Oracle技术系列丛书

Oracle9*i* SQLJ程序设计

Nirva Morisseau-Leroy
（美） Martin K.Solomon 著
Gerald P.Momplaisir

田蕴哲 丁 天 牛志奇 等译

机械工业出版社
China Machine Press

译 者 序

Oracle提供了Java的嵌入式静态和动态SQL，提供了用Java建立数据库应用程序的最简单和最优雅的方式。作为一个被ANSI/ISO标准化组织所接受并被所有主要数据库厂商支持的业界标准，SQLJ提供了一种非常方便的方法来构建各种不同类型的程序——数据库存储过程、触发器、企业JavaBeans（EJB）、CORBA服务甚至applet。在Oracle对SQLJ的全面广泛支持下，用户可以快速、高效地开发结合Java与Oracle数据库的应用程序，并将这些应用程序部署在Internet上。

除SQLJ基本原理外，本书还深入介绍了许多内容，提供了许多例子程序和解决方案。不仅介绍了SQLJ功能（如动态SQL、各种各样的迭代器、各种不同的上下文等等），还包括Oracle SQL的特性（如LOB类型、集合类型、用户定义类型以及许多其他特性），甚至任何一个Java编程模型特性（应用程序、applet、RMI的使用、CORBA、EJB、JSP和Java存储过程）。本书还向读者介绍了各种基本的Oracle Java工具，从JDeveloper集成开发环境到SQLJ和JPublisher命令行实用程序，以及许多程序开发和应用过程中所需的其他工具。总而言之，对所有使用Java和SQL的程序员来说，本书都是非常实用而宝贵的资源！

我们很荣幸能够有机会承担本书的翻译工作。在翻译过程中，我们经常为一句话、一个术语进行反复的讨论，到处查找资料，力图使本书的翻译能正确、贴切地反映原文的意思，同时注意使句子、段落符合中国人的语言习惯。我们真挚地希望读者能从本书中有所收获，这是作者的初衷，也是我们良好的愿望！

本书由田蕴哲、丁天、牛志奇等组织翻译，万方工作室的全体同仁都参加了本书的翻译、校正和输入等工作，他们是：田蕴哲、丁天、牛志奇、刘之望、刘砚、黄春、丁献胜、葛文丽、罗锋、罗浩、王宾、赵文、夏宣欣、李玲、孙庆楠、田宛析、龚膂娜、马万军、马秀芬、田军、牛献忠、金百万、薛鹏飞、叶之哲、邓小燕、邢倩、王玉、李照军、刘思彬、钱凯斌、赵儒策、江南、李浩天、王凌、李锡林、张芝莉、范春蕾、袁垒雷、邓笛涛、李林鸽、聂宛、王笑天、李度飞、天鹏等。本书的出版是集体劳动的结晶，在此特别感谢万方工作室的全体工作人员。

由于时间仓促，且译者经验和水平有限，译文难免有不妥之处，恳请读者批评指正！

万方工作室

2001年11月

序 1

Java由于其强大的功能、可移植性和高效率而成为构建应用程序的一种主要语言。它是仅有的为Internet而设计的先进的面向对象编程语言。为了用Java构建企业级应用，开发人员还需要两个工具，即一个运行Java的高度可伸缩的服务器环境，以及在Java和SQL之间的一个简单而又易于实现的编程链接，世界上绝大多数数据都是以SQL定义和存储的。

Oracle公司认识到Internet计算模型以及这两种基本需求的缺乏，进而作出了重大的战略投资：Oracle Java虚拟机（JVM）和SQLJ。

为了给开发人员提供一个可伸缩性强、高使用率和高性能的Java服务器环境（即Oracle JVM），Oracle公司直接将高度专业化的Java虚拟机设计到Oracle9*i*——Internet数据库。在完全兼容Java标准的基础上，Oracle JVM能支持成千上万个并发客户，而且其硬件配置可在很广的范围内变化（高可伸缩性）。

