

1 Einführung

SAP R/3 ist eine der umfassendsten betriebswirtschaftlichen Standardsoftwarepakete auf dem Markt. Die Stärke von R/3 liegt hauptsächlich in der *Breite der angebotenen Funktionalität*. So wird beispielsweise im produzierenden Gewerbe die komplette Kundenprozesskette von Angebot/Anfrage über Auftragserteilung, Auftragskonfiguration und Einplanung mit Verfügbarkeitsprüfung, Arbeitsvorbereitung, Beschaffung sowie Fertigung, schließlich Montage, Qualitätsprüfung, Auslieferung, Fakturierung bis hin zum After-Sales-Bereich in den verschiedenen SAP-R/3-Modulen unterstützt. Die Tatsache, dass diese Prozessketten in einem einzigen DV-System abgebildet sind, ermöglichen es dem SAP-Kunden, seine betrieblichen Prozesse straff und ohne Medienbrüche zu gestalten. Dies ist bzw. sollte der Hauptgrund für eine SAP-Einführung in einem Unternehmen sein, denn genau in diesem Kontext verbergen sich die Einsparpotentiale.

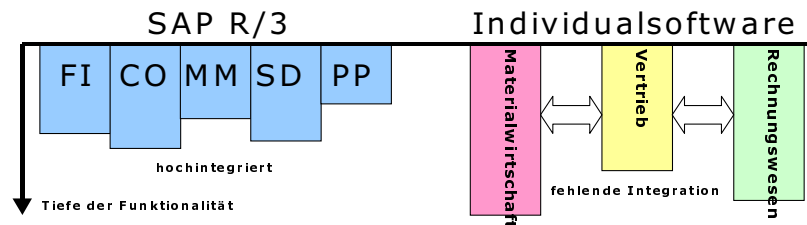
Breit sein ist nicht alles

Wie so häufig bei Standardsoftware ist die *Tiefe der angebotenen Funktionalitäten* in einzelnen Teilbereichen weniger stark ausgeprägt. Während im Rechnungswesen aufgrund der umfangreichen gesetzlichen Bestimmungen nahezu deckungsgleiche Anforderungen seitens der verschiedenen SAP-Kunden an SAP R/3 gestellt werden, sind die Ablauforganisationen z.B. im Vertrieb doch sehr unterschiedlich. Auch in der Logistik reichen die SAP-Standardfunktionen etwa in den komplexen Zulieferketten der Automobilindustrie oder im Sondermaschinenbau nicht aus.

Erschwerend kommt hinzu, dass SAP eigentlich nie die erste betriebswirtschaftliche Software eines Unternehmens ist. Meist sind die bestehenden Abläufe durch eigenentwickelte Individualsoftware gestützt, welche man oft in Form von teilbereichsoptimierten Altsystemen mit tiefer Funktionalität vorfindet.

Diese unterschiedlichen Merkmale der Standardsoftware R/3 im Vergleich zu den Altanwendungen lassen sich wie folgt gegenüberstellen:

Abb. 1-1
Gegenüberstellung
SAP R/3 – Individual-
software-Pakete



Im Rahmen der SAP-Einführungsprojekte ist es den Beteiligten oft unverständlich, dass das so hoch gelobte SAP R/3 in einzelnen Modulen manchmal weniger Funktionskomfort bietet, als die abzulösende Altsoftware. SAP sieht sich somit der schier unerfüllbaren Anforderung konfrontiert, *eine* betriebswirtschaftliche Standardsoftware für *alle* Unternehmenstopologien zur Verfügung stellen zu sollen.

Über die Customizing-Funktionen des Systems SAP R/3 kann der Kunde zwar eine ganze Reihe von Anpassungen an die betrieblichen Bedingungen vornehmen und dies vermag das Problem teilweise lösen, es sind dadurch jedoch nur solche Anpassungen möglich, die SAP bereits vorgedacht hat.

Darüber hinaus bietet die Firma SAP ihren Kunden die Möglichkeit, das System R/3 mit Hilfe der leistungsfähigen ABAP Workbench um eigene Funktionen und Auswertungen zu erweitern.

Bei der Realisierung dieser Kundensoftware werden jedoch meist nur die im Rahmen der einschlägigen ABAP-Schulungen vermittelten Kenntnisse angewandt und wenig Acht auf gute Performance der Programme gelegt. Manchmal ist es sogar die SAP-Standardsoftware selbst, die aufgrund der hohen Funktionalitätsbreite mit viel Ballast behaftet ist, die der einzelne Anwender gar nicht benötigt, die jedoch die Performance der Software negativ beeinflusst. Hier stößt man an den uralten Konflikt Standardsoftware – Individualsoftware, diesmal aus der Blickrichtung der Performance-Optimierung.

*Nicht nur funktionierende,
sondern optimal
funktionierende
Programme entwickeln!*

So entstand bei vielen SAP-Kunden eigenentwickelte Software, die zwar funktional korrekt ist, jedoch das Gesamtsystem ganz erheblich belastet. Dabei sind nur einige wenige grundsätzliche Programmierrichtlinien zu beachten, um nicht nur Programme zu schreiben, die *funktionieren*, sondern auch solche, die *optimal funktionieren*.

Dieses Buch vermittelt Ihnen Tipps und Tricks zur Performance-Optimierung Ihrer eigenentwickelten SAP-Software und zeigt Ihnen, wie Sie es schaffen, insbesondere Auswertungen im Umfeld großer Datenbestände zur Zufriedenheit der Anwender zu realisieren. Dabei darf das Gesamtsystem nicht über Gebühr belastet werden.

Als Ausgangssituation wählen wir ein Beispielprogramm, in dem wir Schritt für Schritt verschiedene Änderungen zur Performanceoptimierung vornehmen. Die Wahl des Beispiels wird im folgenden Kapitel diskutiert.

Die Verfahren, die wir zur Performance-Optimierung anwenden, müssen selbstverständlich in ihren Effekten verifiziert werden; hierzu nutzen wir die *Laufzeitanalyse* und den *SQL-Trace*, die wir in den darauf folgenden Abschnitten 2.2 und 2.3 genauer kennen lernen.

Danach untersuchen wir die Gründe, warum einige Programme mit besonders hohen Laufzeiten zu kämpfen haben, und betrachten dabei etwas genauer, welche Vorgänge in den Systemtiefen ablaufen, wenn ABAP-Programme Daten verarbeiten. Anhand der *Systemarchitektur* wird in Kapitel 2.4 deutlich, wo die klassischen Engpässe für SAP-Programme liegen.

Die Erkenntnisse, die wir hieraus gewonnen haben, lassen uns im Kapitel 2.5 einige *grundlegende Strategien zur Performance-Optimierung* erarbeiten.

1.1 Das Übungsbeispiel

Wir haben wohl alle irgendwelchen Ärger mit langlaufenden ABAP-Programmen – sonst würden Sie dieses Buch wohl kaum lesen ;-).

