SUNGARD®

IT-Symposium 2005

Wie kann man eine Datenbank performant nutzen

Peter M. Hollenstein
06. April 2005

SUNGARD® www.sungard.com

Überblick

- Bei der Anwendung „Octagon“ handelt es sich um eine Applikation zur Derivate-Abwicklung (Futures und Optionen).
- Derivat-Produkte enthalten eine bestimmte Grundkomplexität, die abgebildet werden muss (Call/Put/Future, Verfall, Basispreis, Settlementart, Optionsart, Risikoberechnung sowie -faktor).
- Die Datenhaltung hat sich in den letzten Jahren verändert. Es müssen mehr Daten als vorher online gehalten werden. Dieser Trend setzt sich auf Grund von gesetzlichen Anforderungen und Bestimmungen fort.
- Das Geschäft hat sich verändert, es werden mehr Transaktionen als früher mit weniger Personal getätigt. Auch dieser Trend setzt sich fort.
- Die Anwender erwarten um so schnellere Antwortzeiten, je mehr Transaktionen ein Benutzer abwickelt. Je weniger ein Benutzer an Geschäft tätigt, desto mehr Zeit steht zur Verfügung.

SUNGARD® www.sungard.com

Abwicklung im Vergleich

1995	
Anzahl Geschäfte	Weniger als 10000 pro Tag
Anzahl Mitarbeiter zur Abwicklung	Mehr als 4
Handelsschluss bis Abschluss Abwicklung	17:30 bis 03:00
Anzahl Stunden	8.5 Stunden
Historie	1 Monat
Anzahl Produkte	Etwa 100

2005	
Anzahl Geschäfte	Mehr als 300000 pro Tag
Anzahl Mitarbeiter zur Abwicklung	1 bis max 2
Handelsschluss bis Abschluss Abwicklung	20:00 bis 21:30
Anzahl Stunden	1.5 Stunden
Historie	Min 1 Jahr
Anzahl Produkte	Mehr als 12000

Vorgehen

Im Falle von Laufzeit-Problemen beginnt:

- Ursachenanalyse: Was benötigt eine zu lange Laufzeit?
- Korrektur des oben gefundenen Problems
- Weitere Ursachenanalyse
- Weitere Korrektur ...
- usw.

Performance Probleme von der technischen Seite zuerst beleuchten.

Dann die Geschäftslogik und deren Abbildung in der Datenbank überprüfen, eventuell bietet sich dort auch noch Potential.

Identifizierung des Problems

Laufzeitprobleme können verschiedene Ursachen haben:

- Ein einzelnes Datenbank-Statement, das sehr lange läuft.
- Ein sich wiederholendes einzelnes Datenbank-Statement, das für sich alleine zu viel Zeit verbraucht und sehr oft genutzt wird.
- Nicht genügend verfügbare Ressourcen (Speicher, CPU, Platten) für die zu verarbeitende Datenmenge.
- Das Geschäft ist zu komplex in der Datenbank und/oder der Software abgebildet.

Wir werden uns die obigen Punkte anschauen und vor allem auf den letzten Punkt konzentrieren.

Einzelnes Statement

Dies ist der „einfachste“ Fall.

Analysen haben gezeigt, dass ein einzelnes Statement alleine dafür verantwortlich ist, dass die Ausführung des Programmes zu lange dauert.

Beispiel:

Ein Ausgabe Bericht zeigt den Mark-To-Market Gewinn/Verlust. Dafür ist es notwendig, folgende Informationen für jedes einzelne Geschäft zu ermitteln:

- Anfangsposition am morgen (Ausgang)
- Aktivitäten des Tages
- Schlussposition am Abend
- Settlement Preise, Vortagespreise sowie Handelspreise
- Kundendaten

Lösung: Optimierung des Problem-Statements

Einzelnes Statement

Altes Statement:

```
select ps.account, ps.long_lots, ps.short_lots, a.contract,
   a.contract_typ, a.option_type, a.strike_price,
   a.activ_type, a.activ_amount
  from position_sum ps, account_act a
 where a.account = ps.account
   and a.contract = ps.contract
   and a.contract_typ = ps.contract_typ
   and a.option_type = ps.option_type
   and a.strike_price = ps.strike_price
   and a.activ_type in
   ('NT','CT','KT','XT','ET','AT','ST','EX','AS','RT')
   and a.activ_date = :curr_wrk_day
```


Einzelnes Statement

Neues Statement:

```
select ps.account, ps.long_lots, ps.short_lots, a.contract,
   a.contract_typ, a.option_type, a.strike_price,
   a.activ_type, a.activ_amount
  from position_sum ps, account_act a, deal d
 where d.account = ps.account
   and d.contract = ps.contract
   and d.contract_typ = ps.contract_typ
   and d.option_type = ps.option_type
   and d.strike_price = ps.strike_price
   and a.activ_type in
   ('NT','CT','KT','XT','ET','AT','ST','EX','AS','RT')
   and a.activ_date = :curr_wrk_day
   and d.deal_number = a.deal_number
```

Wiederholendes Statement

Analysen haben gezeigt, dass ein sich wiederholendes Statement dafür verantwortlich ist, dass die Ausführung des Programmes zu lange dauert.