Oracle的另一个投资就是SQLJ，或称为Java的嵌入式静态和动态SQL。它提供了用Java建立数据库应用程序的最简单和最优雅的方式。作为一个被ANSI/ISO标准化组织所接受并被所有主要数据库厂商支持的业界标准，SQLJ提供了一种非常方便的方法来构建各种不同类型的程序——数据库存储过程、触发器、企业JavaBeans、CORBA服务，甚至applet。在Oracle对SQLJ的全面广泛支持下，用户可以快速、高效地开发综合Java与Oracle数据库的应用程序，并将这些应用程序部署在Internet上。

毋庸质疑，Nirva Morisseau-Leroy、Martin K. Solomon和Gerald P. Momplaisir所编写的这本书就是掌握SQLJ编程的必备工具。它首先介绍SQLJ的重要性和怎样着手建立SQLJ应用程序。然后介绍了SQLJ编程的核心背景知识，其中包括怎样将SQLJ与SQL、PL/SQL、Java和JDBC结合起来等内容。最后，本书提供了一系列精心编制的高级编程内容，涉及到服务器端SQLJ编程，将SQLJ与Oracle数据库的对象－关系工具相结合，以及用SQLJ进行分布式系统开发等，其中后者又涉及到远程方法调用（RMI）、CORBA服务和企业JavaBeans。

我非常欣赏这本书的实用性，其丰富完整的内容和大量实例可指导读者利用SQLJ开发现实应用程序。我对本书的出版感到非常兴奋。我相信读者会发现这是一本不可或缺的SQLJ和Oracle JVM使用学习指南。

Thomas Kurian

Oracle公司Oracle9*i* AS和电子商务副总裁

序　2

让我们直接切入问题的本质：

你想使用最好的、最高效的计算机语言，直接访问最好的数据系统，编写可运行在客户机、服务器和中间层上的应用程序吗？你想用最容易和最高效的方法来实现这一点吗？

如果是，那么本书就能帮你实现这些愿望。

- Java是一种用于企业计算（正被支持者和反对者广泛接受）的革命性的、高效的且可移植的编程语言。
- Oracle是数据库，而SQL则是用于访问和操纵Oracle数据库中数据的语言。Oracle9*i*给用户提供了一整套可以对用户的商业数据进行建模的特性（从大对象类型到全部对象-关系功能）。而且，用户还可以使用数据库服务器中的Oracle9*i* JVM来运行Java程序，或用Java来实现存储过程或触发器。因此，用户所选择的应用程序运行位置（客户机、服务器或中间层）与用户所选择的Java编程模型（如直接数据库访问、CORBA、EJB或JSP等）无关，这一点给用户提供了最大的灵活性。
- SQLJ通过在Java程序中直接嵌入SQL的方法将Java与SQL结合起来。这是编写简单、可靠的Java程序（这些程序通过SQL直接访问数据库）的最容易而又最高效的方法。SQLJ可以使用户将精力集中在具体的应用设计上，而将那些低级的细节和对SQL语句的检查交由计算机完成。

下列语句的目的是给用户提供一个SQLJ语句的示例，以说明SQLJ的本质。当然，前提是读者对SQL和Java已经有一定的了解。

```
int no;  String name = "SMITH";
#sql { SELECT empno INTO :no FROM emp WHERE ename=:name };
```

附录A和B分别给出了SQL和Java的简要介绍。

有哪些新内容

本书涉及到了Oracle9*i* SQLJ，这一版本的SQLJ非常令我兴奋！Oracle以前提供的SQLJ只支持静态SQL——在静态SQL语句中，除了绑定表达式的数据值之外的所有一切都必须在翻译时确定。如果要使用动态SQL，就必须用JDBC API，这就意味着用户又要花时间来学习它（且不说还要掏钱买一本500页的大书）。假设用户再也不用转到JDBC上（即使只是有时），那么使用SQLJ这一更为容易、更为高效的语言的最大优点是什么呢？

再也不需用JDBC了！SQLJ 9*i*直接支持动态SQL的所有要求。无论何时，程序员们都可以同时利用两个领域的优点：嵌入式SQL的简单明了以及动态SQL的强大功能。读者可以猜测出下列SQLJ语句做什么吗？

```
double money;  String name="SMITH";
```

```
String column = "sal";  String table = "emp";
#sql { SELECT :{column} INTO :money FROM :{table} WHERE ename=:name };
```

