



دولت جمهوری اسلامی افغانستان
اداره تعلیمات تکنیکی و مسلکی
معاونیت امور اکادمیک
ریاست نصاب و تربیه معلم

PL/SQL

رشته: کمپیوټر سائنس - دیپارتمنټ: دیتاپیس
صنف ۱۴ - سمستر دوم

سال: ۱۳۹۹ هجری شمسی



شناسنامه کتاب

نام کتاب: PL/SQL

رشته: کمپیوتر ساینس

تدوین کننده: پوهندوی شعیب زرین خیل

همکار تدوین کننده: دیوه خیرخواه بصیری

- کمیته نظارت:
- ندیمه سحر رئیس اداره تعلیمات تخصصی و مسلکی
 - عبدالحمید اکبر معاون امور اکادمیک اداره تعلیمات تخصصی و مسلکی
 - حبیب الله فلاح رئیس نصاب و تربیه معلم
 - عبدالمتین شریفی آمر انکشاف نصاب تعلیمی، ریاست نصاب و تربیه معلم
 - روح الله هوتک آمر طبع و نشر کتب درسی، ریاست نصاب و تربیه معلم
 - احمد بشیر هیله من مسؤول انکشاف نصاب، پروژه انکشاف مهارت‌های افغانستان
 - محمد زمان پویا کارشناس انکشاف نصاب، پروژه انکشاف مهارت‌های افغانستان
 - علی خیر یعقوبی سرپرست مدیریت عمومی تألیف کتب درسی، ریاست نصاب و تربیه معلم

کمیته تصحیح:

• دوکتور نظر محمد بهروز

• محمد باقر موسوی

• محمد امان هوشمند مدیرعمومی بورد تصحیح کتب درسی و آثار علمی

دیزاین: صمد صبا و سید کاظم کاظمی

سال چاپ: ۱۳۹۹ هجری شمسی

تیراژ: ۱۰۰۰

چاپ: اول

ویب سایت: www.tveta.gov.af

ایمیل: info@tveta.gov.af

حق چاپ برای اداره تعلیمات تخصصی و مسلکی محفوظ است.



سرود ملی

دا عزت د هر افغان دی	دا وطن افغانستان دی
هر بچی یې قهرمان دی	کور د سولې کور د توري
د بلوڅو، د ازبکو	دا وطن د ټولوکور دی
د ترکمنو، د تاجکو	د پښتون او هزاره وو
پامیریان، نورستانیان	ورسره عرب، گوجردی
هم ايماق، هم پشهيان	براھوي دي، قزلباش دي
لكه لم پرشنه آسمان	دا هيوا د به تل ځليوي
لكه زړه وي جاويдан	په سينه کې د آسيا به
وايو الله اکبر وايو الله اکبر	نوم د حق مو دی رهبر



پیام اداره تعلیمات تخصصی و مسلکی

استادان نهایت گرامی و محصلان ارجمند!

تریتیت نیروی بشری ماهر، متخصص و کارآمد از عوامل کلیدی و انکارنایزدیر در توسعه اقتصادی و اجتماعی هر کشور محسوب می‌گردد و هر نوع سرمایه‌گذاری بزرگ در بخش‌های مختلف اقتصادی نیازمند به پلان‌گذاری و سرمایه‌گذاری در بخش نیروی بشری و توسعه منابع این نیرو می‌باشد. بر مبنای این اصل و بر اساس فرمان شماره ۱۱ مقام عالی ریاست جمهوری اسلامی افغانستان به تاریخ ۱۳۹۷/۲/۱ اداره تعلیمات تخصصی و مسلکی از بدنۀ وزارت معارف مجزا و فصل جدیدی در بخش عرضه خدمات آموزشی در کشور گشوده شد.

اداره تعلیمات تخصصی و مسلکی به عنوان مตولی و مجری آموزش‌های تخصصی و مسلکی در کشور محسوب می‌شود که در چارچوب استراتژی ۵ ساله خویش دارای چهار اولویت مهم که عبارت‌اند از افزایش دسترسی عادلانه و مساویانه فرآگیران آموزش‌های تخصصی و مسلکی در سطح کشور، بهبود کیفیت در ارائه خدمات آموزشی، یادگیری مادام‌العمر و پیوسته و ارائه آموزش نظری و عملی مهارت‌ها به‌طور شفاف، کم‌هزینه و مؤثر که بتواند نیاز بازار کار و محصلان را در سطح محلی، ملی و بین‌المللی برآورده کند، می‌باشد.

این اداره که فراگیرترین نظام تعلیمی کشور در بخش تعلیمات تخصصی و مسلکی است، تلاش می‌کند تا در حیطه وظایف و صلاحیت خود زمینه دستیابی به هدف‌های تعیین‌شده را ممکن سازد و جهت رفع نیاز بازار کار، فعالیت‌های خویش را توسعه دهد.

نظام اجتماعی و طرز زندگی در افغانستان مطابق به احکام دین مقدس اسلام و رعایت تمامی قوانین مشروع و معقول انسانی عیار است. اداره تعلیمات تخصصی و مسلکی جمهوری اسلامی افغانستان نیز با ایجاد زمینه‌های لازم برای تعلیم و تربیت جوانان و نوجوانان مستعد و علاقه‌مند به حرفه‌آموزی، ارتقای مهارت‌های شغلی در سطوح مختلف مهارتی، تربیت کادرهای مسلکی و حرفوی و ظرفیت‌سازی تخصصی از طریق انکشاف و ایجاد مکاتب و انسٹیتوت‌های تخصصی و مسلکی در سطح کشور با رویکرد ارزش‌های اسلامی و اخلاقی فعالیت می‌نماید.

فالهذا جهت نیل به اهداف عالی این اداره که همانا تربیت افراد ماهر و توسعه نیروی بشری در کشور می‌باشد؛ داشتن نصاب تعلیمی بر وفق نیاز بازار کار امر حتمی و ضروری بوده و کتاب درسی یکی از ارکان مهم فرایند آموزش‌های تخصصی و مسلکی محسوب می‌شود، پس باید همکام با تحولات و پیشرفت‌های علمی نوین و مطابق نیازمندی‌های جامعه و بازار کار تألیف و تدوین گردد و دارای چنان ظرافتی باشد که بتواند آموزه‌های دینی و اخلاقی را توازن با دست‌آوردهای علوم جدید با روش‌های نوین به محصلان انتقال دهد. کتابی را که اکنون در اختیاردارید، بر اساس همین ویژگی‌ها تهیه و تدوین گردیده است.

بدین‌وسیله، صمیمانه آرزومندیم که آموزگاران خوب، متعدد و دلسوز کشور با خلوص نیت، رسالت اسلامی و ملی خویش را ادا نموده و نوجوانان و جوانان کشور را به‌سوی قله‌های رفیع دانش و مهارت‌های مسلکی رهنمایی نمایند و از محصلان گرامی نیز می‌خواهیم که از این کتاب به درستی استفاده نموده، در حفظ و نگهداری آن سعی بليغ به خرج دهند. همچنان از مؤلفان، استادان، محصلان و اولیای محترم محصلان تقاضا می‌شود نظریات و پیشنهادات خود را در مورد این کتاب از نظر محتوا، ویرایش، چاپ، اشتباهات املایی، انشایی و تایپی عنوانی اداره تعلیمات تخصصی و مسلکی کتبآ ارسال نموده، امتنان بخشنند.

در پایان لازم می‌دانیم در جنب امتحان از مؤلفان، تدوین‌کنندگان، مترجمان، مصححان و تدقیق کنندگان نصاب تعلیمات تخصصی و مسلکی از تمامی نهادهای ملی و بین‌المللی که در تهیه، تدوین، طبع و توزیع کتب درسی زحمت‌کشیده و همکاری نموده‌اند، قدردانی و تشکر نمایم.

نديمه سحر

رئيس اداره تعلیمات تخصصی و مسلکی جمهوری اسلامی افغانستان

فهرست

عنوان	صفحه
..... مقدمه	ش
..... فصل اول: برنامه اوریکل و محیط توسعه دهنده سیکویل	۱
..... دیتابیس اوریکل (Oracle Database)	۱.۱
..... دانلود کردن اوریکل (Downloading Oracle)	۱.۲
..... نصب و عیارسازی دیتابیس اوریکل	۱.۲.۱
..... توسعه دهنده سیکویل (SQL Developer)	۱.۳
..... استفاده کردن توسعه دهنده سیکویل	۱.۳.۱
..... دیتابیس کاری اوریکل (Working Database)	۱.۴
..... وصل شدن به دیتابیس کاری اوریکل (Connection to Working Database)	۱.۴.۱
..... فصل دوم: قابلیت های PL/SQL (PL/SQL Capabilities)	۲۶
..... خصوصیات PL/SQL	۲.۱
..... موارد استفاده از PL/SQL	۲.۲
..... فواید PL/SQL	۲.۳
..... بلاک ها در PL/SQL Blocks	۲.۴
..... پیام های خروجی در PL/SQL Output Messages in PL/SQL	۲.۵
..... فصل سوم: متتحول های PL/SQL (PL/SQL Variables)	۴۹
..... شناسه های زبان PL/SQL (PL/SQL Identifiers)	۳.۱
..... شناسه های مجاز و غیر مجاز (Valid and Invalid Identifiers)	۳.۲
..... کلمه های ریزرف شده (Reserved Words) PL/SQL	۳.۲.۱
..... متتحول ها در زبان PL/SQL (PL/SQL Variables)	۳.۳
..... موارد و طرز استفاده از متتحول ها	۳.۴
..... انواع دیتای از قبل تعریف شده در PL/SQL (Predefined Datatypes)	۳.۵
..... انواع دیتای ترکیبی (Composite) و سکالری (Scalar)	۳.۶
..... استفاده از PL/SQL %TYPE Attribute	۳.۷
..... فواید استفاده از %Type Attribute	۳.۸
..... متتحول های به هم بسته (Bind Variables)	۳.۹
..... فصل چهارم: جملات قابل اجرا در PL/SQL (Executable Statements)	۷۵
..... اهداف و طرز استفاده از Lexical Units (PL/SQL) در یک بلاک	۴.۱
..... شناسه ها (Identifiers)	۴.1.1
..... کلمه های ریزرف شده (Reserved Words)	۴.1.2

۷۷	حائل‌ها (Delimiters)	۴.۱.۳
۷۸	لیترال‌ها (Literals)	۴.۱.۴
۸۲	نظریات در PL/SQL (PL/SQL Comments)	۴.۱.۵
۸۵	اهداف و طرز استفاده از توابع از قبل ساخته شده در PL/SQL	۴.۲
۸۷	تبدیل کردن نوع دیتا در PL/SQL (PL/SQL Datatype Conversion)	۴.۳
۸۷	تبدیل کردن آشکار (Explicit Conversion)	۴.۲.۱
۸۹	تبدیل کردن پوشیده یا ضمنی (Implicit Conversion)	۴.۲.۲
۹۰	موارد استفاده از تبدیل کردن آشکار و تبدیل کردن پوشیده در PL/SQL	۴.۲.۳
۹۱	اهداف استفاده از متتحول ها، بلاک‌های آشیانه‌بی (Nested Blocks) و لیبل‌ها	۴.۴
۹۱	هدف استفاده از متتحول ها	۴.۴.۱
۹۲	بلاک‌های آشیانه‌بی (Labels) (Nested Blocks) و لیبل‌ها	۴.۴.۲
۹۴	موارد استفاده از ترتیب‌ها (Sequences) با عبارات PL/SQL	۴.۵
۱۰۱	فصل پنجم: سرور دیتابیس اوریکل (Oracle Database Server)	
۱۰۲	دستورهای کار با دیتا (DML) در بلاک PL/SQL	۵.۱
۱۰۵	کنترول ترانزکشن (Transaction) در بلاک PL/SQL	۵.۲
۱۰۸	استفاده از عبارت INTO در بلاک PL/SQL	۵.۳
۱۰۹	کرسرهای در سیکویل و خصوصیات آنها	۵.۴
۱۱۰	کرسرهای ضمنی (Implicit Cursors)	۵.۴.۱
۱۱۱	کرسرهای آشکار (Explicit Cursors)	۵.۴.۲
۱۱۵	فصل ششم: ساختارهای کنترولی (Control Structures)	
۱۱۶	انواع و اهداف استفاده از ساختارهای کنترولی (Control Structures)	۶.۱
۱۱۸	ساختار و موارد استفاده از دستور IF	۶.۲
۱۲۵	ساختار و موارد استفاده از دستورهای CASE	۶.۳
۱۲۵	دستور CASE ساده	۶.۳.۱
۱۲۶	دستور CASE جستجو شده	۶.۳.۲
۱۲۸	ساختار و موارد استفاده از دستورهای حلقه (LOOP Statements)	۶.۴
۱۲۹	دستور حلقه FOR	۶.۴.۱
۱۳۱	دستور حلقه WHILE	۶.۴.۲
۱۳۳	ساختار و موارد استفاده از دستورهای CONTINUE و EXIT در حلقه‌ها	۶.۵
۱۳۳	دستورهای خارج شدن از حلقه	۶.۵.۱
۱۳۵	دستورهای ادامه‌دادن حلقه	۶.۵.۲
۱۴۰	فصل هفتم: انواع دیتای ترکیبی (Composite Data Types)	
۱۴۱	استفاده از ریکوردهای PL/SQL	۷.۱
۱۴۱	ایجاد ریکوردهای تعریف شده توسط استفاده کننده در PL/SQL	۷.۲
۱۴۴	ایجاد ریکوردها به اساس جدول‌ها با خصوصیات /ROWTYPE	۷.۳
۱۴۵	ایجاد ریکوردها به اساس کرسرهای INDEX BY table	۷.۴
۱۴۶	ایجاد کردن	۷.۵
۱۴۹	فصل هشتم: کرسرهای آشکار (Explicit Cursors)	

۱۵۰	کرسرهای آشکار (Cursors)	۸.۱
۱۵۰	فرق بین کرسرهای آشکار (Implicit Cursors) و کرسرهای پوشیده (Explicit Cursors)	۸.۲
۱۵۱	اهداف استفاده از کرسرهای آشکار	۸.۳
۱۵۲	روش ایجاد و کنترول کرسرهای آشکار	۸.۴
۱۵۳	ایجادکردن کرسرهای آشکار	۸.۴.۱
۱۵۴	استفاده از دستورهای حلقه (LOOP) با کرسرهای آشکار	۸.۴.۲
۱۵۷	ایجاد کرسرها با پارامترها	۸.۵
۱۵۸	استفاده از عبارت FOR UPDATE	۸.۶
۱۵۹	استفاده از عبارت WHERE CURRENT OF	۸.۷
۱۶۲	فصل نهم: دستورهای بررسی استثناهای (Exception Handling Commands)	
۱۶۳	استثناهای PL/SQL	۹.۱
۱۶۵	استثناهای بررسی نشده (Unhandled Exceptions)	۹.۲
۱۶۵	تطبیق FORALL به صورت فوری (Immediate)	۹.۲.۱
۱۶۷	تطبیق FORALL بعد از تکمیل دستورهای آن	۹.۲.۲
۱۶۹	انواع بررسی کننده‌های استثناهای (PL/SQL Exception Handlers)	۹.۳
۱۷۱	تکثیرشدن استثناء در بلاک‌های آشیانه‌بی (Exception Propagation)	۹.۴
۱۷۴	پیام‌های استثناهای PL/SQL	۹.۵
فصل دهم: پروسیجرها (Procedures), تابع‌ها (Functions), تریگرها (Triggers), بسته‌ها (Packages) و مجموعه‌ها (Collections)		
۱۷۸	۱۷۸ در زبان PL/SQL (Collections)	
۱۷۹	برنامه‌های فرعی PL/SQL در Subprograms	۱۰.۱
۱۸۱	خصوصیات پروگرام‌های فرعی	۱۰.۱.۱
۱۸۲	بخش‌های یک پروگرام فرعی	۱۰.۱.۲
۱۸۴	بلاک‌های ناشناس (Anonymous Blocks) PL/SQL در	۱۰.۲
۱۸۵	ایجاد یک پروسیجر ساده با بلاک‌های ناشناس	۱۰.۲.۱
۱۸۶	تابع ساده (Simple Function)	۱۰.۲.۲
۱۸۸	دستور RETURN در PL/SQL	۱۰.۲.۳
۱۸۸	استفاده از دستور RETURN در تابع	۱۰.۲.۴
۱۹۰	استفاده از دستور RETURN در پروسیجر	۱۰.۲.۵
۱۹۱	استفاده از دستور RETURN در بلاک ناشناس	۱۰.۲.۶
۱۹۴	فرق بین پروسیجرها و تابع‌ها	۱۰.۳
۱۹۶	تریگرها در PL/SQL	۱۰.۴
۱۹۷	بسته‌ها در PL/SQL	۱۰.۵
۲۰۰	مجموعه‌ها (Collections)	۱۰.۶
۲۰۵	۲۰۵ مأخذ (References)	

مقدمه

دیتابیس‌ها به خاطر تنظیم و استفاده معلومات، یکی از بخش‌های اساسی علوم کامپیوترساینس به شمار می‌روند. استفاده عملی از دیتابیس‌ها و ضرورت آن‌ها در کارهای مختلف از قبیل تحقیق، تجارت، انکشاف صنعت، نوآوری، اطلاعات جمعی، شبکه‌های اجتماعی و غیره موارد به خوبی هویدا است؛ به گونه مثال، با تنظیم و استفاده کردن معلومات در مورد یک موضوع تجاری، زمینه رشد فعالیت یک کمپنی تجاری مهیا می‌شود. تحلیل معلومات ذخیره شده در مورد مردم یک جامعه، کمک مستقیم به کنترول مسائل امنیتی و ایجاد پژوهش‌های رفاهی برای مردم از دیگر موارد استفاده از دیتابیس‌هاست. مثال‌های تنظیم و استفاده از دیتا و ارقام بی‌شمار است و به هر اندازه‌یی که در آن تعمق صورت گیرد، به همان پیمانه اهمیت دیتابیس‌ها بیش‌تر آشکار می‌گردد. دیتابیس توسط سیستم‌های مدیریت دیتابیس تنظیم و استفاده می‌شوند.

سیستم‌های مدیریت دیتابیس به اختصار به نام DBMS یا Database Management Systems یاد می‌شوند. توسط این سیستم‌ها دیتابیس‌ها ایجاد، استفاده و کنترول می‌گردند. سیستم‌های مدیریت دیتابیس به اندازه‌های مختلف موجود است. سیستم‌های کوچک، متوسط و بزرگ با قابلیت‌های گوناگون در مارکیت استفاده می‌شوند. سیستم‌های مدیریت دیتابیس به شکل تنظیم شده در یک کامپیوتر، قابل دسترس در یک شبکه محلی و یا هم قابل استفاده از طریق منابع آنلاین و اینترنت موجوداند.

مثال‌های معروف سیستم‌های مدیریت دیتابیس عبارت از اوریکل (Oracle) مربوط کمپنی اوریکل، سیکویل سرور (SQL Server) و اکسس (MS Access) مربوط کمپنی مایکروسافت، DB2 مربوط کمپنی IBM و غیره‌اند. بعضی از این سیستم‌ها تجاری است و لایسنس‌های استفاده از آن‌ها در مقابل پول خریداری می‌شوند و بعضی دیگر، هم تجاری بوده و هم به شکل منبع باز (Open Source) در محیط‌های تحقیقی و اکادمیک به صورت رایگان قابل استفاده می‌باشند. در این کتاب سیستم مدیریت اوریکل از دسته دوم در نظر گرفته شده و عنوانین کتاب بیشتر مربوط به آن می‌شود.

در فصل اول به توضیحات در مورد سیستم مدیریت دیتابیس اوریکل پرداخته شده است. به دست آوردن نرم‌افزار (دانلود کردن) از وب‌سایت کمپنی اوریکل به صورت رایگان و نصب آن روی کامپیوتر با توضیحات لازم و مثال‌ها ارائه شده است. توسعه‌دهنده سیکویل (SQL Developer) منحیث یک محیطی که در آن دیتابیس اوریکل استفاده می‌شود، به توضیح گرفته شده است. وصل شدن به دیتابیس کاری از طریق توسعه‌دهنده سیکویل و به راه‌انداختن دستورهای اجرایی اوریکل با مثال‌های ابتدایی در این فصل گنجانیده شده‌اند.

در فصل دوم کتاب معلوماتی در مورد قابلیت‌های زبان PL/SQL ارائه شده‌اند. زبان PL/SQL عبارت از زبان کیوری است که توسط دیتابیس اوریکل مورد استفاده قرار می‌گیرد. خصوصیات این زبان و موارد استفاده از آن با ذکر فواید زبان مذکور تحت عنوانی جداگانه توضیح شده‌اند. بلاک‌های کد زبان PL/SQL دارای یک ساختمان منظم بوده و قسمت‌های مختلف بلاک PL/SQL با ذکر مثال‌ها تشریح شده‌اند. آخرین موضوع مورد بحث را در فصل دوم، پیام‌های خروجی زبان PL/SQL تشکیل می‌دهد؛ مثال مشهور "سلام، جهان!" در این درس با استفاده از دو طریقه به صورت عملی نشان داده شده است.

در فصل سوم به موضوعات مربوط به متحولین زبان PL/SQL پرداخته شده است. نام‌گذاری متحولین در PL/SQL، کلمه‌های ریزرفشده و کلمه‌های کلیدی این زبان نشان داده شده‌اند. طریقه تعریف کردن متحولین و طریقه‌های دادن قیمت‌ها به متحولین در بلاک‌های کد زبان PL/SQL با نشان دادن مثال‌ها شرح شده‌اند. توضیح انواع دیتای قابل استفاده در زبان PL/SQL از بحث‌های دیگر در این فصل است. در قسمت انواع دیتا به دو دسته عمومی دیتای ترکیبی (Composite) و دیتای سکالری (Scalar) و موارد استفاده از آن‌ها اشاره شده است. استفاده از مشخصه‌یی به نام %Type Attribute و فواید استفاده از آن در PL/SQL توضیح داده شده و قسمت آخر فصل به توضیح و اهمیت استفاده از متحولین به هم‌بسته (Bind Variables) تخصیص یافته است.

فصل چهارم، معلوماتی در مورد دستورها و جملات قابل اجراء در PL/SQL را دارد. اهداف و طرز استفاده از بخش‌های نحوی به نام Lexical Units و نیز توضیحات لازم در مورد شناسه‌ها، کلمه‌های ریزرفشده، حائل‌ها، لیترال‌ها و تفسیرها در عنوانی مختلف توضیح داده شده‌اند. توابع از قبل ساخته‌شده در PL/SQL با لیست توابع و یک مثال اجرایی نشان داده شده است. تبدیل کردن نوع دیتا به شکل‌های آشکار و ضمنی (Explicit and Implicit Datatype Conversion) یکی دیگر از مسائل مورد بحث در این فصل است. اهداف استفاده از متحولین، بلاک‌های آشیانه‌یی (Nested Blocks) و استفاده از لیبل‌ها با نشان دادن مثال‌های عملی به توضیح گرفته شده‌اند. موارد استفاده از ترتیب‌ها (Sequences) با عبارات زبان PL/SQL تحت آخرين عنوان فصل تشریح و با مثال نشان داده شده‌اند.

فصل پنجم کتاب بیشتر روی موارد سرور دیتابیس اوریکل پیچیده است. در این فصل، به دستورهای کار با دیتا (DML) در بلاک PL/SQL اشاره و توضیحات لازم ارائه شده‌اند. کنترول ترانزکشن در بلاک زبان PL/SQL به خاطر مدیریت بهتر یوزردیتا به حیث یکی از اختیارهای لازم در زمان کار با دیتای استفاده کنندگان توضیح داده شده است. استفاده از عبارت SELECT INTO به خاطر دادن قیمت‌ها به متحولین با مثال واضح شده است. به کارگیری کرسرهای (Cursors) و توضیح انواع کرسرهای PL/SQL یکی دیگر از عنوانی مورد بحث در این فصل است. انواع کرسرهای PL/SQL مانند کرسرهای ضمنی (Implicit)

و کرسرهای آشکار (Explicit Cursors) به اختصار توضیح و با مثال‌های لازم از هم تفکیک شده‌اند.

در فصل ششم، ساختارهای کنترولی زبان‌های پروگرامنویسی پروسیجری در مجموع و به طور اخص در زبان PL/SQL تشریح شده‌اند. انواع و اهداف ساختارهای کنترولی زبان PL/SQL با ارائه مثال‌های مناسب نشان داده شده‌اند. در عناوین این فصل، موضوعات مربوط به دستورهای کنترولی IF و شاخه‌های آن شامل آذین‌بخش موضوعات دیگر در این فصل می‌باشند. استفاده از حلقه‌ها، منحیث ساختارهای معمول در زبان‌های پروگرامنویسی، به خاطر کنترول دستورهای اجرایی زبان PL/SQL بحث شده‌اند. حلقه‌های معروف WHILE و FOR با مثال‌های لازم تحت عناوین جداگانه توضیح شده‌اند. قسمت آخر فصل نیز به موضوعات مربوط به کنترول حلقه‌ها به خاطر توقف‌دادن عملیات و یا هم ادامه پروسه‌های شامل حلقه‌ها، تخصیص یافته و با مثال‌هایی توضیح داده شده است.

در فصل هفتم به موضوعات مرتبط به انواع دیتای ترکیبی در زبان PL/SQL پرداخته شده است. توضیح ریکوردهای PL/SQL و مقایسه قسمی آن‌ها با ریکوردهای جدول‌های دیتابیس‌ها صورت گرفته است. طریقه‌های ایجاد کردن ریکوردهای این زبان با تعریف کردن ریکوردها، دسترسی به فیلدهای ریکوردها و ریکوردها به حیث پارامترهای پروگرام فرعی (Subprogram) تشریح شده‌اند. ایجاد ریکوردها به اساس جدول‌ها و ایجاد ریکوردها به اساس کرسرهای مربوط به این فصل است. ایجاد کردن Index که یک نام معادل با Associative Array است، نیز توضیح داده شده است. مثال‌های عملی در هر قسمت فصل به خاطر وضاحت بیشتر، بخشی از موضوعات ارائه شده است.

فصل هشتم به موضوعات مربوط به کرسرهای آشکار در زبان PL/SQL ارتباط دارد؛ البته در فصل پنجم این کتاب نیز به اختصار به این موضوع پرداخته شده است. در این فصل با تفصیل بیش‌تر روی کرسرهای آشکار تمرکز صورت گرفته است. این فصل با توضیح عمومی در مورد کرسرهای آغاز شده که در آن در مورد دو نوع کرسرهای سیکویل به نام‌های کرسرهای آشکار (Explicit Cursors) و کرسرهای پوشیده یا ضمنی (Implicit Cursors) معلومات ارائه شده است. اهداف استفاده از کرسرهای آشکار، روش ایجاد کردن و کنترول این نوع کرسرهای PL/SQL با اضافه‌ساختن مثال‌ها، توضیح داده شده‌اند. استفاده کردن از حلقه‌های پروگرامنویسی با کرسرهای آشکار و ایجاد کرسرهای پارامترها عناوین دیگر بحث شده در این فصل است. در قسمت استفاده از اختیارهای کنترولی مؤثر در کرسرهای به صورت نمونه عبارت‌های FOR UPDATE و WHERE CURRENT OF نیز با ذکر مثال‌هایی تشریح شده‌اند.

در فصل نهم روی موضوعات مرتبط با غلطی‌های ارائه شده توسط زبان PL/SQL تمرکز شده است. دستورهای بررسی استثناهای، شامل توضیحاتی است در مورد استثناهای این زبان که به خاطر مدیریت کردن

غلطی‌ها استفاده می‌شوند. یک دسته از استثنایا به شکل بررسی نشده، به خاطر اهمیت موضوع در یک عنوان جداگانه به بحث گرفته شده‌اند. انواع بررسی کننده‌های استثنایا و نحوه استفاده از آن‌ها با مثال‌ها تشریح شده‌اند. تکثیرشدن استثنایا در حالات خاص بلاک‌های آشیانه‌یی PL/SQL از جمله موضوعات دیگر بحث شده در این فصل‌اند. پیام‌های استثنایا در زمان اجرای پروگرام‌های PL/SQL با به‌کارگیری تابع‌های مشخص، موضوع تکمیلی فصل نهم است.

فصل دهم منحیث فصل تکمیلی و آخر کتاب، موضوعات کلیدی مربوط به کار با مدیریت کدگذاری زبان PL/SQL را در بردارد. در این فصل به توضیحات در مورد پروسیجرها (Procedures)، تابع‌ها (Functions)، تریگرها (Triggers)، بسته‌ها (Packages) و مجموعه‌ها (Collections) در PL/SQL پرداخته شده است. فصل دهم به ترتیب با معلوماتی در مورد برنامه‌های فرعی در زبان PL/SQL، خصوصیات و بخش‌های پروگرام‌های فرعی آغاز شده است. بلاک‌های ناشناس (Anonymous Blocks) در این زبان و ایجاد پروسیجرها با بلاک‌های ناشناس تشریح شده‌اند. معلومات در مورد ساختن تابع‌های ساده با پارامترها و هم بدون پارامترها ارائه شده‌اند. فرق بین پروسیجرها و تابع‌ها و نیز اهداف استفاده از هر کدام‌شان در جریان فصل توضیح داده شده‌اند. مثال‌های لازم به خاطر وضاحت موضوعات در بخش‌های مختلف به صورت عملی نشان داده شده‌اند.



هدف کلی کتاب

آشنایی با قابلیت ها، متحول ها، کنترول ها، انواع دیتای ترکیبی، کرسر ها، استثنا ها و پروسیجر های PL/SQL

فصل اول

برنامه اوریکل و محیط توسعه دهنده سیکویل (Oracle and SQL Developer)



هدف کلی: شاگردان با نصب برنامه Oracle و محیط SQL Developer آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. خصوصیات Oracle Database 12c و نمونه‌های قبلی آن را توضیح دهند.
۲. دانلود، نصب و عیارسازی Oracle Database 11g را انجام دهند.
۳. محیط SQL Developer را با خصوصیات کلیدی آن شرح دهند.
۴. دیتابیسی را که با آن کار می‌کنند، شرح دهند.

سیستم‌های مدیریت دیتابیس به خاطر تحلیل، تنظیم و استفاده دیتا در کمپیوترها به کار گرفته می‌شوند. توسط این سیستم‌ها، دیتابیس‌ها دیزاین شده و مورد استفاده قرار می‌گیرند. سیستم‌های مدیریت دیتابیس ده‌ها نوع بوده و در مارکیت کار مورد استفاده می‌باشند. سیستم مدیریت دیتابیس اوریکل (Oracle DBMS) یکی از این نوع سیستم‌ها بوده و در ساحة کاری استفاده زیاد دارد؛ همچنان این سیستم به نام دیتابیس اوریکل (Oracle Database) یاد می‌شود. دیتابیس اوریکل یک سیستم منبع باز (Open Source) بوده و اساساً مربوط کمپنی اوریکل واقع در ایالت کلیفورنیای کشور امریکا است. این سیستم به صورت رایگان قابل دسترس عموم استفاده کنندگان است. نرمافزار دیتابیس‌های اوریکل از سایت رسمی کمپنی مربوطه قابل دریافت بوده و بعد از ایجاد کردن یک حساب استفاده کننده، هر شخصی می‌تواند واجد شرایط دانلود کردن برنامه‌های مربوطه شود.

در این فصل در مورد نمونه‌های مختلف دیتابیس اوریکل توضیحات لازم ارائه شده و طریقه‌های عملی استفاده از آن تحت عنوان‌های مختلف نشان داده شده‌اند. به خاطر وصل شدن به دیتابیس اوریکل، یک محیط استفاده کننده به نام توسعه‌دهنده سیکویل (SQL Developer) موجود بوده و از وبسایت رسمی اوریکل قابل دریافت است. نحوه استفاده از این وسیله نیز بخشی از عنوانین فصل اول کتاب به شمار می‌رود. دیتابیس اوریکل که با آن یک استفاده کننده کار می‌کند، از طریق وسیله توسعه‌دهنده سیکویل، قابل وصل شدن است. طریقه وصل شدن به دیتابیس منحیث آخرین موضوع فصل اول تشریح و با مثال نشان داده شده است.

۱.۱ دیتابیس اوریکل (Oracle Database)

اوریکل یکی از سیستم‌های مدیریت دیتابیس است و به نام دیتابیس اوریکل یاد می‌شود. این سیستم مدیریت دیتابیس دارای ساختمان‌های پیش‌رفته به خاطر استفاده کردن دیتابیس‌های رابطه‌بی (Relational DBs) و دیتابیس‌های شی‌ءگرا (Object Oriented DBs) است. به خاطر داشتن امکانات پیش‌رفته، اکثر استندردهای وضع شده جدید از طرف سازمان‌های جهانی^۱ ANSI و ISO^۲ برای زبان سیکویل با مراجعه به دیتابیس اوریکل تعریف می‌شوند؛ بنابراین، کارکردن با اوریکل، انکشاف دهنده‌گان دیتابیس را بیشتر به استندردهای جدید آشنا می‌سازد.

اوریکل در تمام محیط‌های کمپیوتر (سیستم‌های عامل) نصب و قابل استفاده است. این سیستم مدیریت دیتابیس می‌تواند بالای سرورهای بزرگ مدیریت دیتا با چندین پروسیسر نصب و استفاده شود و نیز می‌تواند در یک کمپیوتر شخصی (Personal Computer) به کار گرفته شود. اوریکل قابل توسعه بوده، با کلسترسازی

^۱ اختصار کلمات American National Standards Institute بوده و استندردها را در موارد مختلف تعریف و تأیید می‌کند.

^۲ اختصار کلمات International Organization for Standardization بوده و سازمان جهانی به خاطر تعریف و تأیید استندردها در موارد مختلف می‌باشد.

دیتا و با محاسبات دیتای دوبعدی، در هر نوع محیطی که نصب شده باشد به صورت درست و مطمئن کار می‌کند. یکی دیگر از جمله نکات مهم در استفاده از دیتابیس اوریکل این است که چون استندردها به اساس این DBMS تعریف شده‌اند، کسی که استفاده کردن از اوریکل را بلد باشد، در دیگر سیستم‌های مدیریت دیتابیس می‌تواند به خوبی کار کند.

طوری که گفته شد، سیستم مدیریت دیتابیس اوریکل در یک کمپیوتر شخصی هم قابل نصب (Install) و فعال شدن است. استفاده از اوریکل در کمپیوترهای شخصی بیشتر معمول بوده و کمپیوترهای شخصی را می‌توان یک محیط عام به خاطر استفاده کردن دیتابیس اوریکل نامید. نمونه‌های مختلف دیتابیس اوریکل برای استفاده در محیط‌های مختلف کمپیوترهای شخصی موجوداند؛ به گونه‌مثال، نمونه‌هایی از سیستم مدیریت دیتابیس اوریکل به خاطر نصب روی کمپیوترهای ویندوز (سیستم عامل Windows) موجود بوده و همچنان نمونه‌های این DBMS جهت نصب بالای سیستم‌های عامل لینوکس (Linux) نیز موجوداند. در نصب برنامه اوریکل سهولت‌های بیشتری وجود دارد و اوریکل با انعطاف‌پذیری قابل ملاحظه‌یی در یک سیستم عامل به خاطر استفاده از منابع سیستم، تنظیم (Configure) می‌شود.

سخت‌افزار ضروری به خاطر نصب شدن اوریکل

کمپیوتر جدید با امکانات مناسب سخت‌افزاری به خاطر نصب و استفاده از دیتابیس اوریکل ضرورت است. اگر نمونه‌های جدید اوریکل در کمپیوتر استفاده می‌شوند، سیستم عامل باید 64 بیتی باشد. بعضی سیستم‌های عامل 32 بیتی بوده که به خاطر استفاده از ورژن‌های جدید اوریکل مناسب نمی‌باشند. حافظه کمپیوتر (RAM¹) نیز به خاطر کار مؤثر و اجرای پروسه‌ها با سرعت بالا، مهم است. اندازه حافظه RAM در کمپیوتری که به خاطر این نوع سیستم مدیریت دیتابیس استفاده می‌شود، باید حداقل 2 گیگا بایت (2GB) باشد. کمپیوترهای جدید در بازار معمولاً با اندازه‌های حافظه بالاتر از 2GB به سهولت پیدا می‌شوند.

نکته قابل اهمیت دیگر در نصب و استفاده از دیتابیس اوریکل عبارت از مقدار ساحة خالی روی دیسک است. حداقل ساحة خالی در درایو (Drive) که برنامه نصب می‌شود، باید 10 گیگا بایت (10GB) باشد. این هم معصل کلانی نیست؛ کمپیوترهای عادی در بازار، هارد دیسک‌های حداقل 500GB دارند. اهمیت مسأله در درست تنظیم کردن پارتیشن‌های هارد دیسک کمپیوتر است تا درایوهایی که در آن‌ها برنامه‌هایی مانند اوریکل نصب می‌شوند، سایز مناسب برای شان تعیین شود.

به خاطر چک کردن این که سیستم عامل کمپیوتر به خاطر استفاده از نمونه‌های جدید اوریکل، مناسب است و یا خیر، طریقه‌های ذیل در سیستم عامل ویندوز به کار گرفته می‌شوند:

¹ اختصار کلمات RAM بوده و یکی از قسمت‌های اصلی بروزه جات کمپیوتر است.

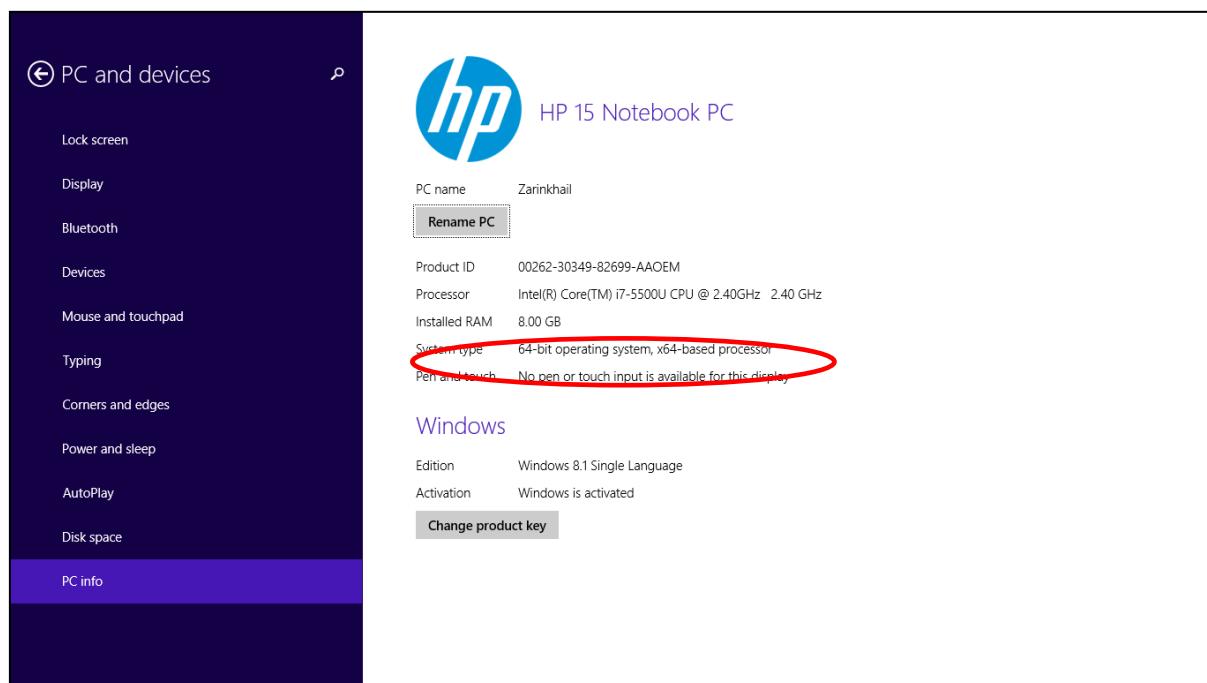
Click on Start >> Accessories >> System Tools >> System Information

Or

ویا

Type "System Information" in Search Box

توسط انجام این دستورها، صفحه‌یی مطابق شکل پایین باز می‌شود. صفحه‌یی مذکور معلوماتی در مورد سیستم کمپیوتر نشان می‌دهد. اگر در لیست دیده شود، در مقابل کلمه System Type نوع سیستم عامل کمپیوتر مذکور نشان داده شده است؛ در این مثال، سیستم عامل کمپیوتر 64 بیتی بوده و برای نصب کردن و به کارگیری دیتابیس اوریکل مناسب است.



شکل ۱-۱

نمونه‌های اوریکل

سیستم مدیریت دیتابیس اوریکل دارای نمونه‌های زیادی است. موجودیت نمونه‌های زیاد نمایان گر انکشاف اوریکل است؛ به این معنا که دیتابیس اوریکل در سال‌های آخر با تغییرات (انکشاف) در ورژن‌های مختلف به مارکیت آمده است. آخرین نمونه اوریکل عبارت از 12c است. نمونه‌های قبلی به ترتیب عبارت از اوریکل 11g، اوریکل 10g و غیره‌اند. به هر اندازه‌یی که ورژن یک برنامه بالا باشد، به همان اندازه کار آن بهتر و مشکلات نمونه‌های قبلی در آن حل شده‌تر است. آخرین نمونه این برنامه، یعنی اوریکل 12c به یکی از چهار شکل زیر نصب می‌شود:

1. Enterprise
2. Standard Edition
3. Standard Edition One
4. Express Edition

هر کدام از چهار شکل نصب شدن اوریکل با سیستم مدیریت دیتابیس رابطه‌یی و حمایت کامل سیکویل روی کمپیوتر نصب می‌شود. تفاوت بین شکل‌های نصب اوریکل مربوط به قابلیت استفاده از ساختمان‌های پیش‌رفته در هر کدام‌شان است. ساختمان‌های پیش‌رفته‌یی که در شکل‌های مختلف، تعدادی از آن‌ها قابل استفاده می‌باشند، عبارت‌اند از:

۱. دسترسی بالا (High Availability)
۲. قابلیت توسعه (Scalability)
۳. قابلیت اجراء (Performance)
۴. قابلیت اداره (Manageability)
۵. انبارکردن دیتا (Data Warehousing)
۶. (BI) Business Intelligence

در اکثر موارد، در کمپیوترهای شخصی، نصب Standard Edition One توصیه می‌شود؛ البته شکل‌های دیگری که در بالا لیست شده‌اند، هر کدام کارآیی خود را داشته و نظر به ضرورت استفاده‌کنندگان، شکل‌های نصب برنامه اوریکل ۱۲c انتخاب می‌شوند.

آمادگی به خاطر نصب کردن اوریکل

سیستم مدیریت دیتابیس اوریکل یک سیستم منبع باز (Open Source) بوده و به دست آوردن آن نیز به شکل قانونی آزاد است. این مسئله یکی از نکات قوت و صفات اوریکل است که اشخاص مسلکی در سراسر دنیا می‌توانند اوریکل را استفاده کرده و در انکشاف آن رول داشته باشند. نرم‌افزار اوریکل از وب‌سایت مربوطه آن بدون مصارف و پرداخت پول قابل دریافت است، البته دانلود کردن آن به اینترنت با سرعت بالا ضرورت دارد؛ چون فایل‌های برنامه به شکل فولدرهای زیپ شده (معمولًا در دو فولدر) دانلود می‌شود که سایز هر کدام‌شان به چند گیگا بایت می‌رسد؛ یعنی به خاطر به دست آوردن سافت‌ویر اوریکل، فولدرهای حاوی فایل‌های نصب شدن، هر کدام باید به یک بار دانلود شود.

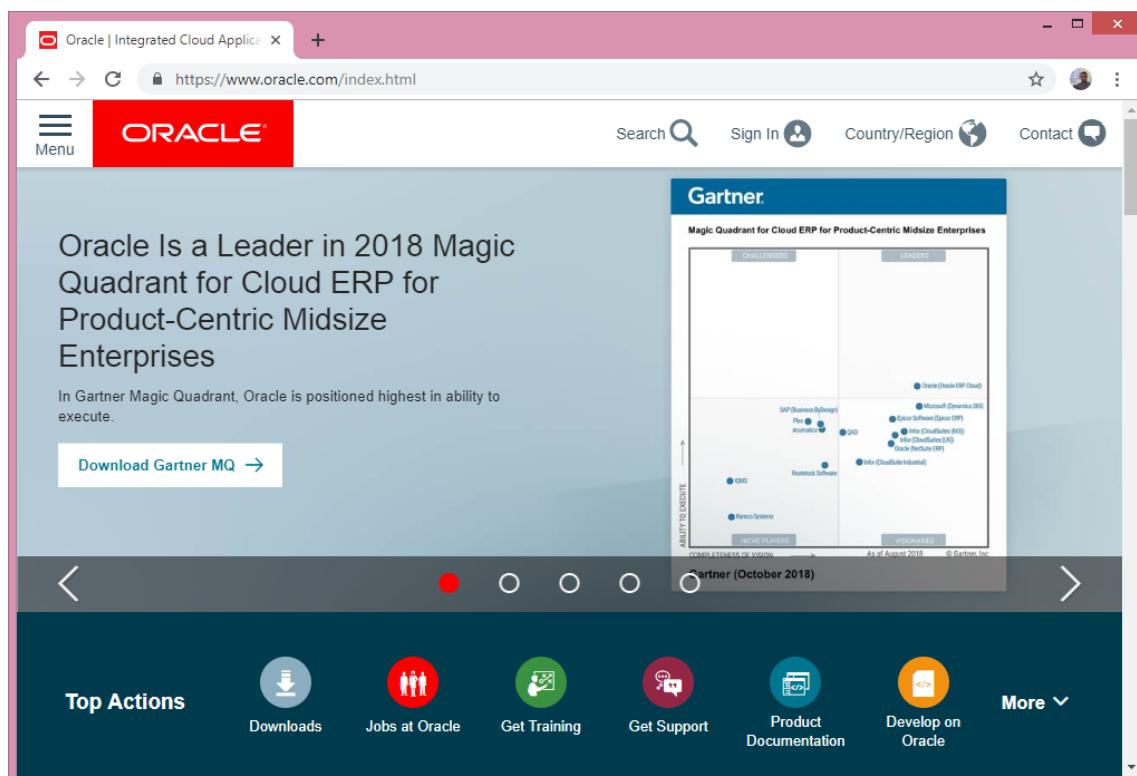
نصب کردن اوریکل در کمپیوتر با نصب کردن اپلیکیشن‌های عادی تفاوت دارد. اکثر اپلیکیشن‌های عادی به صورت عادی در سیستم عامل ویندوز با استفاده از ویزارد های ساده، نصب می‌شوند. نصب اوریکل و تعداد دیگری از DBMS‌ها با در نظر گرفتن اختیارهای اضافی به یک اندازه مهارت و رهنماهی ها نیاز دارد. این نوع سیستم‌ها با بخش‌های سیستم‌های عامل کمپیوتر بیشتر ترکیب می‌شوند. اوریکل بخش‌ها و اختیارهای

بیشتری را نظر به اپلیکیشن‌های معمولی، در کمپیوتر نصب می‌کند. در جریان نصب برنامه، اکثر قسمت‌های سیستم عامل چک می‌شوند تا در آن‌ها کدام مشکلی به خاطر نصب بخش‌های اجرایی اوریکل وارد نشده باشد.

۱.۲ دانلود کردن اوریکل (Downloading Oracle)

طوری که قبلاً یادآوری شد، نرم‌افزار اوریکل از اینترنت بدون پرداخت پول قابل دریافت است. در این بخش راهنمایی و قدم‌های عملی به خاطر دانلود کردن برنامه اوریکل از وبسایت مربوطه آن نشان داده شده است. در قدم نخست، با ایجاد کردن یک حساب استفاده کننده در شبکه تکنالوژی اوریکل (Oracle Technology Network) که به اختصار آن را OTN می‌گویند، پروسه آغاز می‌شود. ایجاد کردن این حساب و داخل شدن به وبسایت مربوطه از شرایط حتمی به خاطر دانلود کردن اوریکل و برنامه‌های حمایوی آن است.

به خاطر ایجاد کردن حساب کاربری در وبسایت اوریکل، از طریق آدرس <http://www.oracle.com> وارد شوید. تصویر پایین، صفحه اوریکل را نشان می‌دهد. شکل صفحه اوریکل اکثراً تغییر می‌کند؛ ولی بخش‌های آن پابرجا بوده و قابل استفاده است.



شکل ۱-۱

بعد از بازشدن صفحه بالا از طریق آدرس <http://www.oracle.com>، بالای صفحه فرعی تحت عنوان که همان OTN است، کلیک شود؛ هم‌چنان می‌شود به طور مستقیم با

استفاده از آدرس <http://www.oracle.com/technology> به صفحه مورد نظر جهت ایجاد کردن حساب رفت. شکل پایین، صفحه OTN را نشان می‌دهد:

The screenshot shows the Oracle Technology Network homepage. At the top, there's a navigation bar with links for Sign In/Register for Account, Help, Select Country/Region, Communities, I am a..., I want to..., and a search bar. Below the navigation is a horizontal menu with links for PRODUCTS AND SERVICES, SOLUTIONS, DOWNLOADS, STORE, SUPPORT, TRAINING, PARTNERS, and ABOUT. A banner for 'Oracle Technology Network' features a photo of a developer at a computer. Below the banner, there's a section for 'Join now; It's free and easy!' followed by a 'What's New' section with links for Java Developers, Database Admins and Developers, System Admins and Developers, and Solution Architects. On the left, there's a sidebar titled 'Essential Links' with links to Software Downloads, Documentation & APIs, Discussion Forums, Critical Patch Updates, Hot Topics, Technical Articles, Hands-on Workshops, Newsletters, and Blogs. To the right of the sidebar is a 'Recently Tagged' section with links to a podcast and two tech articles. Further right is a social media feed for '#oracletechnet' with tweets from Oracle Technet. The bottom of the page includes a footer with links for Oracle COM, TECHNOLOGY NETWORK, PARTNERS, STORE, and SUPPORT.

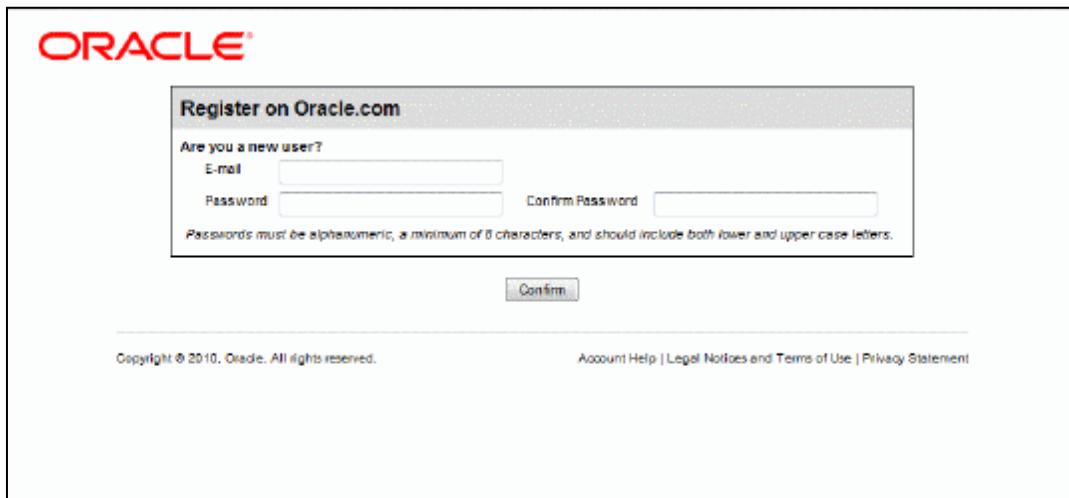
شکل ۳-۱

در این صفحه به خاطر ایجاد کردن حساب استفاده کننده، بالای لینک Sign In/Register در قسمت بالایی صفحه کلیک شود؛ در نتیجه شکل پایین نمایان خواهد شد.

The screenshot shows the Oracle sign-in page. It has a 'Sign In' form where users can enter their Single Sign-On user name and password. There are fields for 'Username' and 'Password' with a 'Forgot your password?' link. To the right of the sign-in form is a 'Why Sign In?' section explaining the benefits of signing in, such as managing subscriptions, accessing downloads, and using applications anywhere. At the bottom of the page, there are copyright notices, links to About Oracle, Contact Us, RSS feeds, Site Maps, Legal Notices and Terms for Use, Privacy Statement, and a note about being powered by Oracle Application Server Portal.

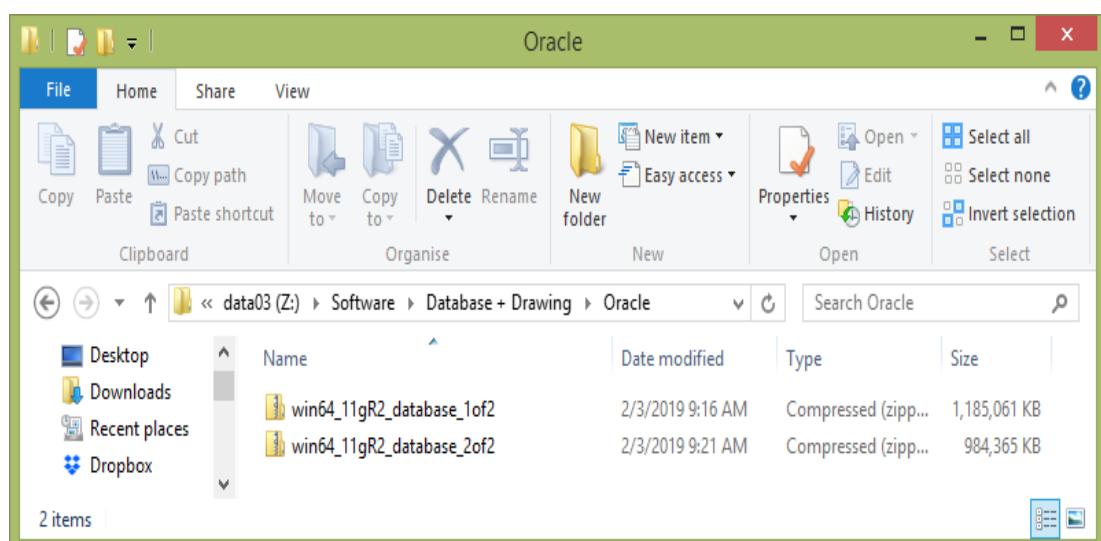
شکل ۴-۱

در صورت داشتن حساب استفاده کننده قبلی، می‌توانید با وارد کردن نام و رمز عبور از این صفحه به وب سایت داخل شوید. در صورت موجود نبودن حساب قبلی و یا هر دلیل دیگری که ضرورت به ایجاد کردن حساب جدید باشد، یک حساب با کلیک کردن بالای دستور Create your Oracle account now در قسمت پایین سمت راست قابل ایجاد است. با انجام دادن این کار، صفحه بعدی به شکل زیر باز خواهد شد:



شکل ۵-۱

با وارد کردن معلومات مورد ضرورت به صفحه بالا و تأیید از طریق ایمیل، حساب جدید OTN ایجاد می‌شود. با استفاده از حساب جدید، یک استفاده کننده می‌تواند به وب سایت اوریکل داخل شود. بعد از دخال شدن به وب سایت، نرم افزار مورد ضرورت به سادگی دانلود می‌شود. زمانی که دانلود فایل‌ها تکمیل شد، سیستم باید به خاطر نصب دیتابیس اوریکل آماده سازی شود. در این مبحث، سیستم مدیریت دیتابیس اوریکل ورژن 11g دانلود شده و در ذیل نشان داده می‌شود:



شکل ۶-۱



طوری که در شکل دیده می‌شود، دو فولدر زیپ شده با سایزهای متفاوت در موقعیتی از کمپیوتر نشان داده می‌شوند. قدم‌های بعدی استفاده از این فایل‌ها به خاطر نصب کردن دیتابیس اوریکل نمونه 11g در کمپیوتر را نشان می‌دهد که در عنوان بعدی به آن پرداخته شده است.