Die Schwierigkeit besteht nun darin, ein allgemein gültiges Übungsbeispiel zu finden, das jeder Leser leicht nachvollziehen kann und das in Ihrer Entwicklungsumgebung auch Laufzeitprobleme mit sich bringt. Modulspezifische Reports z.B. aus dem Rechnungswesen oder der Logistik sind aus verschiedenen Gründen dazu nicht geeignet:



- Entweder setzen Sie das betreffende Modul vielleicht gar nicht ein.
- Sie haben in Ihrem Entwicklungssystem nicht die notwendige Datenmenge.
- Ein Aufbau der notwendigen Datenmenge ist aus Gründen der Prozessintegration nicht oder nur mit größerem Aufwand möglich.
- Sie besitzen ein Customizing, wo diese speziellen Laufzeitprobleme gar nicht oder an anderer Stelle auftreten.

Aus diesen Gründen habe ich ein Programmbeispiel aus dem Ihnen aus dem Kontext der aus diversen ABAP-Schulungen bekannten Flugdatensystem der SAP gewählt. Gründe hierfür waren:



- Das Datenmodell ist auf jedem IDES-Entwicklungssystem verfügbar.
- Der Datenaufbau auch von großen Datenmengen ist sehr einfach durch einen einzigen Report machbar.
- Die Datenbestände sind unabhängig von irgendwelchen Customizing-Einstellungen anzulegen.

Nachdem wir nun einen hinreichend großen Datenbestand für unsere Beispielprogramme haben, besteht die nächste Schwierigkeit, eine irgendwie sinnvolle Anwendung zu finden, die dieses Datenvolumen auswertet.

Als erste Idee käme eine statistische Auswertung von Zahlen in Frage. Das dazu nötige Programmcoding wäre jedoch umfangreich und würde vom eigentlichen Problem der Performance-Optimierung ablenken.

Ich habe daher einen einfachen Report gewählt, der Flugbuchungen als Liste ausgibt. Über die Sinnhaftigkeit dieses Programms lässt sich streiten: In der Praxis würde wohl kaum ein Anwender sich eine Flugbuchungsliste mit zigtausend Zeilen am Bildschirm anzeigen lassen. Das Programm realisiert einen *uncommitted read*, d.h., es wurden keinerlei Maßnahmen ergriffen, um zu verhindern, dass ein anderes Programm zur Laufzeit der Auswertung die Daten verändert. Es kann somit bei diesem Programm vorkommen, dass inkonsistente Aussagen entstehen. In der Praxis würden Sie pro Datensatz mit Hilfe eines Sperrobjects eine Lesesperre (Shared Enqueue) setzen, die einen parallelen Update verhindert. Auch darauf habe ich verzichtet, um das Beispielprogramm klein und übersichtlich zu halten.



Falls Sie die Beispiele am System nachvollziehen möchten, was ich sehr empfehle, so steht Ihnen selbstverständlich frei, die betreffenden WRITE-Anweisungen durch eine andersartige und sinnvollere Verarbeitungslogik zu ersetzen oder einen anderen Langläufer aus Ihrer Praxis als »Versuchskaninchen« zu benutzen.

Wenn Sie mein Beispielprogramm ZPWS0001 verwenden wollen, so können Sie dieses recht einfach in Ihr R/3-Entwicklungssystem laden. Sie finden sämtliche Programmbeispiele im Internet unter der Adresse www.abaps.de unter der Rubrik *Performance-Tipps* zum Download. Die Programme liegen sämtlich als Textdatei z.B. mit Namen *ZPWS0001.txt* vor. Sie können ein Beispielprogramm auch inklusive Textelementen von einem Datenträger laden, wenn Sie das universelle Transportprogramm ZREPTRAN_46 nutzen. Näheres zum Laden der Beispielprogramme mit ZREPTRAN_46 finden Sie im Anhang.

Falls Sie sich dafür entscheiden, die Beispielprogramme einzusetzen, so haben Sie den Vorteil, die in den nachfolgenden Kapiteln beschriebenen Arbeitsschritte synchron nachvollziehen zu können.

Wenn Sie das Programm ZPWS0001 starten, so erscheint zunächst das folgende Selektionsbild:

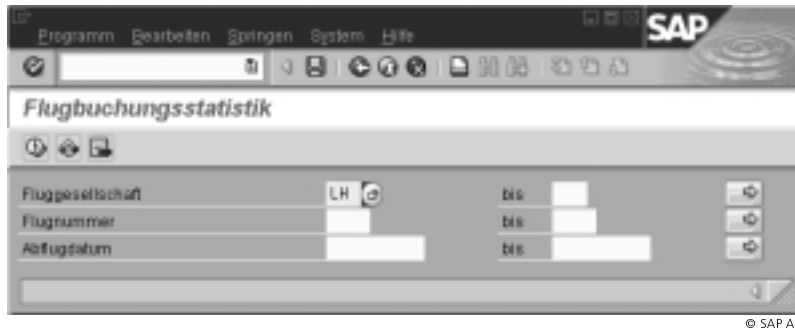


Abb. 1-2
Selektionsbild des
Beispielprogramms
ZPWS0001

Bei diesem Beispielprogramm werden Flugbuchungen ausgewertet; Sie haben die Möglichkeit, wahlweise Einschränkungen der Listanzeige vorzunehmen nach

- Fluggesellschaft und/oder
- Flugnummer und/oder
- Abflugdatum.

Nachdem Sie die Schaltfläche *Ausführen* bzw. die Taste <F8> gedrückt haben, erscheint das Listenbild.

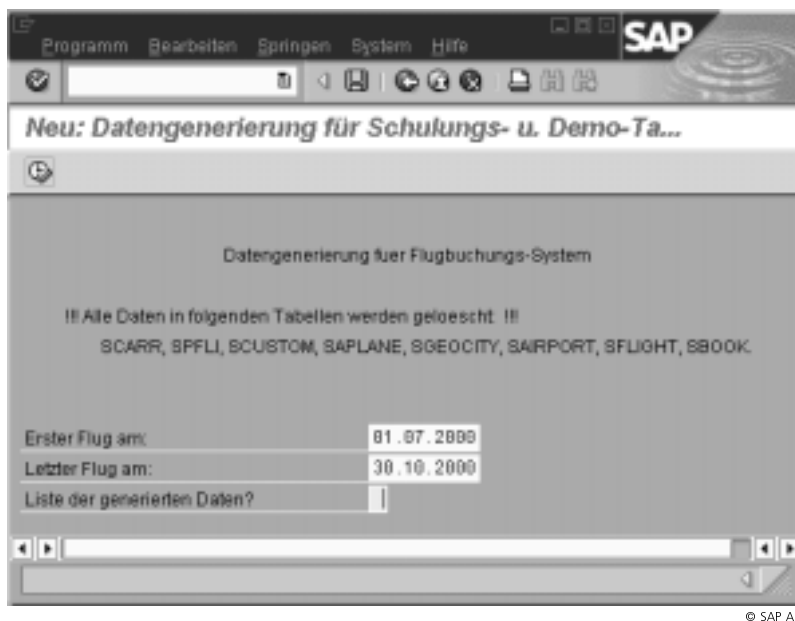


Abb. 1-3
Das Selektionsbild des
Testdatenaufbau-
programms ZSB CDAT2



- Ihre Liste ist leer? Dann haben Sie noch keine Testdaten in Ihrem Entwicklungssystem. Sie können die Testdaten leicht selbst aufbauen, indem Sie das Programm ZSBCDAT2 ausführen (wählen Sie einen Zeitraum von mindestens drei Monaten, damit Sie einen für Performance-Tests ausreichenden Datenbestand erhalten).