不要误解JDBC。如果读者已经开始用JDBC来编程，你也许会觉得已经被SQLJ爱好者忽略了。是的，我们因为将大部分精力都集中在SQL上，而像个被告一样感到有罪。但对于Oracle9*i*情况却不是这样。可以立即直接且更为准确地用9*i* SQLJ重写任何原来的JDBC逻辑。下面再给出一个例子，对于JDBC程序员来说，这个例子看起来似乎有点熟悉。

```
double min = 1500.0; String name; int no;
ResultSetIterator rsi;
#sql rsi = { SELECT ename, empno FROM emp WHERE sal > :min };
while (rsi.next()) {
   #sql { FETCH CURRENT FROM :rsi INTO :name, :no }; ...
}
rsi.close();
```

从本书能学到什么

SQLJ简单、直截，而且需要更少的类型。为什么要买这样一本大部头的书呢？难道不能只通过几页纸就让读者明白它的基本原理？我们会失去编程过程中的乐趣吗？

不可能这么简单！但读者确实可以在附录D中找到SQLJ的概述。本书除了SQLJ基本原理外，还提供了大量更深入的内容。本书中有大量的例子程序和解决方案，它们可以使你开始一项新的事业。请重用这些例子程序，而不是重新设计它们，而且要学会从这些为数众多的程序中学到点什么。本书还涉及了大量的特性。作者几乎没有遗漏任何东西：无论它是SQLJ功能（如动态SQL、各种各样的迭代器、各种不同的上下文等等），还是Oracle SQL中的特性（包括LOB类型、集合类型、用户定义类型以及许多其他特性），亦或是任何一个Java编程模型特性（应用程序、applet、RMI的使用、CORBA、EJB、Java Server Pages和Java存储过程）。本书还向读者介绍了各种基本的Oracle Java工具，从JDeveloper集成开发环境到SQLJ和JPublisher命令行实用程序，以及许多程序开发和应用过程中所需的其他工具。对于所有使用Java和SQL的程序员来说，这本书是一个非常好用的无价之宝！

最后，代表我自己讲几句。希望读者不断地向Oracle公司提供关于SQLJ的反馈意见，以便我们能做得更好。我们真心地希望你在使用SQLJ的过程中体会到快乐，正如我们在创建SQLJ的过程中体会到的一样。期待与你在Oracle技术网络（technet.oracle.com）上见！

Ekkehard Rohwedder
Oracle公司Java平台组
SQLJ首席开发工程师

前　言

在本书的第一部分和第二部分，将开发操纵“纯”关系数据库的SQLJ程序，即一个名为Purchase Order（采购订单）模式的财务数据库模式。采购订单关系数据库模式是《*Design of a Financial Administrative System Using the Semantic Binary Model*》一书所给出的数据库设计的一部分。在本书的第三部分，将开发操纵对象-关系数据库的SQLJ程序，即一个名为Observation（观测）模式的科学数据库模式。“观测”对象-关系数据库模式是《*Atmospheric Observations, Analyses, and the World Wide Web Using a Semantic Database*》一书中给出的科学数据库设计的一部分。注意，本书开发的所有程序都已经在Oracle8*i* 8.1.5、8.1.6和8.1.7版的数据库服务器上测试通过，而且其中一些已经应用在Oracle9*i* Internet应用服务器上(Oracle9*i* AS)。本书还涉及了一些只在Oracle9*i* SQLJ中实现的新概念（特别是Oracle SQLJ中的对动态SQL的支持）。

“采购订单”和“观测”模式都是应位于佛罗里达州迈阿密的大西洋海洋地理及气象实验室（the Atlantic Oceanographic and Meteorological Laboratory，AOML）的要求而设计的。AOML是隶属于美国商业部（Department of Commerce，DOC）的国家海洋和大气局（the National Oceanic and Atmospheric Administration，NOAA）的一个环境研究实验室（Environmental Research Laboratory，ERL）。特别是，“观测”数据库模式是为AOML的飓风研究室（Hurricane Research Division，HRD）而设计的，目前已由HRD实现并成为H*WIND系统的一部分。H*WIND系统用于为国家飓风中心（National Hurricane Center，NHC）的预报员和NHC的FEMA飓风组提供实时洋面风速分析。

