www.ordix.de € 3,20

2|2014

ORDIX® news

einfach. gut. informiert.

„Laravel“

Sind Sie im Bereich Security richtig aufgestellt?

Unser **IT-Check** im Bereich Datenbank-Security

Gemeinsam analysieren wir mit Ihnen innerhalb eines Audits, wie gut Ihre Datenbanken gegen Datendiebstahl geschützt sind. Nach dem Audit erstellen wir ein individuelles, auf Ihre Bedürfnisse abgestimmtes Sicherheitskonzept. Dieses dokumentiert die festgestellten Schwachstellen und zeigt unsere empfohlenen Maßnahmen, Ihre Konfiguration sicherer zu gestalten. Mit der abschließenden, praktischen Umsetzung stellen wir den Schutz Ihrer Datenbank sicher.

Unser Vertriebsteam erstellt Ihnen gerne ein individuelles Angebot und zeigt Ihnen den Mehrwert für Ihr Unternehmen auf.

Kontaktieren Sie uns über: vertrieb@ordix.de

einfach. gut. beraten.

Eine Auswahl unserer IT-Checks finden Sie unter:
www.ordix.de/dienstleistung/ordix-it-checks.html

Böse Buben

Paderborn, Juni 2014

Wer fällt Ihnen zu dieser Überschrift ein? Putin? Sicher ja, denn er ist zur Zeit in Deutschland und den USA bei der Besetzung dieser Rolle sehr beliebt (insbesondere in Nachrichtensendungen und Print-Medien). Aber fragen Sie beispielsweise einen Schalke-04-Fan, Gerhard Schröder oder Helmut Schmidt, da bekommen Sie eine ganz andere Antwort. Bleiben wir im Osten und ziehen die lang gerühmte Julija Tymoschenko (Julia Timoschenko) heran. Ok, sie ist weiblich aber keinen Deut besser als ihre männlichen Kollegen. Immerhin wollte sie dem „bösen Buben" (sie nennt ihn „Drecks-kerl") Putin gleich eine Kalaschnikow an den Kopf halten und „in den Kopf schießen".

Das ist sicher nicht die feine englische Art. Womit wir direkt zu den britischen Royals kommen: Prinz Charles, auch so ein böser Bube? His Royal Highness hat Putin ganz nebenbei mit einem ehe-maligen Größenwahnsinnigen aus Deutschland/Österreich gleichgesetzt. Wobei solche britischen Vergleiche eigentlich weniger überraschend denn üblich sind.

Verlassen wir die rein politischen Bühnen und fragen nach Michael Hayden: böser Bube oder nicht? Kennen Sie nicht? Oh, immerhin ist Hayden, als Ex-NSA-Chef verantwortlich für das Abhören von Millionen Deutschen (und anderen) Weltbürgern, nicht böse – meint zumindest der Generalbundesanwalt Harald Range. Oder doch böse, weil auch Muttis Telefon abgehört wurde? Lange hatte es gedauert und viel Schelte von der kaum spürbaren Opposition musste es geben, bis Herr Range Ermittlungen aufnahm [1].

Ich meine: In jedem Fall böse! Denn Sätze wie „Wir töten Menschen auf Basis von Metadaten" [2] sind unzumutbar. Und jedem von uns, der Metadaten beispielsweise für sein Data Warehouse verarbeitet, müsste zukünftig ein eiskalter Schauer über den Rücken laufen, wenn er mit Metadaten zu tun hat. Ganz abgesehen davon, wenn Sie oder ich versuchen würden, einen Bundeskanzler abzuhören, wir hätten ganz schnell MAD, BND und wer weiß wen vor der Tür stehen. Nur bei der NSA kuschen alle – bis hin zur betroffenen Bundeskanzlerin. Peinlich und nicht vertrauensbildend.

Und was ist mit Michael Rogers (Nachfolger von Hayden), Edward Snowden oder auch Barack Obama/John Kerry? Böse Buben oder nicht? Schreiben Sie mir Ihre Meinung! Bei Snowden ist sich zumindest der amerikanische Außenminister sicher: Böse. Die politische Crème de la Crème in Deutschland ist sich da noch nicht ganz einig: Asyl möchte Snowden keiner anbieten, da unter-stützen unsere Regierung und einige Abgesandte des EU-Parlaments lieber ein paar Rechtsradikale in der Ukrainischen Regierung.

Unsere Autoren sind keine bösen Buben, ganz im Gegenteil: Sie berichten auch in dieser News über interessante Themen. Ein guter Punkt, an das Thema Metadaten anzuknüpfen: Die METADATA Packages in Oracle 11g sind da hoffentlich eine unproblematische Erleichterung.

Während wir zu den neuen Funktionalitäten bei Oracle 12c bereits in Runde vier gehen, starten in dieser Ausgabe Artikelreihen, die Ihnen die Versionen 12.1 (Informix) und 2014 (MS SQL Server) näher bringen sollen.

Artikel zum Thema Kapazitätsplanung, GoldenGate und dem Oracle Mobile Server runden diese stark Datenbank orientierte Ausgabe ab.

Für Entwickler bieten wir mit Laravel einen schnellen Einstieg in das hochgelobte PHP-Framework, Lambda Expressions führen uns bei der Vorstellung von Java 8 weiter und wieder einmal gibt es etwas über GWT zu berichten (über Google hätte ich natürlich auch ein Editorial schreiben können [3]).

Auch unsere Artikel über Microservices und die Fortsetzung der Reihe „Qualitätssicherung in Softwareprojekten" sollten für Entwickler ihren Reiz haben. Und wer in dieser Ausgabe Artikel über Projektleitung und -management vermisst, dem sei unsere Broschüre [4] empfohlen.

Ich wünsche Ihnen viel Spaß beim Lesen und genießen Sie den Sommer, der uns hoffentlich nicht ständig mit Unwettern überrascht [5], an denen die bösen Buben aber ausnahmsweise keine Schuld tragen – oder vielleicht doch?

Ihr

Wolfgang Kögler

[1] http://www.tagesschau.de/inland/generalbundesanwalt-nsa110.html

[2] Gemeint ist beispielsweise, auf Basis von Daten eines Mobiltelefons mit Drohnen Menschen zu „eliminieren" und dabei durchaus in Kauf zu nehmen, zusätzliche Opfer oder auch falsche Personen zu treffen. http://www.golem.de/news/ex-nsa-chef-hayden-wir-toeten-menschen-auf-basis-von-metadaten-1405-106409.html.

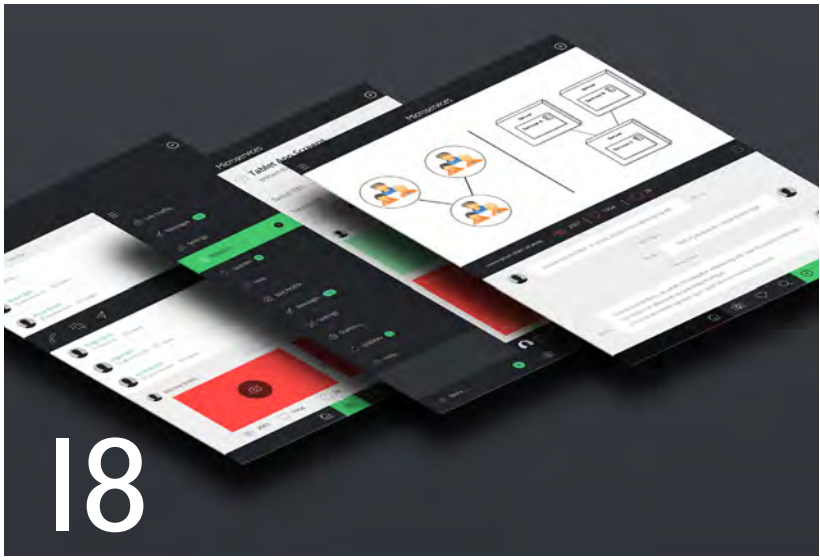
[3] http://www.spiegel.de/netzwelt/netzpolitik/eu-urteil-zum-recht-auf-vergessenwerden-spanier-gegen-google-a-969064.html

[4] http://www.ordix.de/images/ordix/infomaterial/Projektmanagement_Seminare.pdf

[5] http://www.sueddeutsche.de/panorama/unwetter-in-nordrhein-westfalen-keine-entwarnung-menschen-sollen-zu-hause-bleiben-1.1992956

6

GWT und Web Storage



18

Microservices

Datenbanken

Datenbanken

Softwarearchitektur

Lambda Expressions

PHP-Framework „Laravel“

Java/JEE

Open Source

Aktuell

Impressum

Herausgeber:	ORDIX AG Aktiengesellschaft für Softwareentwicklung, Beratung, Schulung und Systemintegration, Paderborn
Redaktion:	Jens Pothmann, Evelyn Ernst
V.i.S.d.P.:	Benedikt Georgi, Wolfgang Kögler
Anschrift der Redaktion:	ORDIX AG | Westernmauer 12 - 16 | 33098 Paderborn Tel.: 05251 1063-0 | Fax: 0180 1673490
Gestaltung/Layout:	Jens Pothmann
Auflage:	7.000 Exemplare
Druck:	Druckerei Bösmann, Detmold
Bildnachweis:	© psdblast.com | Help icon, life belt © flickr.com | Hard Drive Repair | wwarby © canstockphoto.de | csp8080959 | maxxyustas © istockphoto.com | Computer Geek | sdominick © pixeden.com | Perspective Tablet Mock-Up 2 © freepik.com | Free abstract geometric background © grapicburger.com | 3D Wooden Logo Mockup
Autoren:	Andreas Flügge, Sebastian Grimm, Patrick Hecker, Andreas Jordan, Lars Hendrik Korte, Wolfgang Kögler, Maik Krawinkel, Sven Loer, Philipp Loer, Christoph Lütkevedder, Marcus Meisenberg, Klaus Reimers, André Stefaniak, Tobias Ummler
Copyright:	Die ORDIX® news erscheint vierteljährlich. Alle Eigentums- und Nachdruckrechte, auch die der Übersetzung, der Vervielfältigung der Artikel oder von Teilen daraus, sind nur mit schriftlicher Zustimmung der ORDIX AG gestattet.
Warenzeichen:	Einige der aufgeführten Bezeichnungen sind eingetragene Warenzeichen ihrer jeweiligen Inhaber. ORDIX® ist eine registrierte Marke der ORDIX AG.
Haftung:	Eine Haftung für die Richtigkeit der Veröffentlichungen kann trotz sorgfältiger Prüfung durch die Redaktion vom Herausgeber nicht übernommen werden.

Sie können die Zusendung der ORDIX® news jederzeit ohne Angabe von Gründen schriftlich (z.B. Brief, Fax, E-Mail) abbestellen.

GWT und Web Storage

So erstellen Sie eine Offline-Applikation mit GWT

Das Google Web Toolkit (GWT) wird oft eingesetzt, um komplexe browserbasierte Anwendung zu realisieren. Doch was passiert, wenn keine Verbindung zum Internet besteht und der Benutzer somit offline ist? In diesem Artikel zeigen wir, wie eine browserbasierte offline-fähige Anwendung mit dem GWT realisiert werden kann. Zunächst wird aufgezeigt, welche Technologien eingesetzt werden können, um eine Offline-Fähigkeit anzubieten. Im zweiten Schritt gehen wir näher darauf ein, wie die vorgestellte Technologie im GWT eingesetzt werden kann.

Browser-Anwendung goes offline

Ein Browser ist nichts anderes als ein Programm, das Daten von einem Server anfragt und die Antwort dem Benutzer anzeigt. Soweit so gut! Der Teufel steckt aber wie so oft auch hier im Detail. Wenn eine Anfrage an einen Dienst im Internet gestellt wird, setzt dies zwingend eine Verbindung zum entsprechenden Server voraus. Doch was ist, wenn keine Internetverbindung besteht oder diese abbricht?

Oft gibt es die Anforderung an Browser-Anwendungen, dass sie auch offline funktionsfähig sind. Die Frage ist, welche Möglichkeiten gibt es, um Daten lokal (also auf dem System des Browsers) abzulegen und somit eine Anwendung offline-fähig zu machen. In diesem Artikel wird diese Frage allgemein per JavaScript und speziell mit Hilfe des GWT Framework beantwortet.

Wozu eine Offline-Applikation?

Es gibt viele Gründe, die für eine offline-fähige Browser-Applikation sprechen. Ein Argument für die Offline-Fähigkeit, stellt die bereits angesprochene schlechte oder nicht

vorhandene Internetanbindung dar. Jeder, der einmal versucht hat über eine schlechte Internetanbindung effektiv im Internet zu arbeiten, kennt das Problem. Lange Ladezeiten verderben den Spaß und schmählern die Produktivität. Eine Browser-Anwendung, die nicht zwingend eine Internetverbindung benötigt, hat dieses Problem nicht.

Als einfaches Beispiel ist hier eine Nachrichtenseite zu nennen, die im Hintergrund dann alle Artikel des aktuellen Tages herunterlädt, wenn eine Internetverbindung besteht. Fährt der ICE zur Arbeit durch einen Tunnel, kann die Anwendung die im Hintergrund geladenen Artikel trotzdem anzeigen. Im Idealfall bemerkt der Anwender also gar nicht, ob gerade eine Internetverbindung besteht oder nicht.

Die Technologien

Die spannende Frage ist, welche Möglichkeiten es gibt, Daten auf dem Client abzulegen. Eine - schon sehr betagte - Möglichkeit stellen die HTTP Cookies dar. Diese haben aber laut Spezifikation nur 4 Kilobyte für die Datenablage zur Verfügung. Außerdem liegt das Hauptmerkmal der Cookies darauf, die gespeicherten Informationen, wie zum Beispiel Benutzereinstellungen, Passwörter und Warenkörbe, dem Server zu senden. Cookies werden bei jedem Request und Response zwischen Client und Server übertragen. Somit würden die Daten jedes Mal mitgeschickt, auch wenn z.B. nur ein Bild vom Server angefragt wird.

HTML5 bietet eine vom W3C spezifizierte Lösung an: Das Web Storage. Die Speicherkapazität liegt bei 5 Megabyte und wird mittlerweile von allen gängigen Browsern unterstützt. Ebenfalls eignet sich die Programmierschnittstelle IndexedDB, um Daten in einer einfachen Form in der erstellten Datenbank zu speichern. Dies bietet eine bessere Performance und die Möglichkeit, größere Datenmengen als das Web Storage zu halten.

In diesem Artikel legen wir das Augenmerk auf das Web Storage, da hierfür bereits eine Implementierung für GWT vorhanden ist. Dagegen ist IndexedDB aktuell noch nicht in GWT integriert.

Local Storage und Session Storage

Bevor wir uns die Verwendung von Web Storage anschauen, soll zunächst der Unterschied zwischen den beiden Varianten **`localStorage`** und **`sessionStorage`** erläutert werden. Mit der Variante **`localStorage`** werden die Daten dauerhaft auf der Client-Seite abgelegt. Die Daten überleben somit einen Browser-Neustart. Die Daten des **`localStorage`** werden erst gelöscht, wenn der Benutzer diese händisch oder die Anwendung per JavaScript entfernt. Ähnlich in der Handhabung sind die Objekte des **`sessionStorage`**. Diese besitzen, wie der Name schon sagt, die Lebensdauer einer Browsersession. Wird der Browser geschlossen, läuft die Session ab und die Daten werden im **`sessionStorage`** gelöscht.

```
var lStorage = window.localStorage;
lStorage.setItem('key', 'value');
var key = lStorage.key(0);
var value1 = lStorage.getItem(key);
var value2 = lStorage.getItem('key');
lStorage.removeItem('key');
var localStorageCount = lStorage.length;
lStorage.clear();
```

Abb. 1: Verwendung von `localStorage`.

```
lStorage.setItem('key', JSON.stringify(object));
var value = JSON.parse(lStorage.getItem('key'));
```

**Abb. 2: JSON-String erzeugen und ablegen;
JSON String wieder erhalten und Objekt erzeugen.**

Der Umgang mit den Storage-Objekten ist insgesamt sehr einfach gestaltet. Die Daten werden mit Hilfe von Key-Value-Paaren abgelegt. Jemand, der unter Java mit einer **`HashMap`** umgehen kann, wird sich daher auch mit Web-Storage-Objekten anfreunden können. Die Daten werden mit Hilfe eines Schlüssels abgelegt, der nichts anderes als ein String ist. Mit diesem kann man jederzeit an die zuvor abgelegten Daten gelangen. Weiterhin können die Daten von jedem möglichen Typ sein, den JavaScript unterstützt (Strings, Integer, Float, Boolean, ...).

Die Abbildung 1 veranschaulicht, welche Methoden das Web Storage API zur Verfügung stellt. Mit der Methode **`setItem(key,data)`** werden die Daten unter dem angegebenen Key abgelegt. Die Methode **`getItem(key)`** liefert den Wert für den übergebenen Schlüssel. Den i'ten Key aus dem **`localStorage-`** oder **`sessionStorage`**-Objekt erhält man mit der Methode **`key(i)`**. Eine Schleife, die über alle Elemente iteriert, ist also ohne Probleme möglich. Dazu braucht man aber noch das Attribut **`length`**. Dieses Attribut enthält die Anzahl der abgelegten Elemente. Soll ein Element entfernt werden, ruft man die Methode **`removeItem(key)`** mit dem richtigen Schlüssel auf. Soll nicht nur ein Element gelöscht werden, sondern alle, so benutzt man die Methode **`clear()`**.

Ist es möglich komplexere Objekte abzulegen?

Im Moment können wir nur einfache Datentypen wie Zeichenketten oder Zahlen ablegen. Aber was ist mit komplexen Objekten? Hier hilft uns die JavaScript Object Notation - kurz JSON (siehe Abbildung 2). Mit der Methode **`JSON.stringify(object)`** kann ein JSON-String erzeugt werden, welcher anschließend im Storage-Objekt abgelegt werden kann. Die Methode **`JSON.parse(jsonString)`** kann aus einem JSON-String wieder ein JavaScript-Objekt erzeugen.

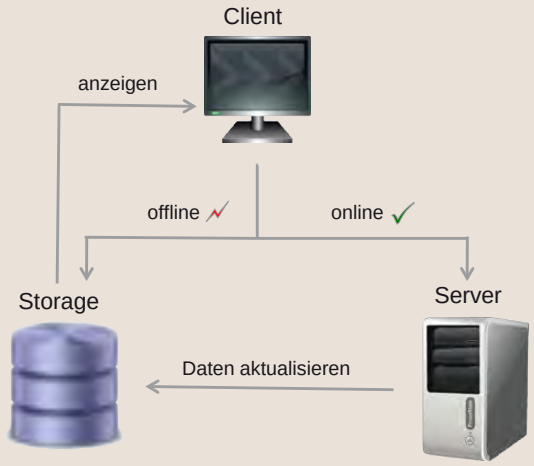


Abb. 3: Aufbau einer Nachrichtendienstseite.

```
/**
 * Holen der aktuellen Nachrichtenheader
 */
public void getAktuelleNachrichten() {
    nachrichtenService.getAktuelleServerNachrichten(new AsyncCallback<List<NachrichtenHeader>>() {
                public void onSuccess(List<NachrichtenHeader> result) {
                    // Aufruf zum vollständigen Nachrichten laden
                    getNachrichten(result);
                }
                public void onFailure(Throwable caught) {
                    // Bei einem beliebigen Fehler wird eine leere Liste übergeben
                    getNachrichten(new ArrayList<NachrichtenHeader>());
                }
            });
}
```

Abb. 4: Verwaltung der Nachrichten im Storage.

```
/**
 * Speicherung und Verwaltung der Nachrichten in einem Storage
 */
@SuppressWarnings("boxing")
public class NachrichtenStorage {
    ...
    private NachrichtenStorage() {
        storage = Storage.getLocalStorageIfSupported();
    }

    public List<NachrichtenHeader> getVorhandeneNachrichtenHeaders() {
        List<NachrichtenHeader> nachrichtenHeader = new ArrayList<NachrichtenHeader>();

        for (int i = 0; i < storage.getLength(); i++) {
            String key = storage.key(i);
            nachrichtenHeader.add((parseNachrichtenHeader(storage.getItem(key))));
        }

        return nachrichtenHeader;
    }

    public void speichern(Nachricht nachricht) {
        storage.setItem(nachricht.getId().toString(), stringify(nachricht));
    }

    public void loeschen(Integer id) {
        storage.removeItem(id.toString());
    }

    public void alleLoeschen() {
        storage.clear();
    }
    ...
}
```

Abb. 5: Aufruf zum Server für die aktuellen Nachrichteninformationen.

Browser-Unterstützung

Die Unterstützung der Web-Storage-Technologie in den unterschiedlichen Browser ist mittlerweile sehr gut. Selbst eine Unterstützung im Internet Explorer ab der Version 8 ist vorhanden. Genauso wenige Probleme wird man mit den aktuellen mobilen Browsern von Android und iOS haben.

Des Weiteren sind auch die Entwicklerwerkzeuge nicht zu vernachlässigen. Die üblichen Verdächtigen (Firefox, Chrome und Internet Explorer ab der Version 9) haben keine Probleme mit den integrierten Entwicklungswerkzeugen, die abgelegten Daten anzuzeigen und zu editieren. Weitere Plugins liefern eine bessere Übersicht über die abgelegten Daten. Beim Internet Explorer ist die Handhabung leider nicht so komfortabel, aber insgesamt trotzdem möglich.

GWT und Web Storage

Wir haben nun geklärt, welche Technologien grundsätzlich möglich sind und wie sie funktionieren. Jetzt zeigen wir den Einsatz von Web Storage mit GWT.

Die gute Nachricht vorweg: Das GWT bietet bereits ein API zum Einsatz von Web Storage an. Wir müssen folglich keinen nativen JavaScript-Code mit Hilfe von JSNI (JavaScript Native Interface) programmieren. Es werden GWT-Klassen zur Verfügung gestellt, um die Web-Storage-Funktionalität einsetzen zu können. Alle notwendigen Klassen befinden sich in dem Paket **`com.google.gwt.storage.client`**.

Da nicht alle Browser die Storage-Funktion unterstützen, muss zunächst festgestellt werden, ob die Funktionalität verfügbar ist. GWT bietet hier die statische Methode **`isSupported`** der Klasse **`Storage`** an, um dies zu überprüfen. Gibt diese Methode **`false`** zurück, können wir die Storage-Funktionalität nicht einsetzen.

Im nächsten Schritt wird ein Storage-Objekt benötigt, mit dem Daten lokal abgelegt werden können. Die beiden Methoden **`getLocalStorageIfSupported`** und **`getSessionStorageIfSupported`** liefern uns die Storage-Typen.

Beispiel anhand einer Nachrichtenseite

Als Anwendungsbeispiel dient eine Seite, welche Nachrichten anzeigt und immer „verfügbar“ sein soll. Verfügbar bedeutet in diesem Fall, dass keine lästigen Verbindungs- oder andere Fehlerseiten beim Laden auftauchen. Fällt die Verbindung aus, soll der lokale Nachrichtenbestand angezeigt werden (siehe Abbildung 3). Somit ist eine Offline-Fähigkeit gegeben.

Um Nachrichtenobjekte lokal verwalten zu können, wird eine Hilfsklasse **`NachrichtenStorage`** bereitgestellt. Diese Klasse bietet Methoden zum Initialisieren des Storage-

Objektes, bzw. zum Speichern, Löschen, Auslesen und Umwandeln der Nachrichten. Das Speichern erfolgt über die Methode **speichern**, die die Nachrichten-ID als Key und die komplette Nachricht (umgewandelt als String) als Wert übergeben bekommt. Zum Auslesen aller Nachrichten, wird die gesamte Storage-Länge durchlaufen und ausgelesen (siehe Abbildung 4).

Der Ablauf der Anwendung ist wie folgt: Zu Beginn werden vom Server die Hauptinformationen (**NachrichtenHeader**) der aktuellen Nachrichten (siehe Abbildung 5) geladen. Mit dieser Klasse kann auf einfachem Weg festgestellt werden, ob eine Nachricht bereits lokal abgelegt ist. Im zweiten Schritt kann geprüft werden, ob diese noch aktuell ist. In einem **NachrichtenHeader**-Objekt steckt ein Hash-Attribut, mit dem wir überprüfen können, ob sich der Nachrichtentext einer Nachricht verändert hat. Ob alle aktuellen Nachrichten lokal abgelegt sind, wird ebenfalls vom Refresh-Button überprüft.

Die Kommunikationen zum Server findet - wie bei GWT üblich - über RPCs (Remote Procedure Calls) statt. In der Klasse **NachrichtenVerwaltungsService** sind die notwendigen Methoden untergebracht. Hier werden die **NachrichtenHeader**-Objekte angefragt und ggf. neue Nachrichten vom Server geladen. Besteht die Verbindung und wird nicht unterbrochen, werden mit der Methode **onSuccess** die neu geladenen Nachrichten als Storage-Objekt abgelegt. Kommt es zu Fehlern, greift die Methode **onFailure** und wir lesen nur die Daten aus dem Storage. Beide Methoden erzeugen ein Event, welches signalisiert, dass das GUI alle vorhandenen Nachrichten anzeigen kann (siehe Abbildung 6).

Datenumwandlung Java-Objekt ‹=› Storage

Da das Web Storage nur String-Werte aufnehmen kann, nutzen wir für die Datenumwandlung zum Speichern und Laden das JSON-Format. Bevor ein neu geladenes Nachrichtenobjekt im Storage abgelegt werden kann, muss es aber in das JSON-Format gebracht werden. Wie ein JSON-Objekt erzeugt bzw. wie es ausgelesen wird, ist in Abbildung 7 dargestellt. Die Methode **parseNachricht** erzeugt aus einem JSON-String ein fertiges Nachrichtenobjekt. Hier werden die einzelnen Attribute aus dem JSON-String herausgezogen, um die Attribute des Nachrichtenobjektes zu füllen.

Den Schritt in die andere Richtung übernimmt die Methode **stringify**. Diese macht aus einem Nachrichtenobjekt ein JSON-Objekt, welches lokal abgelegt werden kann. Hilfsklassen um ein JSON-Objekt zu erzeugen oder auszulesen, sind unter **com.google.gwt.json.client** zu finden. Dazu muss aber in der ***.gwt.xml** das Modul **<inheritsname="com.google.gwt.json.JSON"/>** eingebunden werden.