Das Listbild unseres Beispielprogramms sieht dann wie folgt aus:

Abb. 1-4
Das Listbild des
Beispielprogramms
ZPWS0001 bei Auswahl
nach Fluggesellschaft
Lufthansa

Flug-Nr. aa		Preis		Währung	
Buchungsnr.	Kundennr.	T	K	Buchungsdatum	
LH 0400	02.07.2000	2,459.12	DEM		
1	6	P	Y	25.06.2000	
2	16	P	F	20.06.2000	
3	31	B	C	01.07.2000	
4	6	P	Y	29.06.2000	
5	18	P	Y	25.06.2000	
6	29	B	C	20.06.2000	
7	4	P	F	01.07.2000	
8	33	B	C	29.06.2000	
9	26	P	Y	25.06.2000	

© SAP AG

In der ersten Zeile nach der Überschrift sind die Flugdaten aufgeführt: Fluggesellschaft und -nummer, das Flugdatum sowie Preis und Währung dieses Fluges. Darunter folgt eine tabellarische Aufstellung der verschiedenen Buchungen dieses Fluges mit Buchungsnummer, Kundennummer des Fluggastes, Typ des Kunden (P = Privatkunde, B = Geschäftskunde (Business)) sowie die Buchungsklasse (C/F/Y) und das Datum, wann der Flug gebucht wurde. Am Ende dieser Aufstellung folgt der nächste Flug mit seinen Buchungen usw.

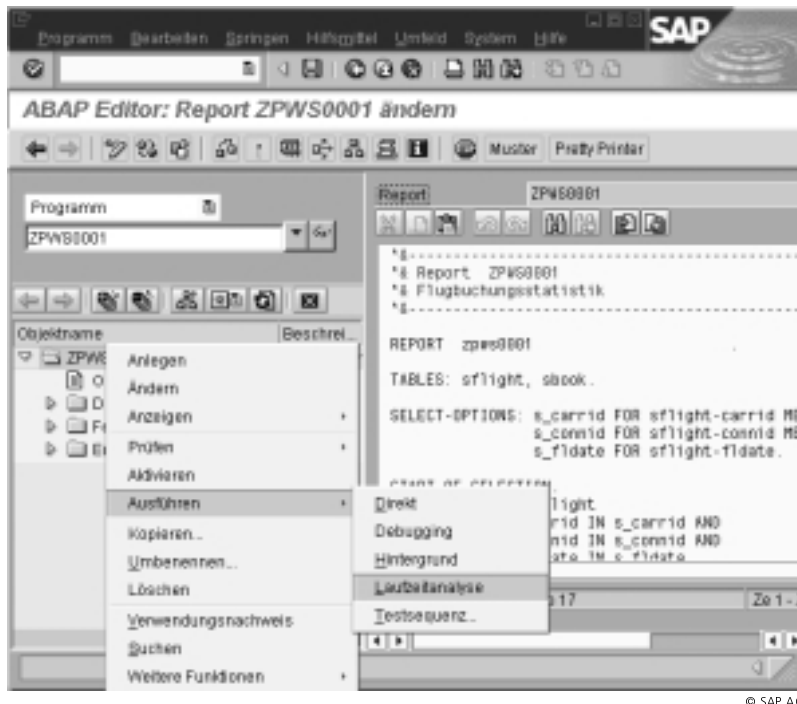
1.2 Die Laufzeitanalyse

Sie prüfen ein Programm in Bezug auf seine Performance, indem Sie das betreffende Programm in den Object Navigator der ABAP Workbench laden.

Warum sind zahlreiche Programme für SAP/R3 Sanduhr-Anzeige-Programme?

Abb. 1-5

Aufruf der Laufzeitanalyse aus dem Object Navigator

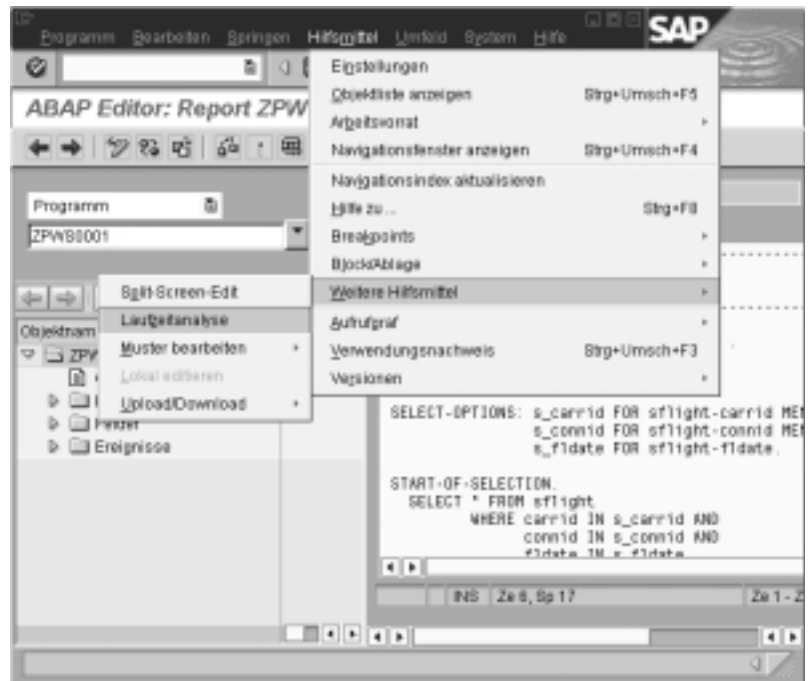


© SAP AG

Vom Object Navigator erreichen Sie die Laufzeitanalyse über das Kontextmenü *Ausführen – Laufzeitanalyse* zum Objektnamen:

Alternativ können Sie die Laufzeitanalyse auch direkt aus dem ABAP-Editor über den Menüpunkt *Hilfsmittel – weitere Hilfsmittel – Laufzeitanalyse* starten:

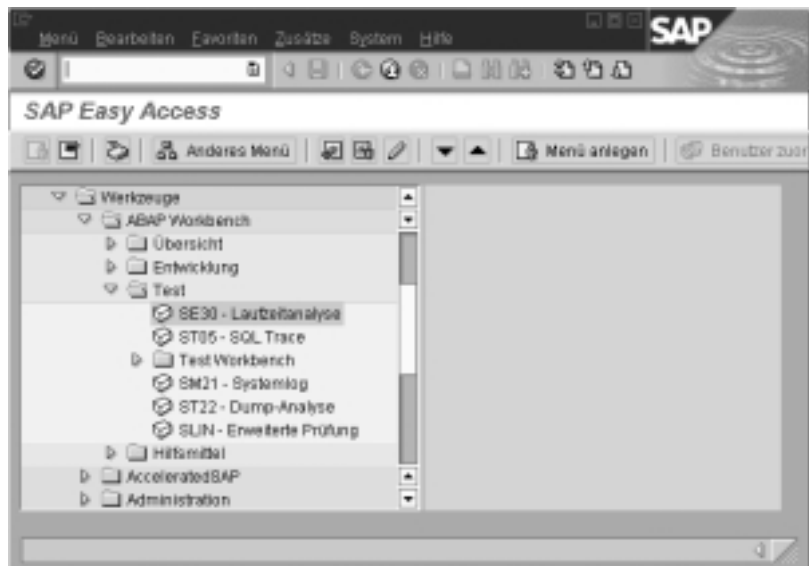
Abb. 1-6
 Aufruf der
 Laufzeitanalyse aus dem
 ABAP-Editor



