CARL HANSER VERLAG

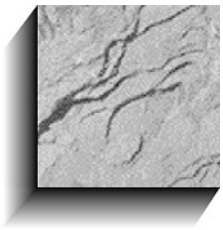
Nirva Morisseau-Leroy, Martin Solomon, Gerald Momplaisir

Oracle9i SQLJ Programmierung

Entwicklung robuster Java-Datenbankanwendungen mit SQLJ
Version 8i und 9i

3-446-21848-3

www.hanser.de



Einführung

In Teil I und II des vorliegenden Buchs werden Sie SQLJ-Programme entwickeln, die eine „rein" relationale Datenbank bearbeiten, und zwar ein Finanzschema namens Purchase Order-Schema. Das relationale Datenbankschema Purchase Order gehört zu dem Datenbankdesign, das im *Design of a Financial Administrative System Using the Semantic Binary Model* vorgestellt wird. In Teil III werden Sie SQLJ-Programme entwickeln, die eine objektrelationale Datenbank bearbeiten, in diesem Fall ein wissenschaftliches Schema namens Observation-Schema. Dieses objektrelationale Observation-Datenbankschema gehört zu einer wissenschaftlichen Datenbank, die in *Atmospheric Observations, Analyses, and the World Wide Web Using a Semantic Database* vorgestellt wird. Beachten Sie, dass alle Programme, die Sie in diesem Buch entwickeln werden, für die Oracle8*i*-Datenbankserver Releases 8.1.5, 8.1.6 und 8.1.7 getestet wurden, und einige wurden auch auf dem Oracle9*i* Internet Application Server (Oracle9*i* AS) eingesetzt. In diesem Buch werden auch einige neue Konzepte behandelt, insbesondere die Unterstützung von dynamischem SQL in Oracle SQLJ, die erst ab der Oracle9*i* SQLJ-Release verfügbar sind.

Die Purchase Order- und Observation-Schemas wurden für das Atlantic Oceanographic and Meteorological Laboratory (AOML) in Miami, Florida, entwickelt, einem Umwelterforschungslabor der National Oceanic and Atmospheric Administration (NOAA), die zum amerikanischen Wirtschaftsministerium gehört. Das Observation-Schema wurde speziell für die Hurricane Research Division (HRD) von AOML entwickelt, und wird derzeit von HRD als Teil des H*WIND-Systems eingesetzt, das in Echtzeit Windflächenberechnungen für die Wetterexperten des National Hurricane Centers (NHC) und des FEMA Hurricane Liaison Teams von NHC erstellt.

Aufbau des Buchs

Diese Buch bietet Ihnen eine gut integrierte und schrittweise Darstellung von SQLJ und der Oracle9*i* JavaVM, wobei Sie von den einfacheren bis zu den fortgeschrittenen Konzepten begleitet werden und am Ende in der Lage sein sollten, Ihre neu erworbenen SQLJ-Techniken in die Praxis umzusetzen.

Das vorliegende Buch behandelt die folgenden Themenbereiche:

- Traditionelle und objektrelationale Einsatzszenarien
- Vollständige praxisbezogene Programme mit Erklärung wichtiger Code-Zeilen
- Umfangreiche Beispiele für Einsatzszenarien: clientseitig, serverseitig, Client/Server und n-schichtig
- Veranschaulichung der engen Integration von SQLJ und Oracle9*i* JavaVM
- Ein Tutorial zu den Grundlagen (für diejenigen, für die Oracle oder Java neu ist)

Teil I: Grundlagen von Oracle9*i* SQLJ

Dieser einführende Teil beginnt mit der Darstellung von SQLJ aus 10000 Meter Höhe, und gleitet dann langsam durch die Wolken wieder zurück auf den Boden. Die Grundlagen der SQLJ-Syntax, ihre Werkzeuge, Implementierung und ihr Einsatz werden hier im Detail behandelt und mit umfangreichen praxisbezogenen Code-Beispielspielen veranschaulicht. Am Ende dieses Abschnitts sollten Sie genau wissen, wie man SQLJ-Programme erstellt, die Daten in der Datenbank bearbeiten und daraus abrufen. Ebenso sollten Sie wissen, wie man Datenbankobjekte anlegt und löscht. Die einzelnen Kapitel sind:

