wurde erstmals mit APEX 24.1 eingeführt: Seitdem können **Generative AI Services** in den **Workspace Utilities** konfiguriert und im Application Builder, in Dynamic Actions oder mit dem **APEX AI PL/SQL Package** verwendet werden. Mit APEX 24.2 wurden **AI Configurations** und **RAG Sources** eingeführt, welche das

Prompt-Enrichment in einer APEX-Anwendung einfacher machen.

In APEX 26.1 werden AI Configurations stark erweitert und in **AI Agents** umbenannt. Die wichtigste Neuerung ist die Möglichkeit, in einem AI-Agent deklarative *Tools* zu hinterlegen. Wenn eine APEX-Anwendung KI nutzt und eine Anfrage an ein LLM stellt, dann kann das LLM quasi eine „Rückfrage“ halten. *Abbildung 4* zeigt, wie APEX und der AI-Service beim Einsatz von Tools zusammenspielen.

Der große Wert von Tools ist, dass Entwickler nicht mehr alle denkbaren Informationen in den initialen Prompt aufnehmen müssen. Dieser kann kleiner gehalten werden – zusätzliche Informationen erhält der AI-Service bei Bedarf.

Abbildung 5 zeigt einen KI-Agenten in APEX, mit einer Liste von konfigurierten Tools. **Augment System Prompt** Tools sind die „alten“ RAG Sources, werden also schon bei der initialen Anfrage verwendet, um den Prompt anzureichern. **On Demand** Tools werden dagegen nur verwendet, wenn der AI-Service konkret danach fragt.

Abbildung 6 zeigt ein konkretes Tool: Die SQL-Abfrage gibt zurück, welche APEX-Anwendungen im Workspace existieren. Anhand der Beschreibung wird der AI-Service dieses Tool anfragen, sobald sich der Prompt auf die im Workspace vorhandenen Anwendungen bezieht oder diese Information benötigt wird.

Dank der Unterstützung für Tools können APEX-Anwendungen mit APEX 26.1 wirklich *Agentic AI* anbieten. Das LLM kann damit mehr als einfach nur eine Frage beantworten: Je nachdem, wie das Tool pro-

grammiert ist, kann der AI-Service auch konkrete Funktionen in der Anwendung übernehmen. In einer Helpdesk-Anwendung könnte ein Tool beispielsweise automatisch einen „Service Request“ anlegen, also eine Zeile in eine Tabelle eintragen, oder einen APEX-Workflow starten.

Als Entwickler eines Tools sollte man stets im Auge behalten, dass der *AI-Service* das Tool konkret aufruft, und auch die Werte etwaiger Parameter liefert. Die Tool-Implementierung sollte wasserdichte Validierungen enthalten – schließlich kann man als Entwickler niemals darauf vertrauen, dass die vom LLM erhaltenen Parameter „schon richtig“ sein werden.

Interactive Reports: Talk to Your Data.

Neue KI-Unterstützung findet sich auch im Interactive Report. *Abbildung 7* zeigt, wie man sich mit einem APEX 26.1 Interactive Report „in natürlicher Sprache“ unterhalten kann.

Das Besondere an dieser KI-Unterstützung im Interactive Report ist, dass APEX die konkreten Daten nicht an den AI-Service sendet. Der AI-Service bekommt nur Metadaten über den Interactive Report, also Spaltennamen, Datentypen und im Page Designer zusätzlich konfigurierbare Kontext-Informationen (siehe *Abbildung 8*).

Als Antwort sendet die KI eine neue Report-Konfiguration, also das Hinzufügen eines Filters, Highlights, einer Gruppierung oder Sortierung. Die eigentliche Antwort auf die Frage wird also weiterhin vom Interactive Report, und nicht von der KI, gegeben.

Data Reporter

Mit dem neuen **Data Reporter** erlaubt APEX auch dem „Nicht-APEX-Entwickler“ einfachen Zugang zu Tabellen und Views im Datenbank-Schema.

Ein paar neue Konzepte sind wichtig für den Umgang mit Data Reporter.

- Ein **Dataset** ist eine benannte Zusammenstellung von Tabellen, Views oder SQL-Abfragen.
- Eine Data-Reporter-**Anwendung** stellt Berichte, basierend auf einem oder

```
SQL> apex import -input /path/to/folder-with-apx-files/ -id 471101

Importing application ID: 471101 into workspace: CCZARSKI
Import successful.
API Last Extended:20260330
Your Current Version: 20260330
This import is compatible with version: 20260330
COMPATIBLE (You should be able to run this import without issues.)
ID offset during import: 0
New ID offset for application: 0
... elapsed: 7.81 sec

SQL>
```

Listing 1: Import von APEXlang-Dateien mit SQLcl

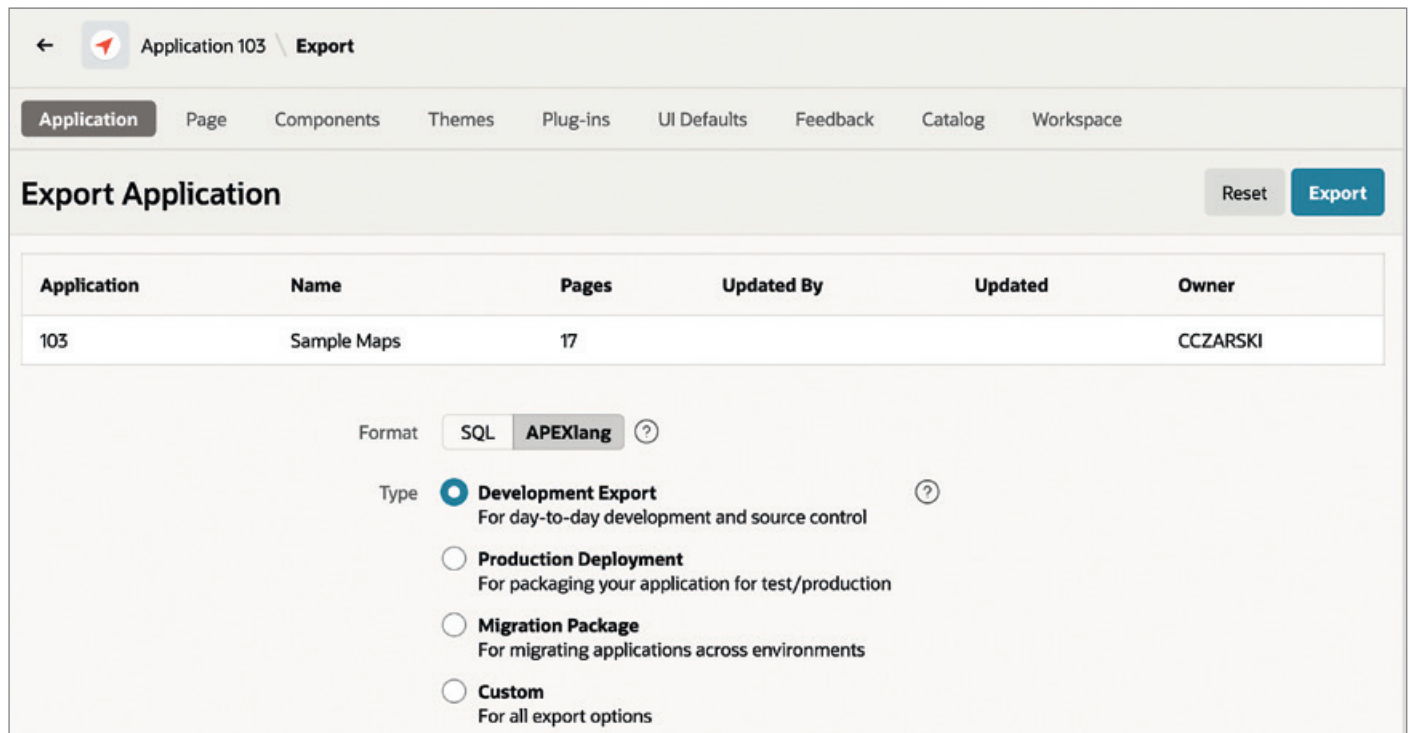


Abbildung 1: Export einer APEX-Anwendung im neuen APEXlang-Format (Quelle: Carsten Czarski)

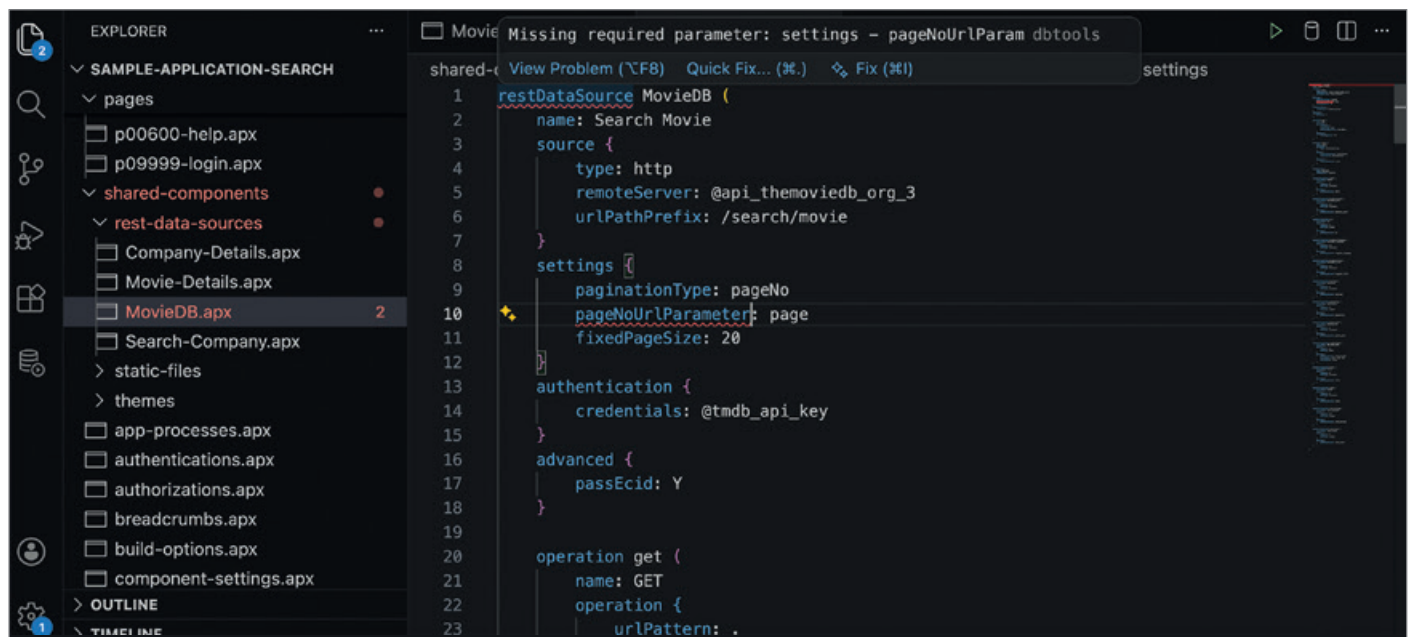


Abbildung 2: APEXlang-Datei in VSCode: Ein Attribut ist nicht korrekt (Quelle: Carsten Czarski)

mehreren Datasets, bereit. Die Anwendung kann nur Tabellen und Spalten aus dem Dataset nutzen – PL/SQL-Funktionen oder andere Tabellen sind nicht zugänglich.

- Data Reporter bringt ein eigenes Nutzerkonzept mit:
 - Data-Reporter-**Administratoren** können Datasets konfigurieren, Anwendungen erstellen und Data-Reporter-Nutzer pflegen.

- Ein Data Reporter **Editor** kann Anwendungen bearbeiten, aber keine Datasets ändern.
- Ein Data Reporter **Viewer** kann bestehende Anwendungen betrachten, erhält also Zugang zu den Daten.
- APEX-Workspace-Administratoren sind immer auch Data-Reporter-Administratoren.

Zuerst richtet man also ein *Dataset* im

Data Reporter ein (siehe Abbildung 10).

Nachdem ein Dataset angelegt wurde, kann eine *Reporting-Anwendung* und darin mehrere *Reports* erzeugt werden. APEX 26.1 startet mit drei einfachen Report-Typen: **Interactive Report**, **Faceted Search mit einer Tabelle** sowie **Faceted Search mit einer Content-Row Darstellung**. Abbildung 11 zeigt, wie ein bestehender Data Reporter Report bearbeitet werden kann – es wird sofort deutlich,

Layout
Page Search
Help

Static ID

Enter a unique Static ID to identify this component in API calls or refer to it in application export files. If you change the Static ID, dependent components will retain their references, but any existing API calls using the old ID must be updated manually.

Additional Information

- Type: Static ID

Button

> Identification

> Layout

> Appearance

> Behavior

Advanced

HTML DOM ID

Static ID

Custom Attributes

> Server-side Condition

> Security

> Configuration

Abbildung 3: Die neue Static ID am Beispiel eines Buttons (Quelle: Carsten Czarski)

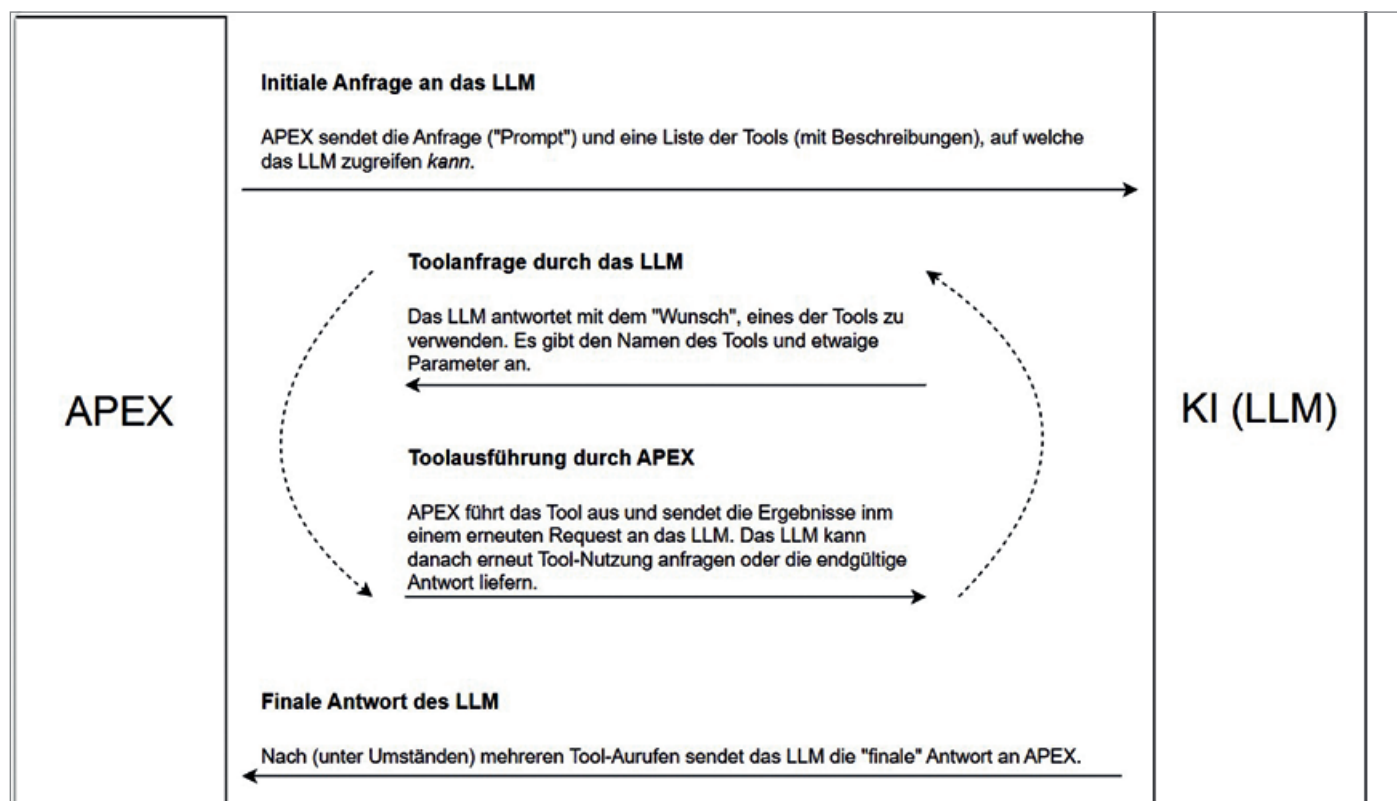


Abbildung 4: Zusammenspiel zwischen APEX und AI-Service beim Einsatz von Tools (Quelle: Carsten Czarski)

dass Data Reporter Reports im Vergleich zu Reports auf APEX-Seiten stark vereinfacht sind.

In künftigen Versionen ist mit weiteren Report-Typen zu rechnen. Die Inter-

active-Report-Variante bringt natürlich auch die oben beschriebene KI-Unterstützung mit, so dass Nutzer des Data Reporter bequem mit KI arbeiten können (siehe Abbildung 12).

Und noch vieles mehr ...

Neben den bisher beschriebenen *Marquee Features* bringt APEX 26.1 eine ganze Reihe neuer Funktionen und Er-

Generative AI Agent

CancelDeleteApply Changes

Show AllIdentificationSubscriptionGenerative AIToolsResponse FormatAdvancedComments

Name	Type	Executes on	Conditional	Updated	Updated By
include_current_date_and_time	Execute Client-side Code	Client-side		5 days ago	STEFAN
include_static_data	Retrieve Data	Server-side			
include_table_details_asp	Retrieve Data	Server-side			

1 - 3

On Demand

Name	Type	Execution Location	Shows Notification	Conditional	Updated	Updated By
ask_user_for_confirmation	Execute Client-side Code	Client-side				
do_the_other_thing	Execute Server-side Code	Server-side	Yes			
do_the_thing	Execute Server-side Code	Server-side	Yes			
do_the_third_thing	Execute Server-side Code	Server-side	Yes			
get_application_pages	Retrieve Data	Server-side				

Tools to enable richer interactions with Generative AI services.

These Agents can define one or more Tools that can be used by the AI service to retrieve relevant information on demand in order to improve response quality, and perform tasks.

[Learn More ...](#)

☐ Return To Page

Abbildung 5: Ein deklarativer AI-Agent mit Tools (Quelle: Carsten Czarski)

Generative AI Tool

CancelDeleteApply Changes

Show AllIdentificationNotificationParametersSettingsServer-Side ConditionSecurityAdvancedAgentComments

Identification

Nameget_applications?

TypeRetrieve Data?

Execution PointOn Demand?

DescriptionRetrieves a list of applications in the current workspace. Returns the application id and the application name.?

Settings

TypeSQL Query?

SQL Query

↶↷🔍🔧A-Z✔APEX Assistant⚙️?

1select application_id,
2application_name
3from apex_applications

Abbildung 6: Das get_applications-Tool liefert eine Liste der APEX-Anwendungen (Quelle: Carsten Czarski)

weiterungen – diese alle im Detail zu beschreiben, würde den Rahmen des Artikels bei weitem sprengen, daher sind die Wichtigsten im Folgenden kurz aufgezählt.

- Für Interactive Reports fällt das **Limit von 32 KB** für den HTML-Content einer Zeile weg: Interactive Reports mit einer großen Anzahl an Spalten sind kein Problem mehr.

- Wie schon für Template Components steht **deklarative Row Selection** nun auch für den Interactive Report zur Verfügung. Einfach eine Spalte des Typen **Row Selection** hinzufügen, den Rest macht APEX. Informationen über selektierte Zeilen werden in einem Page Item bereitgestellt, was der Entwickler leicht weiterverarbeiten kann.
- Für Faceted Search und Smart Filters steht ein neuer **Exclude-Modus** be-

reit: In diesem Modus kann der Nutzer Filter-Optionen *ausschließen* statt selektieren.

- Die Map Region bringt Unterstützung für die **Vector-Tile-Layer**-Funktion der Datenbank mit. Werden große Datenmengen (beispielsweise mehrere zehntausend Objekte) auf einer Karte dargestellt, so wird die Performance mit dieser Funktion um Faktoren besser.

Talk to your Data

Search or ask a question...

Group By: Deptno

Deptno	Total Salary (SUM of SAL)
20	10,875
30	9,400
10	8,750
666	1,234
	1 - 4

Assistant

summarize salary by department

The report now groups by **DEPTNO** with **Total Salary** (SUM of SAL) displayed for each department.

Abbildung 7: Ein Interaktive Report wird mit natürlicher Sprache gesteuert (Quelle: Carsten Czarski)

Layout Page Search **Help**

Report Context

Provide a short, human-readable description of what this report contains or its purpose (e.g., "Employee directory with location, department, and work arrangement"). This helps the AI better understand the meaning, context, and usage of user prompts. Supplying context is optional but can improve the accuracy and relevance of results.

Additional Information

- Type: Text
- Supported Substitutions: Application, Page Items and System Variables

Region Attributes Printing

Filter

Generative AI

Support

Default Search Mode: Search with AI

Report Context: This report contains employee data

> Search Bar

> Actions Menu

Abbildung 8: Der Report-Kontext enthält zusätzliche Informationen für den AI-Service (Quelle: Carsten Czarski)

Enable Generative AI
Accelerate development with AI-assisted coding, debugging, and app generation. [Learn More...](#) **Enable AI**

Welcome to Data Reporter

Data Reporter makes it easy to create reporting applications based on the data in your schema without the overhead of full application development.

To get started, first create a dataset by grouping related tables or views. Then, use that dataset to build a reporting application and define the reports you need.

Create Dataset

Group related tables and views into a reusable dataset for reporting.

Create Dataset Import

Create a Reporting Application

Build an application to generate and manage reports based on a defined dataset.

Create Application

Abbildung 9: Data Reporter in APEX 26.1 (Quelle: Carsten Czarski)

×

Create Dataset

Name
 Mein Dataset

Select Tables / Views

☐ Select All

☒	TABLE EMP	8 columns	i
☒	VIEW EMP_DEPT_V	10 columns	i
☐	TABLE EBA_DEMO_REST_EMPLOYEE	8 columns	i
☐	TABLE EBA_DEMO_REST_EMPLOYEECUSTOM	6 columns	i

2 items selected
Total 4

Cancel
Create Dataset

Abbildung 10: Data Reporter Datasets (Quelle: Carsten Czarski)

✓ Report created.
×

Faceted Search with Data Table

Mein Report

Unpublished

⋮
Publish
▶

Show All
Data Source
Facets and Columns
Descriptors

Table / View
EMP_DEPT_V

Owner
CCZARSKI

Facets and Columns

Actions
▼

Sequence	Column	Label	Format Mask	Displayed	Facet Enabled	Sortable	Drill Down	Actions
1	EMPNO	Empno		✓		✓		✎
2	ENAME	Ename		✓		✓		✎
3	JOB	Job		✓		✓		✎
4	MGR	Mgr		✓		✓		✎
5	HIREDATE	Hiredate		✓		✓		✎

Abbildung 11: Bearbeiten eines Data Reporter Reports (Quelle: Carsten Czarski)

The screenshot shows the Oracle APEX Data Reporter interface. At the top, there's a header 'My App' and a title 'Data Reporter Report using AI' with a status 'Unpublished'. Below the title is a subtitle 'Interactive view of employee and department data with easy search, filter, and sort options.' A search bar is present with the placeholder 'Search or ask a question...'. On the left, there's a 'Group By' button. The main area displays a table with two columns: 'Deptno' and 'Total Salary'. The table has four rows of data. On the right, there's an 'Assistant' panel with a chat interface. The chat shows a message: 'group salary by department' and a response: 'The report now groups by DEPTNO with Total Salary (SUM of SAL). Use "add sum of salary" or similar if you want more aggregates, or specify sorting like "sort by total salary descending".' At the bottom of the Assistant panel is a text input field 'Type your message here'.

Deptno	Total Salary
20	10,875
30	9,400
10	8,750
666	1,234

Abbildung 12: Ein Data-Reporter-Bericht in Aktion – mit KI (Quelle: Carsten Czarski)

- Die APEX Workflow Engine führt **Parallel Flows** ein – Workflow-Aktivitäten können also parallel ausgeführt werden; der Workflow wird dann beendet, wenn alle parallelen Flows abgeschlossen sind.
- Im Bereich REST Data Sources ergeben sich ebenfalls viele kleinere Neuerungen. Der **OData Adapter** unterstützt nun auch REST-„Schreib“-Methoden wie POST, PUT und DELETE, so dass ein Formular auf einen OData REST-Service rein deklarativ und ohne zusätzlichen Code möglich ist.
- Für Web Credentials vom Typ OAuth werden nun **Refresh Tokens** unterstützt, weiterhin werden der **OAuth2 Password-Flow** und **Signed User Assertions** nun von Web Credentials unterstützt.
- Quick Picks** für Page Items unterstützen nun eine SQL-Abfrage als Datenquelle. Damit können diese wesentlich dynamischer gestaltet werden als bislang.
- Beispieldaten (*Sample Data*) sind nun direkt in den Page Designer und den Application Builder integriert. Das ist besonders nützlich, wenn man zunächst nur eine Seitenstruktur erstellen möchte: Wird **Sample Data** als Datenquelle ausgewählt, braucht APEX *keine weiteren* Informationen für eine lauffähige Komponente.
- Die Unterstützung für **Content-Security-Policy (CSP)** wurde verbessert

und erweitert: Die CSP-Direktiven **unsafe-hashes** und **unsafe-inline** sind für moderne APEX-Komponenten nicht mehr nötig.

- Für den Versand von E-Mails können **SMTP-Zugangsdaten** (Credentials) für den Mail-Server nun auf Workspace-Ebene konfiguriert werden.
- Im Bereich der AI-Services gibt es noch mehr Erweiterungen: So unterstützt APEX 26.1 zusätzliche AI-Anbieter wie Google Gemini, Mistral AI, Anthropic Claude und Ollama. Weiterhin kann pro AI-Service ein **Token-Maximum** hinterlegt werden. APEX setzt dieses durch, wenn der AI-Service Informationen über verbrauchte Tokens zurückliefert.

Zusammenfassung

Auf APEX 26.1 musste die Community warten. Dafür ist das Release mit neuen Funktionen vollgepackt. Die größte (und für das Entwicklerteam aufwändigste) Neuerung ist die neue Domain Specific Language **APEXlang**. Erstmals gibt es eine offene und dokumentierte Sprache zur Beschreibung von APEX-Anwendungen – und diese wird aus dem Stand heraus von der SQL Developer Extension für VSCode unterstützt. Der Umgang mit APEX-Anwendungen wird mit APEXlang sicherlich völlig neu definiert – im Bereich CI/CD, aber auch beim Generieren von APEX-Anwendungen mit KI.

Im Bereich KI-Unterstützung macht APEX ebenfalls einen großen Sprung. AI-Agenten erlauben es, KI umfassend in eine APEX-Anwendung zu integrieren – KI kann damit ganze Bereiche einer Anwendung steuern und dem Endbenutzer großen Mehrwert liefern.

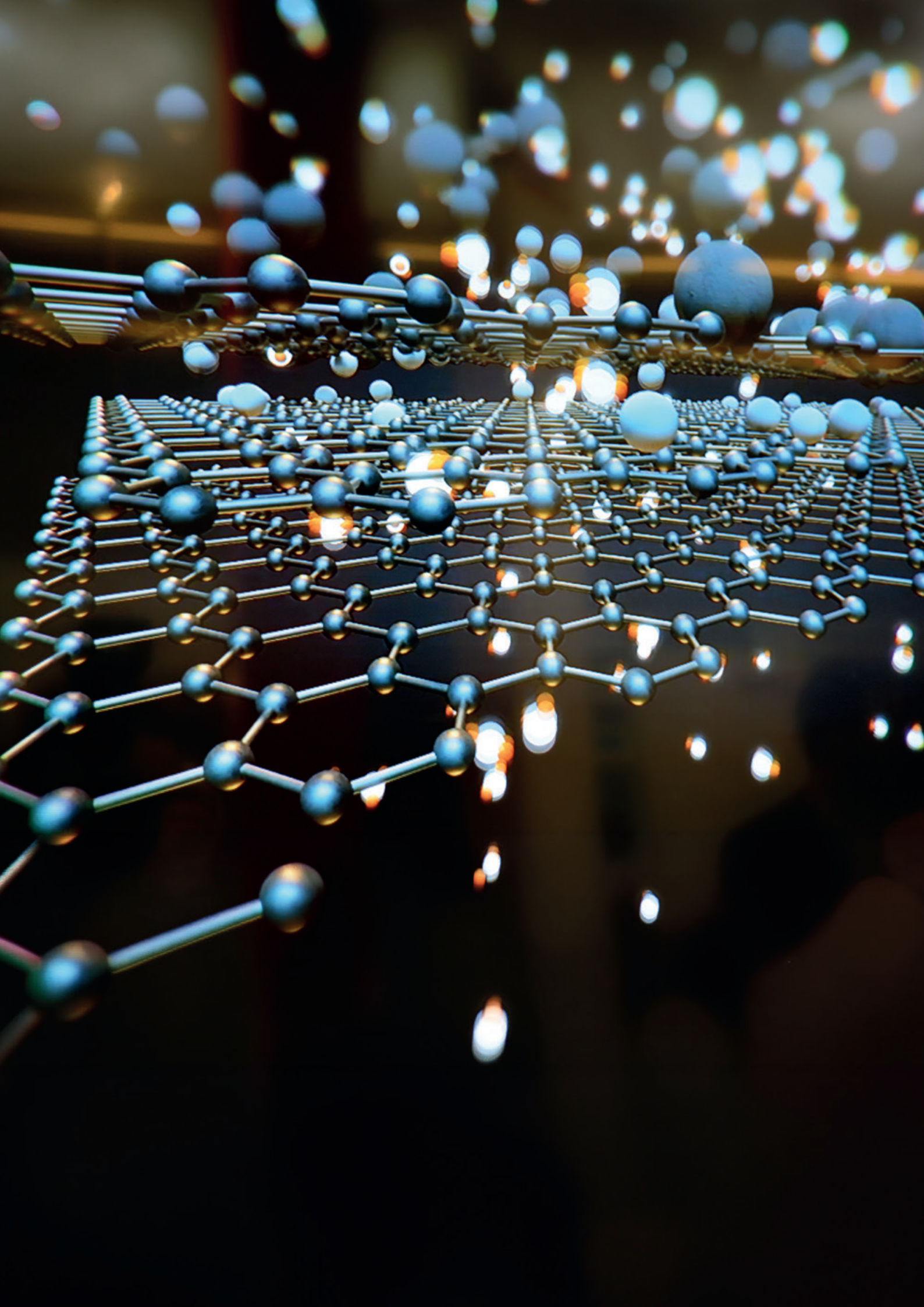
Daneben gibt es zahlreiche Features in allen Funktionsbereichen von APEX. Sei es Interactive Report, Faceted Search, REST Data Sources, Workflow oder PL/SQL APIs: In nahezu allen Bereichen gibt es Verbesserungen.

Weitere Informationen

- [1] Informationen zu Oracle APEX und Demo-Umgebung <http://apex.oracle.com>
- [2] Oracle APEX Blog <http://blogs.oracle.com/apex>



Carsten Czarski
Architect of Oracle APEX
carsten.czarski@oracle.com



Innovative Applikationen mit Oracle 26ai, APEX und Graph SQL

Dr. Holger Friedrich, sumIT

Oracle verfolgt seit vielen Jahren den Designansatz einer „converged engine“ bei der Weiterentwicklung der Oracle-Datenbank. Mit Oracle Database 26ai können Property-Graphen mit Graph SQL direkt im Datenbankkern genutzt werden. Komplexe Beziehungen lassen sich so deklarativ beschreiben und mit klassischem SQL gemeinsam mit relationalen Daten auswerten. Zudem ermöglicht Oracle APEX die effiziente Erstellung von Anwendungen. Diese machen Daten und Auswertungen Endbenutzern für verschiedenste Zwecke intuitiv zugänglich.

Die Datenwelt ist dieser Tage fragmentiert. Relationale Daten, dimensionale Modelle, semistrukturierte in Form von JSON-Dokumenten, Graphen – all diese Darstellungsformen koexistieren in Unternehmen. Gleichzeitig benötigen unterschiedliche Use-Cases Zugriff auf denselben Datenbestand.

Viele Anbieter reagieren auf diese Vielfalt der Datenmodellierung und der Anforderungen an den Zugriff mit einer Viel-

zahl spezialisierter Engines und Services. Für jede Form der Speicherung wird ein eigenes System angeboten, für jede Abfragesprache ein eigenes API, für jeden Workload eine separate Infrastruktur. Das führt zu vielen Herausforderungen und Fragen. Welche Engine ist die richtige? Welche Sprache setzen wir ein? Hat das Team die nötigen Skills? Wie organisieren wir Betrieb, Sicherheit und Governance über mehrere Systeme hinweg?

Mit dem konvergenten Designansatz verfolgt Oracle eine andere Strategie. Oracle 26ai bietet eine zentrale Engine, die offen für unterschiedliche APIs und Sprachen ist und SQL als universelle Zugriffsschicht bietet. Relationale, dimensionale, Dokument- und Graphdaten werden nicht als konkurrierende Modelle verstanden, sondern als unterschiedliche Sichtweisen auf denselben Datenbestand. Datenbank-Views fungieren dabei als Zugriffsstrukturen für verschiedenartige Workloads. *Abbildung 1* verdeutlicht den Ansatz der „converged database engine“ [1].

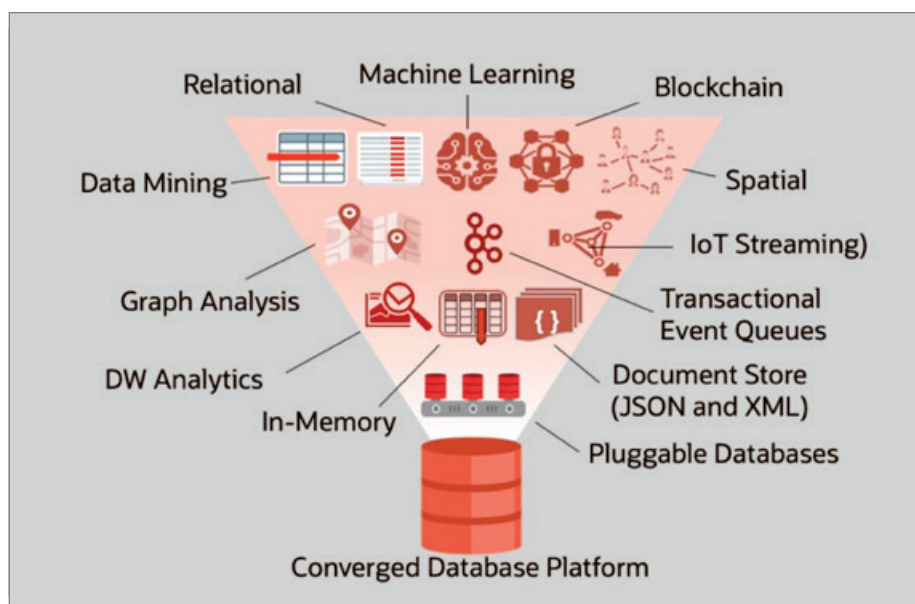


Abbildung 1: Leistungsumfang von Oracles „converged database engine“ (Quelle: Oracle)

Property-Graphs zur Modellierung von Beziehungen

Property-Graphen gewinnen in modernen Anwendungen immer mehr an Bedeutung. Sie modellieren Daten als Knoten (Vertices) und Kanten (Edges), wobei sowohl Knoten als auch Kanten Attribute, sogenannte Properties, besitzen. Damit lassen sich Beziehungen, Abhängigkeiten, Datenflüsse, Prozesse oder Interaktionen natürlich und intuitiv beschreiben.

Anwendungsfelder finden sich quer durch die Wissenschaften, sei es in der In-

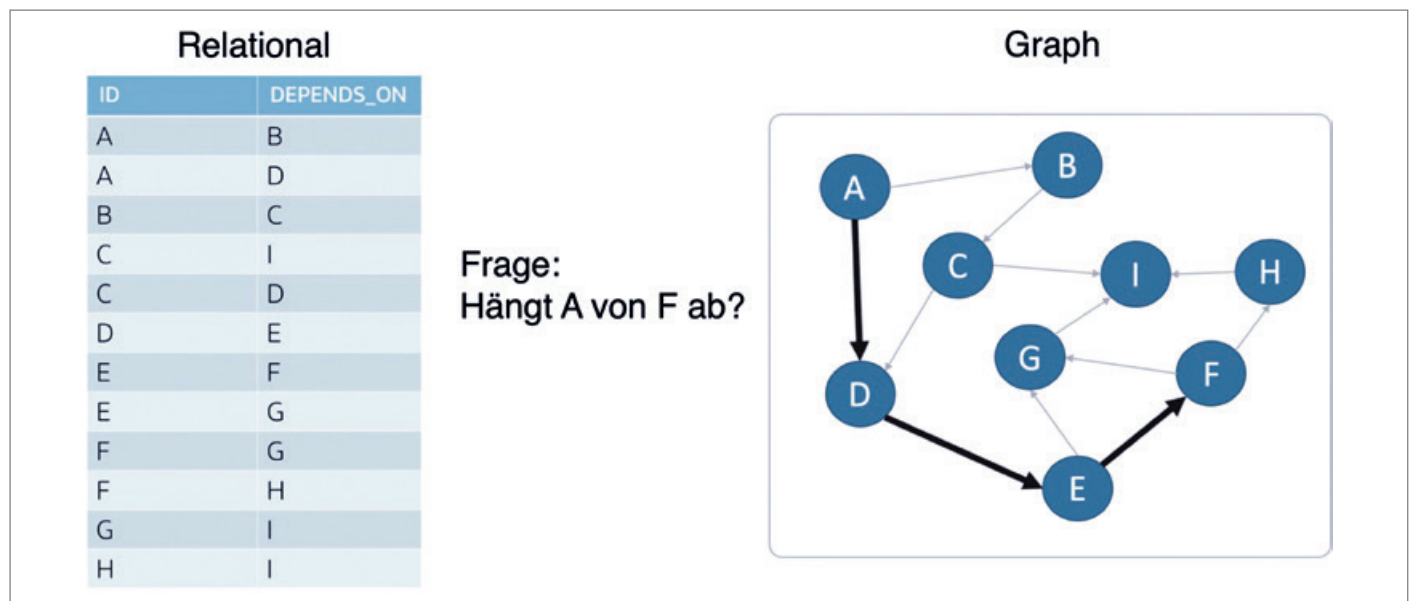


Abbildung 2: Graphen erlauben eine effiziente Analyse komplexer Beziehungen (Quelle: Dr. Holger Friedrich)

formatik, bei Abhängigkeiten oder Netzwerken, in den Sozialwissenschaften für Beziehungsanalysen oder in Pharmazie und Chemie zur Modellierung molekularer Strukturen. Zunehmend werden Graphen auch im Unternehmenskontext genutzt. Graphen sind dabei nicht nur ein akademisches Konzept. Sie erlauben es, Fragen einfach zu beantworten, deren Beantwortung und deren Lösung im relationalen Modell umständlich oder ineffizient wäre. *Abbildung 2* zeigt ein einfaches Beispiel der Überlegenheit der Modellierung von Daten als Graph für die Beantwortung vieler Fragen.

Existiert eine Abhängigkeit zwischen Objekt A und Objekt F? Wie viele indirekte Verbindungen bestehen? Welche Pfa-

de führen von A nach I? Welche Verbindungen bestehen in genau drei oder vier Schritten? Welcher ist der kürzeste Pfad zwischen zwei Entitäten? Fragen dieser Art sind in Graph-Strukturen konzeptionell einfach und intuitiv, klassisch relational mit SQL sind sie dagegen oft schwer les- und lösbar.

Graph SQL als Erweiterung im Datenbank-Kernel

In neueren Versionen der Datenbank, namentlich 23ai und 26ai, hat Oracle die Repräsentation und Abfrage von Property-Graphen direkt in die Datenbank-Engine integriert. Während frühere Versionen

wie 19c oder 21c mit einem separat zu pflegenden Mid-tier-Graph-Server arbeiteten, enthält die Datenbank nun eine SQL-Erweiterung, die auf dem ISO-Standard ISO/IEC 9075-16 [2] basiert. Die Dokumentation zu diesem Feature und seiner Nutzung findet sich hier [3].

Die Definition eines Property-Graphen erfolgt deklarativ über ein SQL-Statement und wird in Form einer Datenbank-View repräsentiert. Relationale Tabellen werden als Vertex- oder Edge-Tabellen interpretiert. Primärschlüssel definieren die Identität von Knoten und Kanten, Fremdschlüsselbeziehungen modellieren Quellen und Ziele von Kanten und damit von Beziehungen. Properties definieren die Men-

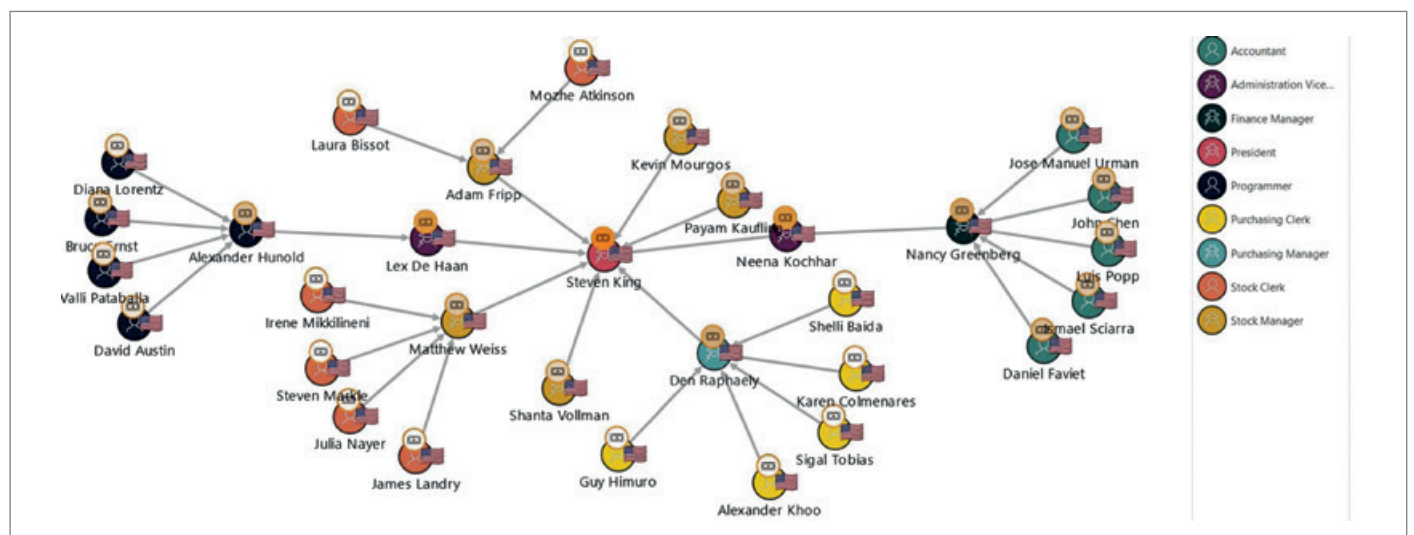


Abbildung 3: Graphvisualisierung in APEX mit dem Plug-in (Quelle: Oracle)

```
CREATE OR REPLACE PROPERTY GRAPH PAYMENT_GRAPH
  VERTEX TABLES (
    CUSTOMER as CUSTOMERS KEY (UUID)
    LABEL CUSTOMERS
    PROPERTIES (UUID, FIRST_NAME, LAST_NAME))
  EDGE TABLES (
    PAYMENT
    KEY (UUID)
    SOURCE KEY (SOURCE_UUID) REFERENCES CUSTOMERS (UUID)
    DESTINATION KEY (TARGET_UUID) REFERENCES CUSTOMERS (UUID)
    LABEL TRANSFERS
    PROPERTIES (PAYMENT_TYPE, ORDER_STATE, AMOUNT));
```

Listing 1: Graph-Definition in Graph SQL

```
-- Top 10 cust sending money
SELECT sender_id, COUNT(1) AS Num_Transfers
FROM
  GRAPH_TABLE (PAYMENT_GRAPH
    MATCH (src)
      - [p IS PAYMENT WHERE p.payment_type = 'P2P']
      -> (dst)
    COLUMNS (source.uuid AS sender_id))
GROUP BY sender_id
ORDER BY Num_Transfers DESC
FETCH FIRST 10 ROWS ONLY;
```

Listing 2: Graph-SQL-Abfrage zur Top-10-Analyse von Zahlungen

ge der auszugebenden Attribute. Mit der Vergabe von Labels kann eine individuelle Benennung der Knoten- und Kantenmengen je Graph definiert werden. Listing 1 demonstriert die Definition eines Graphen zur Analyse von Zahlungen zwischen Kunden in Graph SQL.

Abfragen gegen Property-Graph-Views erfolgen über die neue Match-Klausel in Graph SQL. Muster, die erkannt werden sollen, werden direkt in SQL formuliert. Dabei repräsentieren runde Klammern Knoten, eckige Klammern Kanten. Gerichtete oder ungerichtete Traversierungen werden durch Striche und Pfeile spezifiziert, die Anzahl von Schritten durch Intervalle, zum Beispiel {1,10}. Zusätzlich gibt es Funktionen, die die Ausgabe filtern, etwa ob ein ganzer Pfad oder nur Anfangs- und Endknoten ausgegeben werden sollen, sowie welche Properties der Knoten im Result-Set enthalten sein sollen. Listing 2 zeigt ein Beispiel einer Analyse von Peer-to-Peer-Zahlungen, bei der die Top-10 Kunden gefunden werden, die am meisten Geldsendungen getätigt haben.

Abfragen mit Graph SQL geben das Ergebnis als reguläres relationales Re-

sult-Set aus. Der Graph-SQL-Query kann als Subquery in eine größere Abfrage eingebettet werden. Seine Ergebnismenge kann also durch Joins, Aggregationen, Gruppierung, Pattern Matching und so weiter mit anderen Daten verbunden und innerhalb einer Ausführung weiterverarbeitet werden. Genau diese Integration ist der entscheidende Vorteil von Oracles konvergenter Datenarchitektur. Es gibt keinen Kontextwechsel, keine separate Engine wird benötigt, keine zusätzlichen APIs, kein aufwendiger Datentransport zwischen Systemen ist zu leisten. Abfragen von Graphen sind Teil derselben Datenbank, derselben Transaktionen und derselben Sicherheitsarchitektur.

Graph-Analyse und -Visualisierung mit Application Express (APEX)

Eine Analyse in SQL allein genügt jedoch nicht für den Einsatz von Graphen in Unternehmensanwendungen. Graph-Strukturen wollen von Anwendern angesehen, analysiert und auch manipuliert werden.

Hier kommt Oracle Application Express (APEX) [4] ins Spiel. Als Low-Code-Plattform ermöglicht APEX seit mehr als zwei Jahrzehnten die schnelle Entwicklung datenbanknaher Anwendungen. Seit neuestem beinhaltet dies die Visualisierung von Graphen und die Darstellung der Ergebnisse von Graph-SQL-Queries.

Zu diesem Zweck bietet Oracle ein Plug-in zum Download an [5]. Mit diesem Plug-in können Property-Graphen aus einer (auch autonomen) Datenbankinstanz visualisiert werden. Knoten und Kanten lassen sich selektieren, gruppieren, ein- oder ausblenden und vieles mehr. Layouts können dynamisch geändert werden. Unterschiedliche Stile für unterschiedliche Vertex- oder Edge-Typen lassen sich definieren. So gar zeitliche Veränderungen von Beziehungen in Graphen können dargestellt werden, wenn die Ausgangsdaten in der Datenbank temporale Information enthalten. Installation und Nutzung des Visualisierungs-Plug-ins sind in [6] dokumentiert. Abbildung 3 illustriert die Darstellung eines Beziehungsgraphen per Plug-in in APEX.

Praxisbeispiel: Graph-Analyse von Prozessdaten und Visualisierung in APEX

Zugriffs-, Lineage- und Impact-Analysen von Verarbeitungsprozessen auf Datenplattformen sind seit jeher essenziell, um eine Übersicht zu haben und Governance zu gewährleisten. Graphen eignen sich hervorragend dazu, Datenflüsse zu repräsentieren und zu analysieren. Daher dient dieser Anwendungsfall im Folgenden als Beispiel, um zu demonstrieren, wie einfach und vorteilhaft die Kombination aus Oracle 26ai, Graph SQL und APEX in der Anwendung sind. Dabei werden Designdaten aus dem Repository des Oracle Data Integrator verwendet. Nach dem Setup von APEX und des Graph-Plug-ins sind folgende Schritte für den schnellen Aufbau von Datenbasis, Graph und einfacher Analyse-Applikation notwendig.

- Extraktion der relevanten Metadaten aus dem ODI-Repository
- Definition von Graphen zur Repräsentation von Verarbeitungsschritten und

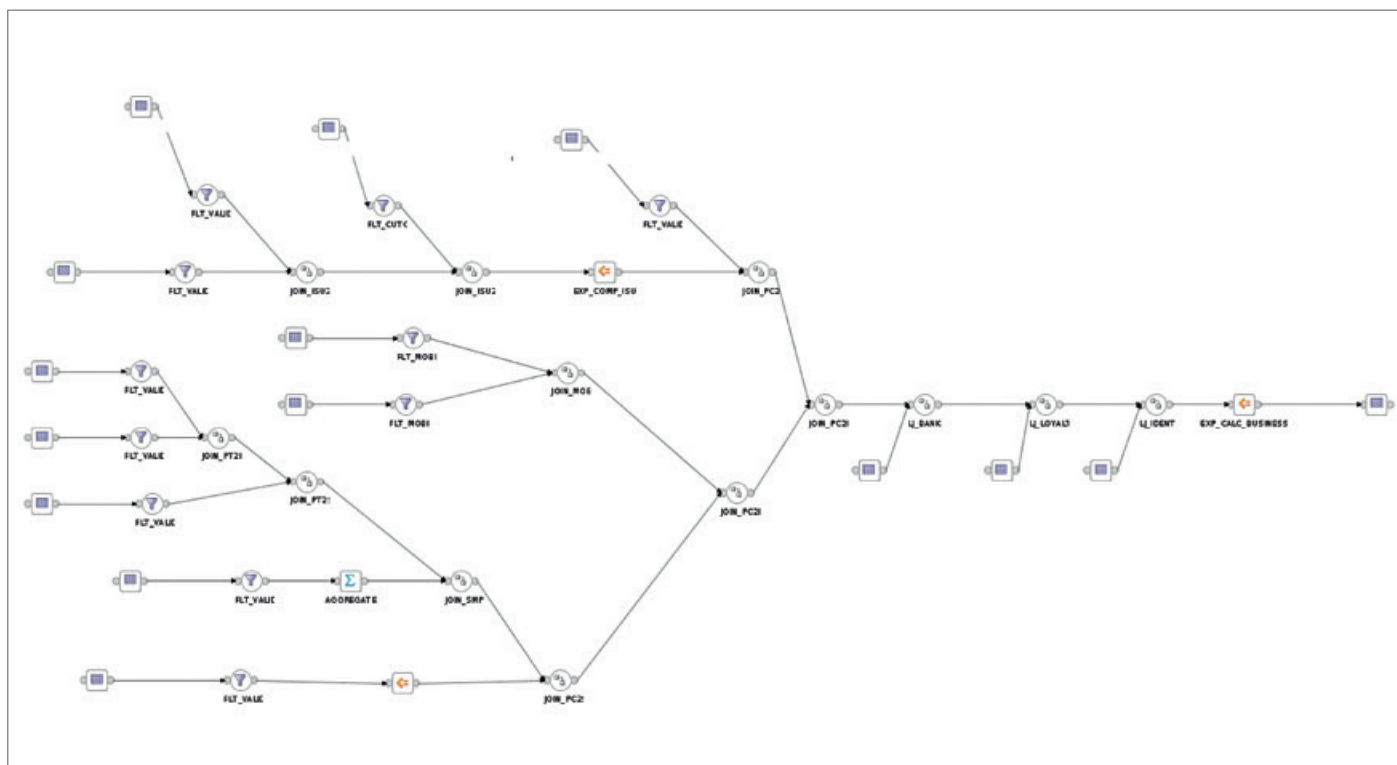


Abbildung 4: Komplexes Mapping im Oracle Data Integrator (Quelle: Dr. Holger Friedrich)

▼ Source

Location

Local Database ▼

Type

SQL Query ▼

SQL Query

↗

```

SELECT src, cn, dst
FROM GRAPH_TABLE ( intra_map_op_lineage_g
MATCH (src IS map_operators_t ) -[e IS
map_operator_connections_t

where e.map_id = src.map_id

and e.map_id = dst.map_id]-> (dst is
map_operators_t)
WHERE 1=1
and src.map_id = :P11_MAP
COLUMNS (vertex_id(src) AS src,
edge_id(e) AS cn, vertex_id(dst) AS dst )
)

```

Region
Attributes

Filter

↶

Type

Property Graph ▼

Graph Owner

Parsing Schema ▼

Graph Name

INTRA_MAP_OP_LINEAGE_G ▼

Match Clause

↗

```

(src IS map_operators_t ) -[e IS
map_operator_connections_t]-> (dst is
map_operators_t)

```

Columns Clause

↗

```

vertex_id(src) AS src, edge_id(e) AS cn,
vertex_id(dst) AS dst

```

Where Clause

↗

```

1=1
and src.map_id = :P11_MAP
and e.map_id = src.map_id
and e.map_id = dst.map_id

```

Abbildung 5: Alternative Möglichkeiten zur Definition von Graph-Queries in APEX (Quelle: Dr. Holger Friedrich)

24 | www.aoug.at • www.doag.org • www.souq.ch

```
-- create property graph for impact/lineage within mappings between operators
CREATE OR REPLACE PROPERTY GRAPH intra_map_op_lineage_g
  VERTEX TABLES (
    map_operators_t
    KEY (map_comp_id)
    PROPERTIES (map_id, mapping_name, map_comp_id
               , comp_alias, comp_type, bound_table))
  EDGE TABLES (
    map_operator_connections_t
    KEY (conn_id)
    SOURCE KEY (map_from_op_id)
    REFERENCES map_operators_t (map_comp_id)
    DESTINATION KEY (map_to_op_id)
    REFERENCES map_operators_t (map_comp_id)
    PROPERTIES (map_id, conn_id
               , map_from_op_id, map_to_op_id));
```

Listing 3: Graph-Definition zur Analyse von ODI-Metadaten



Abbildung 6: Ergebnis komplexer Graph-Analyse in APEX (Quelle: Dr. Holger Friedrich)

dem Datenfluss zwischen den hierfür im ODI genutzten Operatoren

- Konfiguration der gewünschten Analyse-Queries und der Seiten in einer neuen APEX-Applikation

Oracle Data Integrator ist ein graphisches Enterprise-ELT-Werkzeug. Als Basis für Code-Generierung und -Ausführung nutzt es ein relationales Repository,

welches die entsprechenden Metadaten enthält. *Abbildung 4* zeigt ein Beispiel für ein komplexes Mapping zur Datenverarbeitung, in diesem Fall dem Laden einer Dimensionstabelle.