© SAP AG

Weitere Möglichkeiten sind der direkte Aufruf aus dem *Easy Access*-Menü oder der Direktaufruf der Transaktion SE30:

Abb. 1-7
 Direktaufruf der
 Laufzeitanalyse



© SAP AG

Nach Aufruf der Laufzeitanalyse erscheint das Einstiegsbild:

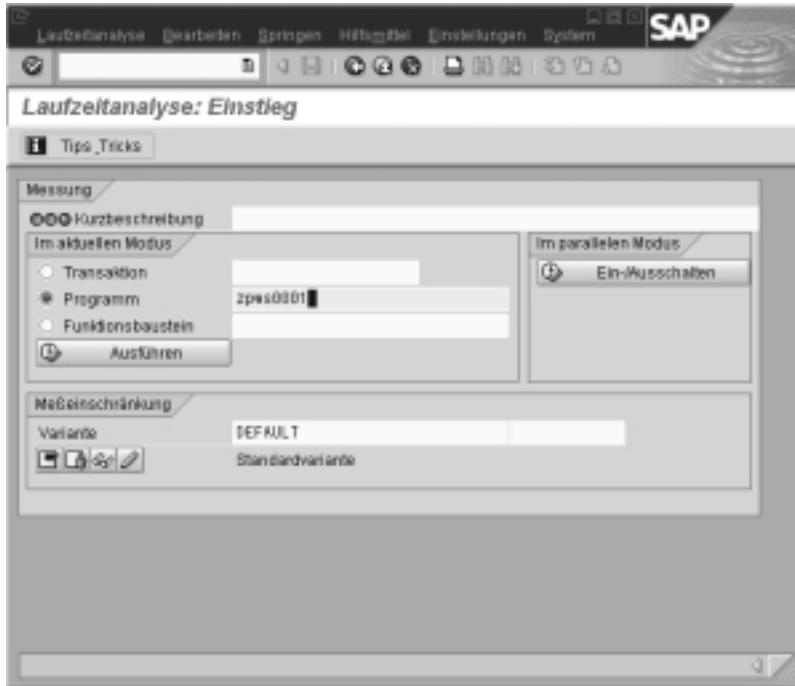


Abb. 1-8
Das Einstiegsbild der
Laufzeitanalyse

Tragen Sie als Programm den Namen des zu analysierenden Programms ein und markieren Sie den nebenstehenden Auswahlknopf.

Durch Mausklick auf die Schaltfläche *Ausführen* starten Sie die Laufzeitmessung. Hierbei werden alle Bilder Ihres Programms ganz normal durchlaufen, performance-relevante Informationen jedoch für die spätere Auswertung gesammelt und in einer Datei gespeichert.

Anstatt unseres Beispielprogramms ZPWS0001 können Sie selbstverständlich jeden anderen Report zur Übung mit der Laufzeitanalyse verwenden.

In unserem Fall erscheint zunächst das Selektionsbild des Reports ZPWS0001. Nachdem Sie eine Eingabe getätigt haben (z.B. Fluggesellschaft »LH«), starten Sie die Auswertung und es erscheint das Listbild.

Alle Systemaktionen, die im Verlauf zur Erstellung der Liste stattfinden, werden vermessen und protokolliert.

Nachdem Ihr Programm durchgelaufen ist, beenden Sie dieses z.B. durch den gelben Pfeil nach oben – es erscheint wieder das Grundbild der Laufzeitanalyse:

Abb. 1-9
Das Einstiegsbild der
Laufzeitanalyse nach der
Messung

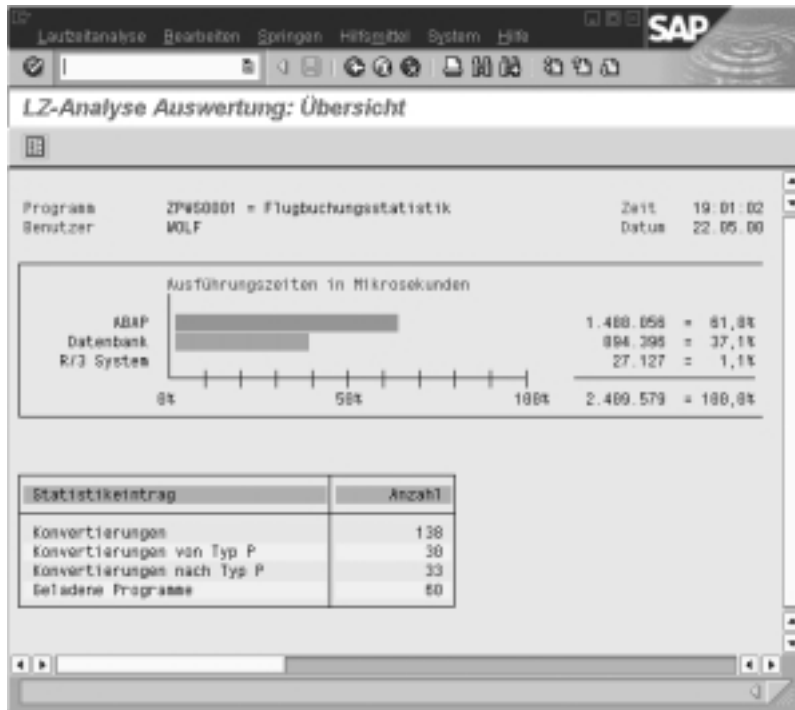


© SAP AG

In der unteren Bildschirmhälfte im Block *Meßdatendatei* befinden sich nun die Angaben zur gerade erstellten Messdatei – bei großen Datenmengen erkennen Sie allein durch die Größe der Datei, dass eine ganze Menge an Informationen gesammelt wurden. Sollte die Laufzeitanalyse abbrechen, so liegt es höchstwahrscheinlich an der Überschreitung der Disk-Quota des R/3-Systems auf dem Applikationsserver; konsultieren Sie in diesem Fall Ihren Systemadministrator.

Klicken Sie auf die Schaltfläche *Auswerten* um die Auswertung zu starten. Dies kann je nach Größe der Messdatei einige Sekunden dauern.

Anschließend erscheint das Übersichtsbild der Auswertung:

**Abb. 1-10**

Das Übersichtsbild der Auswertung einer Messdatei

Die Balken sind im SAP-System eigentlich farbig, eine rote Farbe des ersten Balkens zeigt an, dass während des Programmlaufs eine übermäßige Last auf dem Applikationsserver erzeugt wurde.

Durch Klick auf die Schaltfläche oben links unter dem Titel *LZ-Analyse Auswertung: Übersicht* gelangen Sie in die Hitliste.

Dort wollen wir genauer analysieren, welcher Tabellenzugriff für die hohe Laufzeit verantwortlich war.

Sie können die Aufstellung nach beliebigen Spalten auf- oder absteigend sortieren (z.B. *Netto*), indem Sie die Spaltenüberschrift markieren und die *Sortieren*-Schaltfläche drücken.

Es wird nun deutlich, dass der größte Anteil der Datenbanklast durch die über 43.000 Zugriffe auf die Tabelle SBOOK verursacht wurden.

