Intelligentes Monitoring: Implementierung von Overflow- und Runtime-Prognosen

Dr. Thomas Petrik, Sphinx IT Consulting

Ziel eines intelligenten Monitorings ist es, einige wenige KPIs zu definieren, die einerseits leicht zu interpretieren sind und andererseits die für den Betrieb einer Datenbank wesentlichen Fragen beantworten: Ist die Instanz respektive die Datenbank verfügbar? Ist eine stabile Response-Zeit gewährleistet? Welche Vorsorge ist zu treffen, um Storage-Engpässe zu vermeiden?

Der Schlüssel zum intelligenten Monitoring liegt zum einen in der Betrachtung zeitlicher Entwicklungen und zum anderen in der statistisch korrekten Interpretation der gesammelten Daten. Die weit verbreitete alleinige Verwendung statischer Metriken im Monitoring wie Füllgrade von Tablespaces oder die Auslastung der Flash-Recovery-Area ist definitiv als unzureichend zu betrachten und wird durch zeitabhängige Methoden nahezu vollständig ersetzt.

Statistische Grundlagen für Overflow-Prognosen

Hat man sich erst einmal eine historische Datensammlung angelegt (wie den Platzverbrauch pro Tablespace, pro Filesystem, pro Maschine oder auch für ein ganzes Storage-Tier im zentralen Storage-Pool), so lassen sich diese Messpunkte in einem Koordinatensystem auf der y-Achse gegen die Zeit auftragen. Die Optik eines derartigen Diagramms suggeriert meist bereits einen funktionalen Zusammenhang zwischen Messwert und Zeit, den es schließlich auch zu finden gilt. Im Idealfall hat man es mit annähernd linearem Verhalten zu tun, aber auch andere Abhängigkeiten kommen vor und müssen ebenso behandelt werden. Beim Betrachten von *Abbildung 1* ergeben sich zwei mögliche Fragestellungen:

1. Wann wird ein vorgegebener Maximalwert erreicht (oder anders gefragt: Wann geht der Tablespace über)?
2. Welcher y-Wert wird zu einem bestimmten Zeitpunkt erreicht sein (oder: Wie hoch ist der Platzbedarf in einigen Monaten)?

Während die erste Form der Fragestellung klassischerweise die eines DBAs ist, der einen Overflow verhindern muss, betrifft der Fokus der zweiten Fragestellung die Planung.

Um zu diesen Aussagen zu gelangen, wendet man die Methoden der Regression und Extrapolation an und beurteilt den gefundenen funktionalen Zusammenhang durch Korrelations-Kennzahlen. Wie spätere Betrachtungen zeigen werden, kommt man mit dem Spezialfall der linearen Regression aus – selbst dann, wenn ein System kein durchgängig lineares Verhalten zeigt.