Leider ist das manuelle Ein- und Auslesen der einzelnen Attribute relativ mühselig. Abhilfe schafft an dieser Stelle

```
/**
 * Speicherung und Verwaltung der Nachrichten in einem Storage
 */
@SuppressWarnings("boxing")
public class NachrichtenStorage {

    ...
    private NachrichtenStorage() {
        storage = Storage.getLocalStorageIfSupported();
    }

    public List<NachrichtenHeader> getVorhandeneNachrichtenHeaders() {
        List<NachrichtenHeader> nachrichtenHeader = new ArrayList<NachrichtenHeader>();

        for (int i = 0; i < storage.getLength(); i++) {
            String key = storage.key(i);
            nachrichtenHeader.add((parseNachrichtenHeader(storage.getItem(key))));
        }

        return nachrichtenHeader;
    }

    public void speichern(Nachricht nachricht) {
        storage.setItem(nachricht.getId().toString(), stringify(nachricht));
    }

    public void loeschen(Integer id) {
        storage.removeItem(id.toString());
    }

    public void alleLoeschen() {
        storage.clear();
    }
    ...
}
```

Abb. 6: Aufruf zum Server für die aktuellen Nachrichten.

```
private Nachricht parseNachricht(String jsonString) {

    Nachricht nachricht = new Nachricht();

    JSONObject nachrichtJson = (JSONObject) JSONParser.parseStrict(jsonString);
    nachricht.setId((int) ((JSONNumber) nachrichtJson.get("id")).doubleValue());
    nachricht.setUeberschrift(((JSONString) nachrichtJson.get("ueberschrift")).stringValue());
    nachricht.setText(((JSONString) nachrichtJson.get("text")).stringValue());
    nachricht.setHash(((JSONString) nachrichtJson.get("hash")).stringValue());
    nachricht.setDatum(new Date((long) ((JSONNumber) nachrichtJson.get("datum")).doubleValue()));

    return nachricht;
}

private String stringify(Nachricht nachricht) {

    JSONObject nachrichtJson = new JSONObject();
    nachrichtJson.put("id", new JSONNumber(nachricht.getId()));
    nachrichtJson.put("ueberschrift", new JSONString(nachricht.getUeberschrift()));
    nachrichtJson.put("text", new JSONString(nachricht.getText()));
    nachrichtJson.put("hash", new JSONString(nachricht.getHash()));
    nachrichtJson.put("datum", new JSONNumber(nachricht.getDatum().getTime()));

    return nachrichtJson.toString();
}
```

Abb. 7: Umwandlung von JSON in Java-Objekte.

Glossar

Google Web Toolkit (GWT)
Das Google Web Toolkit ist ein Werkzeug zur Entwicklung von Webanwendungen.

Remote Procedure Call (RPC)
Ein RPC ist ein Aufruf von Methoden, die von einem anderen Rechner/Server ausgeführt werden und die Ergebnisse zurück liefern.

W3C
Das World Wide Web Consortium ist das Gremium zur Standardisierung der das World Wide Web betreffenden Technologien.

Links

- [1] Framework gwt-jackson:
  https://github.com/nmorel/gwt-jackson
- [2] GWT Webseite:
  http://www.gwtproject.org/doc/latest/DevGuideHtml5Storage.html
- [3] Kompatibilitätsübersicht für den Support von HTML5, CSS3, etc.:
  http://caniuse.com/#search=localstorage
- [4] HTML5-Funktionen:
  http://www.html5rocks.com/de/features/storage
- [5] Historie und Ausblick des Local Storage:
  http://diveintohtml5.info/storage.html
- [6] Bericht zur Einstellung von Web SQL:
  http://www.golem.de/1011/79548.html
- [7] Historie und Ausblick HTML5
  http://www.sitepoint.com/html5-browser-storage-past-present-future/
- [8] DOM Storage Guide:
  https://developer.mozilla.org/en-US/docs/Web/Guide/API/DOM/Storage

Bildnachweis

- © flickr.com | Hard Drive Repair | wwarby
- © canstockphoto.de | csp8080959 | maxxyustas

z.B. das Framework „gwt-jackson“. Hier wird mit Hilfe von Annotationen erreicht, dass die Attribute ohne weiteres richtig ein- und ausgelesen werden [1].

Fazit

Mit der Möglichkeit Daten im Web Storage abzulegen, kann man sich sehr schnell anfreunden. Die Handhabung mit JavaScript, aber auch in Verbindung mit GWT ist schnell erlernt. Durch Hilfsklassen können Java-Objekte problemlos im Storage als JSON-Objekte abgelegt werden. Auch über eine gute Browser-Unterstützung und Debugging-Werkzeugen kann sich der Entwickler, aber auch der spätere Anwender freuen. Einfach strukturierte Objekte können effizient abgelegt werden.

Wenn die Objekte komplexer werden, stößt man mit dem Aufbau von Key Value Pairs aber an die Grenzen. Alle Daten die abgelegt werden, landen in einer großen Hashmap. Komplexe Gebilde lassen sich besser in einer relationalen Datenbank abbilden, als mit einer Hashmap. Nichtsdestotrotz ist Web Storage eine gelungene Weiterentwicklung der HTTP Cookies.

Christoph Lütkevedder
(info@ordix.de)

Maik Krawinkel
(info@ordix.de)

Qualitätssicherung in Softwareprojekten (Teil II)

Vom Groben ins Feine

Im ersten Teil dieser Reihe [1] haben wir u.a. das statische Testen dem dynamischen Testen gegenübergestellt. Reviews und Code-Analyse wurden dabei zunächst nur grob umrissen. In diesem Artikel werden nun einige Details zu diesen beiden Themen näher beleuchtet.

Statisches Testen mit Reviews

Typische Prüfobjekte der statischen Tests sind Anforderungsdokumente, Spezifikationen, der Source Code des Softwareproduktes selbst, sowie weitere schriftliche Ergebnisse eines Softwareentwicklungsprozesses. Das Prüfen solcher Dokumente kann mit verschiedenen Arten von Reviews erfolgen. In der Regel sind dabei die folgenden Rollen vertreten: Ein Moderator, mehrere Gutachter, sowie ein oder mehrere Autoren des zu begutachtenden Dokuments. Darüber hinaus sollte es zum Grundverständnis aller Beteiligten gehören, dass ausschließlich Dokumente begutachtet werden und nicht der oder die Autoren zu bewerten sind. In der Norm IEEE 1028 sind folgende vier Review-Arten definiert:

- Technischer Review
- Informeller Review
- Walkthrough
- Inspection

Bevor auf die Unterschiede der einzelnen Review-Arten eingegangen wird, betrachten wir den fünfstufigen Review-Prozess. Dieser dient grundsätzlich als Leitfaden und Orientierung für alle Review-Arten, auch wenn er nicht bei jeder Art von Review im Detail befolgt werden kann:

Planung

In der Planung entscheidet das Management, welche Dokumente berücksichtigt werden, welche Review-Art angewandt wird, wie viel Zeit die Durchführung des Review beanspruchen darf und welche Personen geeignet sind, den Review durchzuführen.

Einführung

Häufig wird diese Stufe auch als „Vorbesprechung“ oder „Initialisierung“ bezeichnet. Sie dient dazu, alle am Review beteiligten Personen (Gutachter) mit Informationen zu versorgen. Dazu werden u.a. die Ziele des Review kommuniziert sowie die Review-Gegenstände (Dokumente) bereitgestellt. Hierbei können auch Checklisten für die Prüfkriterien hilfreich sein. Die Einführung kann in Form einer Einladung (E-Mail) kommuniziert werden oder auch durch ein erstes Vorabtreffen des Review-Teams.

Vorbereitung

Vor der eigentlichen Review-Sitzung sollten sich die Teilnehmer auf die Sitzung gut vorbereitet haben d.h. jeder hat bereits die zur Verfügung gestellten Dokumente gelesen, ist die Checklisten durchgegangen und hat Fragen bzw. Unklarheiten formuliert. Nur auf Basis dieser Vorarbeit macht eine Review-Sitzung überhaupt Sinn.

Review-Sitzung

Eine Review-Sitzung sollte nicht länger als 2 Stunden dauern, kann aber an einem Folgetag fortgesetzt werden. Der Moderator darf die Sitzung abbrechen, wenn ein vernünftiges Ergebnis nicht absehbar ist, zu wenig Teilnehmer erschienen sind oder die Teilnehmer nicht ausreichend vorbereitet sind.

Wichtig ist, wie eingangs bereits erwähnt, dass das Dokument (Prüfobjekt) zur Diskussion steht und nicht der Autor. Dieser darf selbst natürlich nicht gleichzeitig als Gutachter agieren, sollte aber anwesend sein. Jeder Gutachter muss angemessen Zeit haben, seine Befunde vorzutragen. Alle Befunde sind zu protokollieren und zu gewichten. Die Entwicklung von Lösungen ist allerdings nicht die Aufgabe des Review-Teams. Abschließend ist seitens der Gutachter eine Empfehlung hinsichtlich der Abnahme des Prüfobjektes abzugeben.

Nachbereitung

In der Nachbereitung entscheidet das Management, ob sie den Empfehlungen des Review folgen oder sie eigenverantwortlich ablehnen. Der Autor wird seinerseits die protokollierten Befunde korrigieren und dem Management vorlegen. War das Resultat des ersten Review nicht befriedigend, wird ein zweiter Review durchgeführt. Dieser kann dann in verkürzter Form stattfinden.

Auch der Review-Prozess selbst sollte bewertet und ggf. angepasst und damit kontinuierlich verbessert werden. Häufig wiederkehrende Fehlerarten, Mängel und Befunde weisen auf generelle Schwächen im Softwareentwicklungsprozess hin.

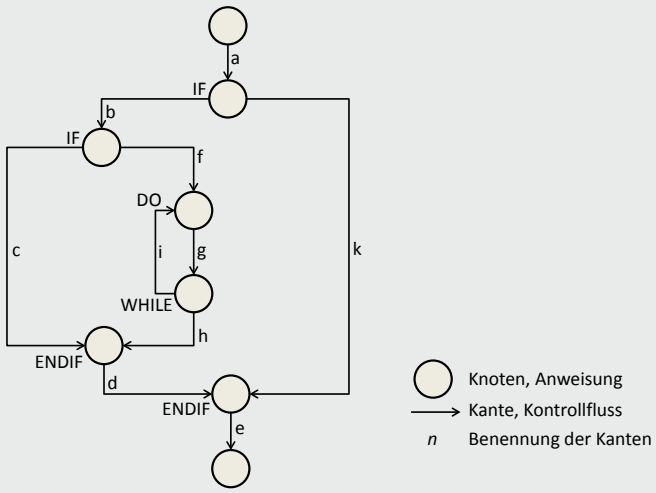


Abb. 1: Kontrollflussdiagramm (Quelle [1]).

Review-Arten

Nachdem wir die fünf Stufen des Review-Prozesses betrachtet haben, gehen wir nun auf die Unterschiede der vier Review-Arten ein:

Technisches Review

Ein technischer Review dient der fachlichen Prüfung eines Dokuments (z.B. Architekturentwurf). Hier werden Fragen geklärt wie:

- Entspricht der Entwurf der Spezifikation?
- Sind noch Fragen offen, die geklärt werden müssen?
- Sind bereits Fehler erkennbar?
- Gibt es Alternativen?
- Was muss entschieden werden?

Der eingangs beschriebene Prozess sollte bei dieser Review-Art genauestens befolgt werden.

Informeller Review

Ein informeller Review entspricht rein inhaltlich dem technischen Review. Man führt ihn jedoch meist durch, um Zeit zu sparen und hält sich dabei auch nicht exakt an den Review-Prozess. Oft wird er vom Autor selbst initiiert und hat den Charakter eines Gegenlesens.

Eine schriftliche Rückmeldung (z.B. per E-Mail) des Korrektors oder das Rücksenden des korrigierten Prüfobjekts sind als Ergebnis akzeptabel. Informelle Reviews erfahren wegen ihres geringen Aufwands eine hohe Akzeptanz.

Walkthrough

Der Walkthrough wird meist im Kreis gleichgestellter Mitarbeiter durchgeführt. Das Ziel ist es Fehler, Unklarheiten und Probleme in Dokumenten zu finden. Hier übernimmt der Autor selbst die Präsentation, dabei gibt es keine ausgewiesene Moderatorenrolle oder Hierarchien. Walkthroughs zeichnen sich u.a. durch theoretische Probeläufe und das Durchspielen von Szenarien aus. Diese Review-Art ist ebenfalls weniger formalistisch als der beschriebene Prozess und generiert daher auch einen geringeren zeitlichen Aufwand.

Inspection

Von allen Review-Arten ist die Inspection die formalste Variante. Die genaue Vorgehensweise wird in den Normen IEEE 610 und IEEE 1028 detailliert beschrieben. Der Review-Prozess wird dabei mit zusätzlichen Details befolgt. Entsprechend genau und detailreich wird in einer Inspection geprüft. Auch hierbei geht es um die Aufdeckung von Mängeln und Fehlern in Dokumenten. Zusätzlich wird das Prüfobjekt auf Verstöße von Standards sowie auf die Nicht-Konformität gegenüber der Spezifikationen untersucht.

Zu Beginn eines Review muss somit abgewogen werden, welchem Zweck er dienen soll und welche Review-Art somit für das Projekt optimal ist. Dies obliegt der Entscheidung des Projektmanagers bzw. des Managements. Grundsätzlich können kleine Reviews in Form eines Gegenlesens durch eine andere Person niemals schaden und verbessern grundsätzlich die Qualität eines Dokuments.

Alle betrachteten Review-Arten haben trotz ihrer Unterschiede Eines gemeinsam: Sie betrachten Produkte oder Teilprodukte (Dokumente), die während des Entwicklungsprozesses erstellt werden. Der Projektverlauf oder der Entwicklungsprozess als solcher wird dabei nicht betrachtet.

Reviews die sich mit dem Projektverlauf beschäftigen, werden Managementreviews (IEEE 1028) oder auch Projektreviews genannt. Das Ziel dieser Art von Review ist die Analyse des Entwicklungsprozesses an sich. Hierbei werden Fragen gestellt wie:

- Wurden die Vorgaben eingehalten?
- Sind die Arbeitsaufgaben umgesetzt worden?
- Wie effektiv waren die Verbesserungen/ Veränderungen?

Des Weiteren wird das Projekt hinsichtlich der Wirtschaftlichkeit und zeitlicher Aspekte betrachtet. Auch das Management des Projektes selbst wird dabei kritisch begutachtet. Managementreviews können weitaus zeitintensiver sein als die anderen vorgestellten Review-Arten.

Insbesondere die kritische Auseinandersetzung mit dem Managementprozess, kann zu einer nachhaltigen Verbesserung im aktuellen aber auch bei zukünftigen Projekten führen.

Code-Analysen im White-Box-Test

Kommen wir nun zum Bereich der Code-Analyse. Wie bereits im ersten Teil dieser Artikelreihe erwähnt, gehört die (Quell-)Code-Analyse zu den White-Box-Tests. „White“ ist hier als Synonym für „sichtbar“ zu verstehen - soll heißen der Quellcode selbst ist das zu analysierende Testobjekt und nicht etwa das Verhalten des laufenden Programms. Die gebräuchlichsten, in der Regel werkzeugunterstützten Testverfahren der Code-Analyse, sind folgende:

- Anweisungsüberdeckung
- Zweigüberdeckung
- Test der Bedingungen
- Pfadüberdeckung

Jedes dieser Testverfahren hat eine spezielle Aufgabe, die den Quellcode hinsichtlich verschiedener Kriterien qualitativ beurteilen soll. Man spricht bei diesen Testverfahren auch von Kontrollflussverfahren. Ein typisches Kontrollflussdiagramm ist in Abbildung 1 dargestellt.

Anweisungsüberdeckung

Bei der Anweisungsüberdeckung stehen die einzelnen Anweisungen im Fokus der Untersuchung. Das Ziel ist es, eine möglichst hohe Anzahl an Anweisungen zu überprüfen. Dabei wird in einem ersten Schritt versucht, die zu überprüfenden Anweisungen in einen Kontrollflussgraphen (siehe Abbildung 1) zu transferieren. Spätestens hier wird klar, dass dies sinnvollerweise nur mit Programmstücken möglich sein wird, da sich komplexe Gesamtprogramme kaum vernünftig in einem solchen Graphen abbilden lassen. Beim Ausführen der Tests ist nun darauf zu achten, dass zuvor festgelegte Anweisungen (Knoten) durchlaufen werden und somit der festgelegte Überdeckungsgrad erreicht wird. Ist dies der Fall, war der Test erfolgreich.

Zweigüberdeckung

Etwas spezifischer wird es bei den Tests hinsichtlich der Zweigüberdeckung. Hier stehen nicht die Knoten im Fokus der Aufmerksamkeit, sondern die Kanten des Kontrollgraphen. Genauer gesagt werden hier die Bedingungen überprüft (**`IF`**, **`THEN`**, **`ELSE`**, **`SWITCH`** etc.). Bei Tests auf Zweigüberdeckung wird verlangt, dass immer alle Bedingungen inklusive Rücksprünge und Schleifenanfänge (nicht aber jeder Schleifendurchlauf) berücksichtigt werden. Dies schließt sogar leere **`ELSE`**-Zweige mit ein. Um eine 100 prozentige Testabdeckung zu gewährleisten muss jede Kante demnach mindestens einmal durchlaufen worden sein.

Test der Bedingungen

Eine weitere und deutlich komplexere Testmethode ist der Test der Bedingungen. Anders als bei der Zweigüberdeckung wird jedoch nicht immer nur eine Bedingung isoliert betrachtet, sondern eine voneinander abhängige Hierarchie von Bedingungen. So kann eine Bedingung von mehreren logisch voneinander abhängigen (atomaren) Teilbedingungen abhängig sein.

Eine solche Bedingung kann von der Zweigabdeckung nicht oder nur sehr schwer berücksichtigt werden. Beim Test der Bedingungen können mehrere Strategien angewandt werden, die hier nur kurz angerissen werden sollen:

- Einfache Bedingungsüberdeckung:
  Hier gilt die Forderung, dass jede atomare Teilbedingung einer Entscheidung einmal mit **`true`** und einmal mit **`false`** getestet werden muss.
- Mehrfachbedingungsüberdeckung:
  Hier gilt die Forderung, dass alle Kombinationen der atomaren Teilbedingungen berücksichtigt werden müssen. Das Problem dieser Art der Überprüfung ist zum einen der Umfang aber auch die Schwierigkeit, dass nicht immer alle Kombinationen durch Testdaten realisierbar sind.
- Minimale Mehrfachbedingungsüberdeckung:
  Die Forderung, alle Kombinationen zu berücksichtigen wird hierbei aufgeweicht und dahingehend reduziert, dass nur solche Kombinationen geprüft werden sollen, die zu einer Änderung des Wahrheitswertes führen. Beispielbedingung: **`if (a && b)`**. Wenn **`a = false`** ist, spielt der Wert **`b`** keine Rolle mehr. Somit entfällt der Test von **`b = true`** oder **`b = false`**.

Pfadüberdeckung

Die bisher vorgestellten Kontrollflussverfahren berücksichtigen Schleifen und Wiederholungen nur unzureichend.

Glossar

Inspection
Eine Inspection ist die formalste Review-Technik mit einem dokumentierten Vorgehen nach IEEE 610 und IEEE 1028. Zweck: Sichtüberprüfung von Dokumenten, um Mängel zu finden (z.B. Nichteinhaltung von Entwicklungsstandards, Nicht-Konformität gegenüber Spezifikationen, usw.).

Norm IEEE 610
Der Standard IEEE 610 bzw. 610.12 ist eine Neuauflage bzw. ein Redesign von IEEE 729. Er definiert und beschreibt über 1000 Begriffe aus dem Software Engineering. Ein großer Teil der Begriffe wird darin erstmalig eingeführt.

Norm IEEE 1028
Der Standard IEEE 1028 ist ein Standard für Software Reviews und Audits. Er definiert 5 Typen von systematischen oder formalen Software Reviews zusammen mit Prozeduren zur Durchführung. Die Typen sind Management Review, Technisches Review, Inspection, Walkthroughs und Audits.

Links

- [1] ORDIX® news Artikel 1/2014
  „Qualitätssicherung in Softwareprojekten (Teil I) -
  QS - Was? Wann? Wer?“:
  http://www.ordix.de/images/ordix/onews_archiv/1_2014/ORDIX_news_1_2014_opf_files/WebSearch/page0044.html
- [2] ISTQB International Software Testing Qualification Board:
  http://www.istqb.org/
- [3] Wikipedia-Definition „Statisches Software-Testverfahren“:
  http://de.wikipedia.org/wiki/Statisches_Software-Testverfahren

Quellen

- [1] Abbildung 1: Spillner, Linz (2004):
  Basiswissen Softwaretest, 2. überarbeitete Auflage,
  dpunkt.verlag, ISBN 3-89864-358-1
- [2] Liggesmeyer, Peter (2002):
  Softwarequalität: Testen, Analysieren und Verifizieren von Software

Marcus Meisenberg
(info@objectsystems.de)

Diese können mit der Pfadüberdeckung getestet werden. Dieses Verfahren fordert die Ausführung aller unterschiedlichen Pfade durch ein Testobjekt. Ein Pfad wird dabei als Abfolge von Programmteilen in einem Programmstück definiert. Die Pfade berücksichtigen zusätzlich die Abhängigkeiten von Zweigen und Schleifen (sämtliche Durchläufe) untereinander. Aufgrund der Betrachtung von Bedingungen und ggf. auch von Schleifen ist die Pfadüberdeckung die komplexeste Art der Code-Analyse und damit auch die aufwendigste.

Fazit

Erneut kann dieser Artikel lediglich Ansporn sein, sich detaillierter mit dem Thema Qualtiätssicherung zu beschäftigen. Insbesondere der Bereich der Code-Analyse ist ein sehr komplexes Thema, in welches man mit mathemathischen Formeln, Metriken und anderen Berechnungsmodellen beliebig tief eintauchen kann. Wer sich damit eingehender beschäftigen möchte, dem sei das Buch von Peter Liggesmeyer [2] zu empfehlen, welches auch als Basis für diesen Artikel diente.

Durch unser umfangreiches Dienstleistungangebot auf dem Gebiet der Qualitätssicherung verfügen wir über qualifizierte Experten, die Sie gern diesbezüglich beraten, unterstützen und Ihre Fragen beantworten.

DBMS_METADATA und DBMS_METADATA_DIFF im Projekteinsatz

Abgleich von Objektdefinitionen

Der Unterschied zwischen der richtigen Objektdefinition und der beinahe richtigen ist derselbe Unterschied, wie zwischen einem Blitz und einem Glühwürmchen (frei nach Mark Twain). Diese Unterschiede aufzudecken und zu beseitigen ist oft komplex und zeitintensiv. Sämtliche Objekte - Tabellen, Indizes, Views und Constraints eines Schemas - müssen dabei detailliert verglichen werden. Seit Oracle 11g Release 2 ist es möglich, mit dem Paket DBMS_METADATA_DIFF DDL-Statements zu generieren, um ein Objekt einem anderen anzugleichen.

SXML: Ein SQL in XML-Form

Bereits seit Oracle 9i ist das Paket DBMS_METADATA verfügbar. Dieses ermöglicht es, die gesamten Eigenschaften eines Objektes entweder als DDL-Statement oder als XML aus dem Data Dictionary auszulesen. Ein mit DBMS_METADATA generiertes XML-Abbild beinhaltet die gesamte strukturelle und logische Komplexität des Data Dictionary. Zusätzlich beinhaltet das mit DBMS_METADATA generierte XML spezifische Werte des Data Dictionary der Instanz.

Mit Oracle 11g Release 2 wurde das Paket erweitert. Zusätzlich ist es ab dieser Version möglich, neben SQL und XML auch die Repräsentation eines Objektes im SXML-Format zu generieren.

Ein SXML ist eine „human-readable"-Version des bereits länger verfügbaren XML. Es enthält keinerlei instanzspezifische Werte mehr, da es im Gegensatz zum XML kein Abbild des Data Dictionary, sondern ein Abbild des DDL-Statements eines Objektes in der XML-Struktur ist. Die XML-Tags entsprechen den gewohnten Bezeichnungen in der Oracle-Datenbank wie z.B. Schema, Column oder Datatype (siehe Abbildung 1).

Ein SXML kann daher auch durch Fremdsoftware generiert oder angepasst werden. Zudem ist es einfach möglich, zwei verschiede SXML-Objekte zu vergleichen und die Unterschiede zu ermitteln.

DBMS_METADATA_DIFF

Der Vergleich von zwei SXML-Objekten kann mit dem ebenfalls mit Oracle 11g Release 2 eingeführten Paket DBMS_METADTA_DIFF vorgenommen werden. Das Ergebnis dieses Vergleichs ist ein um die Unterschiede angereichertes SXML - ein sogenanntes Compare SXML (siehe Abbildung 2).

In diesem Beispiel ist in der ersten Tabelle die Spalte A vorhanden, in der zweiten nicht. Die Spalte B hat zwar den gleichen Datentyp, unterscheidet sich jedoch in der Länge. Eine Umwandlung des SXML in ein DML liefert folgendes Ergebnis:

```
ALTER TABLE ORDIX.ORDIX1 DROP (A);
ALTER TABLE ORDIX.ORDIX1 MODIFY (B CHAR(20));
ALTER TABLE ORDIX.ORDIX1 RENAME TO ORDIX2;
```

Eine Generierung dieser ALTER-DDL-Statements mit DBMS_METADATA_DIFF ist genauso einfach wie riskant. Das erste Statement in diesem Beispiel löscht eine Spalte inklusive der dort enthaltenen Daten. Das dritte Statement versucht die Tabelle **ORDIX1** in **ORDIX2** umzubenennen. Dies führt, da sich beide Tabellen im gleichen Schema befinden, unweigerlich zu einem Fehler.

Es ist daher sehr wichtig, dass die generierten Statements vor ihrer Ausführung sorgfältig überprüft werden. Auf verlorene Daten oder Performance-Aspekte nimmt DBMS_METADATA_DIFF dabei keine Rücksicht.

Customizing

Dem Paket DBMS_METADATA können gewisse Regeln für die Generierung mitgegeben werden. Diese ermöglichen es, sowohl das CREATE-DDL-Statement eines Objektes als auch ein ALTER-DDL-Statement individuell anzupassen. Hierzu ist es notwendig, die einzelnen Schritte der Generierung in PL/SQL nachzubilden.

Im Folgenden legen wir den Fokus auf die Anpassung eines ALTER-DDL-Statements. Die Ausführungen gelten jedoch analog für die Anpassung eines DDL-Statements zur Erzeugung eines Objektes.

Zunächst müssen aus den beiden zu vergleichenden Objekten jeweils SXML-Objekte generiert werden. Mit Hilfe

```
SELECT dbms_metadata.get_sxml('TABLE', EMP) from dual;

<TABLE xmlns="http://xmlns.oracle.com/ku" version="1.0">
  <SCHEMA>SCOTT</SCHEMA>
  <NAME>EMP</NAME>
  <RELATIONAL_TABLE>
  <COL_LIST>
    <COL_LIST_ITEM>
      <NAME>EMPNO</NAME>
      <DATATYPE>NUMBER</DATATYPE>
      <PRECISION>4</PRECISION>
      <SCALE>0</SCALE>
      <NOT_NULL/>
    </COL_LIST_ITEM>
...
```

Abb. 1: Aufbau einer SXML-Datei.

```
CREATE TABLE ordix1 (a number, b char(15));
CREATE TABLE ordix2 (b char(20));

<TABLE xmlns="http://xmlns.oracle.com/ku" version="1.0">
  <SCHEMA>ORDIX</SCHEMA>
  <NAME value1="ORDIX1">ORDIX2</NAME>
  <RELATIONAL_TABLE>
    <COL_LIST>
      <COL_LIST_ITEM src="1">
        <NAME>A</NAME>
        <DATATYPE>NUMBER</DATATYPE>
      </COL_LIST_ITEM>
      <COL_LIST_ITEM>
        <NAME>B</NAME>
        <DATATYPE>VARCHAR2</DATATYPE>
        <LENGTH value1="10">20</LENGTH>
      </COL_LIST_ITEM>
    </COL_LIST>
  </RELATIONAL_TABLE>
</TABLE>
```

Abb. 2: Aufbau einer Compare SXML-Datei.