- Kapitel 1: Einführung in Oracle9*i* SQLJ
- Kapitel 2: Entwicklung von SQLJ-Programmen
- Kapitel 3: Grundlagen der SQLJ-Programmierung

Teil II: Fortgeschrittene SQLJ-Konzepte für relationale Datenbankanwendungen

In Teil II erfahren Sie, wie wirklich robuste, skalierbare SQLJ-basierte Lösungen erstellt werden, indem man die Vorteile der engen Integration von SQLJ in Oracle9*i* nutzt. SQLJ verbindet sich mit PL/SQL (dem proprietären prozeduralen SQL von Oracle) zur Stored Procedure-Sprache von Oracle9*i*. In diesem Abschnitt lernen Sie vielzählige Methoden kennen, um hundertprozentige Java-Datenbanklösungen über SQLJ-Stored Procedures und Trigger zu erstellen. Die Kapitel behandeln im Einzelnen:

- Kapitel 4: Entwicklung von gespeicherten SQLJ-Programmen und Triggern
- Kapitel 5: Fortgeschrittene Konzepte beim Einsatz von SQLJ
- Kapitel 6: Fortgeschrittene SQLJ-Funktionalitäten
- Kapitel 7: Fortgeschrittene SQLJ-Funktionen

Teil III: SQLJ und objektorientierte Entwicklung

Java ermöglicht das objektorientierte Design und die Entwicklung auf Anwendungsebene für die breite Masse. Das objektrelationale Modell von Oracle9*i* ermöglicht, dass die Objektorientierung auch auf der logischen Ebene besteht. Teil III behandelt die objektrelationalen Funktionen von Oracle9*i*, und wie diese über SQLJ-basierte Methoden und Komponenten genutzt werden können. Die Kapitel dazu sind:

- Kapitel 8: Objektrelationale Verarbeitung mit SQLJ
- Kapitel 9: Einsatz von SQLJ-Objekten in kommerziellen und wissenschaftlichen Anwendungen

Teil IV: Der effiziente Einsatz von SQLJ

In Teil IV können Sie Ihr neu gewonnenes Wissen über SQLJ noch einmal verfeinern und die Anwendungen optimieren. Sie lernen bewährte Techniken zur Optimierung Ihrer SQLJ-Methoden und Komponenten kennen, indem Sie die robuste und hochgradig abstimmbare Oracle9*i*-Plattform einsetzen. Falls Ihnen diese Produktivitätsoptimierungen noch nicht ausreichen, können Sie mit weiteren Oracle-Werkzeugen, die explizit für das Internet-Computing und E-Business-Lösungen konzipiert wurden, ein weiteres Feintuning vornehmen. Zu den Kapiteln gehören:

- Kapitel 10: Performance-Tuning bei SQLJ-Anwendungen
- Kapitel 11: Entwicklungswerkzeuge von Oracle9*i*

Teil V: Anhänge

Für erfahrene Java- und Oracle-Anwendungsentwickler stellen diese Anhänge eine willkommene Referenz dar, auf die sie im Bedarfsfall immer zugreifen können. Die übrigen Leser erhalten eine schnelle Einführung in die Funktionsweise von Oracle SQL, Java, JDBC und SQLJ. Die Anhänge umfassen:

- Anhang A: Grundlagen von Oracle SQL
- Anhang B: Grundlagen von Java
- Anhang C: Einführung in Java Database Connectivity (JDBC)
- Anhang D: SQLJ-Kurzreferenz

Das Zielpublikum

Das Zielpublikum für dieses Buch sind Anwendungsentwickler und Datenbank-Designer, die beiden Gruppen, die am wahrscheinlichsten die Vorteile von SQLJ bei Anwendungslösungen (Anwendungsentwickler) oder in Stored Procedures und Triggern (Datenbank-Designer) nutzen werden. Zu diesem Personenkreis zählen wir auch diejenigen, die sich gerade in der Ausbildung zum Anwendungsentwickler, Datenbankdesigner oder Ähnlichem befinden und an einem fundierten Einstieg in das Thema interessiert sind.

JDBC-Progammierer werden nicht nur sehr schnell mit SQLJ vertraut werden, sondern auch eine sofortige Produktivitätssteigerung damit erzielen können. Sie können nun direkt sagen, was sie meinen, ohne dafür auf ein API zurückgreifen zu müssen. Als weiterer Bonus unterstützt SQLJ die SQL Code-Prüfung bei der Kompilierung, und nicht erst bei der Ausführung.