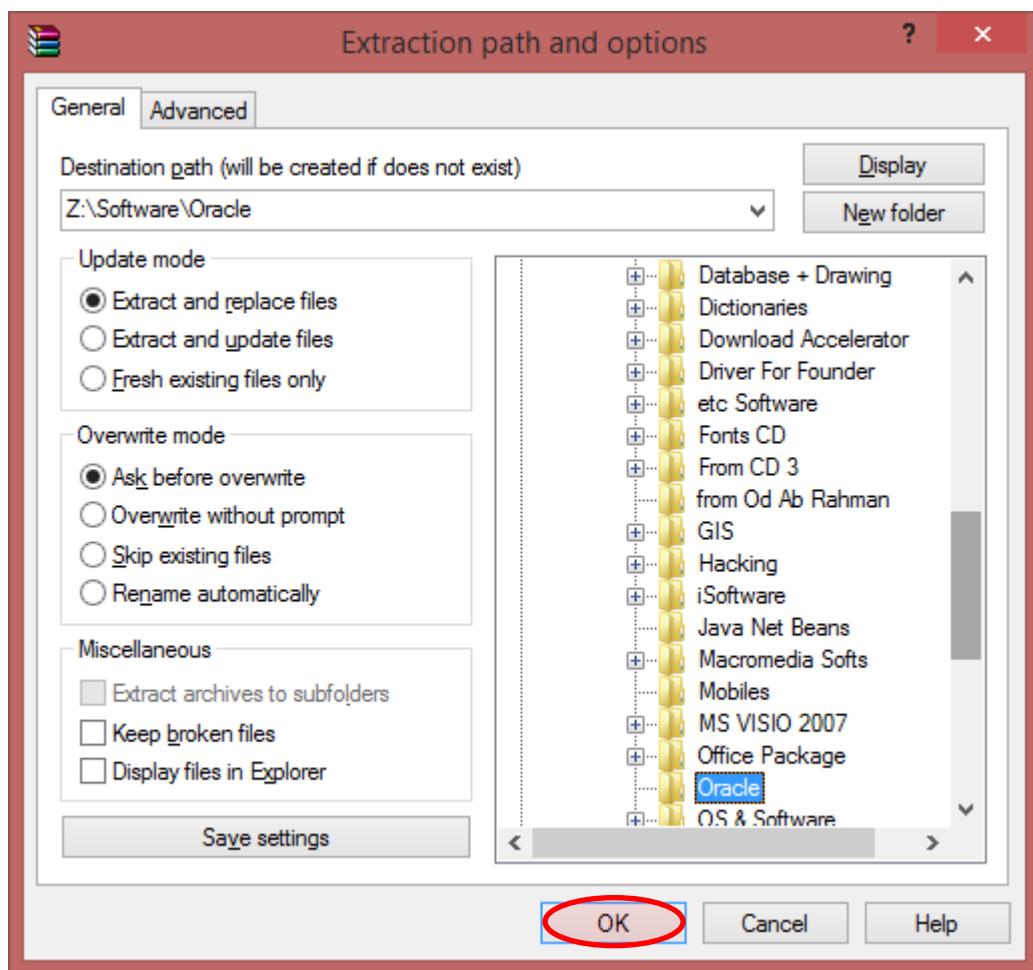
۱۰.۲.۱ نصب و عیارسازی دیتابیس اوریکل

طوری که در درس قبلی دیده شد، نرم‌افزار اوریکل از وب‌سایت رسمی آن دانلود و در کمپیوتر ذخیره شد. ورژن گرفته شده اوریکل از وب‌سایت آن عبارت از اوریکل 11g سلسله 2 بوده که به اختصار آن را به نام Oracle11g Release2 یا (11gR2) یاد می‌کنند. بلندترین ورژن اوریکل تا این زمان (زمستان ۹۷) عبارت از اوریکل 12c است. اوریکل 12c تنها در فارمت 64 بیت بوده و در سیستم‌های عامل 32 بیتی قابل نصب نیست. نمونه‌های قبلي اوریکل در هردو فارمت 64 بیت و 32 بیت به صورت نرم‌افزارهای مستقل، قابل استفاده‌اند. در قسمت عملی استفاده از اوریکل در این صنف، همان نمونه 11g سلسله 2 از نوع 64 بیت در نظر گرفته شده است.

بعد از دانلود کردن اوریکل، فولدرهای مربوطه آن به شکل فشرده شده (Zipped) می‌باشند. فولدرهای نرم‌افزار اوریکل دانلود شده از وب‌سایت آن عبارت‌اند از:

1. win64_11gR2_database_1of2.zip
2. win64_11gR2_database_2of2.zip

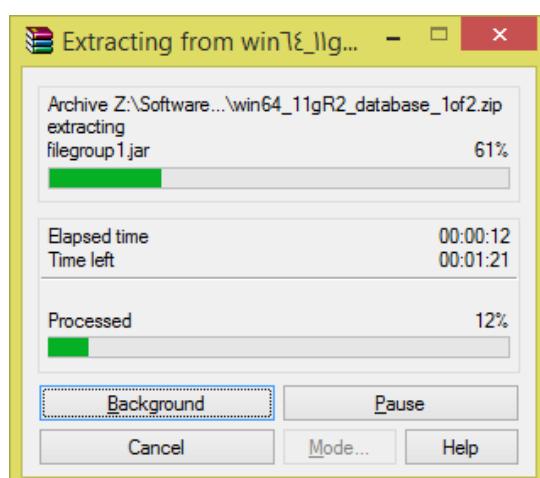
به خاطر استفاده از فولدرهای زیپ شده جهت نصب اوریکل روی سیستم‌عامل کمپیوتر، در ابتدا فولدرها باید در یک دایرکتوری Unzip شوند. قراردادن فایل‌های اوریکل جهت نصب و عیارسازی در یک فولدر حتمی است. تقسیم فایل‌های نرم‌افزار اوریکل در دو فولدر به خاطر سهولت دانلود کردن آن‌ها از منابع آنلاین به کار گرفته شده است. با استفاده از وسیله‌یی (مثلاً نرم‌افزار WinRaR) فایل‌های زیپ شده در یک دایرکتوری به نام Oracle در کمپیوتر مطابق شکل پایین Unzip می‌شود:



شکل ۷-۱

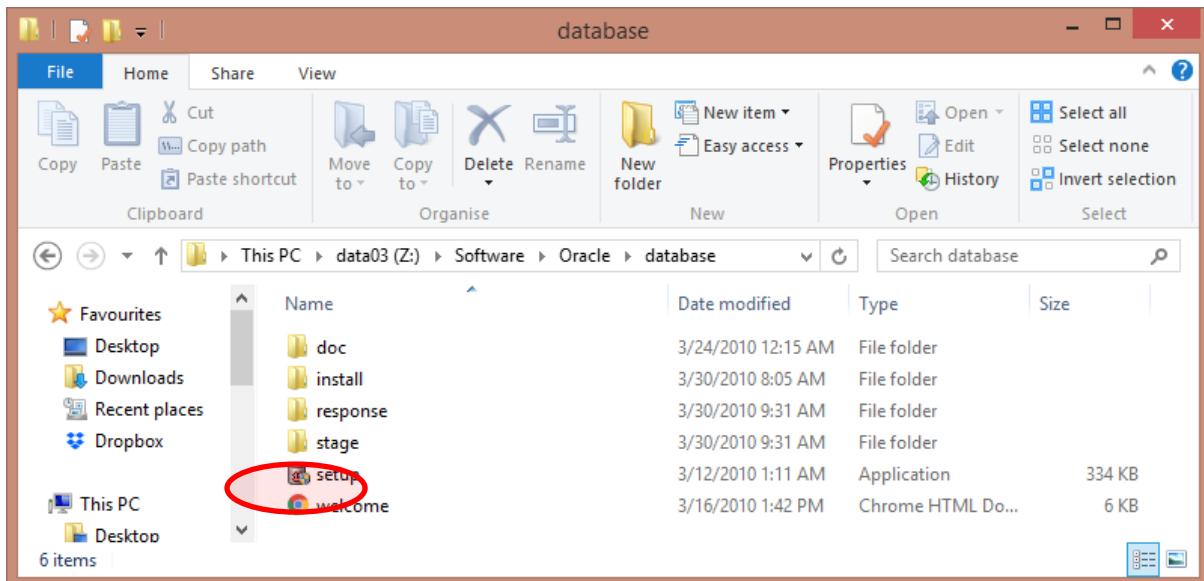
بعد از انتخاب یک موقعیت معین (یک دایرکتوری؛ طور مثال دایرکتوری Oracle در شکل بالا) فایل‌های مربوطه در موقعیت جدید در یک فolder Unzip می‌شوند.

شکل پایین، پروسهٔ Unzip شدن فایل‌های اوریکل را با استفاده از نرم‌افزار مربوطه نشان می‌دهد:



شکل ۸-۱

بعد از انجام این عملیه، در دایرکتوری دیتابیس، دایرکتوری‌های فرعی ذیل به حیث محتوای اصلی نشان داده می‌شوند.



شکل ۹-۱

مرحله بعدی عبارت از نصب (انستالیشن) برنامه اوریکل است. به خاطر تکمیل شدن موفقانه این پروسه، فایل‌های اوریکل همه باید به یک فolder به نام دیتابیس که در شکل بالا دیده می‌شود، انتقال داده شوند. در صورتی که محتوای فولدرهای زیپ شده به دو فolder جداگانه Unzip شده باشد، قبل از اجرای نصب اوریکل، باید تمام محتویات با در نظر گرفتن ساختمان و نام دایرکتوری‌های فرعی داخل یک دایرکتوری به نام database انتقال یافته و تنظیم شوند.

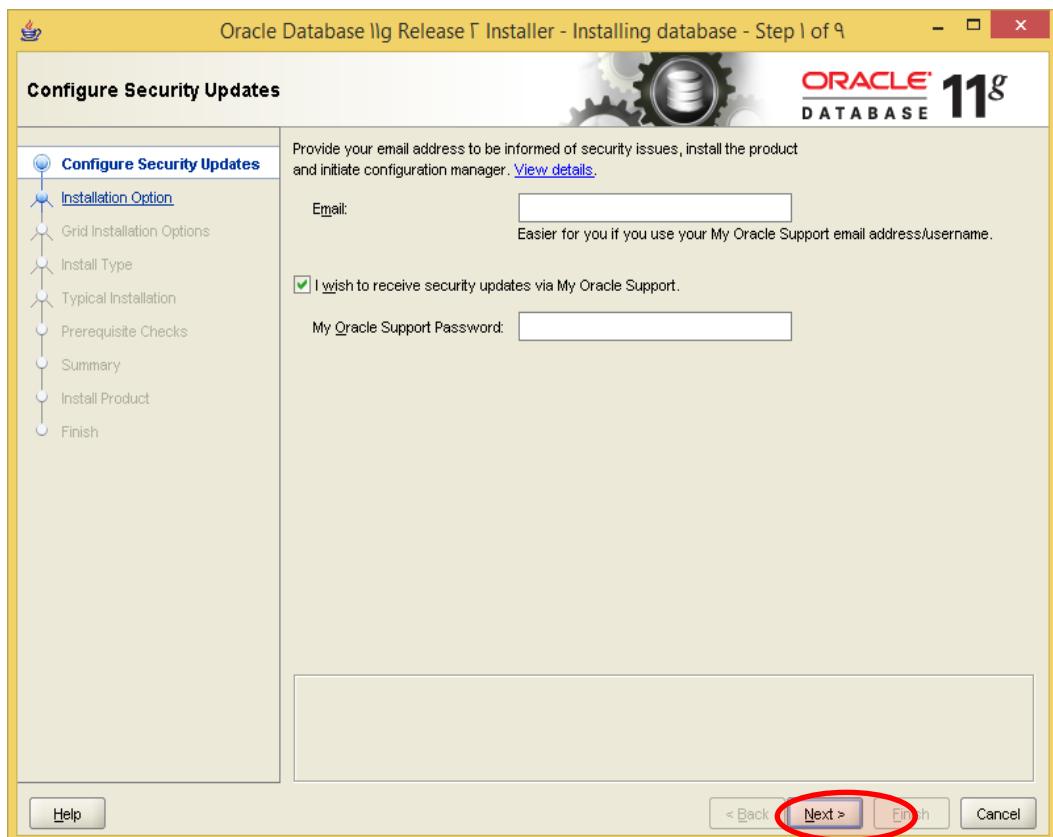
مراحل نصب دیتابیس اوریکل

جهت نصب سیستم مدیریت دیتابیس اوریکل در کمپیوتر، مراحلی در نظر گرفته شده که هر کدام آن‌ها تشریح و به صورت عملی نشان داده می‌شوند.

مرحله ۱ نصب دیتابیس اوریکل

با استفاده از فایل اجرایی setup که در دایرکتوری database قرار دارد، پروسه نصب کردن اوریکل آغاز می‌شود؛ البته در کمپیوتری که نصب اوریکل صورت می‌گیرد، استفاده کننده آن باید اجازه مدیریت سیستم را داشته باشد. با به راه انداختن دستور اجرایی setup سیستم عامل با یک پیام تأییدی از استفاده کننده می‌خواهد تا اجازه نصب برنامه جدید را بدهد. بعد از موافقت استفاده کننده با صلاحیت، پروسه نصب آغاز می‌شود. اولین صفحه این پروسه در شکل پایین نشان داده شده است. در این صفحه داخل کردن معلومات

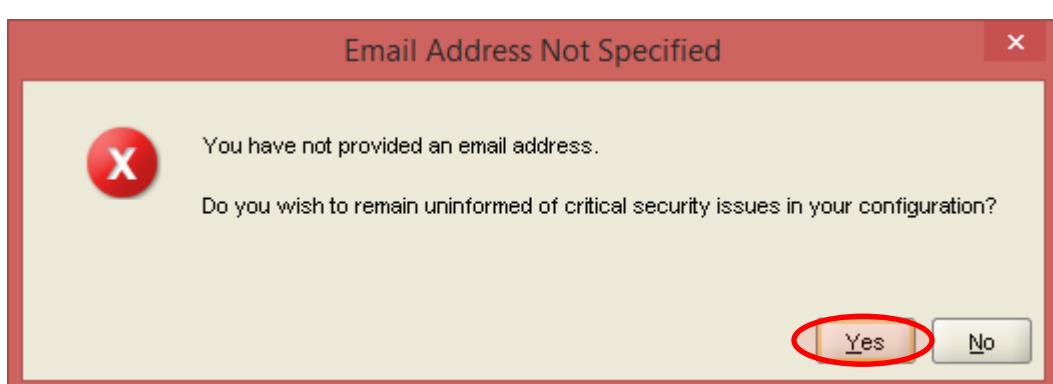
از قبیل ایمیل و غیره حتمی نبوده و استفاده کننده می‌تواند به خاطر پیش‌رفتن پروسه نصب اوریکل فقط دکمه Next را فشار دهد.



شکل ۱۰-۱

مرحله ۲ نصب دیتابیس اوریکل

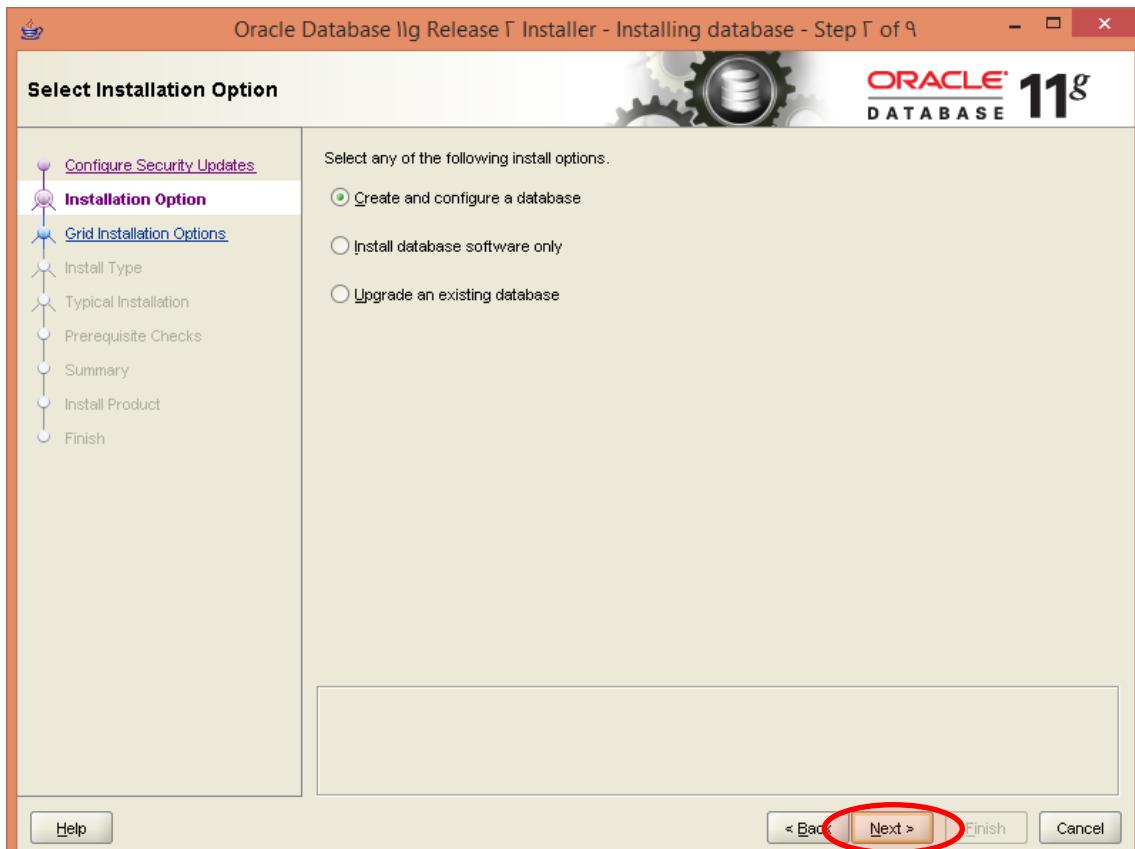
در این مرحله، یک پیام در قسمت اضافه کردن ایمیل آدرس نشان داده می‌شود. با انتخاب کردن دکمه تأیید (Yes) بدون این که ایمیل وارد شود، به پروسه ادامه داده می‌شود.



شکل ۱۱-۱

مرحله ۳ نصب دیتابیس اوریکل

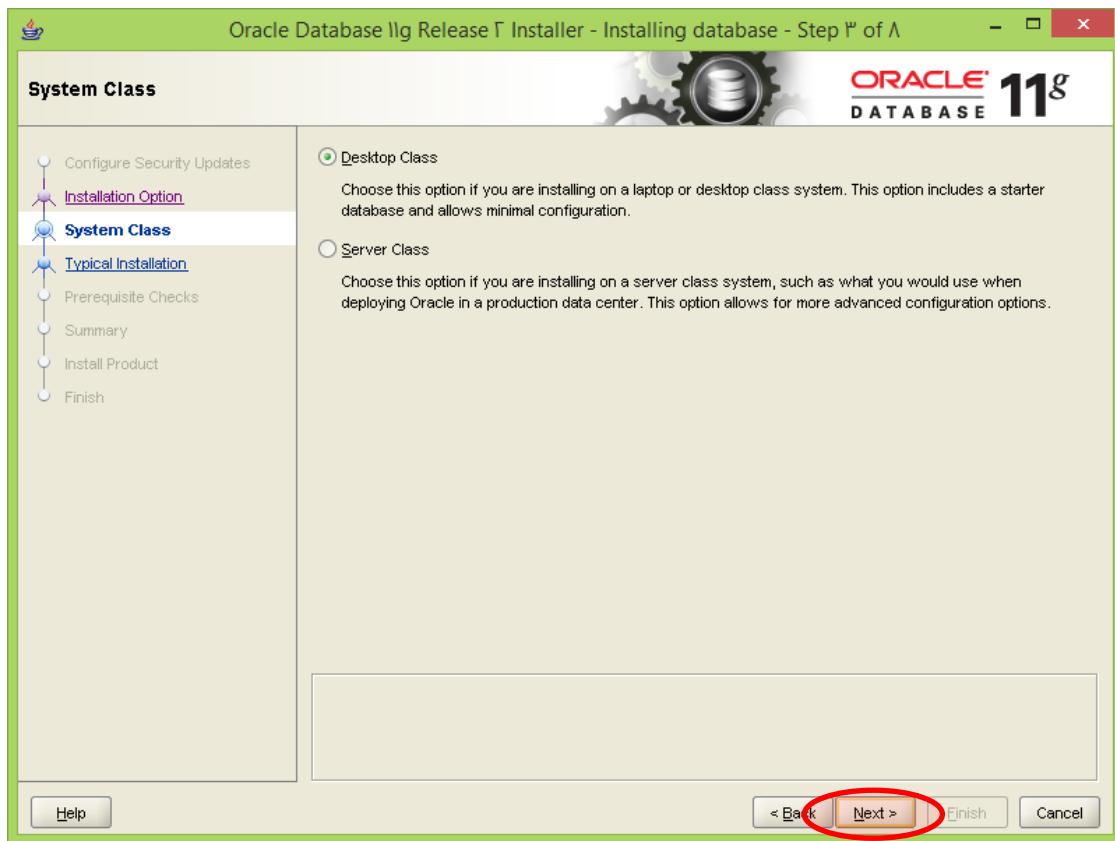
در این مرحله، صفحه پایین بعد از موافقت با مرحله ۲ نشان داده می‌شود. با موافقت با اختیار اول، کلید تأیید کلیک شده و پروسه نصب برنامه به صفحه بعدی ویزارد، پیش می‌رود.



شکل ۱۲-۱

مرحله ۴ نصب دیتابیس اوریکل

با درنظرداشت این که اوریکل به هدف مسائل ابتدایی و آموزش در کمپیوتر دسکتاپ و یا لپتاپ نصب می‌شود، مطابق شکل، اختیار اول آن (Desktop Class) انتخاب شده و پروسه با کلیک کردن روی دکمه Next پیش برد شود.

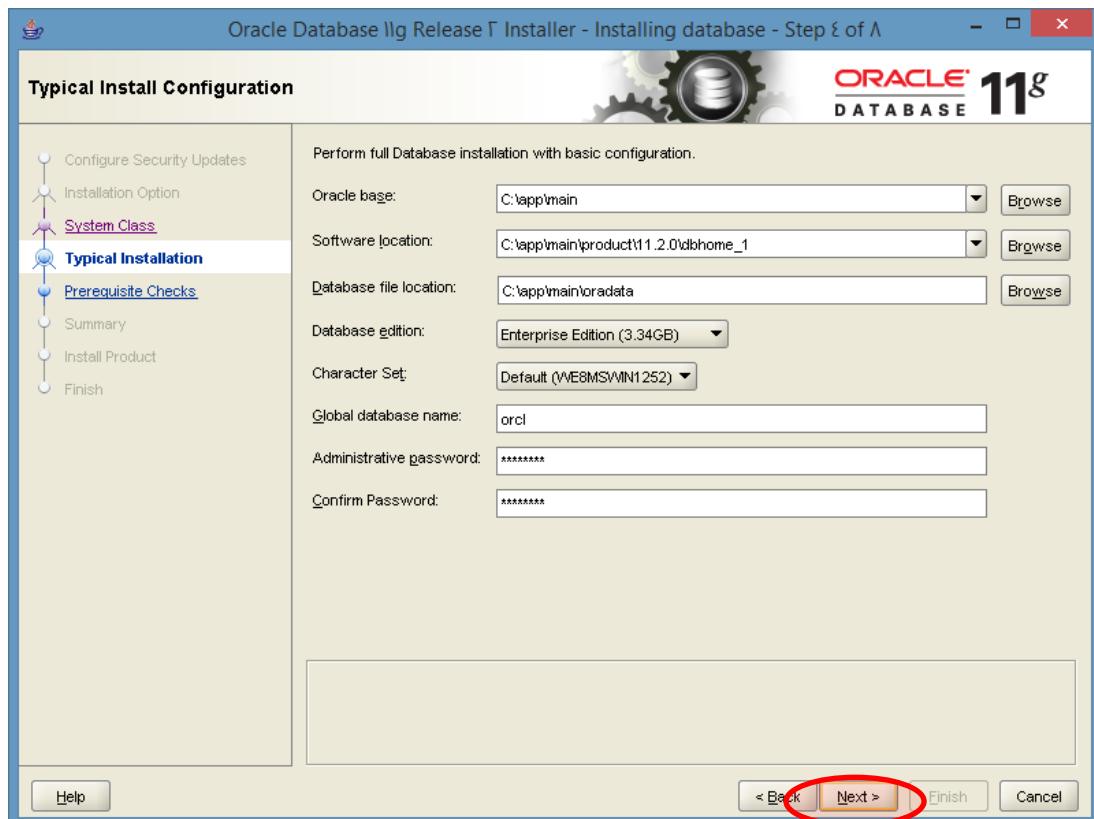


شکل ۱۳-۱

مرحله ۵ نصب دیتابیس اوریکل

در این مرحله، یک موقعیت به خاطر نصب کردن سرور اوریکل در کمپیوเตٽ تعیین می‌شود. در شکل پایین، قسمت اول صفحهٔ ویزارد به نام Oracle base مهم بوده و تنظیم کردن آن به اختیار استفاده کننده است. در این مثال، این آدرس C:\app\main به حیث موقعیت تعیین شده است. زمانی که این قسمت ایدیت شود، اختیارهای پایین در این صفحه به شکل خودکار تنظیم می‌شوند. معلومات دیگری که در این مرحله مهم بوده و باید به سیستم اضافه شود، عبارت از تنظیم کردن رمز عبور (Password) است.

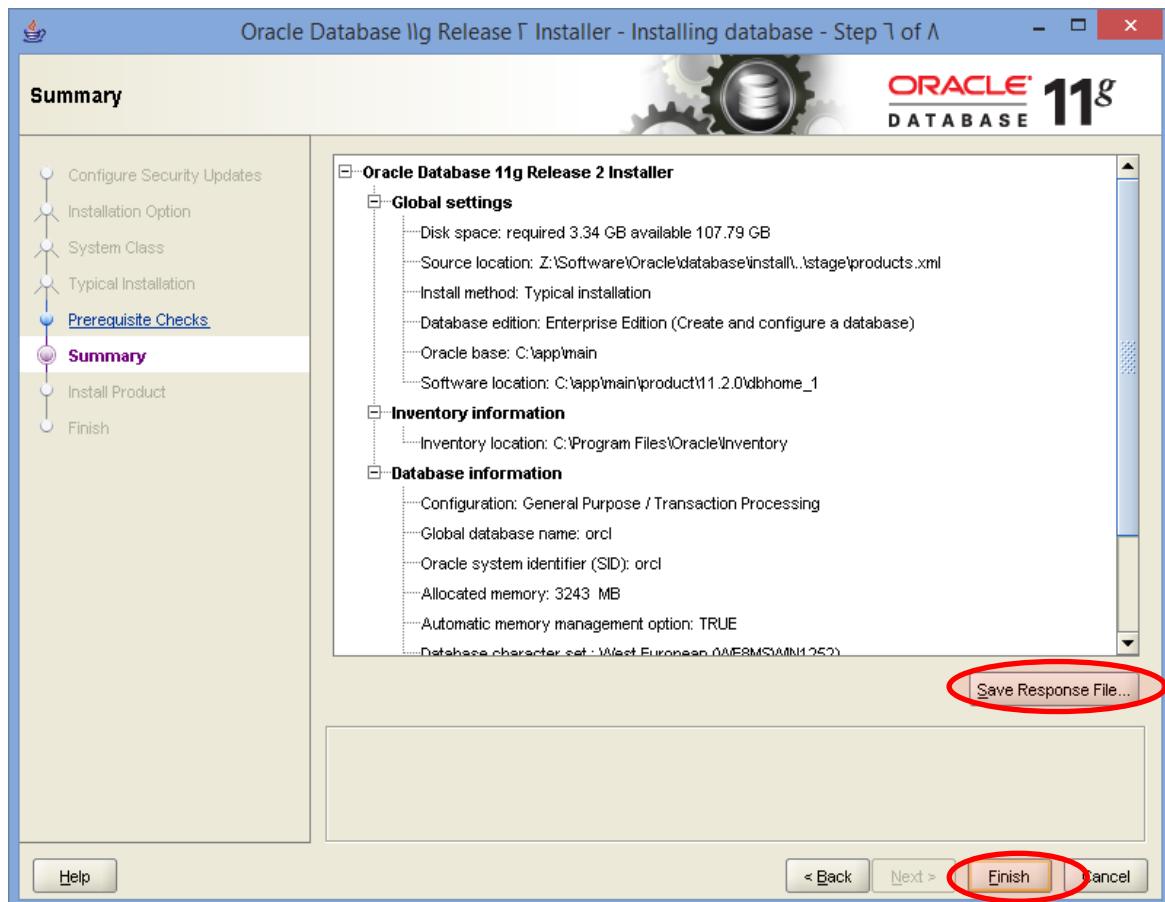
پاسوردی که در این مرحله تنظیم می‌شود، به خاطر تنظیم کارهای مدیریت دیتابیس Database Administration در زمان استفاده از اوریکل به کار گرفته می‌شود. پاسورد بخش مهم سیستم اوریکل بوده و باید حداقل 8 کرکتر باشد. کرکترهای مجاز با تنوع مانند حروف کوچک، حروف بزرگ، نمبرها و بعضی سимвول‌ها در رمز عبور به کار بردگ می‌شوند. بعد از علاوه کردن معلومات ضروری، صفحهٔ ویزارد با کلیک کردن دکمهٔ Next به پیش برده می‌شود.



شکل ۱۴-۱

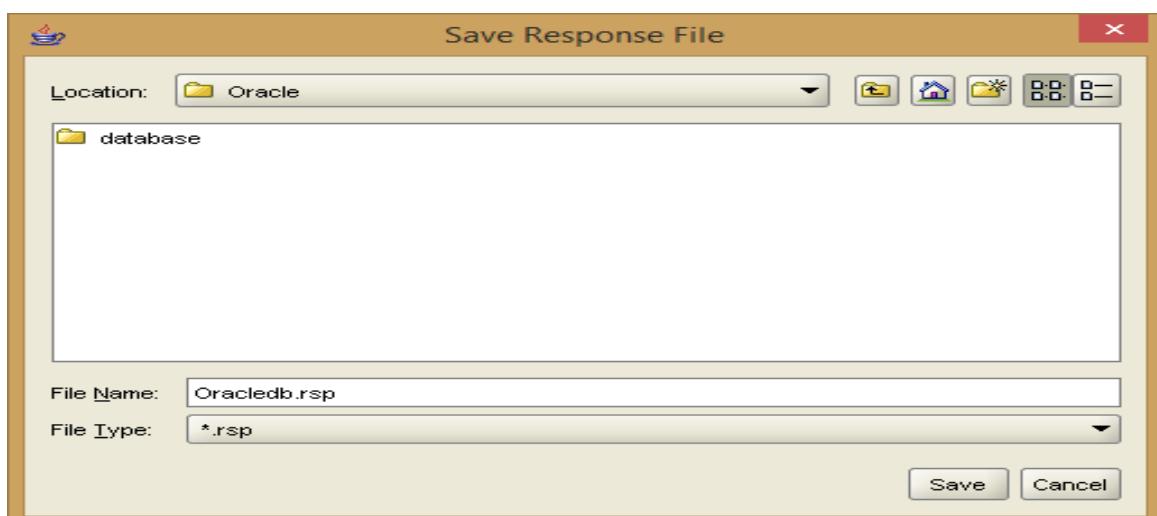
مرحله ۶ نصب دیتابیس اوریکل

در این مرحله، معلومات سیستم داخل شده و وضعیت آماده شدن به نصب برنامه را چک می کند و در صورت عدم وجود کدام مشکلی، صفحه پایین را نشان می دهد. در این صفحه، بعد از مرور معلومات ارائه شده توسط ویزارد، در صورت تأیید استفاده کننده، دکمه ختم این مرحله (Finish) کلیک می شود. با کلیک کردن روی دکمه، انسالیشن یا نصب واقعی سرور آغاز می شود؛ یعنی تا این مرحله، زمینه سازی به خاطر نصب اوریکل صورت گرفته تا در جریان نصب کدام خطایی رخ ندهد.



شکل ۱۵-۱

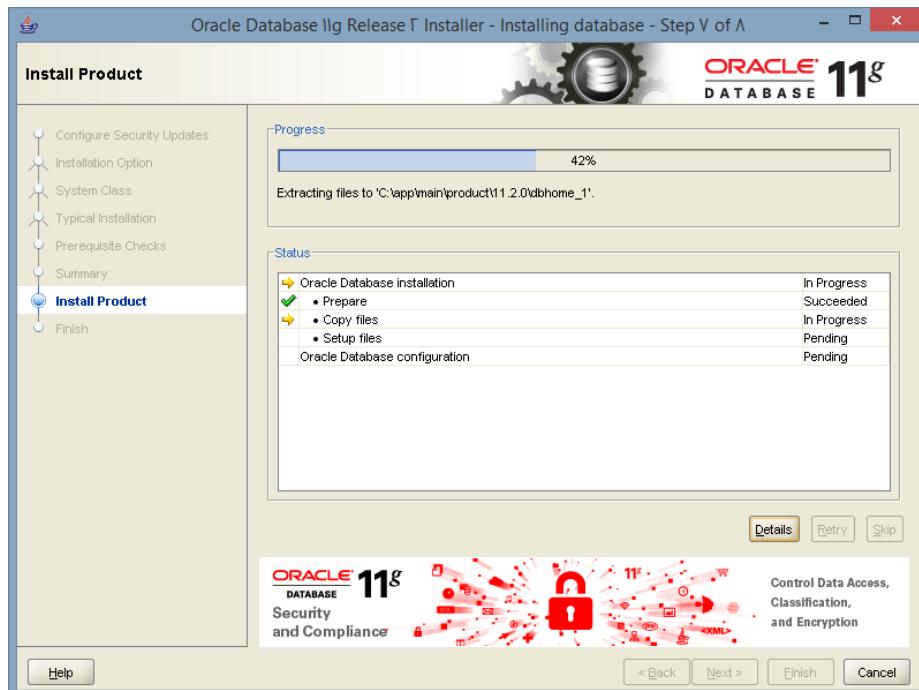
فایل حاوی معلومات، منحیت یک ریفرنس با استفاده از دکمه ... Save Response File... مطابق شکل بالا در کمپیوتر می‌تواند ثبت شود؛ البته این کار اختیاری بوده و شرطی به خاطر نصب برنامه نیست. در صورت ذخیره کردن، فایل معلوماتی به یک نام با اکسشن `rsp` در موقعیتی در کمپیوتر به شکل ذیل، قابل استفاده است.



شکل ۱۶-۱

مرحله ۷ نصب دیتابیس اوریکل

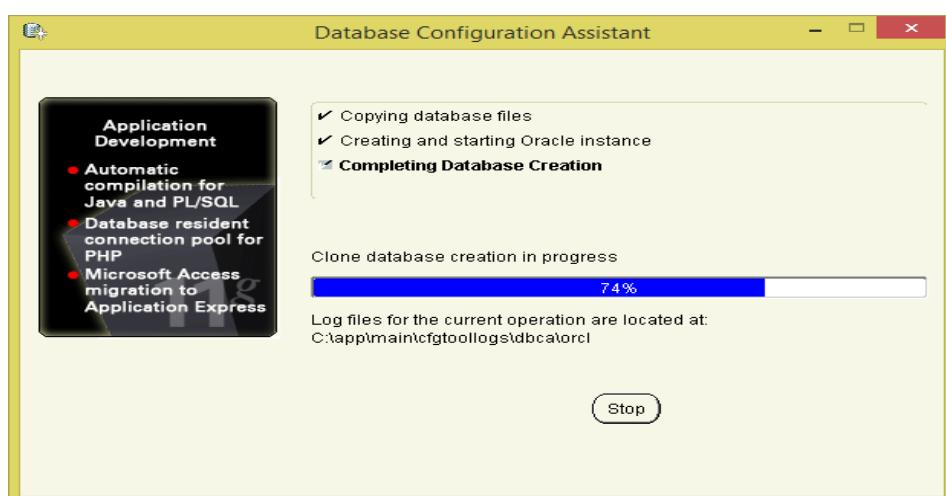
با آغاز شدن نصب از مرحله قبلی، صفحه پایین نشان داده شده و جریان نصب برنامه به خاطر اجرای تنظیمات لازمه، چند دقیقه‌ی را در بر می‌گیرد.



شکل ۱۷-۱

مرحله ۸ نصب دیتابیس اوریکل

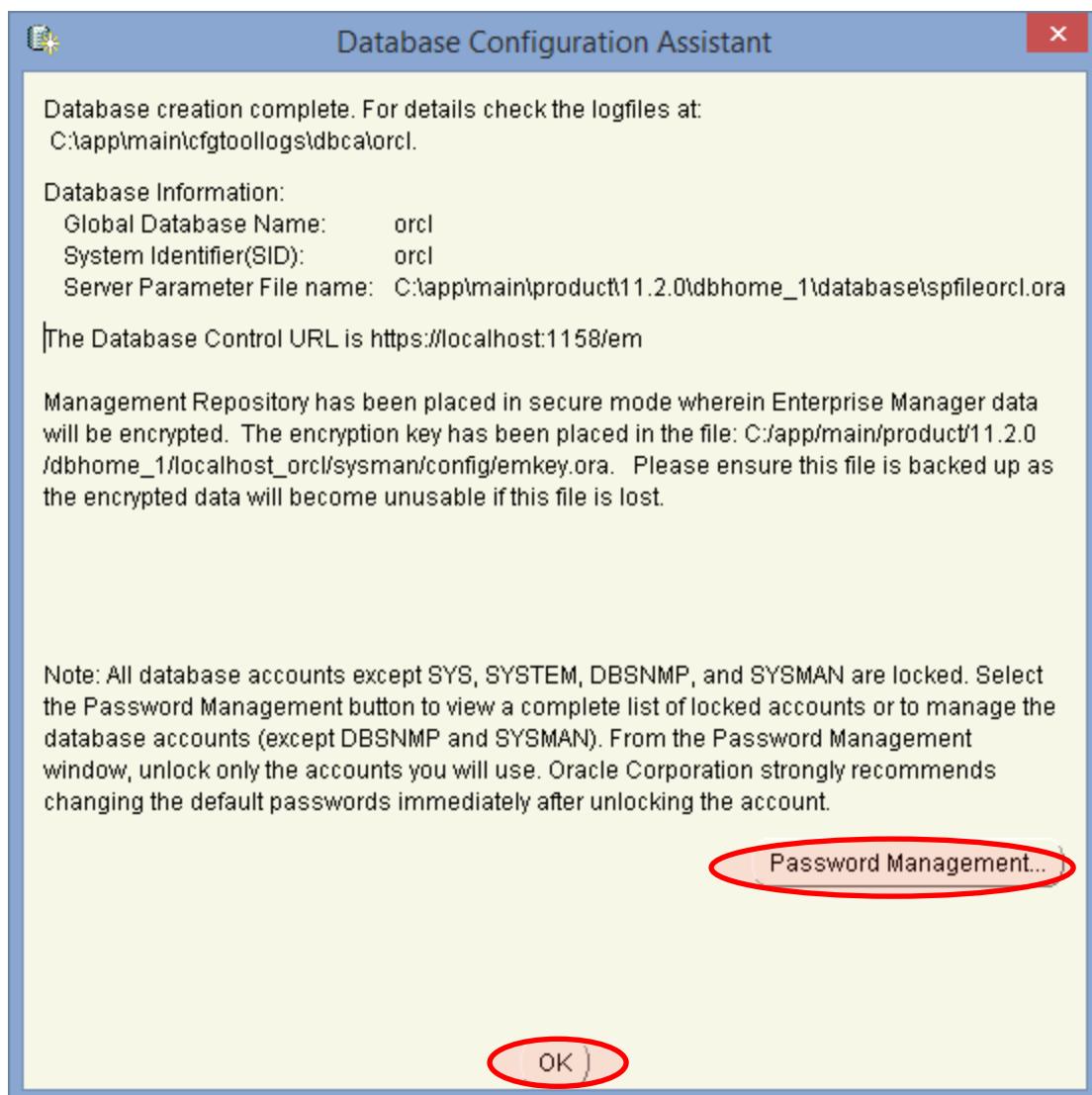
در این مرحله، معلومات ضروری مطابق شکل به خاطر تنظیمات در فایل‌های سیستم کاپی می‌شوند.
این مرحله نیز به چند دقیقه زمان ضرورت دارد.



شکل ۱۸-۱

مرحله ۹ نصب دیتابیس اوریکل

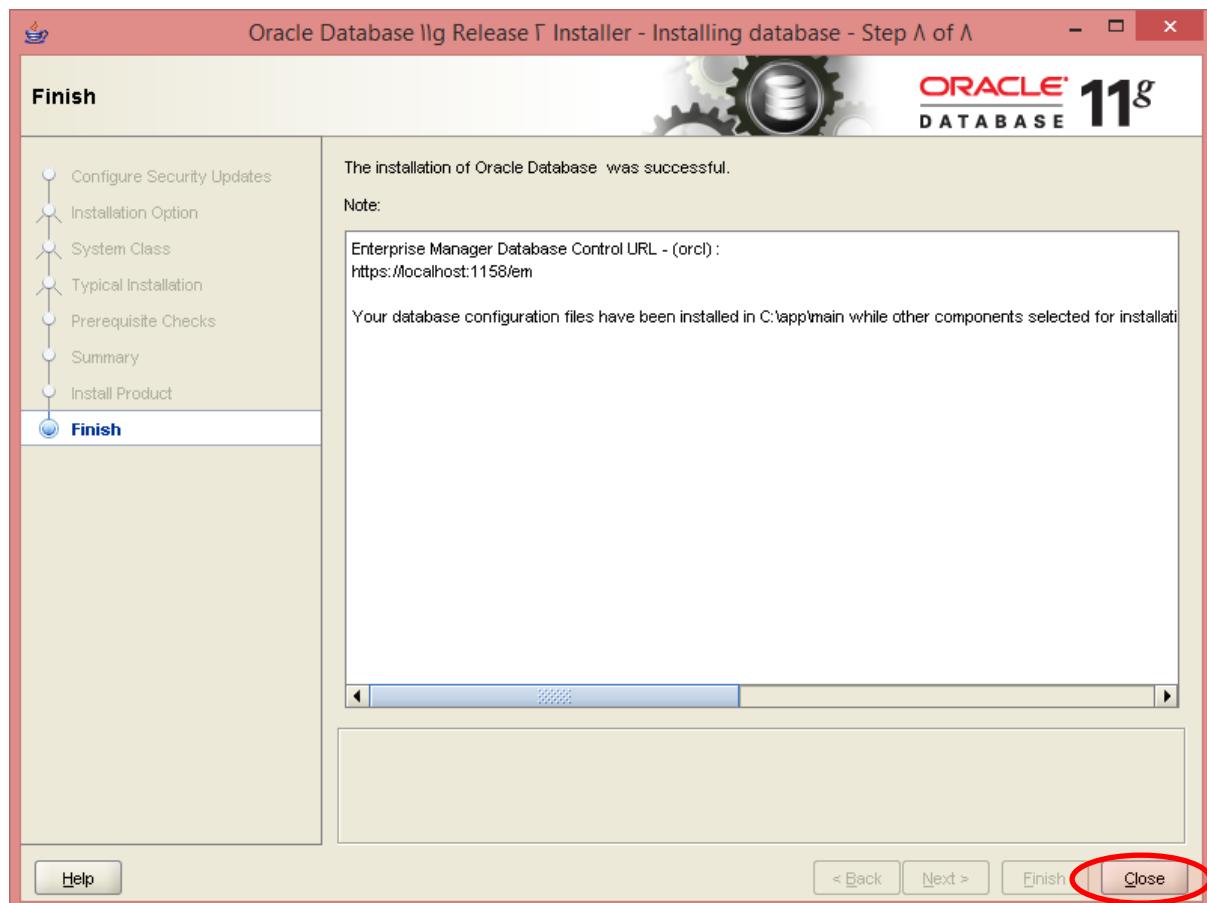
زمانی که کاپی کردن فایل‌های معلومات ضروری دیتابیس تکمیل شد، دیالوگ باکس پایین نشان داده می‌شود. مدیریت رمز عبور (Password) به شکل یک اختیار از طریق این صفحه نیز ممکن است؛ ولی حتمی نیست. با کلیک کردن روی دکمه تأیید (OK) پروسه پیش می‌رود.



شكل ۱۹-۱

مرحله ۱۰ نصب دیتابیس اوریکل

با تکمیل شدن موفقانه نصب دیتابیس اوریکل، صفحه پایین نشان داده می شود. طوری که دیده می شود، تنها اختیار فعال، همان دکمه Close است که با کلیک کردن روی آن، پروسه نصب برنامه اوریکل تکمیل می شود.



شکل ۲۰-۱

مرحله ۱۱ نصب دیتابیس اوریکل

آخرین مرحله نصب کردن اوریکل 11g روی سیستم کامپیوتر شخصی همین مرحله ۱۱ است. در این قسمت، به خاطر اطمینان از نصب شدن درست اوریکل، در کوماند پرامپت (CMD) سیستم‌های عامل ویندوز کد ذیل تایپ شود:

```
sqlplus "/ as sysdba"
```

با اجرای موفقانه دستور، مطابق شکل پایین، پرامپت سیکویل باید در صفحه CMD نشان داده شود. در این صفحه دستورهای کد PS/SQL تایپ شده و به اجراء گذاشته می شوند.

```
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\main>sqlplus "/ as sysdba"

SQL*Plus: Release 11.2.0.1.0 Production on Tue Feb 5 15:29:25 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> _
```

طوری که در شکل بالا دیده می‌شود، دیتابیس اوریکل با موفقیت در سیستم نصب شده است. نمبر نمونه سیستم مدیریت دیتابیس، ساعت و تاریخ وصل شدن به سرور از طریق صفحه CMD نیز در میان معلومات ارائه شده است.

در صورتی که خواسته شود PL/SQL از طریق سیستم عامل به طور مستقیم به راه انداخته شود، از طریق Start Menu نام برنامه پیدا شده و فعال می‌شود. در این طریقه، نام استفاده‌کننده و رمز عبور ضرورت است. در قسمت نام استفاده‌کننده، کلمه SYSTEM تایپ شده و رمز عبور همان پاسورد داده شده در زمان نصب برنامه است. نتیجه بerahandاختن برنامه از طریق Start Menu در شکل پایین نشان داده شده است.

```
SQL*Plus: Release 11.2.0.1.0 Production on Fri Feb 8 16:14:54 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

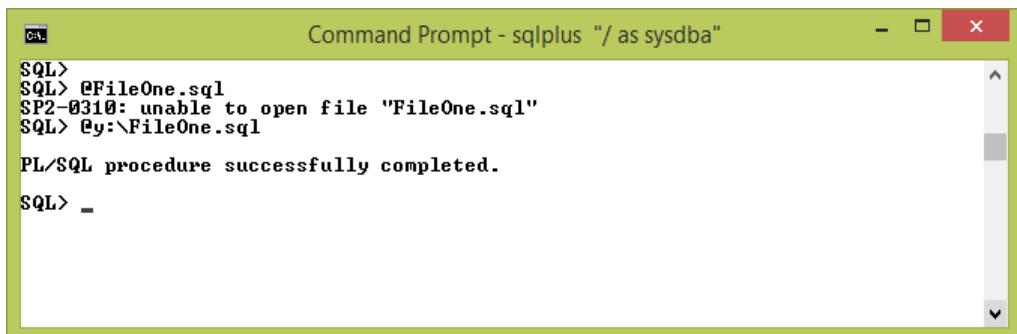
SQL>
```

صفحه کد سیکویل

طوری که در شکل آخر درس قبلی دیده می‌شود، دستورهای PL/SQL در صفحه کوماند پرامپت CMD قابل اجراست. دستورها در این صفحه قابلیت تایپ شدن را دارد. در صورت ایجاد و استفاده از قطعات کد سیکویل که چندین سطر داشته باشد، تایپ کردن شان در این صفحه شاید کار مناسبی نباشد؛ طور مثال، اگر یک اشتباه در یکی از سطرهای بالایی کد سیکویل که در چند سطر تایپ شده باشد، موجود باشد، انتقال کرسر به سطرهای قبلی ممکن نبوده و استفاده کننده ضرورت به تایپ کردن دوباره تمام سطرهای

دستور خواهد داشت. راه حل در چنین مواردی، استفاده از یک فایل متنی به خاطر نوشتن کد سیکویل است. جهت انجام چنین کاری مراحل ذیل باید طی شود:

۱. کد سیکویل در یک برنامه ایدیتر متن مانند EditPlus، Notepad+، Notepad وغیره نوشته شود.
۲. فایل نوشته شده با اکستنشن ".sql" در دایرکتوری home در کمپیوتر ذخیره شود.
۳. برنامه سیکویل پلس از کوماند پرامپت با استفاده از دستور (sqlplus "/ as sysdba") به راه انداده شود.
۴. دستور (@FileName.sql) تایپ شود تا کد حاوی فایل در سیکویل اجراء گردد.



```
SQL> @FileOne.sql
SP2-0310: unable to open file "FileOne.sql"
SQL> Pl/SQL procedure successfully completed.

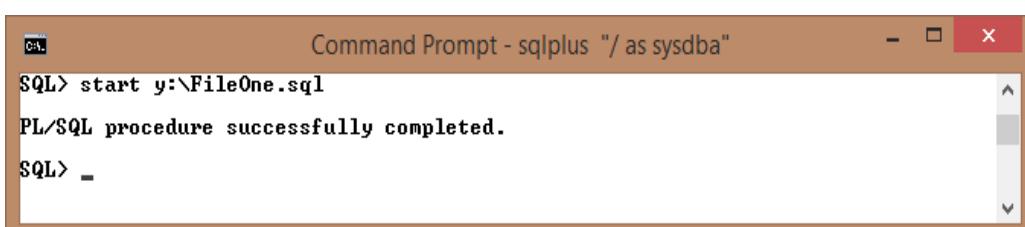
SQL> _
```

طوری که در شکل بالا دیده می شود، فایلی به نام FileOne.sql در دایرکتوری home کاپی نگردیده، بلکه در درایو Y کمپیوتر گذاشته شده است. دستور خواندن فایل متذکره دو دفعه به راه انداده شده است؛ در بار اول سیستم جواب داده است که فایل باز شده نمی تواند و در بار دوم آدرس مکمل فایل به دستور اضافه شده و کد فایل با موفقیت اجراء شده است.

۵. و یا در صورتی که فایل حاوی کد سیکویل، در دایرکتوری home ذخیره نشده باشد، با استفاده از دستور start و به تعقیب آن آدرس مکمل و نام فایل مورد نظر کد فایل به اجراء در می آید؛ طور مثال، به ادامه فعالیت قبلی، دستور پایین، محتوای فایلی به نام FileOne.sql را از آدرس آن در درایو Y کمپیوتر گرفته و کد آن را موفقانه اجراء می کند.

SQL>start Y: \FileOne.sql

نتیجه موفقانه دستور فوق در شکل زیر نشان داده شده است.



```
SQL> start y:\FileOne.sql
Pl/SQL procedure successfully completed.

SQL> _
```

۶. طریقه دیگری که آن نیز در موارد مشابه قابل استفاده است و سهولت کافی نیز دارد، عبارت از کاپی کردن متن از فایل متنی حاوی کد بوده و با راست کلیک کردن روی صفحه سیکویل پلس و کردن در آن به اجراء گذاشته می شود. در چنین حالتی، متن کد نباید دارای فارمت باشد؛ Paste

یعنی کد نوشته شده در فارمت های غنی مانند مایکروسافت ورد قابل تطبیق نباشد، در مقابل متن کاپی شده از صفحه فایل Notepad در صفحه سیکویل پلس قابل اجراء است.

۱.۳ توسعه دهنده سیکویل (SQL Developer)

توسعه دهنده سیکویل یا SQL Developer عبارت از یک وسیله یا پروگرام گرافیکی است که به خاطر ساده ساختن کارهای انکشافی در دیتابیس اوریکل استفاده می شود. این برنامه نیز مانند سیستم مدیریت دیتابیس اوریکل رایگان بوده و از وب سایت رسمی اوریکل قابل دانلود است. وسیله یاد شده ضرورت های استفاده کنندگان اوریکل در زمان کار با دیتابیس ها، ایجاد کردن رابطه بین سرورها و دیتابیس، ایدیت کردن کد PL/SQL، ایجاد کردن راپورهای معلوماتی از دیتا وغیره را از طریق یک محیط مستقل برآورده می سازد. با نصب و استفاده برنامه توسعه دهنده سیکویل در یک کمپیوتر، ضرورت خریداری و نصب برنامه های غیر اوریکل به خاطر استفاده و اجرای دستورهای سیکویل و PL/SQL کاهش می یابد.