Abb. 3: Der Weg vom Vergleich zum ALTER DDL.

```
DBMS_METADATA.SET_PARSE_ITEM(openw_handle, 'TABLESPACE');
DBMS_METADATA.SET_REMAP_PARAM(openw_handle, 'ORDIX',
'&schema');
DBMS_METADATA.SET_TRANSFORM_PARAM(openw_handle, 'STORAGE',
'FALSE');
```

Abb. 4: Individualisieren von SXML-Statements.

der Funktion **DBMS_METADATA_DIFF.COMPARE_SXML** wird nun aus diesen ein Compare SXML generiert. Anschließend wird das Compare SXML zunächst in ein ALTER SXML und schließlich in ein oder mehrere ALTER-DDL-Statements umgewandelt (siehe Abbildung 3).

Über die folgenden Prozeduren (siehe Abbildung 4) ist es möglich ein SXML individuell anzupassen.

Mit Hilfe der Prozedur **SET_PARSE_ITEM** ist es möglich, dass nicht per Default generierte Objekteigenschaften ebenfalls erzeugt werden. Dies sind u.a. abhängige Objekte wie Indizes, Trigger oder der Tablespace des Objektes.

Durch die Prozedur **SET_REMAP_PARAM** ist es möglich, die Bezeichner einiger Objekteigenschaften, wie z.B. den Tablespace- oder Schemanamen, individuell anzupassen.

Weitere Eigenschaften der generierten DDL-Statements können über die Prozedur **SET_TRANSFORM_PARAM** bestimmt werden. Hier kann u.a. die STORAGE CLAUSE, die Generierung nicht referentieller Constraints oder auch eines Package Body abgeschaltet werden.

Das hier gezeigte Vorgehen kann helfen, einige Generierungsprobleme der ALTER-DDL-Statements zu verhindern. Bestimmte Problemkonstellationen (wie das Umbenennen einer Spalte) führen jedoch unweigerlich zu falschen Ergebnissen.

Wird eine Spalte umbenannt, kann DBMS_METDATA_DIFF dies nicht erkennen und generiert ein ALTER TABLE DROP COLUMN- und ein ALTER TABLE ADD COLUMN-Statement. Diese Veränderung gleicht zwar die beiden Tabellen einander an, die Daten der umbenannten Spalte gehen jedoch verloren.

Vergleich über die Zeit

In einem unserer Kundenprojekte verwenden wir das Paket DBMS_METADATA_DIFF um die Veränderungen eines Schemas über die Laufzeit des Projektes nachzuverfolgen. Hierzu wurde in einem Master-Schema ein DDL-Trigger geschrieben. Dieser Trigger speichert vor jeder Veränderung eines Objektes dessen aktuellen Zustand in Form eines SXML ab.

Auf diese Weise wird ein ALTER DDL eines Objektes generiert, das es ermöglicht ein Objekt aus der Softwareversion A auf die Softwareversion B zu migrieren. Falls gewünscht, können auch passende Rollback-Statements erzeugt werden.

Bevor ein solches Statement für die Migration freigegeben wird, muss es von einem Entwickler überprüft werden. Führen die generierten Statements zu unerwünschtem Verhalten (schlechte Performance oder Datenverlust), hat der Entwickler die Möglichkeit einer manuellen Anpassung.

Fazit

Die Erweiterung des Paketes DBMS_METADATA um SXML bietet vielfältige neue Möglichkeiten. Das bereits früher vorhandene XML beinhaltet zu viele Interna des Data Dictionary und bietet lediglich die Möglichkeit der exakten Übertragung einer Objektdefinition in ein anderes Schema bzw. eine andere Oracle-Datenbank.

SXML ist hingegen vollständig dokumentiert und enthält nur die Informationen, welche auch in einem DML-Statement zu finden sind. Es könnte daher, neben den hier vorgestellten Verwendungsmöglichkeiten z.B. auch dazu verwendet werden, ein Oracle-Datenbankobjekt in die Datenbank eines anderen Herstellers zu überführen. Hierzu ist es lediglich erforderlich, einen Algorithmus zu entwickeln, der dieses SXML in die DDL-Variante eines beliebigen anderen Datenbankherstellers umwandelt.

Philipp Loer
(info@ordix.de)

Glossar

DDL
Die Data Definition Language ist eine Datenbanksprache, um Datenstrukturen und verwandte Elemente zu beschreiben, zu ändern oder zu entfernen.

DML
Die Data Manipulation Language ist der Teil der Datenbanksprache um Daten zu schreiben, ändern oder zu löschen.

SQL
Die Structured Query Language ist eine Datenbanksprache für relationale Datenbanken.

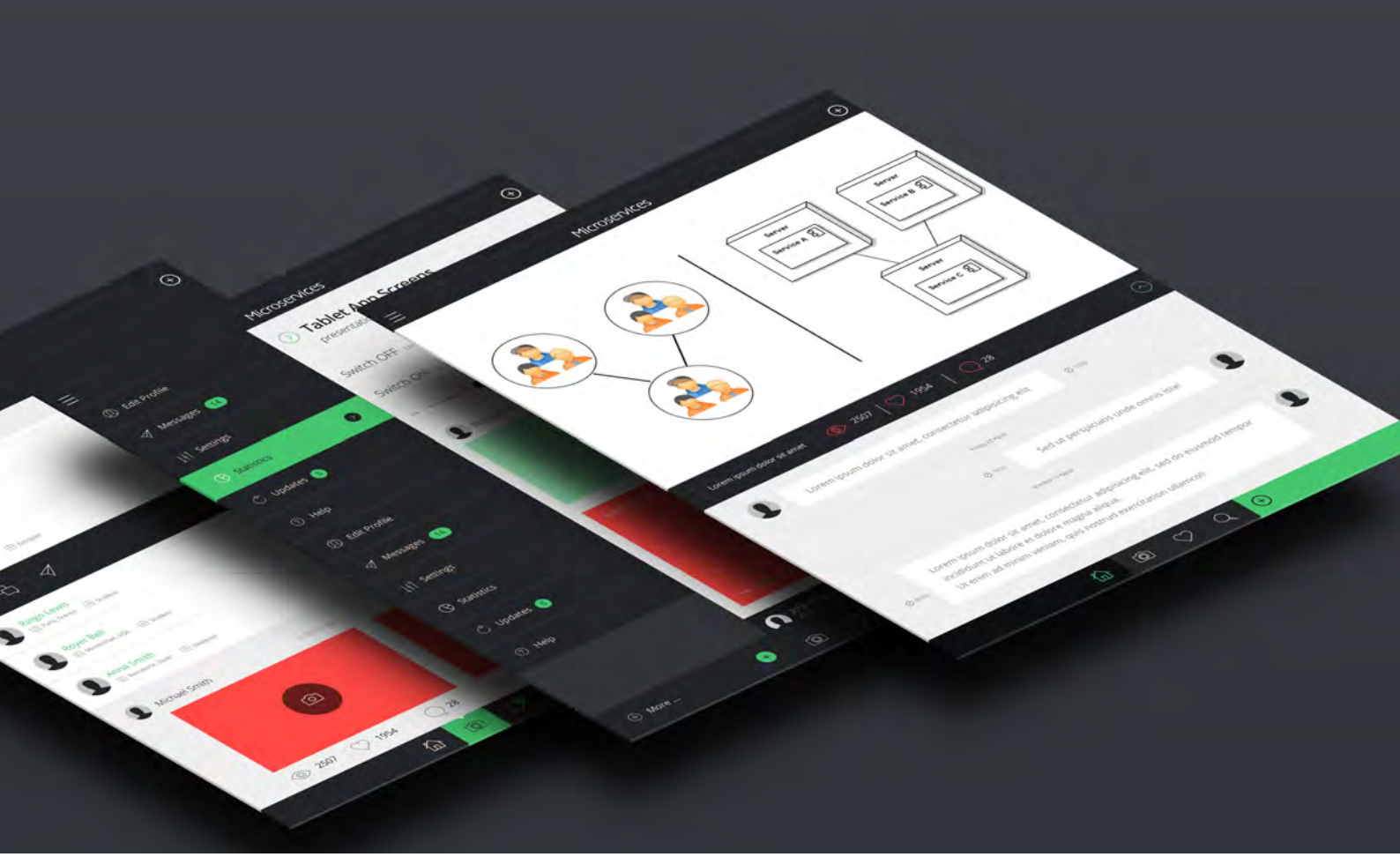
SXML: (SQL XML):
Die SQL Extensible Markup Language ist eine Spezialform des XML-Standards zur Speicherung von SQL-Befehlen.

Tag (engl. Etikett, Auszeichner)
Ein Tag ist ein in spitzen Klammern eingeschlossenes Kürzel, das in XML dazu dient Daten zu klassifizieren und strukturieren.

XML
Die Extensible Markup Language ist eine Sprache zur Darstellung hierarchisch organisierter Daten in Textdateien.

Links

- [1] Oracle-Dokumentation: http://docs.oracle.com/cd/E16655_01/appdev.121/e17602/d_metada.htm#ARPLS026
- [2] Oracle White Paper: http://download.oracle.com/otndocs/products/database/enterprise_edition/utilities/pdf/metadiffer11gr2_twp_1106.pdf



Moderne serviceorientierte Softwarearchitekturen

Microservices

Die letzten Jahre haben tiefgreifende Änderungen in den IT-Landschaften mit sich gebracht. So stellen beispielsweise der Boom mobiler Geräte und das stark veränderte Nutzerverhalten große Herausforderungen für heutige Softwaresysteme dar. Die Frage, wie in immer kürzeren Zyklen Software bei gleichbleibend hoher Qualität an die neuen Anforderungen angepasst werden kann, ist dabei von zentraler Bedeutung. Themen wie Agile, DevOps oder Continous-Delivery versuchen die Probleme zu adressieren und zeigen vielversprechende Ansätze auf.

Auch seitens der Softwarearchitektur scheint sich ein neuer Trend herauszukristallisieren, weg von einzelnen großen monolithischen Systemen, hin zu kleineren, teils autonomen Systemen. Mit den Microservices wird seit einiger Zeit ein Architekturmuster diskutiert, das diesen Ansatz verfolgt. Dieser Artikel beleuchtet die Idee der Microservices näher und zeigt auf, welche Vor- und Nachteile sie mit sich bringen.

Grundidee

Die Idee der Microservices wurde 2011 erstmals im Rahmen eines Workshop für Softwarearchitekturen nahe Venedig diskutiert. Zu den Beteiligten gehörten u.a. James Lewis und Martin Fowler von der Firma ThoughtWorks, welche die Diskussion um das Thema auch heute maßgeblich antreiben [1].

Ausgehend von der Annahme, dass die Ideen der Agile- und DevOps-Bewegung gut und richtig sind und Konzepte wie das Continous-Delivery wichtige und wünschenswerte Aspekte in der Softwareentwicklung darstellen, stellt sich die Frage, wie eine Softwarearchitektur aussehen soll, die dies bestmöglich unterstützen kann.

Für das Architekturmuster der Microservices gibt es keine festgelegte Definition, aber es existieren einige charakteristische Eigenschaften, die Microservices erfüllen sollten. Darunter befinden sich einige Konzepte, die schon seit jeher zu den bewährten Grundprinzipien guter Softwarearchitekturen zählen.

Modularisierung

Dem Prinzip „Teile und Herrsche" folgend wird ein Gesamtsystem in modulare Einheiten heruntergebrochen. Die dabei entstehenden kleineren Komponenten werden jeweils durch Services repräsentiert. Diese Services sollen im Wesentlichen der ursprünglichen Definition einer Softwarekomponente genügen, d.h. sie müssen unabhängig vom Rest des Gesamtsystems austauschbar sein.

Dies gelingt nur dann effizient, wenn die Komponenten möglichst lose gekoppelt sind und eine hohe Kohäsion aufweisen. Dabei muss ein besonderes Augenmerk auf die Ausgestaltung der Schnittstellen gelegt werden, da Änderungen hier besonders große Auswirkungen haben und später häufig nur mit erheblichen Kosten und Mühen durchgeführt werden können.

Des Weiteren betrachtet man Services als völlig autonome Prozesse (siehe Abbildung 1 und 2), die evtl. auch in einer eigenen Ablaufumgebung ausgeführt werden. Damit sind sie unabhängig vom Gesamtsystem und können einfacher ausgetauscht und betrieben werden. Dies bringt enorme Vorteile für die Skalierbarkeit solcher Systeme, hat aber auch den Nachteil, dass Aufrufe über Prozessgrenzen hinweg durchgeführt werden müssen, was tendenziell „teurer" ist als klassische Prozeduraufrufe. Die Ausgestaltung der Schnittstellen muss somit etwas grobgranularer erfolgen, da sich sonst ein überproportional hoher Kommunikationsaufwand ergeben würde. Die technischen Ausgestaltungsmöglichkeiten werden später noch genauer erläutert.

Separation Of Concerns (Trennung der Zuständigkeiten)

Bei der Zerlegung eines Systems in Teilkomponenten, stellt sich immer die Frage nach deren Zuschnitt. Gemeint sind damit die Zuständigkeiten und Grenzen der einzelnen Komponenten. Hier gibt es bzgl. der Microservices eine klare Aussage: Services sollen sich strikt an der Geschäftslogik orientieren. Konkret soll ein Service als eine Art Produkt verstanden werden. Diesem Gedanken folgend, soll es dann ein Team geben, das dieses Produkt vom Design bis zum Betrieb betreut.

Diese Betrachtungsweise erleichtert die agile Entwicklung und die Aufteilung des Gesamtsystems auf einzelne Teams, außerdem entspricht es stark dem DevOps-Gedanken (siehe Abbildung 3). Das Team ist verantwortlich für die Entwicklung und den Betrieb eines Services (Produkt). Es muss entsprechend über alle benötigten Skills verfügen. Als Nebeneffekt wird die Aufteilung des Teilsystems in rein technische Domänen verhindert, wie sie in klassischen Projekten gemäß Conway's Law oft entsteht (siehe Abbildung 4).

Logik in Komponenten

Als weiteren wichtigen Punkt soll sich bei Microservices die Fachlogik ausschließlich in den (Service-)Komponenten befinden. Das hört sich zunächst selbstverständlich an. Deutlicher wird der Focus, wenn man es aus einer anderen Perspektive betrachtet: Es soll keine Intelligenz in Infrastrukturkomponenten vorhanden sein.

Explizit soll verhindert werden, dass sich Code für die Datenkonvertierung, das Routing oder andere Funktiona-

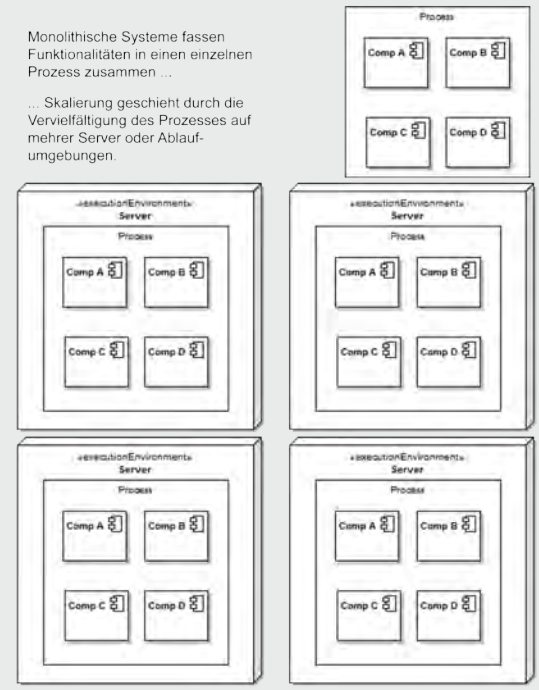


Abb. 1: Monolithische Systeme.

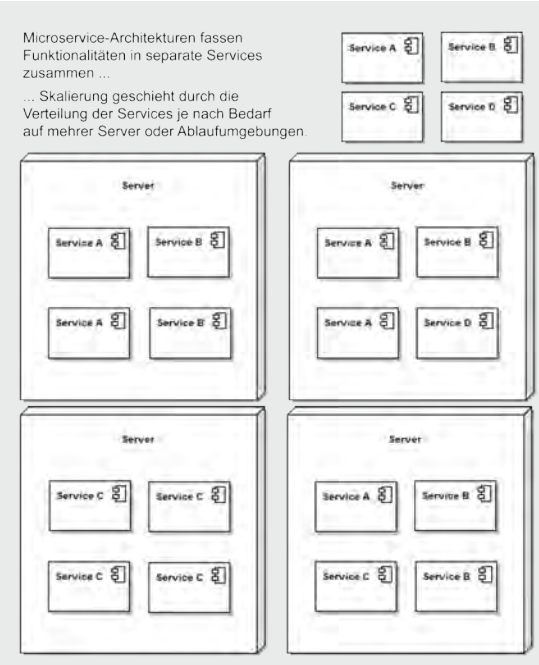


Abb. 2: Microservice-Architekturen.

litäten in der Kommunikationsinfrastruktur verstecken kann. Der Einsatz schwergewichtiger Komponenten wie z.B. eines Enterprise Service Bus ist also nicht erwünscht.

Kommunikation

Damit stellt sich sofort die Frage nach den Kommunikationsmechanismen, die für die Nutzung der Services verwendet werden sollen. Die Forderung nach einer losen Koppelung mit der Konsequenz, dass sich daraus Services als autonome Prozesse darstellen, wurde bereits erläutert. Da wir im aktuellen Umfeld natürlich über verteilte Anwendungen in einem Netzwerk sprechen, liegt der Schluss nahe, auch die dafür passenden Mechanismen und Protokolle zu verwenden.

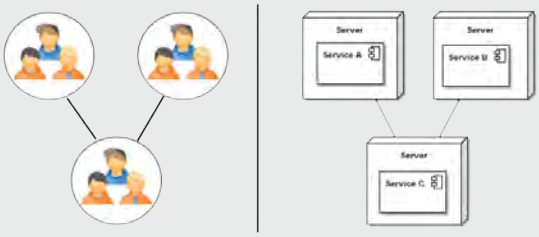


Abb. 3: Teams betreuen „Produkte“ (Services).

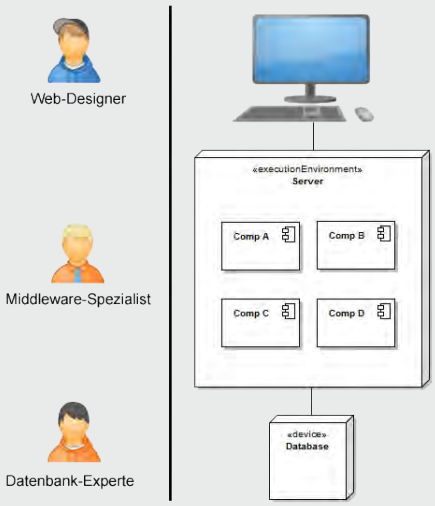


Abb. 4: Conway’s Law bei monolithischen Systemen.

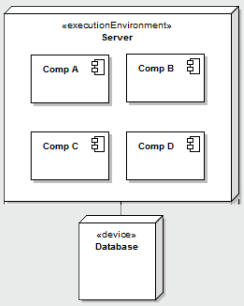


Abb. 5: Datenhaltung bei Monolithen.

Der Vorschlag lautet daher, in Microservice-Architekturen möglichst zwei Ansätze zu verwenden: RESTful Request-Response-Kommunikation über HTTP oder leichtgewichtiges Messaging. Der erste Punkt ist wegen der HTTP-inhärenten Möglichkeiten wie z.B. Caching naheliegend. Der RESTful-Ansatz ist längst etabliert und das Web ist der beste Beweis für dessen Tauglichkeit.

Leichtgewichtiges Messaging meint hier den Nachrichtenaustausch über einfach gehaltene Infrastrukturkomponenten, die außer dem Nachrichtentransport keinerlei zusätzliche Aufgaben übernehmen. Das sind klassischerweise Queues oder ähnliche Messaging-Komponenten.

Spätestens mit dem zweiten Punkt sollte klar geworden sein, dass ein Großteil der Kommunikation in Microservice-Architekturen asynchron abläuft. Die damit verbundenen Entwicklungsaufwände sind höher als in rein synchron kommunizierenden Systemen, was der erhöhten Komplexität in der Koordinierung der Service-Aufrufe geschuldet ist.

Ein weiterer sehr wichtiger Punkt, den es in diesem Zusammenhang zu berücksichtigen gilt, ist die Fehlertoleranz. Insbesondere die Frage nach der Nichterreichbarkeit eines Service steht hier im Mittelpunkt. Grundsätzlich gilt, dass jeder Service auftretende Fehler im Sinne der Gesamtanwendung behandeln muss. Eine entsprechend Fail-Safe-Strategie ist hier unerlässlich. Das verantwortliche Team muss sich der Wirkung eines Ausfalls für das Gesamtsystem bewusst sein und entsprechende Vorkehrungen treffen sowie die geeigneten Testszenarien für die Qualitätssicherung entwickeln.

Wahl der Mittel

„Wenn du als einziges Werkzeug einen Hammer besitzt, dann sieht jedes Problem wie ein Nagel aus.“ Diese oft zitierte Metapher adressieren Microservices-Architekturen sehr konkret. Die Services sollen ausdrücklich die am besten geeigneten Mittel zu Lösung des vorliegenden Problems verwenden. In letzter Konsequenz bedeutet dies eine völlige Freiheit bei der Wahl der Programmiersprache, Ablaufumgebung, Standards etc.

Es gilt jedoch auch, dass das was getan werden kann, nicht getan werden muss. Das entscheidende Kriterium ist und bleibt die Fachlichkeit. Sollte die Problemdomäne eine effiziente Lösung auf Basis eines vom Rest des Gesamtsystems abweichenden Ansatzes ermöglichen, so darf dieser auch gewählt werden. In klassischen Architekturen ist dies meist so nicht vorgesehen, da oft übergeordnete (koordinierende) Plattformen zum Einsatz kommen, die technische Einschränkungen zur Folge haben und solche Optionen nicht ermöglichen.

Verteilung der Datenhaltung

Durch die freie Wahl der Mittel wird, wie oben beschrieben, die Koordination des Gesamtsystems quasi dezentralisiert. Jeder Service trägt in Eigenregie mit den eigenen Mitteln und Methoden zu einem Teil der Lösung des Gesamtproblems bei.

Konsequenterweise erlaubt eine Microservice-Architektur auch die Dezentralisierung der Datenhaltung. Das bezieht sich nicht nur auf die Technologiefrage (RDBMS, NoSQL, Files, Memory, …), sondern auch auf die logische Datenaufteilung. Einzelne Services haben es oft nur mit Teilaspekten der jeweiligen Gesamtdatenbestände zu tun, andere teilen sich gemeinsame Datenpools (siehe Abbildungen 5 und 6).

Beim Zuschnitt der Service-Komponenten dürfen deshalb die Konsequenzen für das Datenmodell nicht vernachlässigt werden. Vorgeschlagen wird hier, sich am Konzept der „Bounded Contexts“ aus Eric Evans‘ Ansatz des „Domain Driven Design“ zu orientieren. Evans beschreibt darin, wie aus dem Gesamtdatenmodell sinnvoll Teilbereiche definiert werden und dann im Gesamtkontext verwaltet werden können.

Abgrenzung zur „klassischen“ SOA

Serviceorientierte Architekturen (SOA) sind vom Prinzip nichts Neues. Leider ist das Thema mittlerweile sogar etwas in Verruf geraten, da die hohen Erwartungen oft nicht erfüllt werden konnten. Die Gründe dafür sind vielschichtig, sollen hier aber nicht weiter betrachtet werden.

Trotzdem ist das Konzept des Services als Architekturelement prinzipiell ein gutes Mittel zur Strukturierung und hat sich in verschiedenen Ausprägungen mehr als bewährt. Auch wenn es viele Parallelen gibt, sollen hier kurz zwei der wesentlichen Unterschiede einer Microservices-Architektur zur „klassischen“ SOA dargestellt werden.

Zum einen fokussieren Microservices die Modularisierung oder Zerlegung eines monolithischen Systems. In der klassischen SOA-Idee liegt der Schwerpunkt dagegen eher auf der Integration von vielen Monolithen.

Zum anderen sollen in Microservice-Architekturen die Abläufe dezentralisiert werden. Einhergehend mit einer deutlichen Reduzierung der Komplexität der Kommunikationsinfrastruktur, soll sich die Intelligenz des Systems möglichst ausschließlich in den Services befinden. Das bedeutet eine Abkehr von beispielsweise schwergewichtigen, mit zusätzlichen Funktionalitäten überfrachteten ESBs oder SOAP-basierten Webservices.

Trade-Offs

Die Softwarearchitektur muss immer an den gegebenen Anforderungen und Randbedingungen gemessen werden. Besonders die nicht-funktionalen Anforderungen entpuppen sich meistens als die wesentlichen Architekturtreiber. Eine Architektur wird in der Regel so ausgestaltet, dass sie die Anforderungen gemäß ihrer Prioritäten erfüllt. Wichtig ist es aber zu verstehen, dass eine Architektur nicht für beliebig viele Anforderungen optimiert werden kann, da es bzgl. einiger Qualitätsmerkmale regelrechte Zielkonflikte gibt (z.B. Benutzerfreundlichkeit und Sicherheit).

Wie zu Beginn beschrieben, haben Microservice-Architekturen eine Ausrichtung auf spezielle Anwendungstypen, für die bestimmte nicht-funktionale Anforderungen wie z.B. die Skalierbarkeit und Erweiterbarkeit im Vordergrund stehen. Zwangsläufig müssen deshalb in verschiedenen Bereichen Kompromisse eingegangen werden. Einige dieser Trade-Offs sollen im Folgenden kurz skizziert werden:.

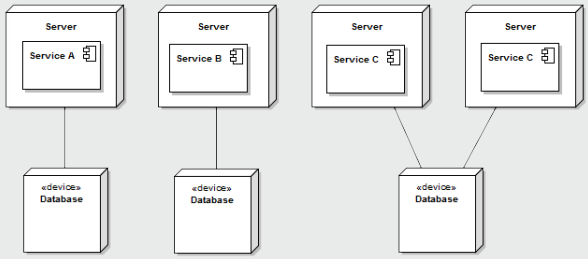


Abb. 6: Datenhaltung in Microservice-Architekturen.

Glossar

Conway’s Law
Das Gesetz von Conway ist eine nach dem US-amerikanischen Informatiker Melvin Edward Conway benannte Beobachtung, dass die Strukturen von Systemen durch die Kommunikationsstrukturen der sie umsetzenden Organisationen vorbestimmt sind. (Quelle: Wikipedia)

DevOps und Continous Delivery
DevOps ist ein Kunstwort aus „Development“ (Softwareentwicklung) und „Operations“ (IT-Betrieb). Es bezeichnet den gewünschten Schulterschluss beider Bereiche, um den traditionellen Konflikt zwischen ihnen aufzulösen. Ziel soll eine effizientere und schnellere Entwicklung von Systemen sein. Einbezogen wird ausdrücklich eine vereinfachte und kontinuierliche Auslieferung (Continous Delivery) sowie der Betrieb unter qualitätssichernden Aspekten.