Lineare Regression

Um also eine geeignete Regressionsfunktion zu finden, betrachten wir die Summe

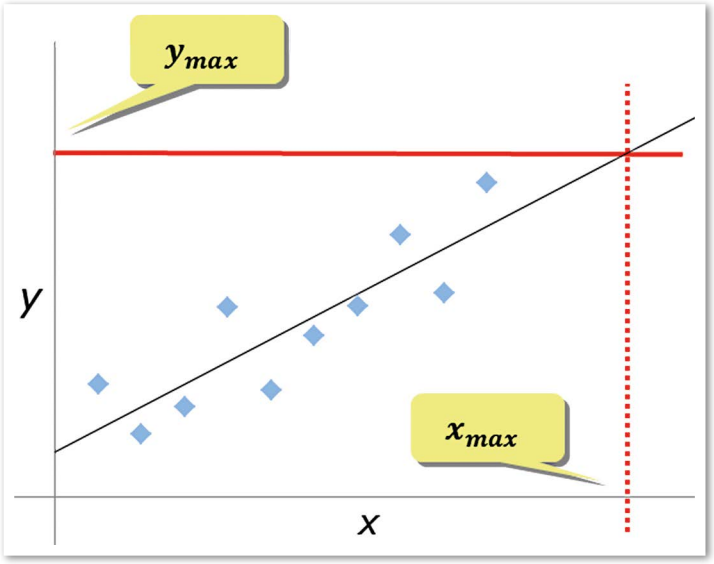

Abbildung 1: Beispiel für annähernd lineare Entwicklung

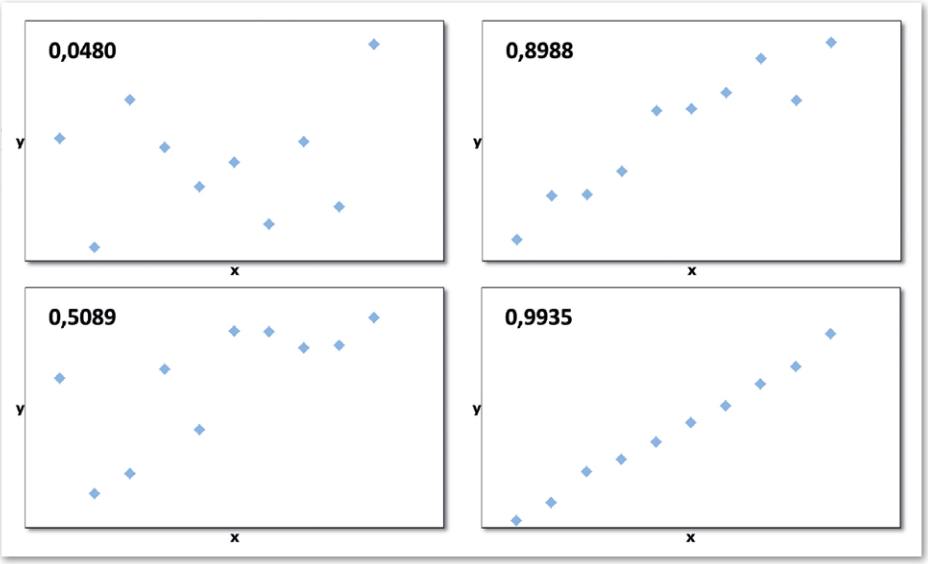

Abbildung 2: r^2 für verschiedene Verteilungen

Diese Aufgabe ist durch einfache Rechnung leicht lösbar (interessierte Leser mögen die genaue Ableitung der Standardliteratur entnehmen) und führt zu den Formeln für Ordinatenabschnitt (Intercept) und Steigung (Slope). Bemerkenswert an diesen Formeln ist die Tatsache, dass die ganze Berechnung lediglich mit Mittelwertbildungen auskommt. Im Grunde ist also die AVG-Funktion, die in allen SQL-Dialekten anzutreffen ist, bereits ausreichend, um derartige Berechnungen via SQL durchzuführen.

Oracle (und mittlerweile ebenso die meisten anderen Datenbanken) stellt für die lineare Regression ein eigenes Set an Funktionen mit dem Präfix „REGR_" zur Verfügung. „REGR_SLOPE(y,x)" berechnet die Steigung, „REGR_INTERCEPT(y,x)" den Ordinatenabschnitt (die erste Expression ist stets der abhängige Wert, im vorliegenden Beispiel also y).

Verteilung und Korrelation

Für eine automatisierte Beurteilung, ob die Grundannahme des linearen Verhaltens gerechtfertigt ist, empfiehlt sich die Berechnung des Korrelations-Koeffizienten r beziehungsweise von r^2, dem sogenannten Bestimmtheitsmaß.

Der Korrelationskoeffizient lässt sich somit auf die Varianzen der x- und y-Werte s_x^2 und s_y^2 beziehungsweise auf die Kovarianz s_{xy} zurückführen. Die dafür vorgesehenen SQL-Funktionen sind „VAR_POP" und „COVAR_POP", aber vor allem „CORR(x,y)" beziehungsweise „REGR_R2(x,y)" für die direkte Berechnung von r^2.

r^2 kann Werte zwischen 0 und 1 annehmen, wobei bei einem Wert von 1 eine perfekte Gerade vorliegen würde und bei einem Wert von 0 hingegen keinerlei Korrelation bemerkbar wäre. Für die Werte zwischen 0 und 1 gilt, dass die Korrelation besser ist, je näher der Wert bei 1 liegt. Empirische Untersuchungen im Bereich der Storage Forecasts haben gezeigt, dass man ab einem Wert von 0,8 berechtigterweise von einem linearen Zusammenhang ausgehen kann; Regressionen mit kleineren Korrelationskoeffizienten sind zu verwerfen. Es muss allerdings betont werden, dass es sich hierbei um einen langjährigen Erfahrungswert handelt, der allenfalls in der Praxis nachjustiert werden sollte.

Abbildung 2 zeigt den Zusammenhang zwischen verschiedenen Verteilungen und dem Bestimmtheitsmaß. In der Literatur sind immer wieder Beispiele für extreme Verteilungen anzutreffen, die zu irreführenden (zu hohen) Werten für r^2 führen, für die gegenständlichen Betrachtungen allerdings nur eine untergeordnete Rolle spielen.