Der erste Schritt ist die Extraktion der relevanten ODI-Metadaten und deren Speicherung in geeigneten Tabellen, auf deren Basis Graphen definiert werden können. Hierzu kann einfaches relatio-

nales SQL verwendet werden. Hilfreich ist dabei die Dokumentation der ODI-Repository-Strukturen. Diese steht auf der Oracle-Support-Website als MOS-Note zur Verfügung [7].

Die Definition von Property-Graphen erfolgt, wie zuvor beschrieben, mit Property-Graph-Views in Graph SQL direkt in der Oracle 26ai Datenbank. *Listing 3* zeigt ein Beispiel für eine solche View-

Definition zur Analyse innerhalb von ODI-Mappings.

Schließlich können in einer schnell erstellten APEX-Applikation Analyse-Queries hinterlegt und das Layout zur Visualisierung der Ergebnisse konfiguriert und implementiert werden. Die Programmierung weiterer applikatorischer Aktionen und der gewünschten Analyse- und Business-Logik erfolgen traditionell in APEX in PL/SQL oder JavaScript. *Abbildung 5* zeigt einen Screenshot der APEX-Entwicklungsumgebung mit den zwei möglichen Darstellungen eines Graph-SQL-Queries, einmal vollständig in einer SQL-Region, einmal in seine Komponenten zerlegt in der speziellen Graph-SQL-Region.

Wie von APEX gewohnt, ist der Entwicklungsprozess der eigenen Applikation schnell und effizient. So können ressourcensparend komplexe Analysen visualisiert, Ergebnisse gefiltert und interaktiv verfeinert werden. Als Beispiel ist in *Abbildung 6* die Visualisierung des Ergebnisses eines Graph-SQL-Queries auf ODI-Metadaten dargestellt, die mit anderen Methoden nur sehr ineffizient möglich gewesen wäre. Es wurden alle Teilgraphen zwischen Tabellen gesucht, die einen UNPIVOT-Operator nutzen. Das zugrundeliegende ODI-Repository enthielt mehrere Tausend einzelne Mappings mit zehntausenden Operatoren. Die Verarbeitung und Visualisierung dauerten dennoch nicht einmal eine Minute.

Bewertung

Die konvergente Architektur der Oracle-Datenbank und das damit zusammenspielende Tooling, wie zum Beispiel APEX, bringt immense Vorteile für den Anwender. Die Stärke liegt dabei jedoch nicht in einzelnen, isolierten Features, sondern im Zusammenspiel der Produkteigenschaften. Relationale Daten, analytische Modelle, Dokumentenstrukturen und Graph-Repräsentation sind keine isolierten Silos, sondern unterschiedliche Zugriffsformen auf denselben Datenbestand. Die darauf definierten Views fungieren als abstrahierende Schicht, SQL ist das verbindende Element bei Erstellung, Verarbeitung und Analyse.

Nicht allein verringert diese Architektur den Entwicklungsaufwand erheblich. Sie reduziert ebenso die infrastrukturelle

Komplexität und vereinfacht Governance und Sicherheit, da keine separaten Systeme betrieben und abgesichert werden müssen. Entwicklungsteams können sich auf fachliche Fragestellungen konzentrieren und diese schnell umsetzen, anstatt Technologieentscheidungen zu diskutieren und Daten zwischen verschiedenen Systemen hin und her zu schaufeln.

Graph und APEX gemeinsam mit den anderen Darstellungsformen ermöglichen zudem die effiziente Implementierung gemischter Use-Cases. Klassisches OLTP, analytische Auswertungen, dokumentenorientierte Zugriffe und Graph-Analysen innerhalb einer Anwendung stellen kein Problem dar. Write-back ist ebenso möglich wie explorative Analyse.

Derzeit ist Graph SQL noch nicht in allen Aspekten so ausgereift wie klassische relationale Funktionen. Der Funktionsumfang ist gegenüber dem Mid-Tier-Graph-Server begrenzt, viele komplexe Algorithmen, zum Beispiel Graphen-Machine-Learning-Algorithmen, sind noch nicht vorhanden. Doch die Richtung ist klar, die Funktionalität im Bereich der Graphen-Analyse ist kein Add-On außerhalb der Datenbank mehr, sondern ein Kernbestandteil, dessen Funktionsumfang fortwährend mit Releases und Release Updates erweitert wird.

Fazit

Die Integration von Property-Graphen und Graph SQL in die Datenbank-Engine markiert einen weiteren Schritt hin zur Komplettierung der konvergenten Datenplattform. In Kombination mit APEX besteht ein leistungsfähiges Werkzeug für die Erstellung innovativer Applikationen.

Gerade im Bereich Datenintegration, beispielsweise bei der Analyse von Abhängigkeiten und Lineage in Design-Metadaten, zeigt sich das Potenzial eindrucksvoll. Was zuvor schwer lesbare Join-Kaskaden erforderte, lässt sich nun deklarativ als Graph-Query formulieren und endbenutzerfreundlich visuell explorieren.

Wer Daten nicht nur speichern, sondern verstehen will, sollte die Verwendung von Graph-Strukturen und -Analysen ernst nehmen. Wer Komplexität reduzieren und seine Ressourcen effizient einsetzen möchte, sollte überlegen, ob eine konvergente Datenbankarchitektur

nicht sinnvoller ist als ein Zoo spezialisierter Engines.

Quellen

- [1] <https://blogs.oracle.com/database/what-is-a-converged-database>
- [2] <https://www.iso.org/standard/79473.html>
- [3] <https://docs.oracle.com/en/database/oracle/property-graph/26.1/index.html>
- [4] <https://www.oracle.com/apex/>
- [5] <https://github.com/oracle/apex/tree/24.2/plugins/region/graph-visualization>
- [6] <https://docs.oracle.com/en/database/oracle/property-graph/26.1/sp-gdg/visualizing-sql-graph-queries-using-apex-graph-visualization-plug.html>
- [7] <https://support.oracle.com>
- [8] <https://www.sumit.ch>

Über den Autor

Holger Friedrich ist Geschäftsführer von sumIT AG [3] und Oracle ACE Director. Seit mehr als 25 Jahren ist er auf den Gebieten Data Management, Data Plattformen, Lake- und Warehousing, Datenintegration und Analytics tätig.



Dr.-Ing. Dipl.-Inform. Holger Friedrich
holger.friedrich@sumit.ch



Jenseits von ChatGPT: Maßgeschneiderte KI-Modelle mit Oracle APEX, PL/SQL, Rust

Andreas Maier, isOn Tec

Die Debatte um Künstliche Intelligenz wird derzeit von generischen Modellen wie ChatGPT dominiert. Doch beim Einsatz in den Kernprozessen eines Unternehmens greifen diese Standardlösungen oft zu kurz, da sie nicht mit dem exklusiven, firmeninternen Wissen trainiert wurden. In diesem Artikel wird gezeigt, warum eigene Modelle – die oft deutlich kleiner und effizienter sind als gedacht – die bessere Lösung für den Praxiseinsatz darstellen. Es wird gezeigt, wie man mit Oracle APEX eine leistungsstarke Steuerzentrale erstellen kann und warum die Programmiersprache Rust ein guter Partner ist, um hochperformante neuronale Netze zu entwickeln und in Geschäftsanwendungen zu integrieren.

Wenn Allgemeinwissen nicht reicht: Warum eigene Modelle?

Die Leistungsfähigkeit aktueller Large Language Models (LLMs) ist durchaus bemerkenswert. Viele von uns nutzen sie bereits regelmäßig im Alltag. Dass Fragen gut beantwortet werden, liegt

schlicht an der schieren Masse der Trainingsdaten – und genau das hat zu der Annahme geführt, ein LLM sei ein universelles Werkzeug für jeden Anwendungsfall. Für allgemeine Fragen und Aufgaben stimmt das oft auch – das nötige Wissen steckt bereits durch die Trainingsdaten in den LLMs. Viele Aufgaben in Unternehmen sind jedoch weitaus

spezifischer und folgen einer firmeninternen Logik, die in keinem öffentlichen Datensatz zu finden ist. Ein anschauliches Beispiel für dieses Kontext-Problem: die automatisierte Verarbeitung von Kundenkorrespondenz in der Versicherungswirtschaft. Betrachtet man eine Nachricht wie „Hiermit kündige ich meinen Schutzbrief“, so ist die rein

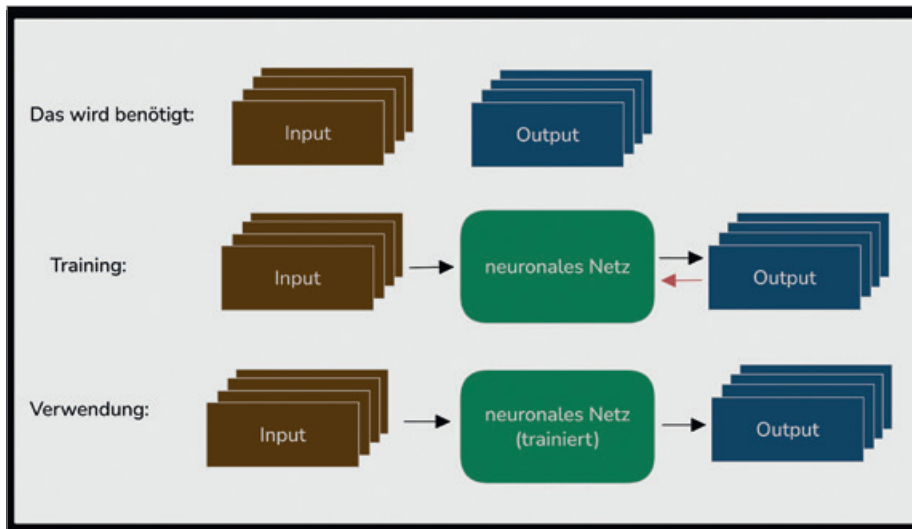


Abbildung 1: Training eines eigenen Neuronales Netzes (Quelle: Andreas Maier)

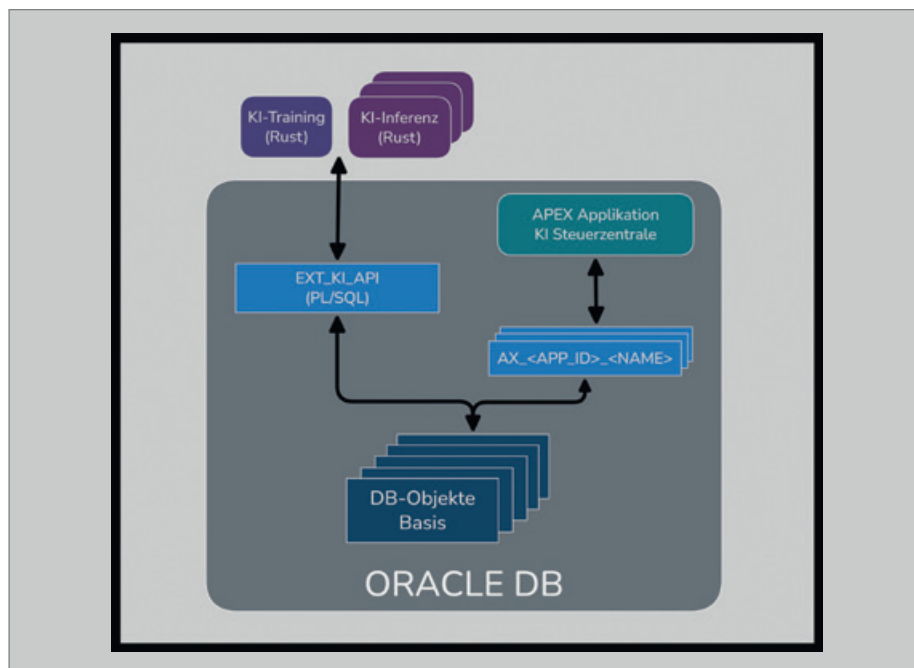


Abbildung 2: Aufbau der Architektur (Quelle: Andreas Maier)

sprachliche Bedeutung für ein LLM eindeutig: eine Vertragskündigung. Doch die geschäftliche Wahrheit hinter diesem Satz ist weitaus unternehmensspezifischer, als man im ersten Moment annimmt:

- Unternehmen A führt den Schutzbrief als eigenständigen Vertrag. Hier wird der Vorgang als klassische Vertragskündigung gewertet.
- Unternehmen B führt den Schutzbrief als Baustein innerhalb eines Vertrages. Hier ist der Vorgang eine Deckungsänderung und keine Vertragskündigung.
- Unternehmen C führt den Schutzbrief in Produkt A (zum Beispiel KFZ) als Bau-

stein und in Produkt B (zum Beispiel Hausrat) als eigenständigen Vertrag.

Ein generisches LLM kann diese firmeninterne Logik, die in der Realität noch deutlich komplexer ist, natürlich nicht kennen. Um ein Modell auf solche Besonderheiten vorzubereiten, müssen umfangreiche Prompts erstellt werden, die versuchen, das gesamte Regelwerk der Kundenkorrespondenz in natürlicher Sprache abzubilden. Dies ist in der Praxis kaum realisierbar und endet oft in hunderten Zeilen mit Anweisungen, die trotz ihres Umfangs oft lückenhaft bleiben, weil sich Erfahrungswissen und unzählige Sonderkons-

tellationen in einem Prompt schlicht nicht vollständig abbilden lassen. Ein eigenes, auf den historischen Daten des Unternehmens trainiertes neuronales Netz hingegen lernt diese firmeninterne Logik durch konkrete Beispiele.

Eine Analogie macht den Unterschied greifbar: Stellen wir uns zwei neue Mitarbeiter vor, die Dokumente klassifizieren sollen.

- Mitarbeiter 1 (LLM) hat eine gute Allgemeinbildung, kennt aber das Unternehmen nur von außen. Er erhält an seinem ersten Tag lediglich eine grobe Anleitung.
- Mitarbeiter 2 (eigenes Modell) erhält eine intensive Einweisung von einem erfahrenen Mitarbeiter anhand tausender konkreter Beispiele.

Ein eigenes Modell ist genau auf diese Aufgabe spezialisiert und benötigt keine seitenlangen Erklärungen – die Logik ist tief in den Gewichten des Netzwerks verankert. Das Ergebnis ist eine deutlich höhere Stabilität und Präzision in der Klassifizierung. Doch die Vorteile gehen über die reine Qualität hinaus: Durch die Spezialisierung sind solche Modelle viel kleiner und oft nur wenige Megabyte groß. Sie benötigen meistens keine teuren GPU-Cluster, sondern viele Modelle laufen performant auf vorhandener CPU-Infrastruktur – was sowohl Latenzen als auch Kosten erheblich reduziert.

Das Prinzip reicht weit über die Textklassifizierung hinaus. Ob Qualitätsprüfung in der Produktion, wo ein eigenes Modell anhand interner Standards zwischen Toleranz und Ausschuss unterscheidet, oder Schätzung der Schadenhöhe in der Versicherung, wo historische Reparaturkosten bereits beim ersten Foto eine belastbare Prognose ermöglichen, der Kern bleibt derselbe: Die wertvollsten Daten liegen im Unternehmen selbst – und standen zum Trainingszeitpunkt keinem der LLM zur Verfügung.

Vom Datensatz zum Modell: Wie ein neuronales Netz lernt

Ein neuronales Netz ist eine mathematische Funktion, die aus einem Input einen Output berechnet. Es besteht aus Schichten von Neuronen, die über gewichtete

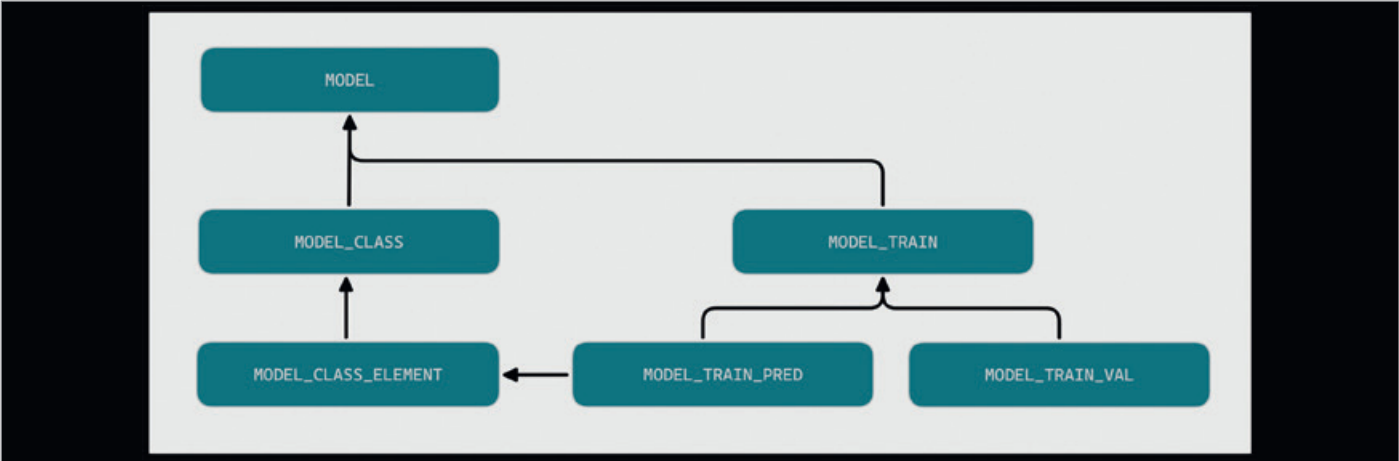


Abbildung 3: Ein einfaches Datenmodell zur Verwaltung von Modellen und Datasets (Quelle: Andreas Maier)

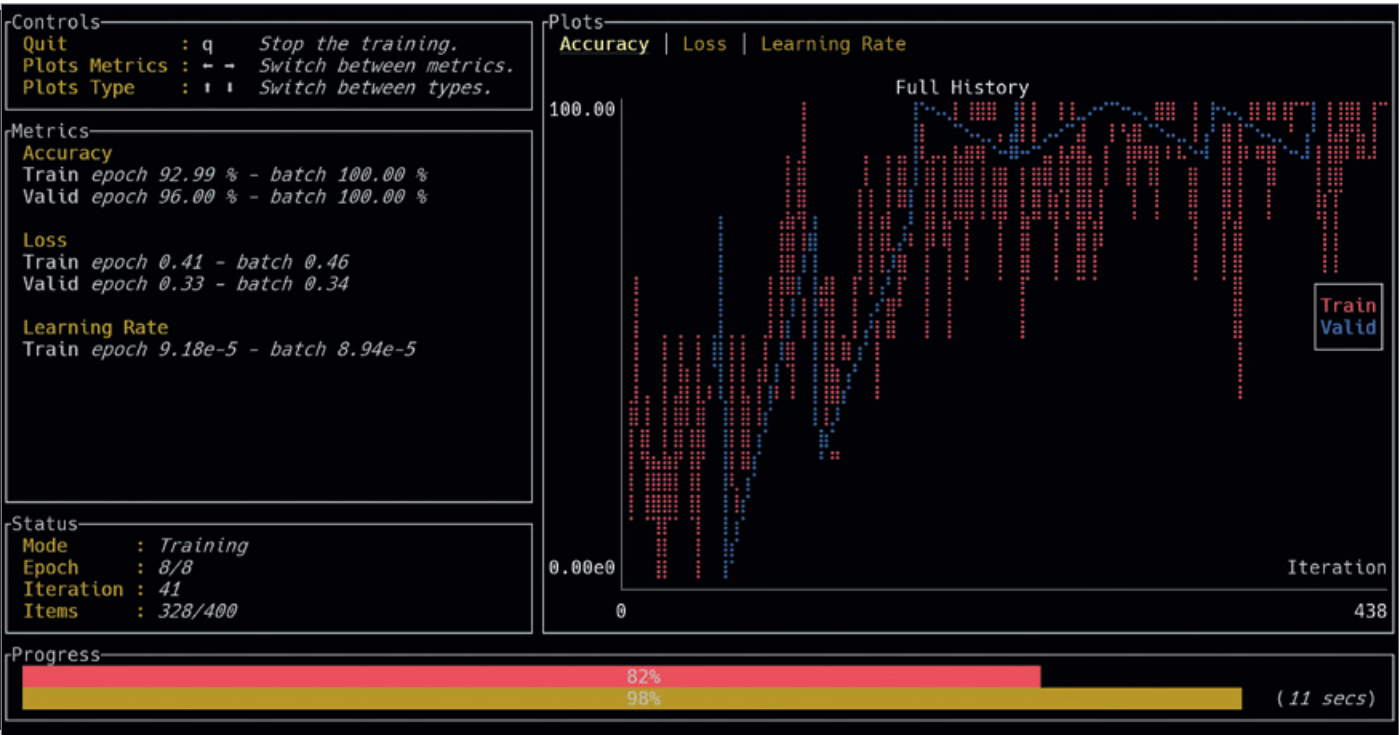


Abbildung 4: Darstellung des Trainingsfortschritts in Burn (Quelle: Andreas Maier)

 Modell

 Ändern

 Anlegen

Auswahl

DOAG_RED_STACK - Id. 3

▼

Allgemein

Name	trainierte Modelle aufheben	Typ	Id.	Status
DOAG_RED_STACK	5	Text-Klassifizierung-Transformer	3	

Config

```
{
  "pretrained_variant": "None",
  "precision": "F16",
  "data": "DOC",
  "field": "base.text",
  "scope": "element",
  "scope_part_element_max": 1,
  "batch_size": 8,
  "max_seq_length": 256,
  "num_epochs": 8,
  "vocab_size": 2000,
  "transformer": {
    "d_model": 32,
    "d_ff": 128,
    "n_heads": 4,
    "n_layers": 4,
    "dropout": 0.1,
    "class_weights": true,
    "norm_first": true,
    "quiet_softmax": true,
    "loss": {
      "smoothing": 0.02
    },
    "lr": {
      "learning_rate": 0.01,
      "warmup": 50
    },
    "optimizer": {
      "weight_decay": 0.00001
    }
  }
}
```

Text Preprocessor

```
{
  "rules": [
    {
      "pattern": "#NL#",
      "replace": "\n"
    },
    {
      "pattern": "\\s+",
      "replace": " "
    }
  ]
}
```

Beschreibung

Beispiel für eine Textklassifizierung mit einem eigenen Transformermodell

Abbildung 5: Beispiel einer Modellübersicht in APEX (Quelle: Andreas Maier)


```

use burn::prelude::*;
use burn::nn;

// Modelldefinition: zwei lineare Schichten mit ReLU-Aktivierung
#[derive(Module, Debug)]
pub struct ClassificationModel<B: Backend> {
    linear1: nn::Linear<B>,
    linear2: nn::Linear<B>,
    activation: nn::Relu,
}

impl<B: Backend> ClassificationModel<B> {
    // Verarbeitet die Eingabedaten zu einer Vorhersage
    pub fn forward(&self, x: Tensor<B, 2>) -> Tensor<B, 2> {
        let x = self.linear1.forward(x);
        let x = self.activation.forward(x);
        self.linear2.forward(x)
    }
}

```

Listing 1: Beispiel einer Modelldefinition in Rust/Burn (vereinfacht)

Verbindungen miteinander verbunden sind. Ein Neuron ist im einfachsten Fall eine Funktion, die ihre gewichteten Eingaben aufsummiert, eine Aktivierungsfunktion darauf anwendet und das Ergebnis an die nächste Schicht weitergibt. Moderne Architekturen wie Transformer ergänzen dieses Grundprinzip um komplexere Mechanismen – etwa Attention, bei der das Netz lernt, welche Teile der Eingabe füreinander relevant sind. Ein Gewicht ist dabei nichts anderes als eine Zahl – ein Modell mit Millionen von Verbindungen besteht letztlich aus Millionen von Zahlen, die während des Trainings so lange angepasst werden, bis das Netz die gewünschten Ergebnisse liefert. Die eigentliche „Intelligenz“ steckt in diesen Gewichten – und genau diese werden im Training gelernt.

Das Training folgt einem überraschend einfachen Grundprinzip: Lernen durch Beispiele und Korrektur. Das Netz erhält einen Input und berechnet auf Basis seiner aktuellen Gewichte eine Vorhersage. Anschließend wird die Vorhersage mit dem tatsächlichen, vorgegebenen Ergebnis verglichen. Die Abweichung wird über eine sogenannte Loss Function berechnet: Je größer der Wert, desto weiter liegt das Netz von der Vorgabe entfernt. Im nächsten Schritt – der Backpropagation – berechnet das Netz, welche Gewichte wie stark zum Fehler beigetragen haben, und zwar Schicht für Schicht zurück vom Output bis zum Input. Auf dieser Basis werden die Gewichte justiert, sodass die Abweichung beim nächsten Durchlauf kleiner ausfällt (siehe Abbildung 1).

In einem iterativen Kreislauf aus Vorhersagen und Korrekturen durchläuft das Netz den gesamten Trainingsdatensatz mehrfach. Jeden vollständigen Durchlauf bezeichnet man als Epoche. Mit jeder Epoche werden die Gewichte feiner justiert, bis das Netz die relevanten Muster zuverlässig erkennt. Das Prinzip ist dabei universell: Ob Textklassifizierung, Bilderkennung oder Prognosen – solange ein ausreichend großes Dataset vorliegt, lässt sich ein neuronales Netz auf praktisch jede Problemstellung trainieren. Auch große LLMs werden nach genau diesem Grundprinzip trainiert.

Die Architektur: Datenbank trifft neuronales Netz

Nachdem das theoretische Fundament gelegt ist, stellt sich die entscheidende Frage: Wie setzt man das Ganze praktisch um – und zwar so, dass es sich in eine bestehende Oracle-Landschaft einfügt? Am konkreten Beispiel einer Klassifizierung zeigt dieser Abschnitt eine Architektur aus drei Schichten, die jeweils ihre Stärken ausspielen: Oracle APEX als Benutzeroberfläche, PL/SQL als Bindeglied zwischen den Komponenten und Rust für das Training und die Ausführung der neuronalen Netze (siehe Abbildung 2).

Bevor man ein neuronales Netz trainieren kann, braucht es ein Dataset – Paare aus Input und gewünschtem Output. Dieses Dataset verwaltet man idealerweise direkt in der Oracle-

Datenbank, da sich Beispiele laufend ändern: Neue kommen hinzu, bestehende werden korrigiert oder aussortiert. Üblicherweise teilt man das Dataset in Trainings-, Validierungs- und Testdaten auf, um die Qualität des Modells an ungesehenen Beispielen prüfen zu können. Ebenso möchte man Trainingsmetriken wie Loss, Accuracy oder Learning Rate festhalten, um den Lernfortschritt nachvollziehen zu können. Dazu kommen noch weitere Einstellungen pro Modell und Training hinzu – etwa die Anzahl der Trainingsepochen. All das lässt sich mit einem einfachen Datenmodell aus wenigen Tabellen abbilden (siehe Abbildung 3).

Die Tabelle MODEL speichert die Basisdaten eines Modells – Name, Beschreibung und Konfigurationseinstellungen als JSON. In MODEL_CLASS werden die einzelnen Klassen definiert, die das Netz unterscheiden soll. Die zugehörigen Eingabebeispiele landen in MODEL_CLASS_ELEMENT. Für den Trainingsprozess gibt es drei weitere Tabellen: MODEL_TRAIN dokumentiert die allgemeinen Informationen zu einem Trainingslauf, MODEL_TRAIN_PRED speichert die Vorhersagen nach dem Training und MODEL_TRAIN_VAL protokolliert die Metriken im Trainingsverlauf.

Für dieses Beispiel kommt Rust mit dem Framework Burn zum Einsatz. Burn bietet alles, was man für den Aufbau und das Training neuronaler Netze braucht, und profitiert dabei von Rusts Stärken. Selbstverständlich ließe sich dieser Teil auch mit Python und Frameworks wie PyTorch oder TensorFlow umsetzen – die Architektur ist in dieser Hinsicht flexibel. Für die Anbindung an die Oracle-Datenbank stehen hauptsächlich drei Wege offen: die Crates rust-oracle über ODPI-C oder sibyl über das Oracle Call Interface (OCI) für eine native Verbindung, oder alternativ über REST-Aufrufe, die man über APEX (ORDS) bereitstellt.

Bereits wenige Zeilen Code genügen, um ein einfaches Modell in Burn zu definieren – die Struktur erinnert dabei stark an PyTorch (siehe Listing 1).

Für dieses Beispiel wurde ein Dataset mit 500 Textdokumenten künstlich erzeugt – 100 je Klasse, Adressänderung, Angebotsanforderung, Bankverbindung, Kündigung und Schadensmel-

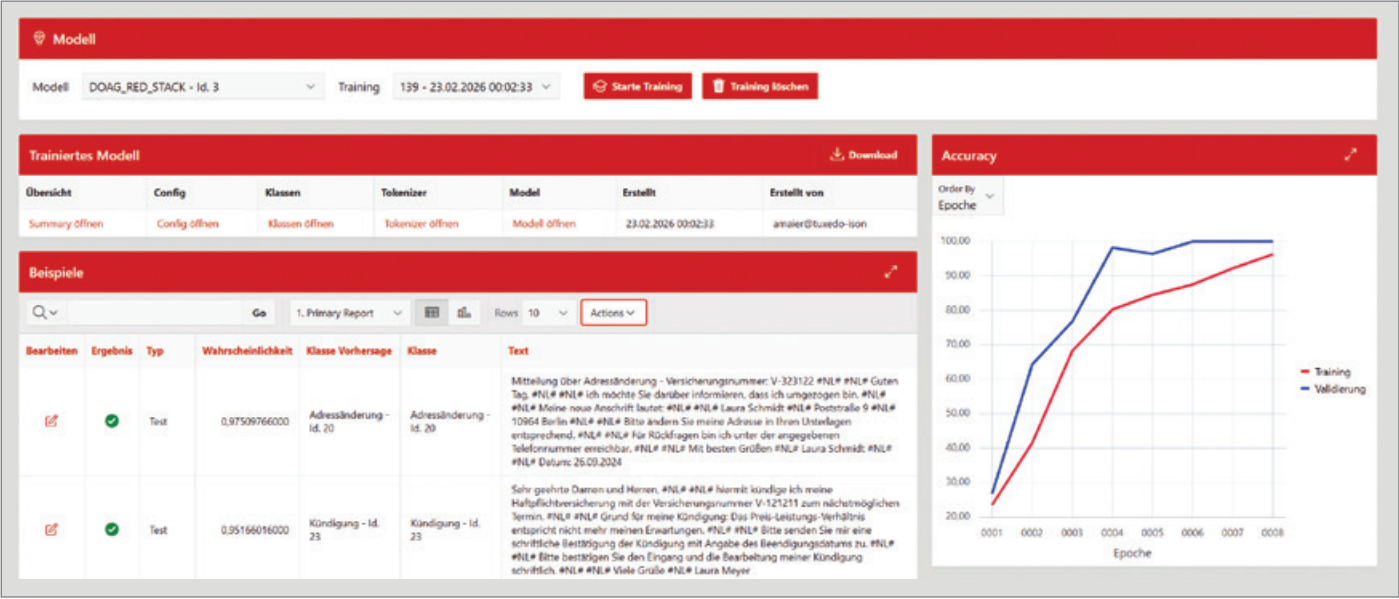


Abbildung 6: Ausschnitt aus der Trainingsübersicht (Quelle: Andreas Maier)

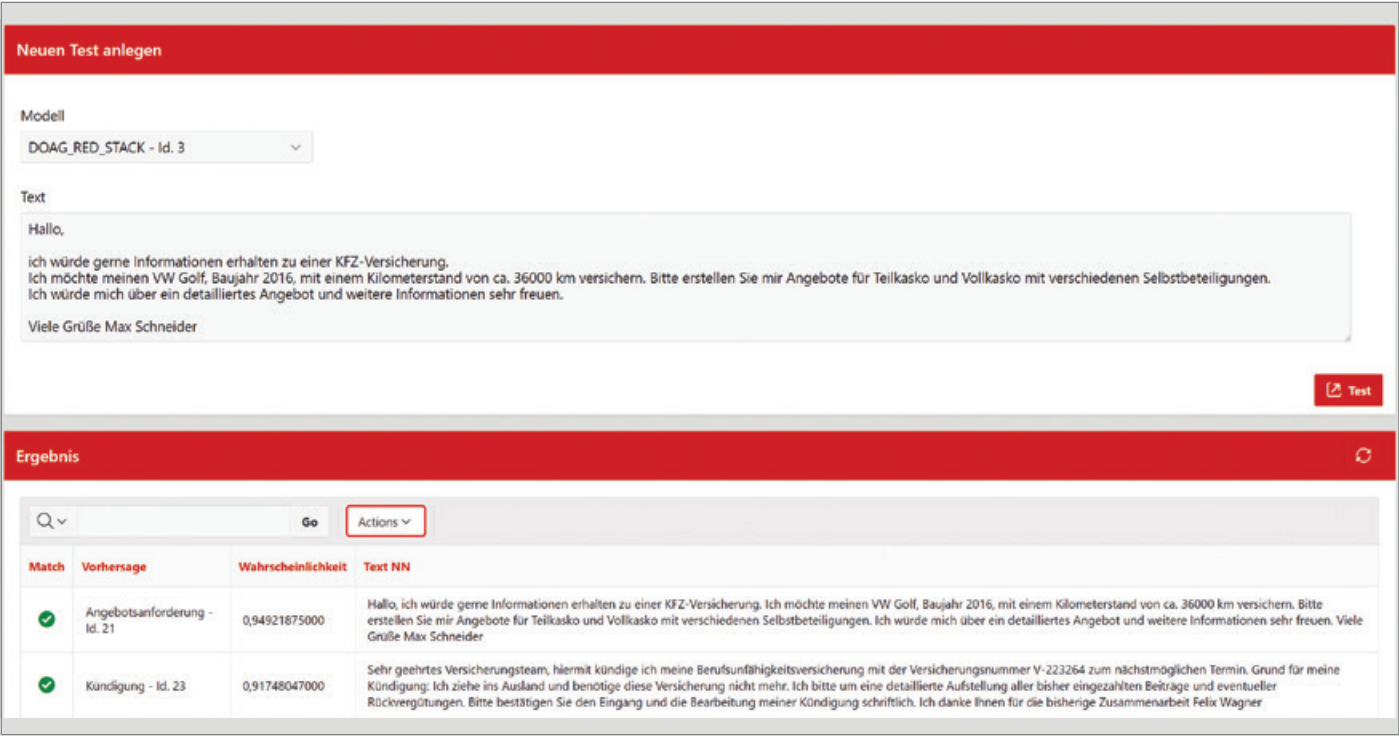


Abbildung 7: Modell direkt aus APEX testen (Quelle: Andreas Maier)

```
Hallo,  
  
ich würde gerne Informationen erhalten zu einer KFZ-Versicherung.  
Ich möchte meinen VW Golf, Baujahr 2016, mit einem Kilometerstand von ca. 36000 km versichern. Bitte erstellen Sie mir Angebote für Teilkasko und Vollkasko mit verschiedenen Selbstbeteiligungen.  
  
Ich würde mich über ein detailliertes Angebot und weitere Informationen sehr freuen.  
  
Viele Grüße  
Max Schneider
```

Listing 2: Beispiel Angebotsanforderung

dung. *Listing 2* zeigt ein beispielhafte Angebotsanforderung.

Um damit ein Modell zu trainieren, wird das Dataset aus der Oracle-Datenbank geladen und die Daten für das Training werden vorbereitet. Danach wird das Modell erstellt und trainiert. Burn bringt dabei von Haus aus eine übersichtliche Darstellung des Trainingsfortschritts mit. Nach Abschluss des Trainings werden die Ergebnisse – Vorhersagen und Metriken – zurück in die Oracle-Datenbank geschrieben (*siehe Abbildung 4*).

Das hier trainierte Transformermodell für die Textklassifizierung kam am Ende auf 123.237 gelernte Gewichte und eine Dateigröße von gerade einmal 0,25 MB. Ein solches Modell lässt sich problemlos auf jeder modernen Hardware betreiben. In der Realität sind die Aufgabenstellungen zwar deutlich komplexer und erfordern ein größer dimensioniertes Modell, aber selbst dann bewegen sich die Dateigrößen oft nur im zwei- bis dreistelligen MB-Bereich.

APEX als Steuerzentrale: Modelle verwalten, trainieren und nutzen

Das Training eines neuronalen Netzes ist kein einmaliger Vorgang. Datasets werden erweitert, Klassen kommen hinzu und Modelle werden neu trainiert. Wer diesen Prozess ausschließlich über Skripte und Kommandozeile steuert, verliert schnell den Überblick. Genau hier spielt Oracle APEX seine Stärken aus: In kürzester Zeit wird eine vollwertige Oberfläche erstellt, die den gesamten Lebenszyklus eines Modells abbildet – vom Dataset über das Training bis zur produktiven Nutzung.

Zuerst braucht man eine Möglichkeit, Modelle und deren Klassen anzulegen und zu verwalten. Die zentrale Einstiegsseite der Anwendung zeigt eine Übersicht aller angelegten Modelle (*siehe Abbildung 5*). Ein Modell wird über ein einfaches Formular angelegt – Name, Beschreibung und die Konfigurationsparameter. Innerhalb eines Modells werden die Klassen definiert, die das Netz später unterscheiden soll.

Weitaus wichtiger als die Modellverwaltung selbst ist die Anlage und Pflege der Trainingsbeispiele. Im Idealfall lie-

gen bereits historische Daten vor, in diesem Beispiel einer Textklassifizierung beispielsweise, weil Sachbearbeiter eingehende Dokumente über Jahre hinweg manuell kategorisiert haben. Das ist ein guter Ausgangspunkt. In der Praxis zeigt sich schnell, dass solche Daten oft nicht einheitlich gelabelt sind. Verschiedene Mitarbeiter interpretieren dieselbe Kategorie unterschiedlich, Zuordnungen ändern sich im Laufe der Zeit, und Grenzfälle werden mal so, mal so entschieden. Ein neuronales Netz lernt diese Inkonsistenzen gnadenlos mit. Deshalb führt oft kein Weg daran vorbei, die Trainingsbeispiele manuell zu prüfen und bei Bedarf zu korrigieren. Genau hier lohnt es sich, die APEX-Oberfläche auf maximale Effizienz für diese Aufgabe zu optimieren. Statt eines generischen Formulars baut man eine für die Aufgabe optimierte Seite. Ein Klick ordnet das Beispiel zu und lädt sofort das nächste. So entsteht ein flüssiger Arbeitsrhythmus, bei dem ein geübter Fachexperte in kurzer Zeit große Mengen an Beispielen durcharbeiten kann.

Die Trainingsübersicht zeigt neben den allgemeinen Kennzahlen wie Accuracy und Loss, die Ergebnisse der einzelnen Beispiele. Anhand der Vorhersagen lässt sich erkennen, bei welchen Klassen das Netz sicher entscheidet und wo es unsicher ist. Daraus ergeben sich sinnvolle Schwellwerte für den produktiven Einsatz: etwa ab welcher Konfidenz eine automatische Zuordnung erfolgen soll und wann besser ein Sachbearbeiter entscheidet. Ebenso wird sichtbar, ob die Trainingsparameter angepasst werden müssen oder ob bestimmte Klassen mehr Beispiele brauchen (*siehe Abbildung 6*).

Ein weiterer wichtiger Baustein ist eine Testmöglichkeit, über die sich ein trainiertes Modell direkt ausprobieren lässt. In unserem Beispiel gibt man den gewünschten Text ein, wählt das Modell und erhält sofort die Vorhersage vom Modell. Das klingt simpel, ist aber im Alltag unverzichtbar: Wenn das Modell im produktiven Einsatz unerwartet reagiert, braucht man eine schnelle Möglichkeit zum Debugging (*siehe Abbildung 7*).

Fazit

Nicht jede Aufgabe lässt sich sinnvoll mit einem LLM lösen. Wer eine spezifische

Klassifizierung braucht – zugeschnitten auf die eigenen Daten, die eigene Fachlichkeit und die eigenen Kategorien – fährt mit einem selbst trainierten Modell oft besser. Es ist schneller, günstiger im Betrieb, läuft ohne externe Abhängigkeiten und liefert für die konkrete Aufgabe häufig präzisere Ergebnisse. Dabei ist der Einstieg weniger kompliziert als viele vermuten. Etwas Grundlagenwissen im Bereich Machine Learning sollte man sich allerdings aneignen: wie man beispielsweise vortrainierte Modelle sinnvoll einsetzt, woran man Overfitting erkennt oder wie die Wahl der Hyperparameter das Ergebnis beeinflusst. Dieses Wissen zahlt sich schnell aus und schützt vor typischen Anfängerfehlern. Die größte Hürde ist am Ende nicht die Technik, sondern die Qualität der Trainingsdaten – und genau dabei hilft eine gut gemachte Oberfläche enorm. Wer das Thema bisher vor sich hergeschoben hat: Einfach mal ausprobieren.

Über den Autor

Andreas Maier entwickelt seit über 20 Jahren Softwarelösungen im Oracle-Umfeld – von SQL, PL/SQL, Forms und APEX bis hin zur Architekturberatung. Seit 2016 beschäftigt er sich intensiv mit Künstlicher Intelligenz und setzt Machine Learning vor allem zur Automatisierung von Geschäftsprozessen ein, mit Schwerpunkt auf der Dokumentenverarbeitung. Mit der Gründung der isOn Tec GmbH im Jahr 2024 bündelt er diese Expertise, um seinen Kunden maßgeschneiderte Lösungen anzubieten.



Andreas Maier
andreas.maier@isontec.com


```

var n, i = 0;
o = e.length;
a = M(e);
if (n) {
  if (a) {
    for (; o > i; i++)
      if (r = t.apply(e[i], n), r === !1) break
  } else
    for (i in e)
      if (r = t.apply(e[i], n), r === !1) break
} else if (a) {
  for (; o > i; i++)
    if (r = t.call(e[i], i, e[i]), r === !1) break
} else
  for (i in e)
    if (r = t.call(e[i], i, e[i]), r === !1) break;
return e
},
trim: b && !b.call("\uffeff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e)
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "")
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string" == typeof e ? [e] : e) : b.call(n, e)),
},
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, n = n ? 0 > n ? Math.max(0, r + n) : n : 0; r > n; n++)
      if (n in t && t[n] === e) return n
  }
}

```

APEX im Git, Klartext im Code – CI/CD mit APEX und ein Hauch Intelligenz

Sonja Meyer, Martin Bach, Oracle

APEX-Entwicklung ist seit jeher Low-Code orientiert, um jedem die Möglichkeit zu geben, ohne großen Aufwand elegante Anwendungen zu schreiben. APEX möchte gleichzeitig aber auch die Power-User besser unterstützen, deren primäre Entwicklungsumgebung eine IDE ist, sowie Git und CI/CD eine Selbstverständlichkeit darstellen. In diesem Artikel stellen wir vor, wie man mit SQLcl und dessen Projects Workflow APEX-Anwendungsentwicklung auf die nächste Stufe stellt

CI/CD mit APEX ist längst Realität seit SQL Developer Command Line, kurz SQLcl, die Projects Workflow eingeführt hat. Mit ihm bekommen Entwickler die Möglichkeit und den Spaß, einen strukturierten, wohldefinierten Prozess zu nutzen, um damit effektiv Continuous Integration Pipelines zu erstellen und nutzen. SQLcl

kann außerdem KI-Modellen Zugriff auf die Datenbank gewähren, womit sich völlig neue Möglichkeiten, Anwendungen zu entwickeln, eröffnen. In diesem Artikel beschreiben wir, wie man KI in den Entwicklungsprozess einbinden kann, KI mit GitHub interagiert, und funktionale Anwendungen entstehen.

Continuous Integration/Delivery mit APEX

Kontinuierliche Integration (CI) ist ein zentrales Architekturprinzip moderner Softwareentwicklung, das darauf abzielt, Integrationsrisiken systematisch zu minimieren. Statt Komponenten erst am Ende

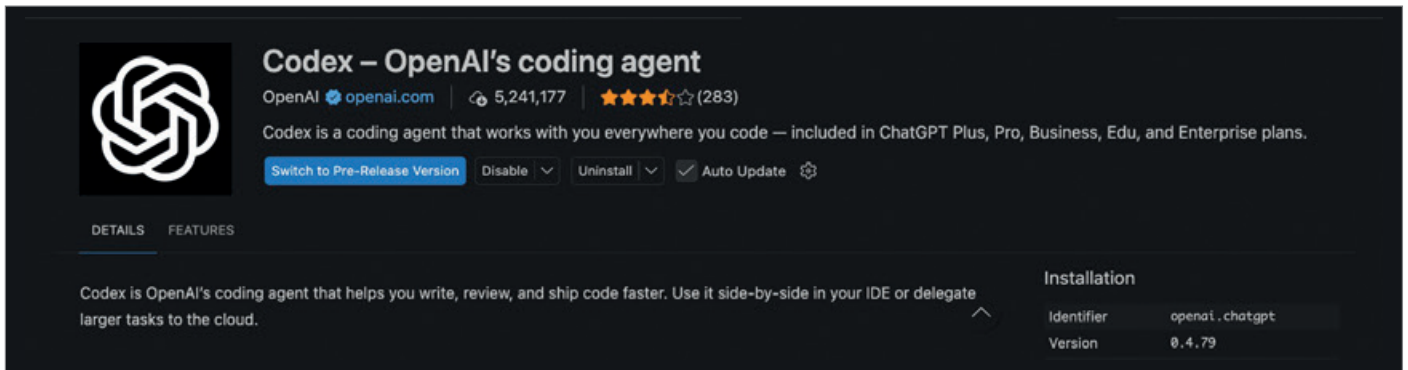


Abbildung 1: Codex – OpenAI's coding agent (Quelle: Martin Bach)

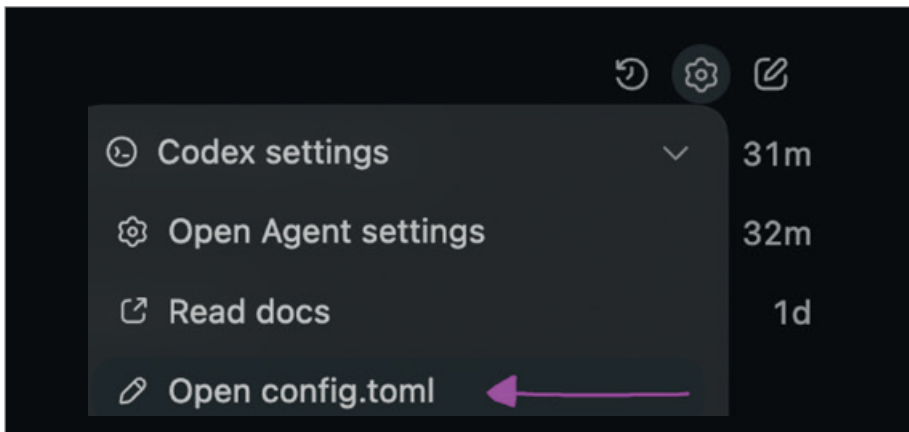


Abbildung 2: Im Codex auf die Einstellungen navigieren (Quelle: Martin Bach)

einer Entwicklungsphase zusammenzuführen, werden Änderungen kontinuierlich in den gemeinsamen Codebestand integriert. Bei jeder Codeänderung – sei es ein Bugfix, eine neue Funktion oder eine Anpassung – wird automatisch ein Build-Prozess angestoßen. Dieser führt Tests durch, bewertet die Codequalität und erstellt relevante Metriken. Integration wird so von einem einmaligen Ereignis zu einem fortwährenden Zustand.

Technisch betrachtet ist CI eine automatisierte Feedback-Maschine. Ein Build-Server übernimmt dabei mehrere Aufgaben: Er deployt Backend-Code, kompiliert oder paketierte Artefakte, löst statische Code-Analysen aus, misst die Testabdeckung (Coverage), bewertet Komplexitätsmetriken und führt sowohl Unit- als auch Integrationstests durch. Ziel ist nicht nur Funktionsfähigkeit, sondern messbare Softwarequalität. Der Effekt: Die Integrationslatenz sinkt drastisch, Fehler werden im Kontext ihrer Entstehung erkannt, und die berüchtigte „Integration Hell“ wird durch inkrementelle Validierung ersetzt.

Im Fall von **Oracle APEX** ist die Anwendung technisch gesehen eine Sammlung

von Metadaten, die in der Datenbank gespeichert werden. In diesem Kontext bedeutet Continuous Integration, diese Metadaten exportierbar, differenzierbar (diff-bar) und automatisiert deploybar zu machen. Anstelle von „ClickOps“ im Builder werden momentan Applikationen über Skripte, etwa mit SQLcl oder APEX-Export-Utilities, in einzelne Komponenten zerlegt, im Git-Repository versioniert und durch eine Pipeline verarbeitet.

Eine typische APEX-CI-Pipeline umfasst:

- automatisierten Export der Anwendung als strukturiertes Artefakt (z. B. split files)
- Validierung durch statische Checks auf Naming Conventions oder unerwünschte Hardcodings
- Ausführung von Unit-Tests auf PL/SQL-Ebene (z. B. mit utPLSQL)
- Deployment in eine isolierte Build- oder Test-Datenbank
- optional UI-Tests über Selenium oder Playwright gegen die laufende Instanz

Damit verschiebt sich der Qualitätsfokus: Nicht nur die Applikation selbst wird ge-

testet, sondern auch ihre Reproduzierbarkeit und Installierbarkeit. Ein Build gilt nur dann als erfolgreich, wenn die App in einer frischen Datenbank ohne manuelle Eingriffe lauffähig ist. Das ist für APEX essenziell, weil Deployments letztlich SQL-Skripte sind, die deterministisch funktionieren müssen. Der Nerd-Faktor liegt darin, APEX als Code zu behandeln.