Abb. 1-11
Hitliste der
Laufzeitfaktoren

	Brutto	Netto	Brut. %
1	2.318.444	1.426.052	96,2
5	852.412	852.412	35,4
9	26.025	26.025	1,1
2	2.298.360	15.235	98,2
18	12.734	12.734	0,5
2	39.518	8.524	1,6

© SAP AG

Dies wollen wir noch detaillierter untersuchen: Markieren Sie die Zeile SBOOK und wählen Sie aus dem Menü *Springen – Quelltext anzeigen*.

Abb. 1-12
Quellcode-Auszug des
betreffenden Programms

```

SELECT * FROM sbook
  WHERE carrid = sflight-carrid AND
         connid = sflight-connid AND
         fldate = sflight-fldate.
WRITE: /12 sbook-bookid NO-ZERO,
       26 sbook-custoid NO-ZERO,
       39 sbook-custtyp.

```

© SAP AG

Es wird deutlich, dass die Zugriffe aus Zeile 30 des Programms ZPWS0001 erfolgten.

Sie befinden sich im ABAP-Editor. Wenn Sie möchten, können Sie das Programm gleich bearbeiten, um die Laufzeit zu optimieren, und dann einen erneuten Test ausführen, die Verfahren hierzu werden in den nachfolgenden Kapiteln behandelt.

Beim Verlassen der Laufzeitanalyse bedenken Sie bitte, dass Sie ggf. die großen Messwertdateien wieder löschen. Dies geschieht üblicherweise nicht automatisch, es sei denn, Ihr Systemadministrator hat eine entsprechende Bereinigung konfiguriert!

Nun haben Sie die ABAP-Laufzeitanalyse etwas näher kennen gelernt. Sie ist ein hervorragendes Instrument, um das Laufzeitverhalten eines ABAP-Programms als Ganzes zu vermessen, und zeigt Datenbankaspekte wie auch die ABAP-Last. Um die Art und Weise zu ermitteln, wie das Datenbanksystem die gewünschten Daten zur Verfügung stellt, reicht sie jedoch nicht aus.

Hierzu hat SAP ein weiteres leistungsfähiges Messinstrument, den SQL-Trace, den wir im nächsten Abschnitt besprechen, bereitgestellt.

1.3 Der SQL-Trace

Sie erreichen den SQL-Trace aus *Easy Access* über den Menüpfad *Werkzeuge – ABAP Workbench – Test – SQL-Trace* oder über die Transaktion *ST05*.

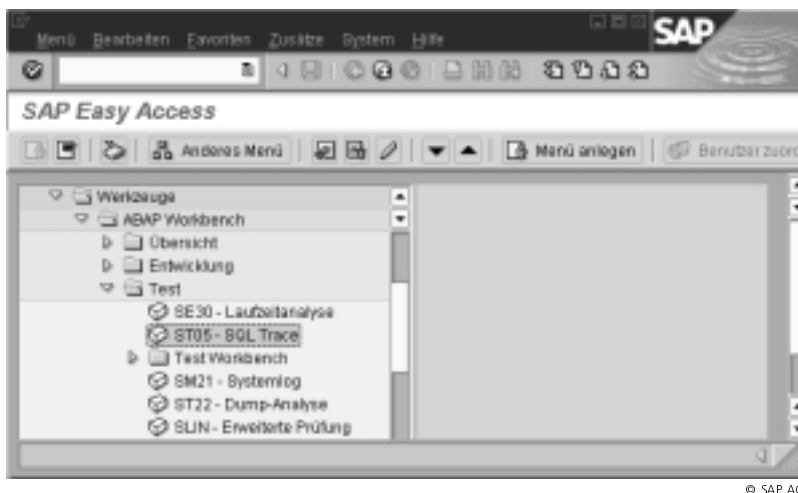
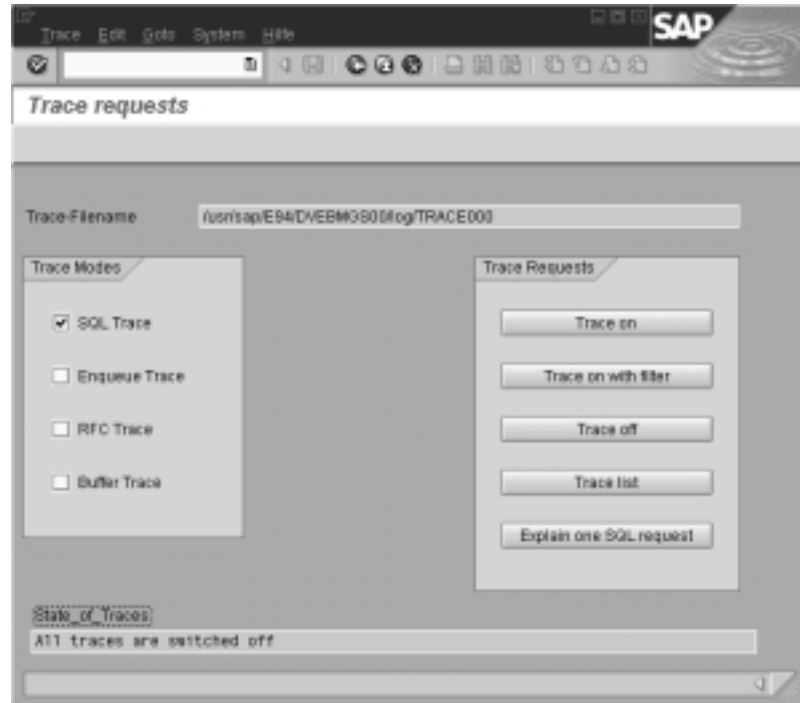


Abb. 1-13
Der SQL-Trace

Bei Aufruf des SQL-Trace erscheint zunächst ein Auswahlbild, in dem Sie einstellen können, welche Systemaktivitäten aufgezeichnet werden sollen. Uns interessiert hier primär die Aufzeichnung der Datenbankaktivitäten mit dem *SQL-Trace*. Die aufgezeichneten Systemaktivitäten werden in einer rundlaufenden Logdatei festgehalten, diese muss über die Systemadministration in ausreichender Größe definiert sein, damit alle Trace-Informationen eines Programmlaufs darin gehalten werden können.

Abb. 1-14
Das Einstiegsbild des
SQL-Trace



© SAP AG

Da uns das Verhalten eines spezifischen Programms interessiert, empfiehlt es sich, die Aufzeichnung *mit Filter* zu starten.

Abb. 1-15
Filtereinstellung für den
SQL-Trace



© SAP AG

Auf dem folgenden Popup-Fenster werden die Filterkriterien eingetragen. Im obigen Beispiel sollen nur Programmläufe des Anwenders WOLF aufgezeichnet werden, und zwar nur, wenn dieser das Programm ZPWS0001 aufruft. Standardmäßig werden die im Rahmen der Programminitialisierung durch das Laufzeitsystem verursachten Zugriffe auf die Dictionary-Tabellen, deren Namen mit D010 und D020 beginnen, sowie die intransparente DDLOG-Tabelle nicht aufgezeichnet. Falls Sie nur Zugriffe auf bestimmte Datenbanktabellen vermessen wollen, so können Sie diese unter der Rubrik *Inclusive* eintragen. Wenn dort Einträge vorliegen, so werden nur die diese Tabellen betreffenden Zugriffe aufgezeichnet.