本书的组织结构

本书将带领读者逐步掌握SQLJ和Oracle9*i* JVM，然后自然地过渡到SQLJ的高级内容，并最终让读者可以开始在现实应用中使用崭新的SQLJ技术。

本书的特点：

- 给出了传统的关系和对象-关系的使用情形。
- 逐行检查过的完整、真实的程序。
- 全方位的应用场景——客户端、服务器端、客户机/服务器以及n层。
- 演示了SQLJ和Oracle9*i* JVM之间的紧密集合。
- 对于那些刚接触Oracle或Java的读者而言，是一本基本原理指南。

第一部分“Oracle9*i* SQLJ基础”　该部分先从整体上论述SQLJ，然后渐渐地深入到各子部分。该部分详细介绍了SQLJ语法、工具、实现和应用的基本原理，并通过全面真实的范例代码来增强读者的理解。学习完这一部分，读者应该可以掌握如何编写可修改和检索数据库中的数据的SQLJ程序，以及如何创建和撤销数据库对象。这一部分包括第1～3章。

第二部分“用于关系型数据处理的高级SQLJ”　在第二部分，我们将学习如何利用SQLJ与Oracle9*i*的紧密集成来建立真正健壮的、可伸缩的基于SQLJ的解决方案。SQLJ可以结合PL/SQL（Oracle特有的过程化SQL语言）一起作为Oracle9*i*的存储过程语言。这部分内容将告诉读者以多种方式通过SQLJ存储过程和触发器来构建100%的Java数据库解决方案。这一部分

包括第4～7章。

第三部分“SQLJ及对象部署” Java推广了应用层次的面向对象设计和开发。Oracle9*i*的对象-关系模型可以使得在逻辑层也进行面向对象设计。该部分涉及到了Oracle9*i*的对象－关系特性，以及如何通过基于SQLJ的方法和组件来同这些特性进行接口连接。这一部分包括第8～9章。

第四部分“有效使用SQLJ” 基于对如何设计、实现和应用SQLJ代码的全新理解；第四部分将帮助读者把应用解决方案提升到一个更高的层次。通过使用稳定的和高可调整的Oracle9*i*平台，学习这些已被证实的技术，以优化用户的SQLJ方法和组件。如果读者认为使用SQLJ后的开发效率还不够高，那么读者就要学会如何通过那些专门为Internet计算和电子商务解决方案而设计的Oracle工具来进一步减轻开发负担。这一部分包括第10～11章。

第五部分“附录” 对于有经验的Java和Oracle应用程序开发人员来说，这些附录将是个非常好的参考指南，它快速而又粗略地介绍了Oracle SQL、Java、JDBC和SQLJ的原理机制。这一部分包括附录A至附录D。

本书的读者对象

本书的传统读者是应用程序开发人员和数据库设计人员，这两类人员最有可能在解决方案（应用程序开发人员）或存储过程和触发器（数据库设计人员）中使用SQLJ。这两类人员也包括那些立志要加入设计者和实现者行列的人，他们包括程序设计专业的已毕业的和未毕业的学生。

JDBC程序员不仅会很快适应SQLJ，而且还会立即体会到SQLJ所带来的高效率。现在如果没有API，他们都不知会怎么办。作为一个附加的优点，SQLJ还会在编译时而不是运行时对用户的SQLJ代码进行检查。

我们也鼓励那些还未大量接触数据库的资深Java程序员通过挖掘本书中的每个信息来拓宽他们的思路。SQLJ简化了从单层的、非永久的Java程序到n层的、数据库驱动的电子商务解决方案的迁移。了解SQLJ将使你紧跟技术发展的脚步。

虽然本书是给程序员和数据库设计人员阅读的，但其他感兴趣的读者也可以参考本书。如果想了解Java和SQL之间的关系，那么不论读者的背景如何，我们希望本书都可以对你有所帮助。只要有适当的机遇和动力，你的事业将前途无量。

创建商业采购订单数据库模式的SQL脚本

使用下面的createposchema.sql中的SQL脚本创建Oracle8*i* 数据库中的采购订单数据库模式：

```
-- 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))
/
```

用下列代码为采购订单模式创建约束：

```
-- 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
/
```

用下列代码为采购订单数据库模式创建序列：

```
-- 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
/
```