Nicht-lineares Verhalten

Storage-Wachstum erfolgt selten streng linear, aber dennoch ist es nicht erforderlich, auf nicht-lineare (polynomische, exponentielle, logarithmische etc.) Regressionen auszuweichen. Es genügt, die Zeitbasis in mehrere Intervalle (ausgehend vom aktuellen Datum) zu unterteilen und für jedes dieser Intervalle eine eigene Regressionsgerade zu rechnen (mehrfache lineare Regression). Beispielsweise bewähren sich für das Tablespace-Monitoring Berechnungen auf der Basis von 3, 9, 27 und 81 Tagen.

Abbildung 3 zeigt ein typisches Verhalten eines Tablespace, in dem periodisch Löschungen vorgenommen werden. Trotzdem lässt sich daraus ein klares Kurzzeit- (Basis drei Tage) und ein Langzeit-Verhalten (alle Tage) ableiten. Prognosen auf einer Basis zwischen diesen beiden Voraussagen mussten im konkreten Fall verworfen werden aufgrund der viel zu schlechten Korrelation.

Extrapolation

Mithilfe der Regressionsgeraden und des gegebenen oberen Limits für die y-Werte (zum Beispiel die maximale Größe des Tablespace) ist es nun ein Leichtes, den

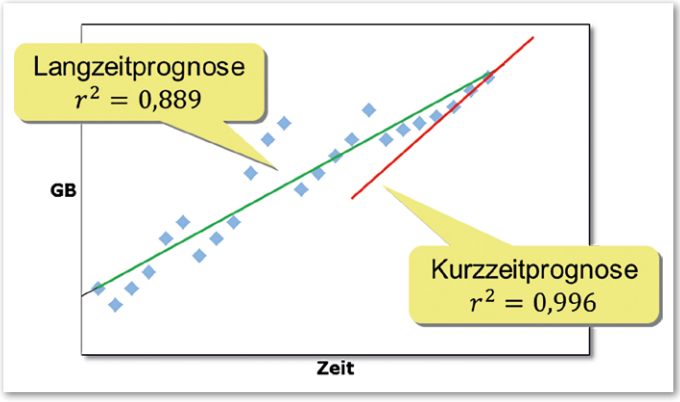

Abbildung 3: Mehrfache lineare Regression

Overflow-Zeitpunkt zu berechnen. Ausgehend von *Abbildung 1* ergibt sich durch einfache Rechnung

$$x_{max} = \frac{y_{max} - a}{b}$$

Steht also in einer Tabelle „REGR1" der Wertevorrat für x und y zur Verfügung und ist der maximale y-Wert beispielsweise mit „10" vorgegeben, so erhält man das dazugehörige x_{max} durch das SQL-Statement in *Listing 1*.

Tablespace Monitoring

Für das Tablespace Monitoring muss zunächst der verbrauchte Platz pro Tablespace periodisch aus „dba_free_space" ermittelt werden. Dies geschieht sinnvollerweise täglich − bei Bedarf auch in kürzeren Intervallen, was allerdings die Datenbank-Last erhöht −, wobei darauf zu achten ist, dass der absolute Verbrauch und nicht der Füllgrad in Prozent zur Auswertung herangezogen wird, da der absolute Space-Usage-Wert invariant gegen Tablespace-Vergrößerungen ist. Mit einer einfachen History-Tabelle wie in *Listing 2* könnte das erforderliche Insert/Select-Statement wie in *Listing 3* aussehen.

Listing 4 zeigt, wie man mit einem einzigen Select-Statement auf ein konkretes Overflow-Datum kommt − in diesem Beispiel auf der Basis von drei Tagen, wobei die aktuelle Tablespace-Größe aus „dba_data_files" ausgelesen wird. Da die Zeitachse in Form von Datums-Zeitpunkten gespeichert wurde, müssen diese Werte zunächst mithilfe eines beliebigen Bezugspunkts („sysdate") in eine Zeitdauer umgewandelt werden. Durch die Addition von „sysdate" im Output ergibt sich dann wiederum ein Overflow-Datum.

Ein tägliches Reporting sollte dem DBA alle jene Tablespaces auflisten, die aufgrund einer der Prognosen (Langzeit bis Kurzzeit) einen Overflow innerhalb einer definierten Frist vermuten lassen. Dank der ebenfalls aufgelisteten Prognosewerte (mit zugehöriger Basis) kann ein mögliches Fehlverhalten von Applikationen rasch erkannt werden, denn exponentielles Wachstum spiegelt sich beispielsweise in zunehmend kurzfristigeren Overflow-Prognosen bei abnehmender Zahl an Basistagen wider. Für andere Applikationen kann eine vorausschauende Planung durchgeführt werden, die auch kurzfristige Spitzen berücksichtigt.