ESB
Der Enterprise Bus ist eine schwergewichtige Infrastrukturkomponente, die Dienste zu Aspekten wie Routing, Security, Datenkonvertierung, u.a. zur Verfügung stellt. Sie ist ein zentraler Bestandteil einer klassischen SOA.

Kohäsion
In der objektorientierten Programmierung beschreibt Kohäsion, wie gut eine Programmeinheit eine logische Aufgabe oder Einheit abbildet. Jede einzelne Komponente ist für eine einzelne, genau definierte (abgegrenzte) Aufgabe zuständig.

SOA
Die serviceorientierte Architektur ist ein Architekturmuster der Informationstechnik aus dem Bereich der verteilten Systeme, um Dienste von IT-Systemen zu strukturieren und zu nutzen.

SOAP
Das Simple Object Access Protocol ist ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und Remote Procedure Calls durchgeführt werden können.

Schnittstellendesign

Den Schnittstellen kommt in lose gekoppelten, verteilten Systemen eine entscheidende Bedeutung zu. Da die Nutzung der Schnittstellen wie bereits erwähnt meist über Prozessgrenzen hinweg stattfindet (z.B. RESTful Webservices oder Messaging-Mechanismen) muss die Wahl der Granularität wohl überlegt sein. Eine zu feine Granularität führt schnell zu einer übermäßigen Schnittstellennutzung und damit zu enormen Kommunikationsaufwänden.

Je loser Systeme gekoppelt sind, umso empfindlicher werden deren Schnittstellen gegenüber Änderungen. Als Folge müssen solche Systeme sehr viel sorgfältiger geplant werden als Monolithen. Das führt ganz allgemein zu erhöhten Aufwänden und erfordert ein zusätzliches Risikomanagement.

Asynchronität

Ein weiterer Punkt, der ebenfalls mit der Verteilung des Systems und dem Schnittstellendesign zu tun hat, ist die Art und Weise, wie die Schnittstellen genutzt werden. Ein großer Teil der Kommunikation in Microservice-Architekturen ist asynchron. Das ist prinzipiell schwierig zu koordinieren und bringt somit eine zusätzliche Komplexität in das Gesamtsystem. Ein klassisches Beispiel für einen Aspekt, der in solchen Situationen oft Probleme verursacht, ist das Transaktionsmanagement, was nahtlos zum nächsten Punkt führt.

Querschnittsaspekte

Aufgrund der oben beschriebenen freien Wahl der Mittel und der Verteilung der einzelnen Services ist es extrem schwierig, übergreifende Aspekte wie z.B. Logging, Security oder Transaktions-Handling in Microservice-Architekturen durchgängig einheitlich zu implementieren. Bei Verwendung komplett unterschiedlicher technologischer Plattformen ist es in der Regel sogar unmöglich. So kann es passieren, dass deutlich höhere Aufwände entstehen, die dem Fakt geschuldet sind, dass bestimmte Querschnittsfunktionalitäten schlicht mehrfach implementiert werden müssen. Dies steht im Widerspruch zum DRY-Prinzip („Don't repeat yourself"), das eine möglichst hohe Wiederverwendung propagiert.

Links

- [1] Fowler, Martin: Microservices: http://martinfowler.com/articles/microservices.html
- [2] Internetseite der Objectsystems GmbH: www.objectsystems.de

Bildnachweis

- © pixeden.com | Perspective Tablet Mock-Up2

Know-how-Bedarf

Die Option, Services mittels unterschiedlicher Technologien und Plattformen zu realisieren, bietet eine Wahlfreiheit, die auch ihre Schattenseiten hat. In gewisser Weise wird hier der Grundsatz der Einfachheit (KISS - „Keep It Simple Stupid") verletzt. Die Komplexität steigt naturgemäß durch z.B. zusätzliche Sprachen oder Technologien an.

Insgesamt nimmt der Know-how-Bedarf bei den beteiligten Teams deutlich zu. Das betrifft nicht nur Fragen der Implementierung, sondern in besonderem Maße auch die betrieblichen Aspekte. Integrationsthemen gewinnen ebenfalls an Bedeutung. Die Wartung und der Betrieb von einzelnen Systemen wie beispielsweise Datenbanken oder Application Server ist oft schon schwierig genug. In Microservice-Architekturen könnte aber die Notwendigkeit bestehen, gleich mehrere unterschiedliche Systeme dieser Art parallel betreuen zu müssen.

Die gleiche Argumentation gilt analog für das Testen bzw. die Qualitätssicherung solcher Systeme. Die Szenarien und Vorgehensweisen verlangen von allen Beteiligten umfangreiche Kenntnisse und Erfahrungen in diesen Bereichen.

Fazit

Der Service als Architekturelement ist in einer Softwarearchitektur ein mächtiges Strukturierungsmittel. Kombiniert mit einer Anzahl von Best Practices aus Softwareentwicklung und Vorgehensmodellen wirken Microservices ein wenig wie die Wiederentdeckung von Einfachheit und Klarheit.

Auf den zweiten Blick lassen sich aber auch Schattenseiten erkennen. Klare Strukturen in Services und Kommunikation verschleiern ein wenig die Komplexität in betrieblichen und planerischen Aspekten.

James Lewis und Martin Fowler haben zur Diskussion und zum Erfahrungsaustauch über Microservice-Architekturen aufgerufen. Da noch keine repräsentative Anzahl an Erfahrungsberichten zu entsprechenden Projekten vorliegt, tun auch sie sich mit einer Zukunftsprognose noch schwer.

Trotzdem bleiben Microservices ein interessanter Ansatz, den es in Zukunft zu beobachten gilt. Insbesondere unter Betrachtung der momentanen Entwicklung im Bereich DevOps und der weiter wachsenden Bedeutung von Cloud- und Mobile-Computing, könnten Microservices ein adäquates Mittel zur effizienten Entwicklung solcher Systeme darstellen.

Andreas Flügge
(info@ordix.de)

Innovationen in der IBM Informix-Version 12 (Teil 1)

Der Centaurus wurde freigelassen

Seit geraumer Zeit ist Informix mit seinem neuen Major-Release auf dem Markt. Am 26. März 2013 ist es unter dem Codenamen „Centaurus“ und der Versionsnummer 12.10 offiziell vorgestellt worden. IBM verspricht mit dieser neuen Version die bewährte Leistungsfähigkeit von Informix mit Verbesserungen in allen Bereichen. Woraus genau diese Verbesserungen bestehen, stellen wir Ihnen in dieser neuen Artikelreihe vor.

Was gibt's Neues?

In der Version 12.10 wurden in Informix teils größere, teils auch viele kleine Änderungen eingeführt, welche sich über alle wichtigen Teilbereiche der Datenbank erstrecken.

Die wichtigsten Erweiterungen und Änderungen gab es in den folgenden Bereichen:

- SQL und SPL
- OpenAdmin Tool (OAT)
- Komprimierung
- Tuning
- Enterprise Replication
- Informix Warehouse Accelerator (IWA)

In diesem Artikel betrachten wir zunächst die neuen Funktionen, welche uns SQL, SPL und das OpenAdmin Tool bieten.

SQL

Die größte Neuerung, die Informix 12.10 im SQL-Umfeld mit sich bringt, ist die Möglichkeit der Anwendung von Spaltenfunktionen mit Fenstern. Diese Funktion, welche schon aus einigen anderen Datenbanksytemen bekannt ist, ermöglicht es, integrierte Spaltenfunktionen wie etwa **SUM**, **MIN**, **MAX** oder **COUNT** nicht über die gesamte Ergebnismenge, sondern nur über bestimmte Bereiche anzuwenden. Die Abbildung 1 zeigt anhand eines Beispiels, wie pro Artikel einer Bestellung trotzdem der Gesamtpreis (**total_price_order**) sowie der Rang innerhalb einer Bestellung anhand des Wertes (**price_rank_in_order**) abgefragt werden kann.

Als weitere Neuerung in den Gruppierungsfunktionen wurde der zulässige Wertebereich um die Anweisung **DISTINCT** - auch in Kombination mit **CASE**-Ausdrücken – erweitert. Das Beispiel in Abbildung 2 zeigt, wie der durchschnittliche **unit_price** aus einer Tabelle **stock** abgefragt werden kann. Doppelte Werte in der Spalte **unit_price** werden bei der Berechnung des Durchschnitts jedoch außer Acht gelassen.

Auch bei der Sortierung mittels **ORDER BY** wurden in der Version 12.10 einige Neuerungen eingeführt. So ist es nun möglich, mit den neuen Schlüsselwortoptionen **NULLS FIRST** oder **NULLS LAST** die Position von Zeilen mit einem Null-Wert in der Sortierspalte innerhalb der Ergebnisreihenfolge festzulegen. Wie dies aussehen kann, zeigen die Beispiele in Abbildung 3. Zusätzlich kann der Anwender nun auch **CASE**-Ausdrücke im **ORDER BY** angeben, um dadurch die Sortierung der Rückgabe anhand von logischen Ausdrücken zu regulieren.

Sowohl als Vereinfachung der SQL-Syntax als auch zu Kompatibilitätszwecken wurden die Operatoren **MINUS** und **INTERSECT** hinzugefügt. Diese verknüpfen wie auch schon **UNION** und **UNION ALL** die Ergebnismengen zweier Abfragen, die ihre rechten und linken Operanden darstellen. So liefert der **INTERSECT**-Operator nur die

```
select order_num
        , item_num
        , sum(total_price) over(
        partition by order_num) total_price_order
        , rank() over (
        partition by order_num order by total_price)
price_rank_in_order
from items;

order_num   item_num   total_price_order   price_rank_in_ord+

     1001         1            $250.00                    1
     1002         2           $1200.00                    1
     1002         1           $1200.00                    2
     1003         1            $959.00                    1
     1003         3            $959.00                    2
     1003         2            $959.00                    3
```

Abb. 1: Spaltenfunktionen mit Fenstern.

```
SELECT AVG(DISTINCT unit_price) FROM stock WHERE stock_num = 110;
```

Abb. 2: Gruppenfunktionen nun auch mit `DISTINCT`.

```
select * from customer
order by address2 desc nulls first;
select * from customer
order by address2 asc nulls last;
```

Abb. 3: Verarbeiten von `NULL` im `ORDER BY`.

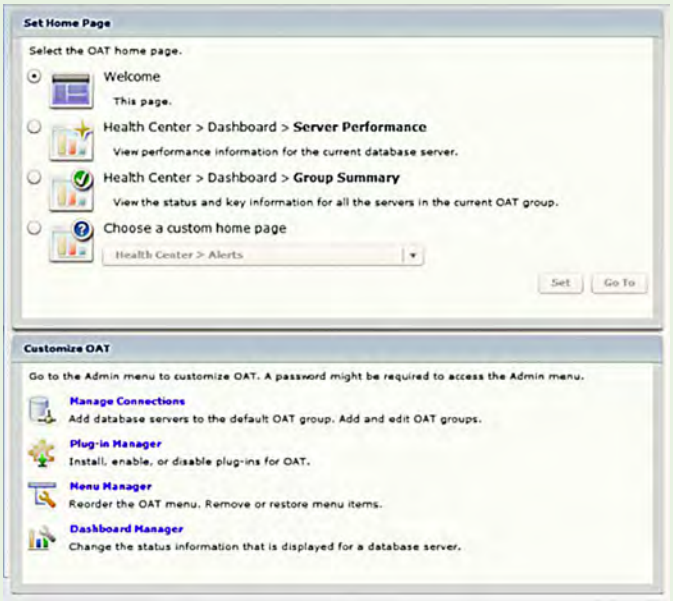


Abb. 4: Die neuen OAT-Startseiten. (Quelle [1])

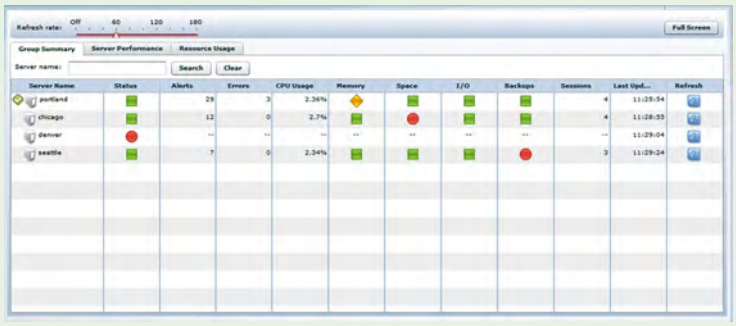


Abb. 5: Gruppenübersicht im OAT. (Quelle [1])

Zeilen zurück, welche sich sowohl in der linken als auch in der rechten Ergebnismenge der Abfrage befinden, also die Schnittmenge der beiden Abfragen. Ein **`MINUS`** verbindet die Abfragen so, dass nur die Zeilen der linken Abfragemenge zurückgeliefert werden, welche nicht in der rechten auftauchen.

Als letzter neuer SQL-Ausdruck soll hier **`CREATE TABLE ... AS SELECT FROM ...`** vorgestellt werden. Mit diesem wird eine permanente Tabelle angelegt, welche anschließend sofort mit den Rückgaben des Befehls **`SELECT`** gefüllt wird. Dies vereinfacht das aus früheren Informix-Versionen bekannte Erstellen einer leeren Tabelle mit **`CREATE TABLE`** und das nachträgliche Füllen der Tabelle mit **`INSERT INTO ... SELECT FROM ...`** ungemein.

SPL

Auch für die Freunde von Prozeduren und Funktionen hält Informix 12 einige Neuerungen bereit - auch wenn diese nicht ganz so zahlreich sind wie im SQL-Bereich.

Die erste nennenswerte Neuerung ist die Einführung der Anweisung **`CASE`** in SPL. Hierdurch ist nun eine schnellere Alternative zum üblichen **`IF`**-Befehl geboten, welcher auch eine Migration von Prozeduren und Funktionen, die für Informix Extended Parallel Server oder andere Datenbanksysteme entwickelt wurden, zu Informix vereinfachen soll.

Eine weitere Neuerung, welche auch auf die Kompatibilität von Informix zu anderen Datenbanken abzielt, stellt die SQL-Paketerweiterung dar. Einmal als DataBlade-Modul registriert stellt diese Routinen zur Verfügung, welche bereits von anderen Datenbanksystemen bekannt sind. Sie bestehen größtenteils aus Paketen für die Handhabung von großen Objekten, zur Nachrichtenverwaltung und für die Zufallszahlengenerierung (**`DBMS_ALERT, DBMS_LOB, DBMS_OUTPUT, DBMS_RANDOM, UTL_FILE`**).

OpenAdminTool - Die neue eierlegende Wollmilchsau?

Innerhalb der Datenbankverwaltung von Informix wurde vor allem ein Werkzeug in der neuen Version weiterentwickelt: das OpenAdmin Tool (OAT). Dieses Werkzeug zur Administration von Datenbanken mittels einer Web-Oberfläche war in den vorangegangenen Versionen zwar bereits verfügbar, wurde jedoch eher stiefmütterlich behandelt. Dies hat sich nun deutlich geändert.

Neben den neuen Begrüßungsseiten, welche dem Benutzer eine schnellere und einfachere Handhabung ermöglichen sollen (siehe Abbildung 4), wurde auch die Möglichkeit geschaffen, dass sich jeder Anwender mit seinem eigenen Benutzer anmelden kann und zugleich auch Berechtigungen für die Nutzer eingestellt werden können.

Eine weitere große Neuerung im OAT, welche viele Nutzer in vorherigen Versionen schmerzlich vermisst haben, ist die

Möglichkeit, mehrere Server gleichzeitig zu überwachen (siehe Abbildung 5).

Zusätzlich wurde ein mobiler Client für den OAT entwickelt. Dieser ermöglicht es den Administratoren, nun auch einzelne Datenbanken oder gesamte Datenbankgruppen mittels Smartphones und Tablets mit iOS oder Android zu überwachen.

Weiterhin zählen viele neue Administrationsfunktionen, wie beispielsweise die Verwaltung von Komprimierungsoptionen, die Möglichkeit zum Definieren automatischer Backups oder auch die Konfiguration des IWA, zum Umfang des OAT.

Fazit

Bei der neuen Datenbankversion aus dem Hause IBM kann man zunächst zwei wichtige Richtungen feststellen: Zum einen wurden im Bereich der Entwicklung viele neue Funktionen zur Verfügung gestellt, um die Kompatibilität zu anderen Datenbanksytemen zu gewährleisten und damit eine Migration zu erleichtern. Zum anderen wurde das OAT deutlich weiterentwickelt und soll zukünftig eine zentrale Rolle bei der Überwachung und Administration einnehmen. Inwieweit dies gelingt, wird sich zeigen.

Ausblick

Nachdem wir in diesem Artikel die Neuerungen in den Bereichen SQL, SPL und OpenAdmin Tool beleuchtet haben, werden wir im zweiten Teil dieser Reihe etwas tiefer unter die Haube von Informix schauen und uns mit den Neuerungen rund um die Komprimierung und das Tuning, sowie mit der Enterprise Replication und dem Informix Warehouse Accelerator beschäftigen.

Glossar

IWA
Der Informix Warehouse Accelerator ist eine auf den Einsatz im Data Warehouse angepasste Erweiterung von Informix, welche Daten in stark komprimierter Form im Hauptspeicher bereithält.

OAT
Das Open Admin Tool ist eine PHP-basierte Administrationsoberfläche für die Verwaltung von einem oder mehreren Informix Servern.

Links

- [1] Informix V12.10 Dokumentation:
  http://pic.dhe.ibm.com/infocenter/informix/v121/index.jsp

Quellen

- [1] Neuigkeiten des OAT:
  http://www.openadmintool.com/demos/oat_311_whatsNew.swf
- [2] Informix 12.10 Information Center:
  http://pic.dhe.ibm.com/infocenter/informix/v121/index.jsp?topic=%2Fcom.ibm.po.doc%2Fnew_features_ce.htm

Patrick Hecker
(info@ordix.de)

Datenbanken

Juli - Dezember 2014

DB-DB-03	Data Warehouse Grundlagen	3 Tage	1.290,00 €	14.07.2014 | 29.09.2014 | 01.12.2014
DB-ORA-01	Oracle SQL	5 Tage	1.890,00 €	04.08.2014 | 15.09.2014 | 17.11.2014
DB-ORA-01A	Oracle SQL für Experten	3 Tage	1.290,00 €	29.09.2014
DB-ORA-02	Oracle Datenbankprogrammierung mit PL/SQL Grundlagen	5 Tage	1.890,00 €	06.10.2014 | 01.12.2014
DB-ORA-34	Oracle Datenbankprogrammierung mit PL/SQL Aufbau	5 Tage	1.890,00 €	auf Anfrage
DB-ORA-42	Oracle PL/SQL Tuning	3 Tage	1.890,00 €	18.08.2014 | 27.10.2014
DB-ORA-03	Oracle Datenbankadministration Grundlagen	5 Tage	1.990,00 €	07.07.2014 | 18.08.2014 | 22.09.2014 | 24.11.2014
DB-ORA-04	Oracle Datenbankadministration Aufbau	5 Tage	1.990,00 €	28.07.2014 | 08.09.2014 | 13.10.2014 | 15.12.2014
DB-ORA-32	Oracle Backup und Recovery mit RMAN	5 Tage	1.990,00 €	04.08.2014 | 01.09.2014 | 27.10.2014 | 08.12.2014
DB-ORA-07	Oracle Tuning und Monitoring	5 Tage	1.990,00 €	01.09.2014 | 15.09.2014 | 10.11.2014
DB-ORA-41	Oracle AWR und ASH Analyse und Interpretation	3 Tage	1.290,00 €	28.07.2014 | 13.10.2014
DB-ORA-11	Oracle Troubleshooting Workshop	5 Tage	1.990,00 €	14.07.2014 | 20.10.2014
DB-ORA-08	Oracle 11gR2 RAC und Grid Infrastructure	5 Tage	2.050,00 €	21.07.2014 | 08.09.2014 | 17.11.2014
DB-ORA-33B	Oracle Security	5 Tage	1.890,00 €	01.09.2014 | 24.11.2014
DB-ORA-31	Oracle Data Guard	5 Tage	1.990,00 €	18.08.2014 | 06.10.2014 | 01.12.2014
DB-ORA-35	Oracle Cloud Control	3 Tage	1.290,00 €	21.07.2014 | 22.09.2014 | 24.11.2014
DB-ORA-40	Oracle Capacity Planning	3 Tage	1.290,00 €	11.08.2014 | 03.11.2014
DB-ORA-46	Oracle APEX Anwendungsentwicklung Grundlagen	3 Tage	1.290,00 €	11.08.2014 | 20.10.2014
DB-ORA-47	Oracle APEX Anwendungsentwicklung für Fortgeschrittene	3 Tage	1.290,00 €	25.08.2014 | 03.11.2014
DB-ORA-48	Oracle Golden Gate	3 Tage	1.290,00 €	25.08.2014 | 13.10.2014 | 15.12.2014
DB-ORA-49	Oracle 12c Neuheiten	5 Tage	1.990,00 €	15.09.2014 | 10.11.2014
DB-INF-01	IBM Informix SQL	5 Tage	1.790,00 €	04.08.2014 | 27.10.2014
DB-INF-02	IBM Informix Administration	5 Tage	1.990,00 €	18.08.2014 | 03.11.2014
DB-INF-04	IBM Informix Backup und Recovery	3 Tage	1.290,00 €	10.11.2014
DB-INF-03	IBM Informix Tuning und Monitoring	5 Tage	1.990,00 €	14.07.2014 | 01.09.2014
DB-DB2-01	IBM DB2 für Linux/Unix/Windows SQL Grundlagen	5 Tage	1.890,00 €	07.07.2014 | 22.09.2014 | 01.12.2014
DB-DB2-02	IBM DB2 für Linux/Unix/Windows Administration	5 Tage	1.990,00 €	14.07.2014 | 06.10.2014 | 08.12.2014
DB-DB2-05	IBM DB2 für Linux/Unix/Windows Monitoring und Tuning	3 Tage	1.290,00 €	04.08.2014 | 20.10.2014
DB-MY-01	MySQL Administration	3 Tage	1.190,00 €	21.07.2014 | 15.09.2014 | 17.11.2014
MS-SQL-01	Querying SQL Server 2012	5 Tage	1.890,00 €	14.07.2014 | 08.09.2014 | 03.11.2014
MS-SQL-02	Administering Microsoft SQL Server 2012	5 Tage	1.890,00 €	25.08.2014 | 20.10.2014
MS-SQL-03	Schreiben von Abfragen mit MS SQL Server 2008 T-SQL	3 Tage	1.190,00 €	25.08.2014 | 20.10.2014
MS-SQL-04	Verwalten einer MS SQL Server 2008 Datenbank	5 Tage	1.890,00 €	15.09.2014 | 10.11.2014
MS-SQL-05	Implementieren eines Data Warehouse mit MS SQL Server	5 Tage	1.890,00 €	04.08.2014 | 06.10.2014 | 08.12.2014

Entwicklung

Juli - Dezember 2014

P-PHP-01	PHP Programmierung Grundlagen	5 Tage	1.690,00 €	18.08.2014 | 10.11.2014
P-PERL-01	Perl Programmierung Grundlagen	5 Tage	1.690,00 €	08.09.2014 | 01.12.2014
P-PERL-02	Perl Programmierung Aufbau	5 Tage	1.690,00 €	07.07.2014 | 22.09.2014 | 08.12.2014
P-UNIX-01	Shell, Awk und Sed	5 Tage	1.690,00 €	15.09.2014 | 17.11.2014
P-XML-01	Einführung in XML	3 Tage	1.190,00 €	04.08.2014 | 24.11.2014

Web- und Application Server

Juli - Dezember 2014

INT-04	Apache Web-Server Installation und Administration	3 Tage	1.190,00 €	29.09.2014 | 15.12.2014
INT-07	Tomcat Konfiguration und Administration	3 Tage	1.190,00 €	18.08.2014 | 27.10.2014
INT-08	WebSphere Application Server Installation und Administration	3 Tage	1.390,00 €	01.09.2014 | 24.11.2014
INT-11_7	Administration und Konfiguration für JBoss 7	3 Tage	1.190,00 €	08.09.2014 | 03.11.2014
DB-ORA-50	Oracle Weblogic Administration Grundlagen	3 Tage	1.390,00 €	14.07.2014 | 06.10.2014 | 15.12.2014

Informationen und Anmeldung

Zentrale:
ORDIX AG
Westernmauer 12 - 16
33098 Paderborn
Tel.: 05251 1063-0

Seminarzentrum:
ORDIX AG
Kreuzberger Ring 13
65205 Wiesbaden
Tel.: 0611 77840-00

Online-Anmeldung, aktuelle Seminarinhalte und Termine unter:

http://training.ordix.de

Unser Seminarstandort ist Wiesbaden.

Die hier angegebenen Termine entsprechen dem ersten Seminartag.

Die Preise gelten pro Seminar pro Teilnehmer in Euro zzgl. ges. MwSt., Inhouse-Preise auf Anfrage.