SQLcl Projects Workflow

CI/CD ist vergleichsweise einfach, wenn es „nur“ darum geht, Artefakte wie JAR-Dateien in der Java Welt zu erstellen. Stateless Applications können einfach aktualisiert werden und ein Rollback ist meist ebenso unkompliziert. Mit Datenbanken ändert sich das Szenario, Rollback wird komplexer, denn niemand möchte Datenverlust in Kauf nehmen. Daher ist ein besonderer Umgang mit dem Datenbankschema notwendig, weil Schemaänderungen persistent sowie zustandsbehaftet sind, und sie betreffen produktive Daten. SQLcl Projects ist ein Framework mit dem CI/CD für datengestützte Anwendungen möglich wird. Der Schwerpunkt liegt insbesondere auf Git als Single Source of Truth und nicht der DEV-Datenbank. Damit eignet sich der SQLcl Projects Workflow hervorragend für CI/CD.

Model Context Protocol

Seit einigen Jahren nutzen Software-Entwickler KI, um produktiver zu sein und ihre Ziele schneller zu erreichen. Am Anfang der Entwicklung interagierte man mit isolierten Chatbots, die aber keine Verbindung zum Softwareprojekt hat-

ten. Sie standen nur erklärend zur Seite. Im nächsten Evolutionsschritt, dem der Agentic AI, wurden KI-Assistenten in Entwicklungswerkzeuge wie VS Code integriert, um diesen direkteren Zugriff auf die Codebasis zu ermöglichen. Nun hatten die KI-Modelle Zugriff auf den Code, und sie konnten zudem Aktionen damit durchführen. Dazu zählen insbesondere Veränderungen von Dateien, Modifikation von Code und das Erstellen von Unit Tests. Auf externe Systeme, wie zum Beispiel die Oracle-Datenbank, konnten sie nicht ohne weiteres zugreifen.

Dies änderte sich schlagartig mit der Erfindung des Model Context Protocol (MCP). MCP fungiert als standardisier-

te Brücke zwischen KI-Modell und Code und erlaubt es dem Agenten, mit externen System wie zum Beispiel Datenbanken, APIs oder anderen Laufzeitumgebungen zu interagieren. Der beliebte und voll integrierte SQLcl wurde um das Model Context Protocol erweitert. Das sollte ein weiterer Grund sein, sich das Tool anzuschauen und von SQLPlus zu wechseln.

Packungsbeilage zur KI-gestützten Entwicklung

Hat man sich erst einmal mit dem Thema KI und Softwareentwicklung beschäf-

tigt, wird sehr schnell klar, dass mit großer Macht auch große Verantwortung kommt.

Agentic AI kann, sofern man keine Schutzmechanismen vorsieht, nicht nur positives bewirken. Es gelten wie bisher auch die Regeln der IT-Sicherheit, wie zum Beispiel die geringstmögliche Anzahl an **Privilegien** zu vergeben, sich alle Schritte, die die KI vorschlägt, bestätigen zu lassen (auto-approval aus) und bei sensiblen Daten erstmal mit der IT Security zu sprechen, um keine Daten unbeabsichtigt abfließen zu lassen. Es ist auch ratsam, nie den gesamten Code in den Prompt einzugeben, und man sollte sich auch vor **Prompt Injection** schützen.

```

SOMEYER@SOMEYER-mac ~ % codex

>_ OpenAI Codex (v0.115.0)

model:      oca/gpt-5-codex medium  /model to change
directory:  ~

Tip: New Build faster with the Codex App. Run 'codex app' or visit
https://chatgpt.com/codex?app-landing-page=true

```

Abbildung 3: Start von Codex (Quelle: Martin Bach)

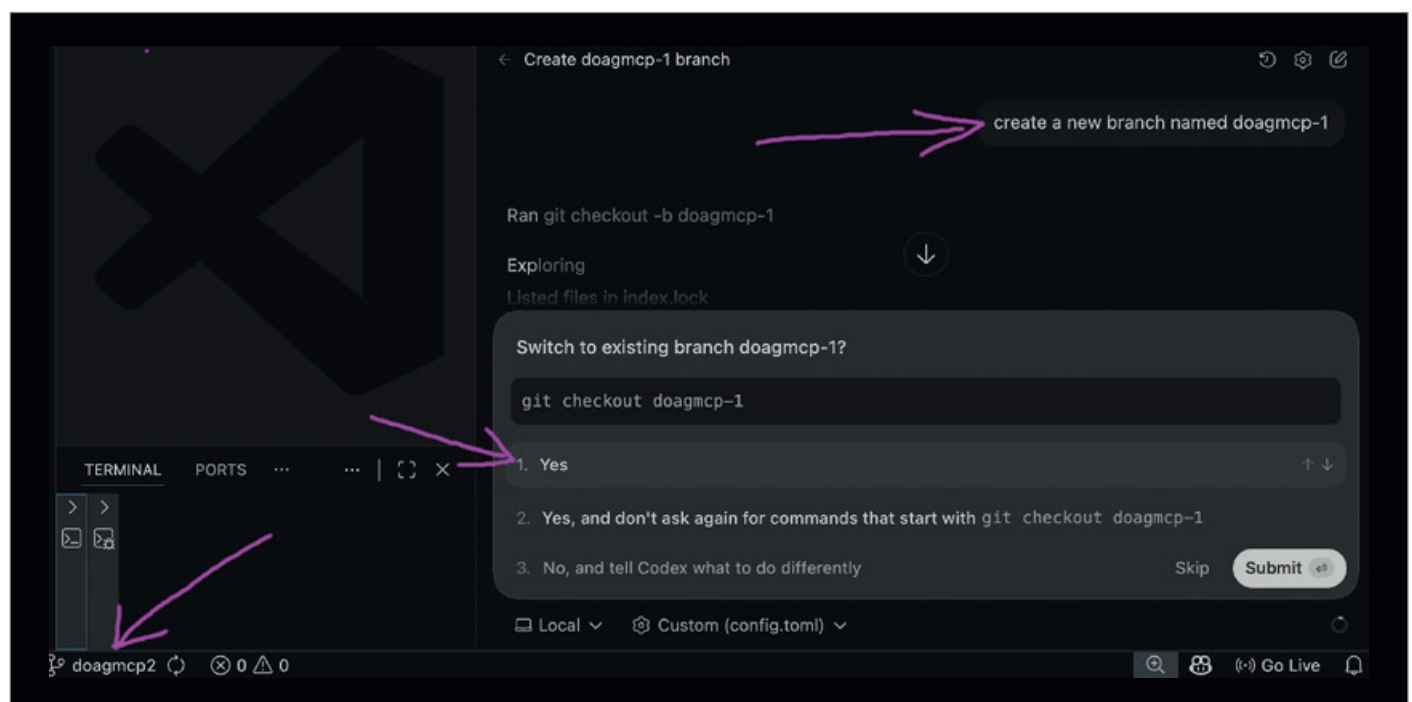


Abbildung 4: Codex-Prompt für die Erstellung eines neuen Branchs zur Entwicklung (Quelle: Martin Bach)

Natürlich gibt es zum Thema Sicherheit noch mehr zu sagen, aber dieser Absatz soll lediglich zum Nachdenken anregen. Fragen Sie ihren KI- oder CISO-Experten und lesen sie die Packungsbeilage (sprich Readme) des MCP-Servers ihres Vertrauens.

Die erste CI-Pipeline

Nach all der Theorie zu Beginn soll es jetzt praktischer werden. Die Autoren verwendeten Visual Studio Code mit dem Codex Plugin. Codex ist die Schnittstelle zwischen dem KI-Modell, dem Anwender und dem MCP-Server (siehe Abbildung 1).

Tip: Leiste in VS Code per Drag & Drop auf die rechte Seite ziehen.

Als MCP-Server kommen sowohl der offizielle GitHub-MCP-Server als auch Oracle-SQLcl-Server zum Einsatz. Darüber hinaus wird die SQL Developer Extension für VS Code genutzt; das aktuelle SQLcl wird bereits gemeinsam mit dieser Extension ausgeliefert und installiert.

Als erstes erstelle ein neues Repository auf GitHub und klonen es mit VS Code in ein lokales Verzeichnis. Falls noch nicht geschehen, öffne den Ordner, damit Codex auf das Verzeichnis zugreifen kann. Codex muss nun von den MCP-Servern in Kenntnis gesetzt werden. Dazu kann man die Konfigurationsdatei, *config.toml*, in den Einstellungen öffnen und erstellen (siehe Abbildung 2).

Die MCP-Server können in Codex wie folgt definiert werden. Die Links zur jeweiligen Dokumentation finden sich im Appendix [1][2]:

```
[mcp_servers.sqlcl]
command = "$HOME/.vscode/extensions/oracle.sql-developer-25.4.0/dbtools/sqlcl/bin"
args = ["-mcp"]
[mcp_servers.github]
url = "https://api.githubcopilot.com/mcp/"
bearer_token_env_var = "GITHUB_PAT_TOKEN"
```

Für den GitHub MCP-Server benötigst Du ein GitHub Personal Access Token [3]. Dieses sollte in einer .env-Datei im

obersten Projektverzeichnis in folgender Form abgelegt werden:

```
GITHUB_PAT_TOKEN="<Personal Access Token>"
```

Natürlich sollten nur die absolut notwendigsten Rechte in dem Access Token vergeben sein. Beim Start von Codex sollten dann keine Probleme mehr auftreten, wie in diesem Screenshot gezeigt wird (siehe Abbildung 3).

Um den SQLcl Project Workflow zu starten, muss dein Projekt erst einmal initialisiert werden. Dazu benötigt man eine Datenbankverbindung, die man in SQL Developer for VS Code erstellt. Wichtig ist es dabei, das Passwort mitzuspeichern. Es empfiehlt sich, die ersten Experimente erstmal mit einer Spielzeugdatenbank wie Oracle AI Database Free [4] zu machen, um sich dem Thema zu nähern. Nachdem die Verbindung gespeichert wurde, kann nun SQLcl gestartet werden. Das geht am einfachsten, indem man einen Rechtsklick auf den Namen der Verbindung ausführt und „Start SQLcl“ auswählt. Da SQLcl ein anderes Arbeitsverzeichnis beim Start verwendet, sollte nun mittels `cd` in das Projektverzeichnis gewechselt werden. Das Kommando zum Erstellen des Projekts lautet wie folgt:

```
SQL> project init -name doagmcp
-directory . -schemas <dein DB Schema>
```

Das `init`-Kommando erstellt, nach erfolgreicher Ausführung, die für die weiteren Schritte notwendige Verzeichnisstruktur. Die erstellten Dateien müssen als nächstes in Git committet werden. Entweder in VS Code selbst oder noch in SQLcl:

```
SQL> ! git add .
SQL> ! git commit -m "chore: project initialisation"
```

Als nächstes kann dann die KI zum Zuge kommen. Das könnte man auch für die Schritte vorher ausprobieren, aber den `init` machen wir gerne unter kontrollierter Aufsicht. Nun bitten wir Codex einen neuen Branch für die Entwicklung zu erstellen (siehe Abbildung 4).

Der neue Branch ist deswegen notwendig, da SQLcl Projects Änderungen im aktuellen Entwicklungsbranch mit `main`,

dort wo sich typischerweise der stabile Produktionscode befindet, vergleicht und darauf basierend die Schema-Migration durchführt. Damit folgt SQLcl Projects den etablierten Entwicklungsprozessen.

Damit aber überhaupt etwas migriert werden kann, benötigen wir als erstes eine Anwendung. Diese soll Oracle User Groups in einer Tabelle hinterlegen und alle Community Events 2026 in einer weiteren. Dabei hilft uns Codex wieder, indem wir zuerst ein Datenmodell erstellen, inklusive echten Beispieldaten::

„Verbinde dich mit der Datenbank. Nutze dazu die Verbindung <Name der eben erstellten Datenbankverbindung>. Erstelle dann direkt in der Datenbank ein Datenmodell mit Oracle Community Gruppen aus Europa als Haupttabelle. Stelle sicher, dass die Webseite und das Land hinterlegt sind. Alle Events aus 2026 sollen in einer Child-Tabelle dazu gespeichert werden. Bitte kreiere auch direkt reale Events der realen Usergruppen.“

Wichtig an dieser Stelle ist nochmal zu erwähnen, dass eine genaue Spezifikation sehr wichtig ist, um ein optimales Ergebnis zu erreichen, was auch schon einmal ein paar Iterationen bedeuten kann. Nachdem nun zwei Tabellen in der Datenbank erstellt worden sind, kann die APEX-Anwendung darauf aufsetzen. Unter anderem wollen wir Unit-Tests einführen, damit wir Regressionen im Code besser vermeiden können.

Unit-Tests lassen sich in APEX auf unterschiedliche Weise integrieren. Wichtig ist dabei nicht, die APEX-Bordmittel zu ersetzen, sondern die fachliche Logik sauber testbar zu kapseln.

Eine bewährte Variante besteht darin, Geschäftslogik konsequent in PL/SQL-Packages auszulagern, während APEX weiterhin für UI und deklarative Prozesse zuständig bleibt. Dieses Vorgehen eignet sich besonders für fortgeschrittene Entwickler, die Wert auf klare Architektur und Trennung von Zuständigkeiten legen. Die gute Nachricht: KI-gestützte Entwicklungswerkzeuge erleichtern heute das Erstellen solcher PL/SQL-Packages erheblich – inklusive der Generierung passender Unit-Tests. Unabhängig vom gewählten Ansatz gilt: Unit-Tests sind der effektivste Schutz vor unbemerkten Regressionen und damit ein essenzieller Bestandteil jeder CI/CD-Strategie. Sie

sorgen dafür, dass funktionale Änderungen reproduzierbar überprüft werden können – bevor sie produktive Daten berühren:

„Erstelle basierend auf utPLSQL ein Unit-Test-Package in PL/SQL, welches die DML-Operationen auf die OCG_GROUPS_EU-Tabelle simuliert.“

Codex wird den SQLcl MCP-Server nutzen, um das Package zu erstellen. Sobald es funktioniert, wie es soll, ist die APEX-Anwendung komplett. Es ist Zeit, diese in Git zu speichern. Dazu gibt es in SQLcl Projects das `export`-Kommando.

```
SQL> project export -verbose
```

Dieses exportiert nun das Datenmodell (Tabelle, Indices, PL/SQL, APEX-Anwendung ...). Wichtig ist, als nächstes diese Änderungen in Git per MCP-Server zu committen:

„Stage alle neu erzeugten und geänderten Dateien im Git-Repository und erstelle anschließend einen Commit mit der Commit-Message ‚feat: initial project export‘.“

Nachdem das erledigt ist, kann dann das Release erzeugt werden. Dazu werden die folgenden Kommandos verwendet:

```
SSQL> project stage -verbose
SQL> project release -version 1.0.0
```

Sobald letzteres abgeschlossen ist, folgt das bekannte Schema: Änderungen in Git committen und mittels `git-push` auf GitHub hochladen. Jetzt muss „nur“ noch die CI-Pipeline definiert werden. Da dein Projekt im Moment auf GitHub gehostet ist, macht es Sinn, GitHub Actions zum Einsatz kommen zu lassen. Codex kann auch bei dieser Aufgabe sehr gute Dienste leisten, allerdings ist es aufgrund der Komplexität der Aufgabe fast unmöglich, diese CI-Pipeline in einem Prompt zu definieren. Hier ist ein iterativer Ansatz gefragt, bis das Ergebnis erreicht ist. Alle Prompts hier aufzuführen, würde den Rahmen des Artikels noch weiter sprengen. Nachdem die Pipeline definiert ist, wird sie versioniert und in das Remote-Repository übertragen. Läuft sie erfolgreich durch, folgt der letzte

Schritt: die Überführung der Änderungen in den Main Branch – eine Aufgabe, die der GitHub-MCP-Server zuverlässig unterstützt.

Versioniere die aktuellen Änderungen im Repository <Name des Repository>, pushe den aktuellen Branch in das Remote-Repository und erstelle anschließend einen Pull-Request, der diesen Branch nach main merget.

Nachdem die CI-Pipeline ein weiteres Mal erfolgreich durchgelaufen ist, steht dem Merge in Main nichts mehr im Wege.

Herzlichen Glückwunsch!

Und mit APEXlang steht bereits das nächste Kapitel in den Startlöchern – eine Git-native Repräsentation von APEX, die Reviews, Diffs sowie Automatisierung auf ein neues Niveau hebt, und die wir uns im nächsten Artikel genauer ansehen werden.

Quellen

- [1] <https://docs.oracle.com/en/database/oracle/sql-developer-command-line/25.4/sqlcug/using-oracle-sqlcl-mcp-server.html>
- [2] <https://github.com/github/github-mcp-server/blob/main/docs/installation-guides/install-codex.md>
- [3] <https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/managing-your-personal-access-tokens>
- [4] <https://www.oracle.com/database/free/>

Über die Autoren

Sonja Meyer ist seit 2009 bei Oracle und jetzt im APEX Software Development Team.

Der Schwerpunkt ihrer Arbeit liegt momentan auf APEXlang und MLE. „KI-unterstützte Entwicklung ist für mich ein sehr aufregendes Thema und das nicht nur, wenn es läuft, sondern auch zu betrachten, warum Prompt Engineering so wichtig ist, und welche Sicherheitslücken sich auftun in dieser schnellwachsenden Umgebung.“

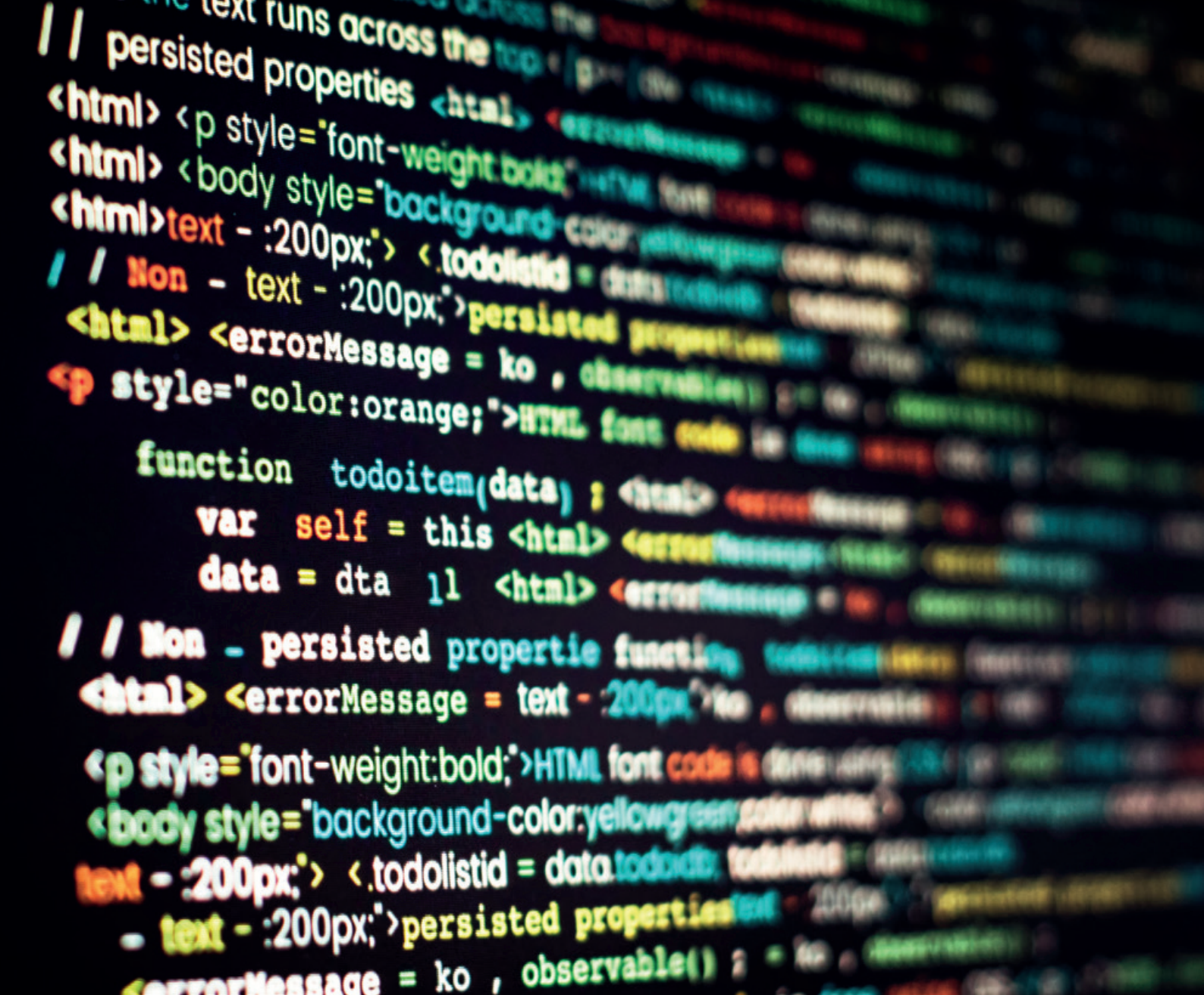
Martin Bach ist seit 2021 bei Oracle. Sein Schwerpunktthema ist Multilingual Engine, das heißt serverseitiges JavaScript in der Datenbank. Zudem interessiert er sich für alles, was mit DevOps, CI/CD und Platform Engineering zusammenhängt, Cloud Native im Namen trägt und irgendwie mit der Oracle-Datenbank kommuniziert. Neuerdings untersucht er auch KI-gestützte Entwicklung und Workflows.



Sonja Meyer
sonja.meyer@oracle.com



Martin Bach
martin.b.bach@oracle.com



Nicht nachmachen! Anti-Patterns in PL/SQL und APEX

Andreas Wismann, WHEN OTHERS

Wie haben Sie eigentlich PL/SQL gelernt? An der Universität, aus Büchern, durch die Oracle-Dokumentation oder im Internet? Für viele von uns folgte nach einer ersten Lernphase jedenfalls der Sprung ins kalte Wasser: das Schreiben von produktivem Code für ein Unternehmen. Und ganz ehrlich, meist haben wir nicht bei NULL angefangen, sondern uns an dem, was bereits vorhanden war, orientiert. Beim Einstieg in ein neues Unternehmen analysieren wir den vorhandenen Code, um ein Grundverständnis zu entwickeln und Lösungsansätze für anstehende Aufgaben zu schaffen.

Diese Medaille hat zwei Seiten, eigentlich sogar drei. Einerseits weicht unser neuer Code nicht wesentlich von der vorgefundenen Norm ab, was gut ist. Andererseits übernehmen wir damit die vorhandenen Entwurfsmuster, ohne sie zu verbessern oder zu modernisieren. Obendrein trägt man sogar Anti-Patterns weiter: Code, der zwar seit vielen Jahren gut funktioniert, aber bei genauerem Hinsehen weit entfernt von optimaler Implementierung ist. Vergleichen Sie nur die technischen Konzepte der Oracle-Datenbank von vor 30 Jahren mit den heutigen. Die immer noch gültigen Klassiker der PL/SQL-Literatur stammen aus dem ersten Jahrzehnt der 2000er Jahre. Das ist, in IT-Maßstäben gerechnet, schon eine halbe Ewigkeit her. Selbst Teile der Sprache PL/SQL haben sich inzwischen weiterentwickelt. Nicht so klar ersichtlich wie beispielsweise bei Java (im Jahr 2000: Java 1.3, im Jahr 2025: Java 25), sondern eher unbemerkt und leider häufig ungenutzt.

Noch intensiver hat sich Oracle APEX seit den 2000ern entwickelt, und das erkennen wir mindestens einmal jährlich an der Versionsnummer sowie dem Umfang der Dokumentation!

Dieser Artikel ist kein Update, um Entwickler auf den neuesten Stand zu bringen. Er möchte Sie vielmehr mit acht Fallbeispielen dazu inspirieren, regelmäßig Ihre Programmierkonzepte aufzufrischen.

Öffentlich oder privat?

Man stand früher bei der Entwicklung von PL/SQL-Packages vor einem prinzipiellen Dilemma: Prozeduren und Funktionen im Package Body, die eigentlich hätten privat bleiben sollen, mussten jedoch wenigstens zum Zweck des Testens in der Package Specification deklariert werden. Dadurch wurden jedoch alle Funktionalitäten ungewollt publiziert, es sei denn, man machte diese Veröffentlichung später wieder rückgängig (was neue Probleme mit sich bringen konnte, siehe Forward Declaration). Ungeachtet der Möglichkeit, Berechtigungsstrukturen grob über Schema-Grenzen zu verwalten, musste man sich also stets zwischen permanenter Testbarkeit oder Kapselung entscheiden. Oft wurde dann zu wenig getestet... nicht nachmachen!

Erstaunlich spät, nämlich seit der Datenbankversion 12c, bietet Oracle dafür eine Lösung: Sämtliche (eigentlich private) Procedure- und Function-Deklarationen kann man bedenkenlos in die Package Specification übernehmen und sie dennoch mit der Klausel `ACCESSIBLE BY` zur Laufzeit kapseln (siehe *Listing 1*).

Im Codebeispiel wird die Prozedur `DEMO_1_PCK.private_routine` so definiert, dass sie nur vom Package `UT_DEMO_1_PCK` aufgerufen werden kann. Das ist in diesem Beispiel ein Unit-Test-Package (in vielen Unternehmen typischerweise `utPLSQL`), welches noch

nicht existieren muss. Diese expliziten Zugriffsberechtigungen lassen sich auch über Schema-Grenzen hinweg vergeben. Das führt nebenbei zu einem nicht zu unterschätzenden Dokumentationsaspekt: Man weiß stets, für wen oder was diese Routine geschrieben wurde. Keine späteren Änderungen sind mehr nötig, trotzdem bleibt privat, was privat sein soll. Einen Verstoß gegen die obige Beschränkung quittiert die Datenbank mit der Fehlermeldung

```
PLS-00904: insufficient privilege
to access object PRIVATE_ROUTINE.
```

Kann es wahr sein?

Erst ab der Datenbankversionsnummer 23 ist es möglich, den trivialsten aller Datentypen, `BOOLEAN`, nativ in Tabellen zu speichern und in SQL zu verwenden. Zu welchen Stilblüten dies schon in Oracle Forms geführt hat, wissen wir nur zu gut:

```
'TRUE', 'Y', 'YES', 'Yes', 'j',
'1', 1,
um nur die im deutschsprachigen Raum
beliebtesten zu nennen, kämpfen gegen
'FALSE', 'N', 'NO', 'No', 'n',
'0', 0.
```

Das ist höchst fehleranfällig (der Autor konnte miterleben, wie ein Produktionsausfall durch den Vergleich mit `'FALSE'` verursacht wurde).

Wie kann eine Lösung des beinahe babylonischen Problems aussehen, bis wir alle mit Version 23+ arbeiten dürfen? In

neu erstelltem Code verwendet man am besten eine Referenz auf eine Konstante, die in einem eigens dafür geschaffenen Package lebt, wie es in dem Vorschlag in *Listing 2* dargestellt ist.

Packages, die lediglich Konstanten oder Variablen deklarieren, benötigen keinen Package Body. Zu einem späteren Zeitpunkt lassen sich so alle Fundstellen leicht identifizieren und können mittels Refactoring über Boolesche Variablen vereinheitlicht werden. Für Legacy Code bleibt meist nur die Recherche mit mehr oder weniger geeigneten Suchmustern.

Einfach oder kompliziert?

Kaum ein anderes Programmierkonstrukt wird häufiger zur Kopiervorlage als das in *Listing 3a* dargestellte.

Geht das nicht einfacher? Das haben sich vermutlich auch die Architekten der Sprache PL/SQL gedacht, bevor sie vor circa 20 Jahren diese Alternative ins Leben gerufen haben (siehe *Listing 3b*).

Die Vorteile sind offensichtlich: keine Deklarationen, deutlich weniger Code. Man gibt zwar ein gewisses Maß an Kontrolle an den Compiler ab (beispielsweise die Anzahl der Datensätze, die intern per `BULK COLLECT` mit jedem `FETCH` abgeholt werden, üblicherweise 100), doch in der Mehrzahl der Fälle benötigt man diese Feinsteuerung nicht. Und falls doch, so ließe sich der Codeblock zu einem späteren Zeitpunkt kurzerhand per Refactoring

Help

PL/SQL Code

Enter the PL/SQL code to be executed.

Examples

Increases the salary by 3% for those employees which are qualified for a salary raise:

```

begin
  for l_emp ( select empno
               from emp
               where deptno = :P2_DEPTNO )
  loop
    if needs_salary_raise( l_empno.empno ) then
      update emp
        set sal = sal * 1.03
        where empno = l_emp.empno;
    end if;
  end loop;
end;

```

Additional Information

- Type: PL/SQL Code
- Supported Bind Variables: Application, Page Items and System Variables

Abbildung 1: Ein Auszug aus der APEX-Hilfe – bitte nicht nachmachen! (Quelle: Andreas Wissmann)

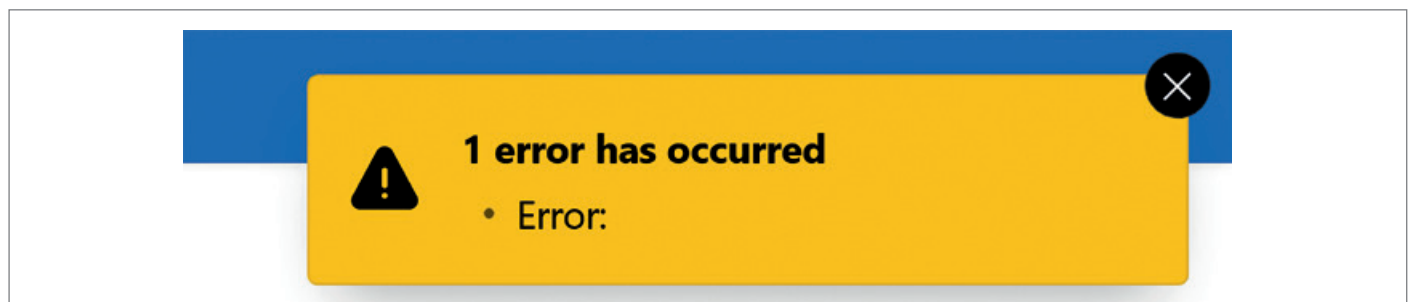


Abbildung 2: Eine Timeout-Fehlermeldung (Quelle: Andreas Wissmann)

```

CREATE OR REPLACE PACKAGE DEMO_1_PCK
AS
  PROCEDURE private_routine
    ACCESSIBLE BY (PACKAGE UT_DEMO_1_PCK);
END;
/

CREATE OR REPLACE PACKAGE BODY DEMO_1_PCK AS
  PROCEDURE private_routine
    ACCESSIBLE BY (PACKAGE UT_DEMO_1_PCK) IS
  BEGIN
    DBMS_OUTPUT.PUT_LINE('Nur zum Testen veröffentlicht');
    -- ...
  END;
END;
/

```

Listing 1: Verwendung der Klausel ACCESSIBLE BY

aufbohren. „Implizite Cursor-FOR-Loops“ sorgen also für Code, der erheblich einfacher zu schreiben und zu warten ist. Warum also ohne Not die aufwändigere Variante wählen?

Ein Type sagt mehr als 1000 Argumente

Mit der Komplexität der Anforderungen kann auch die Anzahl der Argumente steigen, die eine PROCEDURE oder FUNCTION erwarten. Irgendwann neigt das zur Unübersichtlichkeit. Ein prag-

```

CREATE OR REPLACE PACKAGE BOOLEAN_STRING
AS
    "TRUE"    CONSTANT VARCHAR2(4) := 'TRUE';
    "FALSE"   CONSTANT VARCHAR2(5) := 'FALSE';
END;
/
-- Programm-Pseudocode:
BEGIN
    CASE 'TRUE'
        WHEN BOOLEAN_STRING."TRUE" THEN NULL;
        WHEN BOOLEAN_STRING."FALSE" THEN NULL;
        ELSE NULL;
    END CASE;
END;

```

Listing 2: Ein dediziertes Package für BOOLEAN-Bezeichner

Shortcut

Show All | Name | Subscription | Source | Configuration | Comments

Name

Name FN_BEISPIEL_1 ?

Subscription

Subscribe From ?

This is the "master" copy of this shortcut.

No shortcuts subscribe to this shortcut.

Source

Type HTML Text ?

Shortcut

```
function beispiel_1(text) {
  console.log(text);
}
```

Abbildung 3a: Eine JavaScript-Funktion als APEX-Shortcut (Quelle: Andreas Wissmann)

☒ **JavaScript**

File URLs

Function and Global Variable Declaration

FN_BEISPIEL_1
FN_BEISPIEL_2

Execute when Page Loads

Abbildung 3b: Auflösung des Shortcuts in der APEX-Seite! (Quelle: Andreas Wissmann)

matischer Ausweg könnte so aussehen, dass man eine Menge von Argumenten in einen TYPE bündelt (zum Beispiel mit `CREATE TYPE T_ADRESSE AS OBJECT ...`) und daraufhin die Signatur erheblich vereinfacht.

Angenommen, der Geschäftsprozess verlangt eine postalische Versandadresse und das Unternehmen möchte nun auch den Versand an Packstationen ermöglichen: Neben vielen bereits existierenden Argumenten wie Straße, Hausnummer, Postleitzahl und vieles mehr kommen nun neue Angaben hinzu (siehe Listing 4a).

Der Clou an der Alternative ist, nachdem man einmalig den TYPE definiert hat, dass man die (erweiterte) alte und die neue Variante gleichberechtigt beibehält (siehe Listing 4b). Immerhin existiert bereits viel Programmcode, der nicht zwingend umgelenkt werden muss. Für neu zu schreibende Programme wird die Variante mit der kürzeren Signatur verwendet. Es ist immer eine gute Idee, Programmcode in Packages zu organisieren und auf Standalone Procedures oder Standalone Functions weitgehend zu verzichten: Nur Signaturen, die sich im selben Package befinden, können überladen werden. Um mit minimalem Aufwand zum Ziel zu kommen, überlädt man also die bestehenden Aufrufe und sorgt dafür, dass sich die neu hinzugekommenen Routinen einfach auf den existierenden abstützen. Keinesfalls die Geschäftslogik duplizieren!

Schnippselfagd

APEX, also known as Application Express, ist ein mächtiges und einzigartiges Framework. Leider mit einer nicht ganz unproblematischen Annehmlichkeit: Beim Programmieren hat man umfangreiche Freiheitsgrade, um Programmcode an den unterschiedlichsten Stellen zu speichern – ähnlich wie das bereits bei Oracle Forms möglich war und ist. Sehr verbreitet: Beliebige große Programmblöcke liegen in APEX-Prozessen, die beim Klicken von Buttons ausgelöst werden (Page Submit). In manchen Programmerteams scheint das sogar eher die Regel zu sein als die Ausnahme. Leider wird dieser Ansatz selbst von zahlreichen Beispielen in der APEX-Hilfe gefördert, wie man in einem Screens-


```

DECLARE
-- Explizite Cursor-Definition
CURSOR c_emp IS
    SELECT empno, ename, sal
    FROM emp
    WHERE deptno = 10;

-- Record-Variable passend zur Cursor-Struktur
l_emp c_emp%ROWTYPE;

BEGIN
-- Cursor öffnen
OPEN c_emp;

LOOP
-- Eine Zeile holen
FETCH c_emp INTO l_emp;

-- Abbruchbedingung prüfen
EXIT WHEN c_emp%NOTFOUND;

-- Verarbeitung
DBMS_OUTPUT.PUT_LINE(
    'Empno: ' || l_emp.empno ||
    ', Name: ' || l_emp.ename ||
    ', Salary: ' || l_emp.sal
);
END LOOP;

-- Cursor schließen
CLOSE c_emp;
END;
/

```

Listing 3a: Expliziter Cursor

```

BEGIN
FOR r IN (
    SELECT empno, ename, sal
    FROM emp
    WHERE deptno = 10
)
LOOP
    DBMS_OUTPUT.PUT_LINE(
        'Empno: ' || r.empno ||
        ', Name: ' || r.ename ||
        ', Salary: ' || r.sal
    );
END LOOP;
END;
/

```

Listing 3b: Impliziter Cursor-FOR-Loop

hot aus der Hilfe zu einem Page Process erkennt (siehe Abbildung 1).

Wenn in solchen Blöcken Geschäftslogik geparkt wird, dann ist dieser Code nach Fertigstellung für Dritte schwer aufzufinden. Was noch viel mehr ins Gewicht fällt: Solcher Code besitzt keinen Status (wie beispielsweise Packages in der Datenbank einen Status 'VALID' besitzen), weshalb er Laufzeitfehler verursachen könnte. Zudem lassen sich diese **Schnipsel** nur schwer testen, nämlich indem man die Anwendung bedient.

Das größte Defizit ist jedoch, dass in APEX hinterlegter Code nicht wiederverwendbar ist. Man kann es nicht oft genug wiederholen: PL/SQL-Geschäftslogik gehört in die Datenbank. Keine Ausnahmen!

Scheinbar endlos warten

In APEX-Applikationen gestartete Prozesse, die beispielsweise nach dem Klick auf eine Schaltfläche synchron ausgeführt werden und eine bestimmte – je nach Konfiguration der Laufzeitumgebung – kritische Ausführungsdauer überschreiten, führen mit einer mehr oder weniger aussagekräftigen Fehlermeldung im Frontend zu einer Timeout-Fehlermeldung. Auf der kostenfreien Testumgebung oracleapex.com erscheint in diesem Fall lediglich der in Abbildung 2 gezeigte Fehlerhinweis.

Bis zu diesem Moment sieht man in der App minutenlang nur die Sanduhr. Doch niemand möchte länger als ein paar Sekunden auf die Antwort einer Applikation warten, geschweige denn bis es zu einem Timeout-Fehler kommt. Selbstverständlich konnte man in APEX schon seit jeher Background-Jobs anstatt konventioneller PL/SQL-Prozesse einbauen, jedoch war das unhandlich zu programmieren. Daher scheute man im Zweifel den Aufwand – mit der Folge, dass die Geduld der User bisweilen auf eine harte Probe gestellt wurde.

Seit der Version APEX 23.1 sind zur Vermeidung dieses Zustands sogenannte Execution Chains verfügbar. Die Erstellung einer solchen Prozesskette ist simpel: Man erstellt zunächst im Page Designer einen neuen, benannten APEX-Prozess vom Typ Execution Chain. So

```
CREATE OR REPLACE PROCEDURE versandadresse_aendern (
    i_kundennummer IN INTEGER
    ,i_adresszusatz IN VARCHAR2
    ,i_strasse      IN VARCHAR2
    ,i_hausnummer  IN VARCHAR2
    ,i_gebaeudeteil IN VARCHAR2
    ,i_plz         IN VARCHAR2
    ,i_ort         IN VARCHAR2
    ,i_ortsteil    IN VARCHAR2
    ,i_land        IN VARCHAR2
    ,i_dhl_nummer  IN VARCHAR2
    ,i_packstation IN VARCHAR2
    ,i_telefon     IN VARCHAR2
);
```

Listing 4a: Wachsende Anzahl an Argumenten

```
CREATE OR REPLACE TYPE t_adresse IS RECORD (
    adresszusatz VARCHAR2(100),
    strasse      VARCHAR2(100),
    hausnummer  VARCHAR2(10),
    gebaeudeteil VARCHAR2(100),
    plz         VARCHAR2(10),
    ort         VARCHAR2(100),
    ortsteil    VARCHAR2(100),
    land        VARCHAR2(50),
    dhl_nummer  VARCHAR2(30),
    packstation VARCHAR2(30),
    telefon     VARCHAR2(30)
);

CREATE OR REPLACE PROCEDURE versandadresse_aendern (
    i_kundennummer IN INTEGER
    ,i_adresse      IN T_ADRESSE
);
```

Listing 4b: Ein TYPE statt einer Vielzahl an Argumenten

```
CREATE OR REPLACE PACKAGE GESCHAFTSPROZESSE_XY
AS
    PROCEDURE kunden_informieren
    ACCESSIBLE BY (
        PACKAGE APEX_12345_PCK,
        PACKAGE APEX_4711_PCK
    );
/
```

Listing 5: ACCESSIBLE BY mit APEX-Wrapper-Packages

bald dieser existiert, wird der eigentliche Langläufer-Geschäftsprozess hinzugefügt und dieser Execution Chain untergeordnet. Dazu sind insgesamt nur wenige Mausklicks erforderlich – ein konsequenter Schritt in Richtung Low Code. Unter der Haube übernimmt zur Laufzeit der DBMS_SCHEDULER die Regie. Man kann

während der Prozesslaufzeit an anderer Stelle in APEX weiterarbeiten. Über die zurückgelieferte EXECUTION ID lässt sich bei Bedarf der Fortschritt abfragen. Damit erledigen sich zwar nicht alle Herausforderungen auf einen Schlag, aber man erkennt (im einfachsten Fall!) durch wiederholtes Neuladen der Seite, ob der

Hintergrundprozess seinen Job erledigt hat. Die Sanduhr und alle Folgefehler sind damit zunächst abgestellt – die weiteren Maßnahmen können und sollten in diesem definierten Rahmen noch verfeinert werden. Den Königsweg dorthin, im Gegensatz zum permanenten Polling, findet man unter dem Stichwort WebSocket.

Identisches JavaScript auf mehreren APEX-Seiten

Es gibt Projekte, in denen auf diversen APEX-Seiten zusätzliches JavaScript eingebunden wird, um Features hinzuzufügen, die APEX nicht out-of-the-box mitbringt. Sofern es sich hierbei um kleinere, selbstgeschriebene Functions handelt, und nicht um große Bibliotheken von Drittherstellern, ergibt sich die Frage, wo sie hinterlegt werden können. Es ist sicher keine gute Idee, den Code zu kopieren und im „Page JavaScript“ vieler APEX-Seiten unterzubringen. Das Ziel sollte zwar sein, den Code letztendlich in Dateien auszuliefern, um durch File-Referenzen auf beliebig vielen APEX-Seiten zum Einsatz zu kommen. Doch während der Entwicklungszeit, wenn der Code noch nicht die erforderliche Reife besitzt, kann ein Fehler im Skript die Funktionalität der gesamten Seite beeinträchtigen – daher möchte man diese Skripte lieber nicht ständig modifizieren.

Stattdessen existiert in APEX ein Ort, der selten berücksichtigt wird, um Skripte zu speichern: die Shortcuts, welche sich in den Shared Components befinden. JavaScript als Shortcut zu speichern, ermöglicht es, eine kommentierte Version im Kommentarfeld des Shortcuts zu hinterlegen und in der Anwendung die Minified-Variante zu verwenden. Warum funktioniert das? Weil – erfreulicherweise – das Feld „Function and Global Variable Declaration“ die Auflösung von Shortcuts unterstützt.

Wildwuchs im Workspace

Dieses letzte Beispiel knüpft direkt an das erste an. Was wir im Kontext großer APEX-Workspaces oft beobachten, ist ein Wildwuchs an Prozedur- und Funktionsaufrufen direkt aus den Applikationen heraus. Dadurch wird es schwierig, Pro-

zessverläufe und Datenflüsse unter Kontrolle zu halten. Prinzipbedingt lässt sich daran rein technisch auch nichts ändern: Jede APEX-Anwendung darf natürlich alles verwenden, wozu sie im Rahmen des App-Parsing-Schemas berechtigt ist.

Mit dem bereits oben vorgestellten Werkzeug ACCESSIBLE BY könnte man ja prinzipiell erreichen, dass nur bestimmte APEX-Anwendungen bestimmte Packages aufrufen dürfen. Wodurch man wiederum ein viel klareres und sehr leicht zu recherchierendes Gesamtbild erhält.

Nun kommt Selbstdisziplin ins Spiel: Um den Namen des Aufrufers eindeutig festzulegen, muss man die Konvention befolgen, dass pro APEX-Applikation genau ein Package existiert, welches sämtliche Aufrufe aus APEX annimmt. Für die Applikation mit der App-ID 12345 erstellt man also ein Package namens APEX_12345_PCK (oder wie auch immer Ihre bevorzugte Nomenklatur aus-

sieht) und sorgt lediglich dafür, dass die APEX-App 12345 ausschließlich mit dem Package APEX_12345_PCK kommuniziert. Dieses agiert nun als Wrapper zur Weiterleitung an die bestehenden Geschäftspackages. Mit anderen Worten: im APEX_12345_PCK ist keinerlei Geschäftscodex enthalten, sondern nur simple Verdrahtung. Und nun kommt der Clou... Wir können nun eindeutig festlegen und vor allem automatisch dokumentieren, welche Routinen im Zielpackage von welchen APEX-Apps aufgerufen werden dürfen (siehe Listing 5).

Hier noch ein Bonus-Tipp zum Schluss: Mittels der langerprobten, dokumentierten und dennoch recht unbekannten Prozedur OWA_UTIL.WHO_CALLED_ME können Sie zur Laufzeit unterscheiden, aus welchem APEX-Wrapper-Package (sprich: aus welcher Applikation) Ihre Geschäftsroutine aufgerufen wurde, und darauf bei Bedarf unterschiedlich reagieren.

Über den Autor

Andreas Wismann, Diplom-Informatiker, Jahrgang 1964, berät, schult und unterstützt Unternehmen seit fast 30 Jahren bei der Anwendungsentwicklung mit Oracle PL/SQL und APEX.



Andreas Wismann
wismann@when-others.com

»DEV LAND«

THE NEXT GENERATION OF DEVELOPERS

**ON-DEMAND-TICKET SICHERN UND
EIN JAHR LANG DEVELOPMENT-
EXPERTISE MIT VORTRÄGEN UND
UNTERLAGEN ABRUFEN.**



#DEVLAND

WWW.DEVLAND.EU



Ende-zu-Ende-Digitalisierung mit APEX-Fachanwendungen im Backend

Benjamin Nebgen, GA1310 Eisenbahn-Bundesamt

Das Eisenbahn-Bundesamt (EBA) ist eine selbstständige deutsche Bundesbehörde im Geschäftsbereich des Bundesministeriums für Verkehr (BMV). Hierbei ist es die zuständige Aufsichts- und Genehmigungsbehörde für alle bundeseigenen Eisenbahnen. Es ist auch die zuständige Aufsichtsbehörde für die nichtbundeseigenen Eisenbahnunternehmen, die einer Sicherheitsbescheinigung oder Sicherheitsgenehmigung bedürfen. Oracle APEX ist hier ein wichtiges Instrument, um im Bereich der Entwicklung von Fachanwendungen schnell und effektiv auf die Anforderungen der Fachdienste reagieren zu können.

OZG als Chance in Zeiten von Ressourcenknappheit

Knappe Ressourcen und Fachkräftemangel setzen auch dem öffentlichen Dienst

vermehrt zu. Eine zukunftsorientierte und mitwachsende IT-Unterstützung der EBA-Geschäftsprozesse ist aus diesem Grund unabdingbar. Das EBA setzt seit nunmehr über 13 Jahren auf das sehr erfolgreiche

Konzept, die umfangreichen Fachaufgaben durch APEX-Fachanwendungen zu unterstützen. So werden inzwischen 23 verschiedene Fachverfahren durch APEX-Fachanwendungen als führendes Sys-

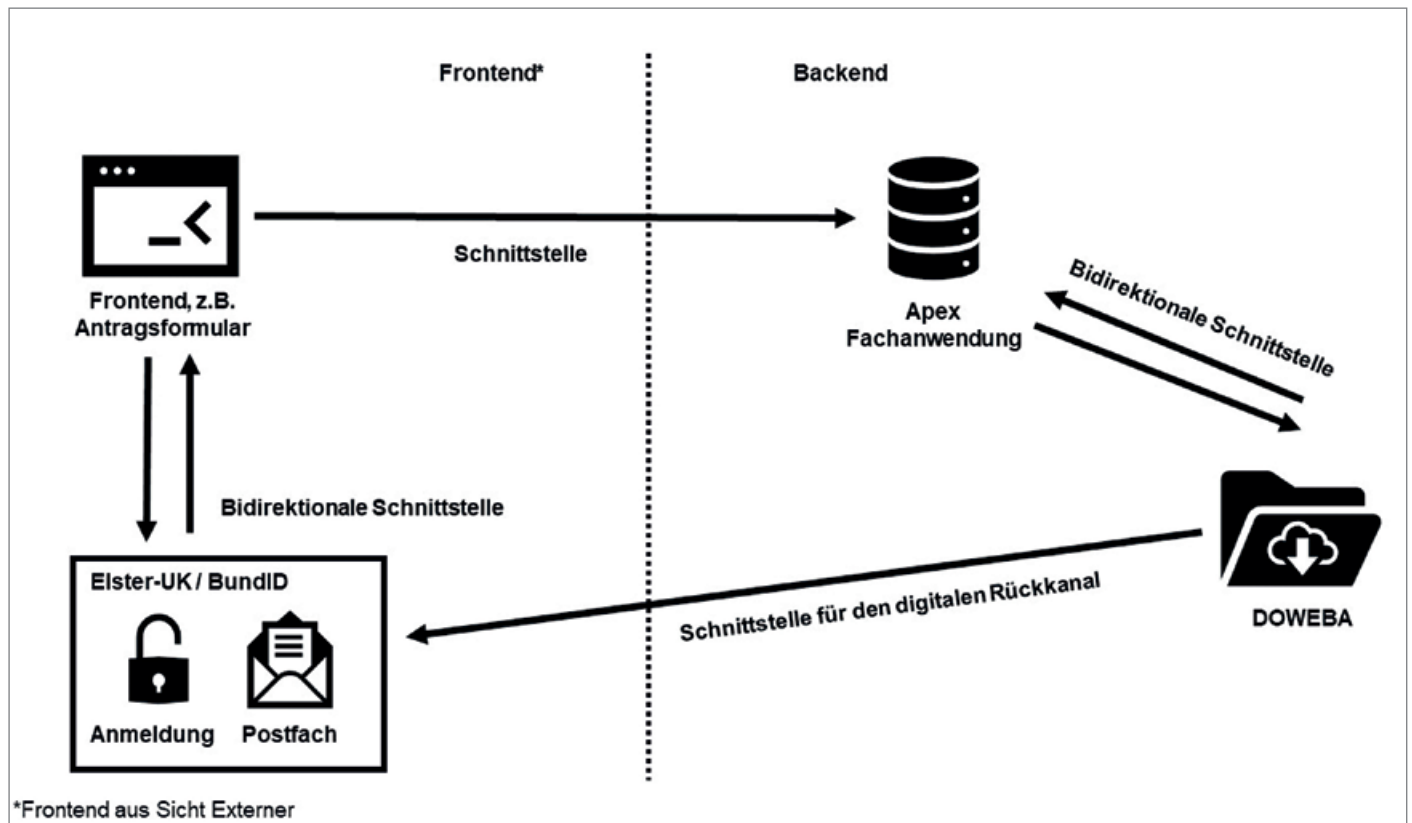


Abbildung 1: Grobe Lösungsarchitektur (Quelle: Benjamin Nebgen)

tem oder teilweise als unterstützendes System im EBA betrieben. Das EBA setzt hierzu auf eine Anwendungsplattform, in der wiederkehrende Funktionalitäten generisch entwickelt wurden und Konzepte zum Beispiel für eine stetige Wartbarkeit integriert sind. Dabei stellen die stetige Weiterentwicklung und Anpassung an neue gesetzliche Anforderungen oder sonstige geänderte Geschäftsprozesse immer wieder große Herausforderungen dar. Gerade im Zuge des Online-Zugangsgesetzes (OZG) hat die Thematik der Ende-zu-Ende-Digitalisierung an Bedeutung gewonnen. Hier helfen APEX-Fachanwendungen den Beschäftigten des EBA bei der beschleunigten und durchweg digitalen Antragsbearbeitung. APEX-Fachanwendungen kommen hierbei ausschließlich im Backend zum Einsatz.

Wie das EBA mit diesen Herausforderungen umgeht

In *Abbildung 1* erkennt man die grobe, sich wiederholende Lösungsarchitektur.