创建科学观测数据库模式的SQL脚本

使用下面的createobjschema.sql中SQL脚本来创建Oracle8*i* 数据库中的科学观测模式：

```
-- 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)
/
```

用下列代码为观测数据库模式创建约束：

```
-- 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)
/
```

用下列代码为观测数据库模式创建序列：

```
-- 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
/
```

本书使用的约定

本书使用如下的约定：

- 文件名和扩展名均小写，例如：.class文件，.ser扩展名。
- SQL关键字全部都用大写字母，例如：CREATE TABLE、INSERT、DELETE。
- 数据库表名字全部都用大写字母，例如：PASSED_OBSERVATION。
- 段落中的Java关键字用粗体字表示，例如：**public**。

向作者提供反馈

作者欢迎各位读者对本书的质量和有用性提出批评和建议。你的反馈对我们非常重要，可以通过电子邮件给出你的批评与建议：

Nirva Morisseau-Leroy：nmorisseauleroy@data-i.com

Martin K. Solomon：marty@cse.fau.edu

Gerald P. Momplaisir：gmomplaisir@data-i.com

检索在线例程

本书所有的模式脚本和程序源代码都可以在CD-ROM、www.data-i.com和http://www.osborne.com上找到。

声明

本书中所给出的程序并不是特意为在那些本来就不安全的应用中使用而设计的。读者自己有责任采取适当的保险、备份、冗余以及其他措施来确保安全地使用这样的应用程序。

关于本书的CD-ROM

本书附带的CD-ROM中包含了本书各章中的许多代码范例，它们包括：

第2章：了解如何编写向数据库表中插入数据和用SELECT语句从数据库表中检索数据的SQLJ程序。另外，该章还介绍了SQLJ翻译过程、sqlj命令行的结构以及如何使用属性文件（而不是sqlj命令行）来设置翻译器选项。

第3章：学习如何声明和使用SQLJ命名与位置迭代器，以及编写SQLJ可执行语句——SQLJ DDL、SQLJ非SELECT DML语句、SQLJ事务控制命令、匿名PL/SQL块和存储过程、函数调用

中的SQLJ可执行语句、SQLJ SELECT语句。

第4章：介绍如何开发SQLJ存储程序和触发器。

第5章：介绍如何声明和使用SQLJ连接上下文实例，如何在SQLJ程序中使用javax.sql.DataSource对象，以及如何在客户端、中间层和Oracle数据库中应用SQLJ程序（Java/SQLJ applet和Java/SQLJ应用程序）。

第6章：介绍如何声明和使用SQLJ ResultSetIterator迭代器、可滚动命名迭代器、可滚动位置迭代器和ScrollableResultSetIterator迭代器。此外，还介绍了如何在SQLJ语句中编写动态SQL。

第7章：介绍如何使用SQLJ流类，如何创建多线程的SQLJ程序，SQLJ和JDBC之间的互操作，以及将SQLJ迭代器作为子类。

第8章：介绍如何定义Oracle8*i*/9*i* SQL用户定义对象类型和用户定义集合类型，如何在Oracle9*i* SQLJ中处理SQL对象类型和SQL集合类型，如何在SQLJ程序中使用SQLData。

第9章：介绍如何设计与开发基于组件的SQLJ对象，如何通过Java远程方法调用（RMI）来部署SQLJ组件，如何通过SQLJ实现来部署企业JavaBeans组件对象，以及如何通过SQLJ实现来部署CORBA组件对象。

第10章：介绍Oracle的增强性能所支持的特性，如何开发实现Oracle增强性能的SQLJ程序，如何用Oracle优化器（关闭自动提交模式、行预取、批量更新、语句缓冲、列定义和参数大小定义）调整SQLJ中的SQL语句。

第11章：介绍如何使用Oracle JDeveloper来开发SQLJ程序。

附录B：介绍与开发访问Oracle数据库的Java应用程序和applet有关的Java结构。

附录C：介绍与开发访问Oracle数据库的JDBC应用程序和applet有关的Java结构。

附录D：该附录对SQLJ做了一个概括。可以在该附录中查看SQLJ的语法，增强读者对本书所学概念的理解，或者是从总体上回顾SQLJ的基本原理。