Betriebssysteme und Monitoring

Juli - Dezember 2014

BS-01	Unix/Linux Grundlagen für Einsteiger	5 Tage	1.690,00 €	14.07.2014 | 15.09.2014 | 08.12.2014
BS-25	Unix Aufbauseminar für Datenbank- und Applikationsbetreuer	5 Tage	1.790,00 €	14.07.2014 | 13.10.2014
BS-02	Linux Systemadministration	5 Tage	1.690,00 €	28.07.2014 | 22.09.2014 | 15.12.2014
BS-09	Linux Hochverfügbarkeits-Cluster	5 Tage	1.890,00 €	21.07.2014 | 20.10.2014
BS-03-11	Solaris 11 Systemadministration Grundlagen	5 Tage	1.990,00 €	07.07.2014 | 06.10.2014
BS-04-11	Solaris 11 Systemadministration Aufbau	5 Tage	1.990,00 €	04.08.2014 | 27.10.2014
BS-06-11	Solaris 11 für erfahrene Unix/Linux-Umsteiger	5 Tage	1.990,00 €	25.08.2014 | 03.11.2014
BS-24	Solaris 11 Administration Neuheiten	3 Tage	1.290,00 €	01.09.2014 | 01.12.2014
BS-18	Solaris Virtualisierung mit ZFS und Container (Zonen)	5 Tage	1.990,00 €	28.07.2014 | 20.10.2014
BS-23	Solaris Virtualisierung mit LDOM	3 Tage	1.290,00 €	08.09.2014 | 08.12.2014
AIX-01	IBM AIX Systemadministration Grundlagen	5 Tage	1.990,00 €	18.08.2014 | 10.11.2014
AIX-02	IBM AIX Installation, Backup und Recovery mit NIM	3 Tage	1.290,00 €	11.08.2014 | 17.11.2014
SM-NAG-01	Systemüberwachung mit Nagios Grundlagen	3 Tage	1.190,00 €	11.08.2014 | 27.10.2014
SM-NAG-02	Systemüberwachung mit Nagios Aufbau	2 Tage	890,00 €	14.08.2014 | 30.10.2014

Projekt-/IT-Management

Juli - Dezember 2014

PM-01	IT-Projektmanagement - Methoden und Techniken	5 Tage	1.990,00 €	18.08.2014 | 13.10.2014
PM-06	Projekte souverän führen - Systemisches Projektmanagement	4 Tage	1.850,00 €	15.09.2014 | 10.11.2014
PM-08	Agiles Projektmanagement mit SCRUM	2 Tage	1.100,00 €	22.09.2014 | 01.12.2014
PM-08-Z	SCRUM Praxis und Zertifizierung	1 Tag	250,00 €	24.09.2014 | 03.12.2014
PM-10	IT-Controlling	3 Tage	1.690,00 €	08.09.2014 | 17.11.2014
PM-07	Krisenmanagement in Projekten	2 Tage	1.100,00 €	11.08.2014 | 20.10.2014
PM-11	Konfliktmanagement	2 Tage	1.100,00 €	13.08.2014 | 22.10.2014
PM-12	Stresskompetenz für IT-Führungskräfte	1 Tag	550,00 €	10.09.2014 | 26.11.2014
PM-13	Stresskompetenz für IT-Mitarbeiter	1 Tag	550,00 €	08.09.2014 | 24.11.2014
MGM-02	IT-Architekturen	3 Tage	1.650,00 €	07.07.2014 | 06.10.2014 | 15.12.2014
MGM-07	IT-Strategien effizient entwickeln	2 Tage	1.100,00 €	04.08.2014 | 03.11.2014
MGM-03	IT-Management	5 Tage	1.990,00 €	29.09.2014 | 08.12.2014
MGM-05	IT-Risikomanagement	3 Tage	1.650,00 €	25.08.2014 | 27.10.2014
PM-05	IT-Projektcontrolling	3 Tage	1.290,00 €	16.06.2014
MGM-04	Geschäftsprozessmanagement	3 Tage	1.650,00 €	01.09.2014 | 24.11.2014

Java/JEE

Juli - Dezember 2014

E-SWA-01	Softwarearchitekturen	5 Tage	1.890,00 €	22.09.2014 | 24.11.2014
OO-01	Einführung in die Objektorientierte Programmierung	3 Tage	1.190,00 €	08.09.2014 | 03.11.2014
P-JAVA-01	Java Programmierung Grundlagen	5 Tage	1.690,00 €	07.07.2014 | 06.10.2014
P-JAVA-03	Java Programmierung Aufbau	5 Tage	1.690,00 €	28.07.2014 | 27.10.2014
P-JAVA-11	Java 8 Neuheiten	2 Tage	990,00 €	05.08.2014 | 28.10.2014
P-JAVA-12	Java EE kompakt - Power Workshop	5 Tage	1.690,00 €	15.09.2014 | 17.11.2014
P-JEE-01	JEE für Entscheider	1 Tag	590,00 €	04.08.2014 | 27.10.2014
P-JEE-03A	JSP und Servlet Programmierung	5 Tage	1.590,00 €	25.08.2014 | 13.10.2014
P-JEE-05	Web-Anwendungen mit JavaServer Faces (JSF)	5 Tage	1.590,00 €	21.07.2014 | 01.09.2014 | 01.12.2014
INT-05	Java Web Services	3 Tage	1.190,00 €	11.08.2014 | 20.10.2014
P-JEE-08	Java Performance Tuning	3 Tage	1.290,00 €	08.09.2014 | 08.12.2014

Ihr Wunschseminar ist nicht dabei? Dann nehmen Sie Kontakt mit uns auf. Wir beraten Sie gerne individuell und kompetent. Fordern Sie unser **Seminarprogramm 2014** mit über 100 verschiedenen Seminaren und über 340 Terminen an. Kontaktieren Sie uns über www.ordix.de

Sie benötigen nur z.B. die Kompetenz-Seminare im Bereich Projekt-/IT-Management? Kein Problem. Wir haben für Sie diese Weiterbildungen in einer Spezialausgabe zusammengefasst. In Kürze werden weitere folgen!

Java 8 - Die neue Version (Teil II)

Lambda Expressions

Eine der wichtigsten und lang ersehnten Neuerungen von Java 8 sind die Lambda Expressions. Lange musste die Java-Gemeinde auf diese Änderung im Java Development Kit (JDK) warten, doch mit Java 8 ist es endlich soweit. In diesem zweiten Teil unserer Artikelreihe stellen wir Ihnen die Funktionsweise von Lambda Expressions und Method References vor und überprüfen, ob das Versprechen von kürzerem und besser verständlichem Java Code erfüllt wird.

Historie

Lambda Expressions oder auch anonyme Funktionen ermöglichen es einer Methode, eine Funktion als Parameter zu übergeben. Lambda Expressions sind bereits seit 1958 in der Programmiersprache Lisp verfügbar. Seither unterstützen viele moderne Programmiersprachen anonyme Funktionen.

Lange mussten Java-Entwickler auf diese Funktionalität verzichten und stattdessen auf anonyme Klassen zurückgreifen. Diese verfolgen zwar ein ähnliches Konzept, führen aber häufig zu überladendem und schlecht lesbarem Code. Mit dem JDK 8 sind Lambda Expressions nun auch in Java angekommen.

Lambda Expressions sollen aber nicht nur zu übersichtlicherem und verständlichem Code führen, sondern bringen durch die Erweiterung des Collection Framework ganz neue Möglichkeiten im Umgang mit Aufzählungselementen mit sich. Deren Verarbeitung kann einfacher parallelisiert und Operationen können erst bei Bedarf durchgeführt werden.

In diesem Artikel wollen wir Ihnen die Funktionsweise von Lambda Expressions und Method References näher bringen. Auf parallelisierte Algorithmen und Multithreading werden wir im dritten Teil dieser Reihe näher eingehen.

Anonyme Klassen

Bevor wir uns näher mit den Lambda Expressions beschäftigen, soll zunächst auf die Funktionsweise und Verwendung von anonymen Klassen eingegangen werden. Anonyme Klassen sind ein Spezialfall der lokalen Klassen. Sie

finden häufig dann Verwendung, wenn ein Interface implementiert werden soll, das nur an einer einzigen Stelle Verwendung findet und eine eigene Klasse zu aufwändig wäre.

Die Implementierung einer anonymen Klasse sollte allerdings kurz und einfach gehalten werden. Im Gegensatz zu anderen Klassenarten werden anonyme Klassen im Kontext von Ausdrücken definiert. Ein Objekt einer anonymen Klasse wird nur erzeugt, wenn der entsprechende Code auch durchlaufen wird und die Klasse mit **new** instanziiert wird.

Eine anonyme Klasse wird bei der Verwendung nicht nur definiert, sondern zeitgleich instanziiert. Das dabei erzeugte Objekt dient als Parameter der Methode. Abbildung 1 veranschaulicht das Interface **Comparator** und die Implementierung der Methode **compare()** in Form einer anonymen Klasse.

```
// Anonyme Klasse
Collections.sort(liste, new
Comparator<NatuerlichePerson>() {
  @Override
   public int compare(NatuerlichePerson person1,
    NatuerlichePerson person2) {
      return person1.getName()
         .compareTo(person2.getName());
  }
});
```

Abb. 1: Anonyme Klasse.

Funktionale Interfaces

Einen Ausschnitt des Interface **Comparator** aus dem Package **java.util** Abbildung 2 aufgezeigt. Bei der Implementierung solcher Interfaces werden häufig anonyme Klassen verwendet.

```
@FunctionalInterface
public interface Comparator<T> {

    int compare(T o1, T o2);

    ...

}
```

Abb. 2: Funktionales Interface Comparator.

Darüber hinaus verdeutlicht die Abbildung 2 bereits, dass Interfaces mit der neuen Java-Version mittels der Annotation **@FunctionalInterface** kenntlich gemacht werden können. Der Compiler der IDE kann so direkt prüfen, ob es sich bei dem annotierten Interface um ein valides funktionales Interface handelt. Ein Interface ist dann ein funktionales Interface, wenn es in der Signatur nur eine einzige Methode deklariert. Darüber hinaus können Interfaces in Java 8 auch Default-Methoden enthalten.

```
@FunctionalInterface
public interface MyFunctionalInterface {

  void takeTime(LocalDateTime time);

  default String timeFormatter(LocalDateTime time) {
    return time.format(DateTimeFormatter.ISO_DATE);
  }
}
```

Abb. 3: Default-Methoden in funktionalen Interfaces.

Wenn eine Klasse ein Interface implementiert, müssen - wie bisher - die abstrakten Methoden des Interface überschrieben werden. Das Überschreiben von Default-Methoden ist optional. Abbildung 3 zeigt ein funktionales Interface, welches die Default-Methode **timeFormatter()** implementiert, die das übergebene Objekt **LocalDateTime** als String im Format „<Jahr>-<Monat>-<Tag>“ (z.B. 2011-12-03) zurückliefert.

Im Gegensatz zur Implementierung einer anonymen Klasse, ermöglichen Lambda Expressions eine alternative und kompakte Schreibweise. Funktionale Interfaces sind somit die Voraussetzung für die Verwendung von Lambda Expressions.

In der Java-Klassenbibliothek existieren bereits eine Vielzahl solcher funktionalen Interfaces und mit Java 8 sind noch weitere hinzugekommen. Somit bieten sich viele Anwendungsmöglichkeiten für Lambda Expressions.

Syntax und Aufbau von Lambda Expressions

Die Implementierung von anonymen Klassen kann schnell zu schwer lesbarem Java Code führen. In den meisten Fällen ist die Funktionalität der anonymen Klasse sehr speziell und man wünscht sich, einfach diese Funktionalität übergeben zu können. Stattdessen muss sie aber in Form einer anonymen Klasse und dem daraus resultierenden Objekt zur Verfügung gestellt werden. Lambda Expressions ermöglichen und vereinfachen den Umgang mit Funktionen als Methodenparameter deutlich.

Syntaktisch besteht eine Lambda Expression aus einer Parameterliste und einem Methodenrumpf.

Mittels des Operator **->** werden Parameterliste und Methodenrumpf miteinander verknüpft. Ein Ausdruck ist dann zuweisungskompatibel, wenn zu einem funktionalen Interface mit nur einer Methode die korrekte Anzahl an Parametern und der passende Ergebnistyp verwendet wird.

```
String[] names = {"Thomas", "Sebastian", "Christoph",
   "Tobias", "Andreas", "Robert", "Hubert"};

// Anonyme Klasse
Arrays.sort(names, new Comparator<String>() {
   @Override
   public int compare(String name1, String name2) {
     return name1.compareTo(name2);
   }
});

// Lambda Expression
Arrays.sort(names, ((name1, name2) ->
   name1.compareTo(name2)));
```

Abb. 4: Anonyme Klasse vs. Lambda Expression.

```
// Lambda Expression
Collections.sort(liste, ((person1, person2) -> {
    int c = person1.getNachname()
       .compareTo(person2.getNachname());
    if (c == 0) {
       c = person1.getVorname()
           .compareTo(person2.getVorname());
    }
    return c;
}));
```

Abb. 5: Mehrzeilige Lambda Expression.

```
String[] names = {"Thomas", "Sebastian", "Christoph",
  "Tobias", "Andreas", "Robert", "Hubert"};

// Lambda Expression
Arrays.sort(names, ((name1, name2) ->
  name1.compareTo(name2)));

// Method Reference
Arrays.sort(names,String::compareTo);
```

Abb. 6: Method Reference.

Glossar

IDE
Integrated Development Environment – Eine Integrierte Entwicklungsumgebung ist ein Werkzeug zur Programmentwicklung, das Editor, Compiler und ggf. auch Linker und Debugger unter einer einheitlichen Oberfläche zur Verfügung stellt. Die einzelnen Programmteile sind in der Lage, miteinander zu kommunizieren, so dass zum Beispiel während der Ausführung auftretende Programmfehler im Quelltext markiert werden und beseitigt werden können.

Ein Beispiel des direkten Vergleichs zwischen einer anonymen Comparator-Implementierung und einer Lambda Expression zeigt die Abbildung 4.

Eine Lambda Expression kann auch mehrere Anweisungen enthalten. Diese müssen dann aber mittels geschweifter Klammern in einen Block gesetzt werden. In der Abbildung 5 ist eine solche Lambda Expression anhand der Methode **`sort`** dargestellt. Zunächst wird der Nachname und anschließend der Vorname des Objektes **`Person`** verglichen und entsprechend sortiert.

Der Compiler ist in der Lage, aus dem Kontext den Datentyp der Parameter einer Lambda Expression selbstständig zu erkennen. Das Voranstellen der Typenbezeichnung ist somit nicht erforderlich.

Der Methodenrumpf einer Lambda Expression kann direkt auf alle lokalen und privaten Variablen zugreifen, die in der umgebenden Toplevel-Klasse und in derselben Code-Ebene definiert sind. Die lokalen Variablen müssen **`final`** deklariert sein oder sich zumindest so verhalten, sprich effektiv unveränderlich sein. Das heißt, dass die Variablen nach der ersten Zuweisung nicht mehr verändert werden. Technisch gelten hier die gleichen Einschränkungen wie auch bei anonymen Klassen. Der Modifier **`final`** muss mit Java 8 jedoch nicht mehr explizit angegeben werden.

Der Einsatz von Lambda Expressions hat neben vereinfachtem und kürzerem Code den weiteren Vorteil, dass im Gegensatz zu anonymen Klassen keine zusätzlichen Bytecode-Dateien erzeugt werden.

Gültigkeitsbereiche

Lambda Expressions definieren keinen eigenen Gültigkeitsbereich wie lokale und anonyme Klassen. Der Gültigkeitsbereich ist hier Teil der Definitionsumgebung und übernimmt deren Variablen. Dies hat zur Folge, dass die Parameter und lokalen Variablen einer Lambda Expression mit lokalen Variablen der umliegenden Methode kollidieren. Anders als bei anonymen Klassen bezieht sich das **`this`** auf die Definitionsumgebung und referenziert nicht auf die Instanz der Klasse innerhalb des Ausdrucks. Ähnliches gilt für den Aufruf **`super`**, welcher in diesem Fall auf die Oberklasse der Basisklasse verweist, in welcher die Lambda Expression definiert wird.

Auch die Anweisungen **`break`** und **`continue`** können durchaus innerhalb einer Lambda Expression zur Steuerung des internen Kontrollflusses verwendet werden.

Referenzen auf Methoden

In einigen Fällen führt eine Lambda Expression nur einen Aufruf auf eine bestimmte Methode aus. Hier ist es oftmals kürzer und übersichtlicher die Lambda Expression durch eine Method Reference zu ersetzen. Die Syntax einer

Method Reference für statische Methoden definiert sich über **`Klasse::statischeMethode`** und übergibt eine Referenz auf eine Methode. Auch nicht-statische Methoden von Objekten lassen sich als Parameter übergeben. In diesem Fall lautet die Schreibweise **`objektvariable::instanzMethode`**. Alternativ bietet Java hierfür die folgende Schreibweise an, die der ersten Variante ähnelt: **`Klasse::instanzMethode`**.

Syntaktisch besteht eine Method Reference folglich aus einem Receiver und dem Namen der statischen Methode oder der Instanzmethode.

Receiver und Methodenname werden durch das Symbol „**`::`**“ verbunden. Der Receiver wird dabei stets vor den zwei Doppelpunkten und der Methodenname jeweils dahinter angegeben. Der Receiver kann einem Objekt oder aber einem Typ entsprechen. Eine Methode unter Verwendung einer Method Reference wird nicht aufgerufen sondern referenziert.

In unserem Beispiel aus Abbildung 6 wird **`sort()`** eine Referenz auf die Methode **`compareTo()`** der Klasse **`String`** übergeben. Alternativ lässt sich das Beispiel aus Abbildung 4 auch mittels einer Method Reference umsetzten.

Hinter der Method Reference verbirgt sich semantisch nichts anders als die folgende Lambda Expression:

```
(name1, name2) -> String.compareTo(name1, name2)
```

Fazit und Ausblick

Lambda Expressions sind eine neue syntaktische Struktur in Java, an die man sich zunächst gewöhnen muss. Dennoch können Lambda Expressions und Method References dazu beitragen, Java Code übersichtlicher zu gestalten. Die Implementierung von anonymen Klassen kann bereits in vielen Fällen durch Lambda Expressions oder sogar Method References ersetzt werden.

Darüber hinaus ermöglichen Erweiterungen der Collection API im Zusammenhang mit Lambda Expressions vereinfachte Möglichkeiten zur parallelen Datenverarbeitung.

Nachdem dieser Artikel die Funktionsweise von Lambda Expressions vorgestellt hat, werden sich die Folgeartikel dieser Reihe mit weiteren Neuerungen in Java 8 beschäftigen. Sie können sich auf Themen wie Stream API - Parallele Algorithmen, Date & Time API und IO/NIO freuen.

Links

- [1] ORDIX® news Artikel 1/2014
  „Java 8 - Die neue Version (Teil I): Ein erster Überblick“:
  http://www.ordix.de/images/ordix/onews_archiv/1_2014/ORDIX_news_1_2014_opf_files/WebSearch/page0038.html
- [2] Internetseite der Objectsystems GmbH
  http://www.objectsystems.de

Quellen

- [1] Lambda Expressions:
  http://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html
- [2] Method References:
  http://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html

Bildnachweis

- © grapicburger.com | 3D Wooden Logo Mockup

Sebastian Grimm
(info@objectsystems.de)

Neuerungen im Microsoft SQL Server 2014 (Teil I)

In-Memory OLTP - Wenn es schneller werden soll, muss es anders gemacht werden

Vor über fünf Jahren wurde unter dem Projektnamen „Hekaton“ mit der Entwicklung begonnen, jetzt hat die Technologie die Marktreife erreicht. In der vor kurzem erschienenen Version 2014 des Microsoft SQL Servers bietet „In-Memory OLTP“ die notwendige Funktionalität, um den wachsenden Anforderungen an die Verarbeitungsgeschwindigkeit gerecht zu werden. Dieser Artikel gibt einen ersten Einblick in die wichtigsten Konzepte und Komponenten dieser Technologie.

Historie

Der Bedarf an immer höherem Durchsatz und geringeren Latenzzeiten ist auch im Datenbankumfeld ungebrochen. In der Vergangenheit wurde die benötigte Steigerung der Performance durch eine verbesserte Software und eine regelmäßige Leistungsverdoppelung der CPUs erreicht.

Diese Entwicklung hat aber inzwischen ihr Ende erreicht, da die Datenbankmanagementsysteme ausgereift sind und die CPUs nicht mehr schneller werden können (siehe Abbildung 1).

Zudem ist die bisherige Speicherung und Verarbeitung der Daten auf Festplatten optimiert. Die 8 KB großen Seiten sind das Maß aller Dinge. Nur der für die aktuelle Verarbeitung notwendige Teil soll sich noch im Hauptspeicher befinden müssen. Konkurrierende Zugriffe werden mit Hilfe von Sperren koordiniert bzw. weitestgehend serialisiert.

Kurz gesagt: Es wird (Haupt-)Speicherplatz eingespart, wofür aber mit Wartezeiten bezahlt werden muss.

Die Idee

Der von Microsoft verfolgte Ansatz versucht, die Anzahl der Verarbeitungsschritte und die benötigte Wartezeit radikal zu reduzieren. Das ist aber nur gelungen, indem grundlegende Änderungen an der Architektur vorgenommen wurden.

Mit den stetig sinkenden Preisen für Hauptspeicher ist bei immer mehr Systemen eine Speicherung zumindest der zentralen Tabellen komplett im Hauptspeicher möglich. Durch den Verzicht auf Sperren und die Versionierung der einzelnen Datensätze mit Hilfe von Zeitstempeln wurde die Grundlage für eine neue Art der Verarbeitung geschaffen.

Gleichzeitig wurde diese neue Architektur aber so in den SQL Server integriert, dass in vielen Fällen keine Änderungen an den Applikationen notwendig sind. Hauptspeicheroptimierte und klassische Tabellen können zusammen in einer Transaktion verwendet werden. Fast der komplette Sprachumfang von Transact-SQL ist auf die neuen Tabellen anwendbar.

Keine Sperren durch Versionierung der Daten

Ein zentrales Merkmal der neuen Speicherstruktur ist die Versionierung der einzelnen Datensätze mit zwei Zeitstempeln: Dem Beginn und dem Ende der Gültigkeit in Form von Sequenznummern der zugehörigen Transaktionen. Damit können parallel verlaufende Transaktionen immer beurteilen, ob der gerade gelesene Datensatz im (zeitlichen) Kontext der eigenen Transaktion gültig ist oder nicht.

Auch wenn es weiterhin möglich ist Datensätze logisch zu ändern, also **UPDATE**-Statements zu verwenden, wird technisch gesehen immer ein neuer Datensatz erzeugt. Die vorherige Version des Datensatzes wird dann durch den Eintrag des Ende-Zeitstempels als nicht mehr gültig markiert.

Schreibende Transaktionen behindern damit nicht mehr lesende Transaktionen und auf das Setzen von Sperren wird komplett verzichtet. Damit entfallen auch die Wartezeiten durch Locks und Latches, was zu einer deutlichen Steigerung der Performance führt.

Allerdings ist hierbei zu beachten, dass es sehr wohl Konflikte zwischen zwei schreibenden Transaktionen geben kann, wenn sie den gleichen Datensatz ändern. Dieser Konflikt wird beim Commit erkannt und führt zu einem Abbruch einer der beiden Transaktionen. Durch die Steigerung der Performance sinkt gleichzeitig die Transaktionsdauer und mit ihr das Risiko, dass solche Situationen entstehen. Die Applikation muss aber in jedem Fall mit einer solchen Situation umgehen können.

Nur noch Indizes - dafür aber zwei neue Arten

Das zweite zentrale Merkmal der neuen Speicherstruktur ist die ausschließliche Nutzung von Indizes zur Verwaltung der Datensätze. Eine unstrukturierte Speicherung der Datensätze, wie in klassischen Heap-Tabellen, ist nicht möglich. Um die speziellen Anforderungen der neuen Technologie zu berücksichtigen, wurden zwei neue Arten von Indizes geschaffen: Hash-Indizes und Bw-Bäume.

Die Basis eines Hash-Index ist eine Hash-Funktion sowie eine Liste mit allen möglichen Ergebnissen dieser Funktion, die im Hauptspeicher abgelegt wird (im Folgenden Hash-Tabelle genannt). Wird nun ein Datensatz eingefügt, so wird er an einer beliebigen Stelle im Hauptspeicher abgelegt und die Hash-Funktion auf die indizierte(n) Spalte(n) angewendet.

Das Ergebnis der Hash-Funktion bestimmt, an welcher Stelle der Hash-Tabelle der Verweis auf die Speicheradresse des Datensatzes eingetragen wird. Ist dort bereits ein Datensatz vermerkt, so wird der Verweis auf den neuen Datensatz in den bereits bestehenden eingefügt. Alle Datensätze, bei denen die Hash-Funktion dasselbe Ergebnis liefert werden somit in Form einer Kette miteinander und mit der Liste verbunden. Die Abbildung 2 zeigt hierzu ein Beispiel, in dem als Hash-Funktion die Länge des Namens verwendet wird.

Ein schneller Zugriff auf die einzelnen Datensätze ist damit aber nur möglich, wenn der exakte Wert der indizierten Spalte(n) bekannt ist und damit die Hash-Funktion angewendet werden kann. Für Abfragen über Wertebereiche (Range-Scans) wird die zweite Index-Art, der Bw-Baum verwendet.

Die Basis dieser Index-Architektur ist der klassische Index-Baum, wie er auch beim SQL Server verwendet wird. Um hier auf Sperren verzichten zu können, werden Änderungen nicht direkt an den einzelnen Knoten vorgenommen. Für jede Änderung wird ein sogenannter Delta-Knoten erzeugt, auf dem die neue Information sowie ein Verweis auf den ursprünglichen Knoten abgelegt wird. Alle ursprünglich auf den Knoten zeigenden Verweise zeigen anschließend auf den Delta-Knoten. Eine erneute Änderung erzeugt einen weiteren Delta-Knoten mit Verweis auf den ersten Delta-Knoten. Bei mehr als 16 Knoten findet eine Konsolidierung der Informationen auf einen neuen Knoten statt, um die Verarbeitung wieder zu beschleunigen und den Hauptspeicherbedarf zu senken.

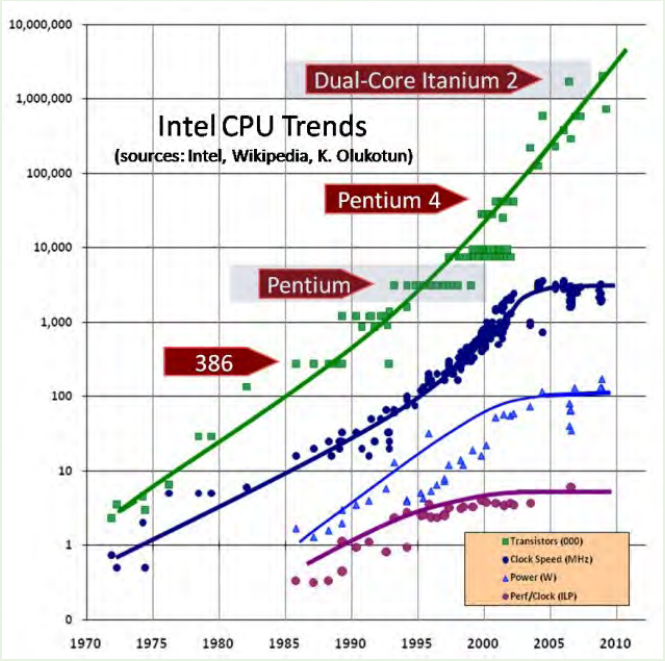


Abb. 1: Die Taktraten (Clock Speed) haben ihr Maximum erreicht. (Quelle [1])

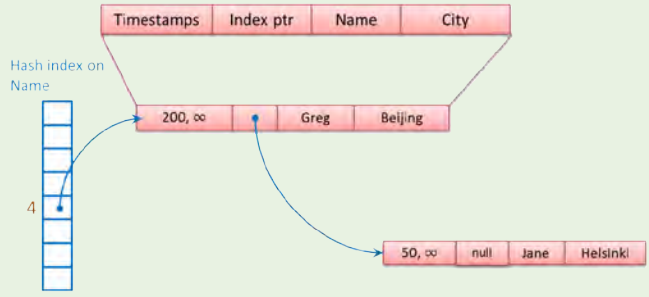


Abb. 2: Ein Hash-Index mit zwei Datensätzen. (Quelle [2])

Auf die Festplatte wird sequenziell geschrieben

Das Prinzip, Daten möglichst nicht zu ändern sondern neue Versionen zu schreiben, wird auch beim Zugriff auf die Festplatte verwendet. Denn selbstverständlich können die Inhalte der speicheroptimierten Tabellen auch auf der Festplatte gesichert werden, um nach einem Neustart der Instanz wieder zur Verfügung zu stehen. Allerdings werden auch hier neue Strukturen geschaffen. Microsoft verwendet als Basis die in der Version 2005 eingeführte Technologie „Filestream“. Aus Sicht der Datenbank werden die Tabellen dabei in separaten Dateigruppen gespeichert. Technisch gesehen sind es Verzeichnisse im Dateisystem.