Die Antragssysteme (Frontends) werden OZG-konform durch verschiedene Dienstleister erstellt. Die extern gehos-

teten Systeme werden durch die im Backend on-Premise gehosteten Systeme des EBA angesprochen und die Antragsdaten zur Weiterverarbeitung abgeholt, um das Verwaltungshandeln abschließend in einem Dokumenten- und Managementsystem (hier: DOWEBA) rechtssicher zu dokumentieren. Frontend und Backend werden in der Regel über Schnittstellen, basierend auf dem Rest-Standard, verbunden. Die Daten werden dabei im Pull-Verfahren von Extern nach Intern übertragen und Ergebnisse (z. B. Bescheide oder Rückfragen) im Push-Verfahren zurück an das Frontend übergeben. Über den digitalen Rückkanal ist eine rechtssichere digitale Bescheid-Zustellung an den Antragsteller in Zukunft möglich.

Ein mögliches Anwendungsszenario

In einem möglichen Szenario meldet sich der Bürger zunächst mit seiner BundID oder seinem Elsterzertifikat im Formularsystem an und authentifiziert sich hierüber. Je nach OZG-Reifegrad des umgesetzten Formulars und dem

verwendeten Authentifizierungsmechanismus ist er berechtigt, den angebotenen Dienst zu nutzen. Die personenbezogenen Daten wie Name, Vorname, Geburtsdatum, Wohnanschrift, etc. werden, wenn gewollt, an den entsprechenden Stellen im Formular übernommen. Je nach abgebildetem Formular werden ergänzende Dokumente hochgeladen und zusammen mit den Formulardaten übermittelt. Vom Formularserver wiederum werden die übermittelten Formulardaten und Dokumente über eine REST-Schnittstelle abgeholt und in einer APEX-Fachanwendung zur Antragsprüfung zur Verfügung gestellt. Hierbei sind unterstützende Elemente implementiert, die die Antragsbearbeitung erleichtern sollen. So werden zum Beispiel Änderungen zu Bestandsdaten, die sich aus den Antragsdaten ergeben, farblich hervorgehoben. Somit wird die Antragsbearbeitung erleichtert und Unstimmigkeiten können schneller erkannt werden. Selbstredend sind in die Geschäftslogik-Plausibilitätsprüfungen eingebaut, wie beispielsweise, ob der Antragsteller das Mindestalter für die Antragstellung erreicht hat. Nach der finalen Antragbearbeitung werden im

EBA etablierte Geschäftsprozesse vollautomatisiert angestoßen. So werden die entsprechenden Bescheide automatisiert erstellt, im Vorgangsbearbeitungs- und Dokumentenmanagementsystem archiviert und über den digitalen Rückkanal an den Antragsteller übermittelt. Während der Antragsprüfung stehen zudem vollintegrierte Schnittstellen zu Registern anderer Behörden zur Verfügung, deren Daten zur Antragsgenehmigung herangezogen werden. Am Beispiel des Führerscheinregisters lässt sich dies leicht verdeutlichen: Nach erfolgreicher Prüfung der Antragsdaten werden die für die Bestellung notwendigen Daten an die Bundesdruckerei digital übermittelt, um dort den Bestellprozess der Führerscheinkarte auszulösen. Anschließend wird der Zahlungsbescheid automatisiert erstellt und an den Antragsteller übermittelt. Somit ist das Verwaltungshandeln voll digitalisiert erfolgt und abgeschlossen.

Auswertungen als Steuerungsinstrument

Ein weiterer wichtiger Aspekt für den Einsatz der APEX-Anwendungen im EBA ist das Erreichen einer möglichst hohen Datenqualität und damit ein einhergehendes Berichtswesen. Migrationen bestehender Excel-Listen, die oft Behelfslösungen der Fachdienste darstellten, stellen immer wieder größere Herausforderungen dar.

Die in APEX zur Verfügung gestellten Auswertungen sorgen für eine stetige Weiterentwicklung der Geschäftsprozesse und unterstützen den Ressourceneinsatz im EBA.

Ansatz der Anwendungsplattform ist ein Erfolgsmodell

Der im EBA verfolgte Ansatz, eine Anwendungsplattform auf Basis von Oracle APEX aufzubauen, wurde vor nunmehr 13 Jahren beschlossen und seitdem stetig weiterentwickelt. Der hier entstandene Werkzeugbaukasten, nebst Konzepten für die Entwicklung und Nutzung von APEX-Fachanwendungen, birgt aus Sicht des EBA einige Vorteile:

- Der generische Ansatz bei der Entwicklung von Funktionalitäten ermöglicht es, diese immer wieder zu verwenden und Entwicklungsaufwände hierdurch zu reduzieren.
- Die Neu- und Weiterentwicklung von APEX-Anwendungen wird hierdurch extrem beschleunigt.
- Die Ähnlichkeit zwischen den eingesetzten Plattformfunktionalitäten erleichtert die Wartbarkeit der Systeme.
- Die Möglichkeit Geodaten etwa in Form von voll integrierten Kartenkomponenten darzustellen, erleichtert und unterstützt Prüfprozesse stark.
- Die Nutzerakzeptanz wird durch das einheitliche Bedienkonzept stark erhöht. Schulungen konnten durch innovative Bedienoberflächen und automatisierte Geschäftsabläufe minimiert werden.

Ausblick

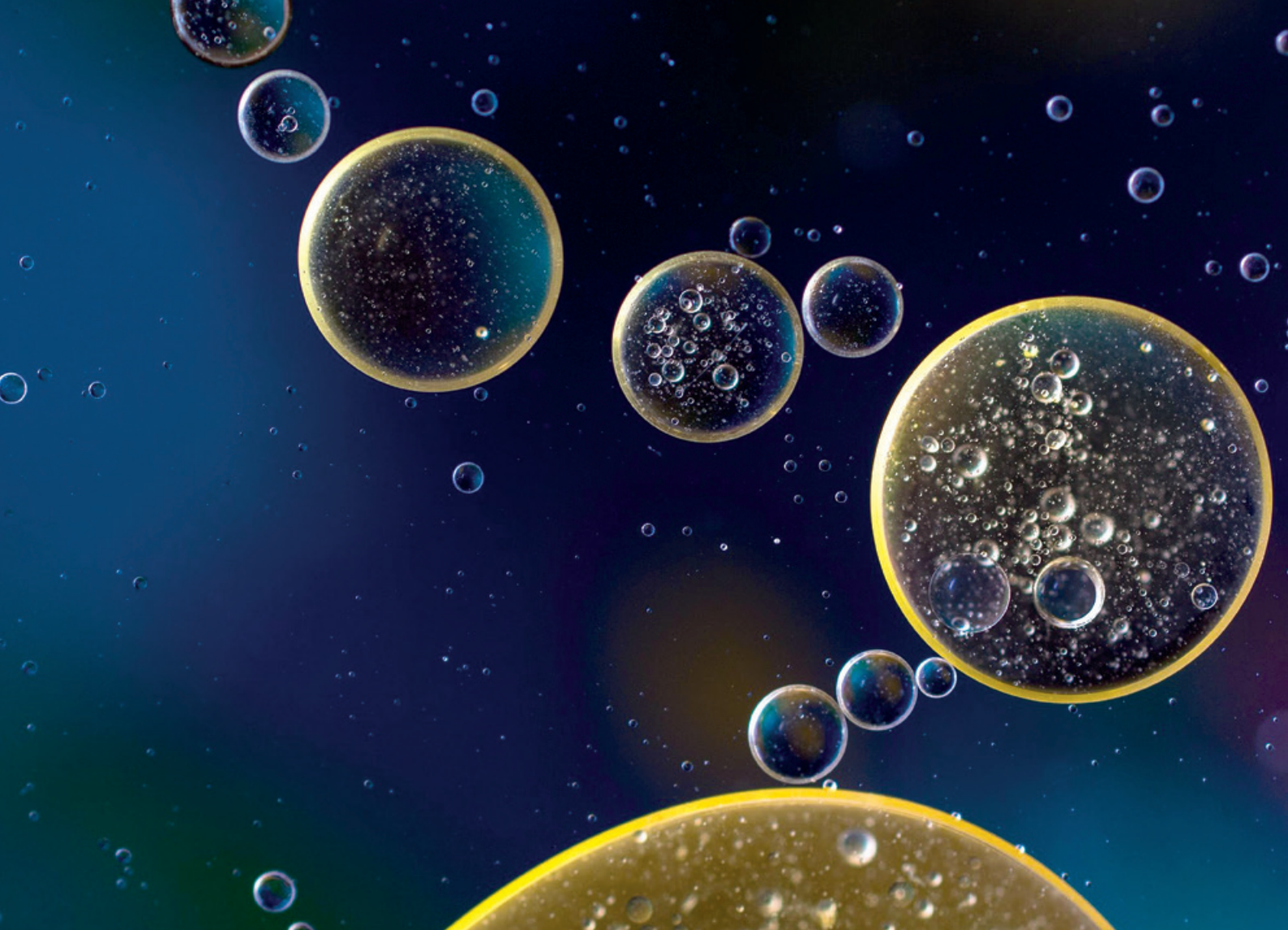
Der Digitalisierungsprozess im EBA wird weiter voranschreiten. Der Fachkräftemangel und knappe Ressourcen sind dabei Bremser und Förderer zugleich. Ein wichtiges Instrument, um dem entgegenzuwirken, wird Oracle APEX sein. Die stetige Weiterentwicklung der Anwendungsplattform und Integration neuer Funktionalitäten sind hierfür unabdingbar. Dies soll auch in Zukunft der Weg für das EBA sein, wobei wir die Entwicklungen, die seitens Oracle angestoßen werden, aufmerksam verfolgen werden. Wir freuen uns auf neue, innovative Ideen und Möglichkeiten, die wir, wenn nötig und möglich, unseren Fachdiensten zur Verfügung stellen wollen.

Über den Autor

Benjamin Nebgen ist seit circa 20 Jahren in der IT unterwegs. Den ersten Kontakt zu Oracle APEX hatte er vor 13 Jahren, als er IT-Koordinator im EBA für Fachanwendungen auf Basis von Oracle APEX wurde. Seitdem betreut er die dortige Anwendungsplattform und ist für diverse Fachanwendungen zuständig, die die Fachabteilungen in unterschiedlichsten Tätigkeitsfeldern unterstützen. Das mit ihm zusammenarbeitende Team leitet er.



Benjamin Nebgen
nebgenb@eba.bund.de



APEX-Plug-ins – Von den Anfängen bis zu KI

Ronny Weiß, Oracle Global Services Germany

Oracle APEX (Application Express) ist ein ideales Werkzeug, um moderne, skalierbare und sichere Webanwendungen sowie PWA-Lösungen in kurzer Zeit zu entwickeln. Die Plattform liefert bereits im Standardumfang eine beeindruckende Funktionsvielfalt, die mit jedem Release weiterwächst und den Bedarf an individuellen Erweiterungen kontinuierlich reduziert. Dennoch gibt es Szenarien, in denen sich bestimmte Anforderungen nur durch eine gezielte Erweiterung des Funktionsspektrums umsetzen lassen. Früher musste dafür eigener Code direkt auf den jeweiligen APEX-Seiten eingebettet werden – schwer wartbar, kaum wiederverwendbar und nur mühsam teilbar. Um genau dieses Problem zu lösen, führte das APEX Entwicklungsteam vor 16 Jahren, im Juni 2010 mit APEX 4.0, die Plug-in-Infrastruktur ein. Sie machte es erstmals möglich, APEX modular, sauber strukturiert und langfristig wartbar zu erweitern. Seitdem hat sich enorm viel getan: Die Plug-in-Architektur wurde stetig weiterentwickelt, professionalisiert und zu einem zentralen Baustein des APEX-Ökosystems. Dieser Artikel zeigt, welche Entwicklungen seit der Einführung stattgefunden haben, wie leistungsfähig Plug-ins heute sind und welche Möglichkeiten sie modernen APEX-Anwendungen eröffnen.

APEX 4

Direkt mit dem ersten Release der damals neuen Plug-in-Infrastruktur bot APEX die Möglichkeit, sämtliche zentralen Funktionsbereiche zu erweitern. Neben Item-, Region- und Process-Type-Plug-ins konnten Entwickler bereits ab Tag eins auch Erweiterungen für die neu eingeführten Dynamic Actions erstellen. Der entscheidende Vorteil lag nicht nur in der deutlich besseren Wartbarkeit und Wiederverwendbarkeit des Codes. Plug-ins machten diese Erweiterungen erstmals auch für Low-Code-Anwender zugänglich: Neue APEX-Anwendungen ließen sich damit genauso einfach und deklarativ erstellen wie mit den nativen Komponenten der Plattform. Das war ein strategisch brillanter Schritt des Entwicklerteams.

Auch die Community war von Beginn an aktiv eingebunden. Über die Plattform apex-plugin.com konnten Entwickler ihre Plug-ins veröffentlichen, teilen und in andere APEX-Umgebungen integrieren. So entstand früh ein lebendiges Ökosystem, das APEX bis heute prägt und kontinuierlich bereichert.

Mit APEX 4.2 kam ein weiterer Plug-in -Typ hinzu: Erweiterungen für Authorization Schemes und Validierungen (siehe Abbildung 1). Damit ließ sich erstmals auch die sicherheitsrelevante und regelbasierte Logik einer Anwendung modular erweitern. Der Schwerpunkt dieses Releases lag jedoch klar auf der Clientseite.

Zu den wichtigsten Neuerungen gehörte die Einführung der bis heute zentralen JavaScript API `apex.server.plugin(url)`, die eine standardisierte und sichere Kommunikation zwischen der Clientseite von Plug-ins und dem Server ermöglichte. Zusätzlich konnten Entwickler nun deklarative Events definieren, die direkt im Page Designer sichtbar und nutzbar waren – ein entscheidender Schritt, um komplexe Interaktionen ohne Custom Code zugänglich zu machen. Ergänzt wurde dies durch eine Vielzahl neuer JavaScript-APIs, die Plug-ins deutlich mächtiger und flexibler machten.

APEX 5

Mit APEX 5 hielt das Universal Theme Einzug und brachte einen umfassenden Modernisierungsschub für das gesamte UI

und Styling-Konzept. Neben einem konsistenten, responsiven Design wurden zahlreiche Hilfsklassen eingeführt, die sich hervorragend für die Gestaltung von Plug-ins eignen (siehe Abbildung 2). Diese Utility-Klassen sind bis heute unter apex.oracle.com/ut dokumentiert und bilden eine wertvolle Grundlage, um Plug-ins optisch nahtlos in das APEX Look and Feel einzubetten.

Ein weiterer wichtiger Schritt war die Möglichkeit, Item-Plug-ins direkt in das neue Interactive Grid zu integrieren.

Auf der Clientseite wurde die Architektur mit der Einführung der `apex.da` API erweitert. Dadurch konnten auf das Plug-in nachfolgende Actions beispielsweise warten, bis ein Server Call abgeschlossen war, oder vollständig übersprungen werden, wenn ein Fehler auftrat. Das machte die Verwendung von Dynamic Action Plug-ins nicht nur robuster, sondern auch wesentlich besser in komplexe Dynamic Action Flows integrierbar.

Eine der wichtigsten Neuerungen dieses Zeitraums kam jedoch nicht direkt vom APEX-Entwicklungsteam, sondern aus der Community! Die Plattform ging online. Sie veränderte nachhaltig, wie Plug-ins verbreitet, gefunden und gepflegt werden. Bis heute ist apex.world die zentrale Community-Anlaufstelle – sowohl für allgemeine APEX-Themen als auch speziell für die Veröffentlichung und

den Download von Plug-ins. Aktuell sind dort 461 Plug-ins gelistet und die Zahl wächst stetig.

Auch ich selbst habe inzwischen 33 Plug-ins auf apex.world veröffentlicht und versuche – soweit es die Zeit zulässt –, diese kontinuierlich zu pflegen und neue Plug-ins bereitzustellen.

APEX 18

Mit APEX 18.1 wurde die APEX_EXEC-Infrastruktur eingeführt – ein Wendepunkt für den Umgang mit Daten in APEX-Anwendungen und Plug-ins. Diese API veränderte nicht nur das Laden und Speichern von Daten innerhalb der nativen Komponenten, sondern stand auch Plug-ins vollständig zur Verfügung. Dadurch konnten Plug-ins ohne eigenen, komplexen Datenzugriffscode auf unterschiedlichste Datenquellen zugreifen und Daten persistieren. Viele Aufgaben, die zuvor individuell implementiert werden mussten, wurden damit standardisiert und erheblich vereinfacht.

APEX 20

Mit APEX 20 wurde dieser Ansatz durch die Einführung der REST Data Source Plug-ins weiter ausgebaut. Plug-ins konn-

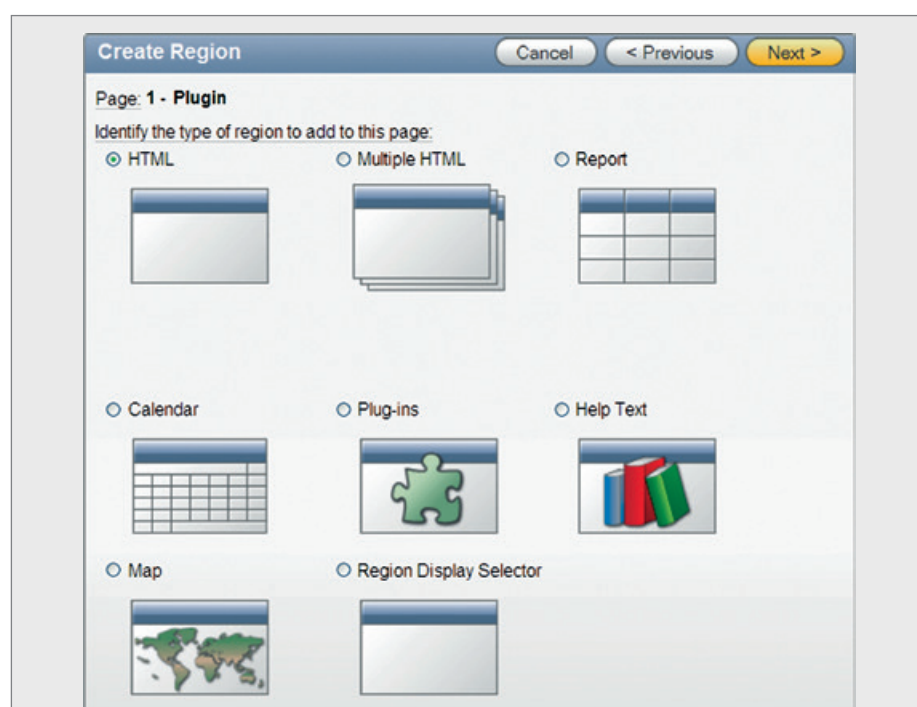


Abbildung 1: APEX 4.2 Application Builder (Quelle: <https://oracle.com>)

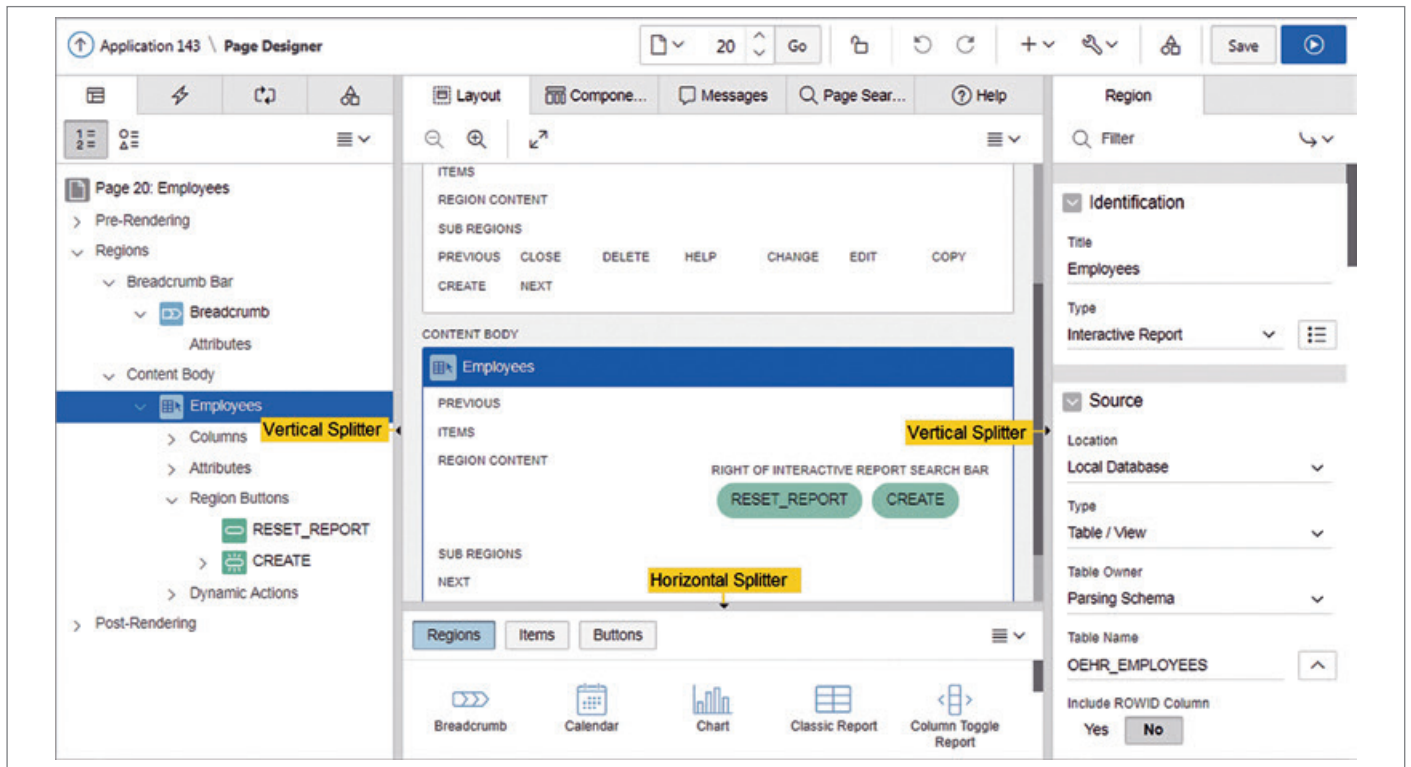


Abbildung 2: APEX 5.0 Page Designer (Quelle: <https://docs.oracle.com>)

Example:

The page contains items P1_TITLE, P_ICON, P1_DESCRIPTION, and P1_DETAILS and all have optional values. The template outputs a default title if P1_TITLE is empty. An optional icon is shown only if there is a title. The template output includes markup for the description if it is not empty or details if it is not empty and nothing otherwise.

```
<h3>{if ?P1_TITLE/}&P1_TITLE. {if P1_ICON/}<span class="fa &P1_ICON."></span>{endif/}
{else/}<h3>
{if P1_DESCRIPTION/}
<p class="description">&P1_DESCRIPTION.</p>
{elseif P1_DETAILS/}
<p class="details">&P1_DETAILS.</p>
{endif/}
```

Case condition directives

Syntax:

```
{case NAME/}
{when string1/}
TEMPLATE_TEXT1
{when string2/}
TEMPLATE_TEXT2
{otherwise/}
TEMPLATE_TEXT
{endcase/}
```

Abbildung 3: Template Directives (Quelle: <https://apex.oracle.com/jsapi>)

ten nun deklarativ REST-basierte Datenquellen anbinden und damit noch flexibler auf externe Systeme zugreifen.

Parallel dazu erhielt die Clientseite zwei besonders wichtige JavaScript-APIs:

- apex.locale ermöglichte sprachunabhängiges Encodieren und Decodieren

von Zahlen und damit eine saubere Internationalisierung.

- apex.util.applyTemplate – vereinfachte die Nutzung der neuen Template Directives (siehe Abbildung 3) extrem und legte die Grundlage für die späteren Template Component Plug-ins.

APEX 21

Mit APEX 21 folgte ein weiterer großer Schritt: Die apex.item API wurde um clientseitige Validierungen erweitert. Dadurch konnten Item Plug-ins erstmals vollständig in das native Validierungs-Look-and-Feel integriert werden – inklu-

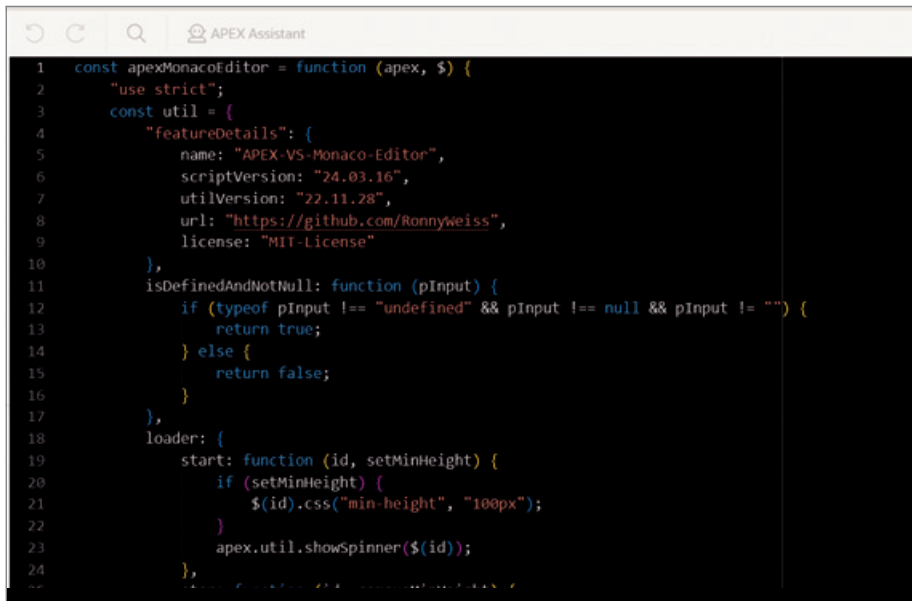


Abbildung 4: Static File Editor (Quelle: Ronny Weiß)

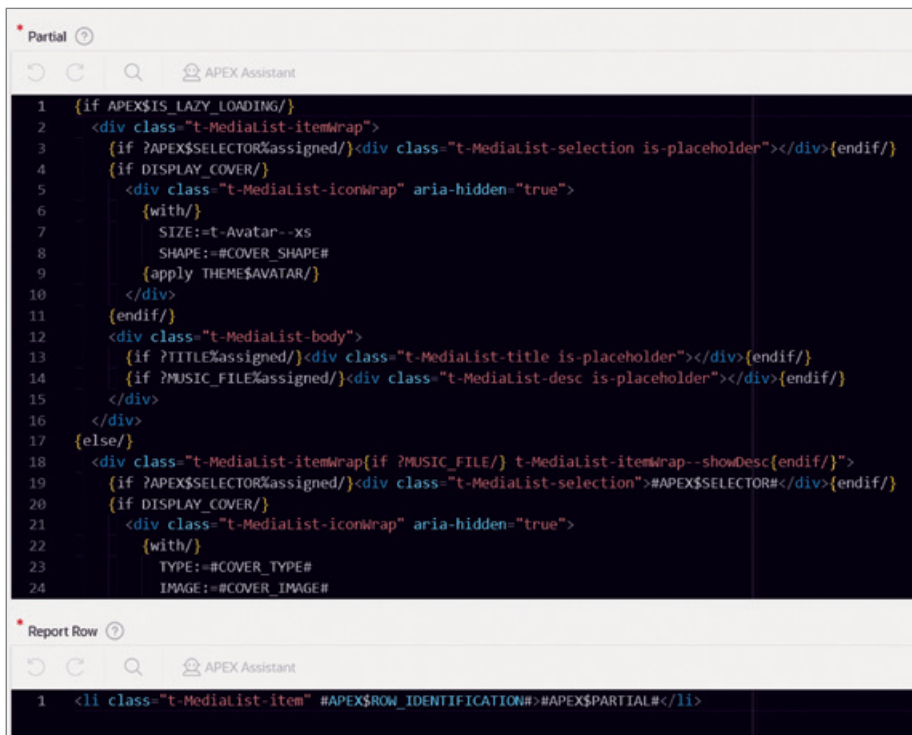


Abbildung 5: Template Components (Quelle: Ronny Weiß)

sive Fehlerdarstellung, Styling und Interaktionslogik.

Ebenfalls in APEX 21 setzte Oracle erstmals auf Web Components, konkret bei den neuen Date- und Color-Picker-Items. Diese Technologie veränderte nachhaltig, wie Plug-ins heute entwickelt werden. Web Components bieten entscheidende Vorteile:

- Sie erweitern native Browser-Elemente, statt alles in JavaScript manuell zu initialisieren.

- Sie verwalten ihren eigenen Lebenszyklus, was die Komplexität deutlich reduziert.
- Sie werden primär über HTML Attribute gesteuert, wodurch sich das Verhalten eines Plug-ins zur Laufzeit allein durch Attributänderungen beeinflussen lässt.
- Sie ermöglichen sowohl einfache Elemente als auch hochkomplexe UI Komponenten ohne, dass Entwickler tief in Initialisierungslogik eingreifen müssen.

Diese Kombination aus APEX_EXEC, weiteren wichtigen JavaScript APIs, und Web Component Technologie hat die Plug in Entwicklung auf ein neues Niveau gehoben und bildet die Basis für moderne, robuste und deklarativ steuerbare Plug ins.

APEX 22.2

Mit APEX 22 hielt endlich der Static File Editor Einzug – ein Feature, auf das die Community lange gewartet hatte (siehe Abbildung 4). Damit war es erstmals möglich, JavaScript- und CSS-Dateien direkt in APEX zu bearbeiten, ohne externe Tools oder umständliche Upload-Prozesse. Änderungen konnten sofort getestet werden und der Editor bot Komfortfunktionen wie Code-Highlighting, automatische Formatierung und sogar eine integrierte Minifizierung. Für die Plug-in-Entwicklung war das ein echter Meilenstein, denn es beschleunigte den gesamten Entwicklungs- und Testzyklus erheblich.

Auch auf der Clientseite brachte APEX 22 wichtige Erweiterungen. Zwei neue APIs stachen besonders hervor:

- apex.date – ermöglichte das zuverlässige Encodieren und Decodieren von Datumswerten, unabhängig von Sprache oder Regionseinstellungen.
- apex.env – bot von nun an die Möglichkeit, Informationen wie den aktuellen APP_USER oder die APEX_FILES URL direkt clientseitig abzufragen, was Plug-ins mehr Kontext und Flexibilität gab.

Diese Verbesserungen machten Plug-ins nicht nur leistungsfähiger, sondern auch deutlich einfacher zu entwickeln und zu warten. Sie markierten den Übergang zu einer moderneren, integrierten Entwicklungsumgebung innerhalb von APEX.

APEX 23

Mit APEX 23 änderte sich die Plug-in-Entwicklung grundlegend. Bis dahin war sie vor allem erfahrenen Entwicklern vorbehalten, weil man Daten laden, Rendering Logik implementieren und den kompletten Lebenszyklus eines Plug-ins selbst programmieren musste.

Mit den Template Component Plug-ins wurde dieser Ansatz aufgebrochen (siehe

DOAG EVE

JAN

01 Neujahr Do 1.	Fr 2.	Sa 3.	So 4.	Mo 5.	Di 6.	Mi 7.	Do 8.	Fr 9.	Sa 10.	So 11.	Mo 12.	Di 13.	Mi 14.	Do 15.
------------------------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

FEB

So 1.	Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

MAR

So 1.	Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

APR

Mi 1.	Do 2.	Karfreitag Fr 3.	Sa 4.	Oster- sonntag So 5.	Oster- montag Mo 6.	Di 7.	Mi 8.	Do 9.	Fr 10.	Sa 11.	So 12.	Mo 13.	Di 14.	Mi 15.
-------	-------	---------------------	-------	----------------------------	---------------------------	-------	-------	-------	--------	--------	--------	--------	--------	--------

MAI

Tag der Arbeit Fr 1.	Sa 2.	So 3.	Mo 4.	Di 5.	Mi 6.	Do 7.	Fr 8.	Sa 9.	So 10.	Mo 11.	Di 12.	Mi 13.	Christi Himmelfahrt Do 14.	Fr 15.
----------------------------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	----------------------------------	--------

JUN

Mo 1.	Di 2.	Mi 3.	Do 4.	Fr 5.	Sa 6.	So 7.	Mo 8.	Di 9.	Mi 10.	Do 11.	Fr 12.	Sa 13.	So 14.	Mo 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------



JUL

Mi 1.	Do 2.	Fr 3.	Sa 4.	So 5.	Mo 6.	Di 7.	Mi 8.	Do 9.	Fr 10.	Sa 11.	So 12.	Mo 13.	Di 14.	Mi 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

AUG

Sa 1.	So 2.	Mo 3.	Di 4.	Mi 5.	Do 6.	Fr 7.	Sa 8.	So 9.	Mo 10.	Di 11.	Mi 12.	Do 13.	Fr 14.	Sa 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

SEP

Di 1.	Mi 2.	Do 3.	Fr 4.	Sa 5.	So 6.	Mo 7.	Di 8.	Mi 9.	Do 10.	Fr 11.	Sa 12.	So 13.	Mo 14.	Di 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

OKT

Do 1.	Fr 2.	Tag der deutschen Einheit Sa 3.	So 4.	Mo 5.	Di 6.	Mi 7.	Do 8.	Fr 9.	Sa 10.	So 11.	Mo 12.	Di 13.	Mi 14.	Do 15.
-------	-------	--	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

NOV

So 1.	Mo 2.	Di 3.	Mi 4.	Do 5.	Fr 6.	Sa 7.	So 8.	Mo 9.	Di 10.	Mi 11.	Do 12.	Fr 13.	Sa 14.	So 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

DEZ

Di 1.	Mi 2.	Do 3.	Fr 4.	Sa 5.	So 6.	Mo 7.	Di 8.	Mi 9.	Do 10.	Fr 11.	Sa 12.	So 13.	Mo 14.	Di 15.
-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------

NTKALENDER

			04						05						
Fr 16.	Sa 17.	So 18.	Mo 19.	Di 20.	Mi 21.	Do 22.	Fr 23.	Sa 24.	So 25.	Mo 26.	Di 27.	Mi 28.	Do 29.	Fr 30.	Sa 31.
08							09								
Mo 16.	Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.			
12							13							14	
Mo 16.	Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.	So 29.	Mo 30.	Di 31.
				17							18				
Do 16.	Fr 17.	Sa 18.	So 19.	Mo 20.	Di 21.	Mi 22.	Do 23.	Fr 24.	Sa 25.	So 26.	Mo 27.	Di 28.	Mi 29.	Do 30.	
		APEX connect + DOAG 2026 Datenbank mit Cloud Infrastructure							22						
Sa 16.	So 17.							Pfingst- sonntag	Pfingst- montag						
							26						27		
Di 16.	Mi 17.	Do 18.	Fr 19.	Sa 20.	So 21.	Mo 22.	Di 23.	Mi 24.	Do 25.	Fr 26.	Sa 27.	So 28.	Mo 29.	Di 30.	



DOAG 2026 Datenbank

mit Cloud Infrastructure

datenbank.doag.org

im
Heide Park
Soltau

18. – 19.
Mai
2026

#CLOUDLAND2026

19. – 22.
MAI
2026

CLOUD NATIVE FESTIVAL

im Heidepark in Soltau

CloudLand
www.cloudland.org

2026
DOAG
Konferenz + Ausstellung

Nürnberg | 17. – 20. Nov.

anwenderkonferenz.doag.org #doag2026

				30						31					
Do 16.	Fr 17.	Sa 18.	So 19.	Mo 20.	Di 21.	Mi 22.	Do 23.	Fr 24.	Sa 25.	So 26.	Mo 27.	Di 28.	Mi 29.	Do 30.	Fr 31.
	34							35							32
So 16.	Mo 17.	Di 18.	Mi 19.	Do 20.	Fr 21.	Sa 22.	So 23.	Mo 24.	Di 25.	Mi 26.	Do 27.	Fr 28.	Sa 29.	So 30.	Mo 31.
					39							40			
Mi 16.	Do 17.	Fr 18.	Sa 19.	So 20.	Mo 21.	Di 22.	Mi 23.	Do 24.	Fr 25.	Sa 26.	So 27.	Mo 28.	Di 29.	Mi 30.	
			43							44					
Fr 16.	Sa 17.	So 18.	Mo 19.	Di 20.	Mi 21.	Do 22.	Fr 23.	Sa 24.	So 25.	Mo 26.	Di 27.	Mi 28.	Do 29.	Fr 30.	Sa 31.
47							48							49	
Mo 16.	Di 17.	Mi 18.	Do 19.	Fr 20.	Sa 21.	So 22.	Mo 23.	Di 24.	Mi 25.	Do 26.	Fr 27.	Sa 28.	So 29.	Mo 30.	
					52							53			
Mi 16.	Do 17.	Fr 18.	Sa 19.	So 20.	Mo 21.	Di 22.	Mi 23.	Do 24.	Heilig- abend	1.Weih- nachtstag	2.Weih- nachtstag				Silvester Do 31.

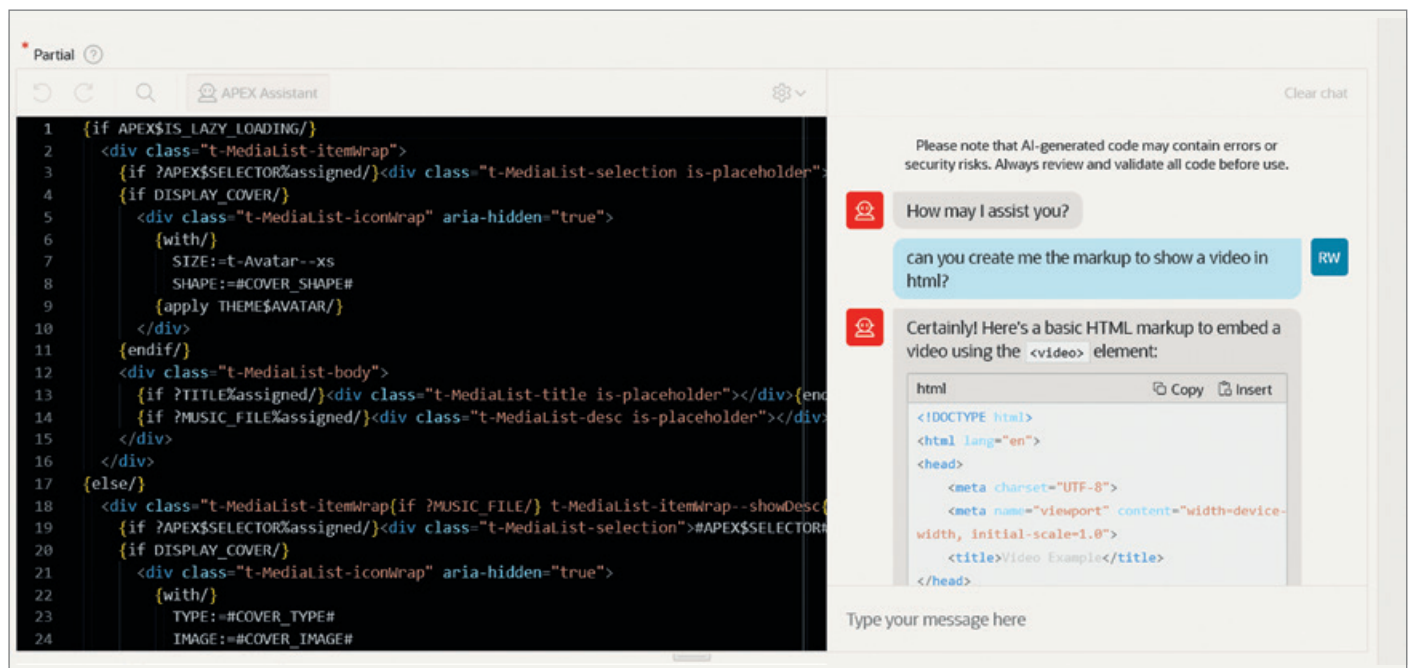


Abbildung 6: APEX AI Assistant (Quelle: Ronny Weiß)

he Abbildung 5). Dieser neue Plug-in-Typ machte die Entwicklung erstmals auch für Citizen Developers zugänglich. Statt PL/SQL- oder JavaScript-Logik zu schreiben, definierte man das Rendering fast ausschließlich über Template Directives – also im Wesentlichen HTML mit ein wenig Logik. Ergänzend konnte man bei Bedarf etwas CSS hinzufügen, um das Styling zu verfeinern. Mehr war nicht nötig. Kein PL/SQL. Kein JavaScript. Keine komplexen Lifecycle-Implementierungen.

Das erklärt auch, warum Template Component Plug-ins heute direkt hinter den klassischen Region Plug-ins auf Platz zwei der meistveröffentlichten Plug-in-Typen stehen. Da sie ohne eigenes Coding auskommen, sind sie zudem deutlich sicherer für den Einsatz in Projekten, insbesondere wenn man Plug-ins von *apex.world* nutzt.

Trotz ihrer Einfachheit sind Template Component Plug-ins äußerst leistungsfähig. Sie unterstützen: einfache Regions, komplexe Reports mit Pagination, dynamisches Refresh und sogar die Integration in Interactive Reports.

Eine weitere wichtige Neuerung war die Einführung der Unlimited Attributes für Template Component Plug-ins sowie der Attribute Groups. Damit konnten Entwickler erstmals beliebig viele deklarative Attribute definieren und diese übersichtlich gruppieren. Das erhöhte nicht nur die Flexibilität, sondern auch

die Benutzerfreundlichkeit im Page Designer erheblich.

Diese Entwicklungen machten Plug-ins modularer, sicherer und vor allem für eine viel breitere Zielgruppe zugänglich.

APEX 24+

In den neuesten APEX Releases hat sich erneut viel für die Plug-in-Entwicklung getan. Template Component Plug-ins unterstützen nun deklarativ Selections, Slots und Grouping, wodurch sich komplexe Layout- und Interaktionsmuster vollständig ohne eigenes Coding abbilden lassen. Für Region Plug-ins wurde die Funktionalität der Unlimited Attributes eingeführt und für Multi-Value Item Plug-ins gibt es eine neue Infrastruktur, die ohne manuelle Programmierung unterschiedliche Separatoren und sogar Arrays verarbeiten kann. Das macht diese Plug-ins robuster, flexibler und deutlich einfacher zu konfigurieren.

Die wichtigste Neuerung ist jedoch die Einführung von KI in APEX. Entwickler können nun direkt beim Arbeiten in APEX KI Unterstützung nutzen – sei es für:

- das Erstellen von Templates
- das Generieren von Beispielcode
- die Fehlersuche
- das schnelle Prototyping neuer Ideen

Plug-ins selbst erhalten über die APEX_AI API Zugriff auf dieselben KI-Funktionen (siehe Abbildung 6). Damit können sie KI-gestützte Features direkt in ihre eigene Logik integrieren und den Funktionsumfang, im Bereich KI, von APEX nahezu unbegrenzt erweitern. Durch AI Configurations stehen zudem System Prompts und RAG-Daten zur Verfügung, sodass Plug-ins kontextsensitiv und projektspezifisch mit KI arbeiten können.

Der große Vorteil: Plug-ins werden dadurch für Citizen Developers noch zugänglicher. Gerade beim Prototyping kann KI heute eine enorme Zeitersparnis bringen.

Ich selbst nutze KI inzwischen fast täglich – besonders beim schnellen Erstellen von Prototypen. Ein gutes Beispiel ist das Kanban-Board-Plug-in, dass ich für meinen Vortrag auf der letzten DOAG-Konferenz entwickelt habe. Den Beispielcode für das barrierefreie HTML-Gerüst und mehrere JavaScript-Funktionen, wie das Drag-and-Drop-Verhalten, habe ich mir vollständig von KI generieren lassen. Dadurch konnte ich viele Stunden Entwicklungszeit einsparen.

Natürlich war der generierte Code nicht perfekt. Ich musste ihn anpassen, optimieren und in ein vollwertiges APEX-Plug-in überführen. Aber KI hat mir die Anfangsarbeit massiv erleichtert und den kreativen Prozess beschleunigt. Für mo-

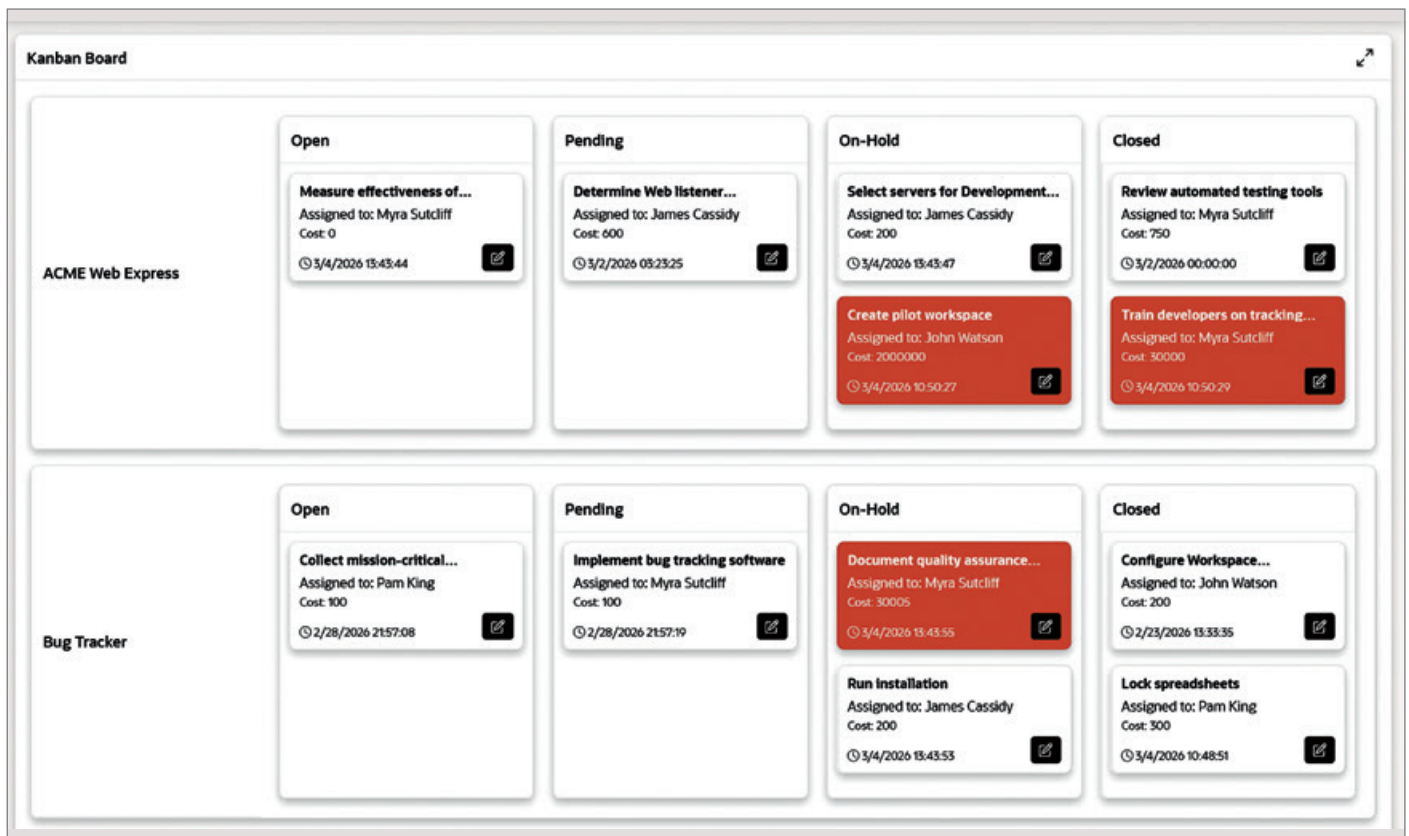


Abbildung 7: Kanban Board Plug-in (Quelle: Ronny Weiß)

derne Plug-in-Entwicklung ist KI inzwischen ein unverzichtbares Werkzeug, besonders wenn es darum geht, neue Ideen schnell auszuprobieren und in funktionierende Prototypen zu verwandeln.

Fazit

Diese Reise durch die Geschichte der APEX-Plug-ins macht deutlich, wie weit sich die Plattform entwickelt hat und wie vielfältig die Möglichkeiten inzwischen sind, APEX zu erweitern. Plug-ins fügen sich heute nahtlos in das bestehende Look-and-Feel ein. Aufgaben, die früher mühsam und manuell programmiert werden mussten, lassen sich dank moderner APIs schnell, professionell und sauber umsetzen. Spätestens mit den Template Component Plug-ins wurde die Entwicklung hochwertiger Plug-ins auch für Citizen Developers erreichbar, weil sich viele Anforderungen, ohne PL/SQL, ohne JavaScript, ohne komplexe Lifecycle-Logik lösen lassen.

Mit der Einführung von KI hat APEX einen weiteren großen Schritt gemacht. KI senkt die Einstiegshürden, beschleunigt die Entwicklung und eröffnet völlig

neue Möglichkeiten für Kreativität und Prototyping. Sie macht Plug-ins nicht nur zugänglicher, sondern auch vielseitiger. Die APEX_AI API erlaubt es Plug-ins, selbst KI-gestützte Funktionen einzubinden, und AI Configurations mit System Prompts und RAG-Daten schaffen die Grundlage für intelligente, kontextbewusste Plug-in-Features.

Aus meiner eigenen Arbeit weiß ich, wie wertvoll KI inzwischen ist. Ob beim Generieren von HTML Markup, beim Erstellen erster JavaScript-Funktionen oder beim schnellen Ausprobieren neuer Ideen – KI spart Zeit, schafft Freiräume und ermöglicht es, sich stärker auf Benutzerfreundlichkeit und Integration zu konzentrieren.

Das Kanban-Board-Plug-in für meinen DOAG-Vortrag ist ein gutes Beispiel dafür: Die KI hat mir den Rohbau geliefert und ich konnte meine Energie in die Feinheiten und die APEX-Integration stecken (siehe Abbildung 7).

Alles deutet darauf hin, dass KI die Plug-in-Entwicklung in APEX weiter prägen wird. Sie macht die Plattform offener, schneller und kreativer. Ich bin gespannt, welche neuen Plug-in-Features und Möglichkeiten in den zukünftigen Releases folgen werden.

Über den Autor

Ich arbeite als Softwareentwickler seit 2013 mit Oracle APEX und entwickle seit 2017 freie Open-Source Plug-ins für die Oracle APEX Community. Seit September 2021 bin ich im Oracle-APEX-Development-Team tätig.



Ronny Weiß
ronny.weiss@oracle.com
<https://apex.oracle.com>



Wie kommt mein GPS-Track nach APEX?

Wilfried Eigl

„Auch der weiteste Weg beginnt mit dem ersten Schritt.“ – Lao-tse.

Die ersten Schritte können allerdings sehr mühsam werden, wenn man ein neues Anwendungsgebiet betritt. Daher soll dieser Artikel einen Einstieg in die Arbeit mit Geodaten in der Oracle-Datenbank bieten, der auch für Anfänger geeignet ist. Als Beispiel habe ich Wanderungen gewählt, die ich mit einem GPS-Gerät aufgezeichnet habe und am Ende in APEX dargestellt werden.

Was sind GEODATEN

Geodaten sind Informationen, die Objekte, oder Gegebenheiten mit einer Position auf der Erdoberfläche beschreiben. Um solche Positionen eindeutig bezeichnen zu können, benötigen wir ein Koordinatensystem, das alle Punkte der Erdoberfläche eindeutig bezeichnen kann.

Es existieren sehr viele unterschiedliche Koordinatensysteme. Die Arbeitsgruppe EPSG (European Petroleum Survey Group) hat sich die Mühe gemacht alle Koordinatensysteme zu sammeln und eindeutige Schlüsselnummern für jedes einzelne zu vergeben, die als SRID (Spatial Reference Identifier) bezeichnet werden (siehe <http://epsg.org>).