Beispiel:

Ein Ausgabebericht zeigt den Mark-To-Market Gewinn/Verlust. Innerhalb der Tagesaktivitäten wird die Schliessungsmethode für jedes Geschäft abgefragt. Dafür ist notwendig:

- Abfrage der Schliessungsmethode der entsprechenden Transaktion.

Lösung: Optimierung des Statements

Wiederholendes Statement

Altes Statement:

```
select a.activ_type, a.deal_number
  from account_act a
 where a.account = :account
   and a.contract = :contract
   and a.contract_typ = :contract_typ
   and a.option_type = :option_type
   and a.strike_price = :strike_price
   and a.activ_type in ('ST','EX','AS','TT','EP')
   and a.activ_date = :curr_wrk_day
```

Wiederholendes Statement

Neues Statement:

```
select a.activ_type
  from account_act a
 where a.deal_number = :deal_number
   and a.activ_type in ('ST','EX','AS','TT','EP')
   and a.activ_date = :curr_wrk_day
```


Nicht genügend Resourcen

Analysen haben gezeigt, dass Teile des Rechnern voll ausgelastet sind.

Beispiel:

Ein Ausgabebericht zeigt den Mark-To-Market Gewinn/Verlust. Innerhalb der Berechnung der Zahlen stehen CPU sowie IO bei der maximalen Auslastung.

Lösung: Hinzufügen von weiteren Ressourcen

Zu komplex abgebildet

Analysen haben gezeigt, dass eigentlich alles bestens läuft:

- Ressourcen stehen genügend zur Verfügung.
- Es können keine Statements als direkte Ursache erkannt werden.

Es stellt sich die Frage, ob das Geschäft zu komplex abgebildet ist und ob in der Geschäftslogik Vereinfachungen zur notwendigen Verbesserung führen.

Auf diese Möglichkeit gehen wir jetzt näher ein und zeigen anhand von Beispielen aus Octagon, wie diese umgesetzt worden sind.

Geschäftsdaten - Beispiel 1

Geschäfte (Transaktionen) stehen in einer Tabelle „DEAL“

- Status (status_)
- Anzahl Kontrakte (anzahl)
- Art (deal_type)
- Seite (contra_ind)
- Spiegelung (omni_ind)

In der Tabelle stehen alle Geschäfte (historisch, wie auch offen).

Um gültige Geschäfte zu finden:

SQL:

```
select * from deal
where status_    = 'L'
  and anzahl     > 0
  and deal_type  in ('O')
  and contra_ind != 'Y'
  and omni_ind   != 'Y'
```

Geschäftsdaten - Beispiel 1

status_	deal_type	anzahl	contra_ind	omni_ind	account	contract	deal_num
L	O	10			0001693	ODAX	36
D	O	5		Y	SFSFRAP	FDAX	41
L	R	4			573753	ODAX	45
L	C	88			2000	ODAX	49
L	A	12			124345	OSMI	53
L	S	34			0001693	FSMI	57
L	O	0	Y		68466	OSTK	61
L	O	23			0001693	FSTK	65
L	O	45			2000	FGBL	69

Geschäftsdaten - Beispiel 1

Geschäfte stehen in einer Tabelle „DEAL“ neu

- Status (status_)

Analysen hatten gezeigt, dass das Auffinden offener Transaktionen zu lange dauerte. Um offene Transaktionen zu finden ist eine Prüfung von 5 Feldern notwendig. Ein Index auf diesen Feldern hat nicht die gewünschten Resultate gebracht.

Das System wird dahingehend angepasst, dass die 5 Felder im Feld Status zusammengefasst werden (Das Feld Status hat mehr mögliche Werte). Daraufhin wird ein Index auf dem Feld Status definiert. Alle anderen Felder werden entfernt.