Glossar

DLL
Dynamic Link Library (DLL) bezeichnet allgemein eine dynamische Programmbibliothek. Jeglicher Programmcode, der von mehr als einer Anwendung benötigt wird, wird deshalb in einer einzelnen Datei gespeichert und nur einmal in den Hauptspeicher geladen, wenn mehrere Programme dieselbe Programmbibliothek benötigen.

Latches
Latches sind kurzfristige Sperren einer Seite im Hauptspeicher während des physikalischen Zugriffs durch einen Prozess.

Locks
Sperren sind ein Mechanismus, der von Datenbankmanagementsystemen zum Synchronisieren des gleichzeitigen Zugriffs auf dieselben Daten durch mehrere Benutzer verwendet wird. Ein Prozess setzt Sperren, um andere Prozesse darüber zu informieren ob Daten gelesen oder geändert werden. Andere Prozesse werden so unter Umständen davon abgehalten, auf diese Daten zuzugreifen.

Heap-Tabelle
Eine Heap-Tabelle ist eine Tabelle ohne gruppierten Index, die Datensätze sind damit nicht logisch geordnet.

Quellen/Links

- [1] Syed Muhammad Usman Pirzada:
  Moores Law will be Dead by 2020 Claim Experts –
  Last Fab Process for CPU/GPUs to be 5nm
  http://wccftech.com/moores-law-will-be-dead-2020-claim-experts-fab-process-cpug-pus-7-nm-5-nm/
  Verwendete Abbildung:
  http://cdn4.wccftech.com/wp-content/uploads/2013/09/CPU-Scaling.jpg
- [2] Delaney, Kalen:
  SQL Server In-Memory OLTP Internals Overview for CTP2
  http://sqlblog.com/blogs/kalen_delaney/archive/2013/10/20/sql-server-2014-in-memory-oltp-hekaton-whitepaper-for-ctp2.aspx
- [3] Microsoft Developer Network:
  In-Memory OLTP (In-Memory Optimization)
  http://msdn.microsoft.com/en-us/library/dn133186(v=sql.120).aspx
- [4] Levandoski, Justin J. / Lomet, David B. / Sengupta, Sudipta:
  The Bw-Tree: A B-tree for New Hardware Platforms
  http://research.microsoft.com/pubs/178758/bw-tree-icde2013-final.pdf

Hier werden die Daten in 128 MB großen Dateien (genannt Datendatei) abgelegt, die immer nur sequenziell geschrieben werden. Werden Datensätze gelöscht, so werden diese nicht aus den Dateien entfernt, sondern Einträge in einer zusätzlichen Datei (genannt Deltadatei) erzeugt. Für jede Datendatei existiert immer genau eine zugehörige Deltadatei.

Ein Hintergrundprozess analysiert regelmäßig die Anzahl der noch aktiven Zeilen in den Datendateien und fügt zwei oder mehr benachbarte Datendateien zu einer Datei zusammen, wenn sie jeweils weniger als 50% aktive Zeilen beinhalten.

Native Kompilierung beschleunigt den Zugriff zusätzlich

Die Zugriffe auf hauptspeicheroptimierte Tabellen erfolgen zwar aus Sicht des Anwenders mit den klassischen SQL-Befehlen (**`SELECT`**, **`INSERT`**, **`UPDATE`**, **`DELETE`**), der direkte Zugriff auf die Daten im Hauptspeicher erfolgt jedoch mit DLLs, die bei der Erstellung der Tabelle erstellt, kompiliert und dann geladen werden. Dies ist auch der Grund, warum eine solche Tabelle nicht nachträglich geändert werden kann.

Auch benutzerdefinierte Prozeduren, die ausschließlich auf hauptspeicheroptimierte Tabellen zugreifen, können kompiliert abgelegt und verwendet werden, um die Performance weiter zu steigern.

Fazit

Microsoft hat mit „In-Memory OLTP“ neue Wege beschritten und erreicht damit eine signifikante Steigerung der Verarbeitungsgeschwindigkeit, je nach Art der Verarbeitung kann ein Faktor von 5 bis 20 erreicht werden.

Um diese Steigerung zu erreichen, muss aber auch die Applikation unter Umständen neue Wege gehen. Bei der Ermittlung der notwendigen Änderungen unterstützen wir Sie gerne.

Für einen tieferen Einblick in die technischen Details empfehlen wir das Whitepaper von Kalen Delaney [2].

Andreas Jordan
(info@ordix.de)

Von Oracle Database Lite 10.3.0.3 zum Oracle Database Mobile Server 11g

Die Hürden einer Migration

„Oracle Database Mobile Server 11g“ heißt der Nachfolger des verteilten Applikationssystems „Oracle Database Lite“. Vielen ist aber nur der alte Name ein Begriff. Dabei gibt es die Version 11g nun schon seit 3 Jahren und mit dem dritten Patch Set. Aber wie wechselt man am besten auf den Oracle Database Mobile Server und auf welche Änderungen muss besonders geachtet werden? Diese und weitere Fragen beantworten wir in diesem Artikel.

Hintergrund

Oracle Database Mobile Server 11g - so heißt das aktuelle System von Oracle für ein mobiles und flexibles Datenbanksystem zum Verwalten und Steuern eines Master-Client-Applikationssystems. Die Funktionsweise und die Architektur wurden bereits ausführlich in einem früheren ORDIX® news Artikel [1] vorgestellt.

Wir erinnern uns: Das Architekturmodell des Oracle Database Mobile Server (OMS) besteht aus drei Grundkomponenten (siehe Abbildung 1):

- Client
- Oracle Database Mobile Server
- Oracle Database

Als Grundlage dient zwingend eine Oracle-Datenbank in der Standard oder Enterprise Edition. Der OMS legt ein Repository in dieser Datenbank an, in dem die Konfiguration und Meta-Informationen zu den verwalteten Applikationen und Replikationsvorgängen gespeichert werden.

Der Oracle Database Mobile Server verwaltet die Verteilung der Client-Applikationen und ist für die Datenreplikation zwischen der Master- und der Client-Datenbank zuständig.

Der Client zeichnet sich dadurch aus, dass er mit verschiedenen Plattformen kompatibel ist (Windows-Systeme, Tablet-Computer, Smartphones …). Der OMS Client besteht aus einer Client-Datenbank, einem Device Management Agent und einer Sync Engine. Detaillierte Beschreibungen zum Aufbau und weitere Informationen zum Datenabgleichvorgang finden Sie in dem vorangegangenen Artikel [1].

Ende 2012 hat die Version 10g den Status des Premier Support verlassen und Ende 2015 wird auch der Extended Support auslaufen. Der Nachfolger ist Oracle Database Mobile Server 11g. Dieser ist aktuell in der Version 11.3.0.0 vorhanden und bringt im Vergleich zu 10g einige Änderungen mit sich. Im Folgenden befassen wir uns mit den Architekturänderungen und deren neuen Voraussetzungen an das Gesamtsystem. Aufbauend darauf werden Möglichkeiten diskutiert, wie eine Migration durchgeführt werden kann.

Architekturänderungen der Version 11g

Die Client-Datenbank Oracle Lite Database ist verschwunden. Stattdessen stehen als Client-Datenbank die zwei folgenden Alternativen zur Verfügung:

- SQLite Database
- Berkeley DB with SQLite

Beide Datenbanken können als kleine, kompakte Leichtfüßler unter den Datenbanken bezeichnet werden. Das Datenbanksystem besteht jeweils aus einer Datei, die lediglich einige KB groß ist. Im Gegensatz zu vielen anderen Datenbanken wird sie als fester Bestandteil in eine Applikation integriert und läuft nicht als eigenständiger Prozess.

SQLite ist ein relationales Datenbanksystem, welches lediglich aus einer C-Programmbibliothek besteht. Von ihr wird ein Großteil des SQL-92-Standards unterstützt. Die Berkeley DB (BDB) ist ähnlich wie SQLite eine eingebettete Datenbankbibliothek. Sie wurde von Sleepycat

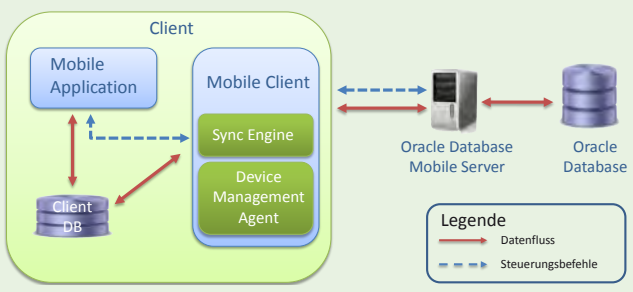


Abb. 1: Architekturmodell des Oracle Database Mobile Server.

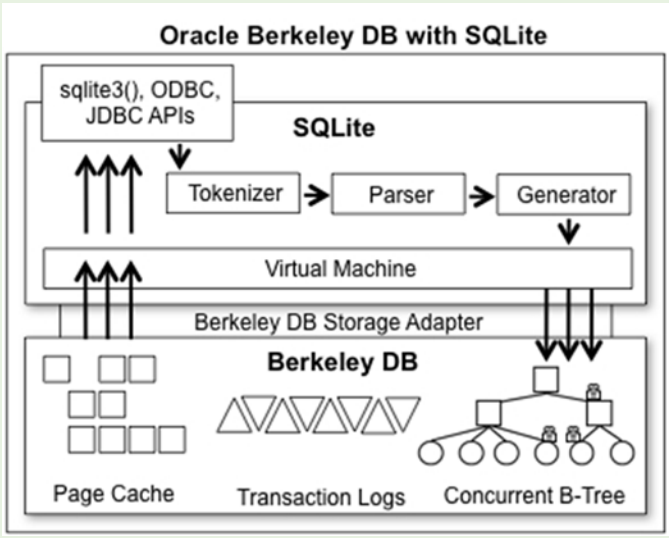


Abb. 2: SQL-Interface der Oracle Berkeley DB. (Quelle [3])

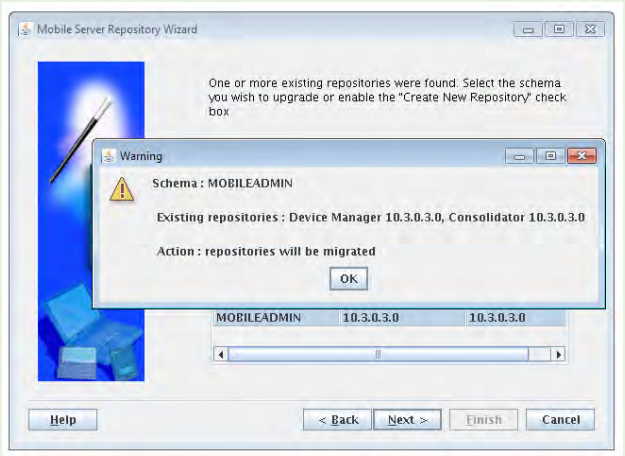


Abb. 3: Repository Migration - Hinweismeldung des repwizard.

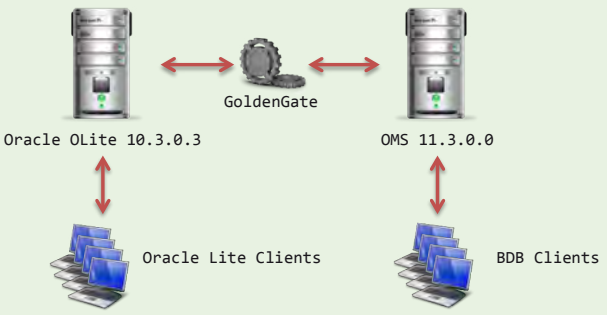


Abb. 4: Migrationskonzept mit Oracle GoldenGate.

Software entwickelt, welche im Jahr 2006 von Oracle aufgekauft wurden. Im Kern ist sie ein Key Value Store und zählt somit zu den NoSQL-Datenbanken. Die Berkeley DB wird von Oracle in drei Versionen vertrieben: Als vollständige Java-Implementierung mit dem Namen „Berkeley DB Java Edition“, als C++-Variante mit XML-Interface namens „Berkeley DB XML“ und als „Berkeley DB“ geschrieben in C. Oracle bietet für die C-Implementierung der BDB ein SQL-Interface an (siehe Abbildung 2). Der Standard Mobile Client ist für diese SQL-Schnittstelle konzipiert. Das Interface selbst basiert auf SQLite. Somit kann diese Variante der Berkeley DB mit allen Befehlen verwendet werden, die auch SQLite unterstützt.

Neben der Client-Datenbank gibt es auch neue Voraussetzungen für den Applikationsserver. Oracle Database Lite wurde mit oc4j-Containern ausgeliefert. Mit diesen konnte ein kleiner Standalone-Applikationsserver gestartet werden, der die Grundlage für den Mobile Manager bietet. OMS bringt diese Möglichkeit nicht mehr von Haus aus mit. Stattdessen wird auf externe Lösungen gesetzt. Zertifiziert ist der OMS für folgende Applikationsserver:

- Oracle WebLogic Server 11g Release 1 und Oracle WebLogic Server 12c
- Oracle GlassFish Server 3.1
- Oracle Application Server 10g
- Apache TomEE 1.5 Web Profile

Für den Datenbankserver wird eine Oracle-Installation der Version Oracle 10g Release 1, Release 2 oder Oracle 11g vorausgesetzt. Oracle 9.2 wird somit nicht mehr unterstützt.

Konsequenzen für die Applikation

Abhängig von dem eingesetzten Client-Datenbanksystem, muss ggf. auch auf Client-Seite eine Migration geplant werden. Obwohl die beiden Systeme SQLite und Berkeley DB SQL unterstützen, kann der Umstieg trotzdem weitere Hürden aufwerfen. Eine ausführliche Validierung des SQL-Code ist daher Pflicht.

Das SQL-Interface der BDB basiert auf SQLite, welches wiederum ein Großteil der SQL-92-Spezifikationen unterstützt. Typische Abfragen auf Data Dictionary Views (z.B. **`ALL_TABLES`**, **`ALL_SQUENCES`**) oder auch Oracle-typische Funktionen (z.B. **`nvl()`**) werden nicht eins zu eins unterstützt. Größtenteils gibt es Pendants zu den weggefallenen Funktionen, Tabellen und Views. Informationen zu existierenden Objekten in der Datenbank können z.B. über die Tabelle **`SQLITE_MASTER`** abgefragt werden. Gleiches gilt für **`C$SEQ_CLIENTS`** statt **`ALL_SEQUENCES`** oder **`ifnull()`** statt **`nvl()`**.

Es gilt jedoch zunächst einmal, die inkompatiblen Statements in der Applikation zu finden, wofür gegebenenfalls der Client-Applikationscode durchsucht werden muss.

Konsequenzen für die Server-Architekur

Für den Einsatz ist eine Oracle-Datenbank Pflicht. Oracle Database Lite unterstützte die Versionen 9.2, 10g Release 1, 10g Release 2 sowie 11g. Vom Oracle Mobile Server entfällt die Zertifizierung für Oracle 9.2. Jedoch reicht weiterhin eine Standardversion aus.

Die Build-In-Bibliotheken **`oc4j`** von Oracle Database Lite werden bei dem Oracle Mobile Server nicht mehr mitgeliefert. Hier empfiehlt sich der Umstieg auf den Oracle WebLogic Applicatio n Server. Das Softwarepaket des Oracle Mobile Server beinhaltet für die Installation einen Oracle Universal-Installer. Dieser erstellt auf Wunsch eine neue Domäne (oder benutzt eine existierende) innerhalb des Applikationsservers. Die entsprechende Domäne kann über das Verzeichnis **`DOMAIN_HOME`** einfach gestartet werden und der User kann wie gewohnt über seinen Browser auf den Mobile Manager zugreifen.

Die wichtigste Server-Komponente ist das Repository von Oracle Lite bzw. vom Oracle Mobile Server innerhalb der Datenbank. Dieses kann nicht direkt übernommen werden, sondern muss migriert werden.

3 Wege zum OMS

Im Folgenden werden drei Konzepte beschrieben, mit denen eine Umstellung von Oracle Database Lite zum Oracle Mobile Server durchführt werden kann:

Migration

Der Oracle Database Mobile Server wird mit einem Oracle Universal-Installer ausgeliefert. Dieser besteht aus vier Bestandteilen:

- einem (De/)Installer für den Oracle Mobile Server
- einen (De/)Installer für das Mobile Development Kit (ein Paket bestehend aus Dokumentationen, Libraries und Entwicklungsumgebungen für Oracle Mobile Server Applikationen)
- dem **`repwizard`**
- dem (De/)Installer für eine Demo-Applikation.

Der **`repwizard`** (Oracle Mobile Server Repository Wizard) ist ein zentraler Bestandteil und erlaubt die Erstellung eines Repository in einer Datenbank. Er erkennt außerdem vorhandene Repositories von Oracle Lite 10.3.0.3 und bietet eine Migrationsmöglichkeit an.

Die Abbildung 3 zeigt einen Ausschnitt einer Migration von Oracle Lite zum Oracle Mobile Server. Der abgebildete Assistent ist der **`repwizard`**. Der Wizard migriert auf Wunsch die vorhandene Version 10 sowie das Repository auf die aktuelle Version 11.3.0.3.

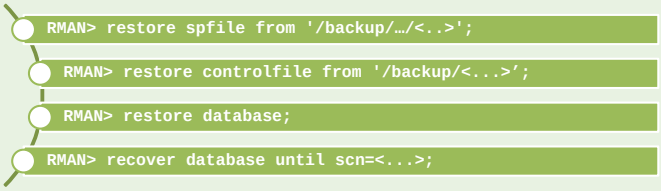


Abb. 5: Testsystemaufbau mit RMAN.

Glossar

BDB
Die Berkeley Database ist eine eingebettete Datenbankbibliothek mit Programmierschnittstellen zu C, C++, Java, Perl, Python, Tcl und weiteren Programmiersprachen.

OMS
Der Oracle Mobile Server ist der Nachfolger von Oracle Database Lite.

Key Value Store
Unter einem Key Value Store versteht man eine NoSQL-Datenbank, deren Hauptmerkmal das paarweise Speichern von Daten ist. Eine Spalte eines Datensatzes bildet dabei einen suchbaren Schlüssel, während die andere die Daten beinhaltet.

Oracle RMAN
RMAN ist ein Backup und Recovery Manager für Oracle-Datenbanken.

Ein Umstieg inklusive Wechsel der Client-Datenbank erfolgt in drei Schritten:

1. Finaler Datenabgleich zwischen Master- und Client-Datenbanken
2. Migration des Repository und damit Upgrade auf den Oracle Mobile Server
3. Getrenntes Update bzw. (De-)Installieren der Client-Software (Abhängig von der Applikationskomplexität)
4. Erster Sync zwischen Master und Client zur Erstellung und Füllung der Client-Datenbank

Neuaufbau

Alternativ ist es möglich, eine komplett neue Installation des Mobile Server mit einem neuen Repository durchzuführen. Anwendungsschema, Publikationen/Publikation-Items und Oracle Mobile User müssen dabei allerdings neu angelegt werden.

GoldenGate

In dieser Variante wird parallel zu dem vorhanden System, ähnlich wie bei dem Neuaufbau, ein neues System aufgebaut. Mit GoldenGate können beide Systeme parallel betrieben werden (siehe Abbildung 4). Dabei sorgt

Links

- [1] ORDIX® news Artikel 3/2013
  „Oracle Database Mobile Server 11g - Database to go“:
  http://www.ordix.de/ordixnews/ordix-news-archiv/3-2013.html
- [2] Oracle Mobile Server:
  http://www.oracle.com/technetwork/database/database-technologies/database-mobile-server/overview/index.html
- [3] Oracle Berkeley DB:
  http://www.oracle.com/technetwork/database/database-technologies/berkeleydb/overview/index.html
- [4] Oracle WebLogic Server:
  http://www.oracle.com/de/products/middleware/cloud-app-foundation/weblogic/overview/index.html

Quellen

- [1] Oracle: „Documentation Oracle Database Lite 10.3.0.3“
  http://docs.oracle.com/cd/E12095_01/nav/portal_booklist.htm
- [2] Oracle: „Documentation Oracle Database Lite 11.3“
  http://docs.oracle.com/cd/E48200_01/nav/portal_booklist.htm
- [3] Bildnachweis Abbildung 2:
  http://www.oracle.com/ocom/groups/public/@otn/documents/webcontent/160911.jpg

Tobias Ummler
(info@ordix.de)

GoldenGate für einen gleichen Datenbestand zwischen dem Oracle Mobile Server und Oracle Database Lite. Die Clients können dabei Stück für Stück migriert werden.

Aufbau einer Testmigration

Ein Testsystem kann relativ einfach erstellt werden. Die Grundlage bilden hierfür die Daten in der Datenbank, das Repository des Mobile Server bzw. Oracle Database Lite. Dieses lässt sich mit **`RMAN`** oder mit anderen Import-Export-Werkzeugen auf einem separaten Testsystem wieder einspielen. Sind **`RMAN`**-Backups vorhanden, so kann mit wenigen Befehlen (siehe Abbildung 5) die Datenbankinstanz mit dem System wiederhergestellt werden.

Anschließend kann z.B. direkt der Oracle Mobile Server installiert werden, um eine Migration (insbesondere die Migration des Repository durch den **`repwizard`**) zu verifizieren.

Fazit

Ein Umstieg auf den Oracle Mobile Server 11g bringt einige Änderungen an den bestehenden Server- und Client-Strukturen mit sich. Die Client-Applikation und der gegebene Umstieg von einer Oracle Lite Client-Datenbank darf dabei nicht außer Acht gelassen werden. Abhängig von der Komplexität und Größe der Applikation steckt hier viel Potential für das Übersehen von notwendigen Anpassungen. Im Vergleich zu der Oracle Lite Client-Datenbank steigen auch die Lizenzkosten für den Berkeley DB Support.

Mit dem WebLogic Server - als empfohlenen Applikationsserver - versucht Oracle auch mit dem Mobile Server seinen Applikationsserver weiter zu verbreiten. Für die Systemarchitektur bedeut et dies eine weitere Softwareinstallation (und auch weitere Lizenzkosten) oder aber eine nahtlosere Integration in die bestehende Landschaft und einen weiteren Schritt weg von dezentralen und differenzierten Applikationen.

Queuing-Theorie in Oracle-Datenbanken

Capacity Management - Viel hilft viel?

Wer kennt diese Situationen nicht: Zur Hauptverarbeitungszeit ist die Auslastung eines Systems minimal höher als üblich und sofort beklagen sich die Anwender über deutlich längere Antwortzeiten. Häufig ist dieses Phänomen auf Queuing-Probleme zurückführen. Dieser Artikel widmet sich den Grundlagen der Queuing-Theorie in Bezug auf Oracle-Datenbanken.

Zum Einstieg ein Beispiel

Die Queuing-Theorie beschäftigt sich mit dem Verhalten von Warteschlangenmodellen. Ein typisches Beispiel für ein solches Modell ist die Prozessverarbeitung durch CPUs. In Abbildung 1 ist die Abarbeitung zweier Prozesse durch eine CPU beispielhaft dargestellt. Die Abbildung stellt den zeitlichen Verlauf der Abarbeitung zweier Prozesse durch eine CPU dar. Beide Prozesse benötigen zur Verarbeitung jeweils 400 ms CPU-Zeit.

Zum Zeitpunkt t_1 fordert Prozess 1 CPU-Zeit an. Da zu diesem Zeitpunkt kein anderer Prozess verarbeitet wird, kann Prozess 1 sofort verarbeitet werden. Zum Zeitpunkt t_2 fordert nun auch Prozess 2 CPU-Zeit an. Da sich Prozess 1 noch in der Verarbeitung befindet, muss Prozess 2 zunächst warten. Erst 300 ms später ist Prozess 1 abgeschlossen und Prozess 2 erhält die angeforderte CPU-Zeit. Für Prozess 2 ergeben sich 400 ms Verarbeitungszeit und 300 ms Wartezeit, also insgesamt 700 ms Antwortzeit.

Und was ist jetzt Queuing?

Queuing beschreibt die Bildung einer Wartschlange für die wartenden Prozesse und die dadurch entstehenden Wartezeiten. Ohne Queuing entspricht die Laufzeit oder Antwortzeit eines Prozesses der benötigten CPU-Zeit. Mit Queuing setzt sich die Antwortzeit aus Ausführungszeit und Wartezeit zusammen.

Die Entstehung von Wartezeiten durch konkurrierende Ressourcenanfragen ist ein normales Verhalten, welches uns sowohl bei Computersystemen als auch im Alltag begegnet. Das eigentliche Problem stellt das Wachstum dieser Wartezeiten bei einer steigenden Anzahl an Anfragen dar.

Der Queuing-Effekt

In einer idealisierten Welt dauern alle Anfragen gleich lang und kommen mit einer exakten Taktung am Server an. In solchen Szenarien - häufig die Idealform des Benchmark - lassen sich Server bis nahezu 100 % auslasten, ohne dass es zu nennenswerten Verzögerungen kommt. In der Realität sind jedoch sowohl das Eintreffen von Aufgaben als auch die Länge der Verarbeitung nicht gleichverteilt, sondern erfolgen zufällig.

Die Verarbeitungszeit und die Ankunftsrate unterliegen hierbei in der Praxis häufig einer exponentiellen Verteilung. Hier setzen die Queuing-Modelle an und beschreiben ein exponentielles Wachstum der Warte- und Antwortzeiten bei steigender Auslastung. Dieses Verhalten eines Queuing-Modells wird auch als Queuing-Effekt bezeichnet.

Antwortzeitkurven

Zur Veranschaulichung des Queuing-Effekts bieten sich Antwortzeitkurven an. Diese stellen die Veränderung der mittleren Antwortzeit in Abhängigkeit von der Auslastung oder der Ankunftsrate dar. Die Auslastung entspricht der Anzahl der verwendeten Ressourcen (z.B. CPUs) geteilt durch die Anzahl an verfügbaren Ressourcen. Die Ankunftsrate beschreibt dabei, wie viel Arbeit pro Zeiteinheit anfällt, z.B. 10 Anfragen pro Sekunde.

In Abbildung 2 sind zwei Antwortzeitkurven für Server mit unterschiedlicher CPU-Anzahl gegenübergestellt. Eine für Queuing-Modelle typische Antwortzeitkurve steigt zunächst nur minimal an und verläuft nahezu konstant. In diesem Bereich besteht die Antwortzeit fast nur aus der Ausführungszeit. Mit zunehmender Auslastung steigt die Antwortzeitkurve jedoch drastisch an bis ihre Steigung (theoretisch) unendlich groß wird. Der Bereich, in dem aus einem moderaten Wachstum ein enormes Wachstum wird, bezeichnet man auch als „knee of the curve“ oder „elbow of the curve“.