- ➔ Insbesondere bei Langläufern ist es wichtig, hier sinnvolle Einschränkungen vorzunehmen, damit die Logdatei nicht überläuft.

Sie schalten den Trace ein, indem Sie einfach die *Enter*-Taste drücken. Anschließend können Sie den zu untersuchenden Report starten. Üblicherweise erzeugt man einen zweiten Bildschirmmodus und führt das Programm dort aus. So ist ein rascher Wechsel zwischen Programmquellcode und SQL-Trace mit der Tastenkombination *Alt-Tab* möglich.

Nachdem der Report-Lauf beendet wurde, verzweigen wir erneut in den SQL-Trace und schalten zuerst den Trace wieder aus (Schaltfläche *Trace off*).

- ➔ Bitte versäumen Sie nicht dies zu tun, denn das Sammeln der Trace-Daten bedeutet für das Gesamtsystem einen hohen Aufwand.

Schließlich schauen wir uns über die Schaltfläche *Trace list* die Aufzeichnung an. Es erscheint zunächst ein Popup-Fenster, worin Sie die Trace-Anzeige filtern können.

- ➔ Die Anzeige unterscheidet sich leicht von Datenbanksystem zu Datenbanksystem, den folgenden Beispielen liegt Oracle 8 zugrunde, welches derzeit das am meisten verbreitete Datenbanksystem für SAP R/3 sein dürfte.
- ➔ Bitte versäumen Sie nicht die voreingestellten *Objektnamen* zu entfernen, sonst erhalten Sie kaum eine Ergebnisanzeige.

Ausgewertet wird auf jeden Fall die systemweite Trace-Datei. Da es durchaus vorkommen kann, dass in der Datei noch Trace-Infos anderer Anwender bzw. Programme stehen, muss auch an dieser Stelle der Filter entsprechend eingestellt werden. Bei Start des Trace legen Sie fest, welche Infos gesammelt werden sollen, bei der Trace-Liste bestimmen Sie, welche aus der globalen Trace-Datei angezeigt werden sollen.



Abb. 1-16
Auswahl zur Anzeige der
Trace-Liste



Falls Sie mehrere Messungen zu einem Programm vorgenommen haben, so sollten Sie auch Start- und Endezeit eingeben, damit Sie den richtigen Programmablauf »erwischen«.

Für die Anzeige der Liste stehen Ihnen zwei Layouts zur Verfügung: Die *erweiterte Liste* kann auch aus der *Grundliste* erreicht werden und bietet weitere Spalten u.a. mit Ausführungszeitpunkt, Programmname.

Die Grundliste zeigt für unser Beispielprogramm ZPWS0001 folgendes Layout:

Abb. 1-17
Trace-Liste für das
Programm ZPWS0001

The screenshot shows the 'Detailed statement' window in SAP. The table displays execution statistics for program ZPWS0001. The columns are Duration, Objectname, Oper, Rec, RC, and Statement. The data is as follows:

Duration	Objectname	Oper	Rec	RC	Statement
39.123	SFLIGHT	PREPARE	0	0	SELECT WHERE "MANDT" = ? AND "CARRID" =
40	SFLIGHT	DECLARE	0	0	SELECT WHERE "MANDT" = ? AND "CARRID" =
80	SFLIGHT	OPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
78.226	SFLIGHT	FETCH	428	0	
5.183	SBOOK	PREPARE	0	0	SELECT WHERE "MANDT" = ? AND "CARRID" =
37	SBOOK	DECLARE	0	0	SELECT WHERE "MANDT" = ? AND "CARRID" =
66	SBOOK	OPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
68.259	SBOOK	FETCH	43	100	
69	SBOOK	REOPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
31.087	SBOOK	FETCH	43	100	
68	SBOOK	REOPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
5.844	SBOOK	FETCH	43	100	
88	SBOOK	REOPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
5.749	SBOOK	FETCH	43	100	
68	SBOOK	REOPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
5.719	SBOOK	FETCH	43	100	
68	SBOOK	REOPEN	0	0	SELECT WHERE "MANDT" = '100' AND "CARRID
5.761	SBOOK	FETCH	43	100	

Die Spalten zeigen neben Namen der Datenbanktabelle die Dauer und Kurzform der Datenbankzugriffs-Anweisung und die Anzahl der durch diese Anweisung von die Datenbank auf den Applikationsserver übertragenen Datensätze.

Die Liste kann nach beliebigen Spalten sortiert werden und Sie können sich technische Tabelleneinstellungen aus dem Data-Dictionary anzeigen lassen. Des weiteren besteht die Möglichkeit, direkt ins Programm zu springen, um die verursachende ABAP-Anweisung zu sichten. Dies funktioniert jedoch nur für solche Zugriffe, die aus ABAP-Programmen heraus erfolgten (einige Funktionen des SAP-Kernels führen ebenfalls Datenbankzugriffe durch).

Die Detailanzeige der Anweisung kann auch statt der aktuellen Werte die Namen der betreffenden Variablen anzeigen, die zur Ausführungszeit diese Werte enthielten. Diese Ersetzung erreichen Sie über die Schaltfläche *Var. replace*. Durch Doppelklick auf eine Zeile in der Spalte *Duration* oder *Statement* oder Auswahl der Schaltfläche mit der Lupe gelangen wir in die Detailanzeige.

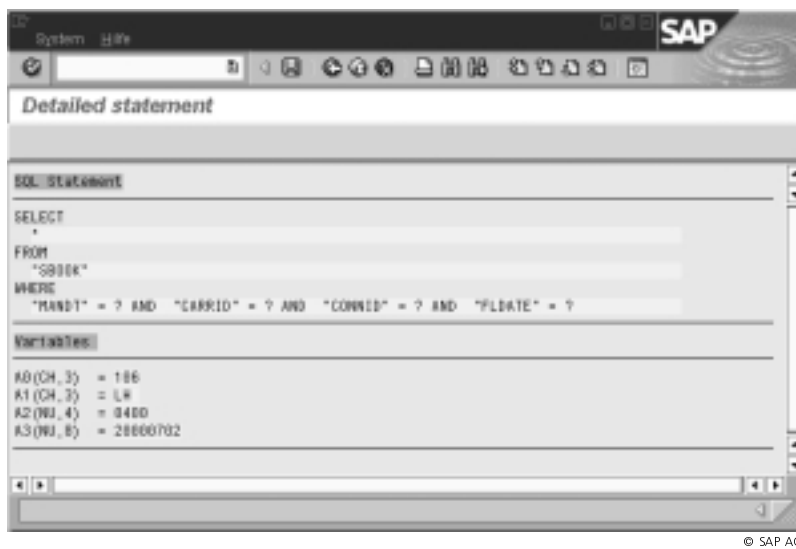


Abb. 1-18
Details zur SELECT-
Anweisung

Hier ist die vom SAP-Datenbank-Interface generierte Native-SQL-Anweisung (hier: Oracle 8) einschließlich der bei diesem Aufruf mitgegebenen Variableninhalten erkennbar.

Durch Doppelklick auf den Tabellennamen auf dem Übersichtsbild sehen wir einige wichtige technische Merkmale der betroffenen Datenbanktabelle. Diese Informationen werden aus dem Data-Dictionary bereitgestellt und zeigen z.B., welche Indizes mit welchen Schlüsselfeldern die betreffende Datenbanktabelle besitzt.