Wir möchten auch erfahrene Java-Programmierer, die noch keine Datenbankerfahrung haben, ermutigen, ihr Wissen in diesem Bereich mit diesem Buch enorm zu erweitern. SQLJ vereinfacht den Übergang aus monolithischen, nicht persistenten Java-Programmen auf n-schichtige, datenbankgesteuerte E-Business-Lösungen. Nehmen Sie hiermit den nächsten Entwicklungsschritt in Angriff und schließen Sie sich der SQLJ-Entwicklergemeinde an.

Obwohl dieses Buch vor allem auf Programmierer und Designer ausgerichtet ist, sollen andere Interessierte nicht ausgeschlossen werden. Wenn Sie etwas über die Verbindung von Java und SQL erfahren möchten, hoffen wir, dass Sie dieses Buch unabhängig von Ihrem Hintergrund interessiert. Sie wissen nie, wohin Ihre Sie Karriere führt – die richtige Chance und den geeigneten Katalysator vorausgesetzt.

SQL-Skripts zur Erstellung des Financial Purchase Order-Schemas

Mit dem folgenden SQL-Skript `createposchema.sql` erstellen Sie in der Oracle8*i*-Datenbank das Purchase Order-Schema:

```
-- File Name:  createposchema.sql
DROP TABLE DEPARTMENT_LIST CASCADE CONSTRAINTS
/
CREATE TABLE DEPARTMENT_LIST(
deptno       NUMBER(5),
shortname    VARCHAR2(6),
longname     VARCHAR2(20))
/
DROP TABLE ACCOUNT_LIST CASCADE CONSTRAINTS;
/
```

```
CREATE TABLE ACCOUNT_LIST (
accountno     NUMBER(5),
projectno     NUMBER(5),
deptno        NUMBER(5)),
PRIMARY KEY ( accountno ))
/
DROP TABLE EMPLOYEE_LIST CASCADE CONSTRAINTS
/
CREATE TABLE EMPLOYEE_LIST(
employeeno    NUMBER(7),
deptno        NUMBER(5),
type          VARCHAR2(30),
lastname      VARCHAR2(30),
firstname     VARCHAR2(30),
phone         VARCHAR2(10))
/
DROP TABLE CREDITCARD_LIST
/
CREATE TABLE CREDITCARD_LIST (
cardno            VARCHAR2(15),
employeeno        NUMBER(7),
expirationdate    DATE)
/
DROP TABLE CHECKACCOUNT_LIST
/
CREATE TABLE CHECKACCOUNT_LIST(
accountno     NUMBER(5),
employeeno    NUMBER(7))
/
DROP TABLE VENDOR_LIST
/
CREATE TABLE VENDOR_LIST(
vendorno    NUMBER(6),
name        VARCHAR2(30),
address     VARCHAR2(20),
city        VARCHAR2(15),
state       VARCHAR2(15),
vzip        VARCHAR2(15),
country     VARCHAR2(15))
/
DROP TABLE PROJECT_LIST
/
CREATE TABLE PROJECT_LIST (
projectno        NUMBER(5),
projectname      VARCHAR2(20),
start_date       DATE,
amt_of_funds     NUMBER,
PRIMARY KEY( projectno );
/
```

```
DROP TABLE PURCHASE_LIST
/
CREATE TABLE PURCHASE_LIST (
requestno        NUMBER(10),
employeeno       NUMBER(7),
vendorno         NUMBER(6),
purchasetype     VARCHAR2(20),
checkno          NUMBER(11),
whenpurchased    DATE)
/
DROP TABLE LINEITEM_LIST
/
CREATE TABLE LINEITEM_LIST (
requestno        NUMBER(10),
lineno           NUMBER(5),
projectno        NUMBER(5),
quantity         NUMBER(5),
unit             VARCHAR2(2),
estimatedcost    NUMBER(8,2),
actualcost       NUMBER(8,2),
description      VARCHAR2(30))
/
```