Systemvergleiche

Schauen wir uns nun die beiden Antwortzeitkurven etwas genauer an. Die eine Kurve zeigt das Antwortzeitverhalten eines Servers mit 8 und die andere eines

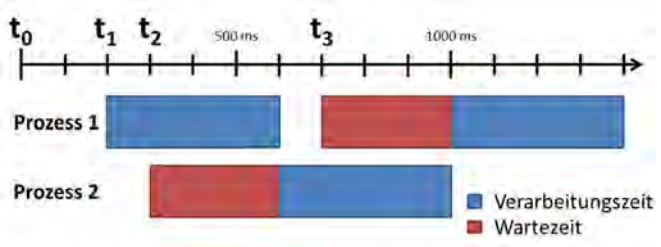


Abb. 1: Queuing am Beispiel der Prozessverarbeitung durch CPUs.

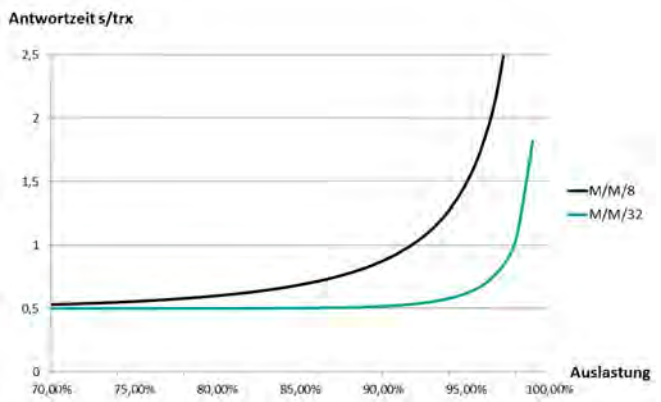


Abb. 2: Vergleich der Antwortzeitkurven zweier CPU-Systeme.

$$E_C = \frac{\frac{(m * S_t * \lambda_q)^m}{m!}}{(1 - m * S_t)\sum_{k=0}^{m-1}\frac{(m * S_t * \lambda_q)^k}{k!} + \frac{(m * S_t * \lambda_q)^m}{m!}}$$

Legende:

m – Im Beispiel Anzahl CPUs

S_t – Im Beispiel Verarbeitungszeit je Prozess

λ_q (lambda) – Im Beispiel Ankfunftsrate in Prozesse pro Millisekunde

Ec – Erlang-C-Faktor; hiermit wird die Antwortzeit exakter berechnet

Abb. 3: Beispiel der Erlang-C-Formel.

Servers mit 32 CPUs. Bei ca. 70 % Auslastung ist die mittlere Antwortzeit mit 8 CPUs nur geringfügig schlechter als mit 32 CPUs.

Bei zunehmender Auslastung liefern die 8 CPUs allerdings immer schlechtere Antwortzeiten, sodass die mittlere Antwortzeit ab ca. 92 % Auslastung über eine Sekunde beträgt. Die Antwortzeit der 32 CPUs liegt bei diesem Auslastungsgrad immer noch bei unter 0,6 Sekunden. Erst ab einer Auslastung von 98 % beträgt die mittlere Antwortzeit auch über eine Sekunde.

Das Queuing-Modell beschreibt somit, dass ein Server mit vielen CPUs im Vergleich zu einem Server mit wenigen CPUs prozentual höher ausgelastet werden kann, ohne die mittlere Antwortzeit negativ zu beeinflussen. Das Modell setzt hierbei allerdings perfekt skalierende CPUs voraus, sodass jede zusätzliche CPU zu 100 % für die Verarbeitung genutzt werden kann.

Auswirkungen von Queuing

Das Tückische am Queuing-Effekt ist die plötzliche Verschlechterung der Antwortzeit bei steigender Auslastung. Betrachtet man die Antwortzeitkurve für 8 CPUs, so sind die Auswirkungen von 10 % zusätzlicher Auslastung stark von der Ausgangsauslastung abhängig. Beträgt die momentane Auslastung 75 % und steigt auf 85 %, so erhöht sich die mittlere Antwortzeit von ca. 0,55 auf ca. 0,68 Sekunden (siehe Abbildung 2).

Ist der Ausgangspunkt jedoch bei 85 % und steigt auf 95 % so wird die mittlere Antwortzeit ca. 1,48 Sekunden betragen. Die Antwortzeit verschlechtert sich somit im zweiten Szenario signifikant stärker als im ersten. Allerdings zeigen beide Varianten, dass ab einer Auslastung von 75 % mit einer Verschlechterung der Antwortzeit durch Queuing-Effekte gerechnet werden muss.

Queuing bei großen Systemen

Für einen Server mit 32 CPUs treten massive Queuing-Probleme erst ab einer Auslastung von mehr als 90 % auf. Dafür werden in diesem Bereich bereits durch minimale Laststeigerungen enorme Performance-Probleme auftreten. Eine Steigerung der Auslastung von 94 % auf 98 % bewirkt z.B. einen Anstieg der Antwortzeit von ca. 0,58 auf etwa 1,05 Sekunden. Somit lösen bei sehr hoher Auslastung und vielen CPUs bereits geringe Laststeigerungen massive Performance-Probleme aus. Daher ist es die Aufgabe des Capacity Managements, solche Engpässe frühzeitig durch eine gezielte Datensammlung und deren Auswertung zu erkennen und Gegenmaßnahmen zu ergreifen.

Es geht genauer

Die Queuing-Theorie bietet eine Reihe mathematischer Zusammenhänge zur Berechnung von Queuing-Modellen.

Diese Zusammenhänge unterschätzen jedoch die Auswirkungen des Queuing-Effekts für Systeme mit vielen CPUs und einer hohen Auslastung. Als Folge liefern die Queuing-Modelle zu optimistische Werte für die Warte- und Antwortzeiten. Um die Wartezeit realistischer berechnen zu können, hat der dänische Mathematiker Agner Krarup Erlang (1878-1929) die nach ihm benannte Erlang-C-Formel (siehe Abbildung 3) entwickelt. Anhand dieser Formel wird ein Erlang-C-Faktor Ec bestimmt, mit dessen Hilfe eine realistischere Wartezeit bestimmt werden kann. Die Abweichungen zwischen den Ergebnissen der Standardformeln und der Erlang-C-Formel können dabei durchaus 7 % und mehr betragen. Für Systeme mit wenigen CPUs und/oder einer niedrigen Auslastung sind die Abweichungen allerdings minimal.

Ausblick

Tiefere Einblicke in die Queuing-Theorie vermitteln wir Ihnen gerne in unserem Seminar „Oracel Capacity Planning“. In einem zukünftigen Artikel werden wir uns mit der Datensammlung und Modellierung von Queuing-Modellen in Oracle-Datenbanken beschäftigen.

Glossar

Benchmark
Das Ziel eines (Computer-)Benchmark ist es, die maximale Leistungfähigkeit eines Systems zu bestimmen. Für diese Bestimmung werden die vorhandenen Ressourcen bis nahe zu 100% ausgelastet.

Queuing
Das Queuing beschreibt die Entstehung von Wartschlangen und Wartezeiten in Systemen mit einer begrenzten Anzahl an Ressourcen.

Sven Loer
(info@ordix.de)

Seminarempfehlung: Oracle Capacity Planning

► **Informationen/Online-Anmeldung: http://training.ordix.de**

In diesem Seminar lernen Sie, wie Sie die Kenndaten zur Performance und damit die Kapazität Ihrer Oracle-Datenbank erfassen und planen können. Wir zeigen Ihnen die Key-Metriken sowie verschiedene Verfahren zur Hochrechnung der zukünftigen Entwicklung Ihres Oracle-Systems, um effiziente Erweiterungen vornehmen zu können. Auch die Schätzgüten und Störgrößen werden dabei berücksichtigt.

Seminarinhalte

- Hauptmetriken eines Oracle-Systems
- Zusammenhang zwischen den Teilsystemen
- Grundlagen zur Performance-Vorhersage
- Antwortzeitkurven
- Einführung in mathematische Grundlagen der Vorhersage
- Methodische Performance-Vorhersage
- Vertiefung der Theorie durch praktische Übungen und Beispiele

Termine

11.08. - 13.08.2014 in Wiesbaden
03.11. - 05.11.2014 in Wiesbaden

Seminar-ID: DB-ORA-40
Dauer: 3 Tage
Preis pro Teilnehmer:
1.290,00 € (zzgl. MwSt.)
Frühbucherpreis:
1.161,00 € (zzgl. MwSt.)

Buchen Sie gleich hier!

Ein PHP-Framework für Web Artisans im Web

„Laravel“

Laravel ist eins von vielen PHP-Frameworks, die derzeit auf dem Markt verfügbar sind. In dem vorliegenden Artikel stellen wir Ihnen dieses Framework vor und zeigen Ihnen, welche Funktionen es abdeckt und wie leicht damit eine Anwendung erstellt werden kann.

Wer die Wahl hat…

Die Anzahl der verfügbaren PHP-Frameworks hat mittlerweile eine enorme Anzahl erreicht. Es entsteht fast der Eindruck, dass sich jeder Entwickler sein eigenes Framework schreibt, welches genau auf seine Bedürfnisse abgestimmt ist. Häufig scheint es, als diene dies nur dazu, sich nicht mit den bereits vorhandenen Frameworks auseinander setzen zu müssen und Zeit mit der Auswahl des Passenden zu verschwenden.

Ist erst einmal die Wahl auf ein Framework gefallen, ergibt sich eventuell erst später im Entwicklungsprozess die Erkenntnis, dass eine bestimmte Funktionalität doch nicht ganz den Erwartungen entspricht oder gar fehlt.

Hohe Erwartungen

In diesem Artikel werfen wir einen Blick auf das PHP-Framework „Laravel“. Wir starten mit hohen Erwartungen, denn immerhin soll das Framework für „Web Artisans“ (dt.: Kunsthandwerker des Webs, Web-Künstlern) entworfen sein. Somit erwarten wir filigrane Strukturen und facettenreiche Elemente, die uns beim Entwickeln einer Anwendung konstruktiv zur Seite stehen. Auf dieser Grundlage erwarten wir folgende Eigenschaften:

- Einfache Installation und Konfiguration
- MVC-Konzept
- Erzeugung von validem HTML-Code
- HTML-Template-Engine
- Formularvalidierung
- Datenbankanbindung
- ORM-Integration
- Dokumentation oder Support-Forum

Ein Komponist als Grundlage

Der erste Schritt besteht in der Installation von Laravel, wozu wir uns des Werkzeuges „Composer“ [1] bedienen. In der PHP-Entwicklung hat sich das Programm als

Standard etabliert, um die Einbindung von Drittanbieterprodukten in die eigenen Programme und das Auflösen von Abhängigkeiten innerhalb der einzelnen Pakete zu vereinfachen. Der „Composer“ unterstützt bei der Erstellung eines neuen Projektes auf Laravel-Basis und macht die Installation somit zu einem Kinderspiel.

Der folgende Kommandozeilenaufruf erstellt ein Verzeichnis **mein-projekt** und legt dort alle benötigten Dateien des Framework und dessen Abhängigkeiten zu Drittanbietersoftware, wie z.B. Teilen des „Symfony“-Framework oder des Logging Framework „Monolog“ ab:

```
$> composer create-project laravel/laravel mein-projekt
```

Der „composer“ lädt die benötigte Software aus dem zentralen Verzeichnis **packagist.org** herunter. Nach dem Ausführen des Befehls sind alle benötigten Pakete vorhanden und es kann sofort mit der Entwicklung begonnen werden.

Wo liegt was?

Unterhalb der entstandenen Struktur spielen die Verzeichnisse **app/config/**, **app/controller/**, **app/model/**, **app/views/** und **public/** zunächst die wichtigste Rolle. Im ersten Verzeichnis werden Konfigurationsdateien für die Anwendung abgelegt. Hier ist besonders die Datenbankkonfiguration in der Datei **database.php** hervorzuheben. In dieser können die Einstellungen für die möglichen Datenbankverbindungen hinterlegt werden.

Laravel implementiert das MVC-Entwurfsmuster. Daher liegt es nahe, dass für jeden Teil dieses Musters ein entsprechendes Verzeichnis existiert. So werden die Controller-Klassen, die Model-Klassen und Views in den oben genannten Verzeichnissen abgelegt.

Die gesamte Anwendung wird über eine zentrale index.php gesteuert, welche im Verzeichnis **public/** liegt. Da dies den Einstiegspunkt der Anwendung darstellt, wird empfohlen, das Verzeichnis **public/** als Document Root der Anwendung zu definieren. Zudem kann auf diesem Wege nicht direkt per Web auf die Framework-Dateien zugegriffen werden, was beim Thema Sicherheit eine große Rolle spielt.

Wo soll es hingehen?

Eine zentrale Datei zur Steuerung der einzelnen Anwendungsbereiche ist **app/routes.php**. Hierin wird, vereinfacht gesagt, das Mapping von URIs, die in der Anwendung genutzt werden sollen, und den entsprechenden Controllern bzw. deren Methoden eingetragen. In Abbildung 1 ist ein Beispielaufbau dieser Datei dargestellt. Hier werden zwei URIs definiert.

Der erste nimmt HTTP-Post-Anfragen unter der URI **/benutzer/anlegen** entgegen und ruft zu deren Verarbeitung die Methode **create** im **BenutzerController** auf. Auf diese Weise können innerhalb der Controller-Klasse z.B. die Formulardaten verarbeitet und in eine Datenbank geschrieben werden. In der zweiten Variante wird beim Aufruf der URI **/start** eine View erzeugt, welcher ein Parameter („Name“) mit einem entsprechenden Wert („Max Mustermann“) übergeben wird. Die View ist eine normale PHP-Datei und liegt unter **app/views/hallo.php**. Sie enthält lediglich den in Abbildung 2 gezeigten Inhalt und der übergebene Parameter wird als gewöhnliche PHP-Variable angesprochen.

```
Route::post('/benutzer/anlegen','BenutzerController@
create');

Route::get('/start', function()
{
  return View::make('hallo')->with('name', 'Max
Mustermann');
});
```

Abb. 1: Beispiel einer `routes.php`.

```
<h1>Hallo <?= $name ?></h1>
```

Abb. 2: Beispiel einer `view`.

```
$alle = DB::table('benutzer')->get();

$allePB = DB::table('benutzer')->where('ort', 'Paderborn')-
>get();

$sort = DB::table('benutzer')->where('gehalt', '>',
99000)->orderBy('name', 'asc')->get();
```

Abb. 3: Sprechende Syntax für den Datenbankzugriff.

Datenbankzugriff

Für die Interaktion mit der Datenbank stellt Laravel seine Unterstützung in Form einer Klasse bereit, die den Zugriff auf die konfigurierte Datenbank mittels einer sehr sprechenden Syntax ermöglicht. Die Abbildung 3 zeigt drei Beispiele für die Abfrage von Daten aus der Tabelle **benutzer** mittels des sogenannten „Query Builder“. Im ersten Beispiel werden alle Datensätze dieser Tabelle selektiert, das zweite Beispiel begrenzt die Ergebnismenge auf die Benutzer aus Paderborn und im dritten Beispiel werden die „Hochverdiener“ nach ihrem Namen sortiert selektiert. Als Rückgabe erhalten wir jeweils ein Array bestehend aus **stdClass**-Objekten, welche als Attribute genau die Spalten aus der abgefragten Tabelle besitzen.

```
class Benutzer extends Eloquent {
   protected $table = 'benutzer';
   protected $primaryKey = 'lfd_nr';
}
```

Abb. 4: Beispiel einer Eloquent-Model-Klasse.

```
class Benutzer extends Eloquent {
   public function adresse() {
      return $this->hasOne('Adresse');
   }
}
```

Abb. 5: Beispiel einer Eloquent-Model-Klasse mit Relation zur Adresse-Tabelle.

```
echo Form::open(array('url' => 'email/anlegen', 'method'
=> 'post'));
echo Form::label('email', 'E-Mail-Adresse:');
echo Form::text('email', 'info@ordix.de');
echo Form::submit('Absenden');
echo Form::close();
```

Abb. 6: Ein Formular ohne HTML-Code.

Glossar

Composer
Der Composer ist ein Werkzeug zum Einbinden von Drittanbieter-Software in die eigene Anwendung und zum automatisierten Lösen von Abhängigkeiten der einzelnen Pakete.

MVC
Der Model View Controller ist ein Paradigma für Benutzeroberflächen, das die getrennte Behandlung von Daten, deren Darstellung und die darauf wirkenden Benutzeraktivitäten propagiert.

ORM
Das OR-Mapping ermöglicht es, den Zustand eines Objektes in einer relationalen Datenbank zu speichern und aus entsprechenden Datensätzen wiederum Objekte zu erzeugen.

Packagist
Packagist ist ein Online-Verzeichnis und eine Softwareablage, die automatisiert über den Composer eingebunden werden kann.

URI
Der Uniform Resource Identifier dient hauptsächlich der Identifikation einer Ressource, wie z.B. einer Webseite im Internet und besteht aus einer speziellen Zeichenfolge.

ORM

Wer als Rückgabeobjekte lieber selbst definierte und damit auch sprechende Namen bekommen möchte, sollte sich näher mit dem in Laravel integrierten ORM „Eloquent“ beschäftigen. Jede Tabelle in der Datenbank sollte durch eine entsprechende Model-Klasse repräsentiert werden und die Interaktion mit der Tabelle ermöglichen. In Abbildung 4 ist ein Beispiel für die Tabelle **`benutzer`** dargestellt. Zunächst muss die eigene Klasse **`Benutzer`** von der **`Eloquent`**-Klasse aus Laravel abgeleitet werden, damit dem Model die benötigten Methoden zur Verfügung stehen.

Das ORM versucht standardmäßig eine Verknüpfung zu einer Tabelle herzustellen, die dem Plural des Klassennamens entspricht. In diesem Fall: **`Benutzer`**. Um **`Eloquent`** hierbei unter die Arme zu greifen, kann über das Attribut **`$table`** der wirkliche Name der Tabelle hinterlegt werden. Zudem würde **`Eloquent`** den Primärschlüssel der Tabelle in der Spalte **`id`** vermuten, was durch Angabe des Attributs **`$primaryKey`** entsprechend verändert werden kann.

Über die folgenden Befehle können jetzt z.B. alle Einträge der Benutzertabelle ausgelesen werden oder nur der Benutzer mit der ID 5:

```
$alleBenutzer = Benutzer::all();
$benutzer = Benutzer::find(5);
```

Der große Vorteil bei der Benutzung eines ORM ist die Möglichkeit, die Relationen zwischen Tabellen innerhalb der Klassen mit abzubilden. Sie sehen in Abbildung 5 eine Erweiterung der Klasse **`Benutzer`**. Es wird hier eine Verbindung zur Tabelle **`Adressen`** hergestellt.

In diesem einfachen Beispiel besitzt ein Benutzer genau einen Datensatz. Nun kann, je nach Tabellenstruktur, z.B. über **`Benutzer::find(1)->adresse->strasse`** auf den Straßennamen des Adressdatensatzes des Benutzers mit der ID 1 zugegriffen werden.

In **`Eloquent`** stehen die bekannten Relationstypen One-to-One, One-to-Many und Many-to-Many zur Verfügung.

Formulare ohne HTML

Was wäre ein Framework, wenn es uns nicht bei der Erzeugung von HTML-Code unterstützen würde? Wir müssten uns stets selbst darum kümmern, validen Code zu schreiben und nur ja kein schließendes HTML-Tag übersehen. Laravel unterstützt den Programmierer sowohl bei der Erzeugung von korrekten Verlinkungen als auch der Erstellung ganzer Formulare und deren Eingabefelder.

Um den generierten HTML-Code muss sich der Entwickler also keine Gedanken machen. Die Abbildung 6 zeigt die Erstellung eines Formulars innerhalb einer View mit reinem Laravel-PHP-Code.

Fazit

Laravel überzeugt! Durch die einfache Installation mittels „Composer“ lässt sich sehr schnell eine neue Laravel-Anwendung aufsetzen. Die sprechende Syntax an allen Stellen macht die Entwicklung sehr einfach und den Code besonders gut lesbar. Es stehen viele nützliche Funktionalitäten bereit und selbstverständlich kann das Framework auch um eigene erweitert werden.

Das eingebaute ORM „Eloquent“ arbeitet schnell und die Benutzung ist sehr einfach. Ein Manko ist hierbei jedoch eine fehlende Transaktionssteuerung. Hierfür muss dann leider doch wieder der Weg über den „Query Builder“ gegangen werden.

Die Online-Dokumentation [3] ist für den Einstieg ausreichend und bietet viele Beispiele, die den Einstieg erleichtern. Bei entsprechend komplexer Programmlogik ist diese jedoch oft nicht ausreichend und es empfiehlt sich die Benutzung des Support-Forums [4]. Hier erhalten Sie oft sehr schnell eine zufriedenstellende Antwort auf Ihre Frage.

Links

- [1] https://getcomposer.org/
- [2] https://packagist.org/
- [3] Offizielle Website des Laravel-Framework: http://laravel.com/
- [4] Forum für Hilfe und Unterstützung bei der Entwicklung mit Laravel: http://laravel.io/

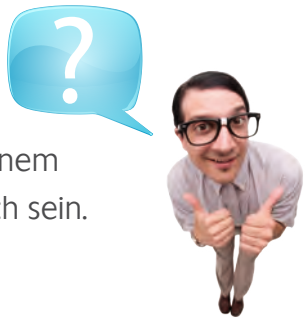
Bildnachweis

- © freepik.com | Free abstract geometric background

Lars Hendrik Korte
(info@ordix.de)

Rätsel

Larry ist in Eile



Larry ist ein wenig spät dran und es ist der erste Tag im neuen Projekt bei einem Kunden. Er möchte natürlich einen guten Eindruck hinterlassen und pünktlich sein.

Noch 5 Minuten Zeit

Er hat noch 5 Minuten, um sich im Büro des Projektleiters im 33. Stock zu melden. Als er endlich im Gebäude steht, sagt ihm der Herr am Empfang, dass es zwei Fahrstühle gibt. Der eine Fahrstuhl hält immer in jeder Etage, der andere dagegen fährt ohne Zwischenstopp hoch zur gewählten Etage. Beide Fahrstühle warten, doch welcher bringt ihn schnell nach oben?

Der Herr am Empfang hat einen Tipp für Larry: In beiden Fahrstühlen gibt es einen "Liftboy". Einer von beiden sagt immer die Wahrheit, der andere lügt immer.

Beide wissen natürlich, welcher der "Bummelzug" und welcher der schnelle Fahrstuhl ist.

Mit einer einzigen Frage an einen der beiden Liftboys kann Larry herausfinden, welchen er nehmen muss, um rechtzeitig oben zu sein. Welche Frage muss Larry stellen?

Larry freut sich auf Ihren Lösungsvorschlag!

Senden Sie Ihre Antwort bis zum 11. Juli 2014 an **kniffel@ordix.de**.

Die Lösung des letzten Rätsels und die Teilnahmebedingungen entnehmen Sie unserer Larry-Internetseite: **http://www.ordix.de/ordixnews/larry.html**

Das Rätsel aus der letzten Ausgabe wurde gelöst

Larry bedankt sich mit einer kleinen Aufmerksamkeit für die richtige Lösung bei Dirk Weil.

Neuerungen in der Oracle Database 12c (Teil IV)

Welche Veränderungen gibt es im Bereich Tuning?

Nachdem im dritten Teil dieser Reihe [3] die SQL-Erweiterungen in Oracle 12c vorgestellt wurden, zeigen wir Ihnen in diesem Artikel, was es Neues und Wissenswertes im Umfeld der Optimierung gibt.

Database Express

Das aus den Oracle Versionen 10g und 11g bekannte Werkzeug „Database Control" ist mit 12c nicht mehr verfügbar. Der Nachfolger heißt „Database Express". Im Unterschied zum Database Control benötigt Database Express keine Middleware mehr, sondern bedient sich nativ aus der XML-DB innerhalb der Datenbank. Beim Aufsetzen der Datenbank mit dbca wird Database Express direkt mit installiert und generiert.

Sollte man den Zugriff später konfigurieren wollen, so muss die **listener.ora** gepflegt werden:

```
dispatchers="(PROTOCOL=TCP)(SERVICE=<sid>XDB)
```

Anschließend muss innerhalb der Datenbank der Port konfiguriert werden, über den Database Express angesprochen werden soll:

```
exec DBMS_XDB_CONFIG.SETHTTPSPORT(5500);
```

Der Aufbau und die Menüführung sind ähnlich wie in Cloud Control. Über den Performance-Reiter sind all die Dinge möglich, die wir auch schon von Oracle 11g kannten. Allerdings muss dafür das Diagnostic Pack und teilweise auch das Tuning Pack lizensiert sein.

PGA

Bei vielen parallelen Leseaktionen konnte in der Vergangenheit das Problem auftreten, dass die PGA zu groß wurde und es im Extremfall zu einem Absturz der gesamten Instanz kam. Hier haben sich einige Datenbankadministratoren mit dem nicht dokumentierten Parameter **_PGA_MAX_SIZE** beholfen. Mit Oracle 12c gibt es nun aber auch eine offizielle Lösung.

Der Parameter **PGA_SIZE_LIMIT** begrenzt das ungebremste Wachstum der PGA. Wenn die definierte Maximalgröße der PGA erreicht ist, dann wird automatisch der Prozess entfernt, der momentan den größten Anteil an der PGA belegt. Ob dieses Vorgehen immer der gewollte Weg zur Begrenzung der PGA sein soll, ist sicher im Einzelfall abzuwägen.

Adaptive Pläne

Die statistischen Daten, die über **dbms_stats** gesammelt werden, können unter Umständen stark von den realen Verhältnissen abweichen, da sie immer einen historischen Blick darstellen. Bis Oracle 11g führte dies unter Umständen zu schlechten Ausführungsplänen. Dieser Missstand ist mit Oracle 12c durch die Einführung der adaptiven Pläne zumindest teilweise behoben worden.

Der Optimizer erstellt - wenn sinnvoll - unterschiedliche Varianten, wobei eine Variante den Default darstellt, die anderen Varianten werden alternativ im Hintergrund gehalten.

Mit einem Explain Plan können mit Hilfe des folgenden Aufrufs die unterschiedlichen Pläne angezeigt werden:

```
select * from table(dbms_xplan.
display(format=>'+adaptive'));
```

Wird das Statement anschließend gestartet, so erfasst der **statistic_collector** zunächst die tatsächlichen Größen der einzelnen Mengen und entscheidet dann, ob auf einen alternativen Plan gewechselt wird.

Statistiken

Die Komponente **dbms_stats**, mit der Statistiken erzeugt werden können, hat mit 12c diverse Erweiterungen erfahren.

Concurrent Statistics

Werden für mehrere große Tabellen oder große Partitionen neue Statistiken berechnet, so erfolgte dies bisher immer seriell. In Oracle 12c besteht nun die Möglichkeit, diese Aktionen zu parallelisieren. Dazu muss die Preference **CONCURRENT** gesetzt werden:

```
exec dbms_stats.set_global_
prefs('concurrent','all');
```

Extended Statistics

Die erweiterten Statistiken sind bereits mit Oracle 11g eingeführt worden. So sind die Kardinalitäten auf zwei

Spalten jeweils einzeln bekannt, aber die Selektivität in einer **AND**-Abfrage ist häufig nicht klar. Hier helfen die Extended Statistics weiter. In Oracle 11g musste man diese noch manuell erzeugen. In Oracle 12c kann man die Kandidaten hingegen erkennen und automatisch generieren lassen.