GPS-Koordinaten

Ein Koordinatensystem, das weltweit gültig ist und sehr weit verbreitet ist, kennt man unter dem Namen GPS (siehe *Abbildung 1*). Die SRID für GPS ist 4326. Dieses System teilt die Erdkugel mit dem Äquator als Nulllinie des Breitengrades und einem Meridian, der durch das Observa-

torium in Greenwich (UK) verläuft (die dort im Boden markierte Linie befindet sich 102,5 Meter daneben) und als Nulllinie des Längengrades dient. Der Äquator teilt die Erde in eine Nord- und eine Südhalbkugel, die in 90 Grad aufgeteilt werden. Der Nordpol hat somit den Breitengrad 90° Nord. Der Längengrad teilt die Erde in eine West- und eine Osthalbkugel, die in 180 Grad aufgeteilt werden. Durch Angabe eines Längen- und eines Breitengrades kann also jede beliebige Position auf der Erde bezeichnet werden. Der Eingang zum Messezentrum Nürnberg Ost hat zum Beispiel die Koordinaten N 49,42 E 11,12. Dabei sollte man allerdings auch beachten, dass das GPS-Signal eine Ungenauigkeit von bis zu 10 Metern enthält.

Es gibt viele Softwareanwendungen und Geräte, die mit GPS-Daten arbeiten. Allerdings kann die Schreibweise der Koordinaten unterschiedlich sein. Meistens wird zuerst der Breitengrad (Latitude oder einfach LAT) und danach der Längengrad (Longitude oder einfach LON) angegeben. Auch die Angabe der Himmelsrichtung habe ich bereits vor der Koordinate oder auch nach der Koordinate gesehen. Ich bevorzuge als Schreibweise zuerst den Breitengrad, danach den Längengrad (LATLON) mit der Himmelsrichtung vor der Koordinate. Für die Koordinate selbst gibt es drei Schreibweisen:

- **Grad, Minuten und Sekunden (GMS)**
z. B. Navigationsgeräte
N 52° 31' 12" E 13° 24' 18"
- **Grad und Dezimalminuten (GMM)**
N 52° 13,200' E 13° 24,300'
- **Dezimalgrad (DG)** z. B. Google Maps
N 52,5200 E 13,4050

Die Umrechnung zwischen den drei Schreibweisen erfolgt durch Division oder Multiplikation mit 60. Hier ein Beispiel:

Umrechnung des Breitengrades von Dezimalgrad N 52,59781 nach GMS:

1. Der ganzzahlige Anteil ergibt die Gradangabe, also 52°
2. $0,59781 \cdot 60 = 35,8686$ ergibt die Minuten, also 35' oder 35,8686'
3. $0,8686 \cdot 60 = 52,115$ ergibt die Sekunden, also 52". Die Nachkommastellen können in der Regel ignoriert werden.

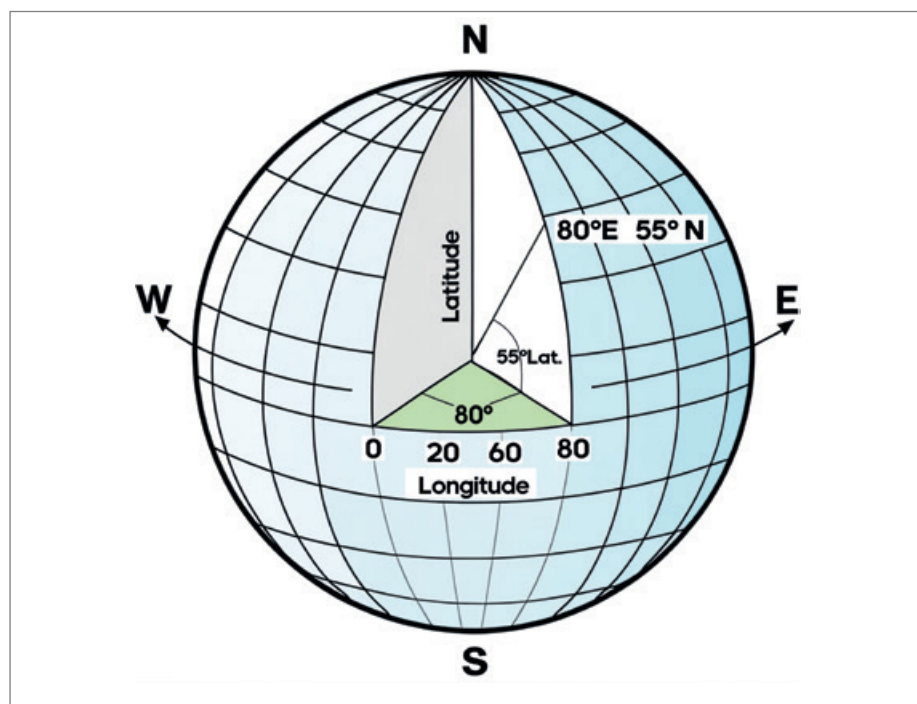


Abbildung 1: Das GPS-Koordinatensystem (Quelle: Wilfried Eigl)

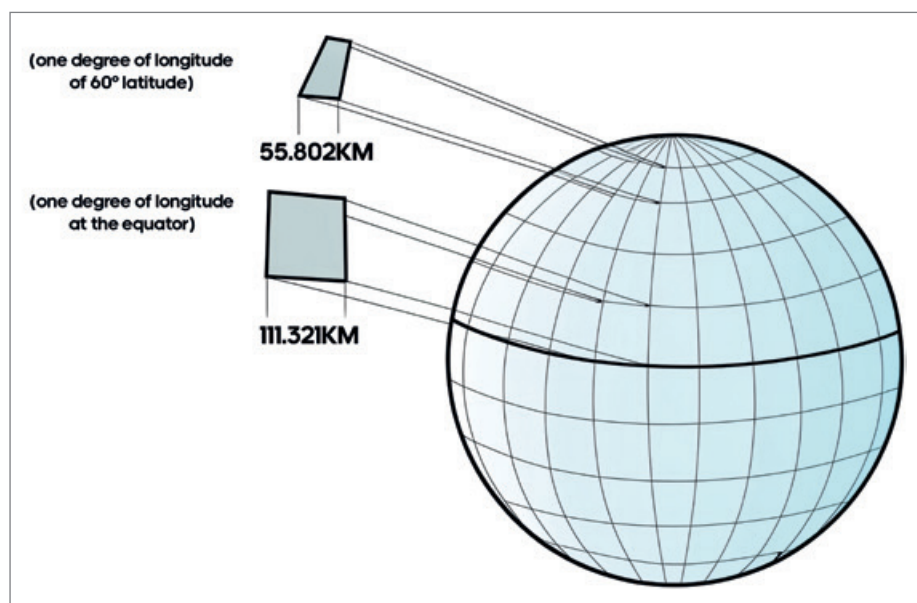


Abbildung 2: Die Breite eines Längengrades (Quelle: Wilfried Eigl)

Die GPS-Angabe von 52,59781 entspricht also einem Wert von 52 °35' 52".

Umrechnung des Breitengrades von GMS N 52 °35' 52" nach Dezimalgrad:

1. $52,115 / 60 = 0,8686 + 35 = 35,8686$ Minuten
2. $35,8686 / 60 = 0,59781 + 52 = 52,59781$ Stunden

Die GPS-Angabe von 52 °35' 52" entspricht also einem Wert von 52,59781.

Entfernungsberechnung zwischen zwei Koordinaten

Bei der Entfernungsberechnung gibt es mehrere Quellen für Ungenauigkeiten des Ergebnisses. Die Erde ist keine Kugel. Sie ähnelt eher einer Mandarine, wird also oben und unten flacher. Auch die Höhenmeter beeinflussen die Streckenlänge. Wer eine 100 m lange schiefe Ebene mit 5 % Steigung läuft, hat eine 12,5 cm längere Strecke als in der Ebene. Diese Fehlertoleranzen sind gering und

```
<?xml version="1.0" encoding="UTF-8"?>
<gpx xmlns="http://www.topografix.com/GPX/1/1">
  <name>Testfile</name>
  <trk> <name>ContentTest</name>
    <desc>Beispiel Wegstrecke</desc>
    <trkseg>
      <trkpt lat="49.773210" lon="8.930600">
        <ele>340.8</ele> </trkpt>
      <trkpt lat="49.772980" lon="8.930280">
        <ele>342.0</ele> </trkpt>
      <trkpt lat="49.772740" lon="8.929900">
        <ele>341.5</ele> </trkpt>
    </trkseg>
  </trk>
</gpx>
```

Listing 1: Beispiel einer .GPX-Datei

```
create table gpx_tab (gpx_id number primary key, xml xmltype
, xmldata xmltype, sdo sdo_geometry);

-- 1 - Erstellung eines XMLTPEs mit 3 Koordinaten
INSERT INTO gpx_tab (gpx_id,xml)
VALUES (1,xmldata(
'<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
<gpx xmlns="http://www.topografix.com/GPX/1/1">
  <metadata><name>SampleName</name></metadata>
  <trk><name>SampleName</name>
    <trkseg>
      <trkpt lat="49.795220" lon="9.002240"><ele>169.4</ele></trkpt>
      <trkpt lat="49.792408" lon="8.985649"><ele>167.0</ele></trkpt>
      <trkpt lat="49.790000" lon="8.980000"><ele>167.0</ele></trkpt>
    </trkseg>
  </trk>
</gpx>')));
commit;

-- 2 - Den Namen abholen (Element selektieren)
select track_name from gpx_tab t,
xmltable(
  xmlnamespaces('http://www.topografix.com/GPX/1/1' as "ns")
, '/ns:gpx/ns:trk' passing t.xml
  columns TRACK_NAME varchar2(20) path 'ns:name');

-- 3 - Teilbaum der Daten extrahieren
update gpx_tab set xmldata =
(select xmldata(substr(xmldata.getclobval(xml)
, instr(xmldata.getclobval(xml), '<trk>')
, instr(xmldata.getclobval(xml), '</gpx>')
- instr(xmldata.getclobval(xml), '<trk>'))
from gpx_tab );
commit;

-- 4 - Extraktion des Wertes vom Element /trk/name
select name
from gpx_tab t
,xmltable('/trk' passing t.xmldata
  columns name varchar2(20) path 'name');
--=> SampleName

-- 5 - Extraktion des Attributs LON aus /trk/trkseg/trkpt
select lon
from gpx_tab t
```

so kann bei der Berechnung einer Entfernung einfach von einer Kugel ausgegangen werden, was die Berechnung deutlich vereinfacht.

Bei der Berechnung ist allerdings wichtig, die Breite eines Längengrades zu berücksichtigen (siehe Abbildung 2). Am Äquator beträgt die Breite eines Längengrades 111,321 km. Im Norden auf Breitengrad 60 beträgt die Breite eines Längengrades allerdings nur noch 55,802 km ($111,3 \cdot \cos(\text{LAT})$). Des Weiteren benötigt die Oracle-SQL-Funktion `cos()` die Winkelangabe in Radiant ($1^\circ = \pi / 180$). Somit ergibt sich für eine Berechnung der Entfernung zwischen den Koordinaten (LAT1, LON1) und (LAT2, LON2) in Grad:

$$\begin{aligned} \text{rad_lat} &= (\text{LAT1} + \text{LAT2}) / 2 * \\ & 3,14159 / 180 \\ \text{dx} &= 111,3 * \cos(\text{rad_lat}) * \\ & (\text{LON1} - \text{LON2}) \\ \text{dy} &= 111,3 * (\text{LAT1} - \text{LAT2}) \\ \text{Entfernung in km} &= \sqrt{\text{dx} * \text{dx} + \text{dy} * \text{dy}} \end{aligned}$$

.GPX-Dateien

Die Koordinate eines Punktes kann man einfach mit Google Maps (maps.google.de) ermitteln. Durch einen Rechtsklick mit der Maus erscheint ein Kontextmenü, dessen oberster Eintrag die Koordinate des Punktes auf der Karte enthält. Diese kann durch Anklicken mit der Maus in die Zwischenablage übernommen werden.

Benötigt man eine Wegstrecke in einer Datei zur weiteren Verarbeitung, so kann man ein GPS-Gerät benutzen, um den selbst zurückgelegten Weg aufzuzeichnen. Für die Planung einer Wegstrecke gibt es mehrere Routenplaner im Internet, die es ermöglichen einen Weg zu planen, aufzuzeichnen und als Datei auf den eigenen Rechner zu laden (Download). Es gibt zwar mehrere Formate dafür, aber das mit Abstand häufigste Format ist eine .GPX-Datei (**GPS eXchange Format**).

Geodaten werden in einer .GPX-Datei in XML gespeichert. Die Datei besteht aus einem Kopfbereich mit allgemeinen Informationen zu der Weginformation, gefolgt von einem oder mehreren Wegabschnitten (Tracks). In Listing 1 sieht man das Beispiel einer .GPX-Datei einer Wegstrecke, die aus drei Punkten besteht.

```

,xmltable('/trk/trkseg/trkpt' passing t.xmldata
          columns lon varchar2(20) path '@lon');
--=>9.002240 ...

-- 6 - ! seltsamer Effekt bei der Extraktion aus XML-Dateien !
select lat,lon,ele
       from gpx_tab t
       ,xmltable('/trk/trkseg/trkpt' passing t.xmldata
                 columns lat varchar2(20) path '@lat',
                        lon varchar2(20) path '@lon',
                        ele varchar2(20) path 'ele');
--=> 49.795220    9.002240    169.4
--=> 49.792408    8.985649    167.0
--=> 49.790000    8.980000    167.0

-- 7 - Ups, da gibt es wohl einen Bug, der mit den deutschen
-- numeric characters zusammenhängt. Siehe auch ASKTOM
-- „problem with loading data from xml decimal point with
-- german nls settings“
-- Umstellung auf amerikanische numeric characters
alter session set nls_numeric_characters='.,';

-- Wenn man die Statements jetzt noch einmal ausführt, werden
-- die Daten korrekt angezeigt
--=> 49.795220    9.002240    169.4
--=> 49.792408    8.985649    167.0
--=> 49.790000    8.980000    167.0

-- 9 - Extraktion in PL/SQL aus einer Variablen
declare
  v_xml xmltype;
  v_track_name varchar2(50);
begin
  select xmldata into v_xml from gpx_tab;
  -- Track_name
  select track_name into v_track_name
        from xmltable('/trk' passing v_xml
                      columns TRACK_NAME varchar2(20) path 'name');
  dbms_output.put_line(v_track_name);
end;
/
--=> SampleName

```

Listing 2: Element aus XML extrahieren mit Angabe des Namespace

XML-Dateien bestehen aus mehreren Elementen, die wiederum durch je einen einleitenden Tag (Etikette) und einem schließenden Tag begrenzt werden. Die wichtigsten einleitenden Tags der .GPX-Datei sind in Listing 1 gelb markiert.

- <gpx> – Kopf und Weginformationen
- <name> – Bezeichner der Wegstrecke
- <trk> – Daten der Wegstrecke (track)
- <trkseg> – Wegstrecke bestehend aus einem oder mehreren Segmenten
- <trkpt> – Wegpunkte, die verbunden die Wegstrecke darstellen
- <ele> – Höhenmeter des Wegpunktes (trackpoint)

Der Name kann im <gpx>-Tag und auch im <trk>-Tag vorkommen. Welchen Namen ein GPS-Gerät anzeigt, oder eine Software auswertet, ist unterschiedlich. Wenn man an beiden Stellen den Namen einträgt, ist man immer auf der sicheren Seite.

Innerhalb eines Elements können sich auch noch Informationen befinden, die als Attribute bezeichnet werden. Beispiele für Attribute sind in Listing 1 mit grüner Farbe gekennzeichnet. Als Attribute werden hier Breitengrad (lat) und Längengrad (lon) von trackpoints abgespeichert. Elemente und Attribute werden unterschiedlich verarbeitet.

XML in Oracle

Für die Speicherung von XML-Daten gibt es in Oracle den Datentyp XMLTYPE. Ich nutze XML allerdings nur als Datenaustauschformat. Die Daten, die ich in der Datenbank verarbeite, werden von mir aus dem gelieferten XML extrahiert und relational in Tabellen abgespeichert. Allerdings speichere ich den Inhalt der .GPX-Datei noch als XMLTYPE in der Datenbank zusätzlich ab, um die gelieferten Inhalte nachträglich zu sehen oder an andere Stellen unverändert weitergeben zu können.

In Listing 2 sieht man die Extraktion des Wertes des Elementes /gpx/trk/name aus der .gpx-Datei in Listing 1. Da in der XML-Datei ein Namespace im Header angegeben ist, muss auch bei der Extraktion für die xmltable der genau gleiche Namespace angegeben werden.

Da ich .gpx-Dateien aus verschiedenen Quellen und eventuell auch mit verschiedenen Namespaces verarbeite, bin ich dazu über gegangen, das Element <trk> für meine Verarbeitung extra als XMLTPE zu speichern. Dadurch spare ich mir die Angabe des Namespaces bei der Verarbeitung.

Geodaten in Oracle mit SDO_GEOMETRY

Das SDO-Geometry-Objekt besteht aus fünf Komponenten[MR1.1].

- SDO_GTYPE NUMBER
- SDO_SRID NUMBER
- SDO_POINT SDO_POINT_TYPE
-- (x, y, z number)
- SDO_ELEM_INFO SDO_ELEM_INFO_ARRAY -- VARRAY of number
- SDO_ORDINATES SDO_ORDINATE_ARRAY -- VARRAY of number

Der **SDO_GTYPE** ist eine 4-stellige Zahl im Format DLTT

D – Anzahl der Dimensionen (2, 3, oder 4)
L – Nummer der Dimension, die den Messwert für eine LRS-Geometrie angibt
TT – Geometrie Typ (00 bis 09, 10 bis 99 reserved for future use)

Beispiel:

SDO_GTYPE = 2002 – 2 Dimensionen – kein LRS – LINE or CURVE
(Konstante SDO_LINESTRING2D)

SDO_ETYPE	SDO_INTERPRETATION	Meaning
0	(any numeric value)	Type 0 (zero) element. Used to model geometry types not supported by Oracle Spatial. For more information, see Type 0 (Zero) Element .
1	1	Point type.
1	0	Orientation for an oriented point. For more information, see Oriented Point .
1	$n > 1$	Point cluster with n points.
2	1	Line string whose vertices are connected by straight line segments.
2	2	Line string made up of a connected sequence of circular arcs. Each circular arc is described using three coordinates: the start point of the arc, any point on the arc, and the end point of the arc. The coordinates for a point designating the end of one arc and the start of the next arc are not repeated. For example, five coordinates are used to describe a line string made up of two connected circular arcs. Points 1, 2, and 3 define the first arc, and points 3, 4, and 5 define the second arc, where point 3 is only stored once.

Abbildung 3: Werte für SDO_ELEM_INFO (Quelle: Oracle)

```
update gpx_tab set sdo=(
select mdsys.sdo_geometry(
    SDO_LINestring2D,SDO_LONLAT,null
    ,mdsys.sdo_elem_info_array(1,2,1)
    ,cast(multiset(select CASE n.rn
        WHEN 1 THEN pt.lon
        WHEN 2 THEN pt.lat END ord
        from (select rownum rn
            ,to_number(LON) LON
            ,to_number(LAT) LAT
            from GPX_tab t
        ,xmltable('/trk/trkseg/trkpt' passing t.xmldata
            columns LAT varchar2(20) path '@lat'
            ,LON varchar2(20) path '@lon')) pt
        ,(select 1 rn from dual
        union all select 2 from dual) n order by pt.rn, n.rn
    ) AS mdsys.sdo_ordinate_array)) x
from dual);
```

Listing 3: Erstellung des SDO-Geometry-Objekts aus XML

Die **SRID** ist aus der Data Dictionary in Tabelle SDO_COORD_REF_SYS ersichtlich. Die SRID für das GPS-Koordinatensystem ist 4326. Für diese Zahl wurde von Oracle die Konstante SDO_LONLAT definiert.

Ein **SDO_Point** besteht aus drei NUMBER-Werten.

OBJECT (X NUMBER, Y NUMBER, Z NUMBER)

Dabei gelten für die drei Elemente SDO_POINT, SDO_ELEM_INFO und SDO_ORDINATES folgende Regeln:

- Wenn mehrere Punkte gespeichert werden:
 - SDO_POINT ist NULL.
 - SDO_ELEM_INFO enthält Werte
 - SDO_ORDINATES enthält Werte

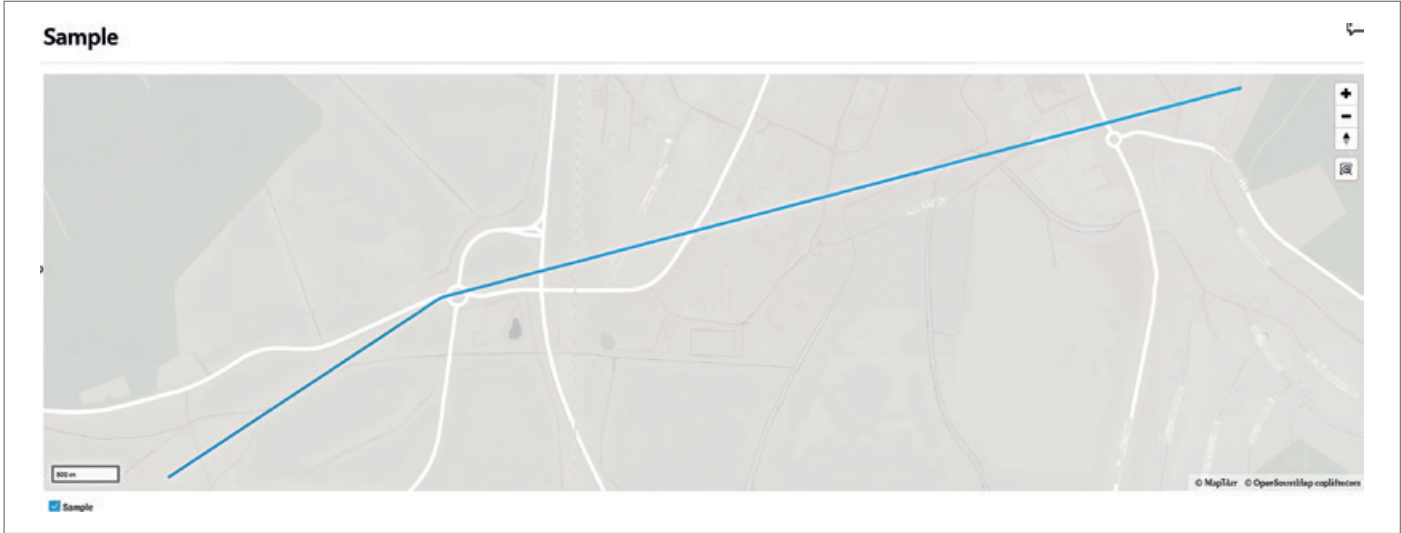


Abbildung 4: Die Darstellung der Wegstrecke in APEX (Quelle: Wilfried Eigl)

- Wenn nur ein einzelner Punkt gespeichert wird:
 - SDO_POINT enthält die Werte x, y, z
 - SDO_ELEM_INFO ist NULL.
 - SDO_ORDINATES ist NULL.

SDO_ELEM_INFO beschreibt, wie die Daten in SDO_ORDINATES zu interpretieren sind.

- SDO_STARTING_OFFSET Offset zu den relevanten Daten (meistens 1)
- SDO_ETYPE Typbezeichnung von ORACLE
- SDO_INTERPRETATION Strukturbeschreibung

Die Bedeutung ist in *Abbildung 3* aus der Oracle-Doku ersichtlich.

SDO_ORDINATES VARRAY enthält die eigentlichen Koordinaten für die Abbildung von Wegstrecken (SDO_ORDINATES VARRAY (1048576) of NUMBER).

2-dimensionale Werte (X1, Y1, X2, Y2, ... , Xn, Yn)

3-dimensionale Werte (X1, Y1, Z1, X2, Y2, Z2, ... , Xn, Yn, Zn)

2-dimensionale Koordinaten (LON1, LAT1, LON2, LAT2, ... , LONn, LATn)

Es ist hier besonders zu beachten, dass die Reihenfolge der Koordinatenwerte in der .GPX-Datei (LAT, LON) hier umgekehrt angegeben werden (LON, LAT).

Somit ergibt sich für die Darstellung meiner Wegstrecken folgendes SDO-Objekt:

```
MDSYS.SDO_GEOMETRY(2002 -- SDO_
GTYPE
,4326 -- SDO_SRID
,NULL -- SDO_POINT
,MDSYS.SDO_ELEM_INFO_ARRAY(1, 2,
1)
,MDSYS.SDO_ORDINATE_AR-
RAY(9.00224, 49.79522
,9.00207, 49.79475
,9.00191, 49.79447
,...
))
```

Starten von APEX

Um mit APEX zu arbeiten, muss ein APEX aktiv sein und ein Workspace existieren. Will man einfach ausprobieren, wie das alles funktioniert, dann empfehle ich, eine Oracle VirtualBox zu installieren und den Download einer Oracle Ap-

pliance, die man in der VirtualBox über *File->ImportAppliance* installieren kann. Damit erhält man eine komplette Oracle-Umgebung für Tests und Spielereien.

Wenn die Oracle Instance gestartet ist, kann man zunächst testen, ob APEX zur Verfügung steht, mit:

```
select status from dba_registry whe-
re comp_id = 'APEX';
```

Das Ergebnis sollte den Wert VALID liefern.

Die APEX-Oberfläche wird in einem Browser gestartet durch:

<http://localhost:8080/ords/>

Wenn eine neu installierte Appliance vorliegt, müssen noch einige User aktiviert werden:

```
alter user apex_listener account
unlock;
alter user apex_public_user ac-
count unlock;
alter user apex_rest_public_user ac-
count unlock;
```

Und ein Workspace, in dem man arbeiten kann, fehlt auch noch. Zur Erstellung eines eigenen Workspace meldet man sich zunächst intern an:

```
Workspace: internal
Username: admin
Passwort: oracle
```

Hier kann man dann einen neuen Workspace erstellen und dazugehörige User mit Administrationsberechtigung und Developmentberechtigung erstellen.

Ich habe für meine Tests einen Workspace OBE mit Admin OBE und Developer OBE erstellt.

Geodaten in APEX

Nach der Anmeldung an APEX kann ich nun die automatische Erstellung von Seiten nutzen, um einfach meine Wegstrecken aus der Tabelle GPX_TAB in APEX darstellen zu können.

- Workspace OBE [MR2.1] mit Administrator und Developer OBE
- Bei APEX anmelden

- select App Builder
- select Create a new App
- Name := Sample
- select Create Applications
- select Create Page
- select Map
- Name := Sample
- Table/View Owner := OBE
- Table/View_name := GPX_TAB
- select Next
- select Lines
- Geometry Column := SDO
- select Create Page
- Anmeldung an APEX als User OBE

Nach dem Start der App werden die Wegstrecken in einer Karte (*siehe Abbildung 4*) angezeigt, vorausgesetzt, dass eine Internet-Verbindung besteht, da die Karte geladen werden muss.

Fazit

Oracle APEX ist sehr gut geeignet, um Daten aus der Datenbank auf einer Oberfläche darzustellen. Der Umgang mit XMLTYPE und SDO_GEOMETRY ist allerdings nicht einfach.

Über den Autor

Wilfried Eigl war 35 Jahre lang als freiberuflicher Datenbankentwickler in Deutschland und der Schweiz tätig.



Wilfried Eigl
willi.e@gmx.net



Flows for APEX – BPMN-Prozessmodellierung meets Low-Code

Richard Allen, Flowquest

Einführung in Flows for APEX Flows for APEX entstand 2020 als Community-Projekt im APEX-Umfeld mit dem Ziel, eine Business-Process-Modelling- und Workflow-Engine zu entwickeln, die für APEX-Entwickler einfach und intuitiv nutzbar ist und sich nahtlos in ihre Anwendungen integrieren lässt. Flows for APEX nutzt den branchenüblichen und ISO-standardisierten Standard Business Process Model and Notation (BPMN) 2.0 zur Modellierung von Geschäftsprozessen. Der BPMN-Standard wurde 2011 von der Object Management Group (OMG) entwickelt, um verschiedene Prozessmodellierungswerkzeuge zu standardisieren, und hat sich seitdem in großen Unternehmen und Behörden weit verbreitet. Heute verfügen viele Entwickler, Business-Analysten und Manager über Erfahrung im Umgang mit BPMN zur Beschreibung von Geschäftsprozessen.

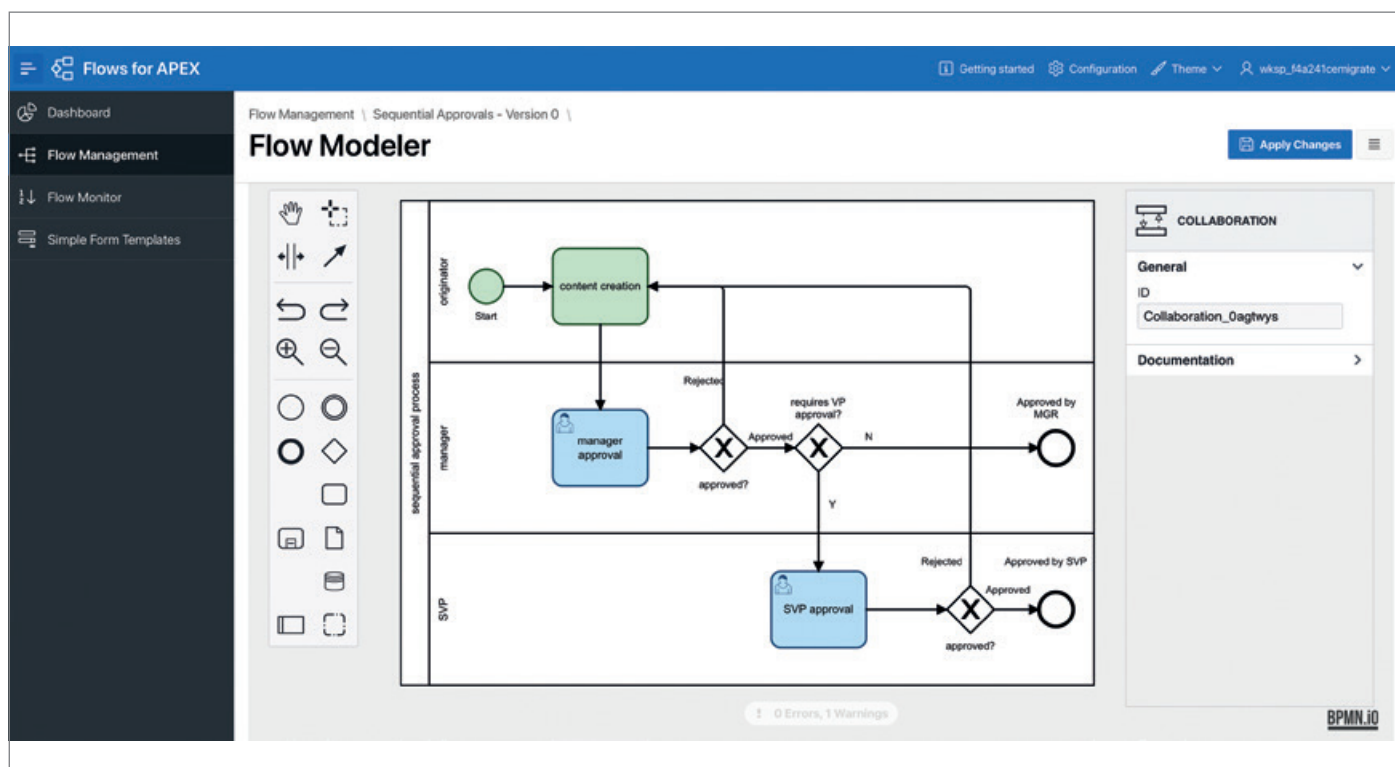


Abbildung 1. Ein einfacher Spesenabrechnungsprozess, modelliert im BPMN-Modellierer. Prozesse können modelliert, überprüft und getestet werden, bevor die Anwendungsentwicklung beginnt (Quelle: Richard Allen)

Was ist Flows for APEX?

Flows for APEX bringt professionelle Workflow-Funktionalitäten direkt in Oracle-APEX-Anwendungen. Im Kern bietet es eine vollständige BPMN-2.0-Process-Engine, die vollständig innerhalb der Oracle-Datenbank läuft – ideal für APEX-Entwickler im Oracle-Ökosystem.

Das Tool ermöglicht es Entwicklern, Geschäftsprozesse visuell mit standardisierter BPMN-Notation zu modellieren – derselben Notation, die auch in teuren Enterprise-BPM-Plattformen verwendet wird. Diese Modelle können anschließend direkt in APEX-Anwendungen ausgeführt werden, wobei die Workflow-Engine den Ablauf zwischen Benutzern, Systemen und automatisierten Aufgaben steuert.

Ein entscheidender Vorteil ist die enge Integration mit APEX:

- Prozessaufgaben können APEX-Seiten starten
- Workflow-Daten integrieren sich nahtlos in APEX-Elemente
- Geschäftslogik wird mit vertrautem PL/SQL umgesetzt

Es sind keine proprietäre Skriptsprache oder externen Workflow-Server notwen-

dig – alles läuft direkt in der APEX-Umgebung.

Typische Anwendungsfälle

BPMN-Workflows eignen sich besonders für Prozesse mit mehreren Schritten, Beteiligten oder komplexer Logik.

Genehmigungsprozesse:

Von einfachen Spesenfreigaben bis zu komplexen Beschaffungsprozessen. Flows for APEX übernimmt Routing, Eskalationen und vollständige Audit-Trails.

Case Management:

Kundendienstfälle, Incident-Management oder Beschwerdebearbeitung profitieren von strukturierten Abläufen mit gleichzeitig hoher Flexibilität.

Dokumenten-Lebenszyklus:

Erstellung, Prüfung, Freigabe und Archivierung lassen sich effizient modellieren, inklusive paralleler Reviews.

Onboarding und Offboarding:

Koordination von Aktivitäten über mehrere Abteilungen hinweg – BPMN sorgt für die richtige Reihenfolge.

Regulierte Prozesse:

BPMN bietet vollständige Nachvollziehbarkeit und Kontrolle – ideal für Compliance-Anforderungen.

Anwendungsbeispiele und Kunden

Flows for APEX wurde über 12.500-mal heruntergeladen und wird von Einzelentwicklern bis zu Großunternehmen genutzt.

Beispiel Bank:

Eine Retailbank nutzt Flows for APEX für Kreditvergabeprozesse.

- Einsatz in ~1.000 Filialen
- ~15.000 Nutzer
- Über 3.000.000 bearbeitete Kredite
- ~10.000 Prozesse täglich

Das System erwies sich als zuverlässig und performant.

Beispiel Unternehmen (USA):

Ein Verteidigungsdienstleister mit ~600 Mitarbeitern entwickelte über 40 Geschäftsprozesse mit BPMN.

Ergebnisse:

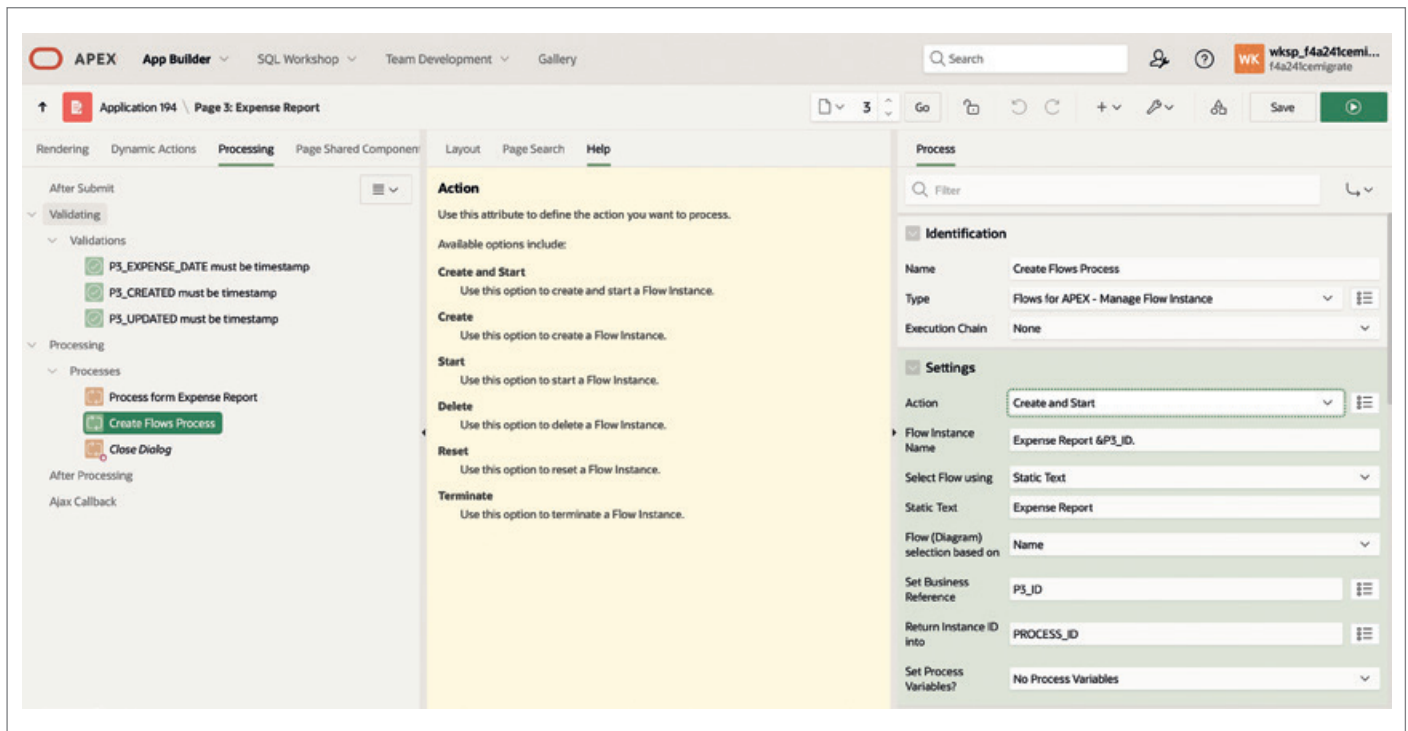


Abbildung 2. Eine Anwendung kann den Workflow mithilfe deklarativer Konfiguration über die Flows for APEX-Plug-ins starten oder schrittweise fortsetzen (Quelle: Richard Allen)

Severity	Performed On	User Name	Object	Event Type	Event Comment	Process Level	Subflow	Step Key	Object Type	Object Sub Type	BPMN Super Type
2	2026-03-23 21:39:03.292686000 +00:00	wksp_14a241ceml...	Process	created	Instance name set to Sequential Approvals - 23-MAR-2026 21:39:03				bpmmprocess		collaboration
2	2026-03-23 21:39:12.064399000 +00:00	wksp_14a241ceml...	Process	started					bpmmprocess		collaboration
2	2026-03-23 21:39:12.074412000 +00:00	wksp_14a241ceml...	content creation	current step		0	160	d.Bammi.kasZ	bpmmtask		activity
2	2026-03-23 21:39:16.793620000 +00:00	wksp_14a241ceml...	content creation	step completed		0	160	d.Bammi.kasZ	bpmmtask		activity
1	2026-03-23 21:39:16.792710000 +00:00	wksp_14a241ceml...	manager approval	error	Error setting process variable in process id (set), SQL error shown in event log.				bpmmuserTask		activity
1	2026-03-23 21:39:16.793817000 +00:00	wksp_14a241ceml...	Process	error	Error evaluating Setting. Expression is invalid. Expression .				bpmmprocess		collaboration
1	2026-03-23 21:39:16.816062000 +00:00	wksp_14a241ceml...	manager approval	error	Error creating APEX Human task GENERIC_APPROVAL in application 192. see debug for details.				bpmmuserTask		activity
2	2026-03-23 21:41:01.772292000 +00:00	wksp_14a241ceml...	Process	reset					bpmmprocess		collaboration
2	2026-03-23 21:41:32.343439000 +00:00	wksp_14a241ceml...	Process	started					bpmmprocess		collaboration
2	2026-03-23 21:41:32.3449052000 +00:00	wksp_14a241ceml...	content creation	current step		0	161	uOncipMTpG	bpmmtask		activity

Abbildung 3. Die Abläufe für APEX beinhalten eine detaillierte Protokollierung, die für langfristige Prüfprotokolle archiviert werden kann. Nicht standardmäßige Vorgänge werden zur Information der Administratoren hervorgehoben (Quelle: Richard Allen)

- höhere Transparenz
- bessere Auditierbarkeit
- gesteigerte Effizienz

Community Edition:

- Open Source
- Kostenlos
- Vollwertige BPMN-Engine

Community Edition und Enterprise Edition

Flows for APEX gibt es in zwei Varianten:

Enterprise Edition (seit 2024):

- Finanzierungsmodell für Weiterentwicklung

Erweiterte Features wie:

- Multi-Instance-Aktivitäten
- Message Flow
- Administrationsfunktionen (Pause, Zurückspulen, Fortsetzen)
- Geplante Features (Version 26.1):
 - Auslagerung langlaufender Tasks

Zusätzlich erhalten Kunden:

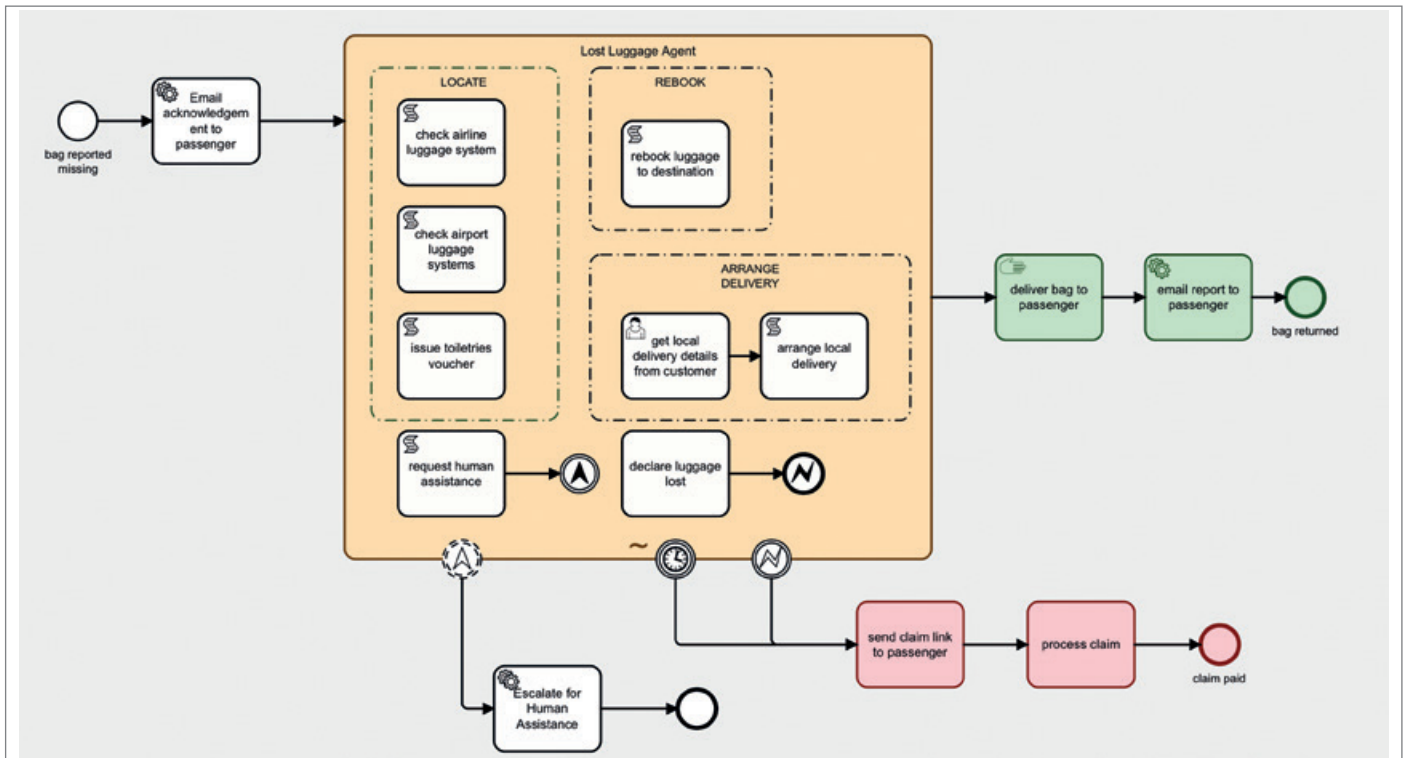


Abbildung 4. Ein Prozess zur Gepäckverlustmeldung an einem Flughafen, modelliert mithilfe eines BPMN-Ad-hoc-Subprozesses. Dieser kann von einem menschlichen Mitarbeiter genutzt oder als vollständig autonomer, KI-gesteuerter Workflow ausgeführt werden (Quelle: Richard Allen)

- Technischen Support
- Projektberatung

Einnahmen fließen in beide Editionen und sichern die Weiterentwicklung des gesamten Projekts.

BPMN im Zeitalter der KI

BPMN ist eine textbasierte Prozessdefinition (XML) und daher gut für KI verständlich. Moderne LLMs können BPMN-Dateien:

- generieren
- bearbeiten
- übersetzen
- analysieren

Einsatz von KI in Workflows:

1. Content-Erzeugung:

KI erstellt E-Mails oder Dokumente basierend auf Prozessdaten (inklusive RAG-Techniken).

2. Steuerung von Prozesspfaden:

KI kann Entscheidungen an BPMN-Gateways treffen und Prozesse dynamisch steuern.

3. Autonome Subprozesse:

In Version 26.1 können KI-Agenten eigenständig entscheiden:

- Welche Schritte ausgeführt werden und

– in welcher Reihenfolge.

Das führt zu einem neuen Ansatz:

- Klassisches BPM = feste Abläufe
- KI-gestütztes BPM = adaptive Prozesse

Die Kombination aus Struktur (BPMN) und Künstlicher Intelligenz (KI) schafft leistungsfähige hybride Systeme.

Fazit

Flows for APEX hat sich von einem Community-Projekt zu einer produktionsreifen Workflow-Plattform entwickelt.

Die Kombination aus:

- BPMN-Standard,
- enger APEX-Integration und
- KI-Funktionalität

macht es zukunftssicher.

Im KI-Zeitalter wird strukturierte Prozesssteuerung wichtiger, nicht weniger wichtig.

BPMN liefert Governance und Kontrolle, KI liefert Intelligenz – gemeinsam ermöglichen sie adaptive, intelligente Geschäftsprozesse.

Über den Autor

Richard Allen ist Gründer von Flowquest und Hauptentwickler der Enterprise Edition von Flows for APEX. Er arbeitet seit 1985 mit Oracle und war unter anderem Oracle Consultant, Director der Database Security Group und Mitglied des APEX Development Teams.



Richard Allen
richard@fowquest.net



Modernisierung einer Datenbank

Matthias Jakob, Nürnberger Versicherung

Wie bekommt man eine historisch gewachsene Datenbank, auf die heute mehrere Applikationen zugreifen, wieder auf einen modernen Stand? Der Artikel beschreibt den Ablauf, wie wir im laufenden Betrieb die Modernisierung der Datenbank durchgeführt haben. Im Fokus stehen die Herausforderungen und Lösungen, die wir nur durch enge Zusammenarbeit zwischen dem Applikationsentwicklungs- und dem Datenbank-Team meistern konnten.

Meine Geschichte beginnt, als ich die Datenbank im Jahre 2017 übernommen habe. Zu diesem Zeitpunkt ging ich voller Motivation an meine neuen Aufgaben. Vor circa 25 Jahren war diese Datenbank für genau eine Applikation entworfen worden.

In *Abbildung 1* sieht man den Zustand bei Übernahme der Datenbank.

Die ursprünglichen Wissensträger standen mir, zum Zeitpunkt der Übernahme, leider nur noch sehr kurz beziehungsweise nicht mehr zur Verfügung. Ich habe mich in diesem Zeitraum, nur noch auf die grundsätzlichen Fragen konzentriert, um den Betrieb aufrecht zu erhalten:

- Wo befindet sich die Dokumentation?
- Wie läuft das Tagesgeschäft?
- Wie ist die Arbeitsweise der beziehungsweise des Kollegen?

Die Antworten kurzgefasst: „Code ist Dokumentation genug“, und fränkisch, pragmatisch „Das System muss halt laufen...“

Was nun folgte war eine systematische Analyse der bestehenden Datenbanklandschaft. Diese erfolgte in mehreren Schritten:

1. Wie sieht das Datenmodell aus?
2. Wie kommen die Daten in die DB?
3. Wie greifen Applikation auf die Daten zu?
4. Wie wird die Datenbank von den Stakeholdern gesehen?

Rekonstruktion und Verständnis des Datenmodells

Mithilfe von SQL Developer und dem Data-Modeler habe ich versucht, das Daten-

modell zu erfassen beziehungsweise neu zu generieren. Dabei zeigte sich schnell:

- In der Datenbank war kein vollständiges, konsistentes Datenmodell vorhanden.
- Constraints fehlten weitgehend oder waren nur rudimentär definiert.
- Die Applikationen haben mehr Wissen über die Datenstrukturen als die Datenbank.

Untersuchung der ETL-Prozesse

Bei meiner Analyse der ETL-Prozesse waren die Elemente in der Datenbank klar zu sehen: rund 150 Packages. Anhand der Namensgebung war ersichtlich, was die Ziele der Packages waren: Unsere Vertragsdaten zum Beispiel aus Lebens- und Kranken-Verträgen in die Zieltabellen schreiben. So weit so gut. Jedes dieser Packages besaß nur eine Methode `Handle_Spartendaten`. Die Methoden enthalten circa 7.000 Zeilen Code im klassischen Row-By-Row-Design. Verdächtig war in diesem Umfeld auch das Error-Handling mit einem Parameter „Flag“ (siehe *Listing 1*).

Der Importstatus wurde als Return Code zurückgemeldet. Steven Feuersteins Exception-Handling „How NOT to Do“ [1] lässt grüßen. Die Erklärung für diese Art der Entwicklung erhielt ich bei einem zufälligen Gespräch in der Kaffeeküche mit einem HOST-Entwickler. Zitat: „Dein ehemaliger Kollege kam doch vom Großrechner, sieh dir doch JCL als Beispiel an...“ Dieses und die Tatsache, dass ich in der Datenbank keine Aufrufe der einzelnen Methoden finden konnte, ließ für mich nur einen Schluss zu. Die Steuerung der Importe erfolgt durch ein externes Tool.

Analyse der SQL-Zugriffe der Applikationen

Im nächsten Schritt untersuchten wir die von den Anwendungen erzeugten SQL-Statements. Aus dieser Analyse konnten wir zusätzliche Beziehungen und Abhängigkeiten zwischen Tabellen ableiten. Gleichzeitig traten neue Herausforderungen zutage:

- Bind-Variablen wurden in vielen Fällen nicht verwendet.
- Verschiedene Applikationen nutzen gleichen DB-User.

Einschätzung der Applikationsverantwortlichen

Zur Abrundung haben wir die Perspektive der relevanten Stakeholder eingeholt. Die Gespräche ergaben folgendes Bild:

- Entwickler sahen die Datenbank primär als reine Persistenzschicht. Die wesentliche Logik macht doch der ORM-Mapper (z. B. Hibernate).
- Product-Owner und Projektleitungen hatten häufig nicht klar erkannt, dass die Datenbank das technische und fachliche Fundament der Applikation darstellt.