Die Auswertung des momentanen Füllgrads erweist sich als überflüssig.

Systeme, die in keinem Intervall ausreichendes lineares Verhalten zeigen, entziehen sich naturgemäß jeglicher Prognose. Dies ist im Speziellen bei Entwicklungs-Systemen zu beobachten. Produktions-Systeme, die ein derartiges Verhalten zeigen, sollten dringend hinterfragt werden. Ebenfalls nicht detektierbar mit dieser Methode sind Ad-hoc-Zuwächse (durch Beladungen, Importe, Upgrades) innerhalb des Beobachtungs-Intervalls.

Overflow-Prognose für die Flash Recovery Area

Mit Einführung der Flash Recovery Area (FRA) hat Oracle die automatische Selbstverwaltung der Datenbank weiter vorangetrieben. Archivelogs (und allenfalls Flashbacklogs) müssen nicht manuell oder beim Backup gelöscht werden, sondern die Datenbank kümmert sich selbst darum. Leider gibt es bis heute keinen Automatismus in der Datenbank selbst, der ein RMAN-Archivelog-Backup anstoßen würde, wenn dies aufgrund des Füllgrads der FRA nötig wäre. Daher ist es erforderlich, die FRA analog zu den Tablespaces zu monitoren und bei Bedarf ein Backup-Skript zu starten.

Im Unterschied zur Tablespace-Prognose interessiert man sich in diesem Fall in der Regel nur für das Kurzzeit-Verhalten, um einem drohenden Overflow rasch durch den Start des Backups begegnen zu können. Daraus ergeben sich ein kurzes Sampling-Intervall von beispielsweise fünf Minuten und eine Regressionsbasis von etwa einer Stunde. Auf eine mehrfache li-

```
SELECT 10 - REGR_INTERCEPT(y,x) /
REGR_SLOPE(y,x) as xmax
   FROM regr1;
```

Listing 1

```
CREATE TABLE space_history
(
      log_date       DATE
   ,tablespace_name  VARCHAR2(30)
   ,bytes_used       NUMBER
);
```

Listing 2

```
INSERT INTO space_history
   SELECT sysdate, tablespace_name, SUM (bytes) AS bytes
      FROM dba_free_space
GROUP BY tablespace_name;
```

Listing 3

```
   SELECT tablespace_name
          ,COUNT(1) AS base
          ,SYSDATE + (max_bytes - REGR_INTERCEPT(bytes_used,days)) /
          REGR_SLOPE(bytes_used,days) AS overflow_date
          ,REGR_R2 (bytes_used, days) AS r2
      FROM (SELECT SYSDATE - a.log_date AS days
              ,a.bytes_used
              ,b.max_bytes
              ,a.tablespace_name
          FROM space_history a
              ,( SELECT tablespace_name, SUM (bytes) AS max_bytes
                  FROM dba_data_files
              GROUP BY tablespace_name) b
          WHERE a.tablespace_name = b.tablespace_name)
   WHERE days <= 3
GROUP BY tablespace_name, max_bytes;
```

Listing 4

neare Regression kann in diesem Fall gut verzichtet werden.

Die Daten für das Monitoring stellt Oracle über die View „v$flash_recovery_area_usage“ zur Verfügung, wobei zwei Prozent-Werte für die Datensammlung zu ermitteln sind:

3. % Available Space = 100 – percent_space_used – percent_space_reclaimable
4. Anteil der noch nicht gesicherten Archivelogs aus percent_space_used – percent_space_reclaimable

Es ist in diesem Fall zulässig, mit Prozentwerten zu agieren, da man im Beobachtungsintervall in der Regel nicht mit einer Größenveränderung der FRA rechnen muss. Will man dennoch mit absoluten Werten rechnen, ist die Gesamtgröße der FRA aus dem Parameter „db_recovery_file_dest_size“ heranzuziehen.

Die Extrapolation auf 100 Prozent ergibt eine Abschätzung für den Overflow-Zeitpunkt, der Füllgrad kann zusätzlich als statische Metrik verwendet werden, um eine bestimmte Reserve an freiem Platz nicht zu unterschreiten. Das RMAN-Skript selbst kann über einen „DBMS_SCHEDULER“-Job gestartet werden. Wenn der Anteil an noch ungesicherten Archivelogs zehn Prozent unterschreitet, ist ein Backup kaum mehr sinnvoll. In diesem Fall muss es zu einem Alert kommen, weil die FRA offenbar zu klein dimensioniert ist.