Mit dem folgenden Codeausschnitt erstellen Sie die Contraints für das Purchase Order-Schema:

```
-- File Name: poconstraints.sql
alter table DEPARTMENT_LIST
  ADD CONSTRAINT deptno_pk PRIMARY KEY(deptno)
  using index tablespace INDX
/
ALTER TABLE ACCOUNT_LIST
  ADD CONSTRAINT acc_deptno_fk
  FOREIGN KEY(deptno)
  REFERENCES DEPARTMENT_LIST(deptno)
  using index tablespace INDX
/
ALTER TABLE EMPLOYEE_LIST
  ADD CONSTRAINT employeeno_pk PRIMARY KEY(employeeno)
  using index tablespace INDX
/
ALTER TABLE EMPLOYEE_LIST
  ADD CONSTRAINT emp_deptno_fk
  FOREIGN KEY(deptno)
  REFERENCES DEPARTMENT_LIST(deptno)
  using index tablespace INDX
/
ALTER TABLE CREDITCARD_LIST
  ADD CONSTRAINT cardno_pk PRIMARY KEY(cardno)
  using index tablespace INDX
/
```

```
ALTER TABLE CREDITCARD_LIST
  ADD CONSTRAINT credit_employeeno_fk
  FOREIGN KEY(employeeno)
  REFERENCES EMPLOYEE_LIST(employeeno)
  using index tablespace INDX
/
ALTER TABLE CHECKACCOUNT_LIST
  ADD CONSTRAINT accountno_pk PRIMARY KEY(accountno)
  using index tablespace INDX
/
ALTER TABLE CHECKACCOUNT_LIST
  ADD CONSTRAINT check_employeeno_fk
  FOREIGN KEY(employeeno)
  REFERENCES EMPLOYEE_LIST(employeeno)
  using index tablespace INDX
/
ALTER TABLE vendor_list
  ADD CONSTRAINT vendorno_pk PRIMARY KEY(vendorno)
  using index tablespace INDX
/
ALTER TABLE Purchase_list
  ADD CONSTRAINT requestno_pk PRIMARY KEY(requestno)
  using index tablespace INDX
/
ALTER TABLE LINEITEM_LIST
  ADD CONSTRAINT lineno_pk
  PRIMARY KEY(requestno,lineno,projectno)
  using index tablespace INDX
/
```

Mit dem folgenden Code werden Sequences für das Purchase Order-Schema erstellt:

```
-- File Name: posequences.sql
CREATE SEQUENCE deptno_SEQ
  START WITH 200
  INCREMENT BY 1
/
CREATE SEQUENCE projectno_SEQ
  START WITH 300
  INCREMENT BY 1
/
CREATE SEQUENCE employeeno_SEQ
  START WITH 100
  INCREMENT BY 1
/
CREATE SEQUENCE accountno_SEQ
  START WITH 1000
  INCREMENT BY 1
/
```

```
CREATE SEQUENCE cardno_SEQ
  START WITH 311200
  INCREMENT BY 1
/
CREATE SEQUENCE vendorno_SEQ
  START WITH 400
  INCREMENT BY 1
/
CREATE SEQUENCE requestno_SEQ
  START WITH 500
  INCREMENT BY 1
/
CREATE SEQUENCE lineno_SEQ
  START WITH 1
  INCREMENT BY 1
/
```