با استفاده از SQL Developer ایجاد رابطه به دیتابیس های غیر از اوریکل نیز امکان پذیر است. دیتابیس های غیر اوریکل شامل دیتابیس MySQL، دیتابیس مایکروسافت سیکویل سرور، مایکروسافت اکسیس، دیتابیس سایبیس (Sybase Adaptive Server) و غیره اند. توسط برنامه توسعه دهنده سیکویل ساختمان های دیتابیس های یاد شده (می تادیتا) سیستم های غیر اوریکل قابل دید است؛ به عین ترتیب، دیتا و ساختمان های دیتابیس های غیر اوریکل به کمک SQL Developer می تواند به دیتابیس اوریکل منتقل شود.

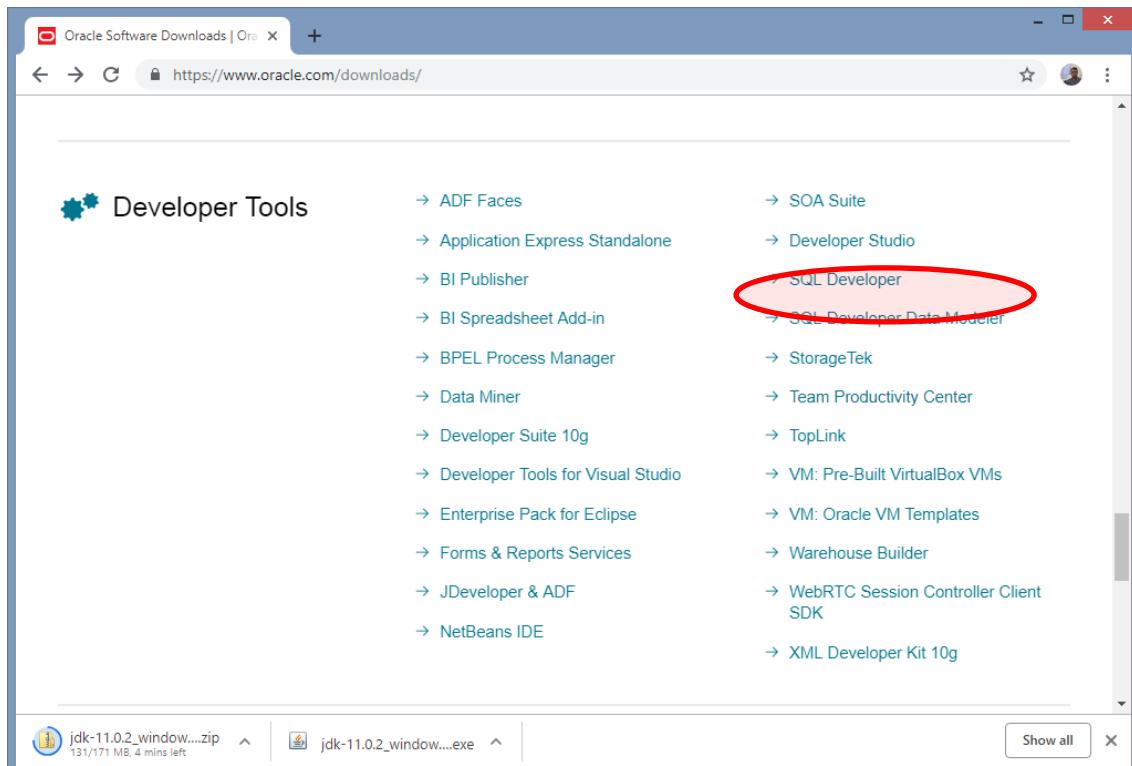
توسعه دهنده سیکویل در محیط جاوا (JDeveloper IDE) ایجاد شده است؛ بناءً به خاطر نصب آن ضرورت به وسیله بی به نام^۱ JDK در کمپیوتر است تا زمینه را به خاطر به اجرا گذاشتن توسعه دهنده سیکویل مساعد سازد. در صورت ضرورت، بسته JDK از منابع آنلاین دانلود و به کمپیوتر مورد نظر باید نصب گردد. آخرین نمونه توسعه دهنده سیکویل، JDK را به همراه داشته و ضرورت به دانلود کردن و نصب جداگانه ندارد.

۱.۳.۱ استفاده کردن توسعه دهنده سیکویل

به خاطر وصل و استفاده کردن از دیتابیس ها در اوریکل، ضرورت به وسیله بی در کمپیوتر است. زمانی که سیستم مدیریت دیتابیس اوریکل در یک کمپیوتر نصب و آماده استفاده گردد، در صورت موجودیت برنامه بی به خاطر وصل شدن به دیتابیس ها، می شود همان برنامه به کار گرفته شود و استفاده کردن از SQL Developer حتمی نمی شود. در صورتی که چنین برنامه بی در کمپیوتر موجود نباشد، با مراجعه به سایت رسمی اوریکل، توسعه دهنده سیکویل به صورت رایگان دانلود و در کمپیوتر استفاده می شود. جهت آغاز کار،

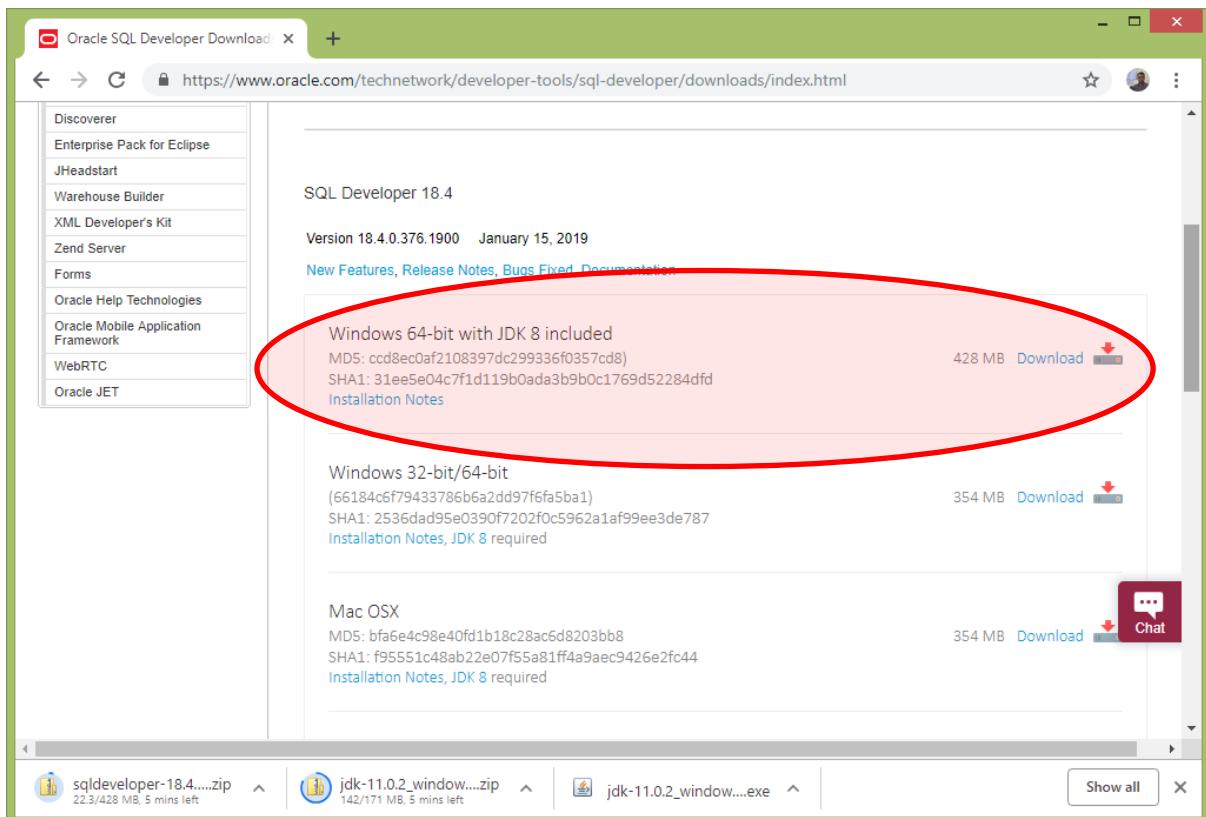
^۱ JDK اختصار کلمات Java Development Kit بوده و به خاطر ایجاد کردن یک محیط جهت به راه انداختن برنامه هایی که به اساس جاوا کار می کنند، استفاده می شود.

یک استفاده‌کننده می‌تواند به سایت اوریکل از طریق آدرس رسمی آن (<http://www.oracle.com>) (رفته و بعد از داخل شدن به کمک نام استفاده‌کننده و رمز عبور، برنامه مورد نظر را مطابق شکل‌های پایین دانلود کند.



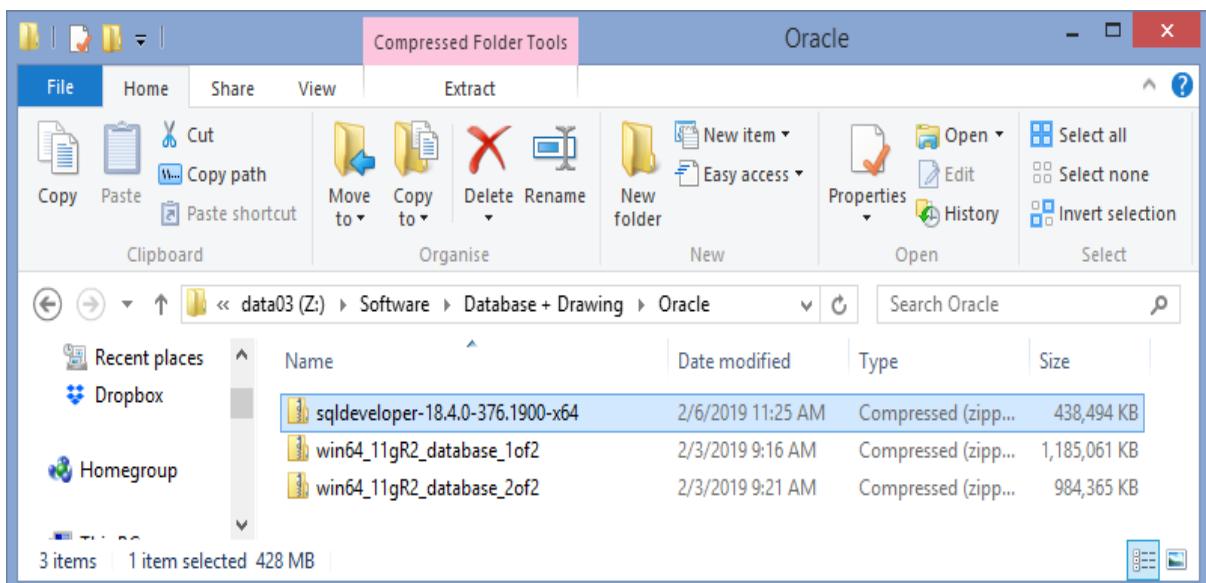
شکل ۲۱-۱

با کلیک کردن روی لینک SQL Developer، در صفحهٔ بعدی لیست ورژن‌های مختلف این برنامه برای محیط‌های مختلف با امکانات مختلف نشان داده خواهد شد. در لیست، آخرین نمونه توسعه‌دهنده سیکویل برای سیستم عامل مورد نظر (در اینجا ویندوز 64 بیتی) انتخاب و روی اختیار دانلود مطابق شکل پایین کلیک می‌شود؛ البته در لینک برنامه SQL Developer همچنان سافت‌ویر JDK که به خاطر به راه‌اندازی آن ضرورت است، وجود دارد. در صورت انتخاب کردن نمونه‌یی از برنامه توسعه‌دهنده سیکویل که بدون JDK باشد، اکسشن جاوا (JDK) می‌تواند به صورت جداگانه و مستقل از همین صفحهٔ رسمی اوریکل دانلود و در سیستم نصب شود.



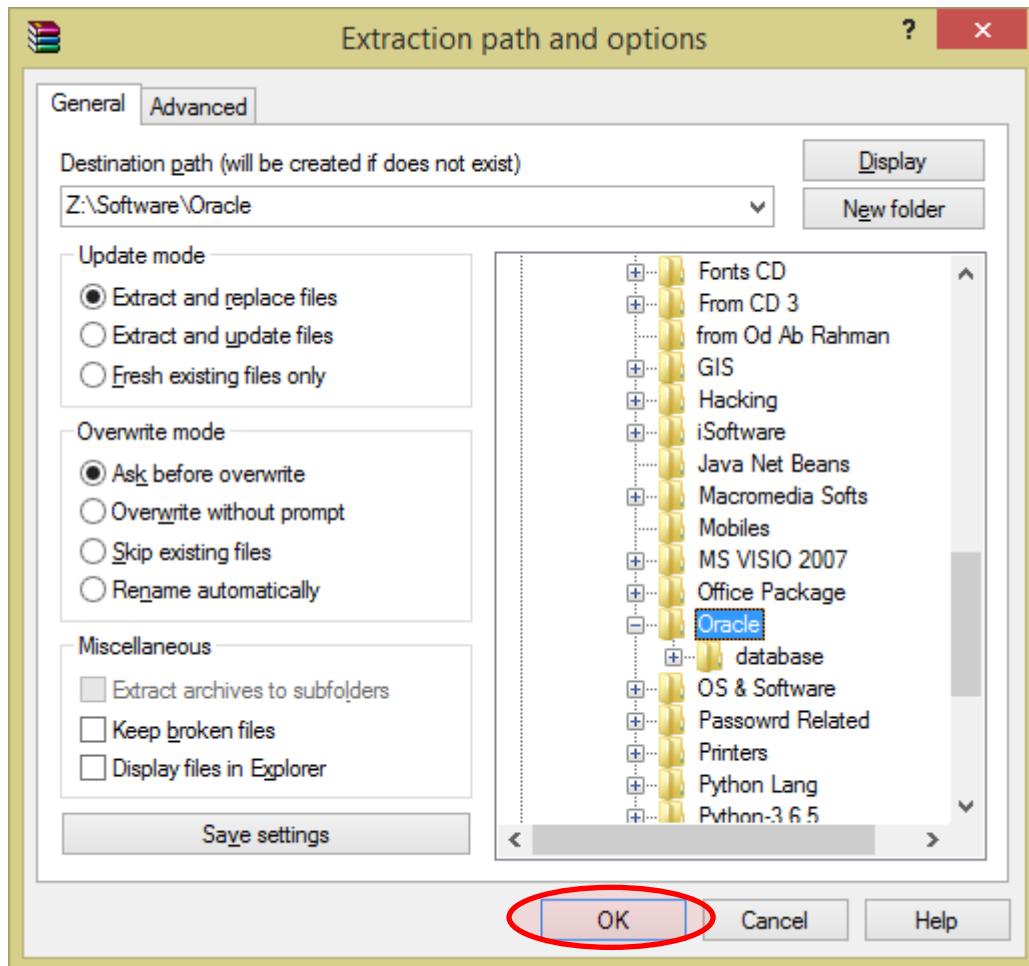
شکل ۲۲-۱

بعد از دانلود شدن موفقانه، وسیله توسعه دهنده سیکویل به شکل زیپ شده مطابق شکل ذیل در کمپیوتر دیده می شود.



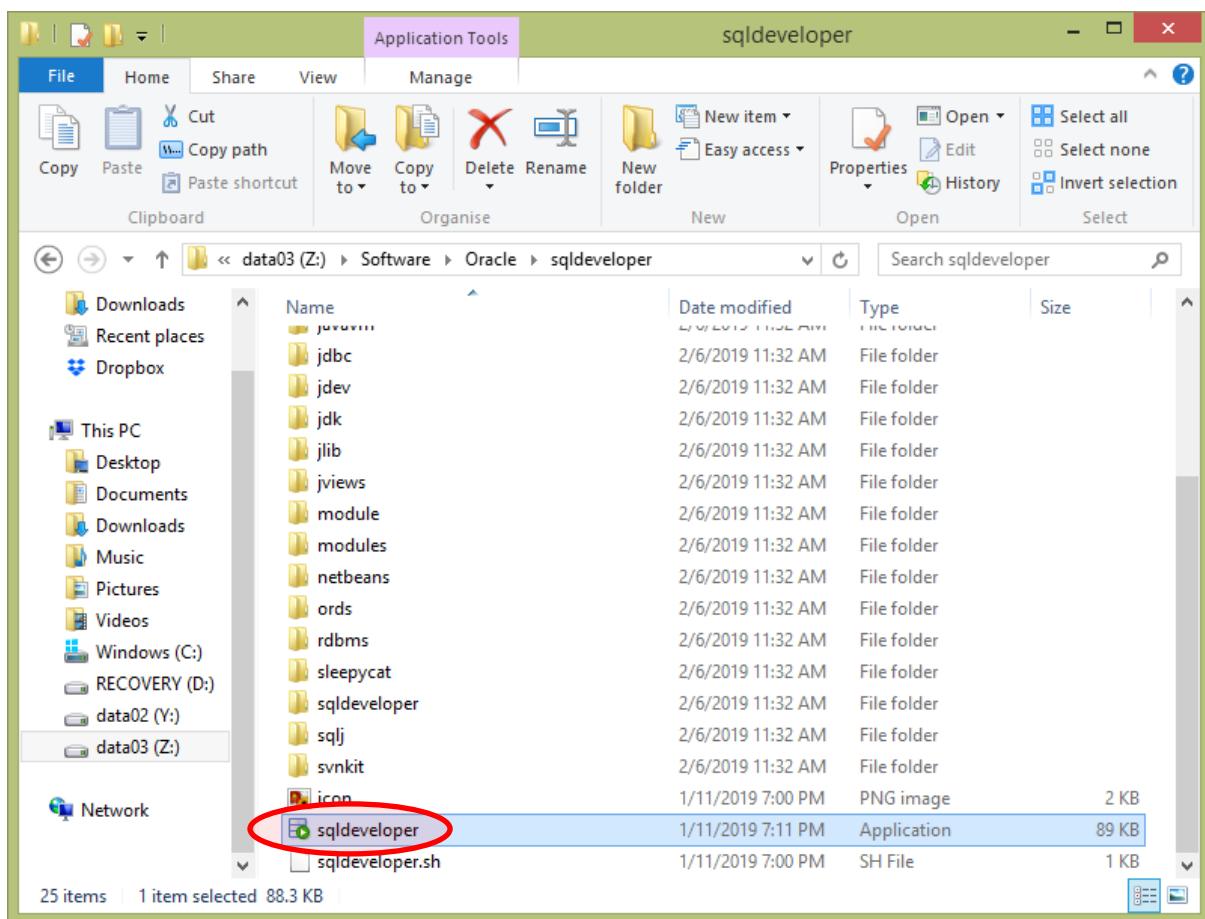
شکل ۲۳-۱

طوری که در درس قبلی سیستم مدیریت دیتابیس اوریکل Unzip شد، به همان ترتیب این برنامه نیز در دایرکتوری مشخصی به نام اوریکل در درایو C و یا هر جای دیگر در کمپیوتر مطابق شکل پایین Unzip می‌شود.



شکل ۲۴-۱

جهت استفاده از توسعه‌دهنده سیکویل به دایرکتوری‌یی که در آن فایل‌ها Unzip شدند، رفته و فایل اجرایی به نام sqldeveloper مطابق شکل پایین به راه انداخته شود. لازم به یادآوری است که این برنامه در کمپیوتر ضرورت به نصب شدن (انсталیشن) نداشته و فقط با اجراء کردن فایل، مطابق مثال اجرا شده، صفحه SQL Developer باز می‌شود.



شکل ۲۵-۱

به خاطر راه اندازی توسعه دهنده سیکویل، مطابق شکل بالا فقط بالای فایل اپلیکیشن دوبار کلیک می شود. به خاطر سهولت کار در سیستم عامل ویندوز یک Shortcut به آن تعریف شده و در موقعیت مناسب مانند صفحه دسکتاپ گذاشته شود. با اجراء شدن فایل sqldeveloper مطابق شکل های پایین، محیط توسعه دهنده سیکویل باز شده و قابل استفاده می گردد.

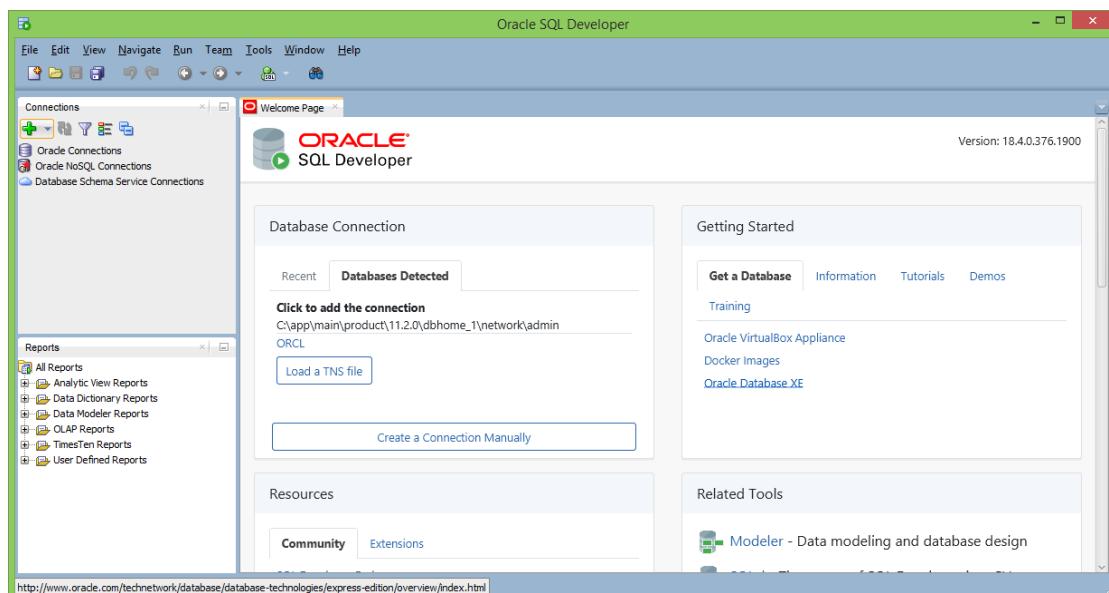


شکل ۲۶-۱

صفحه بالا به خاطر کارکردن با دیتا و مدیریت کردن دیتابیس‌ها استفاده می‌شود. اولین کاری که به خاطر استفاده کردن از دیتابیس توسعه‌دهنده سیکویل انجام می‌شود، عبارت از وصل کردن دیتابیس به این وسیله است. به خاطر اجرای این کار توضیحات لازم در بخش‌های بعدی ارائه شده‌اند.

۱.۴ دیتابیس کاریکل (Working Database)

دیتابیسی که بالای آن از طریق محیط توسعه‌دهنده سیکویل کار می‌شود، به نام دیتابیس کاری این وسیله یاد می‌شود. از طریق صفحه این وسیله، در یک زمان چندین دیتابیس قابل استفاده است.



۲۷-۱ شکل

دیتابیس‌های کاری در توسعه‌دهنده سیکویل به سه دسته عمومی تقسیم می‌شوند: اول دیتابیس‌های رابطه‌یی، دوم دیتابیس‌های غیر سیکویل (NoSQL) و سوم دیتابیس‌های ابری (Cloud).

دیتابیس‌های دسته اول می‌توانند مربوط به سیستم مدیریت اوریکل باشند و یا هم مربوط سیستم‌های مدیریت دیتابیس دیگر مانند اکسس، MySQL، SQL Server، وغیره باشند. شرط اصلی همانا بودن دیتابیس‌ها در فارمت رابطه‌یی (Relational) است؛ یعنی دیتابیس‌های رابطه‌یی که در آن‌ها شرایط مدل رابطه‌یی صدق کند، می‌توانند در دسته اول (اختیار اول) به صفحه SQL Developer وصل شده و به کار گرفته شوند.

در دسته دوم، تنها دیتابیس‌های غیر سیکویل اوریکل به این صفحه وصل و قابل استفاده است. در این اوخر استفاده از دیتابیس‌های غیر سیکویل که با مقادیر بزرگ دیتا با فارمت‌های گوناگون از قبیل فایل‌های

متنی کار می‌کنند، بیشتر شده است. این انکشاف مربوط به استفاده و تنظیم مقادیر بزرگ دیتا متأثر از موجودیت معلومات و سیعی است که ضرورت به تنظیم در کمپیوترها را دارد.

معلومات از شبکه‌های اجتماعی، طور مثال در لحظاتی کم به مقدار بسیار زیاد به سیستم وارد شده و ضرورت به تحلیل دارد. در چنین مواردی، تنظیم و به کارگیری مدل‌های رابطه‌یی دیتابیس‌ها مؤثر نبوده و ضرورت به استفاده از الگوریتم‌های تحلیل دیتای متنی است. تنظیم دیتا با فارمت‌های مختلف (به غیر از فارمت مدل رابطه‌یی) در دیتابیس‌هایی به نام NoSQL صورت می‌گیرد. این کتگوری از دیتابیس‌ها که مربوط اوریکل باشند، از طریق صفحه توسعه‌دهنده سیکویل قابل اتصال بوده و استفاده می‌شوند. فارمت‌های دیگر دیتابیس‌های غیر سیکویل در SQL Developer قابل اتصال و استفاده‌اند.

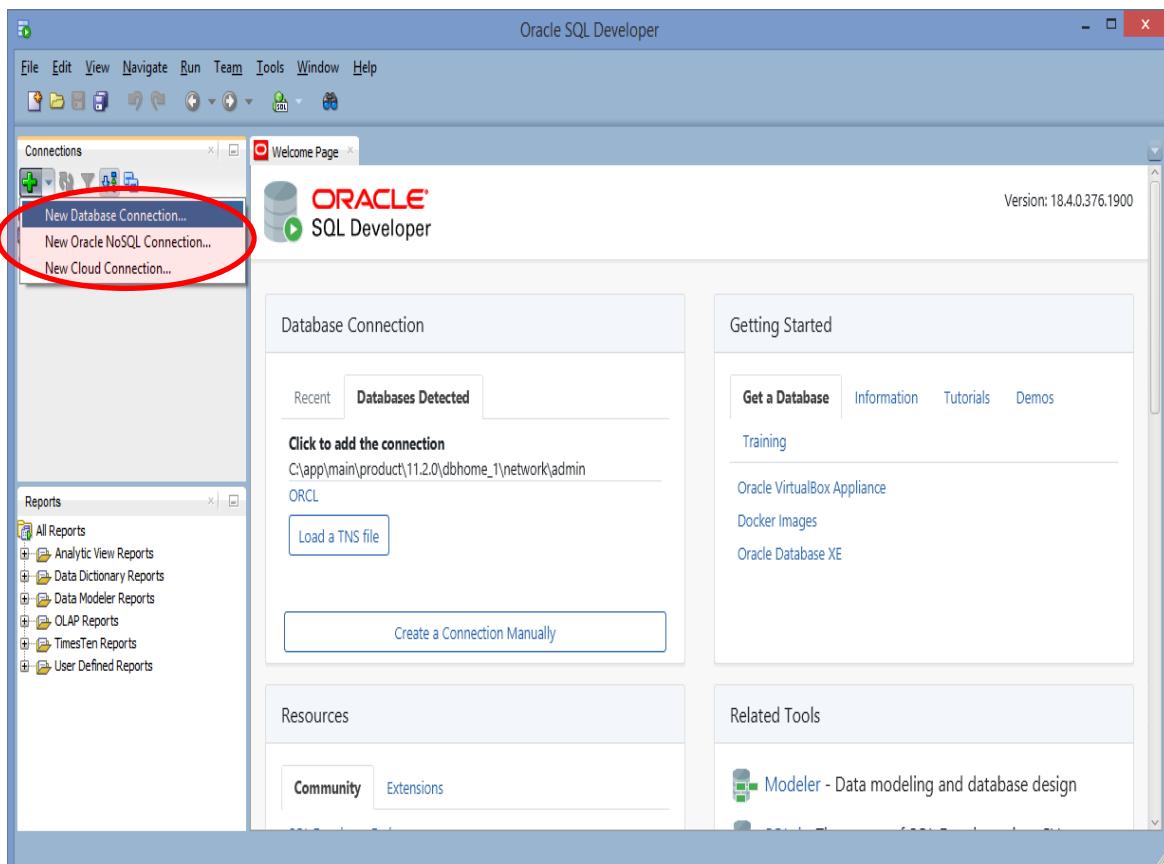
دسته سوم عبارت از استفاده از دیتای ابری در SQL Developer است. به خاطر استفاده‌کردن دیتابیس‌ها در برنامه توسعه‌دهنده سیکویل، دیتابیس‌های یادشده باید به صفحه این وسیله از طریق بخشی به نام Connections وصل شوند. با کلیک کردن روی قسمت معین در صفحه SQL Developer و انتخاب نوع دیتابیسی که باید به سیستم وصل شود، ویژارد مربوطه باز شده و استفاده کننده را کمک می‌کند تا به شکل واضح دیتابیس مورد نظر را به سیستم وصل کند. استفاده و به کارگیری دیتابیس‌های انواع دوم و سوم (دیتابیس‌های غیر سیکویل اوریکل و دیتابیس‌های ابری) خارج از بحث این کتاب‌اند. در اینجا مثال‌ها به نوع اول؛ یعنی استفاده از دیتابیس‌های رابطه‌یی مختص شده و توضیحات لازم در این زمینه ارائه می‌شود.

۱۰.۴.۱ وصل شدن به دیتابیس کاری اوریکل (Connection to Working Database)

توسعه‌دهنده سیکویل یکی از محیط‌هایی است که از طریق آن یک استفاده کننده می‌تواند به دیتابیس‌های کاری اوریکل وصل شود. هر برنامه یا وسیله‌یی که در جهت استفاده مشتری (Client) از یک سیستم دیتابیس قرار داشته باشد، به خاطر استفاده باید به دیتابیسی که در سرور سیستم قرار دارد، وصل شود؛ بناءً اولین کاری که به خاطر اجرای دستورهای سیکویل از انترفیس SQL Developer باید انجام شود، همانا وصل کردن دیتابیس است. همین وصل کردن (Connection) کمک می‌کند تا استفاده کننده (Client) به طور مستقیم، دیتابیس مورد نظر را به کار گیرد.

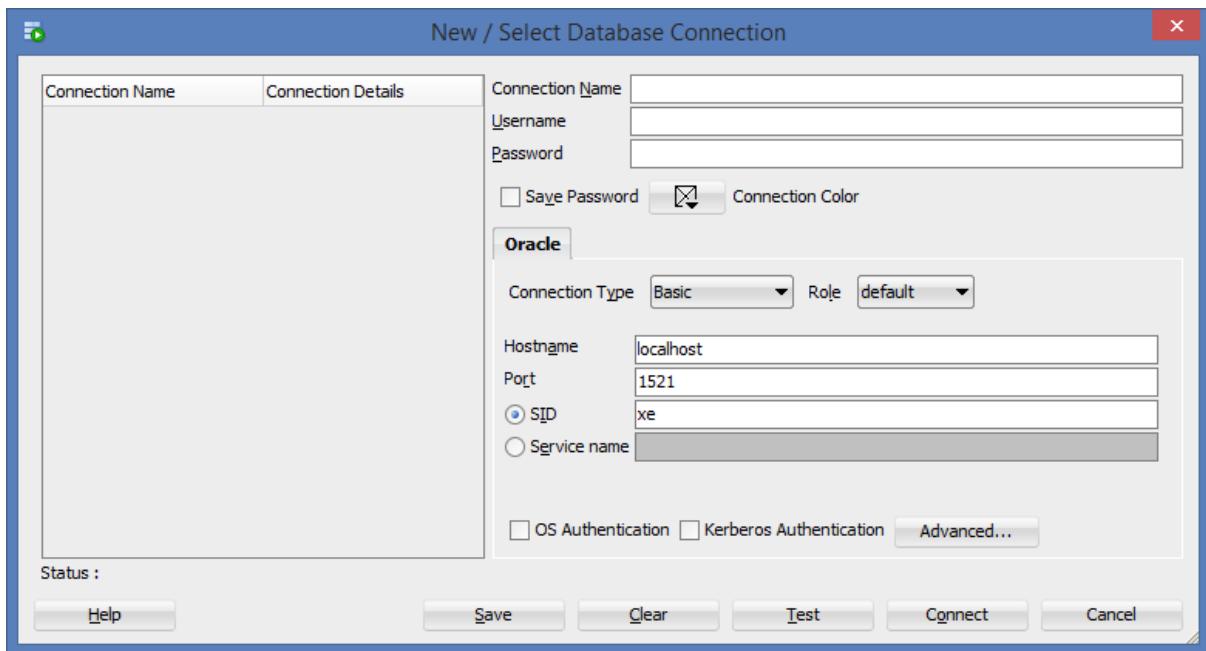
در توسعه‌دهنده سیکویل، Connection به خاطر ایجاد رابطه فعال (active) و غیر فعال (inactive) استفاده می‌شود. زمانی که برای بار اول معلومات مورد ضرورت به خاطر ایجاد کردن رابطه در SQL Developer وارد می‌شود، همان اختیارها و معلومات در سیستم ذخیره شده و در دفعات بعدی ایجاد رابطه به دیتابیس استفاده می‌شود؛ یعنی هر باری که SQL Developer باز می‌شود، ضرورت است تا با دیتابیس رابطه برقرار شده و مورد استفاده قرار گیرد. اگر به یک دیتابیس قبل از برقرار شده باشد، معلومات در مورد آن ذخیره بوده و صرف با وارد کردن نام و رمز عبور (پاسورد) رابطه بین توسعه‌دهنده سیکویل و دیتابیس مورد نظر برقرار شده و دیتابیس می‌تواند مورد استفاده قرار گیرد.

در صفحه توسعه‌دهنده سیکویل، مطابق شکل پایین، در سمت چپ به طرف بالا علامت مثبت تحت عنوان Connections به رنگ سبز دیده می‌شود. با کلیک کردن روی آن سه اختیار وصل‌شدن به دیتابیس، وصل‌شدن به غیر سیکویل اوریکل و وصل‌شدن به دیتابیس ابری (Cloud) نشان داده می‌شود. این همان سه نوع اختیار وصل‌شدن به منبع دیتاست که در بحث قبلی در مورد آن توضیحات ارائه شد. به خاطر وصل‌شدن به دیتابیس اوریکل که از نوع دیتابیس رابطه‌یی است، اختیار اول New Database Connection ... استفاده می‌گردد.



شکل ۲۸-۱

با کلیک کردن روی گزینه مورد نظر، صفحه ویژارد ایجاد‌کردن رابطه جدید (New Connection Wizard) مطابق شکل پایین باز می‌شود. در این صفحه معلوماتی در مورد اتصال جدید ضرورت است. این معلومات به خاطر استفاده‌های بعدی در سیستم ذخیره می‌شود؛ یعنی در ایجاد رابطه‌های عادی بین توسعه‌دهنده سیکیول و یک منبع دیتا، در صورت عدم تغییر در نام و رمز دسترسی به منبع مذکور، معلومات این صفحه ذخیره و دوباره استفاده می‌شود.



شکل ۲۹-۱

در صفحه ایجاد کردن رابطه بین توسعه دهنده سیکویل و یک دیتابیس برای بار نخست، معلومات چهار فیلد باید توسط استفاده کننده داخل گردد.

۱. نام رابطه (Connection Name)

نام رابطه به مقصد شناختن رابطه در محیط توسعه دهنده سیکویل است تا در آینده به وضاحت قابل استفاده باشد. در مثال کتاب، نام system استفاده شده است و آن به خاطر استفاده کننده بخش سیستم است.

۲. نام استفاده کننده (Username)

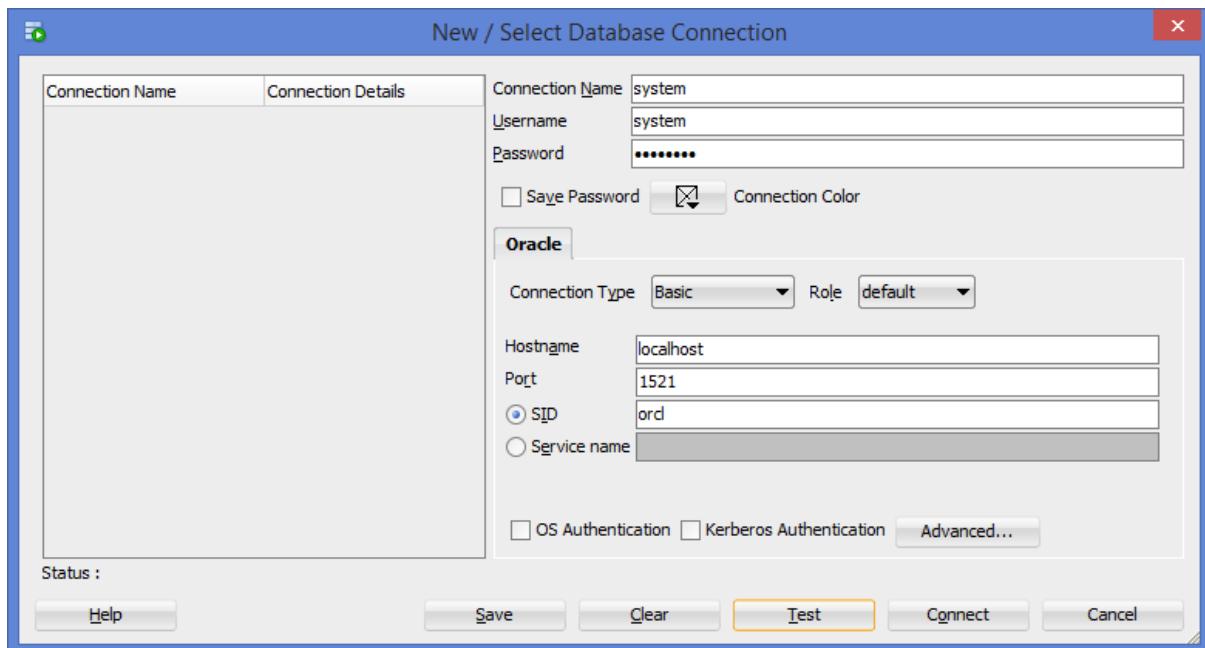
برای نام استفاده کننده نیز کلمه system استفاده شده است؛ به این خاطر که در این مثال استفاده کننده مرکزی در نظر گرفته شده است.

۳. رمز عبور (Password)

رمز عبور نیز همان کلمه یا کلماتی تایپ شوند که در جریان نصب برنامه اوریکل به کار برد شده اند.

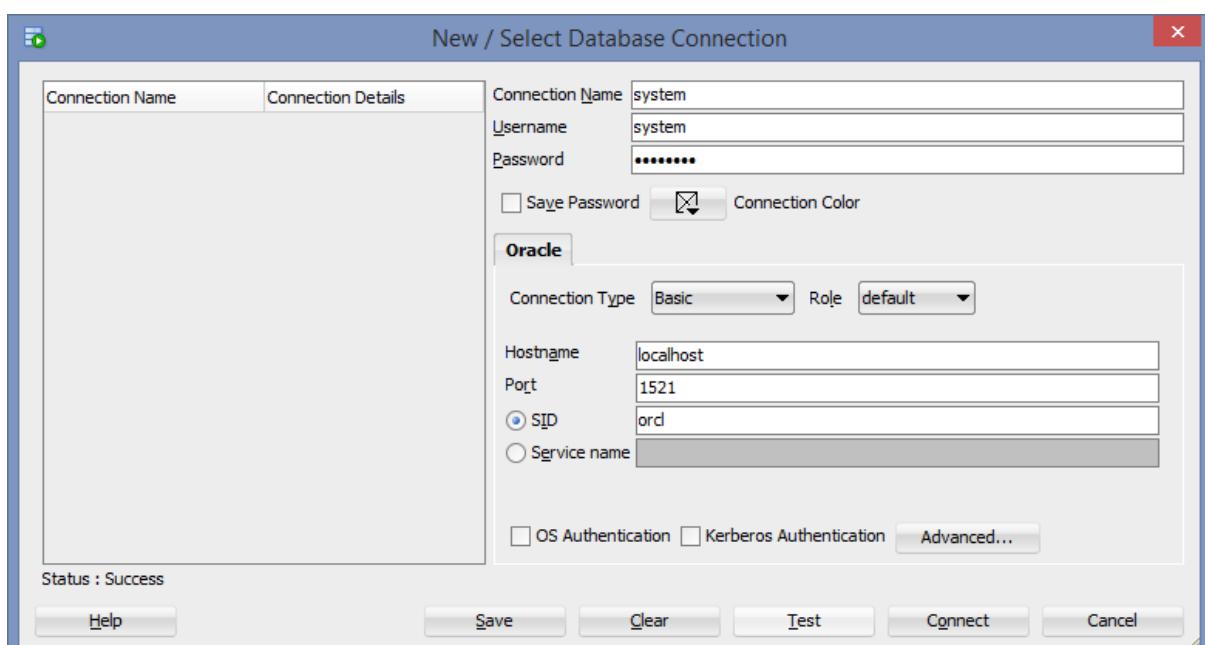
۴. شناسایی سرویس (SID)

در این قسمت همان نامی که برای دیتابیس عمومی در جریان نصب اوریکل استفاده شده بود، تایپ شود. مطابق مثال کارشده، نام دیتابیس عمومی عبارت از orcl بود. در صورت استفاده از کدام نام دیگر، همان نام باید در اینجا به کار برد شود. در صورت نصب نمونه های Oracle Express Edition، در این قسمت نام سرویس عبارت از xe خواهد بود. اختیارهای دیگری که در صفحه به طور خودکار تعیین شده اند، به حالت خود گذاشته شده و در آنها ایجاد کدام تغییری ضرور نیست؛ به هر صورت، به اساس مثال های کارشده، صفحه ایجاد کردن رابطه جدید مطابق شکل پایین، خانه پری می شود.



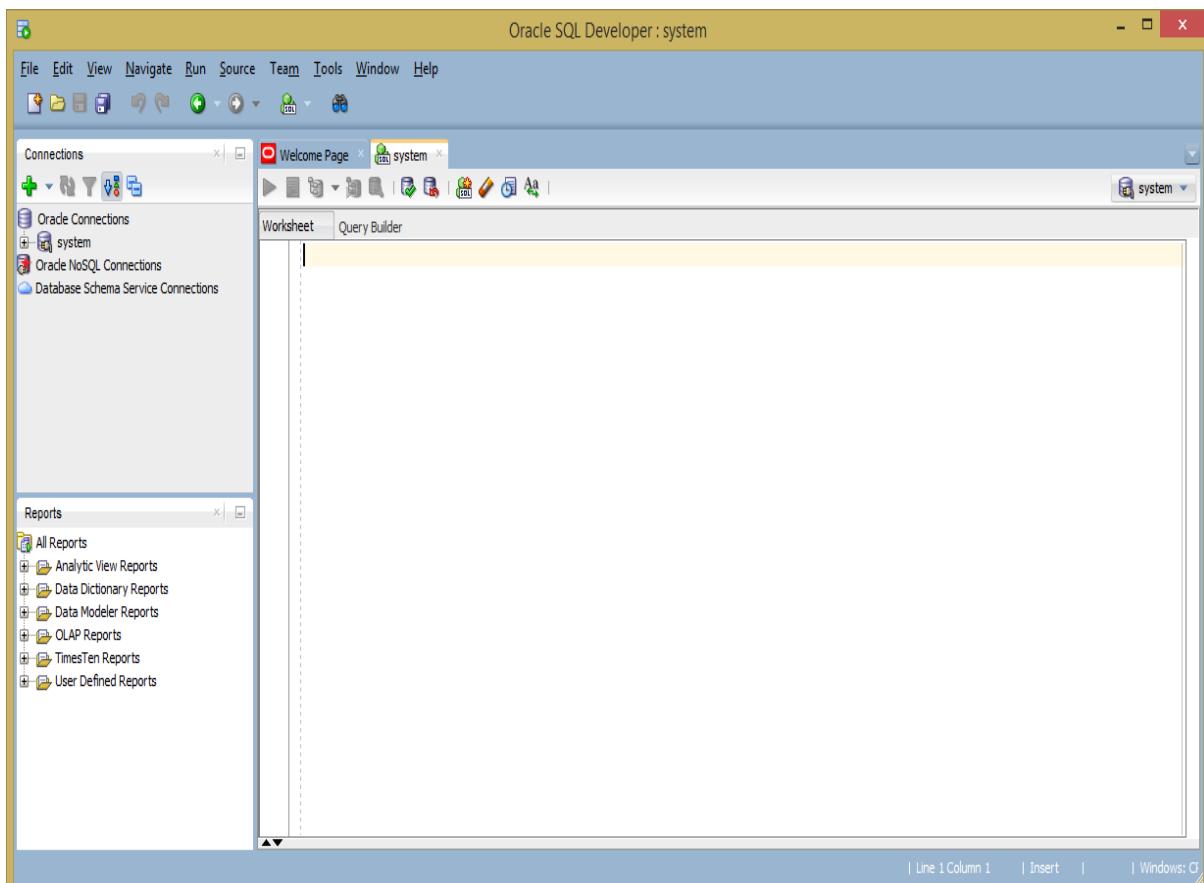
شکل ۳۰-۱

قبل از فشاردادن دکمه Connect به خاطر ایجادکردن رابطه به دیتابیس مورد نظر، پرسه باید یکبار امتحان شود. به خاطر انجام این کار، دکمه‌یی به نام Test روی صفحه موجود است که با کلیک کردن روی آن امکان ایجادشدن رابطه، طی مراحل می‌شود. در صورت عدم بروز مشکل، در قسمت پایین سمت چپ همین صفحه، گزارش موفقیت پرسه مطابق شکل پایین ارائه می‌شود و در صورت وجود کدام مشکل، گزارش عدم موفقیت پرسه به همراه معلومات در مورد مشکل در همین قسمت نشان داده می‌شود. در چنین حالتی به خاطر موفقیت در ایجاد رابطه، باید مشکلات موجود در آن حل گردد.



شکل ۳۱-۱

با درنظرگرفتن شکل بالا و موفقیت در پروسه امتحان ایجادکردن رابطه به دیتابیس، دکمه وصل (Connect) کلیک شود. با انجامدادن این کار توسعهدهنده سیکویل صفحه خالی قابل تایپشدن کد سیکویل را مطابق شکل پایین بازخواهد کرد.



شکل ۳۲-۱

در صفحه شکل بالا، دستورهای سیکویل قابل تایپ و کاپی شدن هستند که همین دستورها اجراء شده و نتایجشان نیز در همین صفحه نشان داده می‌شوند. در این قسمت گفته می‌شود که دیتابیس به وسیله توسعهدهنده سیکویل به صورت موفقانه وصل شده است. هر باری که SQL Developer باز می‌شود، رابطه‌یی به نام سیستم در آن دیده می‌شود. با کلیک کردن (فعال ساختن) آن، سیستم نام استفاده‌کننده و رمز عبور را خواسته، در صورت ارائه کردن معلومات درست، رابطه به دیتابیس ایجاد خواهد شد.



خلاصه فصل اول

در این فصل، به موضوعات اولیه در قسمت استفاده نظری و عملی از سیستم مدیریت دیتابیس اوریکل پرداخته شد. توضیحات لازم در قسمت دیتابیس اوریکل و نمونه‌های مختلف آن ارائه شدند؛ معلومات در مورد این که یک دیتابیس اوریکل حداقل به چه امکانات سخت‌افزاری ضرورت دارد و نیز آمادگی‌های لازم به خاطر نصب کردن دیتابیس اوریکل در عناوین مختلف توضیح داده شدند. این که دیتابیس اوریکل منحیث یک سیستم رایگان چطور و از کجا باید دانلود شود و نیز طریقه‌های نصب و عیارسازی اوریکل با ارائه تصویرهای از صفحات ویزارد نصب برنامه با معلومات لازم نشان داده شدند.

در ادامه فصل، توضیحات لازم در قسمت وسیله‌یی که توسط آن یک دیتابیس اوریکل به سرور وصل و قابل استفاده می‌شد، ارائه گردید. در این بحث به وسیله‌یی به نام توسعه‌دهنده سیکویل (SQL Developer) اشاره گردید و تشریحات لازم در قسمت استفاده از آن داده شد. نمونه عملی وصل کردن دیتابیس از طریق نام استفاده‌کننده با رمز عبور در این وسیله نشان داده شد؛ همچنان امکانات استفاده از دیتابیس‌های مختلف شامل دیتابیس‌های رابطه‌یی، دیتابیس‌های غیر سیکویل (NoSQL Databases) و دیتابیس‌های ابری (Cloud) در این وسیله توضیح گردید و موضوع استفاده از آن مطابق محتوای این کتاب به دیتابیس‌های رابطه‌یی تمرکز داده شد.



سوالات فصل اول

۱. چه اشخاصی واجد شرایط دانلود و استفاده کردن دیتابیس اوریکل‌اند؟
۲. دو نمونه از سیستم‌های عامل را نام بگیرید که دیتابیس اوریکل در آن‌ها استفاده شود.
۳. حداقل مقدار حافظه RAM و ساحة خالی روی هارد دیسک به خاطر نصب کردن دیتابیس اوریکل چقدر است؟ بیان کنید.
۴. آخرین نمونه اوریکل که عبارت از ورژن ۱۲c است، بالای سیستم‌های عامل چند بیتی نصب می‌شود (۶۴ بیتی و یا ۳۲ بیتی)؟
۵. آیا داخل کردن ایمیل استفاده کننده در مرحله اول نصب اوریکل حتمی است و یا خیر؟ ایمیل چرا توسط پروگرام خواسته می‌شود؟
۶. در مرحله ۴ نصب کردن اوریکل، اختیار دومی System Class چیست؟ صرفاً نام گرفته و هدف آن را مختصراً توضیح دهید.
۷. توسعه‌دهنده سیکویل (SQL Developer) به چه منظور استفاده می‌شود؟
۸. آیا امکان ایجاد رابطه به دیتابیس‌های غیر اوریکل از طریق SQL Developer مساعد هست؟ اگر جواب بلی است، با کدام نوع دیتابیس‌ها می‌توان رابطه ایجاد کرد؟
۹. توسط توسعه‌دهنده سیکویل کدام سه دسته از دیتابیس‌ها وصل شده و قابل استفاده هستند؟ صرفاً نام ببرید.
۱۰. آیا توسعه‌دهنده سیکویل در یک کمپیوتر نصب می‌شود و یا مستقیماً استفاده می‌شود؟ واضح سازید.

فعالیت های فصل اول



۱. از طریق آدرس <http://www.oracle.com> به وبسایت رسمی اوریکل بروید.
۲. یک حساب استفاده کننده به اساس ایمیل آدرس تان ایجاد کنید (به خاطر درنظرگرفتن مسائل امنیتی، رمز عبور اوریکل از رمز عبور ایمیل تان باید متفاوت باشد).
۳. به وبسایت رسمی اوریکل از طریق نام استفاده کننده (ایمیل آدرس تان) و رمز عبور جدید داخل شوید.
۴. به صفحه دانلود رفته و از آن جا دیتابیس اوریکل را در مطابقت با سیستم عامل کمپیوتر خود پیدا کرده و دانلود کنید (در صورت ضعیف بودن سرعت اینترنت از انجام دادن این فعالیت صرف نظر کرده و نرم افزار اوریکل را از استاد مضمون تان بگیرید).
۵. از همین وبسایت، توسعه دهنده سیکویل (SQL Developer) را نیز دانلود کنید.
۶. فایل های زیپ شده نرم افزار دیتابیس اوریکل را در یک دایرکتوری مشخص در کمپیوترتان، Unzip کنید.
۷. به فولدر دیتای Unzip شده رفته و سیستم مدیریت دیتابیس اوریکل را در کمپیوترتان نصب کنید.
۸. با استفاده از دستور CMD صفحه کوماند پرامپت را باز کنید.
۹. در این صفحه دستور "sqlplus "/ as sysdba" را تایپ کرده و به راه اندازید.
۱۰. با استفاده از دستور کمک (help index) اندکس، بخش های داخل سیکویل پلس را ببینید.
۱۱. بالای فایل اجرایی sqldeveloper در دایرکتوری توسعه دهنده سیکویل کلیک راست کرده و اختیار Create Shortcut را انتخاب کنید.
۱۲. شارت کت این برنامه را در موقعیت دلخواه کمپیوترتان که به آسانی قابل دسترس باشد، مانند صفحه دسکتاپ انتقال بدھید.
۱۳. با استفاده از فایل اجرایی sqldeveloper و یا هم با استفاده از شارت کت آن، وسیله توسعه دهنده سیکویل را به راه اندازید.
۱۴. با استفاده از اختیار Connections در قسمت چپ و بالایی صفحه توسعه دهنده سیکویل یک رابطه به دیتابیس اوریکل ایجاد کنید (برای نام استفاده کننده کلمه system را به کار ببرید و به خاطر رمز عبور همان پاسوردی که در زمان انسٹالیشن استفاده شده است، به کار بردشود).

فصل دوم

قابلیت‌های PL/SQL (PL/SQL Capabilities)



هدف کلی: شاگردان با قابلیت‌های PL/SQL آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

- .1. خصوصیات PL/SQL را شرح دهند.
- .2. موارد استفاده از PL/SQL را توضیح دهند.
- .3. فواید PL/SQL را شرح دهند.
- .4. بلاک‌ها را در PL/SQL بفهمند.
- .5. پیام‌های خروجی PL/SQL را بیاموزند.

زبان PL/SQL یک زبان پروسیجری به خاطر کار با دیتا است. این زبان در دیتابیس‌های اوریکل مورد استفاده قرار می‌گیرد. در PL/SQL متحولین تعریف شده و استفاده می‌شوند؛ به عین ترتیب در آن از حلقه‌ها (Loops) استفاده می‌شود، غلطی‌های کد پیدا می‌شود و غیره مواردی که در یک زبان پروگرامنویسی قابلیت اجراء را دارند، در PL/SQL نیز تطبیق می‌شوند.