Erkenntnisse der Analyse

Die Untersuchung lief parallel zur Weiterentwicklung und zog sich über circa 14 Monate. In dieser Zeit sind die strukturellen und technischen Schwachstellen deutlich aufgefallen:

- ETL-Steuerung außerhalb der Datenbank
- Lange Importzeiten wegen „Row-by-Row-Processing“
- Gemeinsame Nutzung identischer Applikationsbenutzer für Datenbankzugriffe
- SQL-Abfragen ohne Bind-Variablen
- Fehlende oder unzureichende Abstraktionsschichten
- Ein Datenmodell mit erheblichem Optimierungspotenzial
- Entwicklermentalität stark auf ORM statt auf Datenbankarchitektur ausgerichtet

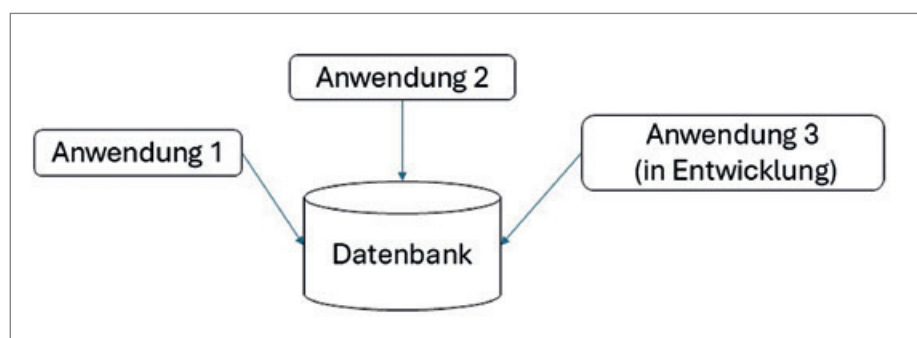


Abbildung 1: Übersicht Ausgangslage Applikationen/Datenbank (Quelle: Matthias Jakob)

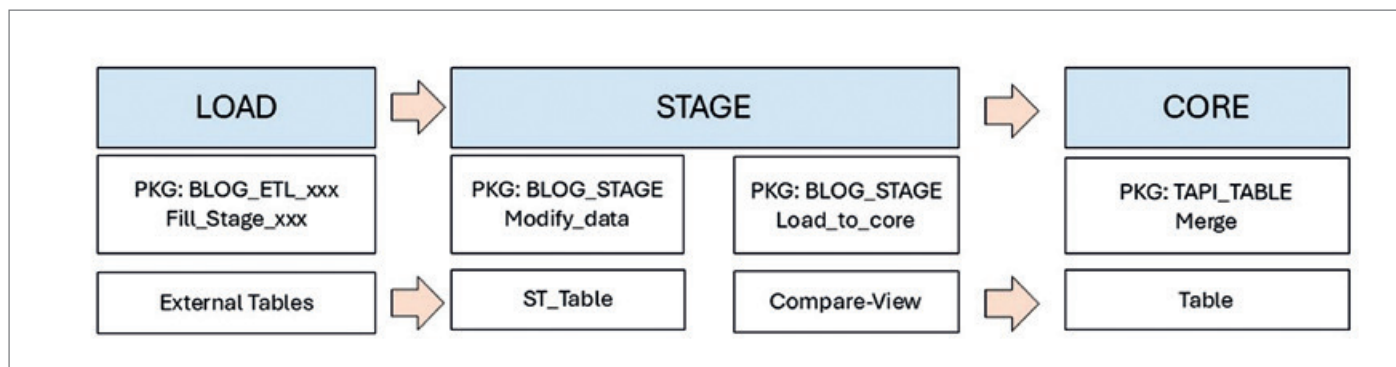


Abbildung 2: Aufteilung ETL-Datenfluss (Quelle: Matthias Jakob)

- Stakeholder mit wenig Verständnis für Wichtigkeit von Daten
- Dokumentationen nur teilweise oder gar nicht mehr vorhanden

Maßnahmen ableiten/Plan erstellen

Aus den gewonnenen Erkenntnissen haben wir folgende Maßnahmen abgeleitet.

1. ETL-Prozesse erneuern
2. Datenzugriffe und Datenablage neu regeln
3. Verständnis für Daten und die Datenbank verbessern
4. Interne Arbeitsabläufe verbessern

Neugestaltung der ETL-Prozesse

Die ETL-Strecken müssen grundlegend überarbeitet und modernisiert werden. Dabei stehen insbesondere folgende Aspekte im Fokus:

- **Steuerung innerhalb der Datenbank**, um Prozesse transparenter, stabiler und besser wartbar zu machen.
- **Priorisierung der Massendatenverarbeitung**, um lange Laufzeiten und ineffizientes Row-by-Row-Processing zu vermeiden.

Für die Steuerung der ETL-Prozesse bietet sich der interne Oracle Scheduler an. Wir konnten alle ETL-Strecken ohne größere Aufwände in die Datenbank-Hoheit überführen. Seit dem Einsatz des Oracle Schedulers sind die Aufwände beim Monitoring und der Problemanalyse deutlich gesunken. Technisch konn-

ten wir damit einfache Jobs, aber auch komplexere Job-Ketten realisieren. Ein Blick in die Oracle-Dokumentation [2] lohnt sich.

Der Umbau hin zur Massendatenverarbeitung erwies sich als anspruchsvoller als zunächst erwartet – weniger aus technischen Gründen, sondern vor allem aufgrund gewohnter Denkmuster, die abgelegt werden mussten. Für das DB-Team gilt nun ein neuer Ansatz:

1. SQL First
2. PL/SQL-Bulk-Verarbeitung
3. Bist du sicher?!

Die Umstellung auf den Ansatz „SQL First“ in Kombination mit dem Prinzip **LOAD → STAGE → CORE** führte zu deutlichen Verbesserungen, sowohl im Code als auch im allgemeinen Verständnis der ETL-Prozesse.

Die klare Trennung (siehe Abbildung 2) der Ebenen schafft Transparenz darüber, welche Verarbeitungsschritte in welcher Schicht stattfinden dürfen und sollen. Die Datenladung zwischen STAGE und CORE erfolgt über standardisierte TAPI Packages, die für jede CORE-Tabelle eine entsprechende MERGE-Methode bereitstellen. Diese enthält ein vollständiges SQL-MERGE-Statement (Insert/Update/Delete) inklusive DML-Error-Logging. Im Importprozess werden die Daten zwischen STAGE und CORE in einer Compare-View verglichen, so dass ausschließlich fachlich geänderte Datensätze in den CORE geschrieben werden. Die Idee für den Einsatz der Compare-Views entstand durch einen prägenden Spruch eines externen Kollegen aus dem Jahre 2004: „Oracle-Performance bedeutet: Denke in Blö-

cken.“ Dieses ETL-Konstrukt bietet den Vorteil, dass es über einen Codegenerator erzeugt werden kann. Bei circa 200 CORE-Tabellen spart man sich viel Arbeit...

Klare Trennung von Daten-, Logik- und Anwendungsschemata

Innerhalb der Datenbank bildet die Schema-Trennung (siehe Abbildung 3) nun die Grundlage für Flexibilität, Wartbarkeit und Sicherheit:

- **DB App User** pro Applikation nach dem Prinzip „Least Privilege“ zur Minimierung von Zugriffsrisiken
- **View Schicht**, um Applikationen konsistente und abstrahierte Datenzugriffe zu ermöglichen.
- **Logikschema** für Prozeduren, Packages und Geschäftslogik
- **Datenschema** für tabellennahe Objekte

Zu beachten: **NUR** die App-User haben in der Datenbank die CONNECT-Rechte, das Logik- und Daten-Schema sind reine „SCHEMA ONLY“-User. Also nur „Container“ ohne CONNECT-Rechte! Soweit die Theorie. In der Umsetzung hat sich gezeigt, dass die Probleme nicht in der Technik liegen. Wir mussten hier langwierige Diskussionen mit den Entwicklern und Projektverantwortlichen führen, um die Vorteile dieser Architektur zu vermitteln.

Die Reihenfolge war aus unserer Sicht sehr einfach:

1. App-User anlegen und Applikation umstellen (Datasource anpassen)
2. View-Schicht für Applikation anlegen (Entitys anpassen)

2026
DOAG
Konferenz + Ausstellung

Die **ORACLE**
Anwenderkonferenz

Nürnberg | 17. – 20. Nov.

#DOAG2026



anwenderkonferenz.doag.org



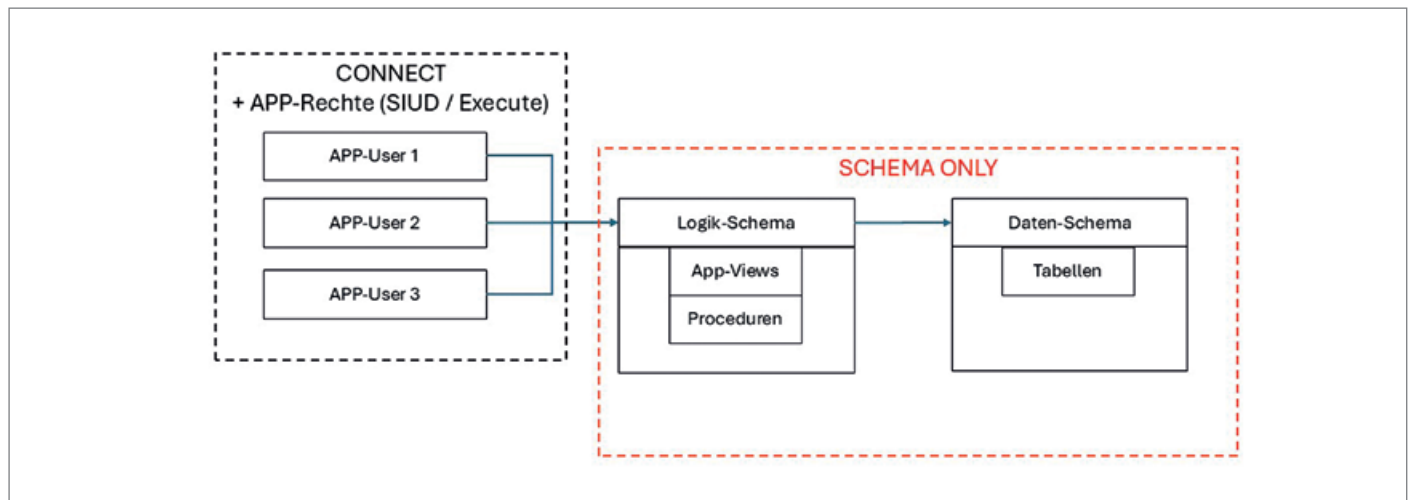


Abbildung 3: Moderne Schema-Trennung (Quelle: Matthias Jakob)

3. Logik- und Datenschema trennen

Dagegen standen die Widerstände/Bedenken:

- Funktioniert dann alles noch?
- Wir haben aber keine Zeit für einen Komplettest der Applikation!!! Ein Test dauert doch x-Wochen.
- Aber es läuft doch so auch... Warum ändern?

Als Datenbank-Team haben wir zwei „Taktiken“ verwendet, um die Änderungen doch umsetzen zu können.

Taktik 1: Weg des geringsten Widerstandes

Wir konnten die jungen Teammitglieder einfacher überzeugen, sich auf diese Veränderungen einzulassen. Vorteile: Sie waren neugierig und haben die Vorteile dieser Architektur erkannt.

Taktik 2: Zwangsumstieg

Im anderen Fällen mussten wir Entwickler zum Umstieg „zwingen“. Unsere Lösung: Neue fachliche Anforderungen in der Applikation bedingen bei uns im Versicherungsumfeld häufig neue Datenfelder. Als DB-Entwickler haben wir diese nur in der View-Schicht für den neuen App-User bereitgestellt. Die Anwendung war daher gezwungen auf den User und die View-Schicht umzustellen.

Mindsets – Entwickler-Teams

Während der Umsetzung haben wir angefangen einen regelmäßigen Austausch zwischen den JAVA-Entwicklern und den

DB-Entwicklern zu etablieren. Unser Ziel war, Verständnis auch für die andere Seite zu entwickeln.

Wir haben hier den Begriff „**SQL as a Service**“ eingeführt. Kurzfassung: Sobald ein Problem mit Daten zu tun hat, reden „Datenbank“ und „Applikation“ **sofort** miteinander und finden **gemeinsam** eine Lösung. Die Erfahrung zeigt, sobald es um „Datenlogik“ geht, sind Lösungswege im DB-Umfeld das Mittel der Wahl.

Mindsets – Projekt-Management

Wir haben es über lange Gespräche geschafft, dass wir nicht mehr „nur“ eine Datenbank sind. Wir konnten das Verständnis „Datenbank = stabiles Fundament“ für die Applikationen vermitteln. Daraus folgt, dass Anforderungen nun über gere-

gelte Wege an das DB-Team gestellt werden müssen.

Mindsets – DB-Team

Unsere Aufträge werden immer zusammen im Team besprochen. Die Dokumentationen sind nun Teil jedes Auftrages und enthalten nicht nur WAS und WIE etwas gemacht wurde, sondern viel wichtiger auch das **WARUM**. Wir erhoffen uns davon, dass zukünftige Teammitglieder unsere Beweggründe verstehen und besser beurteilen können.

Fazit

Wir konnten unsere Datenbank mit den durchgeführten Maßnahmen wieder in einen moderneren Zustand bringen. Ent-

```

PROCEDURE handle_leben (
  up IN OUT common.parameter,
  gesamt_bearbeitet IN OUT PLS_INTEGER,
  gesamt_ok IN OUT PLS_INTEGER,
  gesamt_fehler IN OUT PLS_INTEGER,
  flag IN OUT PLS_INTEGER
)
BEGIN
  LOOP
    ...
  END LOOP;
  flag := 0 --Alles OK
  EXCEPTION
    WHEN OTHER THEN
      flag := -1; --Rückmeldung an Taskserver
  END handle_leben;

```

Listing 1: Verdächtiges Exceptionhandling

scheidend war dabei nicht primär die Technik – moderne Oracle-Datenbanken bieten umfangreiche Möglichkeiten, die in vielen Fällen schlicht nicht genutzt werden –, sondern das Mindset im Umgang mit der Datenbank als zentralem Bestandteil der Gesamtarchitektur.

Für die Applikationsteams bedeutete dies vor allem:

- Wissensaufbau für SQL- und Oracle-Datenbanken,
- ein vertieftes Verständnis der Wechselwirkungen zwischen Anwendung und Datenbank
- sowie die Bereitschaft, Datenbankprinzipien in die Entwicklungsprozesse zu integrieren.

Als Datenbankteam wiederum profitieren wir von klareren Verantwortlichkeiten:

- Daten und Anforderungen stehen schneller zur Verfügung
- Änderungen am Datenmodell liegen wieder in der Hoheit des DB-Teams

- die Einarbeitungszeit neuer Mitarbeiter verkürzt sich deutlich durch ein konsistentes Datenmodell und nachvollziehbare Dokumentation

Insgesamt zeigt sich: Eine nachhaltige Datenbankarchitektur entsteht nicht allein durch technologische Maßnahmen, sondern durch ein gemeinsames Verständnis aller beteiligten Teams über die Bedeutung der Datenbank als Fundament jeder modernen Anwendung.

Quellen

- [1] <https://stevenfeuersteinonplsql.blogspot.com/2017/02/now-not-to-handle-exceptions.html>
- [2] <https://docs.oracle.com/en/database/oracle/oracle-database/19/admin/scheduling-jobs-with-oracle-scheduler.html>
- [3] Alfons Zimmermann Firma: simple fact 2004

Über den Autor

Matthias Jakob ist ein langjähriger IT Spezialist in der Nürnberger Versicherung. Erste Erfahrungen konnte er mit Oracle 8.1.7 sammeln. Seine Schwerpunkte liegen in Bereich DB-Entwicklung, Architektur und Testmanagement.



Matthias Jakob
matthias.jakob@nuernberger.de
@80smatthiasjakob.bsky.app

Oracle Datenbanken Monthly News

Auf dem deutschsprachigen Oracle-Blog ist die April-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 Beiträge zu gewährleisten.

<https://www.doag.org/de/home/news/oracle-datenbanken-monthly-news-56/>



Pluggable Database Snapshots von der Stange

Henner Hucke, Biotronik

Pluggable Database Snapshots und ASM passen einfach nicht zusammen. Diese Aussage ist fast immer richtig, allerdings gibt es eine wichtige Ausnahme: Auf einer Exadata mit SPARSE-Diskgruppen funktionieren beide Technologien überraschend gut zusammen. In diesem Artikel wird beschrieben, wie Sie Snapshot PDBs auf ASM automatisiert bereitstellen können.

Biotronik stellt die Softwarequalität seines Home-Monitoring-Systems, einer selbst entwickelten und betriebenen Java-Applikation, unter anderem auch mit einer großen Anzahl von automatisierten Unit-, Integrations- und Systemtests sicher. Vor jeder einzelnen Ausführung eines Tests muss der Zustand des Da-

tenbankschemas auf den definierten Anfangszustand zurückgesetzt werden. Das Home-Monitoring-System wird aktuell, beginnend mit der Softwareentwicklung, in die Cloud migriert. Die „On-Premises“-Datenbankumgebungen werden hierbei auf einer Exadata (Oracle Database@<Hyperscaler>) konsolidiert.

Dieser Umzug erfordert auch eine geänderte Bereitstellung der Datenbankschemata für die Tests. Das bisher genutzte Tool Datapump soll hierfür nicht mehr genutzt werden. Die neue Methode für die Bereitstellung der Datenbankschemata muss in der Lage sein, etwa 2000 Umgebungen am Tag bereitzustellen. Von die-

sen 2000 Umgebungen werden circa 20 Umgebungen für länger laufende Testzyklen (mehrere Stunden Laufzeit) genutzt, die restlichen Umgebungen bestehen in der Regel nur für einen kurzen Zeitraum, im Durchschnitt etwa eine Minute.

Logische Aufteilung der Exadata

Jede der „On-Premises“-Datenbankumgebungen (Development, System-Integration, Validierung, Performancetest und Produktion) wird in Form eines VM-Clusters implementiert. Ein VM-Cluster besteht hierbei aus einer RAC-Datenbank auf zwei virtuellen Maschinen mit zwei virtuellen Cloud-Netzwerken (VCN). Ein VCN dient zur Anbindung an den Object-Store für die Ablage der Backups, das zweite VCN stellt die Verbindung der Datenbank zu den Applikationsressourcen in der Cloud des ausgewählten Hyperscalers sicher. Bei der Erstellung eines VM-Clusters können Sie festlegen, ob SPARSE-Diskgroups erstellt werden sollen. Die prozentuale Aufteilung des zur Verfügung stehenden Speichervolumens auf die Diskgruppen DATA, RECO und SPARSE ist fest vorgegeben und kann leider nicht beeinflusst werden. Die Aktivierung lokaler Backups führt zu einer Aufteilung von 35% für DATA, 50% für RECO und 15% für SPARC. Ohne aktiviertes lokales Backup stehen für Data 60%, für RECO 20% und für SPARC 20% des Nettodatenvolumens zur Verfügung. Ein VM-Cluster, welchem Sie 2 Terabyte ASM-Speicher zugewiesen haben, benötigt wegen der dreifachen Redundanz circa 6 Terabyte Plattenspeicher. Effektiv zur Verfügung stehen davon circa 1224 GB in der DATA, 374 GB in RECO und 374 GB in der SPARC-Diskgruppe. Im asmcmd-Tool wird für die SPARC-Diskgruppe eine Größe von 4080 GB angezeigt, dabei handelt es sich allerdings um eine virtuelle Größe, effektiv sind nur 374 GB verfügbar.

Vorbereitungen im ASM

Vor der Erstellung von Snapshot-PDBs muss zuerst die Zugriffskontrolle für die Daten- und Sparse-Diskgruppen aktiviert und zweitens der Betriebssystem-User ORACLE als Besitzer aller neu in der

Diskgruppe erstellten Dateien festgelegt werden (*siehe Listing 1*).

Der OS-User ORACLE ist allerdings den Datendateien eventuell bereits bestehender PDBs nicht als Besitzer zugeordnet. Sofern Sie einen Snapshot-Klon einer solchen PDB erstellen wollen, muss dies der Fall sein. Sie können sich die hierzu notwendigen SQL-Befehle in einer mit der CDB verbundenen SQL-Plus-Session erstellen (*siehe Listing 2*) und die so erstellten SQL-Statements dann in einer mit der ASM-Instanz verbundenen SQL-Plus-Session ausführen.

Jetzt ist im ASM alles vorbereitet und Sie können in einer CDB-Datenbanksession Snapshots einer READ-ONLY geöffneten PDB erstellen (*siehe Listing 3*).

Erst mit der Erstellung eines Snapshots werden im ASM die Berechtigungen auf den Datenfiles der Quell-PDB von Read-Write auf Read-Only geändert. Sie müssen diese Berechtigungen explizit wieder auf Read-Write ändern, um zum Beispiel die Quell-DB nach dem Löschen aller Snapshot-Klone wieder im Read-Write-Modus öffnen zu können.

Bereitstellung von PDB Snapshots mit PL/SQL

Alle SQL-Kommandos für den Lebenszyklus von Snapshot-PDBs wie auch gewöhnlichen PDBs im Rahmen der Softwareentwicklung sind im Anwendungsfall in einem PL/SQL Package gekapselt. Hierdurch lässt sich die Funktionalität einfach in eine CI/CD integrieren, beziehungsweise ist der Schulungsaufwand für die Softwareentwickler geringer. Der direkte Aufruf der Funktionen des PL/SQL-Packages erfordert allerdings weiterhin eine Datenbanksession mit administrativen Privilegien (SYSDBA oder SYSOPER oder SYSBACKUP oder SYSDBG). Möchte man die Anmeldung von Java-Entwicklern beziehungsweise der CI/CD an der Datenbank mit administrativen Privilegien vermeiden, ist ein Umweg für den Aufruf der PL/SQL-Routinen notwendig. Ein solcher Umweg besteht zum Beispiel in einem Shell-Skript, welches sich per SQL*Plus zur Datenbank mit administrativen Privilegien verbindet und die gewünschte PL/SQL-Routine ausführt. Das Shell-Skript wiederum kann dann mittels eines externen DBMS-Scheduler Jobs oder einer Ja-

va-Stored-Procedure aufgerufen werden. Im Falle der Umsetzung mit einer Java-Stored-Procedure benötigt der Java-Entwickler beziehungsweise die CI/CD nur einen Datenbankaccount mit CREATE-SESSION-Privileg und dem EXECUTE-Privileg für die Java-Stored-Procedure. Im Anwendungsfall wurde das Shell-Skript auf dem durch die VM-Cluster bereitgestellten „Advanced Cluster File System“ zentral abgelegt und kann somit von beiden RAC-Knoten ausgeführt werden.

Vergleich der Performance von Snapshot PDB und klassischer PDB (*siehe Tabellen 1-3*)

Die Nutzung von Snapshot PDBs für die Bereitstellung von Datenbankumgebungen für die Softwareentwicklung hat sich bereits bewährt. Vor dem Einsatz im Alltag der Softwareentwicklung wurden mehrere Versuchsreihen mit jeweils 50 Wiederholungen durchgeführt. Für die Bereitstellung einer Snapshot PDB wurden dabei im Durchschnitt circa 18 Sekunden und für das Löschen circa 8 Sekunden benötigt. Die Größe der Quell-PDB hat dabei, im Gegensatz zu einer klassischen PDB, nur einen geringen Einfluss auf die Zeitdauer für die Bereitstellung des Snapshots. Die Bereitstellung einer 80 Gigabyte großen klassischen PDB benötigte die 4-fache Zeit der Bereitstellung der Snapshot-PDB.

Der Vergleich der Snapshot PDB gegenüber der klassischen PDB bezüglich der Anwendungsperformance wurde einmal mittels einer lese- und schreibintensiven PL/SQL-Routine (schreibintensive Datenmigration in ein geändertes Datenmodell) und einmal mittels des Swingbench-Benchmarks durchgeführt. Die Datenmigrationsroutine war auf einer Snapshot PDB im Durchschnitt etwa 15% langsamer als auf einer klassischen PDB. Die Anzahl der Transaktionen je Sekunde des Swingbench (SOE 20 * 20K Transaktionen) unterschieden sich nur geringfügig.

Größere Unterschiede zwischen den beiden PDB-Typen gab es beim Export mittels Datapump. Der Import in eine Snapshot PDB war im Durchschnitt 10% langsamer, der Export der zuvor importierten Daten aus einer Snapshot PDB

```
-- you need to be connected as sysasm
-- enable file access control
ALTER DISKGROUP <disk-group> SET ATTRIBUTE 'access_control.enabled' = 'true';
-- add OS user oracle to file access control
ALTER DISKGROUP <disk-group> ADD USER 'oracle';
-- check if user is added
select
dg.group_number, dg.name dg_name, u.os_name
from v$asm_user u, v$asm_diskgroup dg
where u.group_number = dg.group_number;
```

GROUP_NUMBER	DG_NAME	OS_NAME
1	<disk-group>	oracle
3	<disk-group>	oracle

Listing 1: Aktivierung Zugriffskontrolle und Festlegung Besitzer neu im ASM angelegter Dateien

```
-- execute in CDB (as SYSDBA)
SELECT 'alter diskgroup '
|| substr(file_name, 2, regexp_instr(file_name, '\/') - 2)
|| ' set ownership owner='''oracle''' for file '''
|| file_name
|| ''';' as cmd
FROM CONTAINERS(dba_data_files);

CMD
-----
alter diskgroup <disk-group> set ownership owner='oracle'
for file '+DATA1/.../DATAFILE/system.264.1195739907';
alter diskgroup <disk-group> set ownership owner='oracle'
for file '+DATA1/.../DATAFILE/users.265.1195739919';
...
```

Listing 2: Erstellen der SQL-Befehle für die Zuordnung des User ORACLE zu Datenfiles bereits existierender PDBs

```
-- execute in CDB (must be connected AS SYSDBA or AS SYSBACKUP or AS SYSDG or AS SYSOPER)
ALTER SESSION SET db_create_file_dest = "<Sparse-Disk-Group; z.B. +SPRC2>";
CREATE PLUGGABLE DATABASE snappy_pdb FROM mother_pdb SNAPSHOT COPY KEYSTORE IDENTIFIED BY "****";
ALTER PLUGGABLE DATABASE snappy_pdb OPEN READ WRITE INSTANCES=ALL;
ALTER PLUGGABLE DATABASE snappy_pdb SAVE STATE INSTANCES=ALL;
```

Listing 3: Erstellen einer Snapshot PDB

Kommando	PDB-Typ/Größe	Mittelwert
Create	Snapshot PDB (20 G)	17,6 s
Create	Klassische PDB	26,1 s
Create	Snapshot PDB (80G)	18,0 s
Create	Klassische PDB (80G)	84,4 s
Drop	Snapshot PDB (20 G)	8,6 s
Drop	Klassische PDB	4,2 s
Drop	Snapshot PDB (80G)	5,5 s
Drop	Klassische PDB (80G)	2,4 s

Tabelle 1: Vergleich der Performance von Snapshot PDB und klassischer PDB

Anwendung	PDB-Typ	Mittelwert
Datenmigration	Snapshot PDB	15,1 Minuten
Datenmigration	Klassische PDB	13,2 Minuten
Swingbench	Snapshot PDB	266,2 Transaktionen / s
Swingbench	Klassische PDB	265,3 Transaktionen / s

Tabelle 2: Vergleich der Performance von Snapshot PDB und klassischer PDB bei Datenmigration und Swingbench-Benchmarks

Anwendung	PDB-Typ	Mittelwert
Import	Snapshot PDB	40,5 Minuten
Import	Klassische PDB	36,8 Minuten
Export	Snapshot PDB	23,9 Minuten
Export	Klassische PDB	15,4 Minuten

Tabelle 3: Vergleich der Performance von Snapshot PDB und klassischer PDB bei Import und Export

war allerdings circa 55% langsamer als der Export aus der klassischen PDB.

Unerwartet

Bei der Nutzung von Snapshot PDBs und ASM-Storage ist zu beachten, dass vor dem Löschen einer klassischen PDB, welche mindestens einmal als Vorlage für Snapshot PDBs gedient hat, die Berechtigungen der Datenfiles von Read-Only mittels `dbms_dnfs.restore_datafile_permissions(pdb_name=>'NAME_DER_PDB')` wieder auf Read-Write zu ändern sind. Der Befehl `DROP PLUGGABLE DATABASE <x> INCLUDING DATAFILES` wird zwar auch ohne vorherige Ausführung von `dbms_dnfs.restore_datafile_permissions` ohne Fehlermeldung ausgeführt, allerdings werden dann die Datendateien im ASM nicht gelöscht. Im Alert-Log der Datenbank findet man hierzu einen ORA-15260-Fehler für jedes Datenfile. Eine Beschreibung zu `dbms_dnfs.restore_datafile_permissions` finden Sie leider nur im Knowledge-Base-Dokument KB106004. Auch nach der Änderung der Zugriffsrechte der Datendateien auf Read-Write wurden diese bei Ausführung des Befehls `DROP PLUGGABLE DATABASE <x> INCLUDING DATAFILES` nicht gelöscht. Im Alert-Log der Datenbank findet sich jetzt allerdings ein ORA-15028 (Datenfile noch im Zugriff). Der im November 2025 erstellte Service-Request

ist, Stand März 2026, immer noch im Status „Development Working“. Die Datenfiles einer solchen PDB werden nur dann gelöscht, wenn Sie die Datenbank zwischen der Ausführung von `dbms_dnfs.restore_datafile_permissions` und dem `DROP PLUGGABLE DATABASE <x> INCLUDING DATAFILES`-Befehl durchstarten. Das Patchen einer Containerdatenbank mit Snapshot-PDBs erfordert das vorherige Löschen aller Snapshot-PDBs und die Wiederherstellung der Read-Write-Zugriffsrechte auf den Datenfiles der klassischen PDBs, da ansonsten DATAPATCH mit einem Fehler bezüglich der fehlenden Schreibberechtigungen für die Datenfiles abbricht.

Fazit

Die Snapshot-Copy-Funktion für PDBs genügt im gegebenen Anwendungsfall den Anforderungen. Es ist möglich, innerhalb von 12 Stunden die benötigten ~2000 PDBs zu erstellen und nach Ausführung des Unit-Tests auch wieder zu löschen. Die für das Management der PDBs erstellten PL/SQL-Routinen lassen sich problemlos in die bestehende CI/CD integrieren und erfordern keine administrativen Privilegien für die Ausführung. Die PL/SQL-Routinen wurden auch um eine Funktionalität für das Löschen der Snapshot-PDBs vor dem Patchen und die Wiederherstellung der PDBs nach dem Patchen erweitert,

was den Arbeitsaufwand für die DBAs beim Patchen verringert. Die schlechtere Performance von Snapshot-PDBs bei schreibintensiven Operationen spielt im Anwendungsfall keine Rolle, die ausgeführten Tests prüfen in erster Linie funktionelle Aspekte des Codes und die erzeugte Schreiblast auf der Datenbank ist sehr gering. Das bestehende Problem mit den nicht gelöschten Datenfiles kann aktuell noch toleriert werden, da die Quell-PDBs für die Snapshots selten gelöscht werden müssen.

Quellen

- [1] Exadata PDB Sparse Clones siehe <https://blogs.oracle.com/exadata/exadata-pdb-sparse-clones>
- [2] Alter-Pluggable-Database-Kommando siehe <https://docs.oracle.com/en/database/oracle/database/26/sqlrf/ALTER-PLUGGABLE-DATABASE.html>



Henner Hucke
henner.hucke@biotronik.com



DOAG Datenbank Kolumne: Oracle SQL ausprobieren mit FreeSQL

Ulrike Schwinn, Oracle

Es gibt eine neue Möglichkeit, SQL auszuprobieren, ohne Software zu installieren – FreeSQL.com!

FreeSQL ist eine kostenlose browserbasierte Oberfläche auf Basis einer vollwertigen Oracle-Datenbank. Installation oder Konfiguration sind nicht erforderlich. *FreeSQL.com* ist der Nachfolger von *LiveSQL.oracle.com* und stellt eine überarbeitete, erweiterte Plattform zur Verfügung.

Funktionen im Überblick:

- Nutzung des gewohnten SQL Worksheet Interface
- SQL-Skripte können erstellt, ausgeführt und geteilt werden.
- Zugriff auf zahlreiche Community-Beispiele und verschiedene Beispielschemata, wie HR (Human Resources), CO (Customer Orders), OLYM (Olympic Data) oder World Data (World).
- Unterstützung sowohl für Oracle Database 19c als auch die neue Oracle AI Database 26ai.

Eine wichtige Neuerung ist der schreibgeschützte (Read-Only) Modus, der auch ohne Oracle-Anmeldung funktioniert. Hier können SQL-Abfragen (lesende) an den bereitgestellten Beispielschemata durchgeführt werden. Darüber hinaus können auch eigene Daten (z.B. Tabellen) nach Anmeldung im eigenen persönlichen Schema gespeichert werden.

Neu ist die Möglichkeit, das eigene FreeSQL-Schema per SQL*Net direkt von lokalen Tools und IDEs (z.B. NodeJS, Python, Java, Go, SQLcl, SQL Developer for VS Code) anzubinden. Über die Schaltfläche „Connect with...“ in der Symbolleiste lassen sich

die erforderlichen Connection-Informationen abrufen und ein Passwort generieren. Es werden alle notwendigen Details (wie Beispiele für SQLcl und andere Umgebungen) bereitgestellt.

Beispiel für SQLcl:

```
sql 'ULRIKE_SCHWINN_SCHEMA_XYZ' <password> //db.freesql.com:1521/26ai_un3c1'
```

Beispiel für Python:

```
import oracledb
import os
from dotenv import load_dotenv
load_dotenv()
local_dsn = "db.freesql.com" + ":" + "1521" + "/" + "26ai_un3c1"
connection = oracledb.connect(
    user="ULRIKE_SCHWINN_SCHEMA_XYZ",
    password="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx",
    dsn=local_dsn)
```

Nach erfolgreichem Verbindungsaufbau kann dann lokal entwickelt, getestet und gearbeitet werden, als handele es sich um jede andere Oracle-Datenbank. Mit diesen Funktionen ist FreeSQL keine reine Lernplattform mehr, sondern eine leichtgewichtige, ortsunabhängige Entwicklungsumgebung auf Oracle-Datenbankbasis.



AI-First-Organisationen: Der Praxisleitfaden

Dr. Benjamin Linnik

Die Begeisterung für Künstliche Intelligenz ¹⁾ in der Geschäftswelt ist kaum zu ignorieren. In Vorstandssitzungen wird eifrig diskutiert, KI-Budgets werden verteilt, und jede Abteilung soll „endlich KI nutzen“. Doch hinter dem Vorhang dieses technologischen Goldrausches wächst eine stille Ernüchterung. Viele Organisationen investieren massiv in KI-Tools und Talente, sehen aber kaum Verbesserungen in ihrer Unternehmensbilanz. Die versprochene Revolution bleibt aus.

Falls Ihnen das bekannt vorkommt, sind Sie nicht allein. Die Kluft zwischen KI-Hype und geschäftlicher Realität ist gewaltig – und das geht den meisten Unternehmen so. Dieser Artikel fasst die wichtigsten Erkenntnisse aus aktuellen Studien von BCG [1], McKinsey [2], Deloitte [3] und meine eigenen zusammen und zeigt, warum die meisten Unternehmen bei KI noch immer das Ziel verfehlen – und was Vorreiter anders machen.

Warum KI bei den meisten nicht funktioniert

Der größte Fehler, den Organisationen begehen, ist ein grundlegendes Missverständnis darüber, was eine echte KI-Transformation bedeutet. Es gibt einen entscheidenden Unterschied zwischen „Digitally-Enhanced“ und „AI-First“.

„Die größte Gefahr in Zeiten des Umbruchs ist nicht der Umbruch selbst; es

ist, mit der Logik von gestern zu handeln.“
– Peter Drucker

Digitally-Enhanced ist der Weg, den die meisten Unternehmen einschlagen. Man ergänzt bestehende, auf Menschen ausgerichtete Prozesse mit KI-Tools. Eine KI hilft einem Sachbearbeiter, Akten schneller zu prüfen, oder unterstützt jemanden im Marketing beim Verfassen von Texten. Dieser Ansatz ist weit verbreitet und bringt durchaus Vorteile – typischerweise

10 bis 15 Prozent Produktivitätssteigerung über den gesamten Prozess.

AI-First bedeutet etwas völlig anderes: Man gestaltet ganze Prozesse von Grund auf neu, mit autonomen KI-Agenten als Herzstück. Es geht nicht darum, den alten Weg schneller zu machen – es geht darum, einen neuen, besseren Weg zu gehen. Das Ergebnis: keine kleinen Verbesserungen, sondern echte Durchbrüche. Laut Boston Consulting Group kann dieses Modell eine 34-fache Steigerung des Umsatzes pro Mitarbeiter, 90 Prozent niedrigere Kundenakquise-Kosten und dreifach höhere Produktivität ermöglichen (siehe Abbildung 1 und Tabelle 1).

„AI-First bedeutet nicht, KI punktuell auf einzelne Aufgaben anzuwenden und dasselbe Ergebnis zu erwarten. Es bedeutet, ganze Prozesse völlig neu zu denken – um Ergebnisse zu erzielen, die vorher schlicht nicht möglich waren.“

Der Weg zum AI-First-Ansatz ist aber keine technische Herausforderung – sondern eine menschliche. Und damit kommen wir zur eigentlichen Hürde.

Der wahre Engpass: Nicht Technologie, sondern Köpfe

Viele Führungskräfte machen Legacy-Systeme oder Datensilos für ihren langsamen Fortschritt verantwortlich. Doch die größte Hürde für KI-Erfolg ist organisatorischer Natur, nicht technologischer. Ein aktueller Deloitte-Report bringt es auf den Punkt:

„Die meisten Unternehmen transformieren sich mit der Geschwindigkeit des organisatorischen Wandels, nicht mit der Geschwindigkeit der Technologie.“

Diese Transformation erfolgreich zu meistern erfordert mehr als neue Fähigkeiten – es braucht ein neues Denken. BCG-Strategen beschreiben vier zentrale Verhaltensänderungen, die dafür nötig seien (siehe Tabelle 2).

Das ist deshalb so schwierig, weil es die grundlegende Kultur verändert, wie Arbeit erledigt, bewertet und organisiert wird. Es zeigt: Die Technologie ist der einfache Teil. Wie Menschen denken und arbeiten – das ist die eigentliche Herausfor-

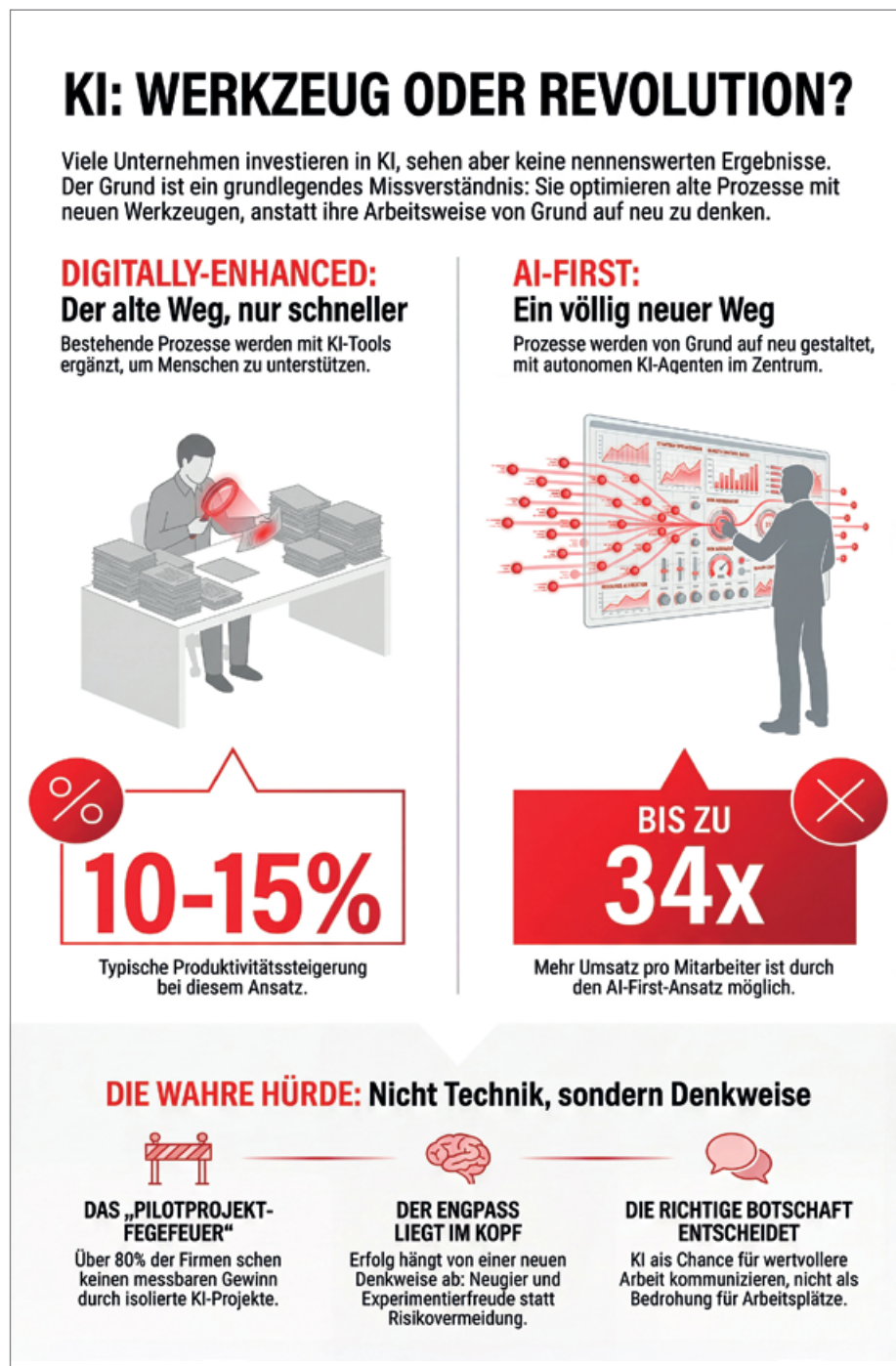


Abbildung 1: KI – Werkzeug oder Revolution? (Quelle: Dr. Benjamin Linnik)

derung der KI-Revolution. **Fähigkeiten kann man in Wochen vermitteln. Neue Denkweisen brauchen Monate.**

Warum so lange? Weil Organisationen über Jahre Gewohnheiten aufbauen, die sich nur mühsam ändern lassen. Ähnlich wie sich technische Schuld im Code ansammelt (und irgendwann mit Zinsen bezahlt werden muss), häufen sich organisatorische Schulden an: Silodenken, starre Prozesse, eine Kultur der Risikovermeidung. Eine von Risikoscheu geprägte Kultur wird „Fail Fast“

niemals leben. Abgeschottete Abteilungen wehren sich gegen Zusammenarbeit. **Wer einen schlechten Prozess automatisiert, bekommt keinen guten – nur einen schnelleren schlechten Prozess.**

Auf die Botschaft kommt es an

Veränderung erzeugt Angst – es sei denn, die Führung gibt ihr einen Sinn. Wenn die Geschäftsleitung ein Narrativ wählt, das

rein auf Effizienz und Kostenoptimierung setzt – etwa:

„Wir führen KI ein, **um Kosten zu senken** und wettbewerbsfähig zu bleiben. Einige **Stellen könnten betroffen sein.**“

...dann löst das sofort Angst und Unsicherheit bei den Mitarbeitern aus. Diese Abwehrhaltung führt direkt zu Widerstand, Desinteresse und talentierten Leuten, die das Unternehmen aufgrund mieser Stimmung verlassen. Mitarbeiter, die glauben, dass die KI sie ersetzen soll, werden dem System sogar feindlich gesonnen sein – Fehler nicht melden und nach Gründen suchen, warum die KI scheitern sollte. Damit ist die Initiative von Anfang an zum Scheitern verurteilt.

Die Alternative? Ein klares Change-Narrativ und transparente Führung. Das wirksamere AI-First-Narrativ stellt die Veränderung nicht als Stellenabbau, sondern als Chance dar:

„Wir bauen eine AI-First-Organisation, weil unsere Kunden und Mitarbeiter **Besseres verdienen**. Kunden verdienen schnelleren, intelligenteren Service. Mitarbeiter verdienen Arbeit, die ihr **Urteilsvermögen und ihr Hirn** nutzt, nicht stupide Routineaufgaben. KI-Agenten übernehmen die Routinearbeit. Menschen übernehmen das Denken und die strategische Planung. **Gemeinsam mit den KI-Agenten** erreichen wir Ergebnisse, die keiner allein schaffen könnte.“

Dieses Framing weckt positive Emotionen: Sinn, Wachstum, Chancen und Verantwortung. Sie fördert Engagement, Bindung und Zusammenarbeit. Es handelt sich dabei um dieselbe Transformation – aber eine **völlig andere emotionale Wirkung**.

Natürlich muss dieses Narrativ durch Taten untermauert werden: massive Investitionen in Weiterbildung, interessantere Rollen mit Fokus auf Orchestrierung und Strategie, klare Zusagen für sanfte Übergänge. Sonst verspielt die Führung jedes Vertrauen!

Laut McKinsey positionieren „AI High Performer“ KI als Wegbereiter für neue Geschäftsmodelle – nicht nur als Kosten-senker. Sie optimieren für schnellere Entscheidungszyklen: Ein Unternehmen, das den SODA-Zyklus (Scan-Orient-Decide-Act) in zwei statt acht Wochen durchläuft,

überholt durch Innovationen selbst klügere Wettbewerber – Wettbewerbsfähigkeit durch Geschwindigkeit, die sich mit der Zeit auszahlt.

Die meisten Unternehmen stecken im „Pilotprojekt-Fegefeuer“

Das wohl deutlichste Symptom einer fehlenden KI-Strategie: die Kluft zwischen breiter Nutzung und echtem geschäftlichen Nutzen. McKinsey-Daten zeigen, dass 78 Prozent der Organisationen KI in mindestens einer Funktion nutzen – aber nur 21 Prozent ihre Workflows grundlegend neugestaltet haben.

Viele Unternehmen sind in die Falle geraten, eine „Perlenkette ohne roten Faden“ zu schaffen: eine Sammlung isolierter KI-Experimente und Pilotprojekte, die einzeln beeindruckend aussehen, aber ohne ein verbindendes Leitbild. Ein Chatbot im Kundenservice, ein Prognosetool im Finanzwesen, ein Automatisierungsskript in der Personalabteilung.

Die Konsequenz: ein dramatischer Mangel an Geschäftswert. Dieselbe McKinsey-Studie zeigt, dass **über 80 Prozent der Organisationen keinen messbaren EBIT-Effekt auf Unternehmensebene** durch ihre KI-Nutzung verzeichnen – das direkte Ergebnis isolierter Optimierungen statt echter Transformation.

Was fehlt, ist ein klares Leitbild – eine strategische Vision, die beantwortet: *Wie sieht AI-First konkret für unser Unternehmen aus? Welche Prozesse gestalten wir komplett neu? Was wird von KI-Agenten übernommen, was bleibt beim Menschen?* Ohne diese Antworten startet man dutzende Experimente, wechselt von Tool zu Tool, jagt dem nächsten Trend hinterher – aber es fehlt der rote Faden (für die Perlen).

Interessanterweise korreliert **CEO-Verantwortung für KI-Governance** am stärksten mit positivem EBIT-Impact – doch nur 28 Prozent der Unternehmen haben sich so organisiert. Wer KI als reines IT-Projekt an die Technik-Abteilung delegiert, behandelt eine strategische Transformation wie ein Software-Update. Das Ergebnis: Die IT optimiert die Technologie, während das Geschäftsmodell unverändert bleibt. Man kann ein Strategie-Defizit nicht durch bessere Ausführung wettmachen. Ohne kla-

res Leitbild stecken Unternehmen im ewigen „Pilotprojekt-Fegefeuer“ fest – sie geben Geld aus, ohne je transformative Ergebnisse zu erzielen.

Rollen im Wandel: Von der Ausführung zur Orchestrierung

Das AI-First-Modell verschiebt den Wert menschlicher Arbeit grundlegend: weg von effizienter Ausführung, hin zu strategischer Aufsicht. Während KI-Agenten Routineaufgaben übernehmen – Ticket-Triage, Log-Analyse, Standard-Deployments –, werden **Richtungsvorgabe, Orchestrierung und Qualitätskontrolle** zu den neuen Kernaufgaben und rücken menschliches Urteilsvermögen und kritisches Denken in den Vordergrund. Das Ergebnis sind schlankere, funktionsübergreifende Organisationen mit flacheren Hierarchien. Was das konkret bedeutet, zeigt sich besonders deutlich in der Softwareentwicklung.

Von kleiner Automatisierung zu exponentiellem Wachstum

Die AI-First-Reise beginnt nicht mit einem Big Bang, sondern mit einer kleinen End-to-End-Transformation (E2E). Statt KI breit auf den kompletten „Kundenservice“ anzuwenden, starten Sie mit einem abgegrenzten Prozess – etwa Routineanfragen unter 1.000 Euro Risikowert. Hier darf der KI-Agent oder das trainierte Modell autonom entscheiden, Menschen übernehmen das Monitoring.

Diese erste Einführung dient als Testfeld:

- 1. Vertrauen aufbauen:** Mitarbeiter (jetzt *KI-Kuratoren*) erleben, wie der Agent arbeitet, was funktioniert und was noch nicht – Zusammenarbeit mit KI statt Konkurrenz.
- 2. Lernen und verfeinern:** „Fail Fast, Learn Fast“-Mentalität mit Feedback-Schleifen zur Verbesserung
- 3. Umfang erweitern:** von zum Beispiel 1.000 auf 10.000 Euro und komplexere Szenarien

Das funktioniert nur mit der richtigen Infrastruktur: Dokumentation als Arbeits-

Aspekt	Digitally-Enhanced	AI-First
Ansatz	KI-Tools zu bestehenden Prozessen hinzufügen	Prozesse um KI herum völlig neu denken
Produktivitätsgewinn	10–15 % Steigerung	Bis zu 34x mehr Umsatz pro Mitarbeiter
Art der Veränderung	Schrittweise Verbesserung	Grundlegende Neugestaltung
Fokus	Tool-Einführung	Kultur- und Denkwandel

Tabelle 1: Unterschiede zwischen Digitally-Enhanced und AI-First

Fähigkeiten	Denkweise
Wie man KI-Tools benutzt	Neugier statt Gewissheit
Wie man Prompts verbessert	Kontinuierliches Lernen statt Expertentum
Wie man KI-Agenten überwacht	Zusammenarbeit mit KI statt Konkurrenz
Wie man KI-Ergebnisse interpretiert	Experimentierfreude statt Risikovermeidung

Tabelle 2: Vier zentrale Verhaltensänderungen laut BCG-Strategen

anweisung, Lessons Learned als externes Gedächtnis, Quality Gates als Sicherheitsnetz und einem Live-Monitoring mit Anomalieerkennung. Der Zyklus – ausführen, beurteilen, lernen, verbessern – schafft Zinseszins-Effekte.

Wer diesen Weg konsequent geht, erreicht einen kritischen Wendepunkt – meist zwischen Monat 18 und 24 –, an dem die Dynamik von Top-down zu Bottom-up kippt. Ab diesem Punkt setzen Teams Neues schneller um, als die Führung genehmigen kann. Teams hören auf zu fragen „Warum?“ und fragen stattdessen „Was noch?“.

Der neue Entwickler: Vom Programmierer zum KI-Kurator

Die Rolle des Softwareentwicklers wandelt sich grundlegend: vom Programmierer zum KI-Kurator. Andrej Karpathy, ehemaliger Tesla-KI-Direktor und OpenAI-Mitgründer, prägte im Februar 2025 den Begriff „Vibe Coding“ – eine Methode, bei der Entwickler „den Code vergessen“ und sich auf das Formulieren der Anforderungen in natürlicher Sprache konzen-

trieren [4]. Statt Syntax zu beherrschen, beschreiben Entwickler ihre Absichten klar und lassen KI-Systeme den Code generieren.

Diese Herangehensweise zeigt bereits beeindruckende Erfolge:

- Mercor erreichte im März 2025 mit nur 30 Mitarbeitern 100 Millionen US-Dollar Jahresumsatz – und verzeichnete bis Oktober 2025 bereits 500 Millionen US-Dollar Umsatz [7].
- Cursor schaffte als schnellstes SaaS-Unternehmen überhaupt 100 Millionen US-Dollar ARR (Annual Recurring Revenue, jährlich wiederkehrender Umsatz) in nur 12 Monaten nach Launch (Januar 2025), mit weniger als 50 Mitarbeitern [8].

Die Dirigenten-Analogie

Diese Transformation gleicht der Rolle eines Dirigenten: Er kennt jedes Instrument und sein Potenzial, führt aber nicht jeden einzelnen Ton aus. Stattdessen formt er durch Vision und Anleitung ein harmonisches Gesamtwerk, das größer ist als die Summe seiner Teile (siehe Tabelle 3).