SQL-Skripts zur Erstellung des wissenschaftlichen Observation-Schemas

Mit dem folgenden SQL-Skript `createobjschema.sql` erstellen Sie das wissenschaftliche Observation-Schema in der Oracle8*i*-Datenbank:

```
-- File Name:  createobjschema.sql
DROP TABLE passed_observation_list
/
DROP TYPE passedObsArray
/
DROP TYPE passedObs
/
DROP TABLE oceanic_observation_list
/
DROP TYPE oceanic_observation_TYPE
/
DROP TYPE oceanic_observation
/
DROP TABLE QC_EVENT_LIST
/
DROP TYPE QUALITY_CONTROL_TYPE
/
DROP TABLE ATMOSPHERIC_EVENT_LIST
/
DROP TYPE ATMOSEVENT
/
DROP TABLE SCIENTIST_LIST
/
DROP TYPE SCIENTIST
/
```

```
DROP TABLE PLATFORM_TYPE_LIST
/
DROP TYPE PLATFORM_TYPE
/
CREATE TYPE PLATFORM_TYPE AS OBJECT(
key_id       NUMBER(8),
type         VARCHAR2(50),
description  VARCHAR2(50))
/
CREATE TABLE PLATFORM_TYPE_LIST OF PLATFORM_TYPE
/
CREATE TYPE SCIENTIST AS OBJECT(
usr_id          NUMBER(6),
lastname        VARCHAR2(20),
firstname       VARCHAR2(20),
platform_id     NUMBER,
for_platform    REF PLATFORM_TYPE)
/
CREATE TABLE SCIENTIST_LIST OF SCIENTIST
/
CREATE TYPE atmosevent AS OBJECT(
key_id            NUMBER(8),
when_t            DATE,
name              VARCHAR2(30),
type              VARCHAR2(20),
refkey            NUMBER(8),
transformed_to    REF atmosevent)
/
CREATE TABLE atmosevent_list OF atmosevent
/
CREATE TYPE oceanic_observation AS OBJECT(
latitude_deg              NUMBER(10,4),
longitude_deg             NUMBER(10,4),
windspeed_mps             NUMBER(10,4),
adj_windspeed_mps         NUMBER(10,4),
wind_direction_deg        NUMBER(6),
pressure_mb               NUMBER(6))
/
CREATE OR REPLACE TYPE oceanic_observation_type AS OBJECT(
obs_id             NUMBER(8),
when_t             DATE,
at_time            CHAR(8),
station_id         NUMBER(6),
produced_id        NUMBER(8),
produced_by        REF PLATFORM_TYPE,
obsobj             oceanic_observation)
/
```

```
ALTER TYPE oceanic_observation_type REPLACE AS OBJECT (
obs_id              NUMBER(8),
when_t              DATE,
at_time             CHAR(8),
station_id          NUMBER(6),
produced_id         NUMBER(8),
produced_by         REF PLATFORM_TYPE,
obsobj              oceanic_observation,
    member function get_platform_type
      return platform_type,
    pragma restrict_references( get_platform_type, wnds, wnps ) );
/CREATE OR REPLACE TYPE BODY oceanic_observation_type IS
  member function get_platform_type RETURN PLATFORM_TYPE IS
    pt platform_type;
  begin
    -- Select the PLATFORM_TYPE_LIST record whose OID matches the OID
    --   in the produced_by field of the oceanic_observation_type
    --   instance.
    --
    -- VALUE(pt1) returns the object type record from the
    --   table.  * wildcard or list of platform_type fields
    --   would be incompatible with pt variable.
    --
    SELECT VALUE(ptl ) INTO pt FROM PLATFORM_TYPE_LIST ptl
      WHERE REF( ptl ) = produced_by;
    return pt;
  end;
end;
/
-- List of all oceanic observations by date, time, and platform type
CREATE table oceanic_observation_list OF oceanic_observation_type
/
-- use qc_id_seq to update quality_control_event qc_id
CREATE TYPE quality_control_event AS OBJECT(
qc_id               NUMBER(8),
when_t              DATE,
at_time             CHAR(8),
event_id            NUMBER(8),
for_event           REF atmosevent,
whom_id             NUMBER(6),
by_whom             REF scientist)
/
CREATE TABLE qc_event_list OF quality_control_event
/
CREATE TYPE passedObs AS OBJECT(
obsid         NUMBER(8),
passed        CHAR(1))
/
CREATE TYPE passedObsArray AS TABLE OF passedObs
/
```

```
CREATE TABLE passed_observation_list(
passed_id      NUMBER(5),
qcid           NUMBER(8),
when_t         DATE,
at_time        CHAR(8),
idobj          passedObsArray)
NESTED TABLE idobj STORE AS pobsid_list
/
ALTER TABLE pobsid_list
STORAGE (MINEXTENTS 1 MAXEXTENTS 20)
/
```