Abb. 1-19
Technische
Eigenschaften der
betreffenden
Datenbanktabelle

Information from SAP-Dictionary

Indexfields Table fields Fields all indexes

Overview on table SBOOK

Objectname	SBOOK
SQL object	SBOOK
Table class	TRANSP
Data class	Transaction data, transparent tables
Size category	Tables < 25 MB
SAP buffer	Buffering not permitted

Short info

Classification	Flugbuchung
	BC Basis
	BC-S&B ABAP Workbench
	(Datenmodell) BC_Travel für Workbench Training und D
Author	SAP
Changed by	SAP

Metatab row length 97 Byte

Last DB analysis 29.01.2000

No. of rows 0

DB avg. row length 2731 Byte

Indices of SBOOK

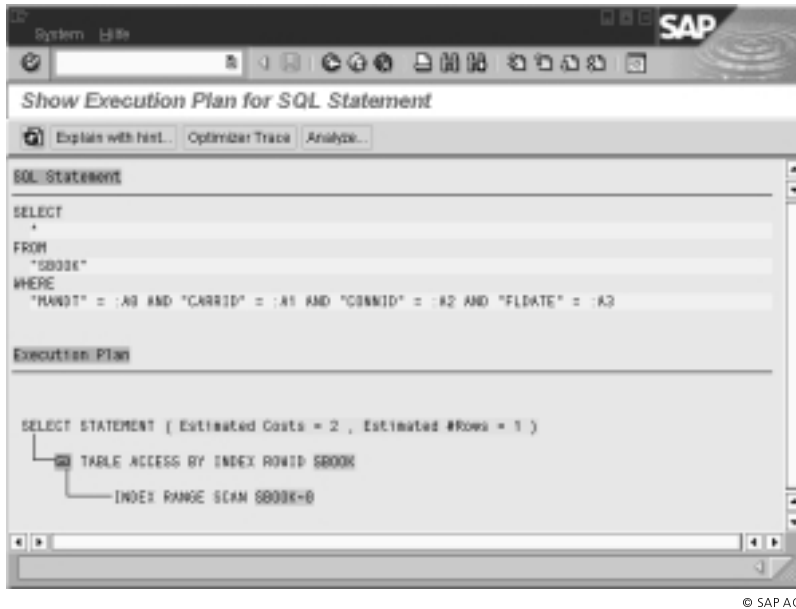
Name	Description	Created by	Uniq
0	Primary key	SAP	X
ACY	Index über die Nummer des Reisebüros	SAP	
CUS	Index über Kundennummer	SAP	

© SAP AG

Die für uns wichtigste Funktion des SQL-Trace ist die *Explain*-Funktion. Über die entsprechende Schaltfläche aus der Grundliste wird die markierte Zeile detailliert analysiert. Diese Funktion ist für die Zeilen mit der Kennung PREPARE und OPEN CURSOR möglich.

Dabei wird das jeweilige Datenbanksystem beauftragt, im Detail zu erklären, auf welche Weise es für genau diese Anweisung die betreffenden Daten bereitstellen würde. Man nennt diese Erklärung den Ausführungsplan, den der DB-Optimizer zur Ausführungszeit für die betreffende SQL-Anweisung erstellt. Es kann so z.B. erkannt werden, welchen Index das Datenbanksystem wählt.

Das Listbild der *Explain*-Funktion unterscheidet sich je nach Datenbanksystem. In unserem Beispiel (Oracle 8®) erkennen Sie, dass das Datenbanksystem den Primärindex SBOOK~0 verwenden würde, um an die gewünschten Daten zu gelangen. Die Schaltflächen führen zu weiteren datenbankspezifischen Auswertemöglichkeiten, auf die wir hier nicht weiter eingehen.

**Abb. 1-20**

Ausführungsplan zum
OPEN CURSOR SBOOK
bei Oracle 8®
(Explain SQL)

Per Doppelklick auf eine Zeile in der *Oper*-Spalte schließlich gelangen Sie direkt in den Quelltext des betreffenden Programms zur verursachenden Anweisung.

- ➔ Eine weitere interessante Möglichkeit ist die Ausweisung identischer Zugriffe. Über den Menüpunkt *Goto – Identical Selects* wird geschaut, ob im aktuellen Auswertebereich einige Datenbankzugriffe mit identischen Schlüsselvorgaben vorkommen. Dies ist eine zusätzliche wichtige Funktion, um schlecht programmierte Anwendungen zu optimieren.



In den nachfolgenden Kapiteln werde ich hin und wieder auf den SQL-Trace und dessen *Explain*-Funktionalität verweisen und dabei weitere Recherchemöglichkeiten dieses Werkzeugs erklären. Durch geschickte Veränderungen der ABAP-SELECT-Anweisung erreichen wir bessere Zugriffsmöglichkeiten für das Datenbanksystem, die es uns mit der *Explain*-Funktion erklärt und die sich in verbesserter Laufzeit auswirken. Mit diesem Werkzeug können wir somit die Auswirkungen unserer Tuningmaßnahmen in Bezug auf die Datenbank bis ins tiefste Detail erforschen.

Wir werden später anhand des Vergleichs der Laufzeitanalysen und SQL-Traces verschiedener Programmvarianten herausfinden, wie sich die verschiedenen Maßnahmen zur Performanceoptimierung auf die Laufzeit unserer ABAP-Programme auswirken. Hierzu werden wir die Programme jeweils in einer Weise abändern, dass ihre

Funktionalität gleich bleibt, und wir so eine direkte Vergleichsmöglichkeit, einen Benchmark, erhalten.

Wenn Sie mit Hilfe der Laufzeitanalyse einige Ihrer eigenen Programme oder SAP-Programme vermessen, so werden Sie feststellen, dass manche ihre Aufgabe recht schnell erledigen, andere im Gegensatz unverhältnismäßig lang laufen. Oft liegt bei letzteren »Kandidaten« die Hauptlast im Datenbank-Balken.

Um letztendlich Schwachstellen im Source-Code der betreffenden Programme professionell identifizieren zu können, müssen wir die Auswirkungen der verschiedenen Anweisungs-Varianten auf das SAP-System beurteilen. Dies wiederum kann nur geschehen, wenn wir die Vorgänge im Innern des SAP-Systems anhand der *Systemarchitektur* genauer untersucht haben.

Schlechte Software-Performance hat als Ursache immer entweder

- ungeschicktes Grunddesign der Anwendung oder
- ungeschickte und unnötige Auslösung von Aktivitäten in den Systemtiefen des SAP-Systems.

1.4 Die SAP-R/3-Systemarchitektur im Hinblick auf Software-Performance

Das Highlander-Prinzip in der SAP-Systemarchitektur oder: »Es kann nur einen Datenbankserver geben«.

In meiner bisherigen Praxis stellte ich anhand der Laufzeitanalyse fest, dass in über 90% der Fälle unperformante Programme eine sehr hohe Datenbanklast erzeugen (der rote Balken). Es drängt sich der Gedanke auf, dass genau die hohe Datenbanklast Ursache für die schlechte Laufzeit ist.

Um diesen Verdacht zu untersuchen, wollen wir uns einmal das Zusammenspiel der verschiedenen Systemkomponenten eines SAP-Systems beim Zugriff auf Datenbanktabellen etwas genauer anschauen.