Runtime Forecasts (RTFC)

Bei der Voraussage von Laufzeiten von Batch-Prozessen geht es nicht nur darum zu wissen, wann der Prozess fertig sein wird. Mindestens ebenso wichtig ist das rasche Erkennen abnormen Verhaltens, nicht reproduzierbarer Laufzeiten und Ausreißern (vor allem nach oben).

Die Klassifikation der Daten im Sinne einer Auftrennung nach zu erwartender Laufzeit ist eine Grundvoraussetzung. *Abbildung 4* zeigt ein typisches Verhalten eines Prozesses, der an bestimmten Tagen aufgrund einer kalenderabhängigen Verarbeitungslogik (Ultimoverarbeitung, Quartalsabschluss, etc.) eine andere Laufzeit aufweist als normalerweise. In diesem Fall ergeben sich also zwei RTFCs für unterschiedliche Tagestypen. Der gezeigte Ausreißer muss in jedem Fall ignoriert werden.

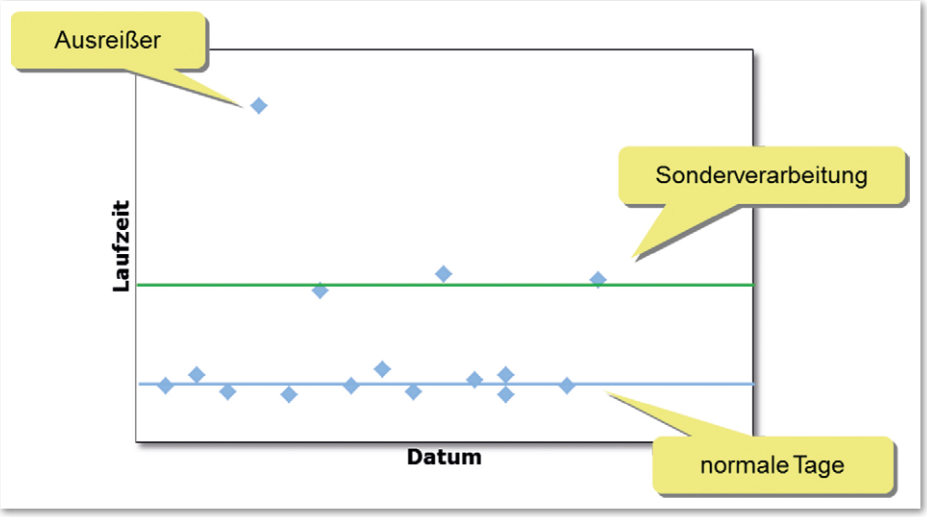

Abbildung 4: Typisches Verhalten eines Batch-Prozesses

Die einfache Mittelwertbildung ist in jedem Fall zu wenig, um zu einer Voraussage zu kommen. Nach erfolgter Klassifizierung (beispielsweise Trennung) der Daten muss der zunächst pro Tagestyp errechnete Mittelwert von Ausreißern befreit und korrigiert werden. Die Statistik sieht für Ausreißer-Tests mannigfache Methoden vor. Für die hier diskutierten Anwendungen hat sich ein (manchmal auch nur angenäherter) t-Test bewährt.

Da man zu Recht davon ausgehen kann, dass die Messwert-Schwankungen einer Normalverteilung unterliegen, berechnet man zunächst den Mittelwert der einzelnen Messpunkte (in diesem Anwendungsfall sind das die Laufzeiten) sowie die Varianz s^2 der Stichprobe. In SQL stehen dafür die Funktionen „AVG“ und „VAR_SAMP“ beziehungsweise „STDDEV_SAMP“ (oder nur „STDDEV“) zur Verfügung, wobei auch hier wiederum aus der Formel ersichtlich ist, dass man ausschließlich mit Mittelwert-Funktionen hinkommen würde.

In *Abbildung 5* ist die Bedeutung der Standardabweichung σ einer unendlich großen Grundgesamtheit in Bezug auf den Mittelwert μ dargestellt. In einem bestimmten Bereich (der als Vielfaches von σ angegeben wird) um den Mittelwert herum findet man alle Einzelwerte mit einer bestimmten Wahrscheinlichkeit. Diese Abweichung

$$\Delta y = \pm \sigma \cdot z_\alpha$$

wird als Vertrauensbereich oder Konfidenzintervall zum Signifikanzniveau α bezeichnet.