Mit dem folgenden Code erstellen Sie die Constraints für das Observation-Schema:

```
-- File Name: objconstraints.sql
ALTER TABLE PLATFORM_TYPE_LIST
 ADD CONSTRAINT PT_KEY_ID_PK PRIMARY KEY(KEY_ID)
 using index tablespace INDX
/
ALTER TABLE SCIENTIST_LIST
 ADD CONSTRAINT SL_USR_ID_PK PRIMARY KEY(USR_ID)
 using index tablespace INDX
/
ALTER TABLE ATMOSEVENT_LIST
 ADD CONSTRAINT AL_KEY_ID_PK PRIMARY KEY(KEY_ID)
 using index tablespace INDX
/
ALTER TABLE oceanic_observation_list
 ADD CONSTRAINT O_OBS_ID_PK PRIMARY KEY(OBS_ID)
 using index tablespace INDX
/
ALTER TABLE qc_event_list
 ADD CONSTRAINT QC_ID_PK PRIMARY KEY(QC_ID)
 using index tablespace INDX
/
ALTER TABLE qc_event_list
 ADD CONSTRAINT qc_whom_id_fk
 FOREIGN KEY(whom_id)
 REFERENCES scientist_list(usr_id)
 ON DELETE CASCADE
/
ALTER TABLE passed_observation_list
 ADD CONSTRAINT passed_id_pk PRIMARY KEY (passed_id)
 using index tablespace INDX
/
ALTER TABLE passed_observation_list
 ADD Constraint po_qc_id_fk
 FOREIGN KEY(qcid)
 REFERENCES qc_event_list(qc_id)
 ON DELETE CASCADE
/
```

```
alter table passed_observation_list
 add constraint passed_qcid_ukey UNIQUE(qcid)
 using index tablespace INDX
/
alter table passed_observation_list
 modify (qcid NOT NULL)
/
```

Mit dem folgenden Codeausschnitt erstellen Sie die Sequences für das Observation-Schema:

```
-- File Name: objsequences.sql
-- key_id sequence for platform_type
CREATE SEQUENCE PT_key_SEQ
 START WITH 1
 INCREMENT BY 1
/
-- usr_id sequence for scientist
CREATE SEQUENCE USERSEQ
 START WITH 1
 INCREMENT BY 1
/
-- key_id sequence for atmosevent
CREATE SEQUENCE atm_key_seq
 START WITH 1
 INCREMENT BY 1
/
CREATE SEQUENCE obsid_seq
 START WITH 1
 INCREMENT BY 1
/
-- qc_id sequence for quality_control_event
CREATE SEQUENCE qc_id_seq
 START WITH 1
 INCREMENT BY 1
/
-- passed_id sequence for passed_observation
CREATE SEQUENCE passed_id_seq
 START WITH 1
 INCREMENT BY 1
/
```

Schreibweisen in diesem Buch

Diese Buch verwendet folgende Konventionen:

- Klassen werden in der Schriftart Courier dargestellt.
  Beispiel Die Java-Standardklasse `Java.lang.*`
- Datentypen werden in der Schriftart Courier dargestellt.
  Beispiel Der Datentyp `REF CURSOR`
- Dateinamen und -erweiterungen werden kleingeschrieben und in der Schriftart Courier dargestellt.
  Beispiel `.class`-Dateien; die `.ser`-Erweiterung
- Funktionen und Prozeduren werden in der Schriftart Courier dargestellt.
  Beispiele `InsertPurchaseOrder(); a function GetObsId()`
- SQL-Schlüsselwörter werden großgeschrieben und in der Schriftart Courier dargestellt.
  Beispiele `CREATE TABLE; INSERT; DELETE`
- Datenbanktabellennamen werden großgeschrieben und in der Schriftart Courier dargestellt.
  Beispiel `PASSED_OBSERVATION`
- Java-Schlüsselwörter werden in der Schriftart Courier und fett dargestellt.
  Beispiel **`public`**

Feedback an die Autoren

Die Autoren freuen sich über Kommentare und Anregungen zur Qualität und Nützlichkeit des Buchs. Ihre Meinung ist wichtig für uns. Sie können uns per E-Mail erreichen:

- Nirva Morisseau-Leroy über nmorisseauleroy@data-i.com
- Martin K. Solomon über marty@cse.fau.edu
- Gerald P. Momplaisir über gmomplaisir@data-i.com

Online-Zugriff auf die Beispiele

Die Schema-Skripts und den Programmquellcode finden Sie auf der CD-ROM, unter www.data-i.com sowie unter http://www.osborne.com.

Haftungsausschluss

Die hier vorgestellten Programme sind nicht dazu gedacht, in inhärent gefährlichen Anwendungen eingesetzt zu werden. Es obliegt der Verantwortlichkeit des Lesers, die entsprechenden Ausfallsicherungs-, Backup-, Redundanz- und sonstigen Maßnahmen zu treffen, die den sicheren Einsatz derartiger Anwendungen garantieren.