زبان PL/SQL دارای خصوصیات مرتبط به خود می‌باشد و موارد استفاده از این زبان نیز مشخص است. مانند زبان‌های دیگر برنامه‌نویسی، این زبان هم دارای فوایدی است که در جریان فصل به آن‌ها پرداخته شده است. ساختمان‌های دستورهای زبان PL/SQL به نام بلاک‌ها یاد می‌شوند. بلاک‌های کد سیکویل عبارت از ساختمان‌هایی‌اند که در اوریکل به اجراء گذاشته می‌شوند. زبان PL/SQL مانند زبان‌های دیگر دارای دستورهایی به خاطر نشان‌دادن پیام‌های خروجی است. در قسمت آخر فصل همین موضوع با ذکر مثال‌هایی توضیح داده شده است.

۲.۱ خصوصیات PL/SQL

زبان PL/SQL عبارت از یک زبان برنامه‌نویسی است که در محیط سیستم مدیریت دیتابیس اوریکل استفاده می‌شود. این زبان از خصوصیات و ویژگی‌هایی برخوردار است که باعث شده تا استفاده‌کنندگان آن را با رضایت خاطر به کار گیرند.

۱. به خاطر استفاده کردن از کلمات زبان نوشتاری انگلیسی، نحوه نوشتن PL/SQL، واضح و قابل فهم است.

۲. به کارگیری ساختمان‌های جدید در دستورهای این زبان به موارد زیر کمک کرده است:

a. اجرای (Performance) بهتر پروگرام‌های آن؛

b. کارایی (Functionality) بهتر نسبت به محیط‌های غیر از آن؛

c. انکشاف قابلیت استفاده (Usability) در جهت برنامه‌نویسی در آن.

۳. زبان PL/SQL با سیکویل به صورت وسیع ترکیب شده است؛ یعنی تمام مزایای زبان سیکویل در این زبان نیز موجود بوده و قابل تطبیق است.

۴. اجازه چک و پیداکردن اشتباهات موجود در کد سیکویل را به صورت تفصیلی می‌دهد. در محیطی که در آن PL/SQL استفاده می‌شود، اختیارهای کامل پیداکردن غلطی‌های کد سیکویل موجوداند.

۵. انواع زیاد دیتا (Data Types) در این زبان موجود بوده و قابل استفاده است.

۶. ساختمان‌های مختلف برنامه‌نویسی در PL/SQL موجود بوده و قابل استفاده‌اند.

۷. برنامه‌نویسی ترکیبی (Structured Programming) با امکانات استفاده از تابع‌ها و پروسیجرها در PL/SQL مساعد است.

٨. برنامه‌نویسی شی‌گرا (Object Oriented Programming) توسط PL/SQL حمایت می‌شود.
٩. و در نهایت در PL/SQL اکشاف صفحات و اپلیکیشن‌های وب و صفحات سرورها به صورت درست آن‌ها تطبیق می‌شوند.

٢.٢ موارد استفاده از PL/SQL

زبان PL/SQL اساساً به خاطر استفاده کردن در محیط سیستم مدیریت دیتابیس اوریکل دیزاین شده است. این زبان در موارد مختلف به خاطر به راه‌انداختن دستورهای سیکوبل به کار برده می‌شود. در ذیل در رابطه به موضوع استفاده از PL/SQL، به مواردی اشاره و مختصراً به آن‌ها پرداخته شده است:

١. زبان PL/SQL قابل حمل (Portable) بوده و در آن ترانزکشن‌ها قابل اجراء است. با استفاده از ترانزکشن‌ها، دستورهایی که با تغییرآوردن در دیتا مانند داخل کردن دیتا به دیتابیس، بهروزکردن دیتا و پاک کردن دیتا از دیتابیس سروکار داشته باشد، کنترول می‌شوند.
٢. این زبان از جمله زبان‌هایی به شمار می‌رود که با سرعت اجرای بالا (High Performance) کار می‌کند.
٣. زبان PL/SQL یک محیط مستقل برای استفاده کنندگان مهیا می‌سازد که بستگی به محیط بیرونی (سیستم عامل) ندارد. محیط این زبان به شکل Built-in به خاطر اجرای دستورهای مربوطه مورد استفاده قرار می‌گیرد.
٤. PL/SQL را می‌توان هم از محیط CMD یا محیط دستوری ویندوز و هم از طریق انترفیس SQL*Plus استفاده کرد. PL/SQL از طریق انترفیس SQL*Plus مطابق شکل پایین قابل استفاده می‌گردد.

The screenshot shows the SQL*Plus interface with the title bar "SQL Plus". The command line area displays the following text:

```

SQL*Plus: Release 11.2.0.1.0 Production on Fri Feb 8 12:08:51 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Enter user-name: system
Enter password:

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> ? index
Enter Help [topic] for help.

      COPY          PAUSE          SHUTDOWN
      DEFINE        PRINT          SPOOL
      /             PROMPT        SQLPLUS
ACCEPT        DESCRIBE       QUIT          START
APPEND       DISCONNECT     RECOVER        STARTUP
ARCHIVE LOG  EDIT           REMARK         STORE
ATTRIBUTE    EXECUTE        REPFOOTER     TIMING
BREAK        EXIT           REPHEADER     TITLE
BTITLE       GET            RESERVED WORDS <SQL>
CHANGE       HELP           RESERVED WORDS <PL/SQL> UNDEFINE
CLEAR        HOST           RUN            WHENEVER OSERROR
COLUMN       INPUT          SAVE           WHENEVER SQLERROR
COMPUTE      LIST           SET            XQUERY
CONNECT      PASSWORD        SHOW

SQL>

```

۵. می‌تواند به صورت مستقیم از طریق زبان‌های پروگرامنویسی مستقل به دیتابیس لینک شود؛ یعنی اجرای دستورها و بلاک‌های^۱ PL/SQL توسط پروگرامهایی که در زبان برنامه‌نویسی

دیگر نوشته شده باشند، فراخوانده (Call) شده و امکان به اجراء گذاشته شدن‌شان وجود دارد.

۶. نحوه عمومی نوشتمن دستورهای PL/SQL بر مبنای زبان‌های پروگرامنویسی^۲ ADA و پاسکال (Pascal) است. کسانی که به زبان‌های مذکور بلدیت داشته باشند، می‌توانند به سادگی دستورهای

PL/SQL را فرا گیرند.

۷. اساساً به خاطر استفاده در اوریکل دیزاین شده است؛ بر علاوه دیتابیس اوریکل، این زبان در سیستم مدیریت دیتابیس DB2 که مربوط کمپنی^۳ IBM است، نیز قابل استفاده است.

۲.۳ فواید PL/SQL

زبان PL/SQL به خاطر استفاده زیاد آن در دیتابیس‌های اوریکل دارای مزايا و فوایدي است. دیتابیس اوریکل يکی از جمله سیستم‌های مدیریت دیتابیس بوده که به صورت وسیع در ساحات علمی و مارکیت کار تجاری استفاده می‌شود. فواید استفاده از PL/SQL در ذیل لیست گردیده است:

^۱ بلاک‌های PL/SQL عبارت از ساختمان‌های قطعات کد این زبان‌اند. توضیحات بیشتر در مورد بلاک‌های کد PL/SQL در عنوانین بعدی همین فصل ارائه شده‌اند.

^۲ ADA یک زبان پروگرامنویسی است که خانمی به نام Ada Lovelace آن را ایجاد کرده است و نام زبان از نام ایجادکننده آن گرفته شده است.

^۳ اختصار کلمات International Business Machine IBM بوده و یکی از کمپنی‌های سابقه‌دار و در عین حال موفق در عرصه‌های تکنالوژی و

کمپیوتر ساینس است.

۱. ترکیب PL/SQL با سیکویل (SQL) یکی از مزایای این زبان به شمار می‌رود. در PL/SQL هر دو شکل دستورهای سیکویل ساکن و سیکویل دینامیک قابل استفاده است. با استفاده از سیکویل، دستورهای تعریف دیتا (DDL^۱) می‌تواند در بلاک‌های کد PL/SQL گنجانیده شود.

نوت: سیکویل ساکن (static SQL) عبارت از دستورهای ساکن می‌باشد که در زمان اجراء، پروگرام تغییر نمی‌کند و سیکویل دینامیک (Dynamic SQL) عبارت از دستورهای می‌باشد که در زمان اجراء، پروگرام تغییر می‌کند.

۲. فرستادن یک بلاک مکمل جملات کد به دیتابیس در یک زمان یکی دیگر از مزایای PL/SQL به شمار می‌رود.

این مسئله باعث کم شدن ترافیک دیتا در شبکه گردیده و اجرای اپلیکیشن‌ها را سرعت می‌بخشد. یک بلاک کد امکان دارد دهها و یا هم صدها دستور را در خود داشته باشد. انتقال دسته جمعی دستورهای یک بلاک می‌تواند تا صدها مرتبه سرعت کار سیستم را در صورت استفاده از PL/SQL بالا ببرد.

۳. توسط PL/SQL کیوری کردن دیتا، انتقال دادن دیتا و به روز کردن (Update) دیتا امکان‌پذیر است. امکان اجرای این کارها، به پروگرامها و اشخاص مسلکی در این عرصه، امکان انکشاف دادن برنامه‌های مؤثر و پر دست آور را مهیا می‌سازد.

۴. با استفاده از PL/SQL زمان کمی به خاطر دیزاین و پیدا کردن غلطی‌های کد ضرورت است. این مزیت به خاطر موجودیت صفات ذیل در این زبان است:

a. رسیدگی و درنظرگرفتن استثناهای (Exception Handling)

b. خلاصه‌سازی (Encapsulation)

c. پنهان‌سازی دیتا (Data Hiding)

d. انواع دیتای شیء‌گرا (Object Oriented Data Types)

۵. اپلیکیشن‌های نوشته شده در PL/SQL به صورت مکمل قابل انتقال (Portable) اند؛ به صورت نمونه می‌توان از نوشتن کد سیکویل در فایل منتهی با اکسنسنsql. یاد کرد. فایل‌های کد سیکویل به حیث فایل‌های مستقل در محیط‌های مختلف قابل انتقال و استفاده هستند.

۶. سطح بلند امنیت در PL/SQL در نظر گرفته شده و قابل تطبیق است.

۷. از طریق PL/SQL بسته‌های کد سیکویل که از قبل موجوداند، قابل دسترس و استفاده است.

^۱ اختصار کلمات Data Definition Language بوده و بخش دستورهای سیکویل را افاده می‌کند که با ایجاد کردن دیتابیس و ساختمان‌های داخلی آن ارتباط دارد.

۸. حمایت از برنامه‌نویسی شی‌گرا، که قبلاً در قسمت خصوصیات این زبان نیز به آن اشاره شد، از جمله مزیت‌های PL/SQL به شمار می‌رود.

۹. و در نهایت حمایت از انکشاف صفحات و اپلیکیشن‌های وب و صفحات سرورها توسط PL/SQL یکی دیگر از فواید این زبان به شمار می‌رود.

۲.۴ بلاک‌ها در PL/SQL (PL/SQL Blocks)

کد سیکویل در محیط PL/SQL به شکل بلاک‌هایی از دستورها نوشته و استفاده می‌شود؛ یعنی PL/SQL زبانی است که در آن قطعات دستورهای کد در بلاک‌های منطقی تنظیم می‌شوند. یک بلاک کد، به صورت بلاک داخلی، امکان دارد در بین بلاک بیرونی استفاده شود؛ یعنی بلاک‌های PL/SQL می‌توانند به ترتیب یکدیگر را به شکل Nested در برداشته باشند. هر بلاک کد PL/SQL متتشکل از سه قسمت است:

۱. اعلامیه بلاک (Declarations): این قسمت نشان‌دهنده آغاز بلاک بوده و با کلمه کلیدی DECLARE مشخص می‌شود. اعلامیه به صورت یک قسمت اختیاری به شمار رفته و در آن متحولین، کرسوها، پروگرام‌های فرعی و دیگر عناصر استفاده شده در پروگرام (بلاک) تعریف می‌شوند.

۲. دستورهای اجرایی (Executable Commands): اساسی‌ترین قسمت بلاک عبارت از قسمت دستورهای اجرایی است. این قسمت با کلمات آغاز (BEGIN) و انجام (END) احاطه می‌شود. استفاده از دستورهای آغاز و انجام در این قسمت حتمی است و در صورت عدم موجودیت یکی و یا هردوی آن‌ها، کد برنامه به راه نیفتاده و پیام خطأ صادر خواهد شد؛ در این قسمت، تمام دستورهای اجرایی پروگرام نوشته می‌شوند. برای داشتن یک پروگرام با نتیجه، در این قسمت باید حداقل یک دستور اجرایی موجود باشد؛ طور مثال، اضافه کردن یک دستور NULL می‌تواند در یک پروگرام استفاده شده و بلاک را قابل اجراء سازد.

۳. درنظرگرفتن استثناهای (Exception Handling): این قسمت نیز در یک بلاک کد اختیاری است. در صورت استفاده، با کلمه کلیدی EXCEPTION به بلاک اضافه می‌شود. هدف از به کارگیری این قسمت تنظیم شرایط غلطی‌های ممکنه در زمان اجرای بلاک است.

طوری که گفته شد، یک بلاک عبارت از ساختمان اساسی در یک پروگرام PL/SQL است. بلاک‌ها راهنمایی‌هایی را به خاطر اجراء در اوریکل، از قبیل نمایش‌دادن معلومات در روی صفحه کمپیوتر، نوشتندیتا به فایل‌ها، Call کردن پروگرام‌های دیگر، کار با ساختمان دیتا و غیره در بر دارد. تمام پروگرام

از بلاک‌ها ساخته شده است. یک پروگرام PL/SQL باید حداقل دارای یک بلاک باشد. بلاک‌های PL/SQL دستورهای کار با یوزر دیتا، یعنی^۱ DML و دستورهای دینامیک سیکویل، یعنی^۲ NDS را استفاده می‌کنند.

استفاده از حروف کوچک و بزرگ در کد PL/SQL مهم نیست. نوشتن کلمه کلیدی NULL معادل نوشتن این کلمه به شکل null و یا Null است. نکته قابل عطف این است که در آخر هر دستور PL/SQL علامه سیمی کولن، یعنی ":" باید استفاده شود. شکل عمومی یک بلاک طور ذیل است:

```
DECLARE
    <declarations section>
BEGIN
    <executable command(s)>
EXCEPTION
    <exception handling>
END;
```

به خاطر به اجراء گذاشتن یک بلاک PL/SQL، یک اسلش، یعنی / در آغاز لاین جدید گذاشته می‌شود. در پروگرام‌های عملی بلاک‌ها، «خط بر» در آخر بلاک استفاده می‌شود و کاربرد آن ضروری است؛ ولی جزوی از شکل عمومی دستور نیست.

کوتاه‌ترین ساختمان یک بلاک در PL/SQL عبارت از قسمت حتمی آن است که در آن دستورهای اجرایی در بین کلمات BEGIN و END نوشته می‌شود. به خاطر داشتن یک بلاک مکمل در پروگرام PL/SQL با حذف قسمت‌های اختیاری آن حداقل یک دستور اجرایی لازم است که در آن نوشته شده باشد. در مثال پایین بسیار مختصر، یک دستور اجرایی (که آن هیچ کاری را انجام نداده است؛ ولی از نشان‌دادن غلطی در پروگرام جلوگیری می‌کند) در یک بلاک گنجانیده شده که موفقانه قابل اجراست:

```
BEGIN
```

```
NULL;
```

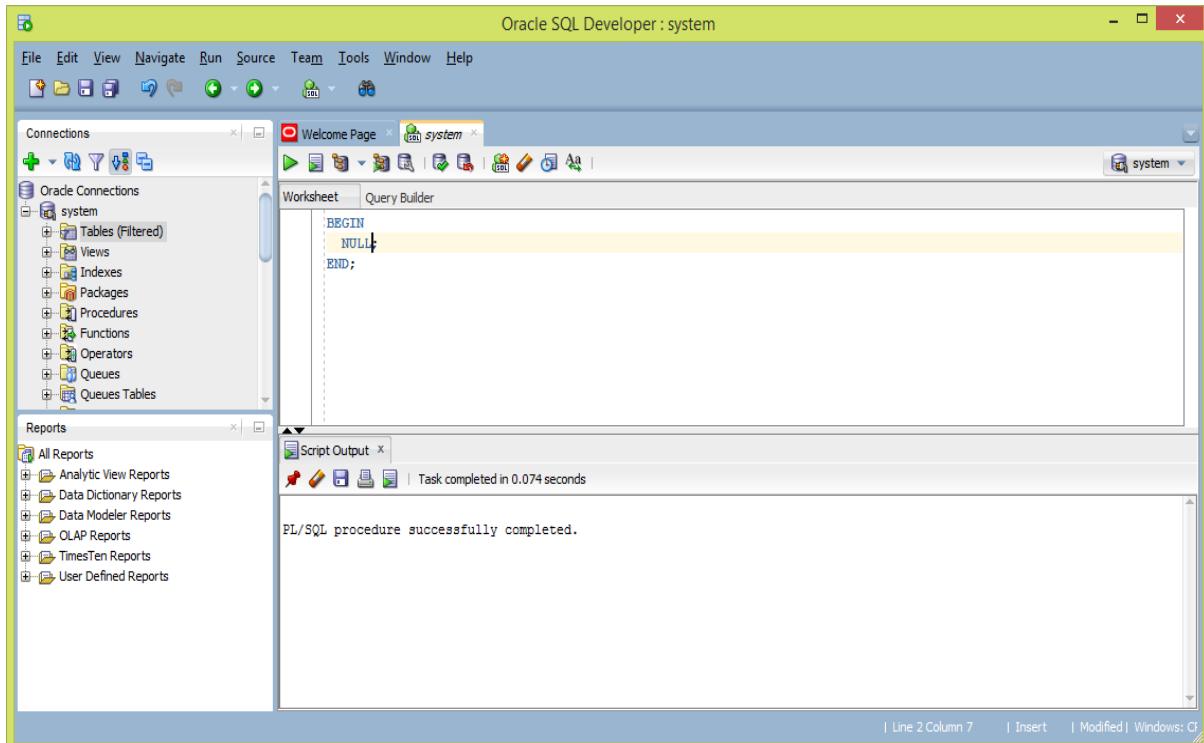
```
END;
```

```
/
```

^۱ اختصار کلمات Data Manipulation Language بوده و یک بخش از دستورهای سیکویل را در بر دارد که با یوزر دیتا (Userdata) کار می‌کند و دستورهایی از قبیل داخل کردن دیتا، به روز کردن دیتا و پاک کردن دیتا را شامل می‌شود.

^۲ اختصار کلمات Native Dynamic SQL بوده و طریقه‌یی است که جهت کار با دیتابیس دینامیک سیکویل استفاده می‌شود.

با به راه انداختن بلاک بالا، پروگرام اجراء شده و بدون این که کدام پیام خطا بدهد، با نتیجه خالی (طبق درخواست NULL) تطبیق می‌شود. در شکل پایین، نتیجه بلاک بالا که از طریق توسعه دهنده سیکویل اجرا شده است، نشان داده می‌شود.



The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations. On the left, there's a "Connections" sidebar with "Oracle Connections" expanded, showing "Tables (Filtered)", Views, Indexes, Packages, Procedures, Functions, Operators, Queues, and Queues Tables. Below it is a "Reports" sidebar with "All Reports", "Analytic View Reports", "Data Dictionary Reports", "Data Modeler Reports", "OLAP Reports", "TimesTen Reports", and "User Defined Reports". The main workspace has a "Worksheet" tab active, containing the following PL/SQL code:

```
BEGIN  
    NULL;  
END;
```

Below the worksheet is a "Script Output" window showing the message: "PL/SQL procedure successfully completed." The status bar at the bottom indicates "Line 2 Column 7" and "Modified".

شکل ۱-۲

اگر در کد بالا، دستور اجرایی حذف شود؛ یعنی کد به شکل پایین تغییر کند، نتیجه آن با غلطی همراه شده و به شکل ذیل نشان داده می‌شود:

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : system". The main window has a toolbar with various icons and a menu bar with "File", "Edit", "View", "Navigate", "Run", "Source", "Team", "Tools", "Window", and "Help". Below the menu is a toolbar with icons for running scripts, navigating, and viewing output. The central workspace is titled "Worksheet" and contains the following PL/SQL code:

```

BEGIN
END;
/

```

Below the workspace is a "Script Output" window showing the results of the execution:

```

Error starting at line : 1 in command -
BEGIN
END;
Error report -
ORA-06550: line 3, column 1:
PLS-00103: Encountered the symbol "END" when expecting one of the following:

( begin case declare exit for goto if loop mod null pragma
  raise return select update while with <an identifier>
<a double-quoted delimited-identifier> <a bind variable> <<
  continue close current delete fetch lock insert open rollback
  savepoint set sql execute commit forall merge pipe purge
06550. 00000 - "line %s, column %s:\n%s"

```

شکل ۲-۲

طوری که در مثال بالا دیده می‌شود، با دورکردن دستور اجرایی، پروگرام ناقص شده و سیستم مدیریت دیتابیس پیام غلطی را با تفصیل نشان می‌دهد؛ این در حالی است که قسمت دستورهای اجرایی با کلمات کلیدی آغاز و انجام تعریف شده است، اما از خود دستورها در آن ذکری به عمل نیامده است.

۲.۵ پیام‌های خروجی در PL/SQL (Output Messages in PL/SQL)

در PL/SQL پیام‌های خروجی از برنامه خواسته شده و در یک بلاک قابل پروگرام هستند. دستور اجرایی به خاطر ارائه پیام خروجی طور ذیل است:

```
>> DBMS_OUTPUT.PUT_LINE(TheOutputMessage);
```

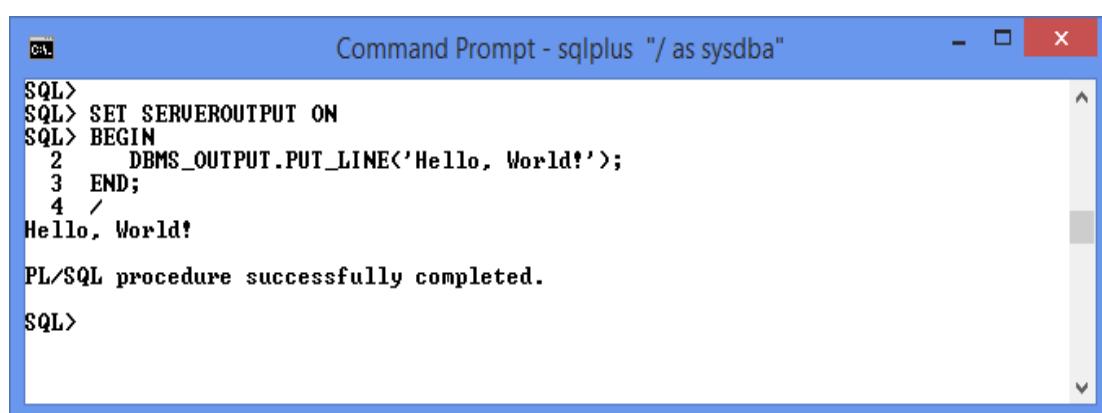
دستور پیام خروجی یکی از دستورهای داخلی PL/SQL با املای بالا است. استفاده از حروف بزرگ در دستور پیام خروجی مانند دستورهای دیگر این زبان ضروری نیست؛ یعنی دستور پیام خروجی در PL/SQL به اشکال ذیل نیز نوشته شده و قابل اجراء است:

```
>> dbms_output.put_line(...  
>> DBMS_Ouptup.Put_Line(...  
>> ...
```

مثال مشهور «سلام جهان» با استفاده از دستور پیام خروجی به دو شکل در ذیل ارائه شده است؛ در حالت اول متن مستقیماً به دستور اضافه شده و در نتیجه نشان داده می‌شود و در حالت دوم یک متتحول به نام message در یک دستور تعریف شده و متن به آن منصوب شده است. در دستور پیام خروجی، محتوای متتحول که عبارت از همان متن مورد نظر است، بر روی صفحه نشان داده می‌شود. حالت اول ساکن بوده و حالت دوم به شکلی است که پیام نمایشی بستگی به محتوای متتحول دارد.

مثال نمایش دادن پیام سلام جهان (حالت اول):

```
SET SERVEROUTPUT ON  
  
BEGIN  
  
DBMS_OUTPUT.PUT_LINE('Hello, World!');  
  
END;  
  
/
```



The screenshot shows a Windows Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The window contains the following PL/SQL code:

```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
 2   DBMS_OUTPUT.PUT_LINE('Hello, World!');
 3 END;
 4 /
Hello, World!
PL/SQL procedure successfully completed.
SQL>
```

The output of the code is "Hello, World!" displayed in the window.

مثال نمایش دادن پیام "سلام جهان" در حالت دوم با استفاده از متحول:

```
SET SERVEROUTPU ON  
  
DECLARE  
  
    Message VARCHAR2(20):= 'Hello, World!';  
  
BEGIN  
  
    DBMS_OUTPUT.PUT_LINE(Message);  
  
END;  
  
/
```

```
SQL> SET SERVEROUTPU ON
SQL> DECLARE
2    Message VARCHAR2(20):= 'Hello, World!';
3    BEGIN
4        DBMS_OUTPUT.PUT_LINE(Message);
5    END;
6/
Hello, World!
PL/SQL procedure successfully completed.
SQL> _
```

طوری که دیده شد در مثال بالا و در حالت اول، فقط از دستور نمایش پیام استفاده شد؛ در حالی که نتیجه هر دو حالت یکی بوده و به شکل ذیل است:

```
>> Hello, World!  
  
>> PL/SQL procedure successfully completed.
```

در حالت دوم مثال بالا، امکانات انعطاف‌پذیری بیشتری در بلاک موجود است. اگر متحول تعریف شده در قسمت اختیاری اعلامیه تغییر داده شود، دستور نمایش پیام، نتیجه تغییر کرده را نشان خواهد داد. یک متحول می‌تواند معلومات را از منابع مختلف بگیرد؛ طور مثال، در جدولی که لیست استفاده‌کنندگان سیستم را داشته باشد، متحولی تعریف شود که در زمان استفاده، نام استفاده‌کننده را از جدول خوانده و در پیام خروجی نشان بدهد و یا پارامتری استفاده شود که در آن، نام از استفاده‌کننده خواسته شود، بعد از داصل کردن نام، پیام خروجی با معلومات گرفته شده ضمیمه و به استفاده‌کننده نشان داده شود.



خلاصه فصل دوم

این فصل با معلومات مقدماتی در قسمت زبان پروسیجری PL/SQL آغاز شد. خصوصیات زبان PL/SQL و موارد استفاده از این زبان در عناوین مستقل تشریح شدند. در قسمت موارد استفاده به موضوعات مهمی اشاره شد که اهمیت استفاده از PL/SQL را برجسته می‌ساخت. فواید استفاده از PL/SQL با وضاحت بیشتر در قسمت ترکیب این زبان با سیکویل و زبان‌های شی‌گرا توضیح داده شده است.

یکی دیگر از فواید استفاده از PL/SQL، به کارگیری بلاک‌ها به خاطر مدیریت و تنظیم کد سیکویل بوده و در عنوان فواید این زبان، به آن اشاره شده است؛ همین ساختمان‌ها، یعنی بلاک‌های کد PL/SQL در عنوان مستقل به تفصیل توضیح داده شده و مثال‌های عملی آن نشان داده شد. قسمت آخر فصل به بحث روی پیام‌های خروجی PL/SQL پرداخته و با ذکر مثال‌های عملی، فصل دوم به پایان رسیده است.



سوالات و فعالیت های فصل دوم

۱. مفهوم جمله «زبان PL/SQL با سیکویل به صورت وسیع ترکیب شده است.» به زبان ساده‌تر واضح سازید.
۲. کدام نوع از ساختمان‌های برنامه‌نویسی در زبان PL/SQL موجوداند؟ نام ببرید.
۳. نحوه نوشتن کد زبان PL/SQL به اساس کدام زبان‌های برنامه‌نویسی انکشاف داده شده است؟
۴. ساختمان بلاک کد PL/SQL از کدام قسمت‌ها تشکیل شده است؟
۵. قسمت‌های اختیاری در بلاک کد PL/SQL کدام‌ها هستند؟
۶. نتیجه بلاک ذیل چه خواهد بود:

BEGIN

NULL;

END;

/

۷. شکل عمومی دستور پیام خروجی را بنویسید.

فعالیت ها

۱. یک بلاک PL/SQL بنویسید که توسط آن متن I AM HERE TO LEARN. روی صفحه نشان داده شود.
۲. یک بلاک PL/SQL بنویسید که با استفاده از متحولی به نام text متن I AM HERE TO LEARN. روی صفحه نشان داده شود.

فصل سوم

متتحولهای PL/SQL (PL/SQL Variables)



هدف کلی: شاگردان با تعریف متتحولها در PL/SQL آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. شناسه‌های مجاز و غیر مجاز در PL/SQL را بشناسند.
۲. کلمه‌های ریزرفشده و کلیدی PL/SQL را بفهمند.
۳. متتحولها را تعریف و استفاده کنند.
۴. موارد و طرز استفاده از متتحولین را بدانند.
۵. انواع دیتای ترکیبی و سکالری را بیاموزند.
۶. فواید استفاده از %Type Attribute را برای تعریف کردن متتحولها توضیح دهند.
۷. متتحولهای بهم‌بسته (Bind Variables) را تعریف و به صورت عملی استفاده کنند.

استفاده از متتحولها (Variables) در زبان‌های پروگرامنویسی امکان انجام دادن کارهای مؤثر را برای پروگرامها مساعد می‌سازد. در زبان PL/SQL متتحولها به اهداف خاص استفاده می‌شوند. توسط متتحولها، عملیه‌های ریاضی، احصائیوی و غیره در پروگرامها انجام داده می‌شوند. متتحولها به خاطر تنظیم کردن نمایش متن و ترکیب کردن متون در پروگرامهای PL/SQL به کار برده می‌شوند. دهها مثال از استفاده مؤثر متتحولها در پروگرامها قابل ذکر است. هر متتحول توسط کلمات شناسایی می‌شوند؛ یعنی یک متتحول در PL/SQL باید یک نام داشته باشد. قواعد نام‌گذاری مجاز و غیر مجاز یکی از نکات قابل بحث محسوب شده و در این فصل به آن پرداخته شده است.

تعریف کردن متتحولها، موارد و طرز کار آن‌ها در زبان PL/SQL از مسائل مهم دیگر به شمار می‌آید. انواع دیتای ترکیبی (Composite Datatypes) و سکالری (Scalar Datatypes) ضرورت به توضیحات داشته و در جریان عناوین، به بحث گرفته شده‌اند. یک نوع مشخصه (Attribute) با استفاده از سimbol %Type به خاطر بهارت‌گرفتن نوعی دیتا برای متتحولها در PL/SQL و نیز فواید استفاده از آن با ذکر یکی دو مثال، توضیح داده شده است. متتحولهای بهم‌بسته که اصطلاح انگلیسی آن عبارت از Bind Variables است، به صورت مفصل و با ذکر مثال‌هایی در این فصل بیان شده‌اند.

۳.۱ شناسه‌های زبان PL/SQL Identifiers

شناسه‌ها یا Identifiers در زبان PL/SQL به خاطر نام‌گذاری اجزاء و بخش‌های پروگرامها به کار گرفته می‌شوند. مثال‌های بخش‌های کد PL/SQL عبارت از ثابت‌ها (Constants)، متتحولها (Variables)، استثناهای (Exceptions)، کرسرهای متتحول (Cursors)، برنامه‌های فرعی (Cursor Variables)، پروسیجرها (Procedures)، کلمه‌های ریزرفشده (Reserved Words)، کلمه‌های کلیدی (Keywords) و پکیج‌ها (Packages) است که توسط شناسه‌ها نام‌گذاری می‌شوند. هر کدام از انواعی که در جمله قبلی نام گرفته شد، در یک برنامه PL/SQL باید با یک نام به خاطر شناسایی مسمی شده باشد. نام‌گذاری در PL/SQL دارای قواعد خاص بوده و محدودیت‌های نوشتاری نیز باید در آن‌ها مراعات شود.

یک شناسه در PL/SQL حداقل با یک حرف نشان داده می‌شود. طول اعظمی یک شناسه می‌تواند تا 30 کرکتر باشد.

۳.۲ شناسه‌های مجاز و غیر مجاز (Valid and Invalid Identifiers)

سمبل‌ها و کرکترهای مجاز در نام‌گذاری شناسه‌ها عبارت از حروف، نمبرها، علامه دالر (\$)، علامه نمبر (#) و خط زیرین (Underscore) است. در ذیل، بعضی مثال‌ها از شناسه‌های مجاز (Valid Identifiers) لیست شده‌اند:

1. X
 2. t2
 3. phone#
 4. credit_limit
 5. LastName
 6. oracle\$number
 7. ...

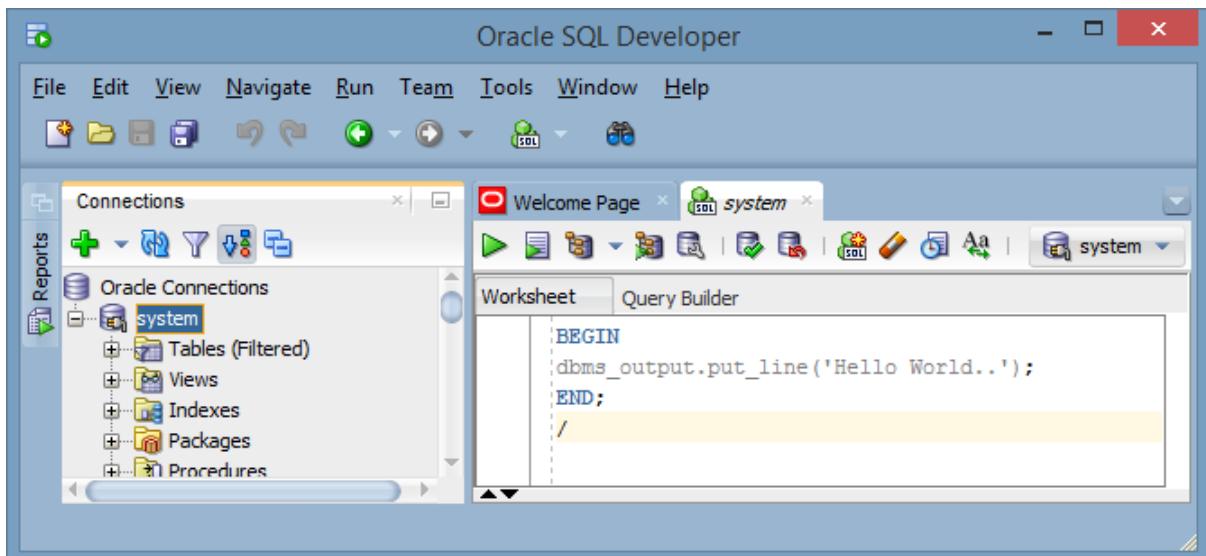
در نام‌گذاری شناسه‌ها یکی از شرایط اساسی، شروع نام شناسه به یکی از حروف الفبای انگلیسی است. سمبول‌هایی که استفاده از آن‌ها در نام شناسه‌های PL/SQL غیر مجاز است، عبارت از خطهای دش (hyphens)، اسلش‌ها (Slashes) و خانه‌خالی‌ها (Spaces) استند.

د. پایه‌نامه‌ها، نمونه‌های شناسه‌های غیر مجاز (Invalid Identifiers) نشان دهنده اند:

استفاده از حروف کوچک و بزرگ در نوشتن شناسه‌های PL/SQL کدام تفاوت ندارد؛ یعنی شناسه‌های LASTNAME و LastName همه معادل‌اند و کلمه‌های ریزرفی شناسه‌های، که در بالا ذکر شد، استفاده نمی‌شوند.

۳.۲.۱ کلمه‌های نزدیک شده (Reserved Words) PL/SQL

بعضی از شناسه‌ها به نام کلمه‌های ریزرفشده PL/SQL یاد می‌شوند. این دسته از کلمه‌ها معانی خاصی را در این زبان افاده کرده و هر کدام آن‌ها به هدف مشخصی به کار گرفته می‌شود. اگر یک شناسه، طور مثال نام یک متتحول، با استفاده از کلمه ریزرفشده تعریف می‌شود، در زمان اجراء، سیستم مدیریت دیتابیس، آن را نادرست گرفته و پیام خطای می‌دهد. مثال‌های کلمه‌های ریزرفی عبارت از BEGIN و END در یک بلاک کد PL/SQL است. کلمه‌های ریزرفی معمولاً به حروف بزرگ نوشته شده و نشان داده می‌شوند. استفاده از حروف بزرگ در نام کلمه‌های ریزرفی حتمی نیست. وسیله‌هایی که در آن‌ها کد زبان PL/SQL را اکشاف داده می‌شود، نیز کلمه‌های ریزرفی را شناسایی کرده و به رنگ مشخص نشان می‌دهد. در شکل یابیز، قطعه کد برنامه PL/SQL به طور نمونه نشان داده شده است.



شکل ۱-۳

طوری که در شکل بالا دیده می‌شود، کلمه‌های ریزرفشده به رنگ آبی نشان داده شده است. دستورهای اجرایی به رنگ سیاه و متن داخل قوس ناخنک نیز به رنگ خاکی دیده می‌شود.

در صورت ضرورت استفاده از یک کلمه ریزرفشده در کد PL/SQL، طریقه‌هایی وجود دارد. یکی از این طریقه‌ها اضافه کردن کلمات و یا سمبل‌های مجاز نام‌گذاری با کلمه ریزرفشده است. در مثال پایین، کلمه ریزرفی END که در ختم دستورهای اجرایی در بلاک PL/SQL استفاده می‌شود، با رشته‌یی از حروف و سمبل‌های مجاز ترکیب شده و مشکل پروگرام توسط آن حل گردیده است.

حالت اول:

DECLARE

end BOOLEAN; (استفاده از کلمه ریزرفی مجاز نیست)

حالت دوم:

DECLARE

end_of_game BOOLEAN; (در این حالت مجاز است و غلطی رخ نمی‌دهد)

در زمان ترکیب کلمه ریزرفی با کلمات دیگر، مسئله محدودیت طول مجاز نام شناسه در نظر گرفته شود که آن عبارت از 30 کرکتر است. در مثال بالا مشکلی ایجاد نمی‌شود؛ چون نام ترکیبی استفاده شده، 11 کرکتر است.

موضوع دیگر در رابطه با کلمه‌های ریزرفی PL/SQL عبارت از کلمه‌های کلیدی (Keywords) در این زبان است. کلمات کلیدی در PL/SQL مانند کلمه‌های ریزرفی دارای معانی خاص خود می‌باشند. بر عکس

کلمه‌های ریزرفی، کلمات کلیدی به حیث نام شناسه‌ها در PL/SQL قابل استفاده‌اند؛ ولی این کار توصیه نمی‌شود. به خاطر وضاحت بیشتر موضوع، لیست‌هایی از کلمات ریزرفی PL/SQL در شکل پایین نشان داده شده است.

Command Prompt - sqlplus "/as sysdba"				
SQL> ? reserved words				
RESERVED WORDS <PL/SQL>				
PL/SQL Reserved Words have special meaning in PL/SQL, and may not be used for identifier names (unless enclosed in "quotes").				
An asterisk (*) indicates words are also SQL Reserved Words.				
ALL*	DESC*	JAVA	PACKAGE	SUBTYPE
ALTER*	DISTINCT*	LEVEL*	PARTITION	SUCCESSFUL*
AND*	DO	LIKE*	PCTFREE*	SUM
ANY*	DROP*	LIMITED	PLS_INTEGER	SYNONYM*
ARRAY	ELSE*	LOCK*	POSITIVE	SYSDATE*
AS*	ELSIF	LONG*	POSITIONEN	TABLE*
ASC*	END	LOOP	PRAGMA	THEN*
AT	EXCEPTION	MAX	PRIOR*	TIME
AUTHID	EXCLUSIVE*	MIN	PRIVATE	TIMESTAMP
AVG	EXECUTE	MINUS*	PROCEDURE	TIMEZONE_ABBR
BEGIN	EXISTS*	MINUTE	PUBLIC*	TIMEZONE_HOUR
BETWEEN*	EXIT	MLSLABEL*	RAISE	TIMEZONE_MINUTE
BINARY_INTEGER	EXTENDS	MOD	RANGE	TIMEZONE_REGION
BODY	EXTRACT	MODE*	RAW*	TO*
BOOLEAN	FALSE	MONTH	REAL	TRIGGER*
BULK	FETCH	NATURAL	RECORD	TRUE
BY*	FLOAT*	NATURALN	REF	TYPE
CHAR*	FOR*	NEW	RELEASE	UI
CHAR_BASE	FORALL	NEXTVAL	RETURN	UNION*
CHECK*	FROM*	NOCOPY	REVERSE	UNIQUE*
CLOSE	FUNCTION	NOT*	ROLLBACK	UPDATE*
CLUSTER*	GOTO	NOWAIT*	ROW*	USE
COALESCE	GROUP*	NULL*	ROWID*	USER*
COLLECT	HAVING*	NULLIF	ROWNUM*	VALIDATE*
COMMENT*	HEAP	NUMBER*	ROWTYPE	VALUES*
COMMIT	HOUR	NUMBER_BASE	SAVEPOINT	VARCHAR*
COMPRESS*	IF	OCIROWID	SECOND	VARCHAR2*
CONNECT*	IMMEDIATE*	OF*	SELECT*	VARIANCE
CONSTANT	IN*	ON*	SEPERATE	VIEW*
CREATE*	INDEX*	OPAQUE	SET*	WHEN
CURRENT*	INDICATOR	OPEN	SHARE*	WHENEVER*
CURRVAL	INSERT*	OPERATOR	SMALLINT*	WHERE*
CURSOR	INTEGER*	OPTION*	SPACE	WHILE
DATE*	INTERFACE	OR*	SQL	WITH*
DAY	INTERSECT*	ORDER*	SQLCODE	WORK
DECIMAL*	INTERVAL	ORGANIZATION	SQLERRM	WRITE
DECLARE	INTO*	OTHERS	START*	YEAR
DEFAULT*	IS*	OUT	STDDEV	ZONE
DELETE*	ISOLATION			

شکل ۲-۳

بعضی از کلمات ریزرفی PL/SQL با سیکویل (SQL) مشترک‌اند. در شکل بالا، کلمات مشترک با علامه ستاره نشانی شده‌اند. به خاطر ارتباط موضوع، کلمات ریزرفی سیکویل نیز در شکل پایین آورده شده‌اند.

RESERVED WORDS <SQL>

SQL Reserved Words have special meaning in SQL, and may not be used for identifier names unless enclosed in "quotes".

An asterisk (*) indicates words are also ANSI Reserved Words.

Oracle prefixes implicitly generated schema object and subobject names with "SYS_". To avoid name resolution conflict, Oracle discourages you from prefixing your schema object and subobject names with "SYS_".

ACCESS	DEFAULT*	INTEGER*	ONLINE	START
ADD*	DELETE*	INTERSECT*	OPTION*	SUCCESSFUL
ALL*	DESC*	INTO*	OR*	SYNONYM
ALTER*	DISTINCT*	IS*	ORDER*	SYSDATE
AND*	DROP*	LEVEL*	PCTFREE	TABLE*
ANY*	ELSE*	LIKE*	PRIOR*	THEN*
AS*	EXCLUSIVE	LOCK	PRIVILEGES*	TO*
ASC*	EXISTS	LONG	PUBLIC*	TRIGGER
AUDIT	FILE	MAXEXTENTS	RAW	UID
BETWEEN*	FLOAT*	MINUS	RENAME	UNION*
BY*	FOR*	MLSLABEL	RESOURCE	UNIQUE*
CHAR*	FROM*	MODE	REVOKE*	UPDATE*
CHECK*	GRANT*	MODIFY	ROW	USER*
CLUSTER	GROUP*	NOAUDIT	ROWID	VALIDATE
COLUMN	HAVING*	NOCOMPRESS	ROWNUM	VALUES*
COMMENT	IDENTIFIED	NOWAIT	SELECT*	VARCHAR*
COMPRESS	IMMEDIATE*	NULL*	SESSION*	VARCHAR2
CONNECT*	IN*	NUMBER	SET*	VIEW*
CREATE*	INCREMENT	OF*	SHARE	WHENEVER*
CURRENT*	INDEX	OFFLINE	SIZE*	WHERE
DATE*	INITIAL	ON*	SMALLINT*	WITH*
DECIMAL*	INSERT*			

SQL>

شکل ۳-۳

در لیست کلمات ریزرفی سیکویل بعضی از کلمات موجوداند که در انتیتوت استندردسازی ملی امریکا (ANSI) نیز به حیث کلمات ریزرفی قبول شده و استفاده می‌شوند. ستاره‌های ضمیمه با نام کلمات در شکل بالا دسته را مشخص می‌کند.

کلمات کلیدی PL/SQL در جدول پایین به صورت جداگانه ارائه شده‌اند.

جدول . ۳ . لیست کلمات کلیدی زبان PL/SQL

شماره	حرف آغازین	کلمه‌های کلیدی زبان PL/SQL
1	A	A, ADD, AGENT, AGGREGATE, ARRAY, ATTRIBUTE, AUTHID, AVG
2	B	BFILE_BASE, BINARY, BLOB_BASE, BLOCK, BODY, BOTH, BOUND, BULK, BYTE
3	C	C, CALL, CALLING, CASCADE, CHAR, CHAR_BASE, CHARACTER, CHARSETFORM, CHARSETID, CHARSET, CLOB_BASE, CLOSE, COLLECT, COMMENT, COMMIT, COMMITTED, COMPILED, CONSTANT, CONSTRUCTOR, CONTEXT, CONVERT, COUNT, CURSOR, CUSTOMDATUM
4	D	DANGLING, DATA, DATE, DATE_BASE, DAY, DEFINE, DETERMINISTIC, DOUBLE, DURATION
5	E	ELEMENT, ELSIF, EMPTY, ESCAPE, EXCEPT, EXCEPTIONS, EXECUTE, EXIT, EXTERNAL
6	F	FINAL, FIXED, FLOAT, FORALL, FORCE, FUNCTION
7	G	GENERAL
8	H	HASH, HEAP, HIDDEN, HOUR
9	I	IMMEDIATE, INCLUDING, INDICATOR, INDICES, INFINITE, INSTANTIABLE, INT, INTERFACE, INTERVAL, INVALIDATE, ISOLATION
10	J	JAVA
11	L	LANGUAGE, LARGE, LEADING, LENGTH, LEVEL, LIBRARY, LIKE2, LIKE4, LIKEC, LIMIT, LIMITED, LOCAL, LONG, LOOP
12	M	MAP, MAX, MAXLEN, MEMBER, MERGE, MIN, MINUTE, MOD, MODIFY, MONTH, MULTISET
13	N	NAME, NAN, NATIONAL, NATIVE, NCHAR, NEW, NOCOPY, NUMBER_BASE
14	O	OBJECT, OCICOLL, OCIDATETIME, OCIDATE, OCIDURATION, OCIINTERVAL, OCILOBLOCATOR,

		OCINUMBER, OCIRAW, OCIREFCURSOR, OCIREF, OCIROWID, OCISTRING, OCITYPE, ONLY, OPAQUE, OPEN, OPERATOR, ORACLE, ORADATA, ORGANIZATION, ORLANY, ORLVARY, OTHERS, OUT, OVERRIDING
15	P	PACKAGE, PARALLEL_ENABLE, PARAMETER, PARAMETERS, PARTITION, PASCAL, PIPE, PIPELINED, PRAGMA, PRECISION, PRIVATE
16	R	RAISE, RANGE, RAW, READ, RECORD, REF, REFERENCE, REM, REMAINDER, RENAME, RESULT, RETURN, RETURNING, REVERSE, ROLLBACK, ROW
17	S	SAMPLE, SAVE, SAVEPOINT, SB1, SB2, SB4, SECOND, SEGMENT, SELF, SEPARATE, SEQUENCE, SERIALIZABLE, SET, SHORT, SIZE_T, SOME, SPARSE, SQLCODE, SQLDATA, SQLNAME, SQLSTATE, STANDARD, STATIC, STDDEV, STORED, STRING, STRUCT, STYLE, SUBMULTISET, SUBPARTITION, SUBSTITUTABLE, SUBTYPE, SUM, SYNONYM
18	T	TDO, THE, TIME, TIMESTAMP, TIMEZONE_ABBR, TIMEZONE_HOUR, TIMEZONE_MINUTE, TIMEZONE_REGION, TRAILING, TRANSAC, TRANSACTIONAL, TRUSTED, TYPE
19	U	UB1, UB2, UB4, UNDER, UNSIGNED, UNTRUSTED, USE, USING
20	V	VALIST, VALUE, VARIABLE, VARIANCE, VARRAY, VARYING, VOID
21	W	WHILE, WORK, WRAPPED, WRITE
22	Y	YEAR
23	Z	ZONE

۳.۳ متغیرها در زبان PL/SQL (PL/SQL Variables)

در زبان PL/SQL مانند زبان‌های دیگر پروگرامنویسی، متغیرها تعریف و به کار می‌روند. یک متغیر در کد PL/SQL به یک نام مشخص با در نظرداشت قواعد شناسه‌ها که قبلاً توضیح داده شد، تعریف می‌شود.

بعد از تعریف کردن یک متتحول در PL/SQL یک قیمت به آن داده می‌شود. متتحول‌ها بعد از گرفتن قیمت در کد برنامه به کار گرفته می‌شوند. قیمت‌های متتحول‌ها امکان دارد که به طور ثابت توسط پروگرامر به آن‌ها داده شود و یا هم متتحول‌ها می‌توانند قیمت‌های شان را به صورت دینامیک از یک منبع به دست آورند.

تعریف کردن متتحول‌ها

یک متتحول زمانی که در PL/SQL تعریف می‌شود، باید به آن یک نوع دیتا نیز تعیین شود. انواع دیتای قابل تعیین‌شدن به متتحول‌ها، به انواع دیتای سیکویل بر می‌گردد؛ مانند:

1. CHAR
2. DATE
3. NUMBER
4. ...

و یا هم انواع دیتا PL/SQL مانند:

1. BOOLEAN
2. PLS_INTEGER
3. ...

برای مثال، متتحول‌هایی تعریف شوند که یکی از آن‌ها دیتای عددی شش رقمی را برای قسمت‌های یک محصول نشان دهد. متتحول دیگر دیتای معلومات بولی را از نوع صحیح/غلط برای موجودیت محصول در ذخیره‌خانه نشان دهد و به عین شکل چند متتحول دیگر نیز در این مثال، به شکل ذیل تعریف می‌گردند:

DECLARE

```
part_no NUMBER(6);  
part_name VARCHAR2(20);  
in_stock BOOLEAN;  
part_price BOOLEAN;  
part_desc VARCHAR2(50);
```

همین قسمت اعلامیه بلاک کد PL/SQL که در آن صرف متتحول‌ها تعریف شده‌اند و قیمت به هیچ کدام آن‌ها داده نشده است؛ صرف به خاطر آموزش در صفحه کد توسعه‌دهنده سیکویل گذاشته شده است. شکل پایین تعریف متتحول‌ها مطابق کد بالا را به شکل رنگ‌آمیزی شده آن‌ها نشان می‌دهد.

```
DECLARE
    part_no NUMBER(6);
    part_name VARCHAR2(20);
    in_stock BOOLEAN;
    part_price BOOLEAN;
    part_desc VARCHAR2(50);
```

شکل ۴-۳

طوری که در شکل بالا دیده می‌شود، کلمات ریزرفشده با رنگ آبی و نام متتحولها به رنگ سیاه نشان داده شده است. رنگ‌آمیزی کد توسط کمپیوتر، به استفاده‌کننده کمک می‌کند تا در قسمت استفاده از شناسه‌های غیر مجاز توجه کرده و اگر احیاناً کدام اشتباه رخ دهد، در قسمت حل کردن آن به سهولت اقدام کند.

۳.۴ موارد و طرز استفاده از متتحولها

زمانی که متتحولها با استفاده از دستور DECLARE در یک پروگرام PL/SQL تعریف شدند، به خاطر استفاده‌کردن از آن‌ها، به متتحولها قیمت داده می‌شود. جهت تعیین کردن قیمت‌ها به متتحولها در برنامه‌های PL/SQL از سه طریقه استفاده می‌شود. طریقه‌های تعیین کردن قیمت‌ها به متتحولها با مثال‌های لازم تحت عناوین فرعی در ذیل ارائه می‌شوند:

طریقه اول دادن قیمت به متتحولها

در این طریقه سمبول =: به تعقیب نام متتحول و اضافه کردن یک قیمت به آن استفاده می‌شود. قیمت متتحول باید نظر به نوع دیتا در فارمت مشخص شده باشد. برای متتحول عددی باید قیمت عددی و برای متتحول نوع کرکتر، دیتا باید از همان فارمت باشد. شکل پایین، یک مثال را با تعریف متتحولها به طریقه اول دادن قیمت‌ها نشان می‌دهد.