So kann man also beispielsweise bei einem Signifikanz-Niveau von $\alpha = 0{,}05$ beziehungsweise $1 - \alpha = 0{,}95$ davon ausgehen, dass 95 Prozent aller Messwerte in einem Bereich von $\pm 2\sigma$ um den Mittelwert liegen ($z_{0{,}05}=2$). Aufgrund der Tatsache, dass man es in der Realität mit einer endlichen Zahl von Datenpunkten zu tun hat (im Fall von monatlichen Batchprozessen können das tatsächlich mitunter auch nur 3 oder 4 sein), ergibt sich eine größere Unsicherheit (also eine breitere Verteilung), was in der Statistik durch die Verwendung der Student- oder t-Verteilung berücksichtigt wird:

$$\Delta y = \sigma \cdot t_\alpha$$

Der Student-Faktor t_α ist für ein bestimmtes Signifikanz-Niveau aus Tabellen in Abhängigkeit von der Zahl der Freiheitsgrade (das ist in diesem Fall die Zahl der Messwerte minus 1) abzulesen, geht aber für große Datenmengen gegen die z-Werte der Normalverteilung.

Als Ausreißer werden nun jene Punkte klassifiziert, die außerhalb des gewählten Vertrauensbereichs zu liegen kommen. Verzichtet man auf die Korrektur für kleine Stichproben, so kann man näherungsweise davon ausgehen, dass alle Werte außerhalb des 3σ-Bereichs (also bei An-

wendung einer Wahrscheinlichkeit von 99,7%) als Ausreißer zu betrachten sind. Diese 3σ-Regel hat sich in der Praxis als brauchbare Näherung erwiesen und liefert in der Regel sehr gute Ergebnisse.

Praktische Durchführung von Ausreißertests

Unter Verwendung der beschriebenen 3σ-Regel ergibt sich folgende Vorgehensweise:

1. Auffinden des Wertes mit der größten Abweichung vom Mittelwert

$$max|y_i - \bar{y}|$$

2. Berechnung des Schätzwertes für z:

$$\hat{z} = \frac{max|y_i - \bar{y}|}{s}$$

3. Ist $\hat{z} > 3$, handelt es sich um einen Ausreißer
4. Der Ausreißer wird eliminiert und Mittelwert und Standardabweichung werden neu berechnet

Dieses Prozedere wird so lange wiederholt, bis kein Ausreißer mehr gefunden wird. Für die Umsetzung in PL/SQL bedeutet dies die Programmierung einer rekursiven Routine. Soll die geringe Datenmenge Berücksichtigung finden, ist anstelle von $z = 3$ der entsprechende t-Wert aus der Tabelle für einseitige t-Tests zu entnehmen. *Abbildung 6* stellt den rekursiven Kreisprozess nochmals dar.

RTFC Alerts

Mit der nun bekannten Baseline (dem eigentlichen Runtime Forecast) lässt sich leicht beurteilen, ob die Laufzeit des nächsten Batchprozesses im erwarteten Rahmen liegt oder nicht. Grob gesagt, ist der nächste Wert innerhalb des berechneten Konfidenzintervalls (also beispielsweise innerhalb des 3σ-Bereichs) zu erwarten. Es liegt daher nahe, mit dem neuen Wert einen Ausreißertest auf Basis der Werte der vorgegebenen Baseline durchzuführen. Bei größeren Datenmengen kann auch in diesem Fall auf eine Endlichkeitskorrektur verzichtet und die Normalverteilung selbst verwendet werden.

Ein praxisorientiertes Alert-Schema könnte also (zumindest für größere Datenmengen) so aussehen, dass bei Überschreitung des Konfidenz-Intervalls (3σ-Bereich) ein Warning-Alert gesendet wird, bei Überschreitung des 6α-Bereichs hingegen ein Critical Alert ausgelöst wird. Speziell für Batch-Prozesse ist in jedem Fall ein hartes unteres Limit zu setzen (etwa eine Minute), unter dem keinerlei Alerting erfolgt.