Über die CD-ROM

Die diesem Buch beiliegende CD enthält mehrere Codebeispiele aus den Kapiteln dieses Buchs. Hierzu gehören:

Kapitel 2: Sie lernen, wie man SQLJ-Programme schreibt, die Daten in die Datenbanktabellen einfügt und mit SELECT-Anweisungen Daten aus Datenbanktabellen abruft. Sie lernen auch den SQLJ-Übersetzungsprozess kennen, die Struktur der SQLJ-Befehlszeile und wie Translatoroptionen mit Attributdateien (statt über die SQLJ-Befehlszeile) festgelegt werden.

Kapitel 3: Sie erfahren, wie man benannte und positionale SQLJ-Iteratoren deklariert und ausführbare SQLJ-Anweisungen kodiert: SQLJ DDL-, SQLJ Nicht-`SELECT` DML-Befehle, SQLJ-Transaktionssteuerungsbefehle, ausführbare SQLJ-Anweisungen in anonymen PL/SQL-Blöcken und Stored Procedure- und Stored Function-Aufrufen, SQLJ SELECT-Anweisungen.

Kapitel 4: Sie lernen, wie man gespeicherte SQLJ-Programme und Trigger entwickelt.

Kapitel 5: Sie erfahren, wie man SQLJ-Verbindungskontextinstanzen deklariert und einsetzt, Sie verwenden `javax.sql.DataSource`-Objekte in SQLJ-Programmen und setzen SQLJ-Programme auf dem Client (Java/SQLJ-Applets und Java/SQLJ-Anwendungen), in der mittleren Schicht und in der Oracle-Datenbank ein.

Kapitel 6: Sie lernen, wie man die SQLJ-Iteratoren `ResultSetIterator`, benannte Iteratoren, positionale und `ScrollableResultSetIterator`-Iteratoren mit Rollbalken deklariert und einsetzt. Ebenso erfahren Sie, wie man dynamisches SQL in SQLJ-Anweisungen kodiert.

Kapitel 7: Sie lernen die Verwendung von SQLJ-Streamklassen kennen, erstellen Multithreaded SQLJ-Programme, lernen die Interaktionen zwischen SQLJ und JDBC kennen sowie die Unterklassenbildung bei SQLJ-Iteratoren.

Kapitel 8: Sie erfahren, wie man benutzerdefinierte SQL-Objekttypen und benutzerdefinierte Collection-Typen in Oracle8*i*/9*i* definiert, wie man SQL-Objekttypen und SQL Collection-Typen in Oracle9*i* SQLJ verarbeitet, und wie `SQLData` in SQLJ-Programmen eingesetzt wird.

Kapitel 9: Sie erfahren, wie man ein komponentenbasiertes SQLJ-Objekt entwirft und entwickelt, eine SQLJ-Komponente mit der Java Remote Method Invocation (RMI) verwendet, ein Enterprise JavaBeans-Komponentenobjekt mit einer SQLJ-Implementierung einsetzt, und wie man ein CORBA-Komponentenobjekt mit einer SQLJ-Implementierung einsetzt.

Kapitel 10: Sie lernen die unterstützenden Funktionen der Oracle-Leistungsoptimierungen kennen, entwickeln effiziente SQLJ-Programme, die die Oracle-Leistungsoptimierungen implementieren und tunen SQL-Anweisungen über SQLJ mit dem Oracle Optimizer. (Deaktivierung des `auto-commit`-Modus, Vorababruf von Zeilen, Aktualisierung im Stapelverarbeitungsbetrieb, Ablegen von Anweisungen im Cache, Spaltendefinitionen und Definitionen der Parametergröße.)

Kapitel 11: Sie erfahren, wie man mit dem Oracle JDeveloper SQLJ-Programme entwickelt.

Anhang B: Hier werden Java-Konstrukte vorgestellt, um Java-Anwendungen und -Applets zu entwickeln, die auf eine Oracle-Datenbank zugreifen.

Anhang C: Sie erfahren, wie man mit Java-Konstrukten JDBC-Anwendungen und -Applets entwickelt, die auf eine Oracle-Datenbank zugreifen.

Anhang D: In diesem Anhang finden Sie eine Zusammenfassung von SQLJ. Hier können Sie die SQLJ-Syntax und die in den verschiedenen Kapiteln gelernten Konzepte nachschlagen oder eine Übersicht über die Grundlagen von SQLJ erhalten.