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Print, along with a "SQL" icon. The left sidebar has sections for Reports and Connections, with "Welcome Page" and "system" selected. The main workspace is titled "Worksheet" and contains a "Query Builder" tab. The code area displays the following PL/SQL block:

```
DECLARE
    wages NUMBER;
    hours_worked NUMBER := 40;
    hourly_salary NUMBER := 22.50;
    bonus NUMBER := 150;
    country VARCHAR2(128);
    counter NUMBER := 0;
    done BOOLEAN;
    valid_id BOOLEAN;
    emp_rec1 employees%ROWTYPE;
    emp_rec2 employees%ROWTYPE;
    TYPE commissions IS TABLE OF NUMBER INDEX BY PLS_INTEGER;
    comm_tab commissions;
BEGIN
    wages := (hours_worked * hourly_salary) + bonus;
    country := 'France';
    country := UPPER('Canada');
    done := (counter > 100);
    valid_id := TRUE;
    emp_rec1.first_name := 'Antonio';
    emp_rec1.last_name := 'Ortiz';
    emp_rec1 := emp_rec2;
    comm_tab(5) := 20000 * 0.15;
END;
/
```

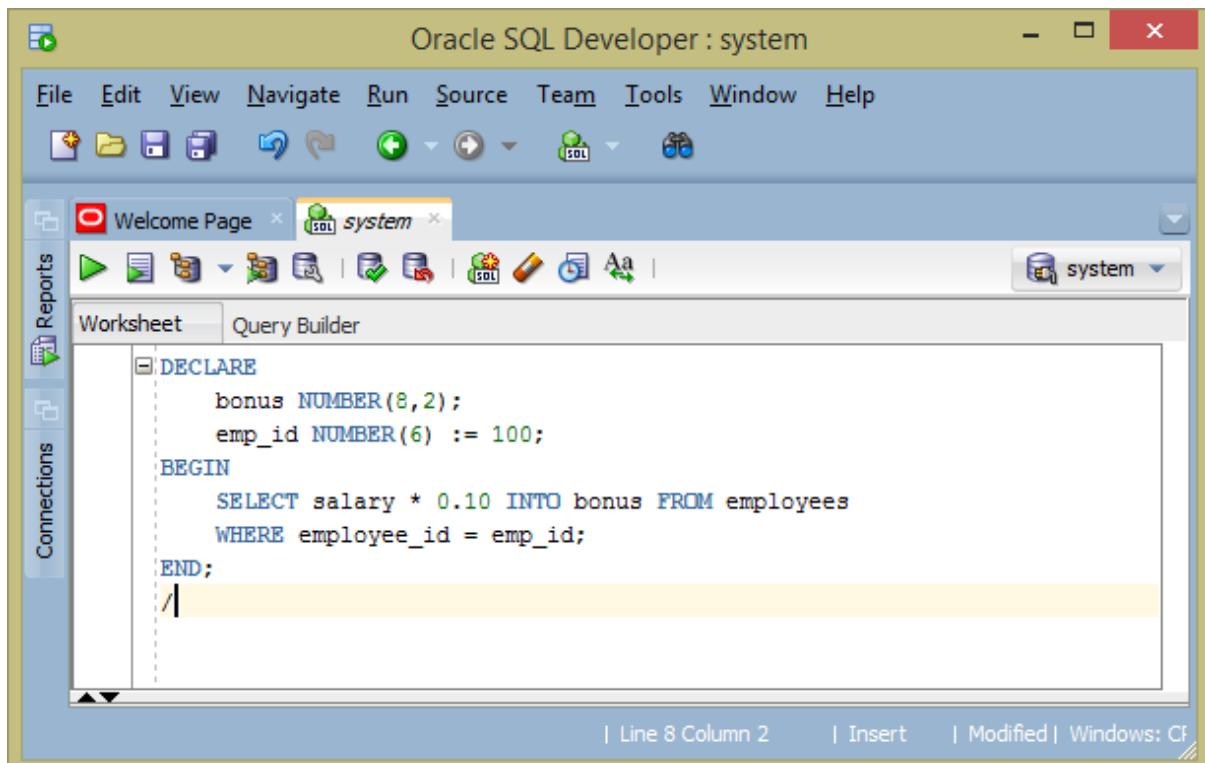
At the bottom of the code editor, status bars show "Line 25 Column 2", "Insert", "Modified", and "Windows: C".

شکل ۵-۳

با آشنایی با فارمت نوشتن دستورهای کد PL/SQL، محصلین می‌توانند سطرهای کد شکل بالا را تفسیر کرده و مفاهیم لازم را از آن‌ها بیاموزند؛ به طور نمونه، اگر سطر سوم دیده شود، در آن متحولی به نام hours_worked از نوع عددی تعریف شده است. در همان سطر به متحول یادشده، قیمت 40 که یک نمبر است، داده شده است. در قسمت دستورهای اجرایی، اولین سطر بعد از کلمه BEGIN، متحول مذکور استفاده شده است. این سطر قیمت‌های سه متحول را می‌گیرد و آن‌ها را مطابق عملیه‌های ریاضی ضرب و جمع و محاسبه کرده، به دستمزد (Wages) علاوه می‌کند.

طريقه دوم دادن قيمت به متتحولها با استفاده از INTO

در طريقه دوم دادن قيمت به متتحولها، از منبع بيرونی جهت گرفتن ديتا استفاده می شود، يعني پروگرام به طور مستقيم قيمتها را به متتحولها نمی نويسد؛ طور مثال، يك ديتابيس منحيث منبع ديتا در نظر گرفته شده و قيمتهاي متتحولها از آن استخراج می شود. قيمت گرفته شده از منبع بيرونی شكل اجرائي متتحول را در پروگرام نوشته شده PL/SQL بازي كرده و نتایج مورد نظر، نشان داده می شود. در مثالی که در شكل پايين کد PL/SQL نشان داده شده است، معاش يك كارمند 10 فيصد تزايد پيدا می کند.



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The left sidebar shows "Welcome Page" and "system" connection. The main workspace is titled "Worksheet" and contains the following PL/SQL code:

```
DECLARE
    bonus NUMBER(8,2);
    emp_id NUMBER(6) := 100;
BEGIN
    SELECT salary * 0.10 INTO bonus FROM employees
    WHERE employee_id = emp_id;
END;
/
```

The status bar at the bottom indicates "Line 8 Column 2" and "Modified".

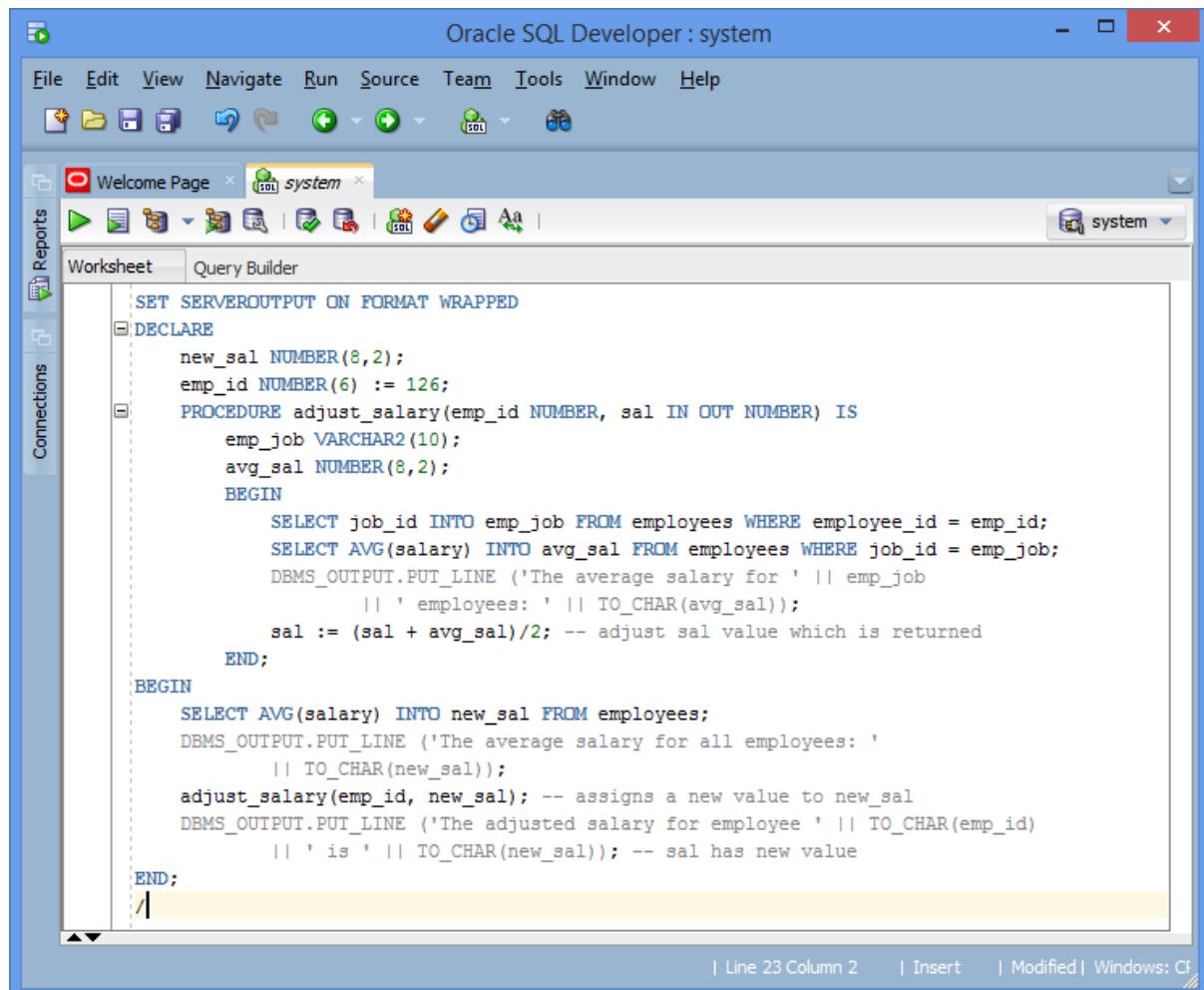
شکل ۶-۳

طوری که در کد شکل بالا دیده می شود، يك متتحول به نام bonus از نوع عددی تعریف شده است و متتحول دومی به نام emp_id نیز از همین نوع تعریف شده است. متتحول دومی قيمت 100 را با استفاده از طریق اول که قبلًا شرح داده شد، گرفته است.

متتحول اول به طریقه جدیدی که در این درس تحت نام طریقه دوم دادن قيمت به متتحولها توضیح شد، به کمک کلمه INTO از دستور اجرائي قيمت می گیرد. در دستور اجرائي بعد از کلمه BEGIN يك کیوری به راه انداخته شده که توسط آن، محتواي فيلد salary برای کارمند شماره 100 از جدول employee گرفته شده است. به محتواي گرفته شده از ستون معاش، ده فيصد علاوه شده و با استفاده از کلمه Rizerfi INTO مقدار تزايد ده فيصدی به متتحول bonus قيمت داده می شود.

طريقه سوم دادن قيمت به متاحولها با استفاده از پروگرام فرعی

طريقه سوم دادن قيمت‌ها به متاحولها، كمی پيچيده است. در اين طريقه، يك قيمت به متاحول به كمک دستور OUT و يا هم دستورهای IN OUT پارامترهای ديتا به يك پروگرام فرعی (Subprogram) انتقال داده شده و بعد از آن قيمت مورد نظر از طريق دستورهای پروگرام فرعی به متاحول داده می‌شود. شكل پايين، كد مكمل اين طريقه را در بر دارد. جهت وضاحت بيشتر موضوع، توضيحات لازم در قسمت پايين تصوير ارائه شده‌اند.



The screenshot shows the Oracle SQL Developer interface with a PL/SQL code editor. The code defines a procedure 'adjust_salary' that takes an employee ID and salary as input parameters. It calculates the average salary for the employee's job and adds it to the current salary. The code uses DBMS_OUTPUT.PUT_LINE to print messages to the console. The code is as follows:

```
SET SERVEROUTPUT ON FORMAT WRAPPED
DECLARE
    new_sal NUMBER(8,2);
    emp_id NUMBER(6) := 126;
    PROCEDURE adjust_salary(emp_id NUMBER, sal IN OUT NUMBER) IS
        emp_job VARCHAR2(10);
        avg_sal NUMBER(8,2);
    BEGIN
        SELECT job_id INTO emp_job FROM employees WHERE employee_id = emp_id;
        SELECT AVG(salary) INTO avg_sal FROM employees WHERE job_id = emp_job;
        DBMS_OUTPUT.PUT_LINE ('The average salary for ' || emp_job
                             || ' employees: ' || TO_CHAR(avg_sal));
        sal := (sal + avg_sal)/2; -- adjust sal value which is returned
    END;
BEGIN
    SELECT AVG(salary) INTO new_sal FROM employees;
    DBMS_OUTPUT.PUT_LINE ('The average salary for all employees: '
                          || TO_CHAR(new_sal));
    adjust_salary(emp_id, new_sal); -- assigns a new value to new_sal
    DBMS_OUTPUT.PUT_LINE ('The adjusted salary for employee ' || TO_CHAR(emp_id)
                          || ' is ' || TO_CHAR(new_sal)); -- sal has new value
END;
/
```

شکل ۷-۳

در مثال بالا، متاحولی به نام sal به يك پروگرام فرعی فرستاده شده است. پروگرام فرعی قيمت اين متاحول را اپديت كرده، قيمت اصلی متاحول sal را با متاحول ديگري به نام avg_sal جمع كرده و تقسيم بر 2 می‌كند. در قسمت آخر پروگرام، يك لاين کد به خاطر نمايش دادن نتیجه نهایی پروگرام به کار رفته و آن عبارت است از:

DBMS_OUTPUT.PUT_LINE ...

طوری که دیده شد، متحولین در PL/SQL تعریف و استفاده می‌شوند. طریقه‌هایی که قیمت به متحول‌ها داده می‌شود، هر کدام با مثالی توضیح داده شدند. در یک پروگرام می‌شود طریقه‌های مختلف دادن قیمت‌ها به متحول‌ها باهم استفاده شوند؛ یعنی محدودیتی مبنی بر این که در یک پروگرام حتماً باید یک طریقه به کار گرفته شود، وجود ندارد، بلکه نظر به ضرورت، هر کدام از طریقه‌های توضیح داده شده در پروگرام‌های PL/SQL قابل استفاده‌اند.

۳.۵ انواع دیتای از قبل تعریف شده در PL/SQL (Predefined Datatypes)

قسمت اعظم دیتایی که در PL/SQL به خاطر تعریف کردن نوعیت یک فیلد استفاده می‌شود، از قبل تعریف شده‌اند. انواع دیتای (Datatypes) از قبل تعریف شده به صورت عمومی به چهار دسته تقسیم می‌شوند که در ذیل به آن‌ها پرداخته شده است. لازم به یادآوری است که دو نوع اول در لیست پایین به صورت مختصر توضیح داده شده و خارج از بحث این کتاب می‌باشند. دونوع دومی با تفصیل بیشتر تحت یک عنوان فرعی توضیح می‌شوند.

۱. نوع دیتا LOB (Large Object)

نوع دیتا LOB دربرگیرنده محتواهایی به نام موقعیت‌دهنده LOB اند که توسط آن‌ها موقعیت آبجکت‌های بزرگ مشخص می‌شود. مثال‌های آبجکت‌های بزرگ عبارت از بلاک‌های بزرگ متن و یا تصویرهای گرافیکی است که به صورت جداگانه از دیتای دیتابیس در کمپیوتر ذخیره‌اند. انواع دیتای این دسته عبارت از BFILE، BLOB، CLOB و NCLOB اند. این کتگوری از دیتا، خارج از بحث در این کتاب بوده و از آن صرف نظر گردیده است.

۲. نوع دیتای ریفرنس (Reference Datatypes)

نوع دیتای ریفرنس قیمت‌هایی را نشان می‌دهد که به نام نشان‌گرها (Pointers) یاد می‌شوند. این نشان‌گرها بخش‌های پروگرام‌های دیگر را در دیتابیس نشان می‌دهد؛ یعنی دیتا در آدرس دیگر و فارمت متفاوت ذخیره می‌باشد و به خاطر بازکردن آن در دیتابیس، ریفرنس یا اشاره به این نوع دیتا ذخیره می‌شود. مثال‌های این دسته از انواع دیتا عبارت از REF CURSORS و REFs به انواع آبجکت‌ها است. این کتگوری از دیتا نیز خارج از بحث این کتاب بوده و از آن صرف نظر گردیده است.

۳. نوع دیتای ترکیبی (Composite Datatypes)

این نوع دیتا دارای عناصر داخلی بوده که به گونه‌انفرادی قابل استفاده می‌باشند. مثال‌های نوع دیتای ترکیبی عبارت از عناصر یک Array، ریکورد و یا جدول در دیتابیس است. تفصیل بیشتر در عنوان بعدی آمده است.

۴. نوع دیتای سkalاری (Scalar Datatypes)

نوع دیتای سکالری برعکس نوع دیتای ترکیبی، هیچ عنصر داخلی ندارد. این نوع دیتا صرف یک قیمت مانند یک نمبر و یا یک کلمه را به خود می‌گیرد. تفصیل بیشتر در عنوان بعدی آمده است.

۳.۶ انواع دیتای ترکیبی (Scalar) و سکالری (Composite)

نوع دیتای ترکیبی طوری که توضیح شد، عناصر داخلی یک دیتابیس را مورد استفاده قرار می‌دهد. دیتای ترکیبی زمینه دسترسی و استفاده را به مجموعه‌های دیتا در یک دیتابیس مهیا می‌سازد. این کار با دیزاین کردن رشته‌هایی از دیتا (Arrays)، لیست‌ها، جدول‌های متداخل (Nested Tables)، ستها و غیره به منابع دیتا در داخل دیتابیس‌ها زمینه استفاده را فراهم می‌سازد. به خاطر انجام‌دادن این کار در دیتابیس‌های PL/SQL ساختمان‌هایی به نام TABLE که همان جدول دیتابیس است و VARRAY یعنی Array هایی با سایزهای متفاوت، استفاده می‌شوند.

نوع دیتای سکالری با قیمت‌های مستقل برای دیتا کار می‌کند. این نوع دیتا با عناصر داخلی دیتابیس مانند Arrayها و دیتای جدول‌ها کاری نداشته، بلکه هر جزء آن یک قیمت مستقل به خود می‌گیرد. مثال‌های نوع دیتای سکالری عبارت از یک عدد (نمبر)، یک قطعه از متن (یک یا چند کلمه) و غیره می‌تواند باشد. نوع دیتای سکالری به چهار بخش عددی، کرکتر، بولی (Boolean) و تاریخ/زمان تقسیم می‌شود.

مثال‌های مکمل انواع دیتای سکالری برای استفاده در PL/SQL در جدول پایین نشان داده شده است.

جدول 2 . 3 انواع دیتای سکالری قابل استفاده در PL/SQL

شماره	نام بخش نوع دیتا	مثال‌های مکمل نوع دیتا
1	PL/SQL Number Types	BINARY_DOUBLE, BINARY_FLOAT, BINARY_INTEGER, DEC, DECIMAL, DOUBLE PRECISION, FLOAT, INT, INTEGER, NATURAL, NATURALN, NUMBER, NUMERIC, PLS_INTEGER, POSITIVE, POSITIVEN, REAL, SIGNTYPE, SMALLINT
2	PL/SQL Character and String Types	CHAR, CHARACTER, LONG, LONG RAW, NCHAR, NVARCHAR2, RAW, ROWID, STRING, UROWID, VARCHAR, VARCHAR2
3	PL/SQL Boolean Types	BOOLEAN
4	PL/SQL Date, Time, and Interval Types	DATE, TIMESTAMP, TIMESTAMP WITH TIMEZONE, TIMESTAMP WITH LOCAL TIMEZONE, INTERVAL YEAR TO MONTH, INTERVAL DAY TO SECOND

۳.۷ استفاده از %TYPE Attribute در PL/SQL

با استفاده کردن از مشخصه %TYPE نوعیت دیتای یک متتحول و یا یک ستون در PL/SQL از یک منبع دیگر که نوعیت دیتای آن از قبل تعریف شده باشد، به ارث برده می‌شود. در کد PL/SQL داده شده در مثال پایین، متتحول‌هایی که با %TYPE تعریف شده‌اند، نوع دیتای خود را از متتحول مذکور همراه با قیمت‌های Default و قیدهای موجود (Constraints) به دست می‌آورد؛ به مثال در شکل ذیل توجه شود:

The screenshot shows the Oracle SQL Developer interface. The title bar reads "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons for file operations like New, Open, Save, Print, and a magnifying glass. The main workspace is titled "Welcome Page" and contains a "system" connection. The left sidebar has sections for Reports, Connections, and a tree view under "Connections". The central area is a "Worksheet" tab, which is active, showing a PL/SQL code editor. The code is as follows:

```
DECLARE
    credit PLS_INTEGER RANGE 1000..25000;
    debit credit%TYPE;
    v_name VARCHAR2(20);
    name VARCHAR2(20) NOT NULL := 'JoHn SmITH';
    -- If we increase the length of NAME, the other variables become longer also
    upper_name name%TYPE := UPPER(name);
    lower_name name%TYPE := LOWER(name);
    init_name name%TYPE := INITCAP(name);
BEGIN
    -- display inherited default values
    DBMS_OUTPUT.PUT_LINE('name: ' || name || ' upper_name: ' || upper_name
        || ' lower_name: ' || lower_name || ' init_name: ' || init_name);
-- lower_name := 'jonathan henry smithson'; invalid, character string is too long
-- lower_name := NULL; invalid, NOT NULL CONSTRAINT
-- debit := 50000; invalid, value out of range
END;
/
```

۸-۳

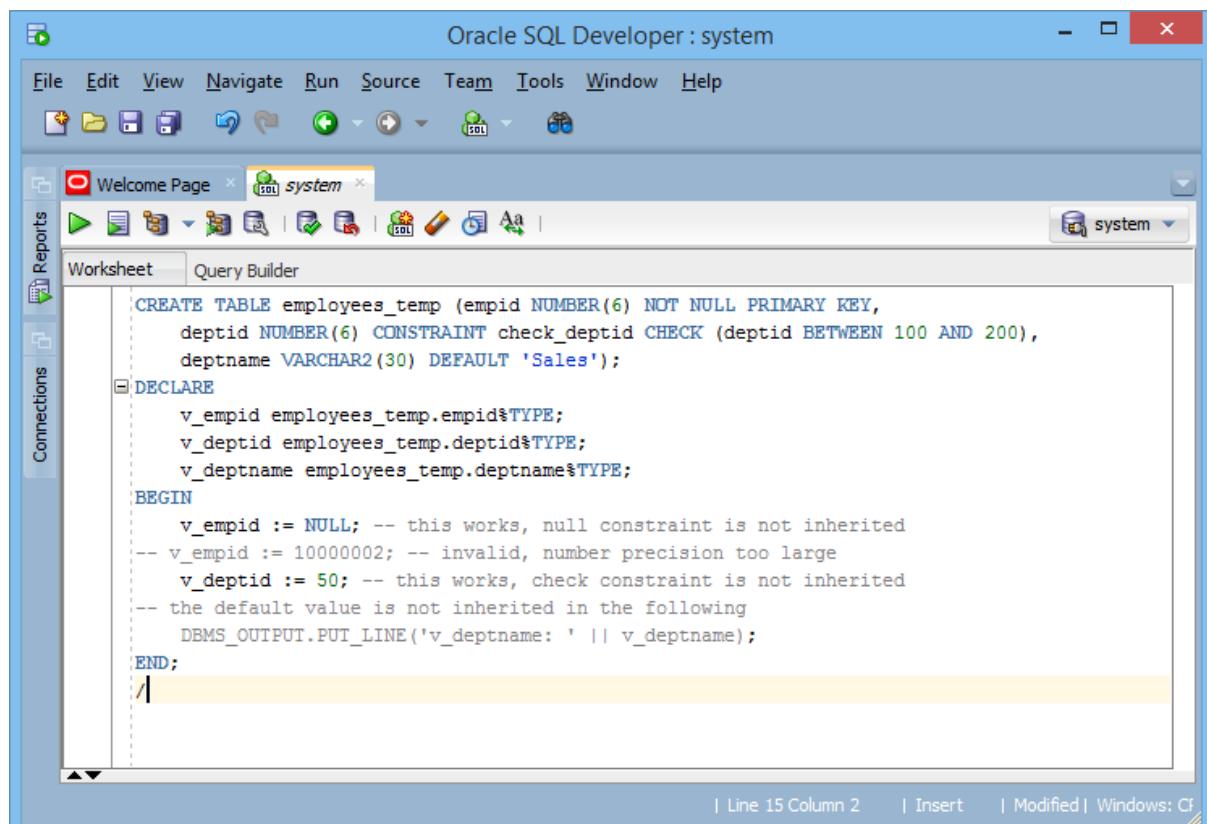
طوری که در شکل بالا دیده می‌شود، متحولهایی که با اختیار Type % تعریف شده‌اند، نوع دیتای خود را از متحولهایی که از قبل تعریف شده‌اند به ارث می‌گیرند. متحول name در قسمتهای اول کد با نوع دیتای VARCHAR2(20) تعریف شده است. سه متحول دیگر به نام‌های upper_name، lower_name و init_name با استفاده از نوعیت دیتای name را به خود می‌گیرند. در صورت تغییر یافتن نوع دیتای name، نوعیت دیتای سه متحول دیگر به صورت خودکار تغییر خواهد کرد.

۳.۸ فواید استفاده از Type Attribute

استفاده از مشخصه Type % زمانی دارای اهمیت می‌شود و موارد استفاده آن بیشتر مؤثر می‌شود که یک متحول، نوع دیتای خود را از ستون یک جدول در دیتابیس بگیرد؛ در چنین مواردی، دیدن نوع دیتای ستون

جدول شاید برای پروگرامر مهم نباشد و فقط می‌تواند ستون یا ستون‌های مورد نظر را با استفاده از %Type در کد PL/SQL تعریف کند. مزیت دیگر این عملیه در قسمت آپدیت کردن ساختمان جدول منبع است. امکان دارد متحول با نوع دیتای یک جدول عیار شده باشد و نوع دیتای ستون جدول توسط مالک آن تغییر داده شود. این تغییر نوع دیتا، باعث سکته‌گی و یا غلطی در پروگرام نشده، بلکه در زمان اجرای کد PL/SQL معلومات اپدیت شده با نوع دیتای جدیداً تعریف شده برای ستون مورد نظر استفاده می‌شود.

به خاطروضاحت موضوع فوق به مثال کد PL/SQL در شکل پایین توجه شود:



```

CREATE TABLE employees_temp (empid NUMBER(6) NOT NULL PRIMARY KEY,
    deptid NUMBER(6) CONSTRAINT check_deptid CHECK (deptid BETWEEN 100 AND 200),
    deptname VARCHAR2(30) DEFAULT 'Sales');

DECLARE
    v_empid employees_temp.empid%TYPE;
    v_deptid employees_temp.deptid%TYPE;
    v_deptname employees_temp.deptname%TYPE;
BEGIN
    v_empid := NULL; -- this works, null constraint is not inherited
-- v_empid := 10000002; -- invalid, number precision too large
    v_deptid := 50; -- this works, check constraint is not inherited
-- the default value is not inherited in the following
    DBMS_OUTPUT.PUT_LINE('v_deptname: ' || v_deptname);
END;
/

```

شکل ۹-۳

در شکل بالا، کد PL/SQL در دو بخش تنظیم شده است. در قسمت اول یک جدول به نام employee_temp با ستون‌های مکمل تعریف و ایجاد می‌شود. در قسمت دوم، بلاک کد PL/SQL در اعلامیه متحول‌هایی تعریف شده‌اند که نوع دیتایشان با استفاده از %Type از نوع دیتای ستون‌های مشخص شده جدول employee_temp گرفته می‌شود. در صورت تغییر نوع دیتای ستون‌های جدولی که توسط متحول‌ها مشخص شده‌اند، در زمان اجرای بلاک PL/SQL نوعیت دیتا تغییر کرده و به صورت اتومات برای متحول‌های مذکور در نظر گرفته می‌شود.

۳.۹ متحولهای بهم‌بسته (Bind Variables)

زمانی که دستورهای کار با دیتای سیکویل مانند داخل کردن دیتا، پاک کردن دیتا و یا هم خواندن دیتا در کد PL/SQL با هم استفاده می‌شوند، متتحولهای را به صورت اتمات به شکل متتحولهای بهم‌بسته یا Bind Variables تبدیل می‌کند. این عمل در زمان تطبیق شرایط و دادن قیمت‌ها در دستورها و کیوری‌ها اجراء می‌شود. اوریکل، این متتحولهای تشکیل شده به شکل بهم‌بسته را در هر بار اجرای همان کد دوباره استفاده می‌کند. به خاطر با اجراء گذاشتن متتحولهای بهم‌بسته با قیمت‌های جدید، سلسله انتقال قیمت‌ها به متتحولهای در پروسیجرهای ثبت شده می‌توانند که این پروسیجرها قیمت‌های جدید را به شکل پارامترها قبول کنند. قیمت‌های جدید، جای گزین قیمت‌های قبلی متتحولهای در جاهای معین آن‌ها شده و استفاده می‌شوند؛ یعنی متتحولهای بهم‌بسته (Bind Variables) کمک می‌کند تا برای متتحولهای آدرس‌هایی در نظر گرفته شود که در زمان اجراء، قیمت متتحول به آدرس مشخص آن تعلق گرفته و نشان داده شود. در زمان اجرای عملیه‌های یک برنامه PL/SQL، آدرس‌ها استفاده شده و وقتی نتیجه نشان داده می‌شود، قیمت‌های متتحولهای Bind اند، نشان داده می‌شود. در جدول‌های کوچک، تأثیر استفاده از متتحولهای Bind زیاد محسوس نیست؛ ولی زمانی که تعداد ریکوردهای دیتا به صدها هزار و میلیون‌ها می‌رسد، فواید استفاده از متتحولهای بهم‌بسته به وضاحت دیده می‌شود. توضیح این مسأله با استفاده از یک مثال عملی در درس بعدی در نظر گرفته شده است.

متتحولهای بهم‌بسته (Bind Variables) با سیکویل دینامیک در قسمت اصطلاحات WHERE و VALUES استفاده می‌شوند، البته استفاده از متتحولهای عادی نیز در همین قسمت کد PL/SQL به کار گرفته می‌شوند؛ یعنی به عوض جاسازی (Concatenating) قیمت‌های متتحولهای در یک رشته (String)، با استفاده از این میتود، متتحولهای تعریف شده به نام متتحولهای بهم‌بسته (Bind) تغییر می‌کنند. نامهای متتحولهای Bind با اضافه‌ساختن علامه شارحه (:) در پیش روی نام آن‌ها مشخص شده و با کلمه ریزرفی USING در یک دستور به کار گرفته می‌شوند. جمله پایین از یک قسمت کد PL/SQL گرفته شده و در آن متتحول Bind به کار رفته است:

```
'DELETE FROM employees WHERE employee_id = :id' USING emp_id;
```

مزیت استفاده از کلمه USING و متتحولهای بهم‌بسته به تناسب استفاده از سیستم یک‌جاسازی (Concatenating) در این است که مشکل استفاده از قیمت‌های متتحولهای در استفاده از کلمه USING کم ساخته شده و جمله‌هایی که قیمت‌ها را به متتحولهای می‌دهد، دوباره استفاده می‌شوند؛ یعنی پروسه‌های اجرایی یک پروگرام نسبتاً بزرگ PL/SQL با استفاده از متتحولهای بهم‌بسته به مراتب کمتر می‌شود.

در شکل پایین، کد یک پروگرام دینامیک PL/SQL نشان داده است که متتحولهای بهم‌بسته در آن استفاده شده‌اند.

The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar contains various icons for file operations like Open, Save, and Print, as well as connection management and search functions. The left sidebar has sections for Reports and Connections, with "system" selected. The main workspace is titled "Worksheet" and contains a PL/SQL block:

```

DECLARE
    sql_stmt      VARCHAR2(200);
    v_column      VARCHAR2(30) := 'DEPARTMENT_ID';
    dept_id       NUMBER(4)  := 46;
    dept_name     VARCHAR2(30) := 'Special Projects';
    mgr_id        NUMBER(6)  := 200;
    loc_id        NUMBER(4)  := 1700;
BEGIN
    -- note that there is no semi-colon (;) inside the quotes '...'
    EXECUTE IMMEDIATE 'CREATE TABLE bonus (id NUMBER, amt NUMBER)';
    sql_stmt := 'INSERT INTO departments VALUES (:1, :2, :3, :4)';
    EXECUTE IMMEDIATE sql_stmt USING dept_id, dept_name, mgr_id, loc_id; --Bind Variable
    EXECUTE IMMEDIATE 'DELETE FROM departments WHERE ' || v_column || ' = :num'
        USING dept_id; -- use of Bind Variable
    EXECUTE IMMEDIATE 'ALTER SESSION SET SQL_TRACE TRUE';
    EXECUTE IMMEDIATE 'DROP TABLE bonus';
END;
/

```

The status bar at the bottom indicates "Line 18 Column 2" and "Modified".

شکل ۱۰-۳

یک مثال استفاده از متتحولهای Bind

یک مثال عملی به خاطر پیدا کردن تفاوت استفاده از متتحولهای بهم بسته در نظر گرفته شده است. در قدم نخست دیتابی که عملیه باید بالای آن انجام شود، توسط دستورهای ذیل ایجاد می‌شود:

```

CREATE TABLE test_bind_var AS
    SELECT level col FROM dual
    CONNECT by level <= 99999
;
COMMIT;

CREATE INDEX test_index ON test_bind_var(col);

SELECT COUNT(*) FROM test_bind_var;

```

در فوق چهار دستور اجرایی سیکویل استفاده شده‌اند که توسط آن‌ها یک جدول امتحانی به نام `test_bind_var` ایجاد شده و در آن 99999 رکورد دیتا داخل می‌شود. یک اندرس به نام `test_index` برای ستون `col` جدول متذکره ایجاد شده است. دستور آخری به خاطر اطمینان از این‌که جدول ایجاد شده واقعاً به تعداد 99999 رکورد دارد و یا خیر، به راه انداخته شده است. شکل پایین نتیجه دستورهای اجرا شده را نشان می‌دهد.

```

C:\Users\main>sqlplus "/as sysdba"
SQL*Plus: Release 11.2.0.1.0 Production on Sun Feb 10 15:47:49 2019
Copyright (c) 1982, 2010, Oracle. All rights reserved.

Connected to:
Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production
With the Partitioning, OLAP, Data Mining and Real Application Testing options

SQL> CREATE TABLE test_bind_var AS
  2   SELECT level col FROM dual
  3   CONNECT by level <= 99999
  4 ;
Table created.

SQL>
SQL> COMMIT;

Commit complete.

SQL>
SQL> CREATE INDEX test_index ON test_bind_var(col);

Index created.

SQL>
SQL> SELECT COUNT(*) FROM test_bind_var;

  COUNT(*)
  -----
  99999

SQL>

```

شکل ۱۱-۳

طوری که در شکل بالا دیده می‌شود، جدولی با 99999 رکورد دیتا موفقانه ایجاد شده و ستونی به نام `col` آن نیز اندرس شده است. حال مرحله چک کردن استفاده و عدم استفاده از متتحوله‌ای به همبسته انجام می‌شود. در نخست کد PL/SQL ذیل به راه انداخته شود:

(این دستور به خاطر ثبت کردن زمان اجرای دستور استفاده شده است.)

DECLARE

```

a number;
i number;
testv varchar2(100);

```

BEGIN

```

i := 1;
WHILE i<=99999
LOOP
    testv := 'SELECT * FROM test_bind_var WHERE col=' || i;
    EXECUTE IMMEDIATE testv
        into a;
    i := 1+a;
END LOOP;
END;
/

```

شکل پایین، نتیجه اجرای قطعه کد بالا را بالای یک جدول ساده ولی با حدود یک صد هزار ریکورد نشان می‌دهد.

```

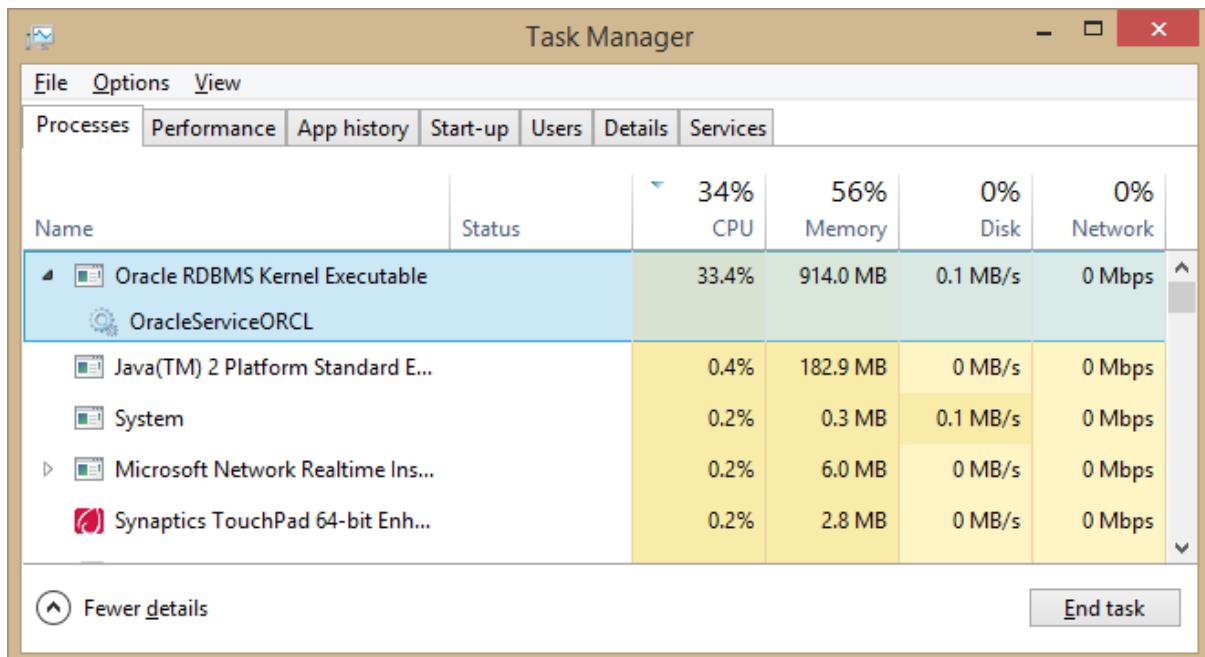
SQL> SET TIMING ON
SQL> DECLARE
  2   a      number;
  3   i      number;
  4   testv  varchar2(100);
  5 BEGIN
  6   i := 1;
  7   WHILE i<=99999
  8   LOOP
  9       testv := 'SELECT * FROM test_bind_var WHERE col=' || i;
 10       EXECUTE IMMEDIATE testv
 11           into a;
 12       i := 1+a;
 13   END LOOP;
 14 END;
 15 /
PL/SQL procedure successfully completed.

Elapsed: 00:10:17.99
SQL>

```

شکل ۱۲-۳

طوری که دیده می‌شود، به تعداد 99999 کیوری در مدت بیشتر از 10 ثانیه اجرا شدند که یک زمان بسیار طولانی به خاطر انجام دادن آن است. کیوری‌های فوق در حالتی اجرا شدند که در آن‌ها از متحول بهم‌بسته استفاده نشده است. شکل پایین اندازه پروسیسر و حافظه کمپیوچر، در حال اجرای کد مثال قبلی PL/SQL بدون استفاده از متحول‌های Bind را نشان می‌دهد.



شکل ۱۳-۳

طوری که در شکل بالا دیده می‌شود، 914 میگابایت حافظه و بیشتر از 30 فیصد کار پروسیسر از نوع Core i7 را اجرای کد PL/SQL مصرف کرده است.

حال به خاطر پیدا کردن تفاوت استفاده و عدم استفاده از متتحول های بهم بسته، قطعه کد بالا در عین ستد دیتا و در عین کمپیوتر، ولی با استفاده از متتحول Bind طور ذیل به اجراء گذاشته می‌شود:

SET TIMING ON

DECLARE

```
a number;
i number;
testv varchar2(100);
```

BEGIN

i := 1;

WHILE i<=99999

LOOP

```
testv := 'SELECT * FROM test_bind_var WHERE col=:1';
```

EXECUTE IMMEDIATE testv

into a

USING i;

```

    i := i+1;
END LOOP;
END;
/

```

شکل پایین، نتیجه اجرای قطعه کد آخر را بالای جدول با حدود یک صد هزار ریکورد نشان می دهد.

```

SQL> SET TIMING ON
SQL> DECLARE
  2   a      number;
  3   i      number;
  4   testv  varchar2(100);
  5 BEGIN
  6   i := 1;
  7   WHILE i<=99999
  8   LOOP
  9     testv := 'SELECT * FROM test_bind_var WHERE col=:1';
10     EXECUTE IMMEDIATE testv
11       into a
12       USING i;
13   i := i+1;
14 END LOOP;
15 END;
16 /
PL/SQL procedure successfully completed.

Elapsed: 00:00:01.10
SQL>

```

شکل ۱۴-۳

طوری که در شکل دیده می شود، استفاده از متتحولین به هم بسته، اجرای دستور را با ایجاد کردن 999999 کیوری در زمان 1.1 ثانیه به انجام رسانده است. تفاوت زمان اجراء در هر دو حالت بسیار زیاد بوده و غیر قابل مقایسه است. اگر به شکل های قبلی مراجعه شود، دیده می شود که در حالت عدم استفاده از متتحول های Bind اجرای دستور حدود 617 ثانیه را در بر گرفته بود. تناسب هر دو حالت اگر دیده شود، نتیجه ذیل به دست خواهد آمد:

$$617.9/1.1 = 561.7$$

يعنى استفاده از متتحول های Bind در مثال بالا، سرعت اجرای کد را 561.7 برابر بیشتر ساخته است. حال فکر شود که اگر تعداد ریکورد های دیتای جدول به میلیون ها بالا برد شود، تفاوت زمانی به کدام اندازه بالا خواهد رفت.



خلاصه فصل سوم

در این فصل درباره متتحولها و استفاده آنها در زبان PL/SQL بحث‌هایی صورت گرفته است. در عنوانیں اول فصل، به نام‌گذاری متتحولها منحیث یک اصل، اشاره صورت گرفته و شناسه‌های مجاز و غیر مجاز در قسمت استفاده از حروف، سمبل‌ها و غیره توضیح داده شده‌اند. لیست‌های مکمل کلمه‌های ریزرفی و کلمه‌های کلیدی PL/SQL به خاطر عدم استفاده در نام متتحول‌ها ضمیمه معلومات شده‌اند. تعریف کردن متتحول‌ها، موارد و طرز استفاده از متتحول‌ها شامل طریقه‌های دادن قیمت‌ها به متتحول‌ها در PL/SQL هر کدام به ترتیب تشریح شده‌اند.

انواع دیتای از قبل تعریف شده در PL/SQL به بحث گرفته شده است و توضیحات و مثال‌هایی در قسمت دو نوع دیتای از قبل تعریف شده، یعنی نوع ترکیبی (Composite Datatype) و نوع سکالری (Scalar Datatype) به محتویات فصل افزوده شده است. استفاده و به کارگیری یک نوع مشخصه به نام %Type با فواید آن به خاطر تعریف کردن نوع دیتای متتحول‌ها به شکل ارشی از نوع دیتای متتحول‌ها و یا ستون‌های جدول‌ها با مثال‌های لازم توضیح و نشان داده شده‌اند. در قسمت آخر فصل، یکی دیگر از عنوانیں مهم مربوط به تعریف متتحول‌ها به نام متتحول‌های به همبسته (Bind Variables) گنجانیده شده است. یک مثال عملی در این عنوان به خاطر نشان دادن اهمیت استفاده کردن متتحول‌های به همبسته در PL/SQL با نتایج به دست آمده آن ضمیمه محتویات فصل سوم شده‌اند.

سوالات فصل سوم



۱. شناسه‌ها (Identifiers) در زبان PL/SQL به چه مفهوم‌اند؟
۲. کمترین و بیشترین طول مجاز برای شناسه‌ها در PL/SQL چند است؟ واضح کنید.
۳. کلمه‌های ریزرفشده (Reserved Words) زبان PL/SQL را توضیح دهید.
۴. کلمه‌های کلیدی (Keywords) با کلمه‌های ریزرفی در PL/SQL چی تفاوت دارد؟ با چند مثال از هر کدام واضح سازید.
۵. چند کلمه را نام ببرید که به حیث کلمات ریزرفشده در هر سه محیط SQL، PL/SQL و ANSI استفاده می‌شوند.
۶. به چند طریقه در زبان PL/SQL به متتحول‌ها قیمت داده می‌شود؟ یکی از طریقه‌های آن را که برای تان واضح است با مثال بیان کنید.
۷. انواع دیتای از قبل تعریف‌شده در PL/SQL را نام بگیرید.
۸. اصطلاح VARRAY به چه مفهوم بوده و در کجا استفاده می‌شود؟
۹. فواید استفاده از %Type Attribute را با یک مثال واضح سازید.
۱۰. نحوه کار متتحول‌های Bind را در PL/SQL به صورت خلاصه توضیح دهید.

فعالیت‌های فصل سوم



۱. با استفاده از دستور "sqlplus "/ as sysdba" برنامه سیکویل پلس را از کوماند پرامپت کمپیوترتان به راه بیندازید.
۲. با استفاده از دستور HELP INDEX لیست بخش‌های معلومات را ببینید.
۳. با استفاده از دستور HELP RESERVED WORDS و یا RESERVED WORDS کلمه‌های ریزرفشده SQL، PL/SQL و ANSI را ببینید.
۴. برنامه توسعه‌دهنده سیکویل (SQL Developer) را با راهاندازی فایل اجرایی sqldeveloper از دایرکتوری مربوطه آن به راه بیندازید.
۵. در صفحه توسعه‌دهنده سیکویل با استفاده از دکمه مخصوص ایجاد رابطه در بخش Connections به دیتابیسی به نام system وصل شوید (به خاطر ایجاد کردن رابطه به این دیتابیس از پاسوردی که در زمان نصب تعیین کرده‌اید، استفاده کنید).
۶. با استفاده از دستور ... DECLARE چند متتحول را در صفحه ایدیتور کد نوشته و به آن‌ها با استفاده از سمبول := قیمت‌های مطابق با نوع دیتای هر کدام را بنویسید (می‌توانید از انواع دینا BOOLEAN، VARCHAR2 و NUMBER استفاده کنید).
۷. یک مثال ساده از پروگرام PL/SQL را بنویسید که در آن نوعیت دیتای متتحول‌ها با استفاده از مشخصه %Type Attribute تعریف شده باشد.

فصل چهارم

جملات قابل اجرا در PL/SQL



هدف کلی: شاگردان با نوشتن انواع جملات قابل اجرا زبان PL/SQL آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. اهداف و طرز استفاده از Lexical Units را در یک بلاک زبان PL/SQL بدانند.
۲. اهداف و طرز استفاده از توابع از قبل ساخته شده (Built in) را با PL/SQL بدانند.
۳. تفاوت و موارد استفاده از تبدیل کردن نوع دیتا به شکل آشکار (Explicit Conversion) و تبدیل کردن نوع دیتا به شکل پوشیده یا ضمنی (Implicit Conversion) را در PL/SQL بفهمند.
۴. اهداف استفاده از متتحولها، بلاک‌های آشیانه‌بی (Nested Blocks) با لیبل‌ها را بیاموزند.
۵. موارد استفاده از ترتیب‌ها (Sequences) را با عبارات PL/SQL بدانند.

بحث جملات قابل اجراء در PL/SQL بیشتر روی توضیح ساختمانهای قابل استفاده در این زبان برنامه‌نویسی می‌چرخد. کد نوشته شده در یک زبان پروگرام‌نویسی باید قواعدی داشته باشد که با ترجمه آن‌ها یک سیستم کمپیوتر جهت اجرای دستورهای مورد نظر به صورت درست کار کند. به عین ترتیب قواعد نوشتاری کد یک زبان کمک می‌کند تا اشخاص مسلکی‌یی که در یک محیط کار می‌کنند و یا زبان پروگرام‌نویسی را می‌آموزند، به صورت دقیق بالای موضوع مورد نظر کار کنند؛ طور مثال، چند شخص مسلکی روی اکشافدادن یک بلاک PL/SQL کار می‌کنند. این افراد در مقاطع مختلف وقت دارند که بالای پروژه متذکره کار کنند. به خاطر این که کار انجام شده را تحلیل کرده بتوانند، همین استندردهای نوشتاری ایشان را کمک می‌کند تا هدف یکدیگر را بفهمند. در صورت عدم موجودیت چنین قواعدی، هماهنگی بین اعضای گروپ و یا هم استفاده کنندگان بعدی پروگرام تهیه شده، ممکن نخواهد بود.

در فصل چهارم به موضوعات مهمی در این عرصه پرداخته شده است. اهداف و طرز استفاده از اکثر مسائل مربوط به قواعد نوشتاری از قبیل Lexical Units، توابع از قبل ساخته شده، تبدیل کردن نوعیت دیتا در بلاک‌های کد، متحول‌ها، بلاک‌های آشیانه‌یی (Nested Blocks) و ترتیب‌ها (Sequences) در PL/SQL تحت عنوانی جداگانه توضیح شده‌اند. نظر به ضرورت در بعضی عنوان‌ها به مسائل فرعی مانند توضیحات در مورد شناسه‌ها، کلمه‌های ریزرفشده، حائل‌ها (Delimiters)، لیترال‌ها (Literals)، طریقه‌های مختلف تبدیل کردن دیتا و غیره نیز پرداخته شده است. بخش‌هایی که استفاده عملی آن‌ها بیشتر است، با مثال‌های تطبیقی مستندسازی شده‌اند.

۴.۱ اهداف و طرز استفاده از Lexical Units (PL/SQL در یک بلاک)

در نوشنون کد زبان‌های پروگرام‌نویسی، قواعدی در نظر گرفته می‌شود. هر زبان قواعد نوشتاری خود را دارد. این امر در زبان‌های تکلمی بشر نیز دیده می‌شود؛ طور مثال، در زبان نوشتاری فارسی دری یک جمله باید دارای قسمت‌های مشخص اسم، مبتدا، فعل، خبر و حروف ربط باشد. هر قسمت یک جمله از نظر دستور زبان و مفاهیم مورد نظر به شکل‌های مختلفی مانند اسم جمع، اسم مفرد، فعل حال، فعل گذشته و غیره نوشته می‌شود. به خاطر نوشنون حالت‌های مختلف قسمت‌های یک جمله، قواعدی وضع شده که به صورت استندرد باید در نظر گرفته شوند. همین مسئله در زبان‌های پروگرام‌نویسی در ساحة کمپیوترساینس نیز تطبیق می‌گردد.

در زبان‌های پروگرام‌نویسی، مراعات کردن استندردهای نوشتاری بخش‌های کد به نام Lexical Units یاد می‌شوند. در نوشنون بلاک‌های کد زبان PL/SQL بخش‌های نوشتاری مجاز آن‌ها به اساس استندردهای این زبان به یکی از بخش‌های پایین ارتباط دارد.

۴.۱.۱ شناسه‌ها (Identifiers)

شناسه‌ها به هدف شناسایی و نام‌گذاری بخش‌های کد PL/SQL استفاده می‌شوند. در مورد شناسه‌ها در فصل قبلی نیز توضیحاتی ارائه شد. در نام‌گذاری بخش‌ها در بلاک کد زبان PL/SQL یک نام (شناسه) باید به یک حرف زبان انگلیسی شروع شود و طول آن از 30 کرکتر تجاوز نکند؛ برعلاوهٔ حروف، در نام شناسه‌ها از نماینده‌ها، علامه‌های \$، # و خط زیرین _ نیز به کار می‌رود. نکته قابل توجه این است که استفاده از اعداد و علامه‌های مجاز در شروع نام شناسه‌ها قابل استفاده می‌باشد.

۴.۱.۲ کلمه‌های ریزرفشده (Reserved Words)

کلمه‌های ریزرفی زبان PL لیستی مکمل از کلمه‌هایی است که در بلاک کد این زبان به حیث کلمه‌های شناخته‌شده با مفاهیم مشخص به کار برده می‌شوند. لیست مکمل این کتگوری از کلمه‌ها نیز در فصل قبلی ارائه شده است. مثال‌های کلمه‌های ریزرفی PL/SQL عبارت از SELECT، END، BEGIN و غیره‌اند. به خاطر دیدن کلمه‌های ریزرفشده PL/SQL از دستور HELP RESERVED WORDS در سیکویل پلس استفاده می‌شود.

۴.۱.۳ حائل‌ها (Delimiters)

این دسته شامل کرکترهایی در PL/SQL می‌شود که به شکل تنها و یا به شکل ترکیبی استفاده شده و معانی خاصی را در این زبان افاده می‌کنند. عملیه‌های ریاضی و علامه‌های نقل قول، از مثال‌های معروف حائل‌ها به حساب می‌آیند. به خاطر وضاحت بیشتر در مورد حائل‌ها، لیست این سمبول‌ها در جدول پایین با توضیح به زبان انگلیسی ارائه شده است

جدول ۴.۱ حائل‌های زبان PL/SQL

شماره	سمبول	توضیح	شماره	سمبول	توضیح
1	+	addition operator	18	:=	assignment operator
2	%	attribute indicator	19	=>	association operator
3	'	character string delimiter	20		concatenation operator
4	.	component selector	21	**	exponentiation operator
5	/	division operator	22	<<	label delimiter (begin)
6	(expression or list delimiter	23	>>	label delimiter (end)
7)	expression or list delimiter	24	/*	multi-line comment (begin)
8	:	host variable indicator	25	*/	multi-line comment (end)
9	,	item separator	26	..	range operator
10	*	multiplication operator	27	<>	relational operator
11	"	quoted identifier delimiter	28	!=	relational operator

12	=	relational operator	29	~=	relational operator
13	<	relational operator	30	^=	relational operator
14	>	relational operator	31	<=	relational operator
15	@	remote access indicator	32	>=	relational operator
16	;	statement terminator	33	--	single-line comment indicator
17	-	subtraction/negation operator			

۴.۱.۴ لیترال‌ها (Literals)

لیترال اصطلاحی است که معنای ابتدایی یک حرف، کلمه و یا سимвول را ارائه می‌کند. در زبان PL/SQL لیترال‌ها مفاهیمی را شامل می‌شود که آن‌ها در شناسه تعریف نشده باشند. شناسه‌های SQL مطابق توضیحات قبلی شامل ثابت‌ها (Constants)، متغول‌ها (Variables)، استثناهای (Exceptions)، کرسرهای (Cursors)، کرسرهای متغول (Subprograms)، برنامه‌های فرعی (Cursor Variables)، پروسیجرها (Procedures)، کلمه‌های ریزرفشده (Reserved Words)، کلمه‌های کلیدی (Keywords) و پکیج‌ها (Packages) هستند. کلمه‌هایی غیر از این‌ها در PL/SQL به نام لیترال‌ها یاد می‌شوند. لیترال‌ها در PL/SQL مانند دیتای سکالری می‌توانند به اشکال عددی، کرکتر، رشته‌یی (String)، بولی (Boolean) و تاریخ/زمان باشند.

۱. لیترال‌های عددی (Numeric Literals): دو نوع لیترال عددی در عبارت‌های ریاضی استفاده می‌شود که عبارت از اعداد صحیح (Integers) و اعداد حقیقی (Real Numbers) اند. یک لیترال عدد صحیح می‌تواند با علامه مثبت و یا منفی، ولی بدون خانه اعشاری نشان داده شود. مثال‌های لیترال‌های صحیح طور ذیل اند:

7, 0, -19, +4221, ...