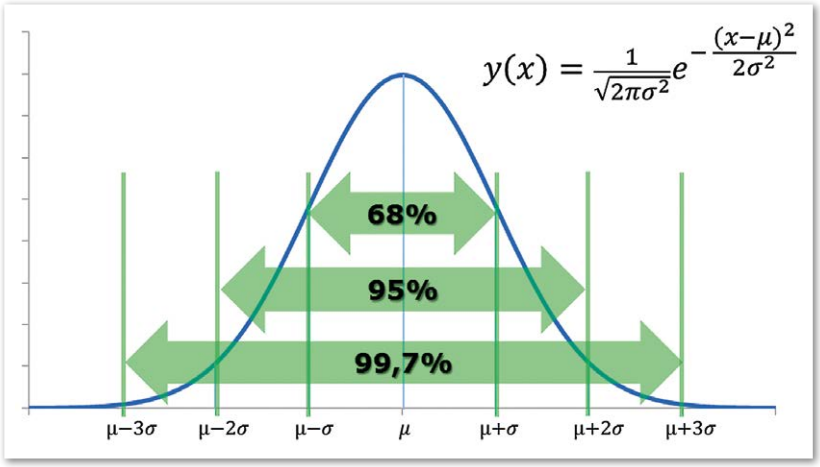

Abbildung 5: Gauß'sche Normalverteilung

Instabile Prozesse lassen naturgemäß keine Aussagen zu, können aber an der Zahl der Ausreißer schnell erkannt werden. Auch im Rahmen von RTFCs sollten lang- und kurzfristige Regressionen gerechnet werden, um einen Anstieg von Laufzeiten zu detektieren. Speziell im Data-Warehouse-Umfeld treten solche Fälle häufig durch wachsende Datenmengen bei gleichzeitig schlecht geplantem oder gar fehlendem Partitioning auf.

Availability Monitoring

Erfolgreiches Datenbank-Monitoring muss in erster Linie folgende zwei Fragen sofort und zu jeder Zeit zweifelsfrei beantworten können:

1. Ist eine bestimmte Instanz verfügbar?
2. Ist die Response-Zeit akzeptabel?

Dass diese Grundfragen allein mit Oracle-Datenbankmitteln und ohne großen Aufwand beantwortet werden können, zeigt exemplarisch der folgende Abschnitt. Zunächst benötigt man eine zentrale, dedizierte Monitoring-Datenbank, die über ein Repository aller Oracle-Instanzen mit Hosts, Listener-Ports und SIDs verfügen muss. Idealerweise wird diese Datenbank gleichzeitig als CMDB genutzt und dient auch als Kollektor für andere Monitoring-Ergebnisse (etwa Tablespace-Prognosen). Mithilfe der Repository-Information werden durch einen Startup-Prozess Database-Links zu allen Instanzen angelegt, wobei darauf zu achten ist, dass nur IP-Adressen

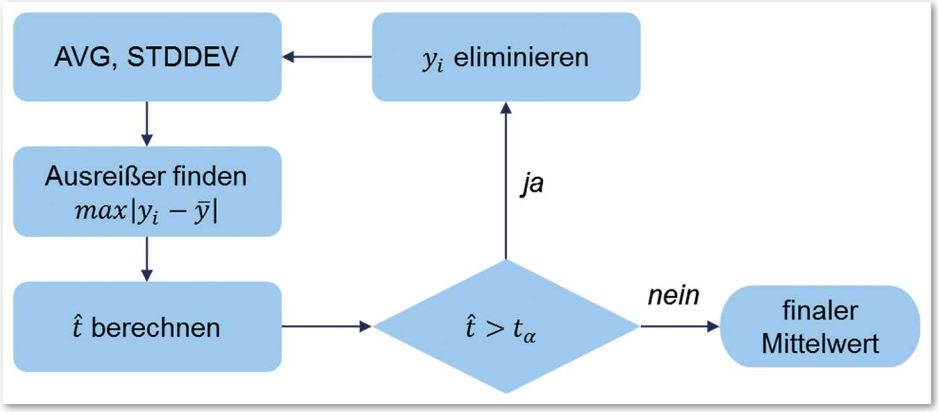

Abbildung 6: Schema eines Ausreißer-Tests

```
create database link db1 con-
nect to … identified by … using '(de
scription=(address=(protocol=tcp)
(host=192.168.0.61)(port=1521))

 (connect_data=(SID=DB1)
(server=dedicated))';
```

Listing 5

```
v_t0 := systimestamp;
select null into v_dummy from dual;
v_t1 := systimestamp;
select null into v_dummy from dual;
v_t2 := systimestamp;
v_r1 := v_t1-v_t0;  -- logon included
v_r2 := v_t2-v_t1;
```