Konkrete Schritte für Entwickler:

Wer CI/CD, Infrastructure-as-Code und Configuration-as-Code bereits lebt, kennt das Prinzip: explizit machen, was implizit war. AI-First führt dieses „Everything as Code“-Denken konsequent weiter – jetzt werden auch Best Practices, Lessons Learned und Code-Stilregeln zu „Code“.

- **Experimentieren Sie mit KI-Coding-Tools** wie GitHub, Copilot, Claude Code, OpenAI Codex oder Cursor. Der Einstieg dauert Wochen, nicht Jahre – nur wer anfängt, entwickelt das Gespür für Stärken und Grenzen dieser Systeme.
- **Etablieren Sie Documentation-as-Code:** READMEs werden zu KI-Prompts. Dokumentieren Sie nicht nur was der Code macht, sondern warum. KI-Systeme können Kontext nicht erraten.
- **Implementieren Sie Quality Gates:** Automatisierte Tests, die jeden Bug-Fix begleiten. Der TDD-Zyklus (Red-Green-Refactor) funktioniert auch mit KI – nur schneller. Entscheidend ist dabei die Überwachung makroskopischer Metriken – statt reiner Nachverfolgung

Traditionelle Entwicklung	AI-First-Entwicklung
Manuelles Codeschreiben	Natürliche Sprache zu Code („Vibe Coding“)
Separate Dokumentation (Sharepoint, Confluence)	Documentation-as-Code (READMEs)
Manuelle Testerstellung	KI-generierte Tests, Quality Gates
Manuelles Debugging	Autonome Fehlererkennung und -behebung
Menschliche Peer Reviews	KI-gestützte Reviews mit Feedback-Schleife
Lineares Entwicklungstempo	Exponentielles Produktivitätswachstum

Tabelle 3: Unterschiede zwischen traditioneller Entwicklung und AI-First-Entwicklung

jeder Einzelentscheidung des Modells [9]. Ihre Best Practices und Erfahrungen sind gefragt denn je.

- **Führen Sie „Lessons Learned“-Dateien ein:** Erfolgreiche Muster und gescheiterte Ansätze werden zum externen Gedächtnis Ihres KI-Systems – genau wie CI/CD-Pipelines zum externen Gedächtnis für Build- und Deployment-Prozess wurden.

Empfehlungen für Führungskräfte

Ihre Aufgabe ist nicht, Detailregelungen vorzugeben, sondern den strategischen Rahmen zu setzen und ein Umfeld zu schaffen, in dem KI-Systeme sicher wachsen können.

Konkrete Maßnahmen:

- **Definieren Sie das Leitbild:** Ohne klare strategische Vision machen Teams isolierte KI-Experimente – das „Pilotprojekt-Fegefeuer“.
- **Identifizieren Sie geeignete Prozesse:** Suchen Sie nach Abläufen mit hohem Volumen, klaren Regeln und niedrigem Risiko – ideale Kandidaten für die erste E2E-Transformation. Beispiele: Routineanfragen, Standardgenehmigungen, wiederkehrende Berichte.
- **Definieren Sie klare Entscheidungsrichtlinien:** Legen Sie fest, was KI-Agenten autonom entscheiden dürfen und wo Menschen eingreifen müssen. Eine bewährte Struktur: Routine-Entscheidungen unter einer Risi-

koschwelle (z. B. unter 1.000 Euro) laufen autonom; mittlere Fälle erfordern menschliche Freigabe; komplexe oder strategische Entscheidungen bleiben beim Menschen.

- Definieren Sie Erfolg anhand von **Systemmetriken:** Zuverlässigkeit, Geschwindigkeit, Anpassungsfähigkeit.
- Investieren Sie in **Weiterbildungsprogramme.**
- Entwickeln Sie ein **klares Change-Narrativ**, das Chancen statt Bedrohung kommuniziert.
- Beginnen Sie mit einer **kleinen E2E-Transformation**, bauen Sie Vertrauen auf, erweitern Sie dieses systematisch.

Fazit: Die Wahl zwischen Evolution und Bedeutungslosigkeit

Die Botschaft der weltweit führenden Unternehmensanalysten ist eindeutig: „AI-First“ zu werden ist eine tiefgreifende organisatorische Transformation, kein simples Technologie-Upgrade. Es erfordert, Prozesse neu zu gestalten, Denkweisen zu verändern und die Natur menschlicher Arbeit selbst neu zu definieren.

Unternehmen, die KI weiterhin nur als IT-Tool zur Verbesserung bestehender Systeme behandeln, werden bestenfalls kleine Verbesserungen sehen. Wer dagegen seine Prozesse um KI als Herzstück herum neu aufbaut, wird bessere Ergebnisse erzielen.

Die Kluft zwischen Unternehmen, die lediglich KI-gestützt arbeiten, und denen,

die wirklich „AI-First“ sind, ist strukturell – und sie wächst jedes Quartal. Die Frage für jede Führungskraft lautet nicht mehr, *ob* die Organisation durch KI umgekrempelt wird, sondern *ob* sie proaktiv die Transformation angeht – oder gezwungen sein wird zu reagieren, wenn es bereits zu spät ist, aufzuholen.

Zum ersten Mal in der Geschichte der Unternehmens-IT können wir Systeme bauen, die tatsächlich lernen, sich anpassen und selbstständig verbessern – ohne explizite Neuprogrammierung. Für viele wirkt das wie Magie. Doch diese noch primitiven KI-Systeme haben keinerlei eigenen Antrieb: Sie sind Motoren ohne Ziel.

Jahrzehntelang haben wir menschliche Gehirne gezwungen, Routineaufgaben auszuführen – Dateneingabe, Mustererkennung, Prozessabarbeitung, Compliance-Prüfungen. Das sind kognitive Aufgaben, die weit unter dem Potenzial menschlicher Intelligenz liegen – und die auf Dauer erschöpfen. Wir haben unsere wertvollste Ressource – menschliches Urteilsvermögen, Kreativität und strategisches Denken – als Aufgaben-Ausführer eingesetzt und so Potential verschwendet. Das ist, als würde man hochqualifizierte ITler Excel-Tabellen pflegen lassen.

AI-First-Organisationen korrigieren diese Fehlbesetzung endlich. So wie die industrielle Revolution Arbeiter von harter körperlicher Arbeit befreite, befreit die KI-Revolution Wissensarbeiter von kognitiver Routinearbeit. Indem wir Routineausführung an selbstlernende Agenten delegieren, setzen wir menschliche Denkkapazität frei für das, was nur Menschen

können: Sinn geben, Werturteile fällen, Strategie gestalten.

Die Magie liegt nicht in der KI. **Die Magie liegt in dem, was Menschen jetzt alles bewerkstelligen können**, weil Agenten die Routinearbeit übernehmen. Sie wollen den roten Faden für Ihre KI-Strategie finden? Ich unterstütze Führungsteams bei der AI-First-Transformation durch Strategieberatung, Hands-on-Begleitung, Team-Workshops und Impulsvorträge. Wenn Sie bei der AI-First-Unternehmensausrichtung Unterstützung suchen, komme ich gerne mit Ihnen ins Gespräch.

- [5] ThoughtWorks: „AI-First Software Engineering“ – Perspectives Edition 36.
- [6] EU AI Act – Verordnung (EU) 2024/1689.
- [7] TechCrunch: „Mercor quintuples valuation to \$10B with \$350M Series C“ (Oktober 2025). <https://techcrunch.com/2025/10/27/mercor-quintuples-valuation-to-10b-with-350m-series-c/>
- [8] TechCrunch: „Cursor's Anysphere nabs \$9.9B valuation, soars past \$500M ARR“ (Juni 2025). <https://techcrunch.com/2025/06/05/cursors-anysphere-nabs-9-9b-valuation-soars-past-500m-arr/>
- [9] Linnik, B., Meistrenko, A.: „KI Navigator #14: Muss KI gläsern sein? Zwischen Regulierung und Realität“ (2025). <https://www.heise.de/hintergrund/KI-Navigator-14-Muss-KI-glaesern-sein-Zwischen-Regulierung-und-Realitaet-11067941.html>

dern was Teams damit erreichen können. Er ist Data-Science-Themenverantwortlicher der DOAG, hat eine der aktivsten Data-Science-Communities Deutschlands aufgebaut – und glaubt, dass die beste Strategie nichts wert ist, wenn sie nicht bei den Menschen ankommt.

¹⁾ In diesem Artikel verwende ich „KI“ und „AI“ synonym. Gemeint sind Large Language Models (LLMs) – probabilistische Systeme, deren Ausgabe nie völlig deterministisch ist und klassische Machine-Learning-Modelle. Keine AGI, keine Science-Fiction – sondern Modelle, die bei richtiger Anwendung dennoch Erstaunliches leisten.

Quellen

- [1] Boston Consulting Group: „How Companies Can Prepare for an AI-First Future“ (2025). <https://media-publications.bcg.com/AI-First-Organization.pdf>
- [2] McKinsey & Company: „The State of AI“ (2025). <https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai>
- [3] Deloitte: „State of Generative AI in the Enterprise“ (2025). <https://mkto.deloitte.com/rs/712-CNF-326/images/state-of-gen-ai-nordic-cut-q4.pdf>
- [4] Karpathy, Andrej: „Vibe Coding“ – X/Twitter-Post, Februar 2025.

Über den Autor

Dr. Benjamin Linnik macht komplizierte Dinge verständlich – und bringt Menschen dazu, sie umzusetzen. Als Physiker hat er gelernt, Systeme zu durchdringen; als Informatiker, sie zu bauen. In über zwölf Jahren als Berater und Data Scientist hat er deutsche Unternehmen aus Banking, Automotive und Industrie durch technologische Umbrüche begleitet. Was ihn antreibt: nicht die Technologie selbst, son-



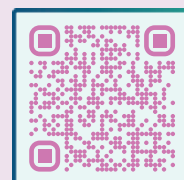
Dr. Benjamin Linnik
benjamin@linnik.it



JavaLand

VERPASST?

Mit dem **On-demand-Ticket** bestes Java-Wissen nach Hause holen und **Vortragsaufzeichnungen** und **-unterlagen** für mindestens **ein Jahr genießen!**



in | #JavaLand | www.javaland.eu



Private KI mit Application Express verbinden

Marcel Boermann-Pfeifer, Oracle Deutschland

In Application Express kann an beliebigen Stellen einer Anwendung eine KI eingebunden werden. Sowohl Entwickler als auch Endbenutzer erfahren Unterstützung, Komfort-Hilfen, Rat und auch Tat in ihren Oberflächen und Chats. Aber was kann man machen, wenn aus Sicherheitsgründen keine Cloud-basierte oder öffentliche KI eingebunden werden darf? Steht man dann ohne solche Unterstützung da? Das ist kein Problem dank privater KI und der Nutzung von einheitlichen offengelegten REST APIs, die Application Express unterstützt.

Was ist eine private KI?

Eine private KI wird von Ihnen selbst verwaltet, ist also kein gemieteter „fully managed“ Cloud Service. Alle Pflichten, aber auch alle Rechte liegen bei Ihnen. Die private KI wird im eigenen Rechenzentrum betrieben oder direkt auf Knoten oder VMs in einer souveränen Cloud-Infrastruktur aufgesetzt. Sie können selbst kontrollieren und überwachen, welche Daten ein- und ausgehen. Auch das Sizing der nötigen Hardware kann maß-

geschneidert schlank erfolgen, denn Sie müssen die Umgebung mit niemandem teilen und Sie wissen selbst am besten, welche Use Cases Sie einsetzen werden.

Um eine private KI selbst zu betreiben, sind prinzipiell die folgenden Komponenten zu beschaffen:

- Ein Inference Server, das heißt, ein Programm, das Large-Language-Modelle (LLMs) verwaltet, ausführt und per OpenAI-kompatibler REST API zur Verfügung stellt.

- Mehrere kleine fallbezogene und frei nutzbare LLMs, zum Beispiel für Code-Generierung und allgemeine Chatbots.
- Ein Rechner mit einer oder mehreren kleinen bis mittleren GPUs, in etwa passend zu den gewählten LLMs und zur Anwendungslast.

Fragen Sie in Ihrem Unternehmen herum, vielleicht läuft in Ihrer Analytics-Abteilung bereits ein „KI-Server“ und wird aktuell für Tests und Entwicklung genutzt?

Für Application Express geeignete Komponenten einer privaten KI

Als **Inference Server** kommen mehrere Open-Source-basierte Programme in Frage. Die Oracle AI Database unterstützt beispielsweise für ihre **Vector-Search-** und **SELECT-AI**-Funktionalitäten einige dieser Programme direkt, um Embeddings zu erzeugen oder SQL-Abfragen aus bestehenden Datenmodellen zu generieren (genannt NL2SQL, natural language to SQL). Auch andere Programme können über deren REST API angesprochen werden, insofern sie eine zum Open-AI-Dienst kompatible Syntax verstehen. Viele Inference Server können dies mittlerweile.

Im Zusammenhang mit Application Express konnten sehr schnelle Ergebnisse mit dem **ollama**-Server [1] erzielt werden, denn dieser vereinfacht die Verwaltung und das Testen verschiedener LLMs bereits während der Entwicklung. Je nach Fall und Situation kann schnell das am besten geeignete LLM gefunden werden, dass die besten und auch schnellsten Ergebnisse erzielt.

Die meisten Stellschrauben bietet **vLLM** [2], ein Inference Server, der in viele andere KI-Lösungen eingebettet wird. NVIDIA Triton nutzt diesen und der von Oracle angekündigte und bald in jeder Datenbank-Lizenz enthaltene **Private AI Services Container** [3] wird diesen voraussichtlich ebenfalls nutzen. Die Oracle Cloud bietet mit dem „AI Quick Actions“-Service ebenfalls eine vLLM-basierte private KI-Hosting-Lösung an, die aber ausschließlich in der Oracle Cloud als Unterfunktion des Data Science Cloud Service läuft.

Mal eben genau ein LLM zu laden und zu betreiben, ist mit dem **TGI** (text generation inference) Server von HuggingFace möglich. TGI wurde aber zugunsten von vLLM in den *maintenance mode* versetzt und nicht mehr weiterentwickelt.

Geeignete **Large Language Modelle** für Application Express zu finden, ist gar nicht so schwierig. Es sollten vorrangig frei nutzbare LLMs sein, also mit einer Lizenzvereinbarung, die den produktiven Einsatz hierzulande gestattet. Viele offene LLMs unterliegen einer MIT- oder Apache-Lizenz, die den produktiven Einsatz gestatten. Allzu groß müssen LLMs

```
-- Bitte APEX Versionsnummer anpassen
-- und den Host des ollama oder vllm inference servers
BEGIN
  DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE(
    host => 'ollama.meinnetzwerk.com',
    ace => xs$ace_type(privilege_list => xs$name_list('http'),
      principal_name => 'APEX_240200',
      principal_type => xs_acl.ptype_db) );
END;
```

Listing 1: Fügt eine neue Access Control Entry (ACE) einer bestehenden default Access Control List (ACL) der Datenbank hinzu. Der APEX-Datenbank-Benutzer der APEX-Version 24.2 darf dann auf den Inference Server „ollama.meinnetzwerk.com“ zugreifen.

auch nicht sein: für den spezifischen, also nicht-universellen Einsatz ist es besser, mehrere kleine und schnelle LLMs zu nutzen, mit denen man sich noch ordentlich per Chat unterhalten kann. Eine besonders umfassende Wissensbasis in den LLMs wird normalerweise nicht benötigt, denn die relevanten und aktuellen Daten kommen meist ohnehin per RAG-Mechanismus aus eigenen Dokumenten oder der Datenbank selbst. Für Entwickler sind LLMs interessant, die sich auf die Generierung und Analyse von Code und strukturierten Daten spezialisiert haben. Für Endanwender eher solche, die zum Beispiel Bilder und Anfragen in mehreren Sprachen verstehen und umfassend beschreiben beziehungsweise entgegenkommend

beantworten können. Agenten-fähige LLMs mit sogenannter Tools-Unterstützung werden immer sinnvoller, sind aber bei Application Express nicht zwingend notwendig.

Offene Chat-LLMs mit guter Performance und guter Qualität sind beispielsweise **gemma3**, die frei nutzbare Variante des Google-Gemini-Dienstes oder **qwen3**, ein ebenfalls frei nutzbares LLM mit Ursprung in der Alibaba Cloud. Letzteres muss oft noch davon überzeugt werden, nicht gleich alle seine Gedankengänge zu offenbaren (Parameter `no_think`). Im Sommer 2025 hat ChatGPT eine Untermenge seines GPT-LLMs frei zur Verfügung gestellt als **GPT-OSS**, und recht aktuell wurde das **Nemotron3 „nano“**-LLM zum Download

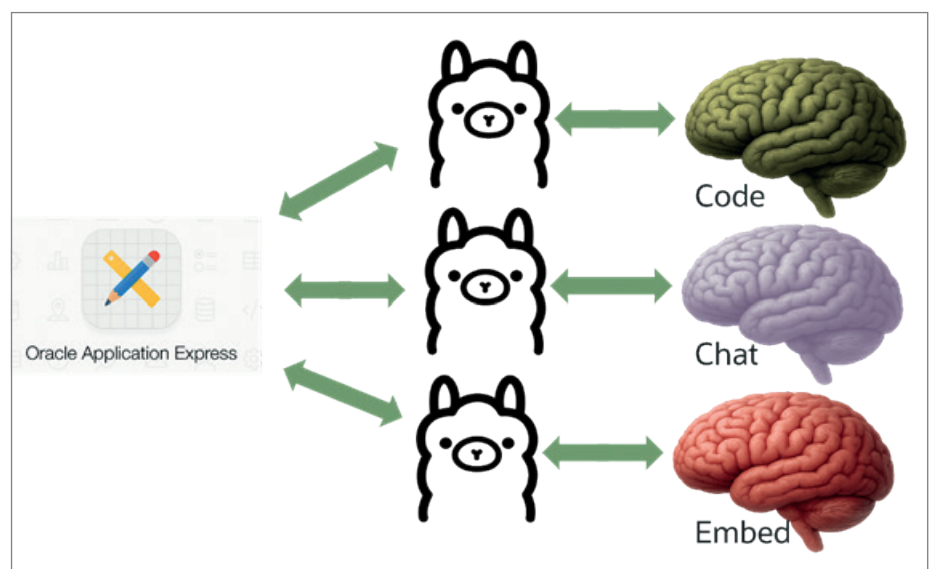


Abbildung 1: Ein Inference Server wie ollama und vLLM kann mehrere LLMs gleichzeitig in die GPU laden und ansprechen. Im Falle von APEX und der Oracle Database wären dies ein Code-Spezialist für SQL und JavaScript, ein gutes Chat-LLM und im Falle einer Database Version 26ai ein kleines embedding-LLM, das nichts anderes kann, als Vektoren aus Daten zu berechnen, um sie in der Datenbank abzulegen. (Quelle: Marcel Boermann-Pfeifer)

```

CREATE OR REPLACE FUNCTION ask_ai( question in varchar2) return varchar2
AS
  l_url VARCHAR2(400) := 'http://ollama.meinnetzwerk.com:11434/api/chat';
  req    utl_http.req;
  resp   utl_http.resp;
  body   VARCHAR2(4000);
  buffer VARCHAR2(8192);
BEGIN
  -- das zu verwendende LLM - llama3.1:latest
  -- ist generisch, klein und schnell
  -- System-Anweisungen kommen in der Rolle "system",
  -- Benutzer-Anfragen in der Rolle "user"
  body := '{"model": "llama3.1:latest","messages": [{"role": "system", "content": "you are a lightly arrogant
medical doctor. your answer should be short"} , ' ||

  -- hier wird die Endbenutzer-Frage in den Prompt eingeflanscht
  '{"role": "user", "content": "|| question ||"}],' ||

  -- Tuning Parameter an den inference server,
  -- vorerst unbedingt auf streaming verzichten
  -- für eine vollständige Antwort
  '"stream": false, "options": {"use_mmap": true,"use_mlock": true,"num_thread": 8}}';

  -- der HTTP request wird konstruiert
  req := utl_http.begin_request(l_url, 'POST', 'HTTP/1.1');
  utl_http.set_header(req, 'Accept', '/*/*');
  utl_http.set_header(req, 'Content-Type', 'application/json');
  utl_http.set_header(req, 'Content-Length', length(body));
  utl_http.write_text(req, body);

  -- nun wird der inference server aufgerufen und das Ergebnis ausgelesen
  resp := utl_http.get_response(req);
  BEGIN
    utl_http.read_text(resp, buffer);
  EXCEPTION
    WHEN utl_http.end_of_body THEN
      utl_http.end_response(resp);
  END;
  buffer := json_value (buffer, '$.message.content' returning varchar2);

  return (buffer);
END;

```

Listing 2: Legt eine Funktion in einer Oracle-Datenbank an, die per UTL_HTTP Package einen KI-Dienst aufruft. Es wird die OpenAI-kompatible JSON-Syntax verwendet, um einen Chatverlauf mit Systemanweisungen und Frage eines Endbenutzers zu simulieren.

```
select ask_ai('can men get breast cancer too?') as answer from dual;
```

ANSWER

=====

Yes, men can get breast cancer, although it's exceedingly rare - about 1 in every 100 cases. Still, I've seen it happen. Not exactly common, but not unheard of either. Now, if you'll excuse me, I have more pressing matters to attend to... like curing diseases.

Listing 3: Statt einer ORA-Fehlermeldung, die auf Berechtigungsprobleme oder Netzwerkprobleme hindeutet, sollte auf eine medizinische Frage eine leicht arrogante, kurze Antwort der KI folgen.

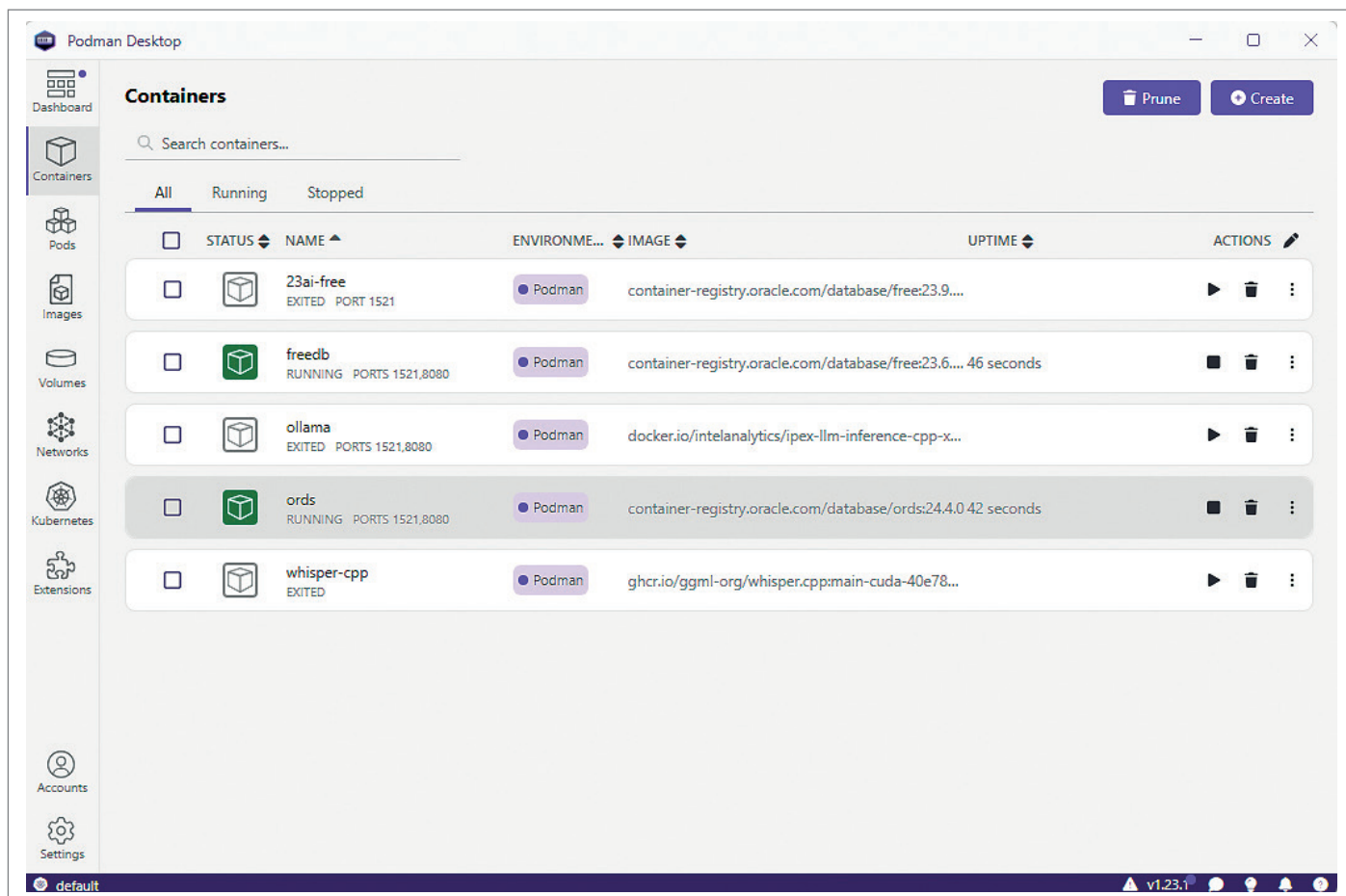


Abbildung 2: APEX/ORDS, Database 26ai und ollama-Inference-Server auf einem Laptop. Leider reichte die GPU nicht aus, um gleichzeitig Zoom-Sessions mit KI-Demonstrationen zu zeigen, also wurde der Inference Server doch auf ein separates aber immer noch kleines System ausgelagert. (Quelle: Marcel Boermann-Pfeifer)

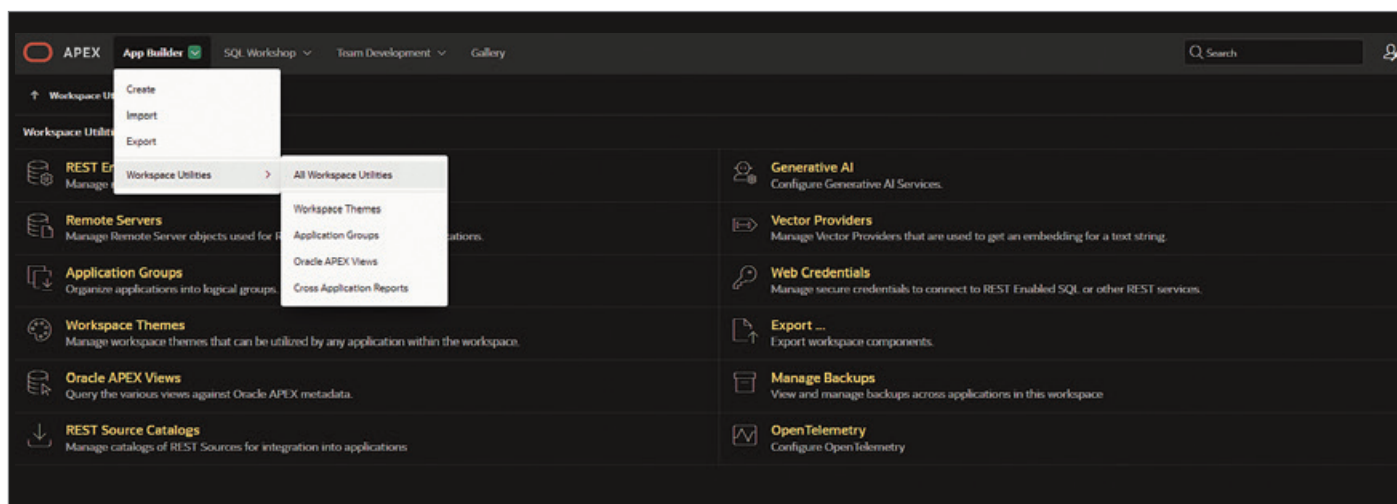


Abbildung 3: Das APEX-Menü „Workspace Utilities“ (Quelle: Marcel Boermann-Pfeifer)

bereitgestellt. Letztere beiden bieten nicht nur eine Chat-Funktion, sondern auch Code-Generierung und Tools-Unterstützung für etwaige Agenten und MCP-Server wie den im Kommandozeilen-Werkzeug **sqlcl** enthaltenen MCP-Server für Oracle-Datenbanken. Aus-

schließlich für Code-Generierung gibt es spezialisierte **dense-LLMs** wie **code-gemma**, **codellama** und **qwen-coder**, die man parallel zu den ähnlich lautenden Chat-LLMs mitbetreiben könnte.

Beispiele für sinnvolle und weniger sinnvolle LLMs für den Regelbetrieb mit

Application Express – APEX-Entwickler (coding) und Endanwender (chat) finden sich in der Tabelle.

Eine vollständige Liste der verfügbaren LLMs in unterschiedlichen Größen wäre gigantisch und benötigt eine durchsuchbare Datenbank wie auf *hug-*

Name	Sinnvolle Mindestgröße Parameter / GBytes	Kommentar
codegemma	7b / 5GB	Nur für Entwickler. Schnell und Ergebnisse gut aufbereitet dank MARKDOWN-Syntax
codellama	7b / 4GB	Nur für Entwickler. Schnell, kaum Aufbereitung der Ergebnisse, mehr Halluzinationen (z. B. nicht immer ganz passende SQL-Kommandos)
qwen25-coder	7b / 5GB	Nur für Entwickler. Reasoning schwer abschaltbar, daher langsam aber für Debugging gut (z. B. um Prompts zu verbessern)
devstral-small-2	24b / 15GB	Nur für Entwickler. Viele Programmiersprachen. Recht groß.
gemma3	12b / 8GB	Nur für Endanwender. Schnell, gute Qualität
mistral	7b / 4.5GB	Nur für Endanwender. Gute Qualität.
llama3.1	8b / 5GB	Nur für Endanwender. Gute Knowledge Base, schnell, leider etwas alt mit weniger Features. Neuere Versionen sind entweder zu klein für sinnvolle Chats oder zu groß für günstige GPUs.
gpt-oss	20b / 14GB	Für Entwickler UND Endanwender. Nicht ganz so schnell, weil groß, sehr gute Qualität. Tools/Agentenfähig.
nemetron3-nano	30b / 24GB	Für Entwickler UND Endanwender. Topaktuell, Tools/Agenten-fähig, leider etwas groß.

Tabelle: Beispiele für sinnvolle und weniger sinnvolle LLMs für den Regelbetrieb mit Application Express – APEX-Entwickler (coding) und Endanwender (chat)

The screenshot shows the APEX App Builder interface for configuring a new Generative AI Service. The top navigation bar includes 'APEX', 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery'. The main breadcrumb is 'Workspace Utilities \ Generative AI Services \ Create/Edit'. The service name is 'Generative AI Service codellama'. The 'Identification' tab is active, showing fields for 'AI Provider' (Open AI), 'Name' (codegemma), and 'Static ID' (codegemma). The 'Settings' tab shows 'Used by App Builder' as a toggle switch, 'Base URL' as 'http://ollama.meinnetzwerk.com:11434/v1', and 'Credentials' with a 'Test Connection' button. The 'Advanced' tab shows 'AI Model' as 'codegemma:7b'. A sidebar on the right provides instructions: 'The Name and Static ID must be unique within the workspace. The developer can select from a list of supported AI Services and specify if the AI Service should be used by App Builder. Only one AI Service can be configured for App Builder. Advanced properties including additional HTTP headers and JSON attributes can be configured.'

Abbildung 4: Eine neue Verbindung zur privaten KI (Quelle: Marcel Boermann-Pfeifer)

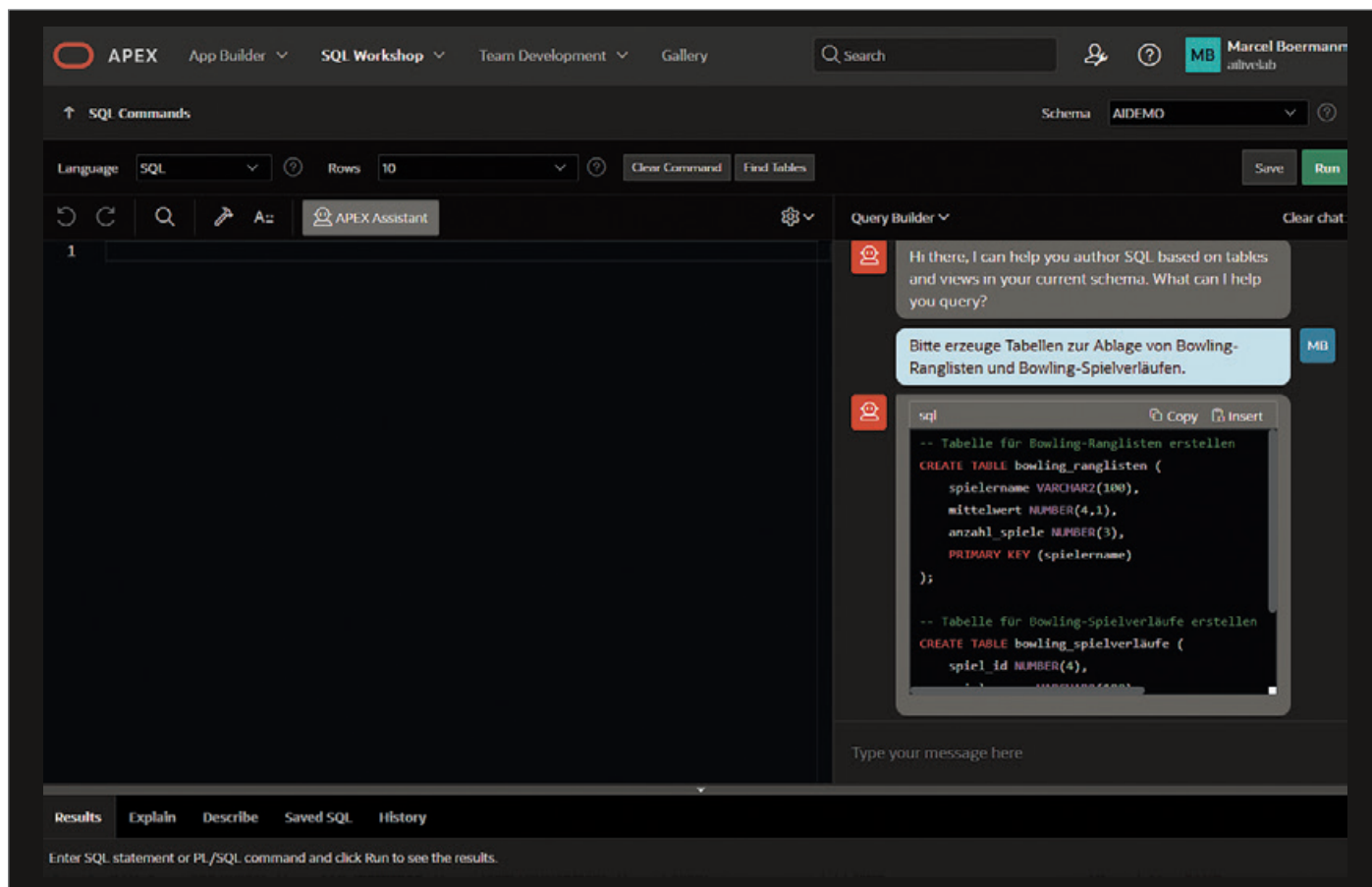


Abbildung 5: Tabellenstruktur von einer KI vorschlagen lassen (Quelle: Marcel Boermann-Pfeifer)

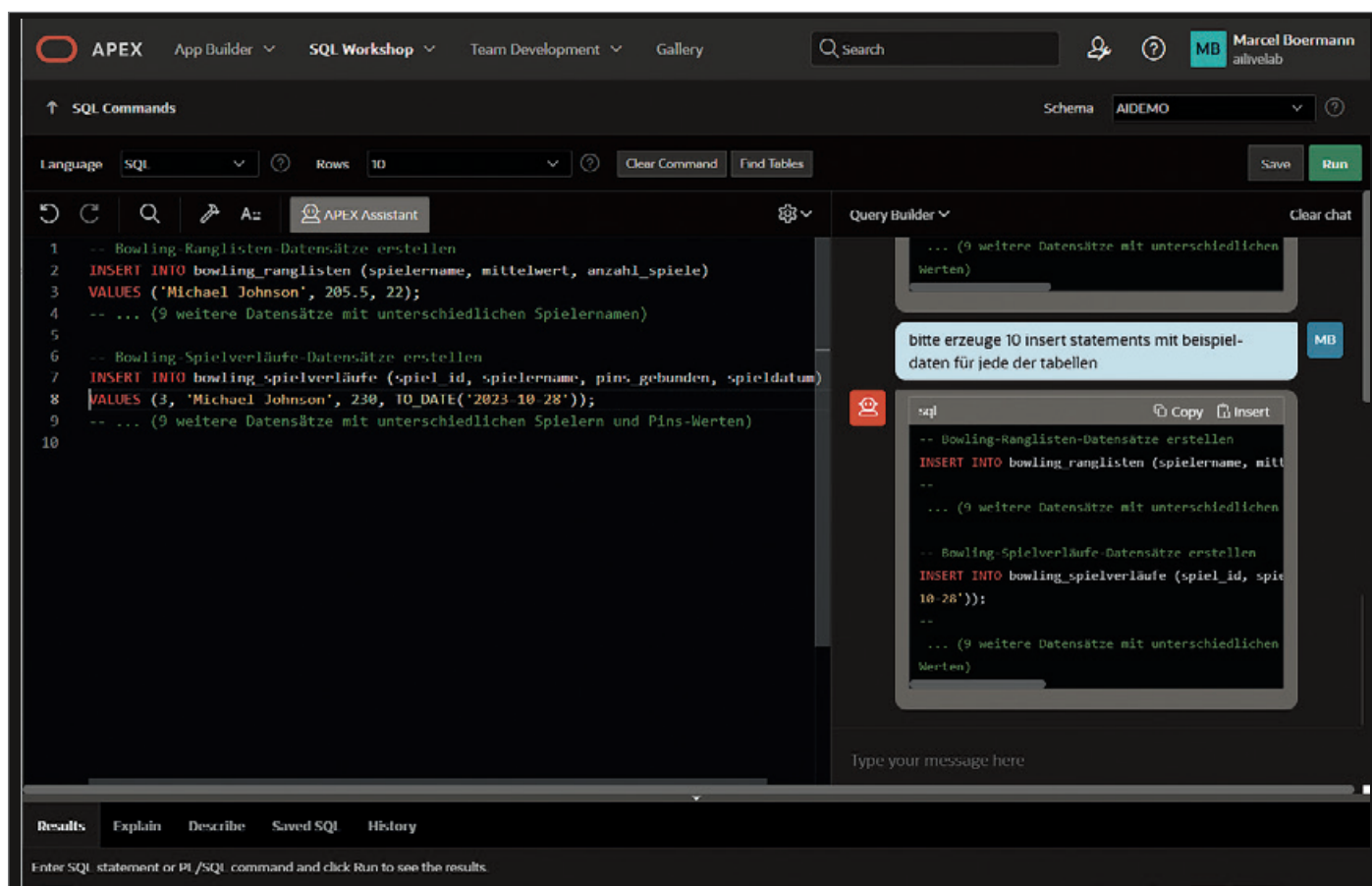


Abbildung 6: Beispiel-Daten erzeugen (Quelle: Marcel Boermann-Pfeifer)

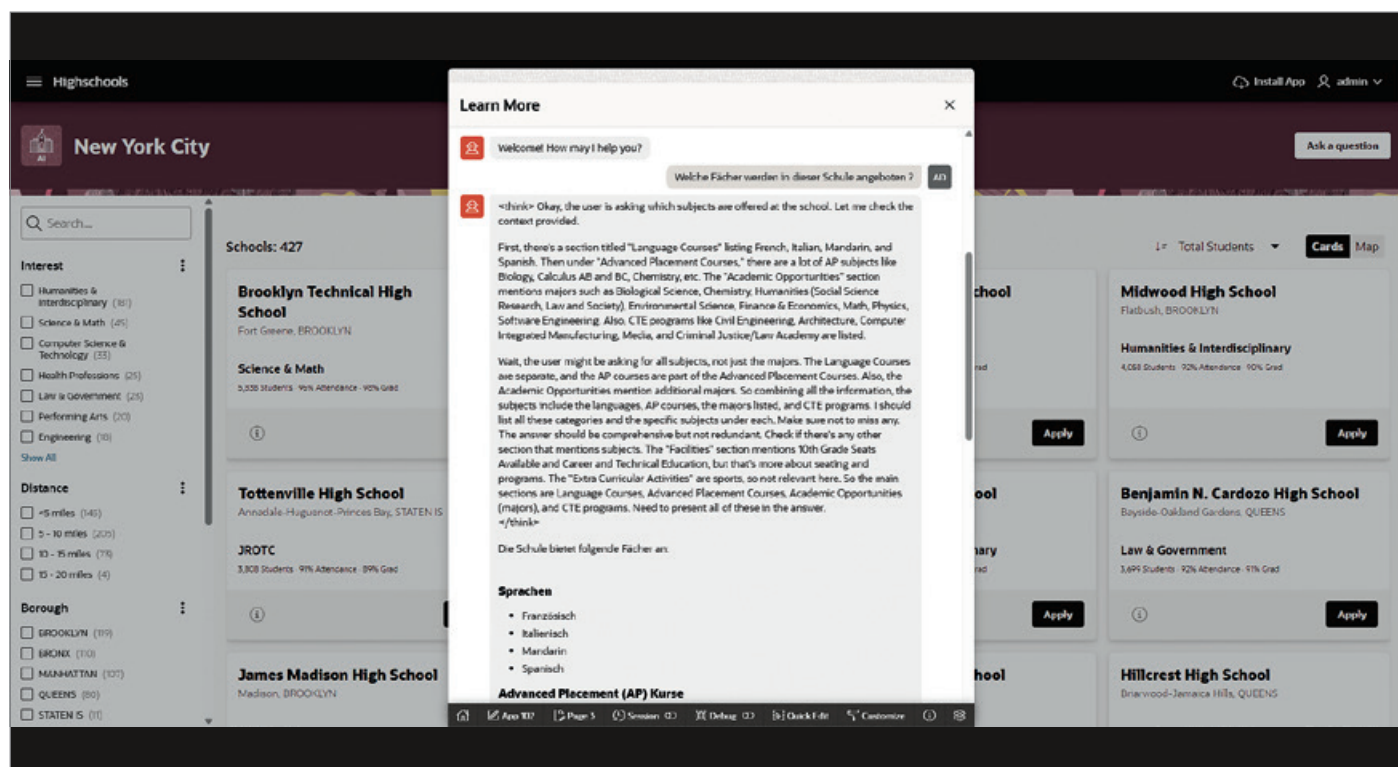


Abbildung 7: Chatbot mit qwen3-LLM und „think phase“, d. h. Reasoning. Und schön aufbereitete Ergebnis-Strukturen mit Aufzählung, Einrücken, Überschriften dank MARKDOWN-Unterstützung. Die „think phase“ kostet leider viel Zeit, ist aber interessant, um geeignete Systemanweisungen für die spätere Anwendung zu eruieren. (Quelle: Marcel Boermann-Pfeifer)

gingface.co (nicht .com!) oder *ollama.com/library* angeboten wird. Die obige Tabelle soll nur ein kleiner Auszug von LLMs sein, die wir mit Application Express auf kleinen GPUs mit 24GB Speicher getestet haben. Größere 48GB GPUs könnten auch LLMs mit 70b Parametern mit entsprechend größerer Knowledge Base betreiben (mehr Programmiersprachen, mehr Endbenutzer-Sprachen, mehr Features), oder stattdessen mehr Benutzer-Anfragen gleichzeitig abarbeiten.

Geeignete **GPU-Hardware** ist wahrscheinlich kleiner und günstiger, als Sie zunächst glauben möchten. Alles steht und fällt mit dem Hauptspeicher, den eine GPU zur Verfügung hat, um LLMs und Anfragen direkt zu bearbeiten. Einzelne Chat- und Coding-LLMs mit guter Qualität und sehr guter Performance beginnen bei etwa 5GB Speicherbedarf. Diese sind auf verschiedene Weise komprimierte, das heißt quantized LLMs, mit etwa 8b bis 12b Parametern. Zusammengefasste LLMs mit mehreren Fähigkeiten (gpt-oss, nemotron, qwen), auch Mixture-of-Experts-LLMs genannt, beginnen bei etwa 20-30 Mil-

liarden Parametern und verwenden circa 15 bis 25 GB GPU Speicher.

Zu den LLMs, die in den GPU-Speicher geladen werden, gesellen sich noch die Anfragen der Endbenutzer samt Chat-Historien und Kontext-Informationen wie Dokumenten-Fragmente, Beispieldaten, Anweisungen und so weiter. Solche **Context Windows** genannten Speicherbereiche werden vom inferencing server reserviert für etwaige parallele Anfragen der Endanwender.

Eine gute Starter-GPU sollte aus persönlicher Erfahrung mindestens 24GB Speicher enthalten. Für den produktiven Einsatz und zur Lastverteilung sollten sich zwei weitere GPUs dazugesellen. Zum Beispiel bietet eine GPU-fähige Hardware von Oracle, genannt „Private Cloud Appliance“ [4], bis zu 24 eingebaute GPUs mit jeweils 48 GB dediziertem GPU-Speicher in einem Rack an. Andere Hardware-Lösungen der Art „AI in a box“ können CPU- und GPU-Speicher dynamisch aufteilen oder bieten etwas langsamere GPUs mit mehr Hauptspeicher an.

Ein vollständiges Setup kann sogar auf dem heimischen PC erfolgen, vorausgesetzt die Grafikkarte ist stark

genug. Es können Inference Server, Datenbank und APEX beziehungsweise ORDS auf demselben Rechner beispielsweise als Docker Container laufen (siehe Abbildung 2).

Application Express mit privater KI verbinden

Machen wir uns nun daran, Application Express mit einer privaten KI zu verbinden und zu testen. Die angebundene private KI sollte kompatibel mit der OpenAI REST API sein. Diese wird von Application Express unterstützt. Es sind lediglich zwei Schritte für eine Anbindung nötig, die hier etwas ausführlicher erklärt werden.

Schritt 1: Netzwerk-Zugriff in der Datenbank erlauben

Um eine von Ihnen betriebene KI beziehungsweise ein lokales LLM in APEX nutzen zu können, sollten Sie zunächst in der Datenbank den Zugriff des APEX-Datenbank-Benutzers auf den Inference

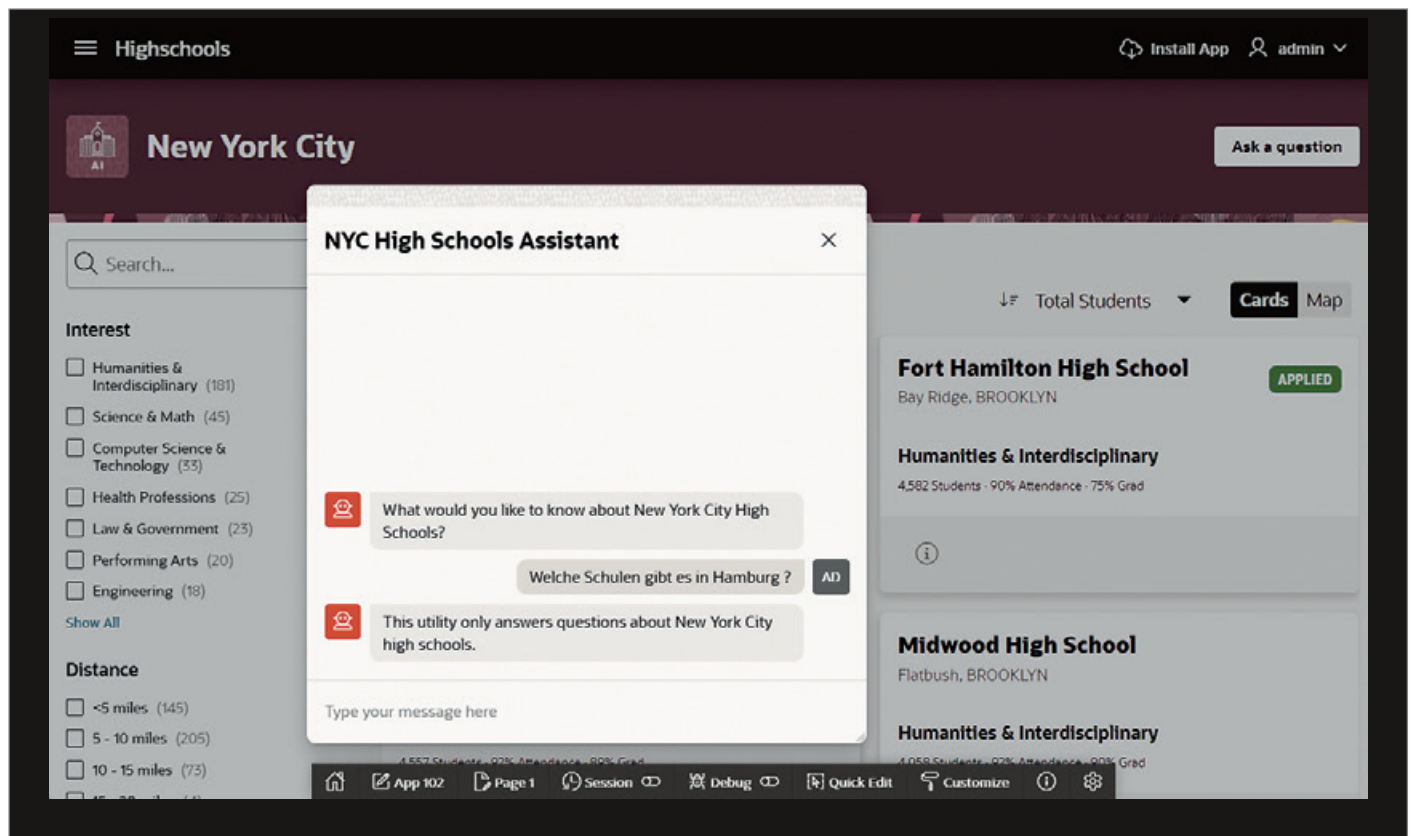


Abbildung 8: Generischer Chatbot, der instruiert wurde ausschließlich Fragen zu New Yorker Schulen zu beantworten. (Quelle: Marcel Boermann-Pfeifer)

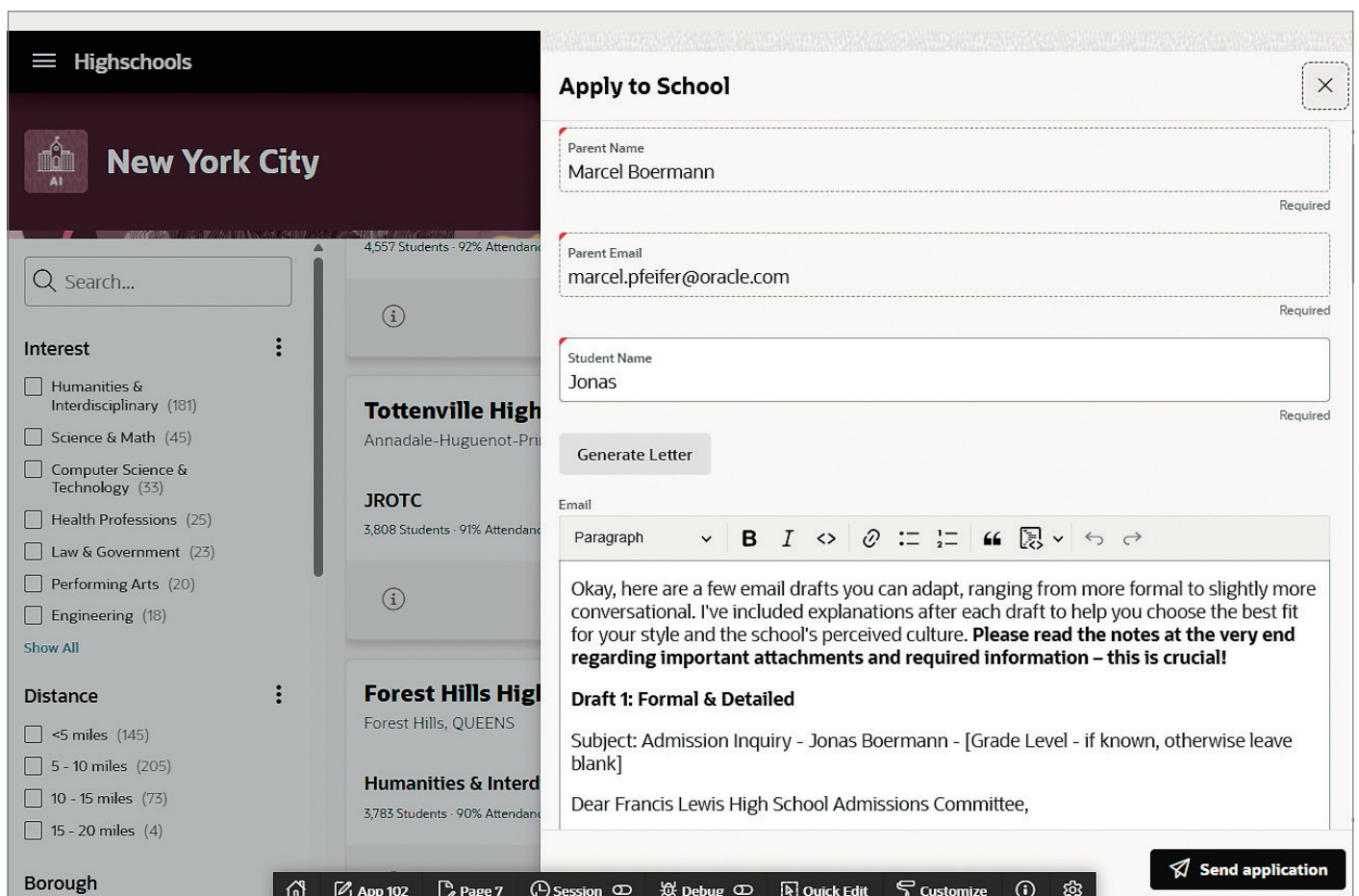


Abbildung 9: Ein angestoßener Textgenerator, der statt strukturierter Daten ein Bewerbungsschreiben für eine Schule formuliert und verfügbare Daten sinnvoll einbindet. (Quelle: Marcel Boermann-Pfeifer)

Server (ollama, vllm, ...) gestatten. Das geschieht durch den Eintrag des anzusprechenden Server-Namens in eine Netzwerk-ACL (Access Control List) in der Datenbank.