یک لیترال نمبر حقیقی نیز می‌تواند با علامه مثبت و یا منفی، ولی با داشتن خانه‌های اعشاری در PL/SQL استفاده شود. تفاوت اعداد صحیح و حقیقی موجودیت خانه اعشاری در آن‌ها است. مثال‌های لیترال‌های حقیقی عبارت اند از:

3.14, 0.55, -14.1, +8000.0001, .75, ...

کد پایین، مثال استفاده از لیترال‌های عددی را در زبان PL/SQL نشان می‌دهد.

DECLARE

```
x BINARY_FLOAT := sqrt(2.0f);
```

```

-- single-precision floating-point number

y BINARY_DOUBLE := sqrt(2.0d);

-- double-precision floating-point number

BEGIN

NULL;

END;

/

```

شکل پایین تطبیق کد بالا را به شکل موقیت‌آمیز آن نشان می‌دهد.

```

SQL>
SQL>
SQL>
SQL> DECLARE
2   x BINARY_FLOAT := sqrt(2.0f);
3   -- single-precision floating-point number
4   y BINARY_DOUBLE := sqrt(2.0d);
5   -- double-precision floating-point number
6 BEGIN
7   NULL;
8 END;
9 /
PL/SQL procedure successfully completed.

Elapsed: 00:00:00.08
SQL>

```

1-۴

۲. لیترال‌های کرکتر (Character Literals): لیترال‌های کرکتر در PL/SQL یک کرکتر (حرف و یا سمبول) را در بین نشانه نقل قول یک تایی (') استفاده می‌کند. در این دسته تمام کرکترهای قابل چاپ شدن شامل می‌شوند. کرکترهای قابل چاپ و شناسایی در PL/SQL می‌توانند حروف، اعداد، خانه‌خالی‌ها و سمبول‌های خاص باشند. مثال‌های لیترال‌های کرکتر طور ذیل اند:

'Z', 'z', '%', '8', ' ', ')', ...

برعکس اکثر موارد استفاده دیتا در PL/SQL، استفاده از حروف کوچک و بزرگ در لیترال‌ها فرق می‌کند. طوری که در مثال‌های بالا دیده می‌شود، حرف Z کلان با حرف z کوچک دو لیترال جداگانه محسوب می‌شوند. به عین شکل لیترال‌های عددی '0' الی '9' نیز به مفهوم عددی‌شان نبوده و معادل لیترال‌های عددی صحیح، که قبلًا توضیح شد، نیستند؛ ولی زمانی که این نوع لیترال‌ها در عبارت‌های ریاضی استفاده

می‌شوند با در نظرگرفتن تبدیلی ضمنی یا پوشیده (Implicitly Convertible) آن‌ها، با قیمت‌های واقعی‌شان استفاده می‌شوند.

۳. لیترال‌های رشته‌بی (String Literals): یک کرکتر و یا رشته‌بی از کرکترها می‌توانند توسط یک شناسه (Identifier) مانند متحول در زبان PL/SQL نشان داده شده و یا هم می‌توانند با استفاده از لیترال‌های رشته‌بی در یک پروگرام PL/SQL استفاده شوند. یک لیترال رشته‌بی می‌تواند هیچ (صفر) کرکتر و یا چندین کرکتر را در برداشته باشد. لیترال‌های رشته‌بی مانند لیترال‌های کرکتر با سمبل نقل قول یک تایی مشخص می‌شوند. تفاوت بین لیترال‌های کرکتر و لیترال‌های رشته‌بی در این است که در نوع اول تنها یک کرکتر استفاده شده و در نوع دوم صفر، یک و یا بیشتر از یک کرکتر قابل استفاده است.

تمام لیترال‌های رشته‌بی بدون لیترال رشته خالی (Null) دارای دیتای نوع CHAR می‌باشند. مثال‌های لیترال‌های رشته‌بی عبارت‌اند از:

'Hello, World!'

'BBC World News'

'13-FEB-19'

...

مسئله درنظرگرفتن تفاوت بین حروف کوچک و بزرگ مانند لیترال کرکتر که قبلاً توضیح شد، در لیترال‌های رشته‌بی نیز مراعات می‌شود؛ طور مثال، لیترال‌های 'STUDENT'، 'Student' و 'student' با هم دیگر تفاوت داشته و در PL/SQL هر کدام‌شان به صورت یک لیترال رشته‌بی مستقل در نظر گرفته می‌شود.

۴. لیترال‌های بولی (Boolean Literals): لیترال‌های بولی که در اصطلاح به آن‌ها لیترال‌های صحیح/غلط و یا درست/نادرست نیز می‌گویند، در زبان PL/SQL قابل تعریف و استفاده می‌باشند. معمولاً این نوع لیترال‌ها را برای محتواهای نامعلوم و غیر قابل تطبیق استفاده می‌کنند. نکته‌بی که در مورد لیترال‌های بولی باید به خاطر سپرده شود این است که این لیترال‌ها رشته‌بی نبوده، بلکه قیمت‌هایی اند که در موارد مشخص مدیریت دیتا استفاده می‌شوند.

۵. لیترال‌های تاریخ/زمان (Date/Time Literals): این نوع لیترال‌ها مطابق مفهوم نام‌شان به خاطر نشان‌دادن معلومات تاریخ، زمان و یا هر دو در کد زبان PL/SQL به کار گرفته می‌شوند. برای

استفاده از این لیترال‌ها، فارمتهای مختلف موجود است. قطعه کد ذیل استفاده از چند فارمت این نوع لیترال‌ها را به گونه نمونه نشان می‌دهد.

DECLARE

d1 DATE := DATE '1998-12-25';

t1 TIMESTAMP := TIMESTAMP '1997-10-22 13:01:01';

t2 TIMESTAMP WITH TIME ZONE :=

TIMESTAMP '1997-01-31 09:26:56.66 +02:00';

-- Three years and two months

-- For greater precision, use the day-to-second interval

i1 INTERVAL YEAR TO MONTH := INTERVAL '3-2' YEAR TO MONTH;

-- Five days, four hours, three minutes, two and 1/100 seconds

i2 INTERVAL DAY TO SECOND :=

INTERVAL '5 04:03:02.01' DAY TO SECOND;

BEGIN

NULL;

END;

/

شکل پایین تطبیق کد بالا را بدون غلطی (قبول شده توسط توسعه PL/SQL) نشان می‌دهد.

The screenshot shows a command prompt window titled "Command Prompt - sqlplus "/as sysdba". The code entered is:

```

SQL> DECLARE
 2      d1 DATE := DATE '1998-12-25';
 3      t1 TIMESTAMP := TIMESTAMP '1997-10-22 13:01:01';
 4      t2 TIMESTAMP WITH TIME ZONE :=
 5          TIMESTAMP '1997-01-31 09:26:56.66 +02:00';
 6      -- Three years and two months
 7      -- For greater precision, use the day-to-second interval
 8      i1 INTERVAL YEAR TO MONTH := INTERVAL '3-2' YEAR TO MONTH;
 9      -- Five days, four hours, three minutes, two and 1/100 seconds
10      i2 INTERVAL DAY TO SECOND :=
11          INTERVAL '5 04:03:02.01' DAY TO SECOND;
12  BEGIN
13      NULL;
14  END;
15 /

```

PL/SQL procedure successfully completed.

Elapsed: 00:00:00.00

شکل ۲-۴

۴.۱.۵ نظریات در PL/SQL Comments

نظریات منحیث بخش آخر اهداف و طرز استفاده از Lexical Unites در زبان PL/SQL یکی از بخش‌های اختیاری پروگرام است؛ یعنی نظریات (Comments) در اجرای دستورها کدام تأثیری نداشته، ولی استفاده از آن‌ها در کد پروگرام PL/SQL عاری از مفاد نیست. نظریات (Comments) در بین کد به خاطر وضاحت مسائل استفاده شده به کار می‌رond. مثال‌های استفاده از نظریات (Comments) بی‌شمار بوده که از جمله به چند مورد آن اشاره می‌شود.

در پژوهه‌هایی که چندین فرد مسلکی روی کد برنامه کار می‌کنند، استفاده از نظریات (Comments) در قسمت‌های مبهم کد، می‌تواند یک راهنمای خوب به خاطر هماهنگی بین افراد ذیدخواهد باشد. نظریات (Comments) همچنان به منظور آموزش می‌تواند توضیحاتی را برای نوآموزان در بر داشته باشد. قطعات اضافی کد در یک پروگرام با استفاده از امکانات نظریات (Comments) به خاطر استفاده کردن در آینده می‌توانند به شکل بدون اجراء حفظ شوند.

نظریات (Comments) در زبان PL/SQL به دو شکل استفاده می‌شوند:

۱. نظریات یک سطری (Single Line Comments): نظریات یک سطری با سمبل دو دانه دش (--) در یک سطر کد PL/SQL آغاز شده و تا انتهای همان سطر ادامه پیدا می‌کند. نظریات یک سطری می‌توانند از شروع خط تنظیم شوند و یا هم از وسط خط تنظیم و اضافه شوند. در حالت اول، تمام خط بدون اجراء بوده و در حالت دوم قسمت پیش از سمبل نظریه به شکل کد قابل اجراء در پروگرام استفاده شده و قسمت بعد از سمبل بدون اجراء گذاشته می‌شود. مثال استفاده از نظریات یک سطری در آخرین شکل درس قبلی دیده می‌شود. سطرهای ۶، ۷ و ۹ به صورت مکمل تفسیرها بوده و قابلیت اجراشدن در پروگرام را ندارند.

۲. نظریات چند سطrix (Multi Line Comments)

در نظریات چند سطrix، طوری که از نام آنها پیداست، یک یا اضافه از یک سطر در پروگرامهای زبان PL/SQL به صورت تفسیرها تنظیم می شوند. این نوع نظریات با سمبول /* آغاز شده و با سمبول */ پایان می یابند. در مثال پایین استفاده از نظریات چند سطrix نشان داده شده است:

```
DECLARE
    some_condition BOOLEAN;
    pi NUMBER := 3.1415926;
    radius NUMBER := 15;
    area NUMBER;

BEGIN
    /* Perform some simple tests and assignments */
    IF 2 + 2 = 4 THEN
        some_condition := TRUE;
        /* We expect this THEN to always be performed */
    END IF;
    /* The following line computes the area of a circle using pi,
       which is the ratio between the circumference and diameter.
       After the area is computed, the result is displayed. */
    area := pi * radius**2;
    DBMS_OUTPUT.PUT_LINE('The area is: ' || TO_CHAR(area));
END;
/
```

نتیجه تطبیق کد در شکل پایین نشان داده شده است.

```
SQL> SET SERVEROUTPUT ON FORMAT WRAPPED
SQL> DECLARE
 2   some_condition BOOLEAN;
 3   pi NUMBER := 3.1415926;
 4   radius NUMBER := 15;
 5   area NUMBER;
 6 BEGIN
 7   /* Perform some simple tests and assignments */
 8   IF 2 + 2 = 4 THEN
 9     some_condition := TRUE;
10   /* We expect this THEN to always be performed */
11 END IF;
12 /* The following line computes the area of a circle using pi,
13 which is the ratio between the circumference and diameter.
14 After the area is computed, the result is displayed. */
15 area := pi * radius**2;
16 DBMS_OUTPUT.PUT_LINE('The area is: ' || TO_CHAR(area));
17 END;
18 /
The area is: 706.858335
PL/SQL procedure successfully completed.
Elapsed: 00:00:00.05
SQL>
```

۳-۴ شکل

طوری که در شکل دیده می‌شود، نظریات چند سطحی به صورت جداگانه در سطوح ۷، ۱۰ و ۱۲ الی ۱۴ استفاده شده‌اند. در سطوح‌ای یادشده، متن جملات به انگلیسی نوشته شده که اگر از حالت نظریه کشیده شوند، کد قابل اجراء نبوده و غلطی‌هایی را مطابق شکل زیر به بار می‌آورد:

```
SQL> SET SERVEROUTPUT ON FORMAT WRAPPED
SQL> DECLARE
 2   some_condition BOOLEAN;
 3   pi NUMBER := 3.1415926;
 4   radius NUMBER := 15;
 5   area NUMBER;
 6 BEGIN
 7   Perform some simple tests and assignments
 8   IF 2 + 2 = 4 THEN
 9     some_condition := TRUE;
10   /* We expect this THEN to always be performed */
11 END IF;
12 /* The following line computes the area of a circle using pi,
13 which is the ratio between the circumference and diameter.
14 After the area is computed, the result is displayed. */
15 area := pi * radius**2;
16 DBMS_OUTPUT.PUT_LINE('The area is: ' || TO_CHAR(area));
17 END;
18 /
Perform some simple tests and assignments
*
ERROR at line 7:
ORA-06550: line 7, column 10:
PLS-00103: Encountered the symbol "SOME" when expecting one of the following:
:= . < > % ;
```

۴-۴ شکل

٤.٢ اهداف و طرز استفاده از توابع از قبل ساخته شده در PL/SQL

استفاده از تابع‌ها (Functions) در زبان‌های پروگرام‌نویسی رول بارزی دارد. به کمک توابع، قسمت‌های مختلف پروگرام‌های نوشته شده در زبان PL/SQL غنی ساخته می‌شوند. در این زبان، تابع‌های مختلفی به هدف کارکردن با دیتا به شکل از قبل ساخته شده (Built in) موجوداند. تابع‌های از قبل ساخته شده یا Built in در زبان PL/SQL به دسته‌های عمومی ذیل تقسیم می‌شوند:

١. تابع‌های گزارش‌دادن غلطی‌ها (Error Reporting Functions)
٢. تابع‌های نمبر (Number Functions)
٣. تابع‌های کرکتر (Character Functions)
٤. تابع‌های تغییر نوعیت دیتا (Datatype Conversion Functions)
٥. تابع‌های تاریخ (Data Functions)
٦. تابع‌های ریفرنس‌دهی آبجکت‌ها (Object Reference Functions)
٧. تابع‌های متفرقه (Miscellaneous Functions)

لیست مکمل توابع از قبل ساخته شده برای PL/SQL در دو جدول پایین به اساس نوع آن‌ها شماره‌گذاری و نشان داده شده است.

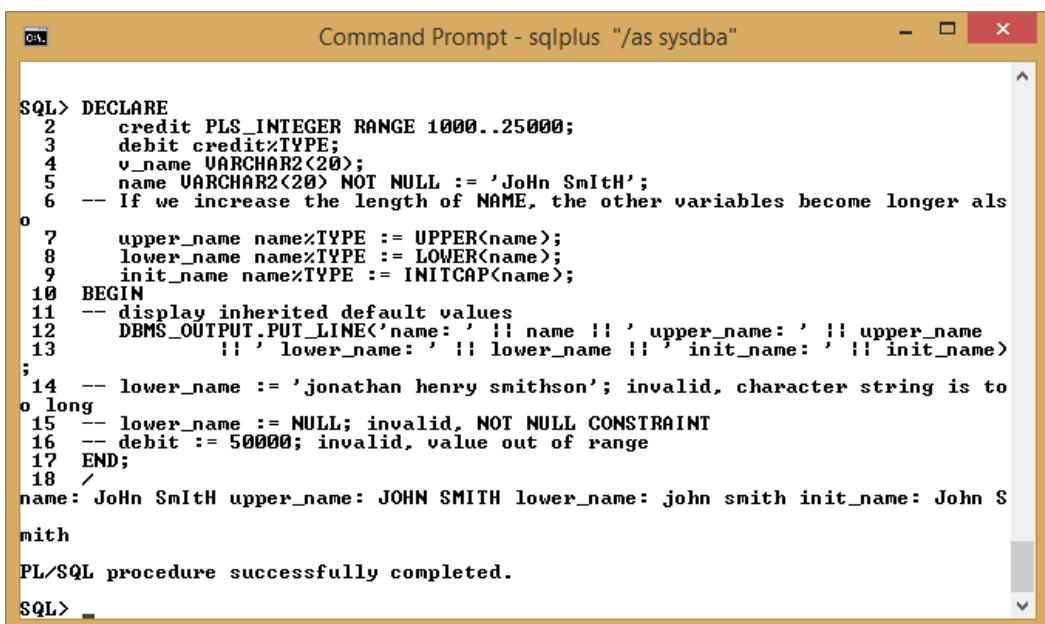
جدول ٤.٢ چهار دسته اول از توابع از قبل ساخته شده زبان PL/SQL

N.	Error	N.	Number		N.	Character		N.	Conversion
1	SQLCODE	1	ABS	22	SQRT	1	ASCII	22	NLS_INITCAP
2	SQLERRM	2	ACOS	23	TAN	2	ASCIISTR	23	NLS_LOWER
		3	ASIN	24	TANH	3	CHR	24	NLSSORT
		4	ATAN	25	TRUNC	4	COMPOSE	25	NLS_UPPER
		5	ATAN2			5	CONCAT	26	REGEXP_INSTR
		6	BITAND			6	DECOMPOSE	27	REGEXP_LIKE
		7	CEIL			7	INITCAP	28	REGEXP_REPLACE
		8	COS			8	INSTR	29	REGEXP_SUBSTR
		9	COSH			9	INSTR2	30	REPLACE
		10	EXP			10	INSTR4	31	RPAD
		11	FLOOR			11	INSTRB	32	RTRIM
		12	LN			12	INSTRC	33	SOUNDEX
		13	LOG			13	LENGTH	34	SUBSTR
		14	MOD			14	LENGTH2	35	SUBSTR2
		15	POWER			15	LENGTH4	36	SUBSTR4
		16	REMAIN			16	LENGTHB	37	SUBSTRB
		17	DER			17	LENGTHC	38	SUBSTRC
		18	ROUND			18	LOWER	39	TRANSLATE
		19	SIGN			19	LPAD	40	TRIM
		20	SIN			20	LTRIM	41	UNISTR
		21	SINH			21	NCHR	42	UPPER

جدول ۴ . ۳ سه دسته دوم از توابع از قبل ساخته شده زبان PL/SQL

N.	Date			N.	Obj. Ref.	N.	Miscellaneous
1	ADD_MONTHS	21	TIMESTAMP_TO_SCN	1	DEREF	1	BFILENAME
2	CURRENT_DATE	22	TO_DSINTERVAL	2	REF	2	COALESCE
3	CURRENT_TIME	23	TO_TIME	3	TREAT	3	DECODE
4	CURRENT_TIMESTAMP	24	TO_TIME_TZ	4	VALUE	4	DUMP
5	DBTIMEZONE	25	TO_TIMESTAMP			5	EMPTY_BLOB
6	EXTRACT	26	TO_TIMESTAMP_TZ			6	EMPTY_CLOB
7	FROM_TZ	27	TO_YMINTERVAL			7	GREATEST
8	LAST_DAY	28	TRUNC			8	LEAST
9	LOCALTIMESTAMP	29	TZ_OFFSET			9	NANVL
10	MONTHS_BETWEEN					10	NLS_CHARSET_DECL_LEN
11	NEW_TIME					11	NLS_CHARSET_ID
12	NEXT_DAY					12	NLS_CHARSET_NAME
13	NUMTODSINTERVAL					13	NULLIF
14	NUMTOYMININTERVAL					14	NVL
15	ROUND					15	SYS_CONTEXT
16	SCN_TO_TIMESTAMP					16	SYS_GUID
17	SESSIONTIMEZONE					17	UID
18	SYS_EXTRACT_UTC					18	USER
19	SYSDATE					19	USERENV
20	SYSTIMESTAMP					20	VSIZE

مثال‌های استفاده از توابع از قبل ساخته شده در کد PL/SQL زیاداند؛ به صورت نمونه در یک مثال، سه تابع از لیست بالا مربوط دسته کرکتر استفاده شده و نتیجه آن در شکل پایین نشان داده شده است.



```

SQL> DECLARE
 2    credit PLS_INTEGER RANGE 1000..25000;
 3    debit credit%TYPE;
 4    v_name VARCHAR2(20);
 5    name VARCHAR2(20) NOT NULL := 'JoHn SmItH';
 6    -- If we increase the length of NAME, the other variables become longer als
o
 7    upper_name name%TYPE := UPPER(name);
 8    lower_name name%TYPE := LOWER(name);
 9    init_name name%TYPE := INITCAP(name);
10  BEGIN
11    -- display inherited default values
12    DBMS_OUTPUT.PUT_LINE('name: ' || name || ' upper_name: ' || upper_name
13    || ' lower_name: ' || lower_name || ' init_name: ' || init_name);
14    -- lower_name := 'jonathan henry smithson'; invalid, character string is to
o long
15    -- lower_name := NULL; invalid, NOT NULL CONSTRAINT
16    -- debit := 50000; invalid, value out of range
17  END;
18 /
name: JoHn SmItH upper_name: JOHN SMITH lower_name: john smith init_name: John S
mith
PL/SQL procedure successfully completed.

SQL> -

```

شکل ۴

در لاین‌های 7، 8 و 9 شکل بالا، تابع‌های LOWER، UPPER و INITCAP استفاده شده‌اند. نتیجه این تابع‌ها بالای نام «جان اسمیت» تطبیق شده و این نام را به سه فارمت JOHN SMITH، JOHN SMITH و john smith تغییر داده است.

۴.۳ تبدیل کردن نوع دیتا در PL/SQL (PL/SQL Datatype Conversion)

در زبان PL/SQL گاهی ضرورت می‌شود تا یک کمیت دیتا از یک نوع به نوع دیگر تبدیل شود؛ طور مثال، یک تاریخ از سنتونی با فارمت تاریخ، گرفته شده و در یک رپور استفاده می‌شود. نوعیت دیتای گرفته شده باید از نوع تاریخ به نوع رشته‌یی از متن (کرکتر) تبدیل شود. در PL/SQL قابلیت تبدیل کردن دیتا از یک نوع به نوع دیگر موجود است. تبدیل کردن دیتا از یک به نوع دیگر در این زبان به دو شکل انجام می‌شود: به شکل آشکار (Explicit) و به شکل پوشیده (Implicit).

به خاطر اطمینان و قابلیت نگهداری بهتر از پرسه تبدیل کردن نوع دیتا، بهتر است از شکل آشکار تبدیل کردن دیتا استفاده شود. نوع دوم تبدیل کردن دیتا که به شکل پوشیده و یا ضمنی انجام می‌شود، مشکلاتی چون عدم پیش‌بینی از درست‌بودن کار و تغییرات احتمالی در قواعد استفاده از آن در نرم‌افزارهای جدید را به بار می‌آورد. مسئله دیگر در قسمت سرعت اجراست؛ سرعت تبدیل کردن پوشیده به تناسب سرعت تبدیل کردن آشکار کم‌تر است.

۴.۳.۱ تبدیل کردن آشکار (Explicit Conversion)

در این نوع، به خاطر تبدیل کردن یک نوع دیتا به نوع دیگر از تابع‌های از قبل ساخته (Built in) زبان PL/SQL استفاده می‌شود. به خاطر تبدیل کردن یک محتوای نوع CHAR به نوع تاریخ و یا نمبر از تابع‌های TO_DATE و یا TO_NUMBER استفاده می‌شود؛ به عین ترتیب به خاطر تبدیل کردن محتوای تاریخ و نمبر به فارمت کرکتر از تابع TO_CHAR استفاده می‌شود. این تابع‌ها تحت عنوان توابع Conversion در درس قبلی لیست شده‌اند.

تبدیل کردن آشکار، از اشتباهات ممکنه و نتایج غیر متوقعه جلوگیری می‌کند؛ به گونه مثال:

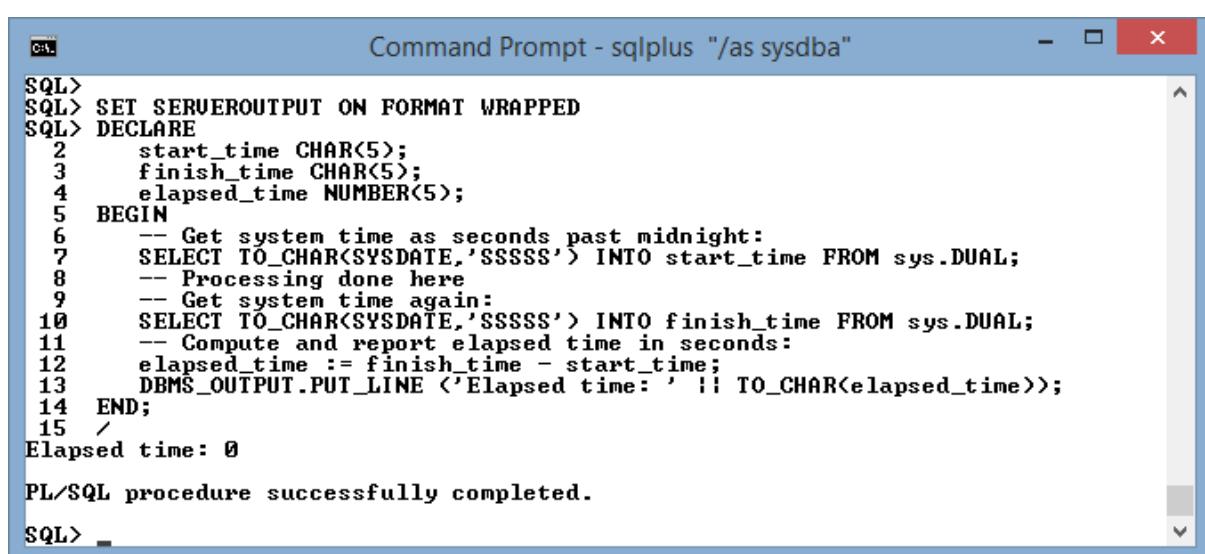
۱. در استفاده کردن از سимвول || به خاطر یک جاسازی (Concatenation) یک رشته و یک عبارت ریاضی در کد PL/SQL امکان به میان‌آمدن غلطی وجود دارد. اگر از تابع TO_CHAR در چنین مواردی به خاطر تبدیل کردن نوع دیتای ریاضی به نوع کرکتر استفاده شود، از به وجود آمدن غلطی در نتیجه پروگرام جلوگیری می‌شود.

۲. فارمت تاریخ استفاده شده در دیتابیس‌ها به خاطر معلومات ثبت شده از نوع تاریخ، یکی دیگر از مسائلی است که امکان دارد در نتیجه یک پروگرام اشتباه رخ دهد. این کار نیز با استفاده از تابع TO_CHAR به

فارمت دل خواه تبدیل و در کد اجرایی پروگرام به خاطر جلوگیری از غلطی در پروگرام استفاده می‌شود؛ به طور مثال:

```
1. DECLARE
2. start_time CHAR(5);
3. finish_time CHAR(5);
4. elapsed_time NUMBER(5);
5. BEGIN
6. -- Get system time as seconds past midnight:
7. SELECT TO_CHAR(SYSDATE,'SSSS') INTO start_time FROM sys.DUAL;
8. -- Processing done here
9. -- Get system time again:
10. SELECT TO_CHAR(SYSDATE,'SSSS') INTO finish_time FROM sys.DUAL;
11. -- Compute and report elapsed time in seconds:
12. elapsed_time := finish_time - start_time;
13. DBMS_OUTPUT.PUT_LINE ('Elapsed time: ' || TO_CHAR(elapsed_time));
14. END;
15. /
```

صفحهٔ پایین نتیجهٔ تطبیق کد را در PL/SQL نشان می‌دهد.



```
SQL> SET SERVEROUTPUT ON FORMAT WRAPPED
SQL> DECLARE
2   start_time CHAR(5);
3   finish_time CHAR(5);
4   elapsed_time NUMBER(5);
5   BEGIN
6   -- Get system time as seconds past midnight:
7   SELECT TO_CHAR(SYSDATE,'SSSS') INTO start_time FROM sys.DUAL;
8   -- Processing done here
9   -- Get system time again:
10  SELECT TO_CHAR(SYSDATE,'SSSS') INTO finish_time FROM sys.DUAL;
11  -- Compute and report elapsed time in seconds:
12  elapsed_time := finish_time - start_time;
13  DBMS_OUTPUT.PUT_LINE ('Elapsed time: ' || TO_CHAR(elapsed_time));
14  END;
15/
Elapsed time: 0
PL/SQL procedure successfully completed.
SQL> _
```

شکل ۶-۴

۴.۳.۲ تبدیل کردن پوشیده یا ضمنی (Implicit Conversion)

تبدیل کردن پوشیده معمولاً توسط خود سیستم به صورت اتومات انجام می‌شود. به همین خاطر به این نوع تبدیل کردن نوع دیتا، تبدیل کردن پوشیده و یا تبدیل کردن ضمنی می‌گویند. کمیت‌های دیتای تبدیل شده از یک نوع به نوع دیگر به نام انواع دیتای سازگار (Compatible Datatypes) یاد می‌شوند. زمانی که دو نوع دیتا با هم سازگار باشند، یکی به عوض دیگری در یک پروگرام قابل استفاده‌اند؛ طور مثال، یک قیمت عددی در یک پروگرام فرعی به عوض یک دیتای رشته‌بی (String) می‌تواند استفاده شود. در چنین حالتی، پروگرام فرعی دیتا نمبر را به حیث دیتای String می‌خواند. این عملیه به واسطه تبدیل کردن پوشیده به صورت خودکار توسط PL/SQL انجام می‌شود.

جهت وضاحت بیشتر موضوع یک مثال عملی در نظر گرفته شده است. در کد پایین متحولین از نوع کرکتر به نام‌های start_time و finish_time رشته‌هایی از دیتا را قبول می‌کند که در آن‌ها تعداد ثانیه‌های بعد از نیمة شب ثبت می‌شوند. تفاصل این دو متحول به یک متحول دیگر از نوع نمبر به نام elapsed_time ذخیره می‌شود. در اینجا محتواها از نوع کرکتر گرفته شده و به صورت اتومات با استفاده از میتوود تبدیل کردن ضمنی، PL/SQL، به شکل نمبر تبدیل شده و در متحول elapsed_time ذخیره می‌شود.

DECLARE

1. D1 DATE;
2. Cd1 VARCHAR(10);
3. Cd2 VARCHAR(10);
4. BEGIN
5. Cd1 := '15-NOV-61';
6. --now assign the srtring to a date variable. The conversion is implicit.
7. D1 := Cd1;
8. -- now assign that date variable to another string.
9. --Again the conversion is implicit, but this time the conversion is
10. -- from a date to a string.
11. Cd2 := D1;
12. -- display the two character strings to show that they are the same.
13. DBMS_OUTPUT.PUT_LINE('CD1 = ' || Cd1);
14. DBMS_OUTPUT.PUT_LINE('CD2 = ' || Cd2);
15. END;

نتیجه پروگرام فوق عبارت است از:

CD1=15-NOV-61

CD2 =15-NOV-61

۴.۳.۳ موارد استفاده از تبدیل کردن آشکار و تبدیل کردن پوشیده در PL/SQL

طوری که گفته شد، تبدیل کردن آشکار توسط پروگرام زبان PL/SQL در کد پروگرام تنظیم می‌گردد و تبدیل کردن پوشیده در زمان ضرورت توسط PL/SQL به صورت خودکار صورت می‌گیرد. در حالت دوم نیز زمینه‌سازی اجرای تبدیل شدن دیتا از یک فارمت به دیگر توسط PL/SQL شخص استفاده کننده یا همان پروگرامر برنامه صورت می‌گیرد؛ یعنی با استفاده کردن از انواع دیتای سازگار، یک پروگرامر می‌تواند PL/SQL را وارد تا دیتا را از یک نوع به نوع دیگر تبدیل کند.

طور مثال، اگر یک پروگرامر دیتا را از ستونی با یک نوع مشخص دیتا انتخاب می‌کند و آن را به متحولی از نوع دیگر دیتا Assign می‌کند، در این حالت PL/SQL به صورت خودکار نوع دیتا را به نوع دیتای متحول تبدیل می‌کند. این عملیه عبارت از تبدیل کردن پوشیده بوده که توسط PL/SQL انجام می‌شود و عامل آن شخص پروگرامر یا نویسنده کد است. جهت تصدیق این مسئله اگر محتوای یک ستون از نوع تاریخ گرفته شده و به متحولی از نوع VARCHAR2 داده شود، در این صورت دیتا به فارمت متحول (VARCHAR2) تغییر نوعیت شده و استفاده می‌شود.

در بعضی حالات تبدیل کردن ضمنی نوع دیتا، PL/SQL تفکیک کرده نمی‌تواند که کدام تبدیل را انجام دهد. در چنین مواردی پیام غلطی مربوط به Compilation توسط سیستم ارائه می‌شود و نیز در چنین شرایطی، بهترین راه حل استفاده از تبدیل نوع دیتای آشکار (Explicit Conversion) خواهد بود.

در شکل پایین یک جدولی از امکانات تبدیل کردن ضمنی در بین انواع مختلف دیتا در PL/SQL نشان داده شده است.

From:	To:									
	BLOB	CHAR	CLOB	DATE	LONG	NUMBER	PLS_INTEGER	RAW	UROWID	VARCHAR2
BLOB								Yes		
CHAR		Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
CLOB	Yes									Yes
DATE	Yes			Yes						Yes
LONG	Yes						Yes			Yes
NUMBER	Yes			Yes		Yes				Yes
PLS_INTEGER	Yes			Yes	Yes					Yes
RAW	Yes	Yes		Yes						Yes
UROWID	Yes									Yes
VARCHAR2	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes		

شکل ۷-۴ امکان تبدیل کردن ضمنی در بین انواع مختلف دیتا PL/SQL

در شکل بالا، انواع مختلف دیتای قابل استفاده در PL/SQL نشان داده شده‌اند که امکان تبدیل شدن شان توسط پروگرام به شکل پوشیده وجود دارد. فرق بین این انواع دیتا به وضاحت مشهود است. در کنار این موضوع، مسائل دیگری مربوط به انواع دیتای مشابه (از یک فامیل) با تفاوت‌های اندک است که در این جدول نشان داده نشده‌اند. این انواع دیتا نیز در صورت ضرورت توسط PL/SQL به صورت ضمنی تبدیل می‌شوند. مثال‌های این انواع دیتا در ذیل لیست شده است:

1. PLS_INTEGER and BINARY_INTEGER
2. CLOB and NCLOB
3. CHAR and NCHAR and VARCHAR and NVARCHAR2

۴.۴ اهداف استفاده از متتحولها، بلاک‌های آشیانه‌یی (Nested Blocks) و لیبل‌ها

زمانی که متتحولها در بلاک کد زبان PL/SQL استفاده می‌شوند، باید هدف استفاده کردن آن‌ها مشخص باشد. این امر باعث می‌شود تا پروگرامر به وضاحت بفهمد تا چه وقت و در کجا، کدام متتحول را چطور استفاده کند؛ به عین شکل، دانستن هدف استفاده از یک متتحول، پروگرامر را کمک می‌کند تا غلطی‌های پروگرام را پیدا کرده و اصلاح کند. متتحولها در قسمت اعلامیه (Declaration) یک بلاک کد PL/SQL تعریف می‌شوند.

۴.۴.۱ هدف استفاده از متتحولها

متتحول‌هایی که در اعلامیه یک بلاک کد PL/SQL تعریف می‌گردند، به شکل محلی از آن‌ها استفاده شده و محدود به همان بلاک کد می‌باشند. طوری که در فصول اولیه کتاب توضیح گردیده است، یک بلاک کد زبان PL/SQL متشکل از سه قسمت است: قسمت اعلامیه، قسمت اجرایی کد و قسمت استثناهای (Exceptions) عبارت از قسمتهای اساسی حتمی و غیر حتمی (اختیاری) بلاک کد است. قسمت حتمی عبارت از قسمت وسطی بلاک یا همان دستورهای اجرایی PL/SQL است.

هدف استفاده از متتحولها عبارت از همان بلاک کد زبان PL/SQL است که در آن متتحول‌های تعریف شده قابل شناسایی‌اند. دوره‌یی که یک متتحول تعریف شده در بلاک کد شناخته می‌شود، از تعریف متتحول در اعلامیه آغاز شده و الی ختم بلاک کد ادامه پیدا می‌کند. زمانی که بلاک کد تکمیل شد، متتحول‌های مربوطه آن نیز غیر فعال می‌شوند.

۴.۴.۲ بلاک‌های آشیانه‌یی (Labels) و لیبل‌ها (Nested Blocks)

یک بلاک کد زبان PL/SQL می‌تواند با یک لیبل نشانی (نام‌گذاری) شود. این کار به خاطر وضاحتدادن و ارتباطدادن بلاک کد به یک موضوع کمک می‌کند. نام بلاک (لیبل بلاک) باید قبل از اولین دستور اجرایی همان بلاک اضافه شود؛ یعنی می‌شود قبل از BEGIN و یا حتی قبل از DECLARE طور ذیل نوشته شود:

```
<< find_stu_num >>
```

```
BEGIN
```

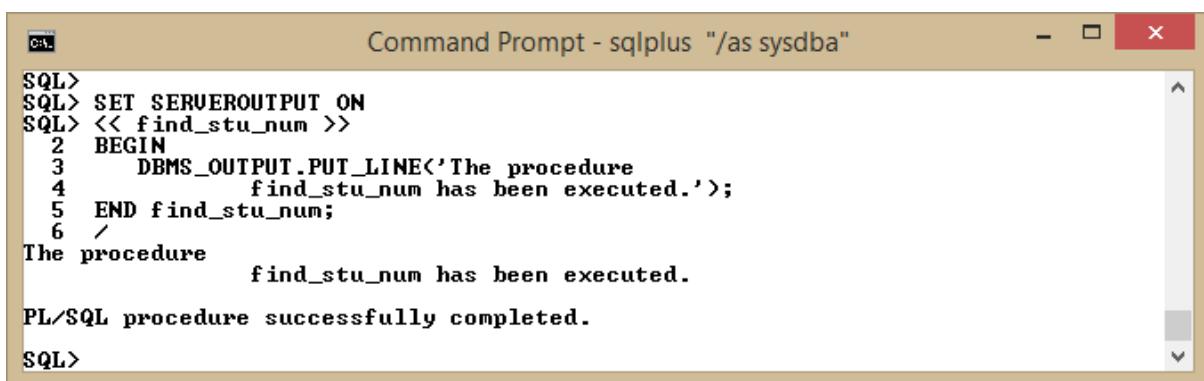
```
    DBMS_OUTPUT.PUT_LINE('The procedure
```

```
        find_stu_num has been executed.');
```

```
END find_stu_num;
```

```
/
```

در بلاک کد بالا، لیبل به نام find_stu_name در قسمت قبل از دستور اجرایی BEGIN با استفاده از سمبول مجاز آن نوشته شده است. در آخر دستورهای اجرایی، یعنی بعد از کلمه END نیز به صورت اختیاری بدون استفاده از سمبول همان لیبل می‌تواند نوشته شود. شکل پایین، اجرای موفقانه کد فوق را نشان می‌دهد.



```
SQL> SET SERVEROUTPUT ON
SQL> << find_stu_num >>
2  BEGIN
3      DBMS_OUTPUT.PUT_LINE('The procedure
4          find_stu_num has been executed.');
5  END find_stu_num;
6 /
The procedure
        find_stu_num has been executed.

PL/SQL procedure successfully completed.

SQL>
```

شکل ۸-۴

بلاک‌های کد زبان PL/SQL می‌توانند یکی در بین دیگری تنظیم شوند. قسمت اساسی یک بلاک کد زبان PL/SQL با کلمه BEGIN آغاز شده و با کلمه END پایان پیدا می‌کند. تمام دستورهای اجرایی یک بلاک در همین محدوده نوشته می‌شوند. زمانی که مسأله بلاک‌های آشیانه‌یی (Nested Blocks) به میان می‌آید، یک یا چند بلاک کد باید به صورت مکمل (آغاز و انجام آن) در داخل بلاک دیگر نوشته شده باشند. بلاک‌های آشیانه‌یی عبارت از بلاک بیرونی و بلاک و یا بلاک‌های داخلی می‌باشند. شکل عمومی Nested Blocks طور ذیل است:

```

BEGIN      -- outer block

BEGIN      -- inner block

...;

END;      -- end of inner block

END;      -- end of outer block

```

بلاک‌های آشیانه‌یی، بالای هدف استفاده و دورهٔ حیات متحول‌ها تأثیر مستقیم دارد. یک متحول زمانی که در یک بلاک داخلی تعریف می‌شود، استفاده آن محدود به همان بلاک داخلی است؛ یعنی هر بلاک داخلی که در یک بلاک بیرونی تنظیم شده باشد، دارای استقلال فردی خود است. جهت وضاحت موضوع یک مثال در نظر گرفته شده و طور ذیل است:

```

<< outer_block >>

DECLARE
    v_test NUMBER := 123;
BEGIN
    DBMS_OUTPUT.PUT_LINE
        ('Outer Block, v_test: '|| v_test);
<< inner_block >>
DECLARE
    v_test NUMBER := 456;
BEGIN
    DBMS_OUTPUT.PUT_LINE
        ('Inner Block, v_test: '|| v_test);
    DBMS_OUTPUT.PUT_LINE
        ('Inner Block, outer_block.v_test: '||
            Outer_block.v_test);
END inner_block;
END outer_block;
/

```

نتیجهٔ تطبیق موفقانه کد بالا در شکل ذیل نشان داده شده است.

```
SQL> SET SERVEROUTPUT ON
SQL> << outer_block >>
2  DECLARE
3      v_test NUMBER := 123;
4  BEGIN
5      DBMS_OUTPUT.PUT_LINE
6          ('Outer Block, v_test: '||v_test);
7  << inner_block >>
8  DECLARE
9      v_test NUMBER := 456;
10 BEGIN
11     DBMS_OUTPUT.PUT_LINE
12         ('Inner Block, v_test: '||v_test);
13     DBMS_OUTPUT.PUT_LINE
14         ('Inner Block, outer_block.v_test: ' ||
15             Outer_block.v_test);
16 END inner_block;
17 END outer_block;
18 /
Outer Block, v_test: 123
Inner Block, v_test: 456
Inner Block, outer_block.v_test: 123
PL/SQL procedure successfully completed.
SQL>
```

شکل ۹-۴

طوری که در نتیجهٔ دیده می‌شود (بعد از سطر 18 در شکل بالا)، متتحول به نام v_test در بلاک بیرونی قیمت 123 را نشان می‌دهد. متتحول دیگری با همین نام در بلاک داخلی قیمت 456 را نشان می‌دهد. از این نتیجه، مستقل بودن بلاک داخلی به صورت واضح هویباً گردیده که در آن‌ها به عین نام متتحول‌ها تعریف شده و هر متتحول در بلاک کد مربوطه به صورت مستقل قیمت خود را نگه داشته است.

۴.۵ موارد استفاده از ترتیب‌ها (Sequences) با عبارات PL/SQL

ترتیب‌ها یکی دیگر از ساختمان‌هایی‌اند که به خاطر مدیریت دیتا در PL/SQL استفاده می‌شوند. با استفاده از ترتیب‌ها، سلسله‌هایی از یوزر دیتا تهیه شده و به شکل خودکار می‌توانند به دیتابیس داخل شوند. قبل از اوریکل 11g، شبه ستون‌هایی (Pseudocolumns) به نام‌های CURRVAL و NEXTVAL در PL/SQL تنها از طریق کیوری‌ها قابل دسترسی بودند. در نمونه‌های جدید اوریکل (11g و 12c) این شبه ستون‌ها از طریق عبارات سیکویل نیز قابل دسترس شده‌اند. این تغییر بر علاوهٔ انکشاف کد زبان PL/SQL، قابلیت اجراء و اندازه‌گیری زمان اجراء (Runtime) را نیز بیشتر ساخته است.

شبه ستون‌های CURRVAL و NEXTVAL به خاطر ایجاد کردن ترتیب‌ها استفاده می‌شوند. توسط این شبه ستون‌ها، نمبرهای متوالی ایجاد و به صورت خودکار به شکل ریکورد‌های دیتا به جدول ذخیره می‌شوند. عدد اولی ترتیب در CURRVAL تعیین شده و عدد بعدی (قابل افزایش) در NEXTVAL اضافه می‌شود. نحوهٔ استفاده کردن از شبه ستون‌ها در یک ترتیب (Sequence) طور ذیل نوشته می‌شود:

sequence_name.CURRVAL

sequence_name.NEXTVAL

جهت وضاحت موضوع مثال پایین در نظر گرفته شده است.

```
CREATE SEQUENCE test_seq START WITH 1 INCREMENT BY 1;
```

```
SET SERVEROUTPUT ON
```

```
DECLARE
```

```
    v_seq_value NUMBER;
```

```
BEGIN
```

```
    v_seq_value := test_seq.NEXTVAL;
```

```
    DBMS_OUTPUT.PUT_LINE ('v_seq_value: '||v_seq_value);
```

```
END;
```

```
/
```

نتیجه اجرای موفقانه مثال بالا، در شکل زیر نشان داده شده است.

```
SQL> CREATE SEQUENCE test_seq START WITH 1 INCREMENT BY 1;
Sequence created.

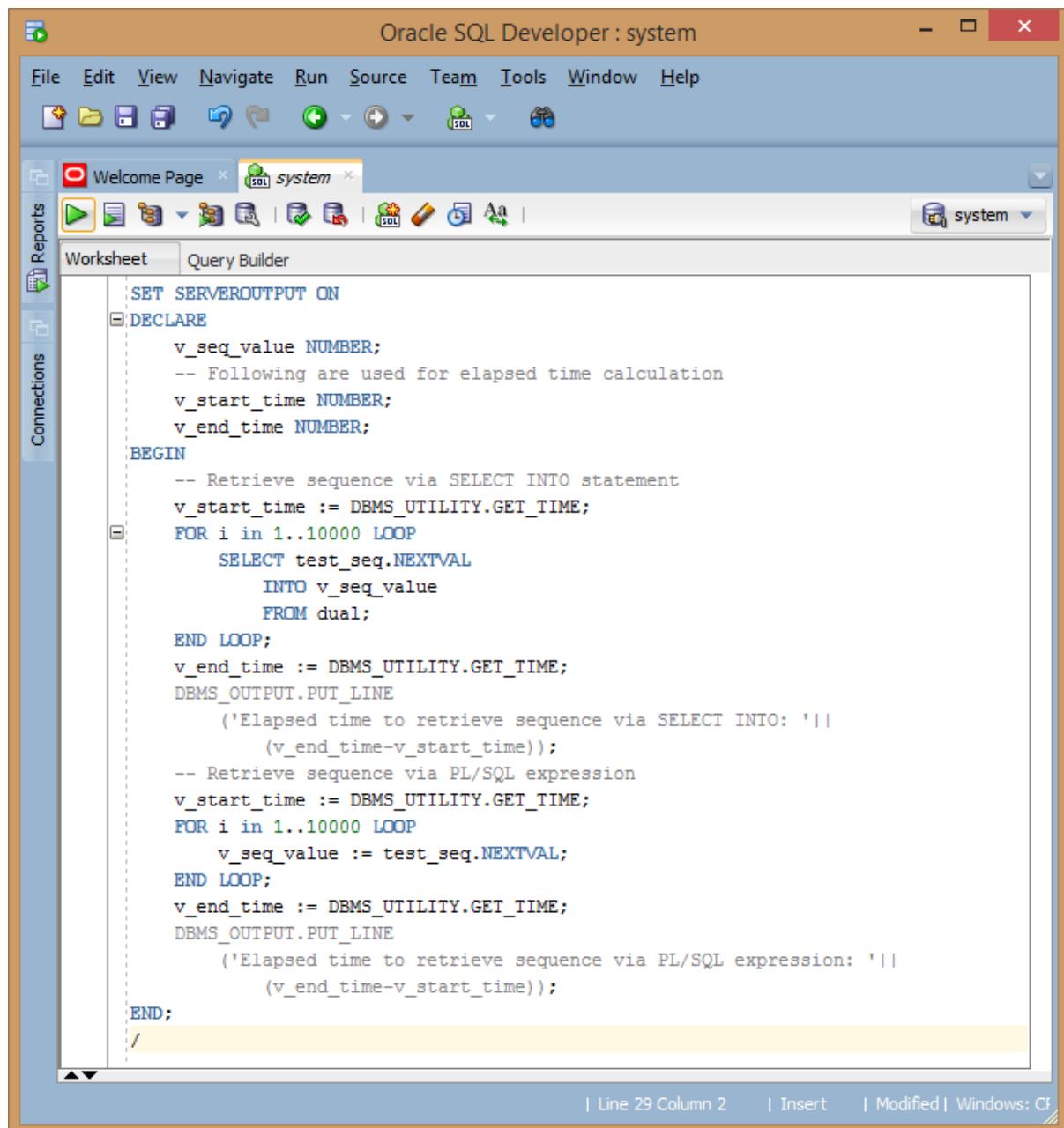
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2     v_seq_value NUMBER;
  3 BEGIN
  4     v_seq_value := test_seq.NEXTVAL;
  5     DBMS_OUTPUT.PUT_LINE ('v_seq_value: '||v_seq_value);
  6 END;
  7 /
v_seq_value: 1
PL/SQL procedure successfully completed.

SQL> _
```

شكل ۱۰-۴

اولین دستور در شکل بالا، ترتیب را به نام test-seq ایجاد کرده است. در دستورهای اجرایی بعد از همان ترتیب ایجادشده با شبه ستون NEXTVAL اولین قیمت را گرفته است که بعد از سطر 7 نتیجه دستور خروجی (سطر 5) نشان داده شده است.

یک مثال دیگر در عین موضوع با تفصیل بیشتر در نظر گرفته شده است.



The screenshot shows the Oracle SQL Developer interface with a connection to the 'system' database. The 'Worksheet' tab is active, displaying the following PL/SQL code:

```
SET SERVEROUTPUT ON
DECLARE
    v_seq_value NUMBER;
    -- Following are used for elapsed time calculation
    v_start_time NUMBER;
    v_end_time NUMBER;
BEGIN
    -- Retrieve sequence via SELECT INTO statement
    v_start_time := DBMS_UTILITY.GET_TIME;
    FOR i in 1..10000 LOOP
        SELECT test_seq.NEXTVAL
        INTO v_seq_value
        FROM dual;
    END LOOP;
    v_end_time := DBMS_UTILITY.GET_TIME;
    DBMS_OUTPUT.PUT_LINE
        ('Elapsed time to retrieve sequence via SELECT INTO: ' ||
         (v_end_time-v_start_time));
    -- Retrieve sequence via PL/SQL expression
    v_start_time := DBMS_UTILITY.GET_TIME;
    FOR i in 1..10000 LOOP
        v_seq_value := test_seq.NEXTVAL;
    END LOOP;
    v_end_time := DBMS_UTILITY.GET_TIME;
    DBMS_OUTPUT.PUT_LINE
        ('Elapsed time to retrieve sequence via PL/SQL expression: ' ||
         (v_end_time-v_start_time));
END;
/
```

The code compares the elapsed time for retrieving a sequence value using a `SELECT INTO` statement versus a PL/SQL expression. It uses `DBMS_UTILITY.GET_TIME` to measure time and `DBMS_OUTPUT.PUT_LINE` to print results.