Dynamic Statistics

In der Regel werden Statistiken für alle Tabellen, deren Inhalt seit der letzten Statistik zu mehr als 10 % verändert worden sind, über Nacht im Autotask erzeugt. Nun kann es aber immer wieder vorkommen, dass sich Inhalte von Tabellen im Laufe des Tages enorm ändern und die statistischen Daten der Nacht schlichtweg falsch sind. Oracle bietet hier jetzt die Möglichkeit Dynamic Statistics zu erzeugen, also auf große Veränderungen dynamisch reagieren zu können.

Incremental Statistics

Die Laufzeit der Erzeugung von Statistiken für partitionierte Tabellen kann unter Umständen sehr lang sein. Mit Oracle 11g ist bei der Erstellung der globalen Statistiken schon eine Optimierung erreicht worden. Die globalen Statistiken wurden nicht mehr jeweils neu erstellt, sondern aus den Veränderungen der einzelnen Partitionen inkrementell abgeleitet (falls dies eingestellt ist). Mit Oracle 12c kann man nun auch einen Level setzen, ab wann die Statistik für die einzelne Partition neu berechnet wird. Die Aktualisierung mit den globalen Statistiken funktioniert auch, wenn die einzelne Partition noch keine neue Statistik erhalten hat.

Automatische Statistikerzeugung beim Bulk-Load

Seit Oracle 10g werden Statistiken automatisch beim **`create index`** und **`alter index rebuild`** generiert. Dieser Automatismus ist jetzt auch auf **`Bulk Loads`** übertragen worden. Diese Funktionalität greift allerdings nur, wenn das Segment vorher komplett leer war.

Multiple Indizes

Mit Oracle 12c ist es möglich, verschiedene Indizes auf dieselbe Spaltenliste zu legen. Das ermöglicht z.B. bei Applikationsmigrationen einen transparenten und übergangslosen Ablauf. Es darf allerdings nur immer ein Index sichtbar (visible) sein. Zugelassen sind folgende Konstellationen auf einer Spaltenliste:

- B-Tree-Index und Bitmap-Index
- verschiedene Partitionierungsstrategien
- Unique und Non-Unique Index

Very Large Network Buffers

Eine Unterstützung für sehr große Netzwerkpakete wurde in Oracle Net hinzugefügt. Die Standardgröße der Session Data Unit (SDU) Size beträgt 8 KB. In der Version 12c kann die SDU Size bis auf 2 MB erhöht werden. Bisher war die maximale Größe auf 64 KB begrenzt. Eine optimierte SDU Size kann zu einem verbesserten Anwendungsdurchsatz und einer besseren Nutzung der verfügbaren Netzwerkbandbreite führen.

Neues Prozessmodell unter Unix

Multi-Process Multi-Threaded Oracle ist ein optionales neues Ausführungsmodell für Unix/Linux, welches verwendet werden kann, um die Anzahl der Prozesse zu minimieren. Das Modell ist ähnlich dem unter Windows. Multi-Process Multi-Threaded Oracle verwendet mehrere Prozesse und mehrere Threads innerhalb eines Prozesses zur Ausführung. Im Threaded-Modus laufen einige Hintergrundprozesse wie PMON und DBWx weiter als Betriebssystemprozesse. Die anderen Oracle- und Client-Prozesse laufen als Threads innerhalb von einigen wenigen Betriebssystemprozessen. Als Folge ist ein Oracle-Prozess (**`v$process`**) nicht mehr zwingend gleich einem Betriebssystemprozess.

Resource Manager

Über den Resource Manager kann mit Oracle 12c jetzt auf SQL-Befehle reagiert werden, die „aus dem Ruder“ laufen. Dazu wird zunächst ein Schwellenwert definiert, der einen SQL-Befehl und damit eine Session als „runway“ deklariert. Gleichzeitig wird definiert, wie auf diesen Zustand reagiert werden soll. Wenn ein SQL-Befehl z.B. länger als 5 Sekunden läuft, dann soll in eine andere Consumer-Group gewechselt werden. Eine Session kann aber auch

Glossar

Cloud Control
Cloud Control ist die zentrale graphische Oberfläche für die Administration und das Monitoring beliebig vieler Oracle-Datenbanken.

Database Express
Database Express ist der integrale Bestandteil in Oracle 12c. Es ist die graphische Oberfläche für die Administration und das Monitoring und löst mit dieser Version Database Control ab.

Database Control
Database Control ist die graphische Oberfläche für genau eine Oracle-Datenbank in Oracle 10g und 11g. Diese Oberfläche ist für die Administration und das Monitoring bestimmt.

dbca
Der Database Configuration Assistent ist die graphische Oberfläche zum Erstellen von Datenbanken.

Optimizer
Der Optimizer ist eine interne Komponente zur Erstellung des Ausführungsplans.

PGA
Die Private Global Area wird u.a. für Sortierungen verwendet.

Resource Manager
Der Resource Manager ist eine interne Komponente zur Priorisierung der Oracle-Prozesse.

beendet werden, wenn ein einzelner SQL-Befehl z.B. länger als eine Minute läuft.

ADDM

Automatic Database Diagnostic Monitoring (ADDM) ist mit Oracle 10g eingeführt worden. Das Werkzeug soll dem Datenbankadministrator grobe Hinweise geben, welche Optimierungen möglich sind. Zu den bekannten drei Komponenten ist jetzt eine vierte hinzugekommen:

- im Zuge des stündlichen AWR Snapshot
- Real-Time ADDM
- Compare ADDM
- Spot ADDM (neu in Oracle 12c)

Spot ADDM erfasst interne Probleme in Echtzeit und schreibt die wichtigsten Informationen direkt in das AWR. Somit können auch Spitzen besser erkannt werden.

Fazit

In Oracle 12c gibt es im Bereich der Optimierung einige nützliche Erweiterungen. Besonders möchten wir die adaptiven Pläne hervorheben, die in ersten Tests sehr vielversprechende Ergebnisse geliefert haben.

In einem weiteren Artikel dieser Reihe werden wir uns mit den Neuerungen rund um das Thema Data Warehousing beschäftigen.

Klaus Reimers
(info@ordix.de)

Links

- [1] ORDIX® news Artikel 3/2013
  „Neuerungen in Oracle 12c (Teil I) - Eine für alle, alle in einer“:
  http://www.ordix.de/ordixnews/ordix-news-archiv/3-2013.html
- [2] ORDIX® news Artikel 4/2013 „Neuerungen in Oracle 12c (Teil II) - Multitenant-Architektur - eine Datenreise durch Raum und Zeit“:
  http://www.ordix.de/ordixnews/ordix-news-archiv/4-2013.html
- [3] ORDIX® news Artikel 1/2014
  „Neuerungen in Oracle 12c (Teil III) - SQL-Neuerungen in Oracle 12c“:
  http://www.ordix.de/ordixnews/ordix-news-archiv/12014.html

Replikation nach Oracle-Art

Oracle GoldenGate 12c - New Features

Mit der neuen Version von GoldenGate und der Datenbankversion 12c geht Oracle den klaren Weg, GoldenGate als alleiniges Oracle Replikationswerkzeug zu etablieren. Dieser Artikel stellt die Neuerungen dieser Version vor und beleuchtet sie genauer.

Streams ist Geschichte

Oracle Streams ist seit Datenbankversion 12c abgekündigt (deprecated). Schon seit der Version 11.2 baut Oracle Bausteine aus Streams in Oracle GoldenGate (OGG) ein. So ist es das erklärte Ziel, die jeweils besten Funktionen aus GoldenGate und Oracle Streams im neuen GoldenGate zu vereinen. Da Streams wesentlich verbreiteter ist als GoldenGate folgt eine kurze Vorstellung, welche Einsatzgebiete OGG abdeckt und für welche Datenbanken es zur Verfügung steht. Denn im Unterschied zu Streams ist mit OGG auch eine übergreifende Anwendung über mehrere Datenbankmanagementsysteme im heterogenen Umfeld möglich.

Anwendungszenarien

Die folgenden Anwendungsszenarien lassen sich mit OGG verwirklichen, wobei der Aufwand stark von der Anwendungsart und deren Komplexität abhängt:

- Minimierung der Downtime
  - Datenmigration
  - Datenbank-Upgrade (Zero Downtime Migration)
- Plattformübergreifende Migration
- Operational Reporting
- Extract, Transformation und Load (ETL)
  (Real Time Data Warehousing)
- Disaster-Recovery-Datenbank,
  auch betriebssystemübergreifend
- Hochverfügbarkeit Active/Active

Ein Standardablauf ist in Abbildung 1 zu sehen.

Hierzu gehören die folgenden Prozesse/Dateien:

- **Extract**: Auslesen der Daten aus der Quelldatenbank
- **Datapump**: Transferieren der Daten auf das Zielsystem
- **Replicat**: Replizieren der Daten auf die Zieldatenbank
- **Manager**: Steuern und starten von anderen Prozessen
- **Trail Files**: Dateien zum Zwischenspeichern der Daten

Die Replikation mit GoldenGate lässt sich mit vielen Datenbankmanagementsystemen realisieren, je nach System muss die richtige zertifizierte OGG-Version ausgewählt werden. Aktuell werden die folgenden Datenbanken unterstützt:

Oracle, SQL Server, MySQL, Sybase, DB2, HP NonStop SQL/MX, PostgreSQL, Teradata

Integrated Capture

Im klassischen GoldenGate Capture-Modus liest der Extract-Prozess direkt aus den Redo- bzw. Archivelogs (siehe Abbildung 2). Der Modus Integrated Capture nutzt dagegen den Database Log Mining Server, welcher die Datenänderungen in Form von Logical Change Records (LCR) liefert. Es werden somit vornehmlich Prozesse innerhalb der Datenbank genutzt und der Extract-Prozess muss nicht selbst auf die Redolog-Dateien zugreifen. Da das Auslesen der Log-Dateien direkt in der Datenbank stattfindet, müssen keine gesonderten Konfigurationen für RAC, TDE oder auch ASM gepflegt werden.

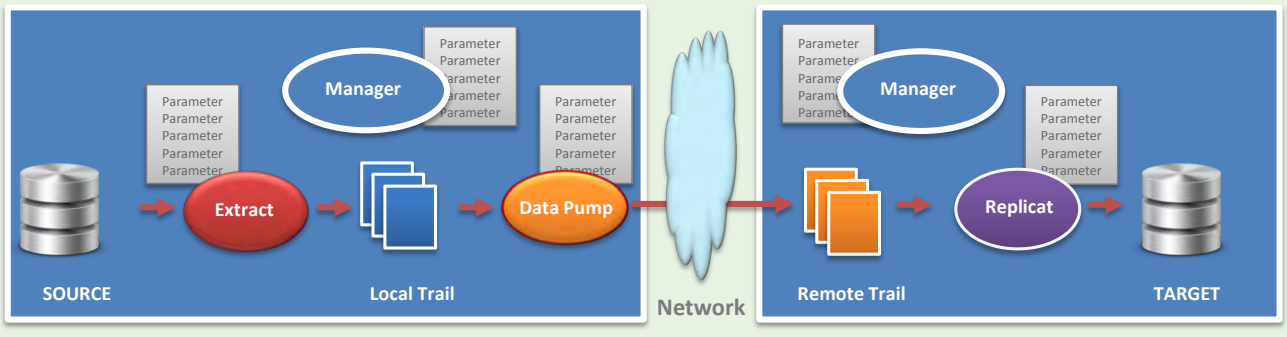


Abb. 1: Standardablauf eines Anwendungsszenariums.

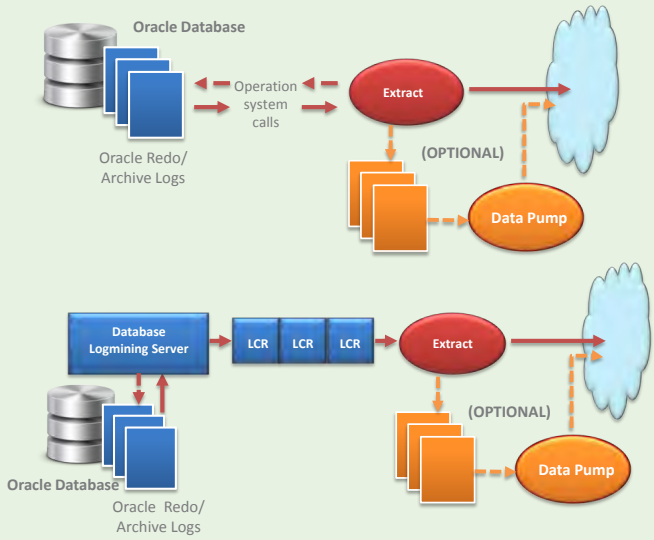


Abb. 2: Classic Capture (oben) vs. Integrated Capture (unten) (Quelle [2]).

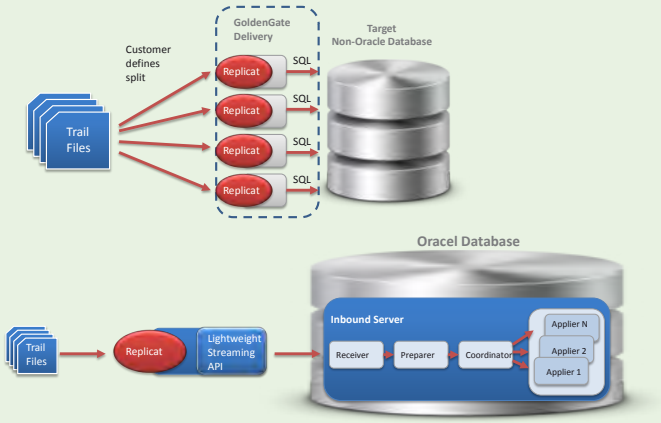


Abb. 3: Classic Delivery (oben) vs. Integrated Delivery (unten) (Quelle [1]).

```
ggsci> CREATE WALLET
Created wallet at location 'dirwlt'
Openend wallet at location 'dirwlt'
```

Ein Schlüssel wird wie folgt hinzuzufügt:

```
ggsci> ADD MASTERKEY
Master key 'OGG_DEFAULT_MASTERKEY' added to wallet at location
'dirwlt'
Informationen zum Wallet:
ggsci> INFO MASTERKEY
Masterkey Name:  OGG_DEFAULT_MASTERKEY
Creation Date:   Wed Mar 5 16:27:23 2014
Version:  Creation Date:             Status:
1         Wed Mar 5 16:27:23 2014    Current
```

Abb. 4: Anlage eines Wallet im Standardverzeichnis.

Um den Extract-Prozess der Datenbank bekannt zu machen und somit das Log Mining zu starten, muss man sich mit ggsci (GoldenGate Software Command Interface) zunächst mit der Datenbank verbinden:

```
ggsci> DBLOGIN USERID gg1_admin PASSWORD gg1_password
```

Nun wird der Extract in der Datenbank registriert:

```
ggsci> REGISTER EXTRACT extr01 DATABASE
```

Als nächster Schritt wird der Extract in OGG konfiguriert. Hier wird das Integrated Capture mit dem Schlüsselwort „Integrated Tranlog“ bekannt gemacht:

```
ggsci> ADD EXTRACT extr1 INTEGRATED TRANLOG,
BEGIN NOW
```

Im Anschluss wird der Trail-Pfad für den Extract bereitgestellt:

```
ggsci> ADD RMTTRAIL /oracle/trail_files/lt,
extract extr1
```

Zum Abschluss wird die Integrated-Capture-Konfiguration der Parameterdatei hinzugefügt:

```
edit params extr1
EXTRACT extr1
USERID gg1_admin, PASSWORD gg1_password
TRANLOGOPTIONS INTEGRATEDPARAMS (MAX_SGA_SIZE 100)
RMTTRAIL /oracle/trail_files/lt
TABLE ora01.mitarbeiter;
```

Nun kann der Extract gestartet werden:

```
ggsci>start extract extr1
```

Integrated Delivery

Das Aufteilen von Workloads auf mehrere Prozesse stellte bei großen Datenmengen bis einschließlich GoldenGate 11.2 einen großen Aufwand dar. So musste je nach Anzahl der gewünschten parallelen Prozesse jeweils ein eigenständiger Replicat-Prozess konfiguriert werden. Dem Replicat musste in der Mapping-Klausel mit Hilfe der Funktion **@RANGE** die gesamte Anzahl der parallelen Replicat-Prozesse und die aktuelle Prozessnummer (Range) übergeben werden (siehe Abbildung 3).

Mit Oracle GoldenGate 12 wird dieser Vorgang enorm vereinfacht. So muss zum Splitten im Replicat nur ein Parameter gesetzt werden, dem die Anzahl der gewünschten Prozesse übergeben wird:

```
DBOPTIONS INTEGRATEDPARAMS(parallelism 6)
```

Der Replicat-Prozess erstellt aus den Trail-Dateien sogenannte Logical Change Records (LCR), die mit Hilfe der Lightweight Streaming API zur Datenbank übertragen werden. Die Dateien werden in der Datenbank mit Hilfe einiger Prozesse verarbeitet:

- **Receiver** liest die LCRs
- **Preparer**
  - prüft die Abhängigkeiten der einzelnen Transaktionen mit Hilfe von Unique-Indizes, Primär- und Fremdschlüsseln
  - führt einzelne Transaktion unter Berücksichtigung der Abhängigkeiten zu Gruppen zusammen
- **Coordinator** verwaltet die gruppierten Transaktionen und gibt sie an den Applier weiter

- **Applier** vollzieht alle Änderungen auf der Datenbank

Um diese Funktionen nutzen zu können, ist eine Oracle-Datenbank ab der Version 11.2.0.4 notwendig.

Neue Sicherheitsfunktionen

Credential Store

Bisher wurden die Passwörter in den Parameterdateien als Klartext oder verschlüsselt dargestellt. Mit dem Credential Store bietet GoldenGate nun die Möglichkeit, in Dateien komplett auf die Angabe von Username-Passwort-Kombinationen zu verzichten. Hierzu muss zunächst ein Credential Store angelegt werden: **`ggsci> ADD CREDENTIALSTORE`**

Damit wird standardmäßig im GoldenGate-Installationsverzeichnis unter **`dircrd`** eine Datei mit dem Namen **`cwallet.sso`** angelegt. Um Zugangsdaten zu hinterlegen, ist folgender Befehl notwendig:

```
ggsci>ALTER CREDENTIALSTORE ADD USER gg1_admin
password gg1_admin ALIAS gg_alias
```

Der Alias ist optional und kann beispielsweise dazu dienen, den Datenbanknutzer für den GoldenGate Nutzer geheim zu halten. Ein Aufruf könnte beim Datenbank-Login vom ggsci beispielsweise wie folgt aussehen:

```
ggsci> DBLOGIN USERIDALIAS gg_alias
```

Wird kein Alias angeben, so ist der Username zu verwenden.

Wallet

Bislang war es mit der sogenannten **`ENCKEYS`**-Methode recht aufwendig, ein Verschlüsselungsverfahren für OGG Trails aufzubauen. Mit dem neuen GoldenGate Wallet wurde dies zum Teil vereinfacht. Mit dem Wallet werden Schlüssel durch OGG erstellt und verwaltet. Es gibt zwei grundsätzliche Möglichkeiten, wo die Wallet-Datei angelegt wird: Standardmäßig im Ordner **`dirwlt`** des Installationsverzeichnisses von GoldenGate oder in einem frei wählbaren Ordner. Hierzu muss in der **`GLOBALS`**-Konfigurationsdatei dem Parameter **`WALLETLOCATION`** ein Pfad angegeben werden. Wie ein Wallet im Standardverzeichnis angelegt wird, zeigen wir in Abbildung 4.

Um ein Wallet sinnvoll nutzen zu können, wird der Inhalt des **`dirwlt`**-Verzeichnis auf alle Zielsysteme verteilt. Nun lässt sich der Chiffrierungsschlüssel nutzen, um Daten verschlüsselt zu übertragen. Auszug aus einer Beispiel-Extract-Parameterdatei:

```
...
ENCRYPTTRAIL AES256
EXTTRAIL /gg_home/dirdat/ee
...
```

Zum Entschlüsseln in der Replicat-Parameterdatei:

```
...
DECRYPTTRAIL AES256
MAP ora00.*, ora01.*
```

Glossar

ASM
Das Automatic Storage Management ist ein im Oracle-Kernel integrierter Storage/Volume Manager zur physikalischen Speicherverwaltung von Oracle-Datenbanken.

Enckeys
Encryption Keys sind Methoden zu Erstellung und Nutzen von Chiffrierungsschlüsseln.

ggsci
Das GoldenGate Software Command Interface ist ein Kommandozeilenwerkzeug zur Administration der Prozesse von Oracle GoldenGate.

LCR
Ein Logical Change Record ist eine Nachricht in einem speziellen Format, welche eine Datenbankänderung beschreibt.

OGG
Oracle GoldenGate ist die Datenbank-Replikationssoftware.

RAC
Innerhalb eines Real Application Cluster greifen mehrere Knoten eines Cluster auf dieselbe Datenbank zu, um die Ausfallsicherheit zu erhöhen und Skalierbarkeit zu erreichen.

TDE
Die Transparent Data Encryption ist eine transparente Verschlüsselung von Datenbankdateien, Data Pump Exports und RMAN Backups. Sie ist verfügbar in der kostenpflichtigen Advanced Security Option der Enterprise Edition.

Links

- [1] ORDIX® news Artikel 2/2011:
  „Oracle GoldenGate - Über eine Brücke musst du gehen!“
  http://www.ordix.de/images/ordix/onews_archiv/2_2011/oracle_goldengate.html
- [2] ORDIX® news Artikel 1/2012:
  „Oracle GoldenGate - Datenintegration in heterogenen Datenbanklandschaften (Teil II) - Jetzt im Angebot: Heterogener Initial Load“
  http://www.ordix.de/images/ordix/onews_archiv/1_2012/oracle_goldengate.html
- [3] Oracle GoldenGate Dokumentation:
  http://docs.oracle.com/goldengate/1212/gg-winux/index.html

Quelle/Bildnachweis

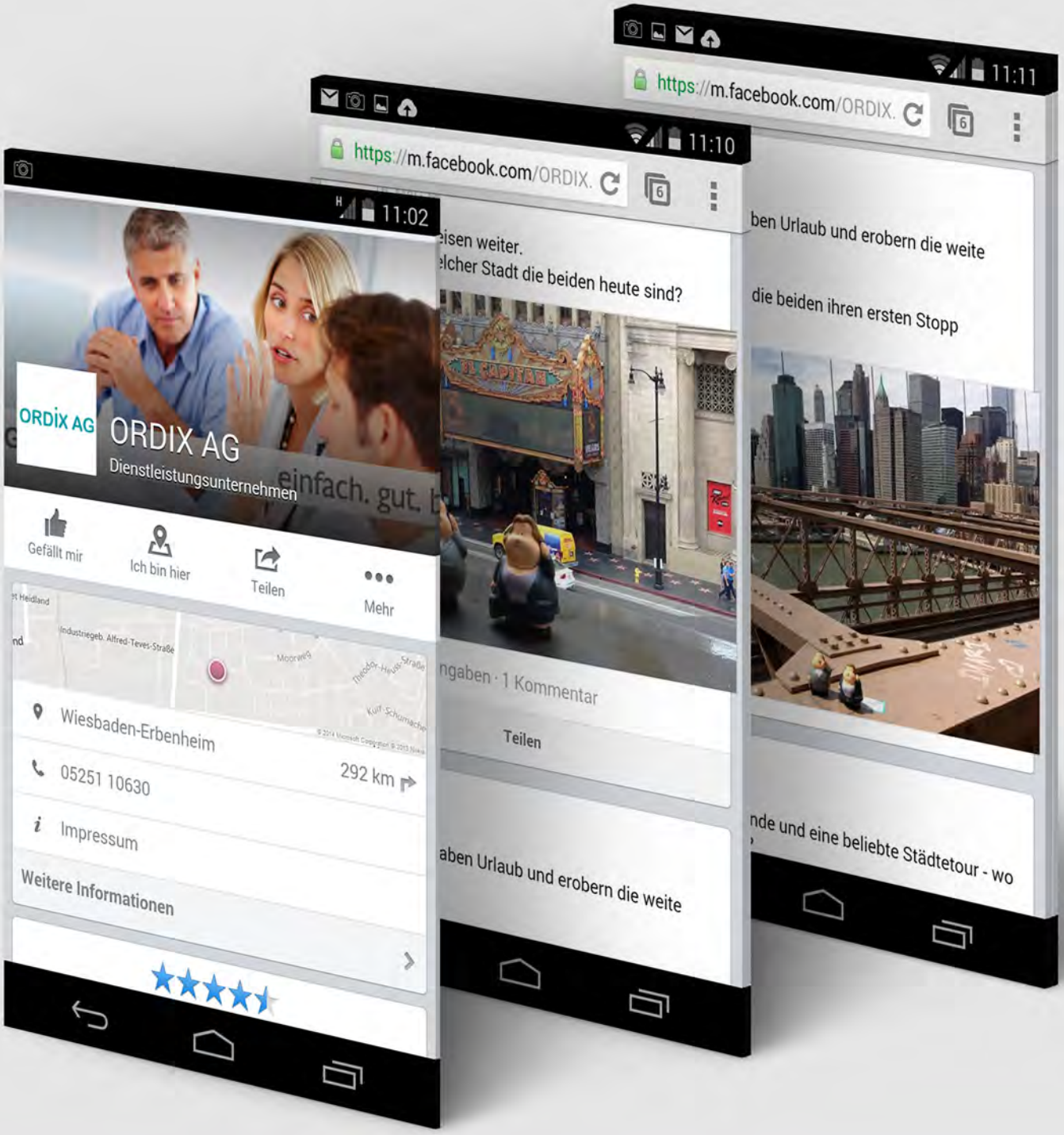
- [1] Oracle White Paper: Oracle GoldenGate 12c Release 1:
  http://www.oracle.com/us/products/middleware/data-integration/oracle-goldengate-features-wp-2030476.pdf
- [2] Oracle GoldenGate 12c Documentation:
  Installing and Configuring Oracle GoldenGate for Oracle Database:
  http://docs.oracle.com/goldengate/1212/gg-winux/GIORA/

Fazit

In den letzten Jahren wurde Oracle GoldenGate kontinuierlich weiterentwickelt. Mit den neuen Funktionen wird sowohl eine einfachere Handhabung als auch eine Performance-Verbesserung erreicht. Die neuen Sicherheitsfunktionen runden den guten Gesamteindruck ab. Eine Funktion, welche viele GoldenGate-Anwender allerdings immer noch vermissen werden, ist die Möglichkeit ganze Datenbanken ohne großen Aufwand zu replizieren.

André Stefaniak
(info@ordix.de)

SIND SIE AUCH IM NETZWERK UNTERWEGS? BESUCHEN SIE UNS!



Werden Sie Fan

und erfahren als erster
von neuen Artikeln, Rätseln
und Veranstaltungen der ORDIX AG

einfach. gut. informiert.

Auf unserem Facebook-Profil haben wir viele nützliche Themen für Sie zusammengestellt. Besuchen Sie uns unter:

https://www.facebook.com/ORDIX.AG