Resultat: Alt: Laufzeit 8 Minuten

Neu: Laufzeit 2 Sekunden

Geschäftsdaten - Beispiel 1

status_	anzahl	account	contract	deal_num
L	10	0001693	ODAX	36
H	5	SFSFRAP	FDAX	41
R	4	573753	ODAX	45
C	88	2000	ODAX	49
A	12	124345	OSMI	53
S	34	0001693	FSMI	57
M	0	68466	OSTK	61
L	23	0001693	FSTK	65
L	45	2000	FGBL	69

Daten - Beispiel 2

Es muss der Geldbetrag des Settlements errechnet werden.

Geschäftslogik (stark vereinfacht):

Aktienoption ohne Kapitalanpassung:

Wenn Calls und Kauf: Anzahl * (Basis – Aktie) * Units

Wenn Calls und Verkauf: Anzahl * (Aktie – Basis) * Units

Wenn Puts und Kauf: Anzahl * (Aktie – Basis) * Units

Wenn Puts und Verkauf: Anzahl * (Basis – Aktie) * Units

+ Index Optionen

+ Aktienoption mit Kapitalanpassung

+ Optionen auf Futures

SUNGARD®

Daten - Beispiel 2

Code:

if Aktienoption und Kapitalanpassung

if Call und Kauf

(Basis – Underlying) * (Faktor – int(Faktor)) * Anzahl

if Call und Verkauf

(Underlying – Basis) * (int(Faktor) - Faktor) * Anzahl

if Put und Kauf

(Underlying – Basis) * (Faktor – int(Faktor)) * Anzahl

if Put und Verkauf

(Basis – Underlying) * (int(Faktor) - Faktor) * Anzahl

elseif Indexoption

usw.

SUNGARD® www.sungard.com

Daten - Beispiel 2

Analysen hatten gezeigt, dass zu viele unterschiedliche SQL Statements benutzt werden müssen und damit jede Transaktion einzeln abgearbeitet werden muss.

Das System wird so angepasst, dass die notwendige Formel unabhängig von den Produkten oder deren Parameter passt. Damit ist nur ein SQL-Statement notwendig um den Settlement-Betrag für alle betroffenen Geschäfte zu errechnen, und alle Transaktionen können gleichzeitig abgearbeitet werden.

Resultat:	Alt:	Laufzeit 20 Minuten
	Neu:	Laufzeit 40 Sekunden

SUNGARD® www.sungard.com

Daten - Beispiel 2

Umorganisation und Darstellung der Daten mathematisch:

Calls/Puts/Futures	=	1/-1/0
Kauf/Verkauf	=	1/-1
Aktien/Index	=	0/1
Kapitalanpassung	=	1

Code:

```
(Basis – Underlying) *
        ((Kapitalanpassung * (Faktor – int(Faktor))
                + (Aktien/Index * Faktor))
* Anzahl * Calls/Puts * Kauf/Verkauf
```


Daten - Beispiel 3

Geschäfte werden von zwei Seiten dargestellt (börsenseitig sowie kundenseitig). Bei Spiegelung der Geschäfte entsteht eine doppelte Spiegelung (Kunde, Börse, Spiegel-Kunde sowie Spiegel-Börse). Aus einer Transaktion entstehen also bis zu 4 Datensätze.

Geschäftslogik (stark vereinfacht):

Die verschiedenen Seiten unterscheiden sich in Kauf/Verkauf, Konto sowie die Gebühren und Provisionen. Grundsätzlich geht es bei dieser Logik nur darum, Geschäfte von verschiedenen Seiten zu betrachten.

SQL:

```
select account from position_sum
 where omni_ind = 'Y' and contra_ind = 'Y'
   and (bought_lots > 0 or sold_lots > 0)
```

Daten - Beispiel 3

Analysen hatten gezeigt, dass in der Datenbank alles neben dem Konto wiederholt werden muss. Dadurch werden mindestens doppelt so viele SQL-Statements ausgeführt wie eigentlich erforderlich.

Das System wird so angepasst, dass in einem Datensatz eine beliebige Anzahl von Seiten dargestellt werden können. Die Gebühren werden ausgelagert und in einer separaten Tabelle je Seite geführt.

Resultat:	Alt:	Laufzeiteinheit 4
	Neu:	Laufzeiteinheit 1.5

Daten - Beispiel 3

Durch die Darstellung der verschiedenen Seiten innerhalb eines Datensatzes sieht das SQL-Statement wie folgt aus:

SQL:

```
select ps.account
  from position_sum ps, account a
 where a.omni_type = 'Y'
   and a.account = ps.account
   and (bought_lots > 0 or sold_lots > 0)
```