Listing 6

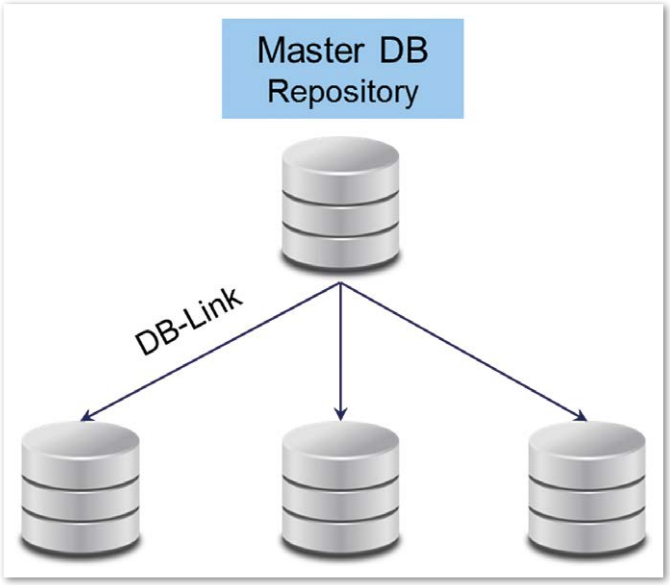

Abbildung 7: Setup für Availability-Monitoring

anstelle von Hostnamen verwendet werden, um bei der späteren Messung des Logon-Prozesses nicht implizit den DNS-Server mitzumessen. Zudem ist auf eine Dedicated-Server-Verbindung zu achten. *Abbildung 7* zeigt schematisch das Setup.

Listing 5 zeigt, wie die Database-Links aufzubauen sind. Pro Instanz wird nun ein eigener minütlicher DBMS_SCHEDULER-Job gestartet, der zwei Selects absetzt, die Zeit misst, in einer Messwert-Tabelle ablegt und danach wieder terminiert. *Listing 6* zeigt den Kernteil der Implementation in PL/SQL. Das erste Select misst den Logon-Vorgang, während das zweite Select die reine Response-Zeit der Instanz widerspiegelt.

Auf diese Art und Weise entsteht pro Instanz eine große Anzahl von Messwerten, aus denen eine Baseline (also ein Referenzwert) mit hoher statistischer Genauigkeit gerechnet werden kann. Die Vorgehensweise ist identisch mit jener, die bereits für die Runtime Forecasts demonstriert wurde, also Berechnung von Mittelwert und Varianz, Eliminieren von Ausreißern, Hinterlegen der gefundenen Referenzwerte. Statistisch signifikante Abweichungen werden so problemlos erkannt und alertiert. Zusätzlich empfiehlt sich auch in diesem Fall eine Regressionsrechnung, um eine eventuell vorkommende Drift zu erkennen.

In Analogie zum Ausreißertest für Einzelwerte lässt sich auch ein Konfidenz-Intervall für den Mittelwert definieren, wobei näherungsweise der Student-Faktor wieder durch den z-Wert der Normalverteilung ersetzt werden könnte:

$$\Delta\bar{y} = \frac{s \cdot t_\alpha}{\sqrt{n}}$$

In den Prüfwert für den (zweiseitigen) t-Test gehen die Differenz der Mittelwerte und die mittlere Standardabweichung dieser Differenz ein:

$$\hat{t} = \frac{|\bar{y}_1 - \bar{y}_2|}{\sqrt{\frac{s_1}{n_1} + \frac{s_2}{n_2}}}$$

Wird eine signifikante Änderung der Baseline festgestellt, deutet dies auf eine Systemveränderung hin. Im Übrigen spricht aber nichts dagegen, die neue Baseline (ob signifikant verschieden oder nicht) als neue Referenz zu verwenden.

Fazit

Intelligentes Monitoring baut auf einfachen Prinzipien auf:

- Know your data
  - Was ist zu monitoren?
  - Welche Verteilungen und Verhaltensweisen liegen vor?
  - Daten kennenlernen
- Keep it simple – einfache und nur wenige KPIs festlegen
- Zeitabhängige Metriken sind unverzichtbar
- Ein zentrales Repository ist die beste Informationsquelle
- Intelligentes Alerting – automatische Schwellwerte definieren (setzt automatische Baselines voraus)

Für die automatisierte Analyse ist eine fundierte Kenntnis der erforderlichen statistischen Methoden unerlässlich, auch wenn sich mitunter in der Praxis Näherungsmethoden als ausreichend erweisen. Speziell bei der Overflow-Analyse von Tablespaces wird man allerdings um eine manuelle Endbeurteilung der Daten in vielen Fällen nicht herumkommen.

Dr. Thomas Petrik
thomas.petrik@sphinx.at