شکل ۱۱-۴

نتیجه اجرای کد فوق در سیکویل پلس طور ذیل نشان داده شده است:

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      v_seq_value NUMBER;
 3      -- Following are used for elapsed time calculation
 4      v_start_time NUMBER;
 5      v_end_time NUMBER;
 6  BEGIN
 7      -- Retrieve sequence via SELECT INTO statement
 8      v_start_time := DBMS_UTILITY.GET_TIME;
 9      FOR i in 1..10000 LOOP
10          SELECT test_seq.NEXTVAL
11              INTO v_seq_value
12             FROM dual;
13      END LOOP;
14      v_end_time := DBMS_UTILITY.GET_TIME;
15      DBMS_OUTPUT.PUT_LINE
16          ('Elapsed time to retrieve sequence via SELECT INTO: ' ||
17           (v_end_time-v_start_time));
18      -- Retrieve sequence via PL/SQL expression
19      v_start_time := DBMS_UTILITY.GET_TIME;
20      FOR i in 1..10000 LOOP
21          v_seq_value := test_seq.NEXTVAL;
22      END LOOP;
23      v_end_time := DBMS_UTILITY.GET_TIME;
24      DBMS_OUTPUT.PUT_LINE
25          ('Elapsed time to retrieve sequence via PL/SQL expression: ' ||
26           (v_end_time-v_start_time));
27  END;
28 /
Elapsed time to retrieve sequence via SELECT INTO: 19
Elapsed time to retrieve sequence via PL/SQL expression: 11
PL/SQL procedure successfully completed.
```

شکل ۱۲-۴

به خاطر SELECT INTO در مقایسه با کیوری انتخاب از نوع PL/SQL در مثال آخر، مزیت استفاده از داکس دین دیتا به جدول نشان داده شده است. در حالت استفاده از کیوری به خاطر داکس دین دیتا 19 ثانیه زمان را در بر گرفته است؛ در حالی که با استفاده از ترتیب زمان کوتاه‌تر (11 ثانیه) را خاطر انجام عین کار استفاده کرده است.



خلاصه فصل چهارم

در این فصل مسائل مختلف مربوط به قواعد نوشتاری دستورهای زبان PL/SQL تشریح شده‌اند. در بخش اول روی اهداف و طرز استفاده از Lexical Unit‌ها با تفصیل در مورد شناسه‌ها، کلمه‌های ریزرفشده، حائل‌ها، لیترال‌ها و تفسیرها در PL/SQL پرداخته شده است. توابع از قبل ساخته‌شده در PL/SQL و موارد استفاده از آن‌ها تشریح شده و لیست مکمل این نوع تابع‌ها در هفت دسته نشان داده شده است. یک مثال استفاده از تابع‌ها نیز در آخر بحث به صورت عملی توضیح گردیده است. تبدیل کردن نوع دیتا به اشکال آشکار و پوشیده توسط کد PL/SQL و موارد استفاده از آن‌ها با ذکر مثال‌ها توضیح شده‌اند.

اهداف استفاده از متتحول‌ها، بلاک‌های آشیانه‌بی و لیبل‌ها از موضوعات دیگر شامل در این فصل‌اند. در این قسمت هدف استفاده از متتحول‌ها و دوره حیات آن‌ها در بلاک‌های کد PL/SQL به بحث گرفته شده‌اند. ضرورت استفاده از بلاک‌های آشیانه‌بی و مدیریت متتحول‌ها در بلاک‌های آشیانه‌بی با مثال‌ها و دلایل موجه توضیح داده شده‌اند. در انتهای فصل به موارد استفاده از ترتیب‌ها و مؤثریت آن‌ها با ذکر مثال‌های عملی پرداخته شده است.



سوالات و فعالیت های فصل چهارم

۱. اصطلاح Lexical Units را در قبال زبان PL/SQL توضیح دهید.
۲. بر علاوه حروف و اعداد، سمبل های مجاز در نام شناسه های PL/SQL کدام هایند؟
۳. با استفاده از کدام دستور PL/SQL لیست کلمه های ریزرف شده در سیکویل پلس نشان داده می شود؟
۴. حائل ها (Delimiters) در زبان PL/SQL به چه منظور استفاده می شوند؟ با مثال واضح سازید.
۵. لیترال ها (Literals) در PL/SQL چه مفهومی را افاده می کنند؟
۶. لیترال ها در زبان PL/SQL به کدام شکل هایند؟ صرف نام بگیرید.
۷. انواع لیترال های عددی در PL/SQL را نام بگیرید.
۸. حد اقل و حد اکثر طول لیترال های کرکتر را مشخص کنید.
۹. آیا استفاده از حروف کوچک و بزرگ در لیترال ها فرق می کند و یا خیر؟ با مثال واضح سازید.
۱۰. تفسیرها (Comments) در زبان PL/SQL به چه منظور استفاده می شوند؟ واضح سازید.
۱۱. انواع تفسیرها در PL/SQL را نام گرفته و با مثال واضح سازید.
۱۲. اختیار SET SERVEROUTPUT ON در کدام قسمت کد پروگرام PL/SQL به چه مفهوم بوده و معمولاً در کدام پروگرام استفاده می شود؟

۱۳. دسته‌های توابع از قبل ساخته شده در PL/SQL را نام بگیرید.
۱۴. تبدیل کردن نوع دیتا به شکل آشکار (Explicit Conversion) در PL/SQL چگونه صورت می‌گیرد.
۱۵. تبدیل کردن نوع دیتا به شکل پوشیده (Implicit Conversion) در PL/SQL چگونه صورت می‌گیرد.
۱۶. جمله «متحولین زبان PL/SQL محدود به یک بلاک‌اند.» را به زبان خود توضیح دهید.
۱۷. یک بلاک دستورهای PL/SQL را با استفاده از لیبل، نام‌گذاری کرده و بنویسید.
۱۸. بلاک‌های آشیانه‌یی (Nested Blocks) در PL/SQL را توضیح داده و شکل عمومی آن‌ها را بنویسید.
۱۹. ترتیب‌ها (Sequences) را در PL/SQL توضیح دهید.
۲۰. اصطلاحات CURRVAL و NEXTVAL را در قبال ترتیب‌ها توضیح دهید.

فعالیت‌ها

۱. با استفاده از دستور "sqlplus / as sysdba" برنامه سیکویل پلس را از کوماند پرامپت کمپیوترتان به راه بیندازید.
۲. در یک مثال پروگرام PL/SQL با استفاده از DECLARE سه متحول را از نوع تاریخ/زمان با فارمتهای مختلف تعریف کنید.
۳. یک بلاک کد SQL/PL را بنویسید که در آن یک عملیه ساده ریاضی انجام شده و از هر دو نوع تفسیر استفاده شده و نتیجه خروجی داشته باشد. (به خاطر داشتن خروجی دستور SET SERVEROUTPUT ON را در آغاز پروگرام اضافه کنید.)
۴. یک پروگرام ساده PL/SQL را بنویسید که در آن تابع‌های UPPER و LOWER مربوط کتگوری کرکتر، محتوای یک متحول را گرفته و با حروف بزرگ و حروف کوچک در سطرهای جداگانه نشان بدهد.
۵. با استفاده از بلاک‌های آشیانه‌یی (Nested Blocks) یک مثال ساده PL/SQL را کار کنید.

فصل پنجم

سرور دیتابیس اوریکل (Oracle Database Server)



هدف کلی: شاگردان با روش کار سرور دیتابیس اوریکل آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. استفاده از دستورهای کار با دیتا (DML) را در بلاک PL/SQL بیاموزند.
۲. استفاده از کنترول ترانزکشن را در بلاک PL/SQL بیاموزند.
۳. استفاده از عبارت INTO را در بلاک PL/SQL بیاموزند.
۴. کرسوها در سیکویل و خصوصیات آنها را شرح دهند.
۵. فرق بین کرسوها اشکار (Explicit Cursors) و کرسوها ضمنی (Implicit Cursors) را شرح دهند.

در کارکردن با دیتابیس‌ها در سیستم مدیریت دیتابیس اوریکل امکانات زیادی برای استفاده‌کنندگان جهت تنظیم بهتر دیتا فراهم است. دیتابیس اوریکل یکی از سیستم‌های قوی و شناخته‌شده در ساحة دیتابیس‌ها بوده و با توجه به صفت منبع باز (Open Source) بودن، این سیستم در ساحت تعليمی و تجاری به پیمانه زیاد استفاده می‌شود. در این فصل بیشتر به مسائل کار با یوزر دیتا پرداخته شده است. دستورهای اجرایی در دیتابیس‌ها به سه دسته DML، DDL و DCL تقسیم می‌شوند. تفسیر نام‌های یادشده در جریان فصل تشریح شده‌اند. در این فصل روی مدیریت نوع DML که اختصار کلمه‌های Data Manipulation Language است، بحث‌هایی صورت گرفته است.

عنوانی که در فصل پنجم کتاب توضیح داده شده‌اند، به ترتیب موضوعاتی از قبیل دستورهای کار با دیتا (DML)، کنترول ترانزکشن و استفاده از عبارت INTO به خاطر قیمت‌دادن به متحولین در بلاک‌های زبان PL/SQL شامل می‌شوند. کرسرهای منحیث منابع ذخیره موقت دیتا و استفاده از آن دیتا در PL/SQL از عنوانی دیگر این فصل‌اند. توضیحات لازم در قسمت انواع کرسرهای طریقه‌های کار آن‌ها در سیکویل شامل معلومات ارائه شده این فصل کتاب‌اند. موضوعاتی که شکل تطبیقی دارند، به خاطر وضاحت بیشتر با استفاده از مثال‌های عملی توضیح داده شده‌اند.

۵.۱ دستورهای کار با دیتا (DML) در بلک

دستورهای کار با دیتا در سیکویل به نام Data Manipulation Language یاد می‌شود که آن را به اختصار DML می‌گویند. دستورهای DML با یوزر دیتا کار می‌کنند. دیتابیس‌های موجود توسط دستورهای DML به خاطر کارکردن با این نوع دیتا استفاده می‌شوند. دسته‌های دیگر دستورهای سیکویل به نام‌های DCL^۱ و DDL^۲ یاد می‌شوند. دستورهای DDL در سیکویل به خاطر کارکردن با میتادیتا یا دیتا ساختمان‌های دیتابیس؛ مانند ایجاد کردن دیتابیس‌ها، جدول‌ها و ساختمان‌های دیگر استفاده می‌شوند. دسته سوم دستورهای سیکویل که به نام DCL یاد می‌شود، به خاطر مدیریت و کنترول دیتابیس‌ها، استفاده‌کنندگان و غیره موارد استفاده می‌شوند. در این بحث، موضوع مربوط دستورهای DML در نظر گرفته شده است.

دستورهای کار با دیتا، استفاده‌کنندگان و پروگرامهای زبان PL/SQL را قادر می‌سازد تا دیتای موجود یک دیتابیس را کیویری کرده و در آن تغییراتی وارد کنند. دستورهای DML بیشترین استفاده را در دیتابیس‌ها دارند. توسط این دستورها کارهای ذیل انجام داده می‌شود:

^۱ اختصار کلمات Data Definition Language بوده و دستورهای تعریف دیتا در سیکویل را شامل می‌شود.

^۲ اختصار کلمات Data Control Language بوده و دستورهای کنترول دیتابیس‌ها و استفاده‌کنندگان دیتابیس را در بر دارد.

۱. کیوری انتخاب (Select): دیدن، ترکیب کردن و گرفتن دیتا از یک یا چند جدول و یا نما (View) توسط این دستورها انجام می‌شود. دیتای گرفته شده امکان دارد از یک دیتابیس باشد و یا هم از چندین دیتابیس مختلف استفاده شود.

۲. داخل کردن دیتا (Insert): اضافه کردن ریکوردهای جدید دیتا به دیتابیس‌ها از طریق جدول‌ها و یا نماها با استفاده از دستور داخل کردن (Insert) صورت می‌گیرد. عناصر دیتای جدید می‌شود توسط لیست محتوای ستون‌ها نوشته شوند و یا هم با استفاده از کیوری انتخاب، عناصر دیتا از موقعیت‌های دیگر خوانده شده و به جدول‌های دیتابیس داخل می‌گردند.

۳. بهروزکردن دیتا (Update): یکی دیگر از کارهایی که با دیتا انجام می‌شود، عبارت از اپدیت کردن یوزر دیتا است. با استفاده از دستور UPDATE در بلک کد زبان PL/SQL محتواهای موجود در یک جدول از طریق جدول و یا نمای دیگر به روز می‌شوند؛ به عین ترتیب با استفاده از دستور MERGE ریکوردهای دیتا از طریق یک جدول و یا نمای دیگر به جدول داخل و یا اپدیت می‌شود.

۴. پاک کردن دیتا (Delete): یکی دیگر از دستورهای DML عبارت از دستور پاک کردن دیتا است. این دستور نیز با دیتای استفاده کنندگان کار کرده و توسط آن ریکوردهای دیتا از طریق جدول‌ها و نماها پاک می‌شوند؛ البته پاک کردن دیتا با استفاده از این دستور می‌تواند با شرایط کیوری‌ها عیار شده و تنها ریکوردهای مورد نظر از دیتابیس پاک شوند. مسئله آخر در تمام دستورهای کار کردن با دیتا (DML) قابل تطبیق بوده و کار با دیتا توسط آن‌ها به وجه بهتر کنترول می‌شود.

مثال‌های پایین، کد زبان PL/SQL را نشان می‌دهد که در آن از دستورهای کار با دیتا (DML) استفاده شده است.

BEGIN

BEGIN -- An inner nested Block starts here.

SELECT * FROM employee;

-- Shows all records from a table named employee.

END; -- The first nested Block ended here.

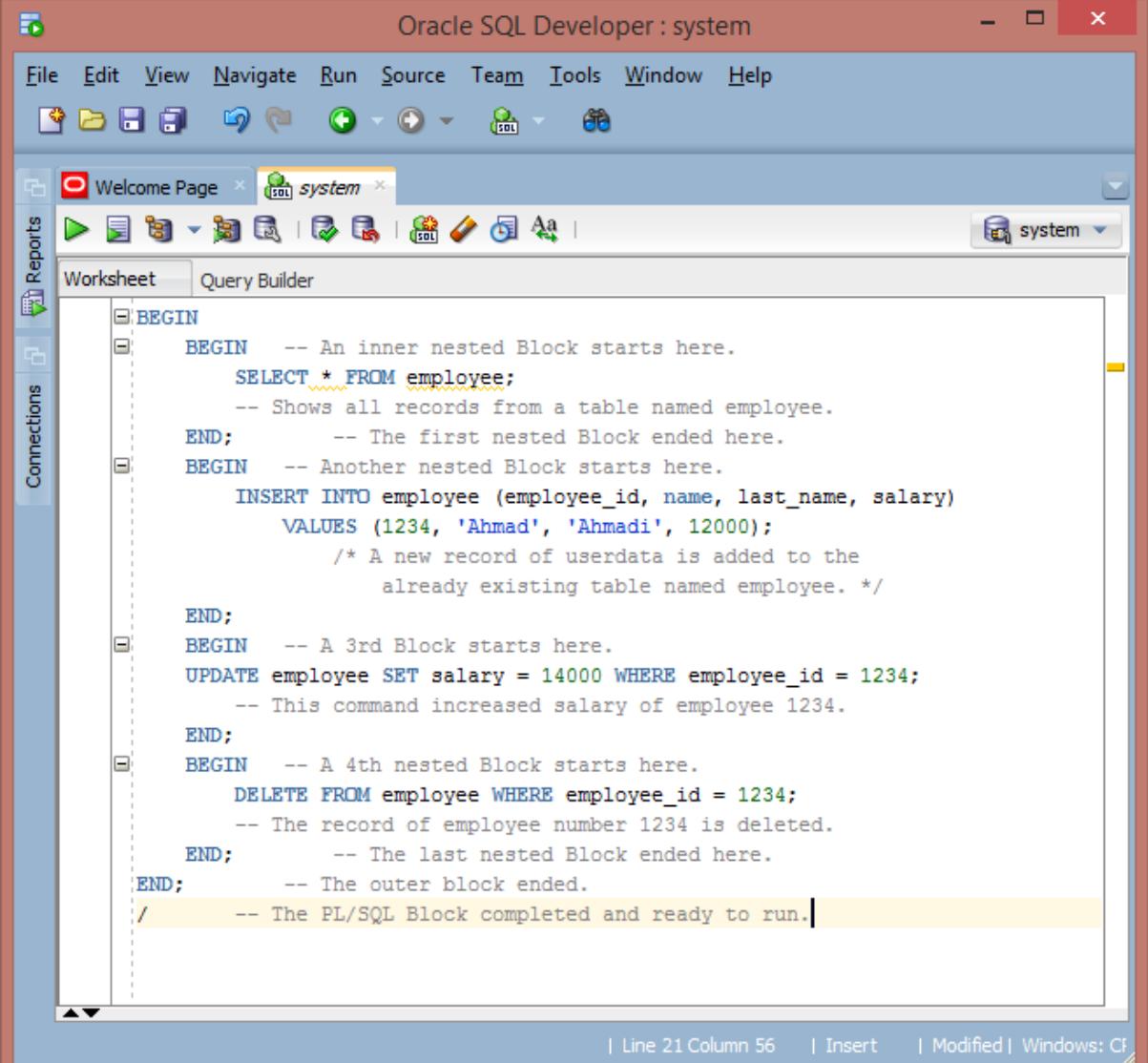
BEGIN -- Another nested Block starts here.

INSERT INTO employee (employee_id, name, last_name, salary)

VALUES (1234, 'Ahmad', 'Ahmadi', 12000);

```
/* A new record of userdata is added to the already existing table named  
employee */  
  
END;  
  
BEGIN -- A 3rd Block starts here.  
  
    UPDATE employee SET salary = 14000 WHERE employee_id = 1234;  
  
    -- This command increased salary of employee 1234.  
  
END;  
  
BEGIN -- A 4th nested Block starts here.  
  
    DELETE FROM employee WHERE employee_id = 1234;  
  
    -- The record of employee number 1234 is deleted.  
  
END; -- The last nested Block ended here.  
  
END; -- The outer block ended.  
  
/ -- The PL/SQL Block completed and ready to run.
```

در شکل پایین، کد رنگ‌آمیزی شده مثال بالا در محیط توسعه‌دهنده سیکویل نشان داده شده است.



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. The toolbar has various icons for file operations like Open, Save, and Run. The Connections panel on the left shows a connection named "system". The main area is the Worksheet tab, which contains the following PL/SQL code:

```
BEGIN
  BEGIN -- An inner nested Block starts here.
    SELECT * FROM employee;
    -- Shows all records from a table named employee.
  END; -- The first nested Block ended here.
  BEGIN -- Another nested Block starts here.
    INSERT INTO employee (employee_id, name, last_name, salary)
      VALUES (1234, 'Ahmad', 'Ahmadi', 12000);
      /* A new record of userdata is added to the
         already existing table named employee. */
  END;
  BEGIN -- A 3rd Block starts here.
    UPDATE employee SET salary = 14000 WHERE employee_id = 1234;
    -- This command increased salary of employee 1234.
  END;
  BEGIN -- A 4th nested Block starts here.
    DELETE FROM employee WHERE employee_id = 1234;
    -- The record of employee number 1234 is deleted.
  END; -- The last nested Block ended here.
END; -- The outer block ended.
/ -- The PL/SQL Block completed and ready to run.
```

The status bar at the bottom indicates "Line 21 Column 56" and "Modified".

شکل ۱-۵

۵.۲ کنترول ترانزکشن (Transaction) در بلاک PL/SQL

یک مجموعه از دستورهای کار با دیتا (DML) که روی یک موضوع استفاده شده باشند، به نام ترانزکشن (Transaction) یاد می‌شوند. در مثال درس قبلی، دستورهای داخل کردن دیتا، بروزکردن دیتا و پاک کردن دیتای یک ریکورد مربوط کارمندی با شماره 1234 می‌شود که می‌توان آن را به نام ترانزکشن یاد کرد. ترانزکشن‌ها کنترول کارکردن با یوزر دیتا را به استفاده کننده می‌دهد. در مثال درس قبلی بعد از هر دستور اجرایی و یا هم بعد از چند دستور اجرایی، نتایج حاصله تغییرات دیتا توسط پروگرام قابل کنترول است؛ یعنی زمانی که یک ریکورد دیتا اپدیت شد، با استفاده از دستور رد (Rollback) در ترانزکشن دیتا می‌توان به شکل قبل از اجرای دستور برگشت و یا هم تغییرات واردہ با استفاده از دستور تأیید (Commit) دائمی گردند.

به خاطر مرتبط بودن موضوع، یک بخش معلومات راجع به ترانزکشن‌ها از کتاب مضمون مدیریت دیتا، که نوشته نگارنده است، گرفته شده و در ذیل ارائه می‌گردد:

یک ترانزکشن عبارت از قطعه‌یی از کار روی دیتا است که این قطعه کار یا باید به صورت کامل انجام شود و یا هم باید به صورت کامل از تطبیق آن جلوگیری صورت گرفته و دیتابیس به حالت اولی خود گذاشته شود؛ یعنی دسته‌یی از دستورها که در یک ترانزکشن دسته‌بندی شده باشند، نباید به طور قسمی اجراء شوند، بلکه مجموعه‌یی دستورها باید به شکل یک قطعه مکمل مدیریت شود. به خاطر اهداف بازیابی (Recovery)، سیستم مدیریت دیتابیس ضرورت به حفظ مسیرهای کار روی دیتا دارد. یک سیستم، جزئیات کار روی دیتا توسط استفاده‌کنندگان را ذخیره می‌کند. در زمان تنظیم و اجرای ترانزکشن، مسائلی که توسط سیستم ذخیره می‌شوند، عبارت از آغاز ترانزکشن و پایان ترانزکشن شامل تطبیق موقافنه ترانزکشن و یا کنسل‌شدن ترانزکشن است؛ پس گفته می‌شود که در بخش مدیریت بازیابی سیستم، مدیریت دیتابیس (DBMS) ضرورت به ثبت عملیات زیر است:

- **آغاز ترانزکشن**: این دستور آغاز کننده ترانزکشن بوده و با دستور (Start Transaction) اجراء می‌شود.
- **خواندن و یا نوشتمن**: عملیات خواندن و نوشتمن دیتا (کیویری کردن، داخل کردن، بهروزکردن و پاک کردن دیتا) در دیتابیس اجراء می‌شود. این دستورها هدف اساسی بخش اجرایی یک ترانزکشن را در بر دارند.

• **ختم ترانزکشن**: دستورهای پایانی ترانزکشن در این کتگوری اضافه می‌شوند. دستورهای پایانی در حقیقت پایان دستورهای اصلی خواندن و نوشتمن دیتا را نشان می‌دهند. در همین قسمت تصمیم موفقیت و عدم موفقیت یک ترانزکشن گرفته می‌شود؛ یعنی نتیجه اجرای دستورها (تغییرات دیتا) در دیتابیس به صورت دائمی تطبیق شود و یا از تغییرات دیتا جلوگیری به عمل آید. در حالت اول تغییرات دائمی وقتی در دیتا اضافه شوند، این تغییرات قابل برگشت (Undo) نمی‌باشند. حالت‌های تأیید و رد تغییرات دیتا که به اساس ترانزکشن باید واقع شوند، توسط دستورهای پایانی Commit و Rollback طور ذیل به اجراء گذاشته می‌شوند:

- **تسليم شده (Commit)**: در صورت تأیید کارهای اجراء شده در ترانزکشن، استفاده‌کننده می‌تواند نتیجه ترانزکشن را تأیید کرده و با استفاده از کلمه Commit تغییرات را دائمی و غیر قابل برگشت بسازد.
- **رد شده (Rollback or Abort)**: استفاده از این دستور در صورتی است که کدام مشکلی در قسمت تطبیق ترانزکشن باشد. استفاده‌کننده و یا در حالاتی سیستم، این را تشخیص می‌دهد که تغییرات نباید اجازه داده شوند. در صورت رد شدن نتیجه ترانزکشن، تمام تغییرات خواندن و نوشتمن دیتابیس لغو (Concel) شده و دیتابیس حالت قبل از آغاز ترانزکشن را به خود می‌گیرد. اهمیت استفاده از ترانزکشن‌ها در مدیریت دیتابیس‌ها زیاد است. به خاطر دانستن این مسئله، مثالی در نظر گرفته شده است. در این مثال، انتقال پول از یک حساب بانکی به حساب دیگر اجراء شده است. با

استفاده از دستور تکمیلی ترازنزکشن، در آن واحد، پول انتقال داده شده و از سوءاستفاده جلوگیری صورت می‌گیرد؛ یعنی اگر در منفی کردن پول از یک حساب و جمع کردن آن در حساب دیگر سکته‌گی وارد می‌شود، امکان استفاده کردن نادرست در همچون موارد محسوس است؛ به مثال توجه شود:

```
CREATE TABLE accounts (account_id NUMBER(6), balance NUMBER (10,2));
INSERT INTO accounts VALUES (7715, 6350.00);
INSERT INTO accounts VALUES (7720, 5100.50);
DECLARE
    transfer NUMBER(8,2) := 250;
BEGIN
    UPDATE accounts SET balance = balance - transfer WHERE account_id = 7715;
    UPDATE accounts SET balance = balance + transfer WHERE account_id = 7720;
    COMMIT COMMENT 'Transfer From 7715 to 7720' WRITE IMMEDIATE NOWAIT;
END;
/
```

نتیجه تطبیق دستورهای بالا و بلاک کد PL/SQL در شکل ذیل نشان داده شده است:

```
Command Prompt - sqlplus "/ as sysdba"
SQL> CREATE TABLE accounts (account_id NUMBER<6>, balance NUMBER <10,2>);
Table created.

SQL> INSERT INTO accounts VALUES (7715, 6350.00);
1 row created.

SQL> INSERT INTO accounts VALUES (7720, 5100.50);
1 row created.

SQL> DECLARE
  2     transfer NUMBER<8,2> := 250;
  3     BEGIN
  4         UPDATE accounts SET balance = balance - transfer WHERE account_id = 7715;
  5         UPDATE accounts SET balance = balance + transfer WHERE account_id = 7720;
  6     COMMIT COMMENT 'Transfer From 7715 to 7720' WRITE IMMEDIATE NOWAIT;
  7 END;
  8 /

PL/SQL procedure successfully completed.

SQL>
```

شكل ۲-۵

طوری که در شکل دیده می‌شود، سه دستور اول، یک جدول را به نام accounts ایجاد و دو ریکورد به آن اضافه کرده است و سپس بلاک کد زبان PL/SQL با اعلامیه DECLARE آغاز و یک متغیر به نام transfer با محتوای عددی 250 تعریف شده است. در قسمت اجرایی بلاک بعد از کلمه BEGIN دو دستور اپدیت به

ترتیب به اجراء گذاشته شده‌اند. دستور کنترول ترانزکشن جزوی از دستورهای همین بلاک بوده و از PL/SQL خواسته شده تا عملیه نوشتن را بدون تأخیر انجام دهد؛ در نهایت، پرسه به روزشدن دیتا با استفاده از دستور تأیید (COMMIT) ترانزکشن را موفقانه نشان می‌دهد.

۵.۳ استفاده از عبارت INTO در بلاک

در زبان PL/SQL به خاطر تعریف کردن متغولین در یک بلاک از دو طریقه استفاده می‌شود: اول با استفاده از سمبول := به طور مستقل یک متتحول در قسمت اعلامیه (Declaration) بلاک متتحول معرفی شده و قیمت به آن داده می‌شود و طریقه دوم با استفاده از عبارت ... INTO ... متحول‌ها در PL/SQL تعریف می‌شوند. طریقه اول در فصول قبلی با ذکر مثال‌ها توضیح شده است. در این درس به توضیحات در مورد طریقه دوم پرداخته می‌شود.

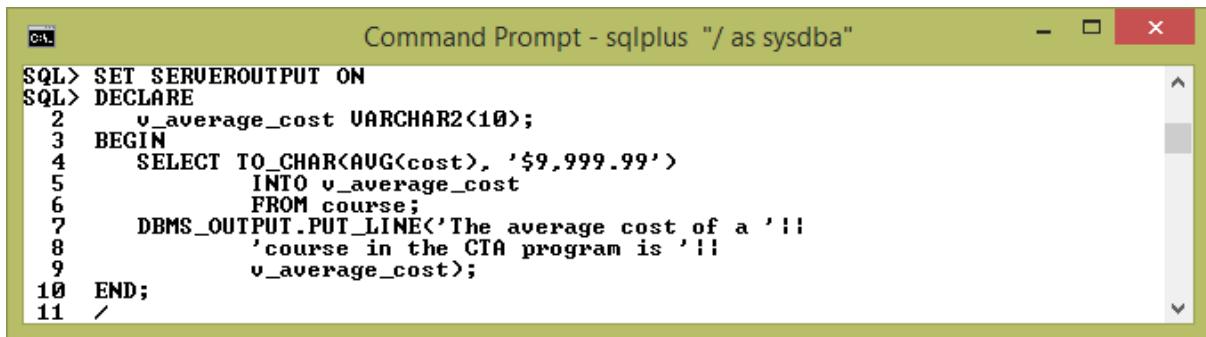
به خاطر استفاده کردن یک متتحول به کمک عبارت SELECT INTO در یک بلاک PL/SQL در قدم نخست متتحول باید در اعلامیه تعریف شده باشد؛ یعنی متتحول فعال در یک بلاک که موجود باشد تا به آن قیمت در قسمت اجرایی بلاک داده شود. شکل عمومی طریقه دادن قیمت به یک متتحول در بلاک PL/SQL طور ذیل است:

```
SELECT item_name  
      INTO variable_name  
      FROM    table_name;
```

در مثال پایین، عبارت SELECT INTO در بلاک PL/SQL به خاطر دادن دیتا به یک متتحول استفاده شده است.

```
SET SERVEROUTPUT ON  
  
DECLARE  
    v_average_cost VARCHAR2(10);  
  
BEGIN  
    SELECT TO_CHAR(AVG(cost), '$9,999.99')  
        INTO v_average_cost  
        FROM course;  
  
    DBMS_OUTPUT.PUT_LINE('The average cost of a'||  
    'course in the CTA program is'||  
    v_average_cost);  
  
END;  
/
```

مثال بالا در صفحه سیکویل پلس طور ذیل دیده می‌شود:



```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      v_average_cost VARCHAR2(10);
 3  BEGIN
 4      SELECT TO_CHAR(AVG(cost), '$9,999.99')
 5          INTO v_average_cost
 6          FROM course;
 7      DBMS_OUTPUT.PUT_LINE('The average cost of a '''
 8          ||course in the CTA program is '''
 9          ||v_average_cost);
10  END;
11 /
```

شکل ۳-۵

دستور اول در شکل بالا به خاطر نشان دادن خروجی استفاده شده است. در سطر ۲ متتحولی به نام `v_average_cost` با نوعیت مشخص تعریف شده است و کدام قیمتی به متتحول مذکور داده نشده است. در سطر ۴ دوتابع استفاده شده‌اند: یکی از آن‌ها عبارت از `AVG()` بوده که اوسع قیمت‌های `cost` را پیدا می‌کند و تابع بیرونی عبارت از `TO_CHAR()` است. تابع بیرونی اوسع پیداشده توسط تابع داخلی و نیز عدد `$9,999.99` را به فارمات کرکتر تبدیل می‌کند. سطر ۵ همین دیتای به دست آمده از سطر قبلی را به متتحول از قبل تعریف شده به نام `v_average_cost` می‌دهد. سطر ۶ جدولی را به نام `course` ریفرنس داده تا از ستون `cost` آن دیتا گرفته شود. سطرهای ۷، ۸ و ۹ یک جمله ترکیبی از سه بخش را به شکل خروجی بلاک کد PL/SQL طور ذیل نشان خواهد داد:

The average cost of a course in the CTA program is (`v_average_cost`)

قیمت متتحول `v_average_cost` به شکل دینامیکی تغییر کرده و در هر بار اجراء، به اساس دیتا جدول `course` تنظیم می‌شود.

نوت: به خاطر اجرای موقانه بلاک کد بالا، جدولی به نام `course` با داشتن فیلد `cost` و دیتای مناسب حتمی است.

۵.۴ کرسوها در سیکویل و خصوصیات آن‌ها

کرسوها (Cursors) عبارت از ساحتی از حافظه کمپیوتراند که در آن‌ها دستورهای کار با دیتا (DML) سیکویل اوریکل اجراء می‌شوند. در برنامه‌نویسی دیتابیس، کرسوها عبارت از ساختارهای داخلی‌اند که در آن‌ها اجرای کیوری‌ها صورت گرفته و نتایج نشان داده می‌شوند. کرسوها مورد استفاده در یک جلسه (Session Cursors) یاد می‌شوند. این نوع کرسوها، معلومات را تا زمان ختم جلسه در خود حفظ می‌کنند. با بسته شدن یک Session استفاده از سیکویل کرسوها مربوطه آن نیز بسته می‌شوند.

معلومات در مورد کرسرهای یک جلسه، با رجوع به خصوصیات جلسه به دست می‌آید. به خاطر انجامدادن این کار از اختیار V\$OPEN_CURSOR استفاده می‌شود. کرسرهای جلسه که توسط PL/SQL در زمان کار با سیستم ایجاد می‌شوند، به نام کرسرهای پوشیده یا ضمنی (Implicit Cursors) یاد می‌شوند. با اجرای هر دستور DML، زبان PL/SQL یک کرس را به شکل ضمنی ایجاد می‌کند. کرسهایی که توسط استفاده‌کننده ایجاد می‌شوند، به نام کرسهای آشکار (Explicit Cursors) یاد می‌شوند؛ پس گفته می‌شود که کرسها در PL/SQL به دو نوع آشکار و پوشیده ایجاد و استفاده می‌شوند.

۵.۴.۱ کرسرهای ضمنی (Implicit Cursors)

یک کرس ضمنی عبارت از کرس جلسه (Session Cursor) بوده که توسط PL/SQL ایجاد و اداره می‌شود. هر باری که یک کیوری و یا دستور دیگر کار با دیتا (DML) توسط استفاده‌کننده به راه انداخته می‌شود، زبان PL/SQL یک کرس ضمنی برای آن باز می‌کند. کرسهای ضمنی توسط استفاده‌کننده قابل کنترول نیستند؛ ولی یک استفاده‌کننده می‌تواند معلومات در مورد آن‌ها را ببیند. نحوه نوشتن دستوری که توسط آن کرسهای ضمنی دیده می‌شوند، به شکل ذیل است:

SQLAttribute

دستور بالا آخرین کرس ضمنی ایجادشده توسط PL/SQL را نشان می‌دهد. اگر هیچ دستور کار با دیتا و کیوری در یک جلسه استفاده نشده باشد و دستور بالا به کار گرفته شود، نتیجه ناشناخته (Null) نشان داده خواهد شد.

یک کرس ضمنی بعد از اجرای دستور مربوطه آن بسته می‌شود. محتوای آن الی صادرشدن دستور بعدی کار با دیتا در یک جلسه در کرس باقی بوده و با آمدن دستور جدید، کرس ضمنی جدید ایجاد شده و معلومات کرس قبلی پاک می‌شود؛ بناءً اگر معلومات کرس ضمنی مربوط یک دستور ضرورت باشد، متعاقب اجرای دستور، معلومات از کرس گرفته شده و در موقعیت دیگر ذخیره شود.

مشخصه‌های کرسهای ضمنی به خاطر گرفتن معلومات عبارت‌اند از:

- SQL%ISOPEN (آیا کرس باز است؟)
- SQL%FOUND (سطرهای متأثرشده موجوداند؟)
- SQL%NOTFOUND (سطرهای متأثرشده موجود نیستند؟)
- SQL%ROWCOUNT (چه تعداد سطرهای متأثر شده‌اند؟)
- ...

از ارائه کردن مثال‌ها برای دیدن معلومات کرسهای ضمنی صرف نظر صورت گرفته است. شاگردان علاقه‌مند می‌توانند به مأخذ ذکر شده در آخر کتاب مراجعه کنند.

۵.۴.۲ کرسرهای آشکار (Explicit Cursors)

کرسرهای آشکار به شکل دستی (Manual) توسط استفاده‌کنندگان ایجاد و استفاده می‌شوند. یک کرس آشکار در یک جلسه سیکویل توسط پروگرام به یک نام مشخص تعریف می‌شود. هر کرس آشکار در زمان ایجاد باید به یک کیوری ارتباط داده شود. زمانی که این کار به صورت درست انجام شود، نتیجه کیوری به یکی از دو شکل زیر استفاده می‌شود:

- کرس آشکار با دستور OPEN باز شود، سطرهای آن با دستور FETCH گرفته شوند و کرس آشکار با دستور CLOSE بسته شود.
- کرس آشکار در یک حلقة پروگرام، مانند FOR LOOP استفاده شود.

کرسرهای آشکار با استفاده از نامشان به خاطر گرفتن دیتا قابل استفاده‌اند. این کار با کرسرهای ضمنی ممکن نیست. مشخصه‌های کرسرهای آشکار نیز مشابه به مشخصه‌های کرسرهای ضمنی‌اند. این مشخصه‌ها به خاطر گرفتن معلومات عبارت‌اند از:

- SQL%ISOPEN (آیا کرس باز است؟)
- SQL%FOUND (آیا سطرهای متأثرشده موجود‌اند؟)
- SQL%NOTFOUND (سطرهای متأثرشده موجود نیستند؟)
- SQL%ROWCOUNT (به چه تعداد سطرهای متأثر شده‌اند؟)
- ...

توضیحات بیشتر در موارد استفاده از کرسرهای آشکار، در فصل هشتم کتاب ارائه شده‌اند.



خلاصه فصل پنجم

در این فصل، عناوین مربوط به مدیریت دیتا در دیتابیس اوریکل ارائه شده‌اند. دستورهای کار با دیتا در کتگوری DML در بلاک‌های کد زبان PL/SQL توضیح شده‌اند. جهت مدیریت بهتر یوزر دیتا در PL/SQL دستورهایی تحت نام ترانزکشن موجود است که در این فصل در یک عنوان مستقل به آن‌ها پرداخته شده و مثال‌های عملی آن نیز نشان داده شده‌اند. استفاده از عبارت SELECT ... INTO به خاطر تعریف کردن متحولین با سهولت‌های لازم در بلاک‌های زبان PL/SQL توضیح شده‌اند.

معلومات در مورد کرسرهای سیکویل و خصوصیات آن‌ها به صورت عمومی ارائه شده‌اند. انواع کرسرهای از قبیل کرسرهای ضمنی و کرسرهای آشکار در PL/SQL با نشان‌دادن مثال‌های عملی توضیح شده‌اند. رول کرسرهای در تنظیم دیتا و به دست آوردن نتایج بهتر از دیتابیس‌های اوریکل نیز به بحث گرفته شده‌اند.



سوالات فصل پنجم

۱. دستورهای DML سیکویل را توضیح دهید.
۲. دستورهای DDL سیکویل را توضیح دهید.
۳. دستورهای DCL سیکویل را توضیح دهید.
۴. کارهایی که توسط دستورهای DML در سیکویل انجام داده می‌شوند، کدامها می‌باشند؟ نام بگیرید.
۵. آیا کیوری انتخاب (Select Query) تنها روی یک دیتابیس در یک زمان تطبیق می‌شود و یا دیتابیس‌های بیشتری هم در یک کیوری قابل استفاده هستند؟ واضح سازید.
۶. ترانزکشن در بلک PL/SQL را توضیح دهید.
۷. در نتیجه استفاده از دستور تکمیلی ترانزکشن به نام COMMIT چه واقع می‌شود؟
۸. در نتیجه استفاده از دستور تکمیلی ترانزکشن به نام ROLLBACK چه واقع می‌شود؟
۹. متحول‌ها در PL/SQL به کدام طریقه‌ها تعریف می‌شوند؟ با ذکر سمبول‌ها مختصراً توضیح دهید.
۱۰. شکل عمومی طریقه دادن قیمت به متحول در بلک PL/SQL را با استفاده از عبارت INTO بنویسید.
۱۱. کرسرهای (Cursors) در سیکویل چه هستند و به کدام منظور استفاده می‌شوند؟
۱۲. کرسرهای ضمنی (Implicit Cursors) را توضیح دهید.
۱۳. کرسرهای آشکار (Explicit Cursors) را توضیح دهید.
۱۴. توسط کدام دستور کرسرهای ضمنی در PL/SQL توسط کدام دستور دیده می‌شوند؟ صرف نام دستور را بنویسید.



فعالیت های فصل پنجم

۱. با استفاده از دستور "sqlplus "/ as sysdba" برنامه سیکویل پلس را از کوماند پرامپت کمپیوترتان به راه بیندازید.
۲. در یک مثال پروگرام PL/SQL با استفاده از DECLARE چند متتحول را تعریف کرده و قیمت برای شان بدهید.
۳. در یک مثال دیگر، یک پروگرام PL/SQL بنویسید که در آن یک متتحول در قسمت اعلامیه تعریف شده باشد؛ ولی قیمت نداشته باشد. به خاطر دادن قیمت به آن از دستور SELECT ... INTO ... استفاده کرده و از یک جدول موجود به متتحول مذکور قیمت داده شود.

فصل ششم

ساختارهای کنترولی (Control Structures)



هدف کلی: شاگردان با طرز نوشتتن ساختارهای کنترولی در زبان PL/SQL آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. انواع و اهداف استفاده از ساختارهای کنترولی در زبان PL/SQL را بدانند.
۲. ساختار و موارد استفاده از دستور IF را شرح دهند.
۳. ساختار و موارد استفاده از دستورهای CASE را شرح دهند.
۴. ساختار و موارد استفاده از دستورهای حلقه (LOOP) را شرح دهند.
۵. ساختار و موارد استفاده از دستورهای EXIT و CONTINUE را در یک حلقه برنامه PL/SQL شرح دهند.

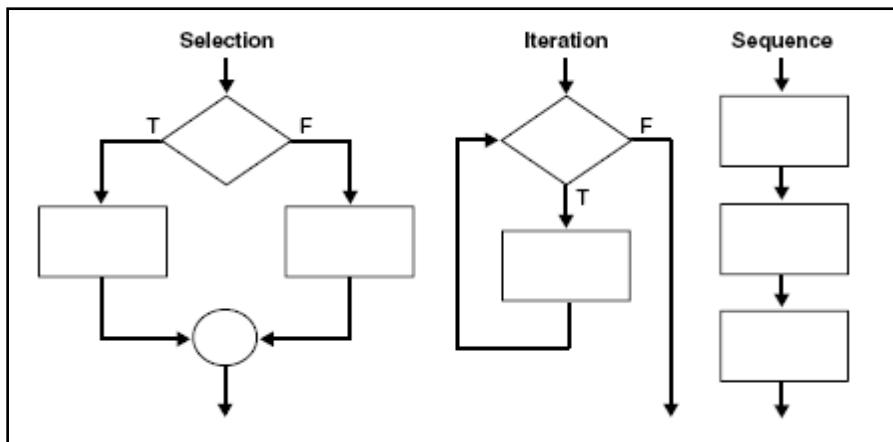
ساختارهای کنترولی یکی از مهم‌ترین بخش‌ها به خاطر پروگرام کردن کارها در زبان‌های پروگرام‌نویسی به شمار می‌رond. با استفاده از این ساختارهای کنترولی، هزاران دستور نظر به ضرورت در چند سطر محدود اجراء و قابل کنترول هستند. استفاده از شرط‌ها در تطبیق ساختارهای کنترولی مؤثربت استفاده از آن‌ها را می‌تواند بیش‌تر سازد. در زبان PL/SQL نیز مانند دیگر زبان‌ها ساختارهایی به خاطر کنترول کردن بخش‌هایی از پروگرام موجود بوده و به طور وسیع استفاده می‌شوند.

در این فصل انواع و اهداف ساختارهای کنترولی (Control Structures) قابل استفاده در پروگرام‌های زبان PL/SQL لیست شده‌اند. بعضی از انواع معروف این ساختارها که موارد استفاده‌شان بیش‌تر است، در عناوین جداگانه شرح شده‌اند. دسته ساختارهای کنترولی IF شامل IF THEN ELSE، IF THEN ELSEIF با مثال‌های لازم و مشابههای در عناوین فرعی تشریح شده‌اند. ساختارهای کنترولی CASE با دو نوع CASE ساده و CASE جست‌وجو شده نیز به ترتیب بحث شده‌اند. دستورهای کنترولی حلقه‌ها که حلقه‌های FOR و WHILE را در بر می‌گیرد، با مثال‌ها توضیح داده شده‌اند. فصل ششم با توضیح ساختارهای CONTINUE و EXIT در حلقه‌ها به پایان رسیده است.

۶.۱ انواع و اهداف استفاده از ساختارهای کنترولی (Control Structures)

در زبان‌های پروسیجری پروگرام‌نویسی مانند PL/SQL ساختارهای کنترولی به صورت عمومی به سه شکل می‌باشند. ساختار انتخاب یک گزینه و اجرای دستورهای مورد نظر در آن با صرف نظر کردن از گزینه دومی عبارت از شکل اول ساختارهای کنترولی است؛ یعنی حالت درست/نادرست در یک دستور دیده شده، نتیجه یکی از حالت‌ها انتخاب شده و به اجراء گذاشته می‌شود. شکل دیگری از ساختارهای کنترولی عبارت از تکرار اجرای یک سلسله از دستورها به شکل تکراری است. در این نوع ساختار، دستورهای مورد نظر تا زمانی اجراء می‌شوند که شرط استفاده شده، صحیح (True) باشد. هر زمانی که نتیجه شرط غلط (False) باشد، اجرای دستورها متوقف می‌شود.

شکل سوم ساختارهای کنترولی در زبان‌های پروسیجری عبارت از ترتیب دستورها است. در این شکل دستورهای مورد نظر در یک ترتیب (Sequence) پروگرام شده و به اجراء گذاشته می‌شوند. تصویر پایین هر سه شکل را با استفاده از شرایط‌شان نشان داده شده است.



شکل ۱-۶ ساختارهای کنترولی پرограмهای پروسیجری کمپیوتر

با درنظرداشت توضیحات بالا در زبان PL/SQL سه نوع دستورهای کنترولی استفاده می‌شوند:

۱. دستورهای شرطی انتخاب حالت (Conditional Selection Statements): در این دسته دستورهای مختلف به اساس دیتاهای مختلف در یک برنامه به راه انداخته می‌شوند. دستورهای انتخاب حالت در زبان PL/SQL شامل دو دستور بوده و عبارت از حالت IF و حالت CASE اند. تفصیل هر کدام از این دستورها با مثال‌ها در عناوین جداگانه شرح شده‌اند.

۲. دستورهای حلقه‌ها (LOOP Statements): این دسته از دستورهای زبان PL/SQL در حلقه‌ها (Loops) تنظیم می‌شوند و به اساس شرایط صحیح و یا غلط وضع شده و دستورها به صورت تکراری اجراء می‌شوند. هر زمانی که شرط مورد نظر به حالت صحیح (True) و یا غلط (False) برسد، نظر به نوع پروگرام، اجرای دستورها توقف پیدا می‌کند. دستورهای حلقه‌ها در زبان PL/SQL عبارت از LOOP و FOR LOOP و WHILE LOOP.

دستورهای خارج شدن (EXIT) و ادامه (CONTINUE) جهت کنترول حلقه‌ها استفاده می‌شوند. (EXIT) به دستورهای اجرایی داخل حلقه پایان داده و دستور دومی (CONTINUE)، حالت ادامه اجرای دستورها را در یک حلقه پروگرام PL/SQL تأیید می‌کند. برای هر دو دستور خارج شدن و ادامه حلقه، شرایطی با استفاده از کلمه WHERE به صورت اختیاری تنظیم می‌شود. وضع شرایط با استفاده از WHERE در حلقه‌ها اختیاری بوده و به خاطر کنترول بهتر پرسه استفاده می‌شود. هر دو موضوع مربوط حلقه‌ها (آغاز حلقه‌ها و خارج شدن و یا ادامه حلقه‌ها) در عناوین جداگانه توضیح و با مثال‌ها نشان داده شده‌اند.

۳. دستورهای کنترول ترتیب‌ها (Sequential Control Statements): این دسته از دستورهای کنترولی در زبان PL/SQL زیاد مورد استفاده ندارد. مثال‌های این دستورها عبارت از دستورهای GOTO و NULL اند. (GOTO) به یک جمله مشخص شده در کد می‌رود و دستور دومی (NULL) کاری را انجام نمی‌دهد. به موضوع دستورهای کنترول ترتیب‌ها در این کتاب پرداخته نشده است.

٦.٢ ساختار و موارد استفاده از دستور IF

دستور IF یکی از دستورهای عام در زبان‌های پروگرام‌نویسی بوده و به خاطر کنترول پروگرام نظر به شرایط وضع شده استفاده می‌شود. در زبان PL/SQL نیز این دستور به عین هدف (کنترول پروگرام) به کار گرفته می‌شود. توسط دستور IF یک سلسله از اجرای دستورها، توقف داده شده و یا ادامه داده می‌شود. کنترول توقف یا ادامه اجرای دستورها نظر به ضرورت استفاده کنندگان تنظیم می‌شود. دستور IF در زبان PL/SQL به سه شکل استفاده می‌شود:

- ٨. IF THEN
- ٩. IF THEN ELSE
- ١٠. IF THEN ELSIF

حالات‌های مختلف استفاده از دستور IF در عناوین بعدی تشریح می‌شوند.

• دستور IF THEN

این دستور ساده‌ترین شکل استفاده از دستور کنترولی IF به شمار می‌رود. در دستور IF THEN یک سلسله از دستورها، توقف داده شده و یا به اجرای آن‌ها اجازه داده می‌شود. در سلسله دستورها، حداقل یک دستور بوده می‌تواند. شکل عمومی دستور IF THEN طور ذیل است:

IF condition THEN

statements

END IF;

در دستور بالا اگر شرط (condition) صحیح بود، سلسله دستورها (statements) به اجراء گذاشته می‌شوند و اگر شرط صدق نکرده باشد (غلط یا FALSE باشد)، در آن صورت دستورها اجراء نمی‌شوند؛ یعنی دستور به اساس نتیجه بولی (Boolean) کار می‌کند.

مثال پایین در نظر گرفته شود.

The screenshot shows the Oracle SQL Developer interface with a PL/SQL procedure named 'p'. The code is as follows:

```

SET SERVEROUTPUT ON
DECLARE
  PROCEDURE p (
    sales NUMBER,
    quota1 NUMBER,
    emp_id NUMBER
  )
  IS
    bonus NUMBER := 0;
    updated1 VARCHAR2(3) := 'No';
  BEGIN
    IF sales > (quota1 + 200) THEN
      bonus := (sales - quota1)/4;
      UPDATE employees
      SET salary = salary + bonus
      WHERE employee_id = emp_id;
      updated1 := 'Yes';
    END IF;
    DBMS_OUTPUT.PUT_LINE (
      'Table updated? ' || updated1 || ', '
      'bonus = ' || bonus || '.'
    );
  END p;
BEGIN
  p(10100, 10000, 120);
  p(10500, 10000, 121);
END;
/

```

شکل ۲-۶

در مثال بالا، دستورهایی که بین کلمات END IF و THEN آمده‌اند، تنها در صورتی اجراء می‌شوند که قیمت متتحول sales بیشتر از قیمت quota1 + 200 باشد. این شرط به اساس دستور IF که به تعقیب کلمه IF آمده است، در نظر گرفته می‌شود. نتیجه اجرای پروگرام بالا (در صورت موجودیت جدول employees با ساختار و دیتای مورد نظر) طور ذیل خواهد بود:

Table updated? No, bonus = 0.

Table updated? Yes, bonus = 125.

• دستور IF THEN ELSE

این دستور نظر به دستور قبلی (IF THEN) اختیارات بیشتری به استفاده کننده می‌دهد. هدف استفاده از دستور IF THEN ELSE باز هم کنترول اختیار دیتا در پروگرام زبان PL/SQL است. شکل عمومی این دستور طور ذیل است:

IF condition THEN

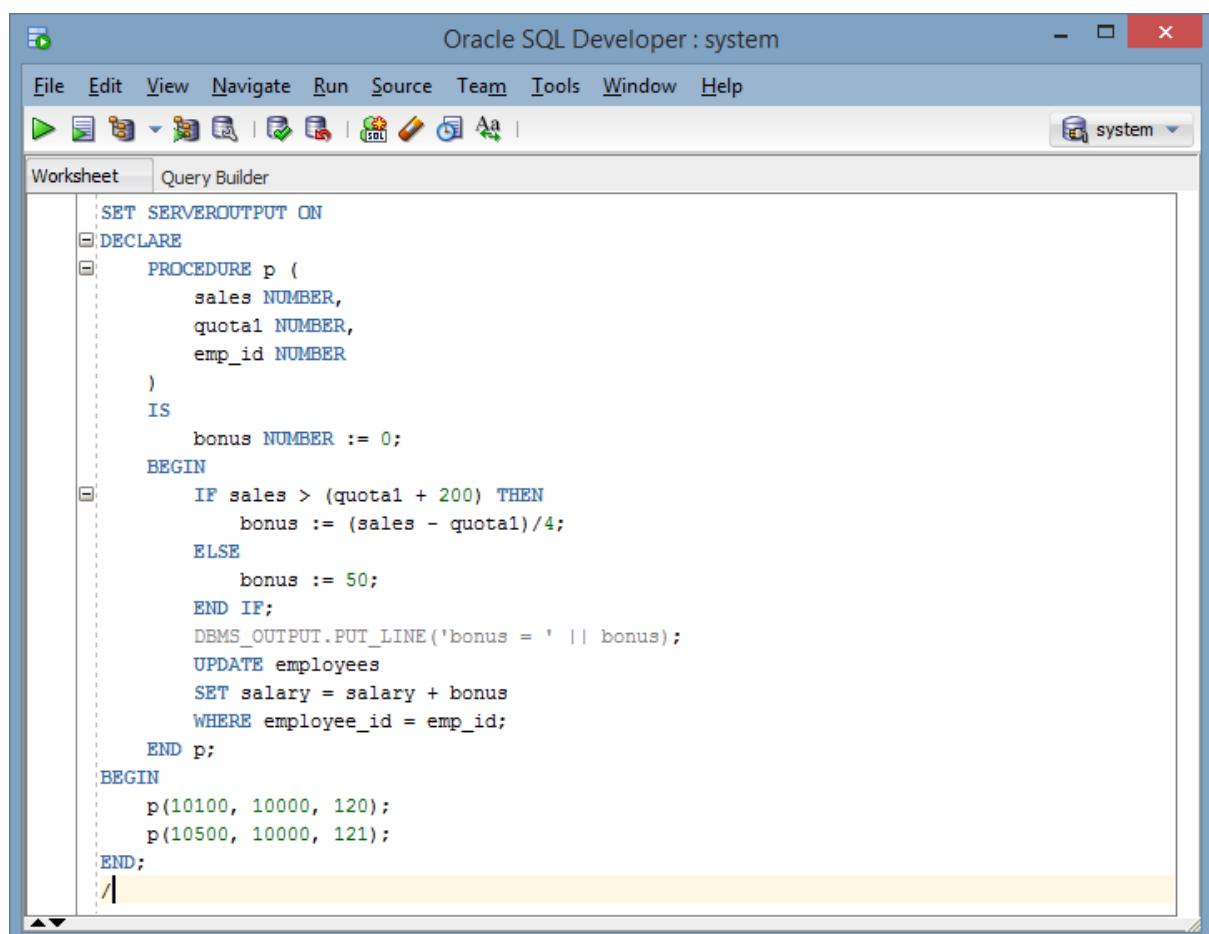
 statements

ELSE

 else_statements

END IF;

در صورت صحیح بودن شرط (condition) دستورهای سطر دوم (statements) اجراء می‌شوند و اگر شرط غلط (FALSE) بود، دستورهای سطر چهارم (else_statements) به اجراء در می‌آیند؛ به مثال پایین توجه شوید:



The screenshot shows the Oracle SQL Developer interface with a PL/SQL procedure named 'p' defined in the 'system' schema. The code includes an IF-THEN-ELSE block to calculate a bonus based on sales, update the employees table, and then call the procedure again with specific parameters.