- ➔ Wenn wir in den späteren Kapiteln die verschiedenen Tuning-Maßnahmen besprechen, so werde ich immer wieder auf diese Abbildung verweisen, zeigt sie doch genau die Systemkomponenten, welche beim Datenbankzugriff angesprochen werden.

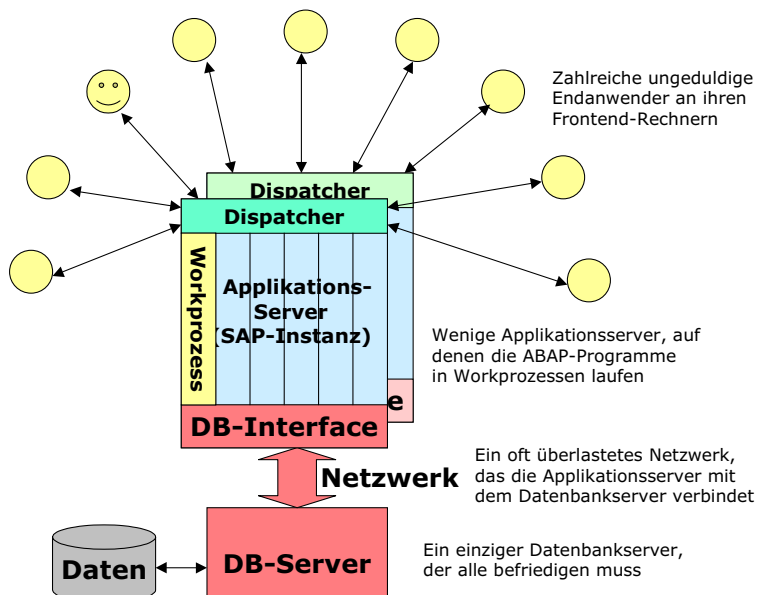


Abb. 1-21
Die SAP-R/3-Systemarchitektur

Die Endanwender bedienen an den Frontend-Rechnern die SAPGUI-Oberfläche des R/3-Systems. Die lokale Rechnerintelligenz wird jedoch kaum genutzt, sondern fast jede Aktion des Anwenders gelangt über das Netzwerk an den Applikationsserver, wo nach einigen Zwischenschritten die ABAP-Programme auf die Aktion reagieren und die jeweilige Anforderung befriedigen.

Die ABAP-Programme allein? Wohl kaum! Ohne Daten aus den zugehörigen Datenbanktabellen können diese dem Endanwender recht wenig bieten. Die ABAP-Datenbankzugriffsbefehle (SELECT...) werden durch das DB-Interface innerhalb des Workprozesses auf dem Applikationsservern in den Native-SQL-Dialekt des jeweiligen Datenbanksystems übersetzt und über das Netzwerk an den Datenbankserver weitergegeben.

Dieser führt die notwendigen Zugriffe aus und schickt die Ergebnismenge der angeforderten Daten wieder über das Netzwerk an den Applikationsserver zurück. Dort fängt die DB-Schnittstelle die Daten ab, führt ggf. notwendige Konvertierungen durch und übergibt schließlich die gelesenen Daten an das ABAP-Programm.

Der Datenbankserver selbst weiß nichts vom anfordernden ABAP-Programm, noch was dieses anschließend mit den Daten macht.

Ich habe nun recht detailliert die Vorgänge beim Datenbankzugriff durch ein Anwendungsprogramm beschrieben. In einer Produktivum-

gebung sind jedoch oft mehrere Anwender gleichzeitig aktiv und lösen Datenbankzugriffe aus.

Wird die *Performance des Applikationsservers* durch zu hohe CPU-Auslastung oder Speichermangel schlechter, so gibt es einen einfachen Lösungsansatz: Man kann diesen aufrüsten oder einen zweiten oder dritten Applikationsserver hinzufügen um so eine Lastverteilung zu erreichen. Es existieren SAP-Installationen mit zehn Applikationsservern und mehr.

Was kann man jedoch tun, wenn der *Datenbankserver* überlastet ist?

Sie werden fragen, ob man nicht auch einen zweiten Datenbankserver installieren kann, SAP R/3 ist doch ein Client/Server-System.

Client/Server à la SAP

Leider ist dies nicht möglich, denn SAP R/3 stellt ein monolithisches Zentralsystem ähnlich einer traditionellen Großrechner-Applikation dar.

SAP R/3 implementiert ein dreistufiges Client/Server-Prinzip:

*Das dreistufige
Client/Server-Prinzip von
SAP R/3*

- Die SAPGUI-Oberfläche oder ein Webbrowser an zahlreichen Frontend-Clients stellt die erste Stufe dar.
- Die SAP-Anwendungssoftware läuft auf einem oder mehreren Applikationsservern – die zweite Stufe.
- Die Daten der SAP-Applikation befinden sich ausnahmslos auf dem einen und einzigen Datenbankserver als dritte Stufe.

Ein SAP-R/3-System ist somit nur eine Client/Server-Anwendung mit Einschränkungen. Insbesondere handelt es sich nicht um eine echt verteilte Anwendung.

SAP hat jedoch kaum eine Möglichkeit, dies zu ändern, denn um ein *echt* verteiltes Anwendungssystem zu realisieren, müssten auch *echt* verteilte Datenbanksysteme verfügbar sein. Nach derzeitigem Stand gibt es jedoch erst einige wenige experimentelle Implementierungen, die die Datenbank-Problematik des two-phase-committing und der kollisionsbehafteten Replikation beherrschen – dann aber nur bei geringen Datenmengen.

Aus diesen Gründen ist eindeutig die Datenbank und deren Netz-anbindung der klassische Engpass eines SAP-R/3-Systems.

- ➔ Überlegen Sie einmal: Gilt dies nur für SAP-R/3 oder ist dies etwa eine universell gültige Feststellung?

Neben dem Datenbanktuning sind eine Aufrüstung oder ein Austausch des DB-Servers die einzigen Maßnahmen, die man zur Verbesserung des Performance-Verhaltens ergreifen kann – außer einem Software-Tuning, dies ist unser Thema.

Nachdem nun eine der wichtigsten Engpassstellen erkannt sind, können wir *Grundsätze der performanten Datenbank-Programmierung* definieren.

1.5 Grundsätzliche Strategien zur Performance-Optimierung von Datenbankzugriffen

Behandle deine Datenbank so, als wäre sie eine VIP:

- ❑ Verlange von deinem DB-Server nur diejenigen Informationen, die Du auch wirklich benötigst.
- ❑ Störe deinen DB-Server so selten wie möglich.
- ❑ Stelle die Fragen an deinen DB-Server möglichst präzise, so dass dieser auch einfach die Antwort findet.
- ❑ Stelle deinem DB-Server niemals die gleiche Frage mehrmals kurz hintereinander.

Wenn Sie diese Regeln beherzigen, dann werden Sie eine erfolgreiche Zusammenarbeit mit den VIPs – und ihrem DB-Server – begründen können.

Diese »Gesetze« stellen grundsätzliche Strategien zur Performance-optimierung von Datenbankzugriffen dar, auf die wir im folgenden Kapitel detailliert eingehen werden.

Die Datenbank – eine heimliche VIP