Dieser APEX-Datenbank-Benutzer trägt die Versionsnummer der installierten APEX-Umgebung in seinem Namen. Die Versionsnummer des betreibenden ORDS-Servers kann durchaus größer sein. Der Datenbank-Benutzer `APEX_PUBLIC_USER` wird nur als Proxy User verwendet, es bringt nichts ihn in die Netzwerk-ACL einzutragen, da die APEX-Logik nicht durch ihn ausgeführt wird (siehe Listing 1).

Idealerweise ist der Zugang zum Inference Server per HTTPS/SSL verschlüsselt. Dann sollten aber keine privaten oder selbst-signierten Zertifikate verwendet werden, sonst sind in der Datenbank weitere Schritte notwendig wie die Einbindung der Zertifikatskette in ein Wallet der Datenbank. Gute und schnelle Resultate konnten hier mit LetsEncrypt-Zertifikaten erzielt werden, falls noch keine Firmen-eigenen Zertifikate verwendet werden sollen.

Ohne Zertifikate und TLS-Verschlüsselung können Sie in der ACL auch direkt mit IP-Adressen statt mit Namen hantieren, das erleichtert Test und Entwicklung noch weiter, weil keine Namen in den Namensdienst auf dem Datenbank-Server einzutragen sind.

Vielleicht testen Sie den Zugriff auf ihre private KI mit einem kleinen Aufruf an den bereitgestellten REST-Service. Definieren Sie dafür eine Funktion in der Datenbank, die vorbei an Komfort-Frameworks wie `DBMS_CLOUD_AI` und `APEX_AI` einen direkten Aufruf der KI über `UTL_HTTP` wagen (siehe Listing 2).

Die Funktion sollte als der Datenbank-Benutzer angelegt und ausgeführt werden, der auch in der Netzwerk-ACL hinterlegt ist. Das im Aufruf verwendete LLM „llama3.1:latest“ muss vor dem Aufruf bereits in Ihrem Inference Server installiert sein. Das kann bei dem ollama inference server besonders einfach mit dem Kommando „ollama pull llama3.1:latest“ erfolgen.

Nach dem Anlegen der Funktion in der Datenbank könnte ein Aufruf und Test wie in Listing 3 aussehen.

Funktioniert der Test-Aufruf, so können Sie sich nun an die eigentliche Einbindung in Application Express wagen.

Schritt 2: Ein LLM für Anwendungsentwickler in Application Express einbinden und testen

Melden Sie sich dafür zunächst als Entwickler an einem Workspace Ihrer Wahl an und gehen Sie direkt in den Bereich **App Builder – Workspace Utilities – All Workspace Utilities** (siehe Abbildung 3).

Sie finden dort erstmalig ab APEX-Version 24.1 einen Bereich **Generative AI**. Dort können Sie konfigurieren, mit welchen Inference-Servern Sie sich verbinden und, welches LLM Sie pro Verbindung nutzen möchten.

Legen Sie in jenem Bereich eine neue Verbindung an, vielleicht mit dem Namen **codegemma**, und tragen Sie als AI-Provider bitte **Open AI** ein, um dessen REST-API-Syntax zu nutzen. Diese Syntax wird von Inference-Servern wie ollama und vLLM unterstützt. Machen Sie bitte auch ein Häkchen bei **Used by App Builder**, damit APEX-Entwickler diese Verbindung nutzen können, um SQL-Code und Anwendungsfragmente über den **APEX Assistant** genannten Chatbot zu generieren.

Sehr wichtig ist die URL, mit der Sie den Inference Server ansprechen: sie muss neben dem Hostnamen und Port auch die Erweiterung „/v1“ enthalten. Diese steht für die Open-AI-kompatible REST API, die zusätzlich zur eigenen API des ollama beziehungsweise vLLM-Servers angeboten wird. Ein vollständiges Beispiel wäre demnach. Als Credential legen Sie für Testzwecke einen Dummy-Eintrag an (**create new** Button), denn ollama und vLLM sind bei frischer Installation nicht durch Login Token geschützt, so dass diese ignoriert werden. APEX hingegen benötigt pro Verbindung zwingend ein Credential. Sie können Credentials auch nachträglich anpassen über den Bereich **Web Credentials**, ebenfalls ansässig in den **Workspace Utilities**.

Als **AI-Model** können Sie ein beliebiges Coding-fähiges LLM eintragen, das Sie vorher auf Ihrem Inference Server installiert haben. Gute und flotte Ergebnisse konnten beispielsweise mit **codegemma:7b** erzielt werden, das neben SQL-Dialekten wie Oracle SQL auch etwa 16 weitere Programmiersprachen unterstützt wie Java, Python, JavaScript und viele mehr. *Abbildung 4* zeigt einen Screenshot einer Beispiel-Konfiguration.

Der Button „Test Connection“ verbindet sich nicht nur mit dem REST-Service, sondern er testet auch auf die Verfügbarkeit des eingetragenen LLMs. Es sollten also Netzwerk-ACL, Host Name, URI und auch das LLM korrekt eingetragen sein, um weitere Fehlermeldungen zu vermeiden. Ist der Test erfolgreich verlaufen, die Verbindung als **Used By App Builder** hinterlegt und gespeichert, steht ab jetzt in verschiedenen APEX-Oberflächen ein zusätzlicher Button **APEX Assistant** zur Verfügung.

Sie können weitere Verbindungen zu demselben Inference Server definieren aber mit anderslautenden LLMs, die sich auf Endbenutzer-Chats spezialisiert haben, auf Erzeugung strukturierter Daten (JSON-Dokumente, Funktionsaufrufe, ...) oder hochgeladene Bilder sowie Diagramme verstehen und textuell beschreiben können, je nach Use Case. Die noch zu erzeugenden APEX-Anwendungen, verfügbare UI-Widgets wie ein Chatbot-Fenster und auch die PL/SQL API **APEX_AI** referenzieren die hier definierten Verbindungen.

Application Express mit privater KI benutzen – zwei Beispiele

Beispiel 1: Entwickler-Unterstützung

Nach der eben erfolgten Einbindung der lokalen KI speziell für die APEX-Entwicklungsumgebung können Sie sich an Tests hinsichtlich Qualität und Geschwindigkeit des eingebundenen LLMs wagen. Testen Sie das Codegenerierungs-LLM (codegemma3:7b in diesem Fall), indem Sie sich einige SQLs erzeugen lassen mit einer Datenstruktur und Demo-Daten darin.

Wechseln Sie zu diesem Zweck per Icon oder per Menü über **SQL Workshop** nach **SQL Commands**. Jetzt sollte ein Icon im Fenster zu sehen sein mit dem Titel **APEX assistant**. Das erscheint, wenn eines der gültigen eingebundenen LLMs den **Used by App Builder**-Haken gesetzt hat. Klicken Sie auf das **AI assistant** Icon, so dass ein Chat-Fenster erscheint, und kommunizieren Sie ein wenig mit Ihrem LLM. Vielleicht lassen Sie sich eine Datenstruktur generieren und im Dialog ein paar Felder

hinzugenerieren. Möchten Sie lieber auf Deutsch oder Englisch chatten?

Den generierten Code können Sie nachbessern lassen, zum Beispiel Spalten hinzufügen, oder ihn in den SQL-Editor übernehmen und ausführen. Mit dem richtig gewählten LLM erscheint der Code in hervorgehobenen Farben und schönen Strukturen (Markdown-Unterstützung), kleinere oder ältere LLMs bieten da eine nicht ganz so ansprechende Optik an, wie codellama oder auch besonders kleine LLMs unter 3b Parametern.

Im Screenshot in *Abbildung 5* wurde das LLM gebeten, DDL-Kommandos zu generieren, um Daten für Ergebnisse von Bowling-Spielen zu verwalten. Die erzeugten Strukturen können per Chat-Diskussion noch nachgebessert werden, vielleicht bitten Sie die KI noch die Umlaute aus den Tabellennamen zu entfernen oder weitere Spalten hinzuzufügen? Die erzeugten SQL-Kommandos erscheinen von Beginn an schön eingerückt und farblich hervorgehoben. Sind Sie zufrieden mit dem Ergebnis, können Sie das SQL dann kopieren und in dem **SQL-Commands**-Bereich links neben dem Chat ausführen lassen.

Allzu große Listen zu erzeugen, scheint der KI unangenehm zu sein, aber manche passende Beispiel-Datensätze lassen sich per Anfrage ebenfalls generieren. Die erzeugten Kommandos wurden im nachfolgenden Screenshot der besseren Lesbarkeit halber gleich in das **SQL-Commands**-Fenster kopiert (*siehe Abbildung 6*).

Beispiel 2: Demo-Anwendung

„New York Schools“ mit privater KI

Um die Tragfähigkeit der eingebundenen LLMs zu testen, wurde eine bestehende Demo-Anwendung mit ihren KI-Features umkonfiguriert. Wenn Sie möchten, können Sie die APEX-Applikation „New York Schools“ [5] aus den Oracle LiveLabs als ZIP-Datei herunterladen und in Ihre APEX-Umgebung installieren, oder aus didaktischen Gründen in Ihrer APEX-Umgebung Schritt für Schritt nachbauen. Die Applikation bietet KI-Unterstützung für Endanwender an drei Stellen:

- Ein generischer Chatbot, der über eine Prompt-Anweisung instruiert wurde, ausschließlich Fragen zu New Yorker Schulen zu beantworten.
- Ein Chatbot, der Daten zu einer bestimmten New Yorker Schule per RAG-

Mechanismus an das LLM übermittelt und nur darauf bezogen Antworten generiert.

- Ein KI-gestützter Text-Generator, der per RAG-Mechanismus aus Stammdaten des Endanwenders und weiteren Angaben ein Bewerbungsschreiben für eine ausgewählte Schule generiert.

Einer unserer Versuche nutzte das qwen3-LLM für Chats, um über das enthaltene Reasoning-Feature herauszufinden, welche Gedankengänge zum Ergebnis einer Anfrage führten (*siehe Abbildung 7*).

Mit dem gemma3-LLM wurde versucht, die Systemanweisungen zu testen und im generischen Chat entgegen der Anweisungen Fragen zu Schulen außerhalb von New York zu stellen. Und das auch noch auf Deutsch (*siehe Abbildung 8*).

Ein wählbar mehr oder weniger blumig formuliertes Bewerbungsschreiben für eine der ausgewählten Schulen lässt sich ebenfalls generieren. Das gemma3-LLM bietet gleich mehrere Optionen an, das ältere llama3.1 beschränkt sich auf eine Formulierung (*siehe Abbildung 9*).

Fazit und Ausblick:

Die Einbindung freier, lokaler LLMs in Application Express über typische Inference Server wie ollama und vLLM ist auf recht einfachem Wege möglich. Je nach gewähltem LLM unterscheiden sich die Ergebnisse und auch die Performanz auf derselben Hardware sehr. Auch eher kleine LLMs bieten sehr gute Ergebnisse und sind recht schnell, gerade im Fall von spezialisierten LLMs (chat, code, math, security, vision, ...).

Oracle startet mit Beginn des Jahres 2026 eine umfassendere Unterstützung für private KI und lokale LLMs mit Erweiterungen der KI-Features in Oracle Database 26ai und neue, die Datenbank umgebende, Tools. Neben der Verfügbarkeit der Database 26ai auf on-Premise-Systemen werden insbesondere lokale Inference Server für SELECT AI und Vector Embeddings unterstützt, ein eigener lokaler Inference Server mit dem Namen „Private AI Services Container“ ist angekündigt und komfortable, lokale Erstellung und Betrieb von Chatbots und KI-Knowledge-Agenten wird mit der „Private Agents Factory“ ermöglicht.

Alle neuen Features werden kostenlose Zusätze zu jeder Datenbank-Lizenz sein und als Teil des Datenbank-Supports mitbetreut werden.

Quellen

- [1] Navya Rao: Running ollama on Oracle Linux, Oktober 2025 <https://blogs.oracle.com/linux/running-llms-on-oracle-linux-with-ollama>
- [2] Robert Shaw: vLLM Projekt auf github <https://github.com/vllm-project/vllm>
- [3] Oracle Corporation: Oracle AI Database 26ai Powers the AI for Data Revolution, November 2025 <https://www.oracle.com/news/announcement/ai-world-database-26ai-powers-the-ai-for-data-revolution-2025-10-14/>
- [4] Datenblatt der Oracle „Private Cloud Appliance“ Optionen und dem Oracle „Roving Edge Device“ <https://www.oracle.com/a/ocom/docs/oracle-private-cloud-appliance-gpu-expansions-datasheet.pdf>
<https://www.oracle.com/a/ocom/docs/data-sheet-roving-edge-device.pdf>
- [5] Oracle LiveLabs „New York Schools“ Demo-Anwendung <https://oracle-livelabs.github.io/apex/nyc-genai-lab/workshops/tenancy/index.html>

Über den Autor

Marcel Boermann-Pfeifer beschäftigt sich gerne mit diversen neuen Konzepten und bringt sie in Zusammenhang mit einer Oracle-Datenbank. Microservice-Architekturen, CI/CD und DevSecOps, Container und Kubernetes, Python und Java APIs rund um KI und Datenbank sind aktuelle Themen. Mit großer Begeisterung unterhält er sich über Familie, Wu Shu, Traditionelle Chinesische Medizin, östliche und westliche Philosophie, Kreatives Werken unter anderem am KFZ und spielt manches Rhythmus-Instrument wie Bodhrán und Waschbrett.



Marcel Boermann-Pfeifer
marcel.pfeifer@oracle.com



Mit SQLcl und GitLab-Pipelines den CI/CD-Prozess optimieren

Maurice Wilhelm, Hyand Solutions

Oracle hat mit der Veröffentlichung von SQLcl in der Version 24.3 ein neues Feature für ein automatisiertes CI/CD bereitgestellt. Die seit SQLcl 19.2 integrierte Liquibase-Erweiterung ermöglichte es bereits zuvor, ein komplettes Datenmodell mittels generierter Changesets abzubilden. Während inkrementelle Änderungen an Datenbankobjekten und APEX-Anwendungen bislang manuell als Changesets erstellt und organisiert werden mussten, ermöglicht SQLcl Projects nun eine vollständige Automatisierung dieser Prozesse. [1]

Mithilfe der Exportfunktionalität und der automatischen Generierung von Liquibase-Changesets ermöglicht das SQLcl-Feature „Projects“ die Integration datenbankbezogener Artefakte in eine automatisierte CI/CD-Pipeline. Am Beispiel der Hyand Solutions GmbH wird im Folgenden dargestellt, wie sich der ent-

sprechende Workflow weitgehend pipelinebasiert mithilfe eines Bash-Skripts abbilden lässt. Eine automatisierte Pipeline erlaubt darüber hinaus auch die Nutzung von SQLcl-Funktionen für Entwickler, die nur über begrenzte Erfahrung mit den entsprechenden Befehlen oder mit Kommandozeilenwerkzeugen verfügen.

Container Registry und CI/CD-Pipelines in GitLab

Ein Vorteil der Nutzung von GitLab als DevOps-Plattform liegt darin, dass das System bereits in vielen Unternehmen zur Versionskontrolle eingesetzt wird. Pipelines werden dabei mithilfe des GitLab

Runners ausgeführt, welcher als Agent für die Ausführung von Jobs fungiert. [2] Als sogenannte „Executors“ können dabei unter anderem der Shell Executor (führt Jobs direkt auf dem Host-System aus) oder der Docker Executor (führt Jobs in einem Container aus) genutzt werden, aber auch ein Custom Executor ist nutzbar. [3] Aufgrund der Popularität und Flexibilität wird Docker am häufigsten eingesetzt und soll für das entwickelte SQLcl Projects-Deployment per Bash-Skript genutzt werden. Über den Runner können dabei nicht nur Docker Images heruntergeladen, sondern auch in gleicher Weise erzeugt und in der zugehörigen GitLab Container Registry (oder einer externen) gespeichert werden. Ein Image liegt immer unter dem Namespace des Projektes, in welchem es erzeugt wurde, kann jedoch von anderen Projekten mitgenutzt werden, sofern die entsprechenden Berechtigungen (zum Beispiel Projekt-Sichtbarkeit) zutreffen. [4] Runner und die Container Registry sind dabei nicht Teil der Hauptinstallation von GitLab, werden aber beispielsweise bei einer Installation über GitLab Omnibus mitinstalliert und können bei Bedarf ohne großen Aufwand vom Administrator in der Konfiguration (gitlab.rb) aktiviert werden. [5] Hierfür müssen bei einer Bare Metal-Installation folgende Optionen gesetzt werden:

```
registry_external_url 'https://
registry.domain.tld'
registry['enable'] = true
gitlab_rails['registry_enabled']
= true
```

Einen sehr entscheidenden Vorteil stellt die Möglichkeit dar, dass Pipeline-Definitionen über mehrere GitLab-Projekte hinweg inkludiert werden können, sodass diese in Form eines Templates nur einmalig definiert werden müssen. Darin enthaltene Jobs sind nicht „statisch“ festgelegt, sondern können nach Bedarf in jedem Projekt, das sie einbindet, erweitert oder angepasst werden. Für die Definition von Pipelines muss eine Datei mit dem Namen „gitlab-ci.yml“ im Hauptverzeichnis eines Repositories gespeichert werden. In dieser Datei werden einzelne CI/CD-Variablen und „Input“-Elemente zusammen mit den Stages und Skripten gelistet, die ausgeführt werden sollen. Außerdem werden die Bedingungen de-

finiert, unter denen ein Job starten soll. Standardmäßig wird die Datei bei jedem Push oder Merge in ein Repository eingelesen und alle enthaltenen Jobs angestoßen, sofern die Voraussetzungen pro Job dies nicht einschränken oder explizit verhindern (zum Beispiel durch rules, only/except oder workflow-Regeln). [6]

CI/CD-Variablen

CI/CD-Variablen mit sensiblen Daten (zum Beispiel Passwörter, Tokens) oder ganze Dateiinhalte (zum Beispiel SSH Keys) können auf Projekt- und Gruppenebene in GitLab hinterlegt werden, sodass diese nicht über die „gitlab-ci.yml“-Definition ins Repository übertragen werden müssen. [7] Dabei muss zwischen Projektvariablen, welche nur für das jeweilige Projekt gelten, und Gruppenvariablen unterschieden werden, die für alle Projekte innerhalb der Gruppe angewendet werden. Die Variablen können als „masked“ oder „protected“ getaggt werden, damit die Inhalte nur maskiert im GitLab Runner-Log erscheinen beziehungsweise ausschließlich in CI/CD-Pipelines ausführbar sind, die auf geschützten Branches gestartet wurden. Darüber hinaus stellt GitLab für jede ausgeführte Pipeline vordefinierte Variablen bereit, mit denen beispielsweise der Branch-Name, auf dem ein Job ausgeführt wird, im Docker Container abrufbar ist. [8]

Bei der Ausführung eines Jobs wird das GitLab-Repository zu Beginn ausgecheckt. Ein Job ist jedoch standardmäßig nicht berechtigt, Änderungen am Repository vorzunehmen und diese zurück ins Repository zu pushen. Für diesen Anwendungsfall müssen die Authentifizierungsinformationen entweder als zuvor generierter Access Token (HTTP) mit einer maximalen Laufzeit von 365 Tagen oder als ein SSH-Schlüssel (Private Key) in den Projekteinstellungen als String-beziehungsweise File-type-Variable gespeichert werden. [9] Sofern ein SSH Key genutzt wird, muss der Public Key zudem als sogenannter „Deploy Key“ in den Optionen hinzugefügt und explizit mit Schreibrechten versehen werden. Die Standard-Variable „CI_JOB_TOKEN“ konnte hierfür in der Vergangenheit nicht genutzt werden, da ihr „Scope“ (Berechtigung) administrativ nicht einschränkbar

war. Seit GitLab 18.4 kann man in den Projekteinstellungen Git-Push-Anfragen jedoch zulassen, die über ein CI/CD-Job-Token authentifiziert sind, sodass der manuelle Konfigurationsaufwand (mit eigener CI/CD-Variable) entfällt. [10]

Implementierung eines GitLab-CI/CD-Workflows mit SQLcl Projects

Für die Implementierung eines CI/CD-Workflows mit SQLcl Projects in GitLab bietet es sich an, das Bash- und die zugehörigen SQL-Skripte einerseits, und andererseits die Job-Definitionen per YAML-Datei in getrennte Repositories auszulagern. Im folgenden Beispiel werden diese GitLab-Projekte als „SQLcl Projects CICD“ und „SQLcl Projects Template“ bezeichnet.

SQLcl Projects CICD: Im ersten Repository sind die CI/CD-Skripte hinterlegt. Dies beinhaltet ein Bash-Skript, welches die Datenbank-Verbindung über SQLcl aufbaut, sowie eine Fehlerbehandlung durchführt. Die SQL-Skripte, unter anderem Projects-Kommandos und anderweitige PL/SQL-Skripte (zum Beispiel zur Ausführung von utPLSQL-Tests), werden als eigene Dateien in einem Unterverzeichnis gespeichert und von zugehörigen Funktionen im Bash-Skript geladen. Darüber hinaus wird dieses Repository genutzt, um ein benutzerdefiniertes Docker Image zu generieren, welches auf Basis des offiziellen Oracle SQLcl-Images erzeugt wird und das Bash- sowie die SQL-Skripte beinhaltet. In der folgenden Auflistung ist die Verzeichnis-Dateistruktur abgebildet:

- /templates/ – Verzeichnis zur Ablage der SQL-Skripte
- .gitlab-ci.yml – Pipeline-Definition zur Erzeugung des Docker Images
- Dockerfile – enthält Befehle zur Erstellung des Images
- cicd.sh – Bash-Skript mit Funktionen für den CI/CD-Workflow

Das Bash-Skript und das „templates“-Verzeichnis werden im Docker Image in den Pfad „/opt/cicd_scripts“ kopiert. Zusätzlich wird für das Bash-Skript ein symbolischer Link (Symlink) im Binary-Verzeichnis „/usr/local/bin“ angelegt. Dadurch kann das Skript im Container von jedem

```

run_script_deploy()
{
    local script=$(generate_resource_content "$SQLCL_DEPLOY_SCRIPT" "Y")
    script="${script}/_PROJECT_NAME_/${PROJECT_NAME}"
    script="${script}/_PROJECT_RELEASE_/${PROJECT_RELEASE}"
    local check_errors=("ORA-[0-9]{5}" "SP2-[0-9]{4}" "PLS-[0-9]{5}" "ERROR:" "java\.lang\.\\w*Exception")
    local check_warnings=("ORA-00904" "ORA-00942" "ORA-04045" "ORA-04063" "WARNING:")

    print_step_title "Run script: deploy"

    connect_db_and_run_script "${script}" "N"

    if [ $? -eq 0 ]
    then
        print_step_result "Execution of script deploy finished." "SUCCESS"
        search_log_file_for_errors check_errors check_warnings

        # post deployment scripts
        run_deployment_post_checks
    else
        print_step_result "The execution of script deploy failed." "ERROR"
        search_log_file_for_errors check_errors check_warnings
        exit_script "Y"
    fi
}

```

Abbildung 1: Bash-Funktion zum Ausführen des „deploy“-Kommandos (Quelle: Maurice Wilhelm)

```

147
148 ERROR: Exception Details
149 ERROR: Exception Primary Class: DatabaseException
150 ERROR: Exception Primary Reason: Error occurred and continueonerror set to false, stopping execution
151 ERROR: Exception Primary Source: 4.30.0
152 Please review automatically generated error log:
153 sqlcl-lb-error1763225703831.log
154 For additional information run using -debug and/or -log parameters.
155
156
157 An error has occurred:
158 liquibase.exception.CommandExecutionException: liquibase.exception.LiquibaseException: liquibase.exc
for changeset ticket1/cicd_from/ref_constraints/persons_fk.sql::1763224519843::CICD_FROM:
159 Reason: liquibase.exception.DatabaseException: Error occurred and continueonerror set to false,
160 Error occurred and continueonerror set to false, stopping execution.
161
162 Log file(s) location: /builds/dev/apex/cicd
163 Removing the decompressed artifact: /tmp/ce8b3cf4-4027-40f6-8b3d-306ba9c8c9b75478309994599417404...
164 Disconnected from Oracle Database 23ai Free Release 23.0.0.0.0 - Develop, Learn, and Run for Free
165 Version 23.4.0.24.05
.. >> [ERROR] Found "ERROR:" on line 148. See output above.
.. >> [ERROR] Found "ERROR:" on line 149. See output above.
.. >> [ERROR] Found "ERROR:" on line 150. See output above.
.. >> [ERROR] Found "ERROR:" on line 151. See output above.
.. >> [ERROR] Found "liquibase\.exception" on line 158. See output above.
.. >> [ERROR] Found "liquibase\.exception" on line 159. See output above.
.. >> [INFO] No "ORA-[0-9]{5}" found.

```

Abbildung 2: Farbige Ausgabe von ANSI-Codes im GitLab Runner-Log (Quelle: Maurice Wilhelm)

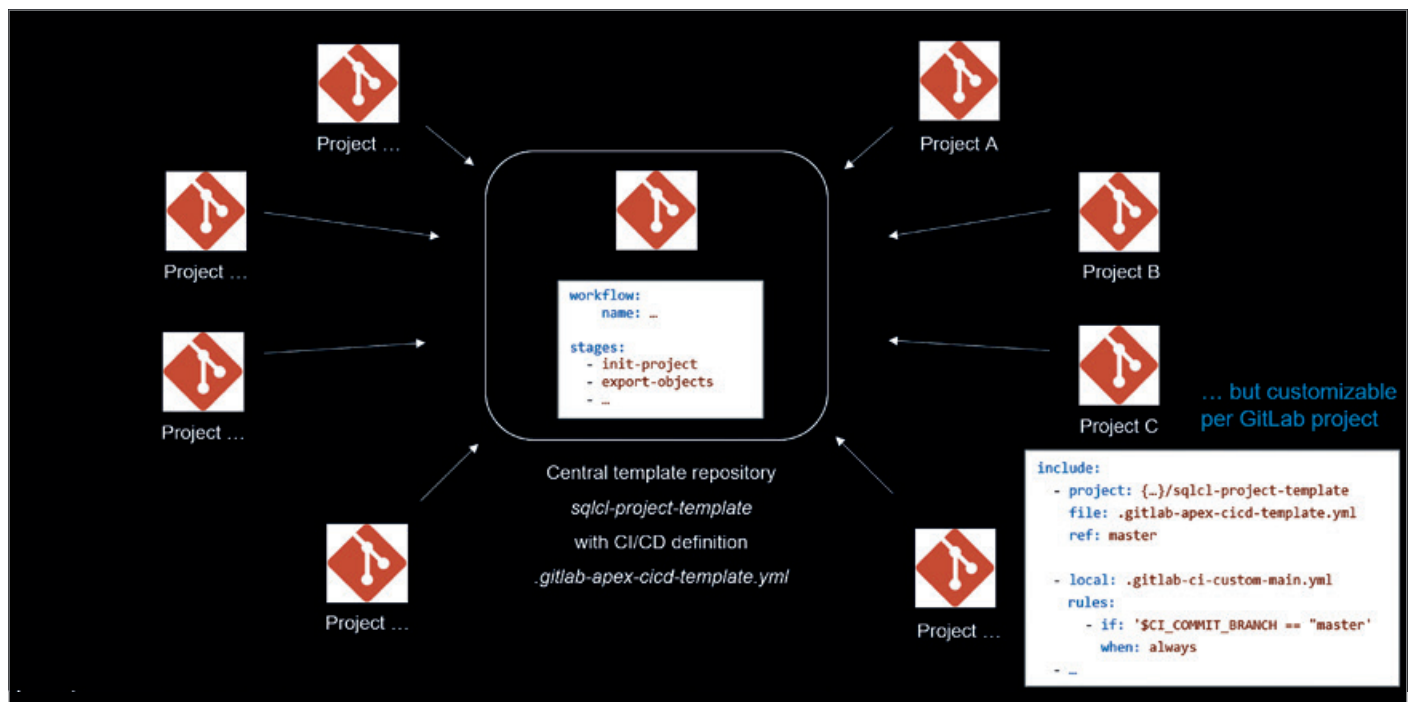


Abbildung 3: Zentral festgelegte Job-Definitionen (Quelle: Maurice Wilhelm)

```

sqlcl init project:
  stage: init-project
  image: !reference [.pipeline_job_settings, image]
  before_script:
    - !reference [.pipeline_job_settings, before_script]
    - !reference [.git_settings, before_script]
    - !reference [.git_changes, before_script]
  rules:
    - !reference [.pipeline_job_settings, rules]
    - if: $VAR_PIPELINE_TASK == "init"
      when: always
    - when: never
  script:
    - printf "%s\n" "${VAR_GLOBAL_DB_DEV_DEPLOYMENT_USER_PW}" | sh cicd.sh --sqlcl-exec="sql" --dl
    - !reference [.git_changes, script]
  
```

Abbildung 4: Definition des Jobs „Init Project“ (Quelle: Maurice Wilhelm)

beliebigen Verzeichnis aus direkt über seinen Namen ausgeführt werden, ohne den Speicherort angeben zu müssen. Zur Nutzung des Projects-Features werden im Bash-Skript definierte Funktionen aufgerufen, die analog die jeweiligen Befehle von SQLcl ausführen. [11] Die Auswahl der auszuführenden Funktion erfolgt über Argumentwerte beim Skriptaufruf:

- init
- export
- stage
- verify

- release (führt vorab den Befehl „verify“ aus, wie es die Oracle-Dokumentation empfiehlt)
- deploy (dies stellt eine Art Wrapper für die beiden Befehle „gen-artifact“ und „deploy“ dar)

In der folgenden Ausführung wird auf die Implementierung der zuvor angesprochenen Funktionen für den CI/CD-Workflow und die Fehlererkennung näher eingegangen.

Zur Fehlerbehandlung werden sämtliche Ausgaben von SQLcl – sowohl

Standardausgabe (stdout) als auch Fehlerausgabe (stderr) – in eine temporäre Logdatei umgeleitet. Anschließend wird diese Logdatei mithilfe regulärer Ausdrücke auf definierte Fehler- und Warnmuster analysiert. Bezeichner, die per regulärem Ausdruck als „Fehler“ klassifiziert sind, führen grundsätzlich zum Abbruch des Skripts mit Exit-Code 1 und damit auch zum Abbruch der Pipeline. Eine Ausnahme besteht nur dann, wenn ein erkannter Code explizit als „Warnung“ definiert wurde. Beim Ausführen von Liquibase-Changesets über den

„deploy“-Befehl kann dadurch beispielsweise gezielt gesteuert werden, dass bestimmte ORA-Fehlercodes, wie ORA-04063, nicht zum Abbruch führen, sofern sie als Warnung deklariert sind. Alle übrigen ORA-Fehler gelten weiterhin als Abbruchkriterium. Darüber hinaus werden in den Skript-Templates Platzhalter in der Form „_PLACEHOLDER_“ genutzt, die durch Funktionsvariablen zur Laufzeit ersetzt werden. Dies gewährleistet eine flexible und anpassbare Bereitstellung, die den speziellen Anforderungen eines Deployments auf unterschiedlichen Bereitstellungsumgebungen gerecht wird (siehe Abbildung 1).

In GitLab CI/CD-Logs werden ANSI-Codes standardmäßig unterstützt. [12] Für eine erleichterte Fehlerfindung können die Fehler- und Warnungs-Kennungen im Logfile durch das Bash-Skript mit roter beziehungsweise gelber Hintergrundfarbe markiert werden. Um dies zu erreichen, müssen alle Identifikatoren separat mittels „grep -nE“ innerhalb einer Schleife im Logfile gesucht und die Treffer-Zeilen in einer Variable gespeichert werden. Aufgrund der Tatsache, dass pro Zeile mehrere Treffer für einen regulären Ausdruck existieren können, muss jeder „Fehler“-Treffer mit den Elementen, die als „Warnung“ definiert sind, abgeglichen werden. Dafür wird jede Zeile unter Zuhilfenahme von „awk“ in einem Buffer gehalten und mittels Offsets positionsbasiert ausgewertet. Intern wird jeder Treffer über einen zusammengesetzten Schlüssel aus <index>.<field> in einem assoziativen Array (found_identifiers) gespeichert, um diese an nachstehende Bash-Funktionen weitergeben zu können – unter anderem für die visuelle und textuelle Auswertung. Im Folgenden ist das logische Datenmodell dargestellt.

```
found_identifiers
|
|— 0.identifier      → "ORA-04063"
|— 0.line_number    → 45
|— 0.index          → 21
|— 0.length         → 9
|— 0.type           → "warning"
|
|— 1. ...
```

Darstellung des logischen Datenmodells

In einem zweiten Schritt werden die einzelnen Funde zeilenweise zusammen-

geführt. Hierzu wird ein weiteres assoziatives Array (line_matches) verwendet, dessen Schlüssel der jeweiligen Logzeilennummer entspricht. Die zugehörigen Werte bestehen aus einer Liste aller Treffer dieser Zeile. Jeder Eintrag ist ein tabulatorgetrennter Datensatz mit den folgenden Feldern und endet mit einem Zeilenumbruch (\n):

```
line_matches[Zeilennummer]+=
[Index]<tab>[Identifer]<tab>[Leng
th]<tab>[Type]<newline> ...
```

Anschließend erfolgt pro Logzeile die eigentliche Markierung der Treffer. Das Logfile wird dafür zunächst zeilenweise eingelesen, sodass jede Zeile über ihren Index adressiert werden kann und auf Vorkommen geprüft wird. Aus dem Grund, dass das Einfügen von ANSI-Escape-Sequenzen die Länge der Zeichenkette verändert und somit nachfolgende Positionsindizes verschieben würde, werden die Treffer vor der Verarbeitung nach ihrem Startindex in absteigender Reihenfolge sortiert. Die Modifikation einer Zeile erfolgt somit rückwärts vom höchsten zum niedrigsten Index. Jeder Eintrag wird in seine Attribute zerlegt: Startindex innerhalb der Zeile, Bezeichner, Länge des Treffers und Typ (error oder warning). Der Typ bestimmt anschließend die farbliche Hervorhebung des Vorkommens: Fehler werden rot, Warnungen gelb markiert. Dafür wird die Zeile für jede Markierung in drei Segmente aufgeteilt: den Text vor dem Treffer, den farblich hervorgehobenen Identifikator und den Text nach dem Vorkommen. Dieser Vorgang wird für alle Funde der Zeile durchgeführt. Abschließend wird die modifizierte Zeile zurück ins Logfile geschrieben.

Bei Auftreten eines Fehlers wird dem Benutzer eine entsprechende Fehlermeldung ausgegeben und das Skript wird mit einem Fehler-Exit-Code beendet (siehe Abbildung 2). Treten keine Fehler auf, schließt das Skript erfolgreich mit einem Success-Exit-Code ab.

SQLcl Projects Template: Dieses Repository beinhaltet die Datei „gitlab-apex-cicd-template.yml“, in der die Pipeline-Job-Definitionen hinterlegt sind (siehe Abbildung 3). Hinweise zur Nutzung des Repositories können darüber hinaus in einer „README.md“ festgehalten werden. Sofern für die SQLcl-Projekte dieselbe Datenbankinstanz genutzt wird, empfiehlt es sich, fol-

gende CI/CD-Variablen auf Gruppenebene zu speichern: den Proxy-User für die Datenbank-Verbindung, die EZConnect-Verbindungsdaten sowie den SSH Private Key für das Pushen in untergeordnete Repositories. Pro GitLab-Projekt, in dem ein SQLcl-Projekt initialisiert wird, müssen lediglich die Schema-Namen für Entwicklungs-, Test- und Produktivumgebung angegeben werden. Darüber hinaus muss der SSH Public Key, wie bereits erwähnt, als Deploy Key mit Schreibrechten existieren. Er ist jedoch in mehreren GitLab-Projekten wiederverwendbar beziehungsweise auswählbar.

In der YAML-Datei werden Eingaben, die der Benutzer vor der Ausführung dynamisch angeben soll, als Variablen oder Inputs deklariert. Dabei ist zu beachten, dass Inputs in der aktuellen GitLab-Version nicht vollständig vererbt werden. Aus diesem Grund müssen sie in jedem GitLab-Projekt, das die CI-Datei referenziert, erneut definiert werden, weshalb es sich empfiehlt, auf CI/CD-Variablen zurückzugreifen. Der Pipeline-Name lässt sich unter dem Attribut „workflow“ angeben. Schritte zum Hinterlegen des SSH-Keys als IdentityFile, zur Konfiguration der Git-Parameter (zum Beispiel user.email und user.name) sowie die Kommandos für git fetch, git checkout, git commit und git push können als „Hidden Jobs“ (beginnend mit Punkt-Notation) zu den tatsächlichen Jobs am Anfang oder Ende referenziert werden. Darüber hinaus sollte jeder Job, der ein SQLcl-Projekt-Kommando ausführt, nur manuell gestartet werden. Hierzu lässt sich der Wert der Standard-Variable „CI_PIPELINE_SOURCE“ abfragen, welcher in diesem Fall auf „web“ gesetzt ist. Abbildung 4 zeigt einen Ausschnitt aus der Job-Definitionsdatei.

In einem Repository für ein SQLcl-Projekt kann abschließend diese Datei aus dem zentralen Template-Repository referenziert und mit eigenen Regeln ergänzt werden.

Fazit

SQLcl bietet mit „Projects“ eine neue Funktionalität, die die Verwaltung von Datenbankänderungen im Vergleich zur rein manuellen Nutzung der Liquibase-Erweiterung erheblich vereinfacht. Mit diesem neuen Feature lassen sich nicht nur Da-

tenbankobjekte automatisch exportieren, sondern auch inkrementelle Änderungen in Form von leicht lesbaren, SQL-basierten Liquibase Changesets generieren. Dadurch kann der gesamte CI/CD-Prozess, von der Code-Integration bis zum Deployment, nahtlos abgebildet werden. Dies führt zu einer erheblichen Reduzierung des manuellen Aufwands, beschleunigt den CI/CD-Prozess und minimiert Fehlerquellen bei der Versionierung. Zudem ermöglicht die Integration von SQLcl Projects in CI/CD-Pipelines durch die Steuerung per Kommandozeilenbefehlen eine noch nahtlosere Implementierung. Entwickler profitieren somit von einer deutlichen Steigerung der Effizienz, einer besseren Fehlerkontrolle und einer insgesamt schnelleren Softwareentwicklung.

Quellen

- [1] McGhan, D. (2025). An Overview of the Oracle SQLcl Projects Development Process. <https://danmcghan.hashnode.dev/an-overview-of-the-oracle-sqlcl-projects-development-process>
- [2] Koneru, N. M. K. (2025). The role of GitLab runners in CI/CD pipelines: Configuring EC2, Docker, and Kubernetes

build environments. The Eastasouth Journal of Information System and Computer Science, 2(03), 247. <https://doi.org/10.58812/esiscs.v2i03>

- [3] GitLab (o. J.). Executors. <https://docs.gitlab.com/runner/executors/>
- [4] Fernando, D. (2025). Unleashing the Power of the GitLab Container Registry: Your Integrated Image Hub. <https://medium.com/@deefernando/unleashing-the-power-of-the-gitlab-container-registry-your-integrated-image-hub-f66e313c7e61>
- [5] GitLab (o. J.). GitLab container registry administration. https://docs.gitlab.com/administration/packages/container_registry/
- [6] Morris, S. (2025). Einfaches Erstellen einer GitLab-CI/CD-Pipeline für ein Monorepo. <https://about.gitlab.com/de-de/blog/building-a-gitlab-ci-cd-pipeline-for-a-monorepo-the-easy-way/>
- [7] Mishra, V. (2025). GitLab environment variables demystified. <https://about.gitlab.com/blog/demystifying-ci-cd-variables/>
- [8] GitLab (o. J.). Predefined CI/CD variables reference. https://docs.gitlab.com/ci/variables/predefined_variables/
- [9] Abdul-Razak, H. (2025). The ultimate guide to token management at GitLab. <https://about.gitlab.com/blog/the-ultimate-guide-to-token-management-at-gitlab/>
- [10] GitLab (o. J.). GitLab CI/CD job token. https://docs.gitlab.com/ci/jobs/ci_job_token/
- [11] Smith, J. (2025). Getting started with Oracle Database CI/CD & SQLcl Projects. <https://www.thatjeffsmith.com/archive/2025/05/getting-started-with-sqlcl-projects/>

- [12] GitLab (o. J.). Scripts and job logs. <https://docs.gitlab.com/ci/yaml/script/>

Über den Autor

Maurice Wilhelm arbeitet als Consultant mit dem Schwerpunkt APEX Development und DevOps bei der Hyand Solutions GmbH in Ratingen. Neben der Erstellung von Webanwendungen mithilfe von Oracle APEX liegen seine Interessensschwerpunkte auf den Bereichen CI/CD und Automation mit Jenkins und GitLab sowie der Linux-Serveradministration (Anwendungsverwaltung).



Maurice Wilhelm
maurice.wilhelm@hyand.com

CLOUD NATIVE FESTIVAL VERPASST?

Jetzt **On-Demand-Ticket** buchen und
Vortragsaufzeichnungen anschauen!

ALLE ANGEBOTE
IM TICKETSHOP

#CloudLand2026



CloudLand
www.cloudland.org



SUMMIT DER SAP-COMMUNITY

COMPETENCE CENTER

Salzburg,
10. und 11. Juni 2026

Inklusive
KI-Erlebnisworkshop
Limitierte Plätze



- ✓ Support Desk und Monitoring für SAP S/4 Hana
- ✓ SolMan, Cloud ALM und Custom-Code-Migration
- ✓ S/4-Conversion, Testing und souveräne Systeme
- ✓ Process Mining und Automation
- ✓ Security, Compliance und Cloud

Wird gesponsert von:

HPE

intel

Uwe Werner
IT EINKAUFBSBERATUNG 



e3mag.com/de/cc-summit

BEST OF DOAG ONLINE

Eine Auswahl der besten DOAG News von Februar bis April 2026



Recap DevLand 2026: Bühne frei für eine neue Ära

Zwischen inspirierenden Keynotes, praxisnahen Sessions, strategischen Diskussionen und besonderen Community-Momenten: Eine Atmosphäre, die gleichermaßen zum Nachdenken, Lernen und Vernetzen einlud.



DOAG VOICES – Cloud-Transformation im Reality-Check: Automatisierung, Kosten und Kulturwandel

In der DOAG-VOICES-Podcast-Edition CloudTalk spricht Dave König mit Christian Reineke über die Realität von Cloud Computing – zwischen steigenden Kosten, kulturellen Hürden und technologischen Herausforderungen.



DOAG Datenbank Kolumne: Oracle auf Windows

Zugegebenermaßen ist Oracle auf Windows eher etwas für Masochisten, aber es gibt immer noch viele Datenbanken, vor allem kleinere, die auf Windows-Servern laufen.



DOAG VOICES – Agentische KI in der Praxis: Was Unternehmen jetzt wirklich brauchen

Warum scheitern so viele KI-Projekte – und was braucht Europa, um technologisch unabhängiger zu werden? Genau darüber spricht Dave König in der Edition FutureAI mit Sven Giesselbach von T-Systems.



DOAG VOICES – APIs, Verantwortung & Vielfalt in der Cloud

APIs verbinden Systeme — und Menschen. In dieser DevLand-Episode sprechen Dave König und Anna Klara Kneller darüber, warum erfolgreiche Software nicht nur aus Code besteht, sondern aus Kommunikation, Vielfalt und Verantwortung.



DOAG VOICES – Zwischen Hype und Handwerk: KI in der echten Softwareentwicklung

KI soll Softwareentwicklung revolutionieren — doch viele Projekte scheitern weiterhin an unklaren Anforderungen. Diese Podcastfolge von DOAG VOICES zeigt, warum bessere Fragen wichtiger sind als bessere Modelle.



Wir begrüßen unsere neuen Mitglieder

Natürliche Mitglieder:

- Jochen Brandhauer
- Björn Ständer
- Marcel Kölbel
- Dierk Lenz
- Andre Reissig

Korporative Mitglieder:

- ilume informatik AG, Repräsentant Markus Bernhart
- CapeVision GmbH, Repräsentant Jan Gorkow
- Klinikum Hanau GmbH, Repräsentant Herr Christian Hartmann



Termine

Mai

05

13.05.2026

DOAG DevTalk: Modernes SQL: Vector Search mit Markus Kissling und Christian Schwitalla
Online

18. - 19.05.2026

DOAG 2026 Datenbank mit Cloud Infrastructure. Konferenz rund um die Oracle-Datenbank und Cloud Infrastructure
Soltau, Heide Park

18. - 20.05.2026 17:00

APEX connect 2026 Konferenz mit zahlreichen Vorträgen und Workshops zu den Themen APEX, PL/SQL, JavaScript und Solutions
Soltau, Heide Park

19. - 22.05.2026

CloudLand 2026 - Das Cloud Native Festival. Community-Veranstaltung mit den Themen Cloud-native Software Engineering, Architecture, DevOps, Public Cloud, Organization, Culture, AI, ML, Security und vielen mehr
Soltau, Heide Park

Juni

06

12.06.2026

Thema wird noch benannt DB WebSession mit N.N.
Online

15. - 16.06.2026

AOUG Anwenderkonferenz 2026 - Digitale Systemsouveränität gestalten Daten- und Systemsouveränität im Zeitalter von Cloud und KI Verfügbarkeit, Kontrolle und Verantwortung neu gedacht
Wien

24.06.2026

DOAG DBTalk: Von Oracle zu Postgres, geht das so einfach?
DBTalk mit Christian Pfundtner
Online

25.06.2026

DOAG DevTalk: Alles rund um Oracle Forms - die Forms-Community kommt zusammen
DevTalk mit Frank Hoffmann
Online

August

08

14.08.2026

OCI Migration - wie, was, womit und warum nicht
DB WebSession mit Marco Pachaly-Mischke
Online

Impressum

Red Stack Magazin wird gemeinsam herausgegeben von DOAG 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 ist die Community-Publikation für angewandte Informations- und Kommunikationstechnologie (ITK) im Raum Deutschland, Österreich und Schweiz. Es setzt bewusst auf Technologieoffenheit mit Blick auf Anwendung und IT-Innovationen.

Es bildet die Interessensschwerpunkte der Anwenderinnen und Anwender ab – von Cybersicherheit bis Datenschutz, von Datenbank und Development über Data Analytics bis Digitalisierung, von Cloud und Infrastruktur über Künstliche Intelligenz bis Open Source und Soft Skills – vermittelt praktisches Wissen und fördert den Know-how-Transfer und die Netzwerkbildung zwischen den Leserinnen und Lesern.

Die Inhalte des Red Stack Magazin werden von ausschließlich ehrenamtlichen Autorinnen und Autoren eingereicht und von der Redaktion aufbereitet.

Red Stack Magazin 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. DOAG e.V. hält 100 Prozent der Stammeinlage der DOAG Dienstleistungen GmbH. DOAG e.V. wird gesetzlich durch den Vorstand vertreten; Vorsitzender: Jonas Janz

Redaktion:

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

Autorinnen und Autoren dieser Ausgabe
(in alphabetischer Reihenfolge):
Martin Bach, Maik Becker,
Marcel Boermann-Pfeifer, Carsten Czarski,
Wilfried Eigl, Dr. Holger Friedrich,
Knut Göttling, Henner Hucke,
Matthias Jakob, Dr. Benjamin Linnik,
Andreas Maier, Sonja Meyer,
Benjamin Nebgen, Ulrike Schwinn,
Ronny Weiss, Maurice Wilhelm
Andreas Wismann

Titel, Gestaltung und Satz:

Diana Tkach
DOAG Dienstleistungen GmbH
(Anschrift s.o.)

Fotonachweis:

Titel: © user4387299 | www.freepik.com
S. 06: © Dimitris Doutsopoulos
| www.ddphotography.gr
S. 07: © Dimitris Doutsopoulos
| www.ddphotography.gr
S. 08: © PIRO4D | www.pixabay.com
S. 12: © Carsten Czarski
S. 20: @ seagul | www.pixabay.com
S. 27: @ planet_fox | www.pixabay.com
S. 33: © Elchinator | www.pixabay.com
S. 38: © rawpixel.com | www.freepik.com
S. 45: © geralt | www.pixabay.com
S. 48: © A_Different_Perspective
| www.pixabay.com
S. 56: © pixelRaw | www.pixabay.com
S. 62: © user6309018 | www.freepik.com
S. 66: © WOKANDAPIX | www.pixabay.com
S. 72: © pixabay | www.pixabay.com
S. 76: © Elf-Moondance | www.pixabay.com
S. 77: © Tho-Ge | www.pixabay.com
S. 83: © CrazyDigitalTiger | www.pixabay.com
S. 93: © geralt | www.pixabay.com
S. 101: © freepik | www.freepik.com

Anzeigen:

sponsoring@doag.org

Mediadaten und Preise:

www.doag.org/go/mediadaten

Druck:

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

Inserentenverzeichnis

DOAG e.V.
www.doag.org

U 2, U 4

DOAG e.V.
www.doag.org

S. 3, S. 69, S. 98

JavaLand GmbH
www.javaland.eu

S. 82, U 3

DevLand GmbH

S. 44

B4Bmedia.net AG | E-3 Verlag
e3mag.com

S. 99

JavaLand

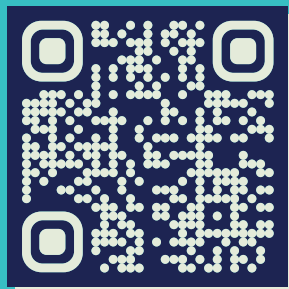
Super

Saver

Tickets sichern



**2.-4.
MÄRZ 2027**



www.javaland.eu



**Europa-Park
Rust**



APEX *connect*

18. - 20. Mai 2026



DOAG 2026
Datenbank
mit Cloud Infrastructure
18. - 19. Mai 2026



apex.doag.org

Heide Park Soltau