```
SET SERVEROUTPUT ON
DECLARE
    PROCEDURE p (
        sales NUMBER,
        quota1 NUMBER,
        emp_id NUMBER
    )
    IS
        bonus NUMBER := 0;
    BEGIN
        IF sales > (quota1 + 200) THEN
            bonus := (sales - quota1)/4;
        ELSE
            bonus := 50;
        END IF;
        DBMS_OUTPUT.PUT_LINE('bonus = ' || bonus);
        UPDATE employees
        SET salary = salary + bonus
        WHERE employee_id = emp_id;
    END p;
BEGIN
    p(10100, 10000, 120);
    p(10500, 10000, 121);
END;
```

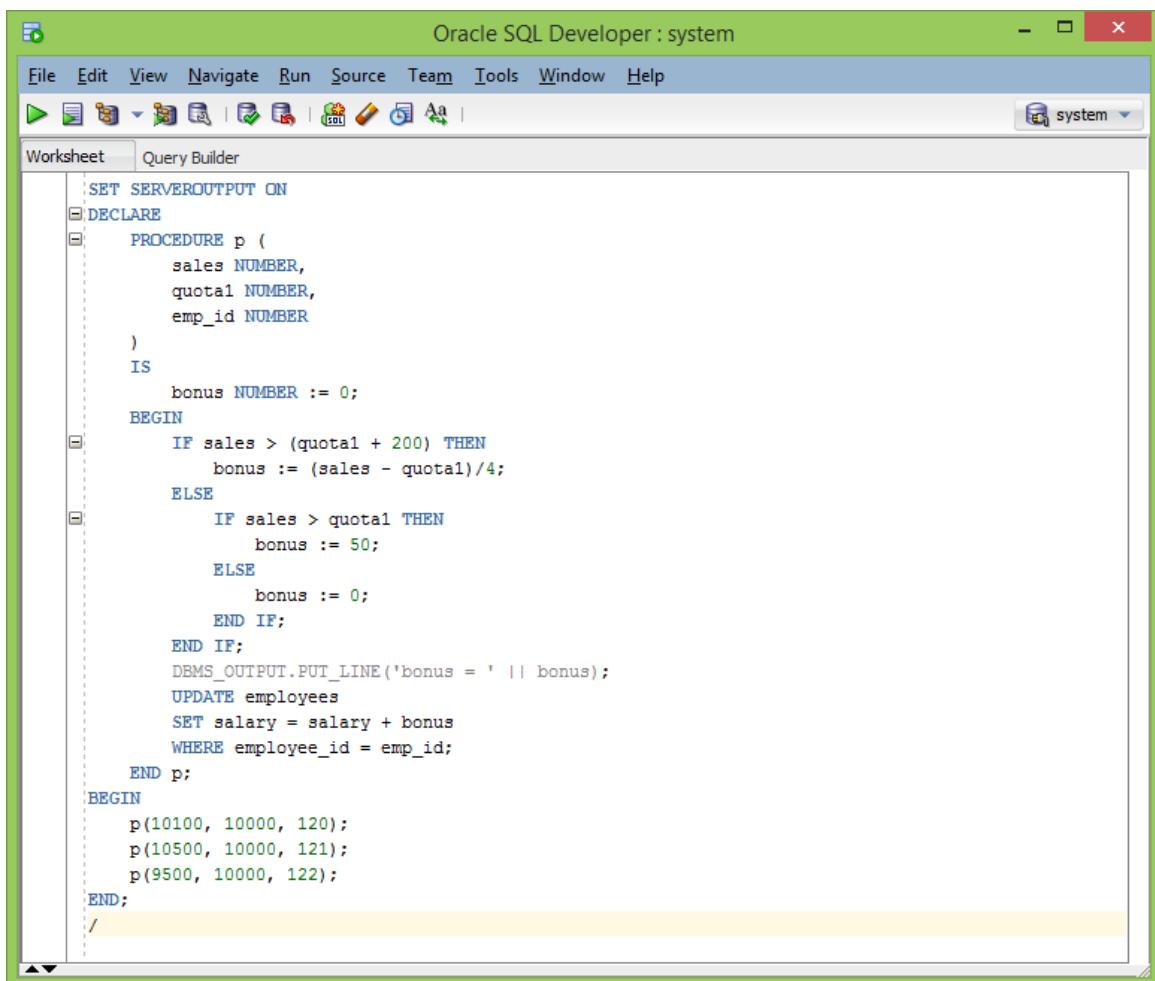
شکل ۳-۶

در این مثال دستورهایی که بین کلمات THEN و ELSE آمده‌اند، تنها در صورتی اجراء می‌شوند که قیمت متتحول sales بیش‌تر از قیمت 200 باشد. در غیر این صورت (قیمت متتحول quota1 مساوی و یا کمتر از قیمت 200 باشد)، دستورهایی که در بین END IF و ELSE اند به اجراء گذاشته می‌شوند. نتیجه اجرای پروگرام بالا (باز هم در صورت موجودیت جدول employees با ساختار و دیتای مورد نظر) طور ذیل خواهد بود:

bonus = 50

bonus = 125

دستورهای IF به شکل آشیانه‌یی (Nested) نیز قابل استفاده هستند؛ به مثال پایین در این مورد توجه شود.



```

SET SERVEROUTPUT ON
DECLARE
  PROCEDURE p (
    sales NUMBER,
    quota1 NUMBER,
    emp_id NUMBER
  )
  IS
    bonus NUMBER := 0;
  BEGIN
    IF sales > (quota1 + 200) THEN
      bonus := (sales - quota1)/4;
    ELSE
      IF sales > quota1 THEN
        bonus := 50;
      ELSE
        bonus := 0;
      END IF;
    END IF;
    DBMS_OUTPUT.PUT_LINE('bonus = ' || bonus);
    UPDATE employees
    SET salary = salary + bonus
    WHERE employee_id = emp_id;
  END p;
BEGIN
  p(10100, 10000, 120);
  p(10500, 10000, 121);
  p(9500, 10000, 122);
END;
/

```

شکل ۴-۶

طوری که در این مثال دیده می‌شود، دستور کنترولی IF با شرایط مربوطه‌اش در بین دستور IF دیگر بعد از اختیار ELSE استفاده شده است؛ به عین ترتیب، در صورت ضرورت دستور IF به شکل آشیانه‌یی یکی در بین

دیگری قابل استفاده است. نتیجه تطبیق پروگرام بالا، مشابه به مثال‌های قبلی، ولی با تفاوت طور ذیل خواهد بود:

bonus = 50

bonus = 125

bonus = 0

• دستور IF THEN ELSIF

این دستور شکل پیچیده‌تری از دستور IF است؛ ولی امکانات بیشتری را به پروگرامر در قسمت استفاده از دستور IF می‌دهد. شکل عمومی دستور IF THEN ELSIF طور ذیل است:

```
IF condition_1 THEN  
    statements_1  
ELSIF condition_2 THEN  
    statements_2  
[ ELSIF condition_3 THEN  
    statements_3  
] ...  
[ ELSE  
    else_statements  
]  
END IF;
```

دستور کنترولی IF THEN ELSIF زبان PL/SQL در صورت صحیح بودن شرط اول conditions_1، جملات اول statements_1 را اجراء می‌کند. در چنین حالتی (صحیح بودن شرط اول) متنباقی شرایط ارزیابی نشده و نتیجه نشان داده می‌شود. به همین ترتیب در صورت درست نبودن شرط اول به شرط دوم رجوع شده و مسأله تکرار می‌شود. در حالتی که تمام شرایط conditions (غلط باشند، دستورهای آخر else_statements)، اگر موجود باشند، اجراء می‌شوند؛ یعنی موجودیت ELSE در این دستور اختیاری است.

همین کاری که توسط دستور کنترولی IF THEN ELSIF انجام داده می‌شود، با استفاده از دستورهای آشیانه‌بی IF THEN ELSE نیز قابل اجراء است. تحلیل و دانستن استفاده از دستور اول به نسبت دستور دوم در جمله قبلی ساده‌تر است. به خاطر وضاحت موضوع نمونه‌هایی از هر دو نوع، به جهت انجام دادن عین کار در ذیل نشان داده شده‌اند:

:IF THEN ELSIF دستور

IF condition_1 THEN statements_1;

```

ELSIF condition_2 THEN statements_2;
ELSIF condition_3 THEN statements_3;
END IF;

```

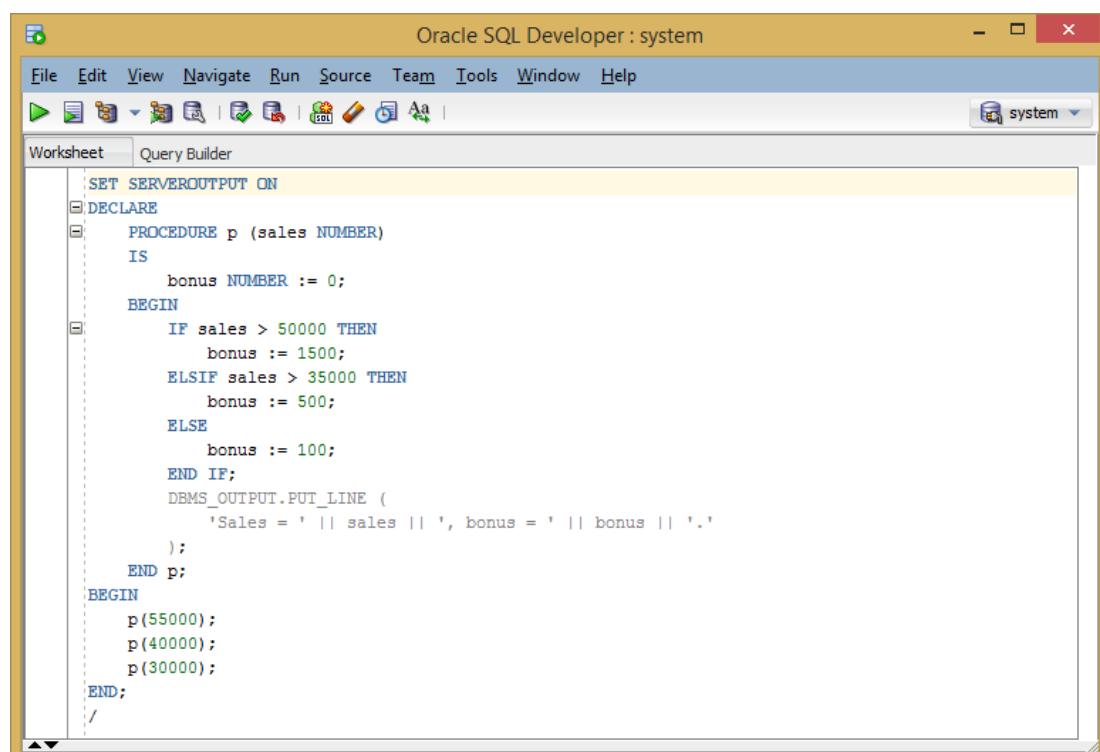
دستور بالا از نظر منطقی معادل دستور آشیانه‌یی IF THEN ELSE در ذیل است:

```

IF condition_1 THEN
    statements_1;
ELSE
    IF condition_2 THEN
        statements_2;
    ELSE
        IF condition_3 THEN
            statements_3;
        END IF;
    END IF;
END IF;

```

در قسمت استفاده از دستور کنترولی IF THEN ELSIF به مثال پایین توجه شود.



The screenshot shows the Oracle SQL Developer interface with the title bar "Oracle SQL Developer : system". The menu bar includes File, Edit, View, Navigate, Run, Source, Team, Tools, Window, and Help. Below the menu is a toolbar with various icons. The main area is titled "Worksheet" and contains a query builder. The code displayed is:

```

SET SERVEROUTPUT ON
DECLARE
  PROCEDURE p (sales NUMBER)
  IS
    bonus NUMBER := 0;
  BEGIN
    IF sales > 50000 THEN
      bonus := 1500;
    ELSIF sales > 35000 THEN
      bonus := 500;
    ELSE
      bonus := 100;
    END IF;
    DBMS_OUTPUT.PUT_LINE (
      'Sales = ' || sales || ', bonus = ' || bonus || '.'
    );
  END p;
BEGIN
  p(55000);
  p(40000);
  p(30000);
END;
/

```

شکل ۵-۶

در مثال بالا اگر قیمت متحول sales بیشتر از 50000 شود، هر دو حالت اول و دوم صحیح می‌شوند؛ چون درست‌بودن حالت اول مقدم است، پس متحول bonus قیمت 1500 را به خود می‌گیرد. بعد از اجرای حالت اول، از حالت دوم صرف نظر گردیده و پروگرام مستقیماً به اجرای دستور خروجی DBMS_OUTPUT.PUT_LINE در سطر 13 (شکل پایین) می‌رود.

چون این مثال ضرورت به دیتای بیرونی ندارد، نتیجه اجرای موفقانه آن در شکل پایین نشان داده شده است.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2    PROCEDURE p (sales NUMBER)
  3    IS
  4      bonus NUMBER := 0;
  5    BEGIN
  6      IF sales > 50000 THEN
  7        bonus := 1500;
  8      ELSIF sales > 35000 THEN
  9        bonus := 500;
 10      ELSE
 11        bonus := 100;
 12      END IF;
 13      DBMS_OUTPUT.PUT_LINE (
 14        'Sales = ' || sales || ', bonus = ' || bonus || '.'
 15    );
 16    END p;
 17  BEGIN
 18    p(55000);
 19    p(40000);
 20    p(30000);
 21  END;
 22 /
Sales = 55000, bonus = 1500.
Sales = 40000, bonus = 500.
Sales = 30000, bonus = 100.

PL/SQL procedure successfully completed.

SQL>

```

شکل ۶-۶

یک مثال دیگر در مورد مشابه در نظر گرفته شده است.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2    grade CHAR(1);
  3    BEGIN
  4      grade := 'B';
  5      IF grade = 'A' THEN
  6        DBMS_OUTPUT.PUT_LINE('Excellent');
  7      ELSIF grade = 'B' THEN
  8        DBMS_OUTPUT.PUT_LINE('Very Good');
  9      ELSIF grade = 'C' THEN
 10        DBMS_OUTPUT.PUT_LINE('Good');
 11      ELSIF grade = 'D' THEN
 12        DBMS_OUTPUT.PUT_LINE('Fair');
 13      ELSIF grade = 'F' THEN
 14        DBMS_OUTPUT.PUT_LINE('Poor');
 15      ELSE
 16        DBMS_OUTPUT.PUT_LINE('No such grade');
 17      END IF;
 18  END;
 19 /
Very Good

PL/SQL procedure successfully completed.

SQL>

```

شکل ۷-۶

در مثال بالا با استفاده از دستور IF THEN ELSIF چندین حالت در نظر گرفته شده و در مقابل قیمت متحول grade ارزیابی شده‌اند. نتیجه درست‌بودن حالت ELSIF در سطر 8 پروگرام تدقیق شده و متن در نتیجه پروگرام از همین سطر نشان داده می‌شود. این مثال با استفاده از دستور CASE نیز قابل اجراء بوده و در یک مثال درس بعدی نشان داده شده است.

٦.٣ ساختار و موارد استفاده از دستورهای CASE

دستور CASE مانند دستور IF یکی از دستورهایی است که به خاطر کنترول کردن حالت یک پروگرام یا انجام‌دادن یک کار در یک پروگرام استفاده می‌شود. توسط دستور CASE کارهای مختلف با قیمت‌های مختلف دیتا انجام داده می‌شوند. در این دستور شرایط داده شده، چک می‌شود و در صورت درست‌بودن شرط، دستور مربوطه آن به راه انداخته می‌شود.

دستور CASE به دو شکل استفاده می‌شود:

1. دستور CASE ساده (Simple CASE Statement): در این حالت یک عبارت گرفته شده و به محتواهای ممکن‌آن مقایسه می‌شود. تفصیل بیشتر و مثال‌ها در درس بعدی توضیح شده‌اند.
2. دستور CASE جست‌وجو شده (Searched CASE Statement): در این حالت چندین شرط ارزیابی شده و اولین آن که درست باشد، انتخاب می‌شود. تفصیل بیشتر و مثال‌ها در درس بعدی توضیح داده شده‌اند. دستور CASE در مواردی بهتر است استفاده شود که در آن‌ها کارهای مختلف ضرورت به اجراء داشته باشند.

٦.٣.١ دستور CASE ساده

شکل عمومی این دستور طور ذیل است:

CASE selector

WHEN selector_value_1 THEN statements_1

WHEN selector_value_2 THEN statements_2

...

WHEN selector_value_n THEN statements_n

[ELSE

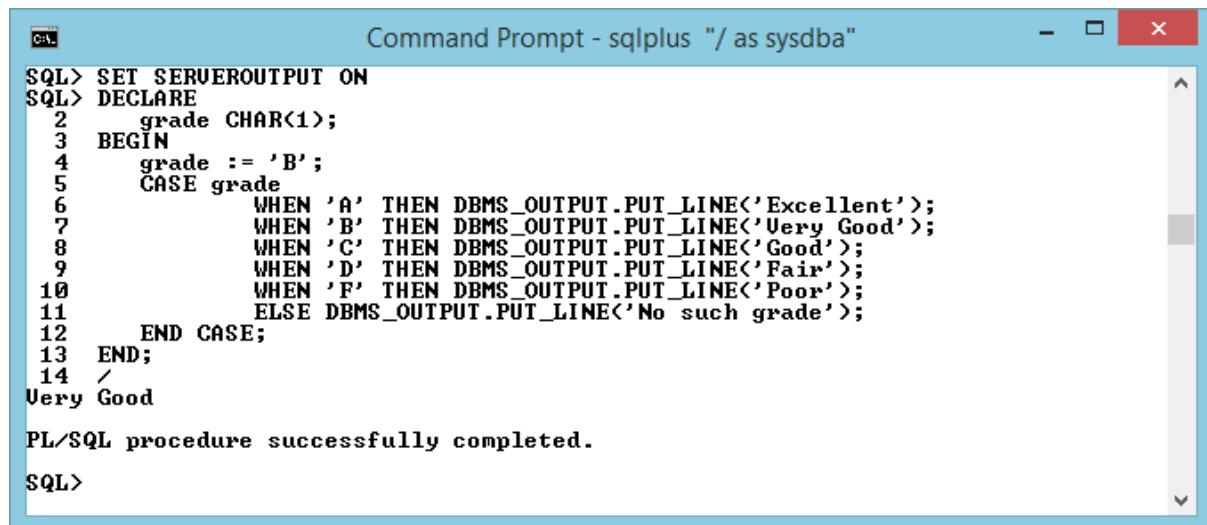
Else_statements]

END CASE;

در شکل عمومی دستور CASE ساده، کلمه selector یک اصطلاح است که به معنای یک متحول منفرد می‌باشد؛ به عین شکل هر selector_value می‌تواند یک حرف و یا یک عبارت باشد. دستور CASE زمانی selector می‌شود که اولین محتوای selector_value مساوی به محتوای selector می‌شود. زمانی که

مساوی پیدا شد، متنباقی حالتها (selector_values) ارزیابی نمی‌شوند. در صورت پیدانشدن قیمت مساوی، جملات بعد از ELSE به اجراء گذاشته می‌شوند. قسمت ELSE در دستور CASE ساده، اختیاری است.

به مثال ذیل توجه شود:



```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   grade CHAR(1);
 3   BEGIN
 4     grade := 'B';
 5     CASE grade
 6       WHEN 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
 7       WHEN 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
 8       WHEN 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
 9       WHEN 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
10       WHEN 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
11     ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
12   END CASE;
13 END;
14 /
Very Good
PL/SQL procedure successfully completed.
SQL>
```

شکل ۸-۶

در مثال بالا، یک دستور CASE ساده استفاده شده است. در مثال مذکور، یک محتوا با چندین محتوای ممکنه مقایسه می‌شود. اگر قیمت متتحول B در سطر 4 تغییر داده شود، بدون شک نتیجه‌یی (Very Good) که فعلًا از سطر 7 گرفته شده است، نیز تغییر خواهد کرد. این مثال از نظر کار، مشابه به آخرین مثال درس قبلی است که در آن از دستور کنترولی IF THEN ELSIF استفاده شده بود.

۶.۳.۲ دستور CASE جستجو شده

شکل عمومی این دستور طور ذیل است:

CASE

WHEN condition_1 THEN statements_1

WHEN condition_2 THEN statements_2

...

WHEN condition_n THEN statements_n

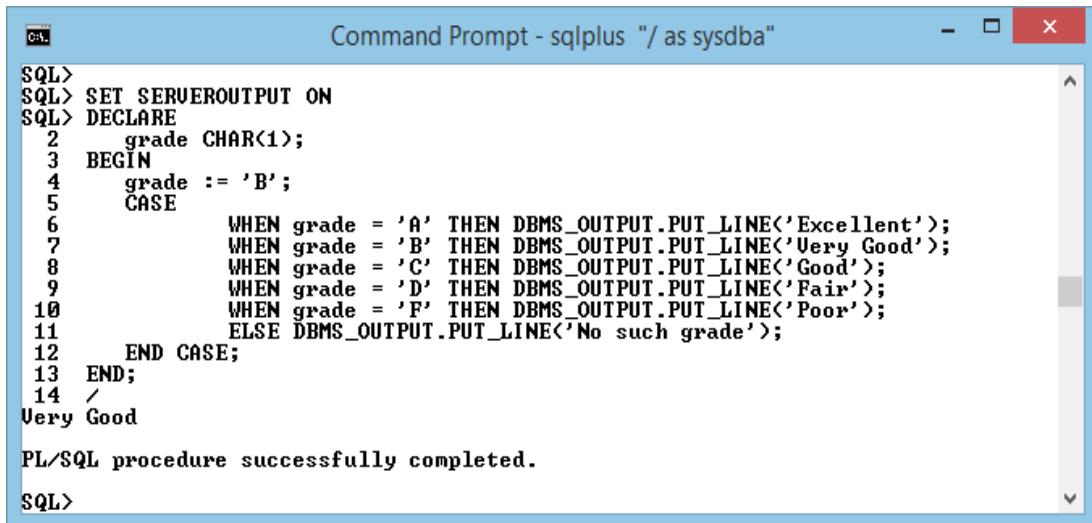
[ELSE

Else_statements]

END CASE;

دستور CASE جستجو شده با پیدا کردن اولین حالت (condition) صحیح، جملات مربوطه را به اجراء می‌گذارد. در چنین صورتی، حالت‌های باقی‌مانده ارزیابی نمی‌شوند و دستور END CASE اجرا می‌گردد. اگر یکی از حالت‌ها هم درست نباشد، بخش ELSE اجرا خواهد شد.

به مثال پایین توجه شود.



```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   grade CHAR(1);
 3   BEGIN
 4     grade := 'B';
 5     CASE
 6       WHEN grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
 7       WHEN grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
 8       WHEN grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
 9       WHEN grade = 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
10      WHEN grade = 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
11      ELSE DBMS_OUTPUT.PUT_LINE('No such grade');
12    END CASE;
13  END;
14 /
Very Good

PL/SQL procedure successfully completed.

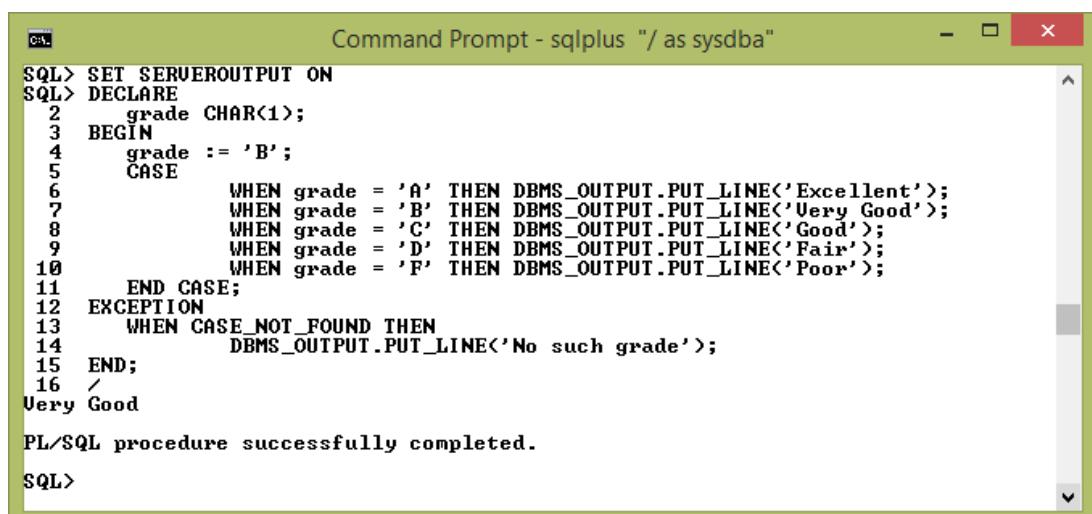
SQL>

```

شکل ۹-۶

این مثال نیز مشابه به مثال‌های اجراء شده در CASE ساده و IF THEN ELSE مربوط درس‌های قبلی است.

در این مثال، در هردو حالت استفاده از CASE ساده و CASE جستجو شده، اصطلاح ELSE با یک قسمت EXCEPTION (تبدیل شده می‌تواند. معلومات در مورد استثناهای (Exceptions) مربوط بلاک PL/SQL است) در فصل‌های بعدی کتاب گنجانیده شده است. در مثال پایین بخش استثنا به عوض بخش اختیاری استفاده شده است.



```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   grade CHAR(1);
 3   BEGIN
 4     grade := 'B';
 5     CASE
 6       WHEN grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
 7       WHEN grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
 8       WHEN grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
 9       WHEN grade = 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
10      WHEN grade = 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
11    END CASE;
12  EXCEPTION
13    WHEN CASE_NOT_FOUND THEN
14      DBMS_OUTPUT.PUT_LINE('No such grade');
15  END;
16 /
Very Good

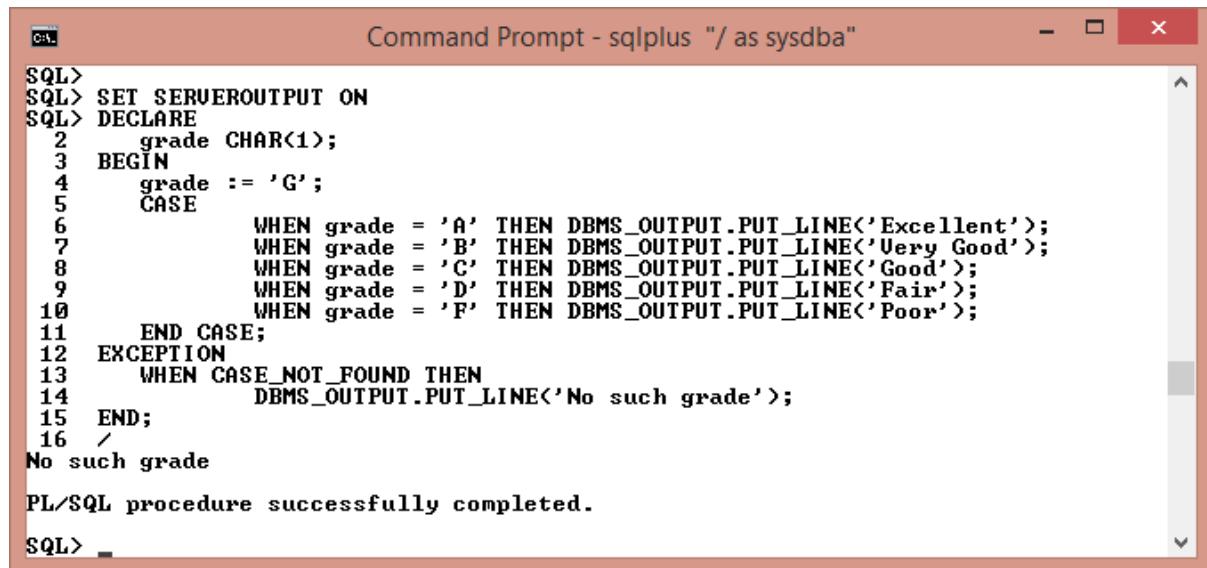
PL/SQL procedure successfully completed.

SQL>

```

شکل ۱۰-۶

طوری که دیده می‌شود، در نتیجهٔ پروگرام تغییر نیامده و همان Very Good نشان داده شده است که درست‌بودن سطر 7 در آن تأیید شده است؛ یعنی حذف کردن قسمت ELSE در دستور CASE و علاوه‌کردن بخش استثناء (Exception)، در نتیجهٔ پروگرام کدام تغییری وارد نکرده است. اگر قیمت متتحول grade چیزی خارج از محدودهٔ تعریف شده (A, B, C, D, E, F) داده شود، بدون شک دستور گنجانیده شده در استثناء به اجراء در آمده و نتیجهٔ طور ذیل خواهد بود:



```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   grade CHAR<1>;
 3   BEGIN
 4     grade := 'G';
 5     CASE
 6       WHEN grade = 'A' THEN DBMS_OUTPUT.PUT_LINE('Excellent');
 7       WHEN grade = 'B' THEN DBMS_OUTPUT.PUT_LINE('Very Good');
 8       WHEN grade = 'C' THEN DBMS_OUTPUT.PUT_LINE('Good');
 9       WHEN grade = 'D' THEN DBMS_OUTPUT.PUT_LINE('Fair');
10       WHEN grade = 'F' THEN DBMS_OUTPUT.PUT_LINE('Poor');
11     END CASE;
12   EXCEPTION
13     WHEN CASE_NOT_FOUND THEN
14       DBMS_OUTPUT.PUT_LINE('No such grade');
15   END;
16 /
No such grade
PL/SQL procedure successfully completed.
SQL> -

```

شکل ۱۱-۶

در مثال بالا به متتحول grade قیمت G داده شده است. در دستور CASE از تمام حالت‌های دیده شده، نتیجهٔ درست به دست نمی‌آید. دستور CASE پایان یافته و دستور استثناء به راه انداخته می‌شود که در نتیجهٔ خروجی داده می‌شود No such grade.

۶.۴ ساختار و موارد استفاده از دستورهای حلقه (LOOP Statements)

دستورهای حلقه‌ها در زبان‌های پروگرام‌نویسی یکی دیگر از وسیله‌های مهم به خاطر تنظیم کردن بخش‌های پروگرام است. توسط LOOP‌ها عین دستورها با قیمت‌های مختلف در حلقه‌هایی تطبیق می‌شوند که همین حلقه‌ها با استفاده از شرط‌های مشخص شده، توسط پروگرامر کنترول می‌شوند؛ یعنی زمانی که یک حلقه در پروگرامی استفاده می‌شود، به خاطر پایان یافتن آن، شرط‌ها و یا دامنهٔ دیتای دخولی به آن وضع می‌شود. عناصر حلقه در زبان PL/SQL عبارت‌اند از:

۱. حلقهٔ اساسی (Basic LOOP): در حلقهٔ اساسی با هر بار تکرار، دستورهای داخل حلقه اجراء شده و کنترول دوباره به نقطهٔ آغاز آن بر می‌گردد. هر حلقه باید دستور پایانی (EXIT) داشته باشد تا از مصروف شدن دائمی کمپیوتر در زمان اجرای حلقه، جلوگیری شود. دستورهای مربوط به پایان دادن حلقه‌ها در قسمت آخر فصل در یک عنوان جداگانه توضیح داده شده‌اند.

شکل عمومی یک حلقه اساسی طور ذیل است:

```
[ label ] LOOP  
statements  
END LOOP [ label ];
```

2. حلقه FOR LOOP (FOR LOOP): یکی از شکل‌های حلقه‌ها است که بیشتر استفاده می‌شود. توضیحات و مثال‌های لازم تحت عنوان جداگانه در صفحه بعدی گنجانیده شده است.

3. حلقه کرسر FOR LOOP (Cursor FOR LOOP): با استفاده کردن از این حلقه، پروگرامر می‌تواند کیوری انتخاب (SELECT) را طوری به کار گیرد که با هر بار تکرار حلقه، نتیجه دستورها به شکل متصل و در عین زمان صادر شود. این نوع حلقه می‌تواند یک کرسر (آشکار و یا پوشیده) را استفاده کند؛ ولی استفاده از متحول در آن مجاز نیست.

4. حلقه WHILE LOOP (WHILE LOOP): این نوع حلقه نیز یکی دیگر از شکل‌های حلقه‌ها است که بیشتر استفاده می‌شود. توضیحات و مثال‌های لازم تحت عنوان جداگانه بعداً نشان داده خواهد شد.

حلقه‌ها می‌توانند با لیبل‌ها نام‌گذاری شوند و نیز می‌توانند به شکل آشیانه‌یی (Nested) یکی در بین دیگری استفاده شوند. لیبل‌گذاری حلقه‌ها، وضاحت بیشتری به پروگرامرها می‌دهد. زمانی که حلقه‌ها به شکل آشیانه‌یی در یک بلاک کد PL/SQL دیزاین می‌شوند، استفاده از لیبل‌ها به خاطر تفکیک و خوانایی بهتر، یکی از امور ضروری پنداشته می‌شود؛ البته استفاده از لیبل‌ها در هر حالت (حالت عادی و یا حالت آشیانه‌یی حلقه‌ها) اختیاری است. مسئله مهم دیگر این است که اگر از لیبل‌ها استفاده شود، حروف و کلمات استفاده شده برای نام لیبل، در آغاز و انجام حلقه باید به یک شکل نوشته شده باشند.

توضیحات عمومی در مورد دو عنصر لیست شده در قسمت بالا به شکل مختصر ارائه شدند. توضیحات مکمل و مثال‌های لازم به خاطر معمول بودن دو عنصر دیگر، هر کدام حلقه FOR و حلقه WHILE، تحت عناوین جداگانه در نظر گرفته شده و در ذیل ارائه می‌شوند.

۶.۴.۱ دستور حلقه FOR

دستور حلقه FOR یک یا بیشتر جملاتی را اجراء می‌کند که اندکس آن جمله‌ها در دامنه (Range) مطابقت داشته باشد. شکل عمومی دستور حلقه FOR طور ذیل است:

```
[ label ] FOR index IN [ REVERSE ] lower_bound. .upper_bound LOOP  
statements  
END LOOP [ label ];
```

طوری که در شکل عمومی دستور حلقه FOR دیده می‌شود، قسمت‌های اختیاری، عبارت از لیبل و کلمه Reverse (معکوس) است؛ یعنی یک حلقه FOR می‌تواند به یک نام مسمی شود و یا هم بدون لیبل باشد. در صورت عدم استفاده از اختیار REVERSE، اندکس یک حلقه FOR از مرز پایین (lower_bound) آغاز شده و تکرار آن الی قیمت مرز بالایی (upper_bound) ادامه پیدا می‌کند. از نظر منطقی، قیمت مرز پایین باید کمتر از مرز بالا باشد؛ اگر بر عکس این (قیمت مرز پایینی بزرگ‌تر از قیمت مرز بالایی) تنظیم شده باشد، در این حالت دستورهای داخل حلقه هیچ‌گاه اجراء نخواهند شد.

با استفاده‌کردن اختیار REVERSE در یک حلقه FOR، اندکس از قیمت بالایی دامنه آغاز و در هر بار اجراء، یک واحد از آن کم شده تا به قیمت پایانی برسد. در این حالت اگر قیمت بالایی کمتر از قیمت پایانی باشد، حلقه FOR باز هم اجراء نخواهد شد.

دستورهای ذیل باعث ایجاد سکته‌گی در یک حلقه FOR می‌شوند:

1. EXIT
2. EXIT WHEN
3. CONTINUE
4. CONTINUE WHEN

توضیحات و مثال‌های لازم در مورد استفاده از دستورهای فوق در یک حلقه FOR بعدتر ارائه شده‌اند.

به مثال ذیل توجه شود:

```

Command Prompt - sqlplus "/ as sysdba"
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
 2   DBMS_OUTPUT.PUT_LINE ('lower_bound < upper_bound');
 3   FOR i IN 1..3 LOOP
 4     DBMS_OUTPUT.PUT_LINE (i);
 5   END LOOP;
 6   DBMS_OUTPUT.PUT_LINE ('lower_bound = upper_bound');
 7   FOR i IN 2..2 LOOP
 8     DBMS_OUTPUT.PUT_LINE (i);
 9   END LOOP;
10   DBMS_OUTPUT.PUT_LINE ('lower_bound > upper_bound');
11   FOR i IN 3..1 LOOP
12     DBMS_OUTPUT.PUT_LINE (i);
13   END LOOP;
14 END;
15 /
lower_bound < upper_bound
1
2
3
lower_bound = upper_bound
2
lower_bound > upper_bound
PL/SQL procedure successfully completed.

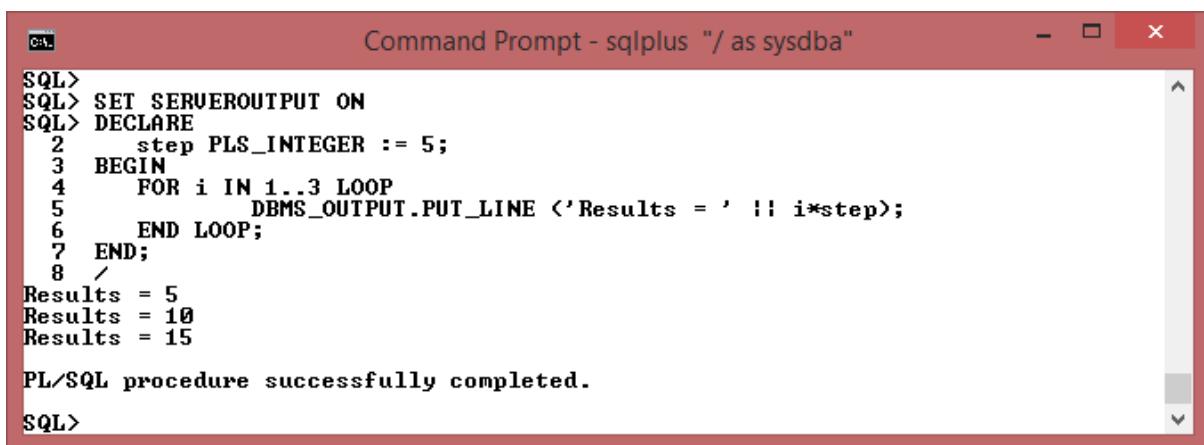
SQL>

```

شکل ۱۲-۶

طوری که در شکل دیده می‌شود، در یک بلاک زبان PL/SQL سه حلقه استفاده شده‌اند. اندکس در این حلقه‌ها عبارت از *i* بوده و مرزهای مختلف بین ۱ الی ۳ استفاده شده‌اند. خروجی هر حلقه با متن مربوطه آن به ترتیب در نتیجهٔ پروگرام چاپ شده است.

در بعضی زبان‌ها، در حلقة FOR با اضافه‌ساختن کلمهٔ کلیدی STEP مقدار تزايدی اندکس تغيير داده می‌شود. در زبان PL/SQL به خاطر ديدن نتيجهٔ متحولی استفاده شده و هر اندکس در هر اجراء، به آن ضرب می‌شود. در مثال پايين همین کار انجام شده و قيمت متحول عدد ۵ تعريف شده است. با هر بار اجراء، عدد ۵ در اندکس ضرب شده و نتيجهٔ حاصله چاپ می‌شود.



```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2   step PLS_INTEGER := 5;
  3   BEGIN
  4     FOR i IN 1..3 LOOP
  5       DBMS_OUTPUT.PUT_LINE ('Results = ' || i*step);
  6     END LOOP;
  7   END;
  8 /
Results = 5
Results = 10
Results = 15
PL/SQL procedure successfully completed.
SQL>

```

شكل ۱۳-۶

۶.۴.۲ دستور حلقة WHILE

دستور حلقة WHILE مانند حلقة FOR یک یا بيشتر جملاتی را اجراء می‌کند که اندکس آن جمله‌ها در دامنه (Range) مطابقت داشته باشد. شکل عمومی دستور حلقة WHILE طور ذيل است:

[label] WHILE condition LOOP

statements

END LOOP [label];

طوری که در شکل عمومی دستور حلقة WHILE دیده می‌شود، یگانه قسمت اختياری عبارت از لیبل بوده که در صورت استفاده در آغاز و انجام حلقة باید از عین کلمه برای نام‌گذاری لیبل استفاده شود؛ یعنی یک حلقة WHILE می‌تواند به یک نام مسمی شود و یا هم بدون لیبل باشد.

در حلقة WHILE اگر شرط (condition) صحيح بود، جملات اجرایی داخل حلقة به راه افتاده و کنترول دوباره به ابتدای حلقة برگرد تا دوباره شرط داده شده (condition) ارزیابی شود. تكرار اين عملية تا زمانی پيش مى‌رود که شرط (condition) نفي شود. بعد از نادرست‌بودن شرط، کنترول از حلقة خارج شده

و به دستور بعد از حلقه (LOOP) می‌رود. به خاطر جلوگیری از تکرار پایان ناپذیر یک حلقه WHILE، باید در یک مرحله، حالتی در نظر گرفته شود که شرط داده شده در داخل حلقه، نادرست (FALSE) شود. دستورهای پایین به خاطر ایجاد کردن سکته‌گی در یک حلقه WHILE استفاده می‌شوند:

1. EXIT
2. EXIT WHEN
3. CONTINUE
4. CONTINUE WHEN

توضیحات و مثال‌های لازم در مورد استفاده از دستورهای فوق در یک حلقه WHILE در درس بعدی ارائه شده‌اند.

در بعضی از زبان‌های برنامه‌نویسی، دستورهای فرعی LOOP UNTIL و REPEAT UNTIL استفاده می‌شوند. با این دستورها، شرط ادامه حلقه WHILE در آخر حلقه چک می‌شود؛ یعنی بعد از یک بار اجراء، شرط دیده شده در صورت درست بودن به کنترول، به شروع حلقه می‌رود؛ در حالی که به صورت عمومی بعد از ختم اجراء، کنترول به ابتدای حلقه رفته و بعد از آن شرط چک می‌شود. در حالات استفاده از REPEAT و یا LOOP UNTIL چانس حداقل یک بار اجراء، به دستورهای داخل حلقه داده می‌شود. شکل استفاده از این نوع حلقه WHILE با به کارگیری EXIT WHEN در زبان PL/SQL طور ذیل نوشته می‌شود:

```
LOOP
  statements
  EXIT WHEN condition;
END LOOP;
```

به مثال پایین توجه شود:

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
2   done BOOLEAN := FALSE;
3 BEGIN
4   WHILE done LOOP
5     DBMS_OUTPUT.PUT_LINE ('This line does not print.');
6     done := TRUE; -- This assignment is not made.
7   END LOOP;
8   WHILE NOT done LOOP
9     DBMS_OUTPUT.PUT_LINE ('Hello, world!');
10    done := TRUE;
11  END LOOP;
12 END;
13 /
Hello, world!
PL/SQL procedure successfully completed.
SQL> _
```

شکل ۱۴-۶

در شکل بالا دستورهای حلقه اول هیچ‌گاه اجراء نمی‌شود؛ ولی دستورهای شامل حلقه دوم برای یک بار اجراء می‌شوند.

۶.۵ ساختار و موارد استفاده از دستورهای CONTINUE و EXIT در حلقه‌ها

حلقه‌ها (LOOPS) در زبان‌های برنامه‌نویسی وسیله‌های مؤثر مدیریت دیتا و پروگرام به شمار می‌روند. اگر یک حلقه در یک پروگرام به صورت درست استفاده نشود، امکان دارد باعث بروز مشکلاتی از قبیل متوقف شدن یک سیستم شود. به خاطر استفاده درست از حلقه‌ها در زبان‌های پروگرام‌نویسی، دستورهای کنترولی وجود دارند تا به واسطه آن‌ها در زمان ضرورت به یک حلقه پایان داده شود و یا به ادامه حلقه اجازه داده شود. این دستورها هم به صورت مطلق استفاده می‌شوند و هم به صورت مشروط، با ذکر شرط‌هایی استفاده می‌شوند.

دستورهایی که به خاطر خارج شدن از یک حلقه (LOOP) استفاده می‌شوند، عبارت‌اند از:

۱. خارج شدن مطلق (EXIT)
۲. خارج شدن مشروط (EXIT WHEN)

به عین ترتیب دستورهایی که به خاطر خارج شدن از تکرار فعلی در یک حلقه (LOOP) و اجازه‌دادن برای ادامه دستورهای بعدی عبارت‌اند از:

۱. ادامه (CONTINUE)
۲. ادامه مشروط (CONTINUE WHEN)

هر چهار دستور فوق تنها در بین یک حلقه قابل استفاده هستند و در بیرون از حلقه‌ها استفاده دستورهای یادشده مجاز نیست.

۶.۵.۱ دستورهای خارج شدن از حلقه

دستور EXIT به خاطر خارج شدن از حالت تکرار دستورهای حلقه استفاده می‌شود. زمانی که این دستور استفاده می‌شود، بدون این که شرط نارمل نفی‌شدن در نظر گرفته شود، به حلقه پایان داده می‌شود. با به کارگیری دستور EXIT در یک حلقه، پروسه‌های آن توقف داده شده و کنترول به دستور بعد از حلقه در پروگرام زبان PL/SQL بر می‌گردد. به خاطر وضاحت بیشتر به مثال پایین توجه شود.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   x NUMBER := 0;
 3   BEGIN
 4     LOOP
 5       DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
 6       x := x + 1;
 7       IF x > 3 THEN
 8         EXIT;
 9       END IF;
10     END LOOP;
11   -- After EXIT, control resumes here
12   DBMS_OUTPUT.PUT_LINE('After loop: x = ' || TO_CHAR(x));
13 END;
14 /
Inside loop: x = 0
Inside loop: x = 1
Inside loop: x = 2
Inside loop: x = 3
After loop: x = 4
PL/SQL procedure successfully completed.
SQL>

```

شکل ۱۵-۶

طوری که در شکل بالا دیده می‌شود، یک حلقه عمومی در برنامه PL/SQL با استفاده از دستور IF THEN استفاده شده است. در این پروگرام، متغیر x با قیمت صفر تعریف گردیده و به ترتیب، یک واحد در هر بار اجراء به آن افزوده می‌شود. زمانی که قیمت متغیر بیشتر از ۳ می‌شود، دستور EXIT در سطر ۸ اجرا شده و به حلقه پایان داده می‌شود. در نتیجه‌یی که به دست آمده است، دیده می‌شود که بعد از چهار بار اجراء شدن دستور داخل حلقه، دستور بیرونی (سطر ۱۲ پروگرام) اجرا شده است.

دستور خارج شدن از حلقه با اضافه کردن شرط‌هایی به خاطر کنترول بیشتر روی پروگرام به شکل EXIT WHEN... نیز استفاده شده می‌تواند. عمل کرد دستور EXIT WHEN مانند دستور EXIT بوده، با تفاوت این که شرط بعد از WHEN باید صحیح باشد تا کنترول از حلقه خارج شود. در مثال پایین این دستور استفاده شده است.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   x NUMBER := 0;
 3   BEGIN
 4     LOOP
 5       DBMS_OUTPUT.PUT_LINE('Inside loop: x = ' || TO_CHAR(x));
 6       x := x + 1; -- prevents infinite loop
 7       EXIT WHEN x > 3;
 8     END LOOP;
 9   -- After EXIT statement, control resumes here
10   DBMS_OUTPUT.PUT_LINE('After loop: x = ' || TO_CHAR(x));
11 END;
12 /
Inside loop: x = 0
Inside loop: x = 1
Inside loop: x = 2
Inside loop: x = 3
After loop: x = 4
PL/SQL procedure successfully completed.
SQL>

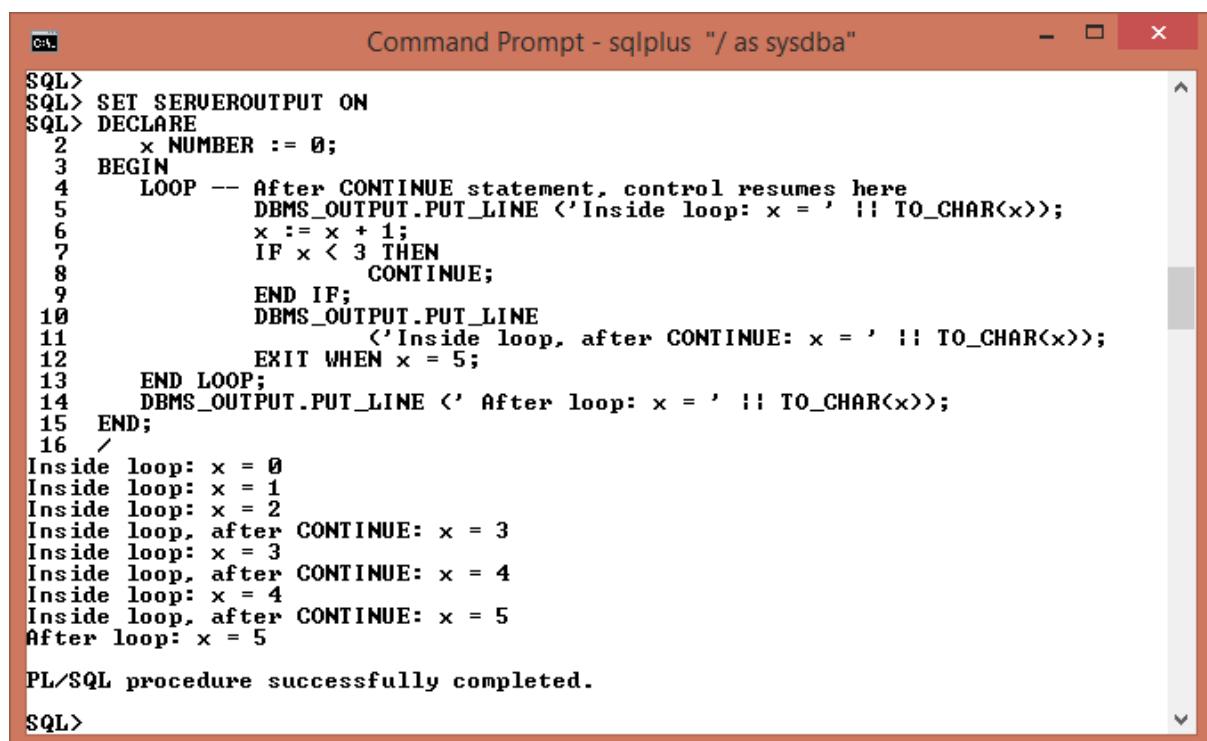
```

شکل ۱۶-۶

طوری که در شکل دیده می‌شود، نتیجه مشابه به مثال قبلی به دست آمده است و در این مثال از دستور EXIT WHEN در سطر 7 استفاده شده است. چون شرط کنترول حلقه در این دستور به کار رفته است، بناءً ضرورت به استفاده از دستور IF THEN در آن نیست.

۶.۵.۲ دستورهای ادامه‌دادن حلقه

دستور CONTINUE جهت پایان‌دادن به دوره فعلی اجرای دستورهای داخل حلقه بدون درنظرگرفتن شرایط استفاده می‌شود. با استفاده از این دستور، کنترول به دوره بعدی داخل همان حلقه و یا هم به حلقه داخلی دیگری، اگر استفاده شده باشد، برمی‌گردد. به خاطر وضاحت بیشتر موضوع، مثال پایین در نظر گرفته شده است.



```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   x NUMBER := 0;
 3 BEGIN
 4   LOOP -- After CONTINUE statement, control resumes here
 5     DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
 6     x := x + 1;
 7     IF x < 3 THEN
 8       CONTINUE;
 9     END IF;
10    DBMS_OUTPUT.PUT_LINE
11      ('Inside loop, after CONTINUE: x = ' || TO_CHAR(x));
12    EXIT WHEN x = 5;
13  END LOOP;
14  DBMS_OUTPUT.PUT_LINE ('After loop: x = ' || TO_CHAR(x));
15 END;
16 /
Inside loop: x = 0
Inside loop: x = 1
Inside loop: x = 2
Inside loop, after CONTINUE: x = 3
Inside loop: x = 3
Inside loop, after CONTINUE: x = 4
Inside loop: x = 4
Inside loop, after CONTINUE: x = 5
After loop: x = 5

PL/SQL procedure successfully completed.

SQL>

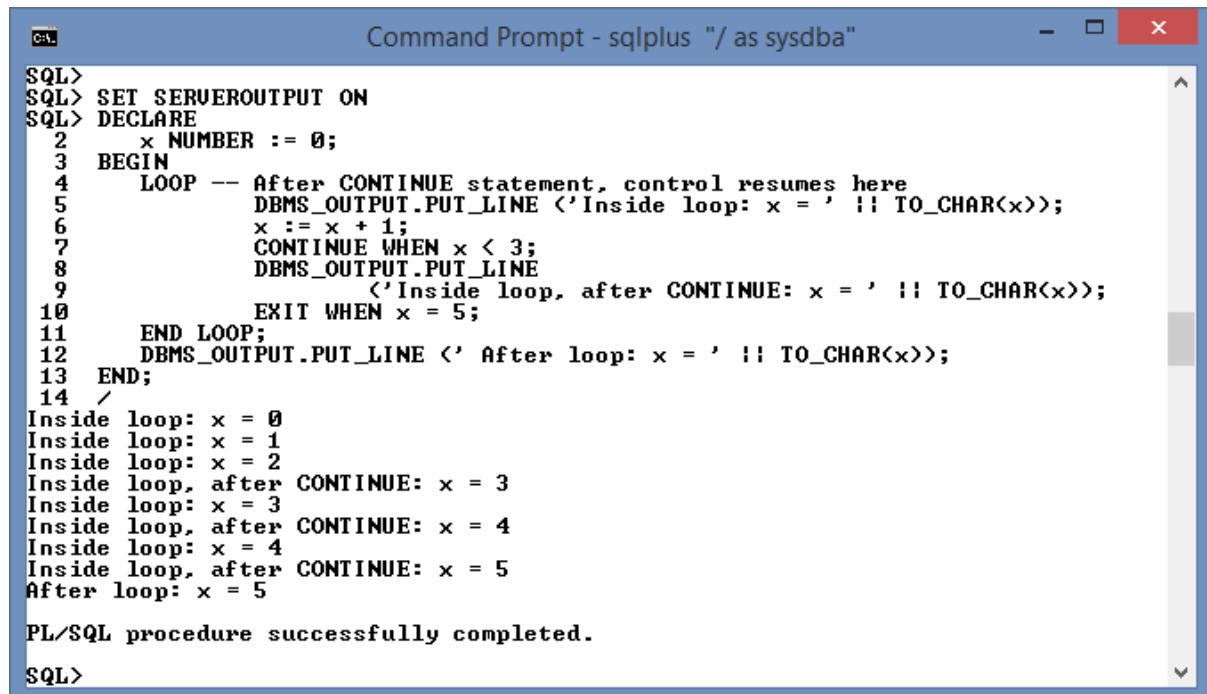
```

شکل ۱۷-۶

طوری که در شکل دیده می‌شود، دستور CONTINUE بدون کدام شرط در سطر 8 استفاده شده است. شرط کوچک‌تر بودن متغیر *x*، در دستور قبلی آن در سطر 7 گنجانیده شده است. طوری که در نتیجه پروگرام دیده می‌شود، سه دوره دستور حلقه به اجراء در آمده است. بعد از خارج شدن از شرط IF THEN ای دستور سطر 12، دستورهای داخل حلقه برای سه بار دیگر ای پوره شدن شرط وضع شده در این سطر اجراء شده‌اند. در نهایت بعد از درست شدن شرط سطر 12 کنترول از حلقه خارج شده و سطر 14 اجراء می‌شود.

دستور پایان‌دادن دوره فعلی اجرای دستورهای داخل حلقه، مانند دستور خارج شدن از حلقه که قبلاً توضیح شد، با اضافه کردن شرایطی به خاطر کنترول بیشتر روی پروگرام به شکل... CONTINUE WHEN...

نیز قابل استفاده است. عملکرد دستور CONTINUE WHEN مانند دستور WHEN است، با این تفاوت که شرط بعد از WHEN باید صحیح باشد تا دستور CONTINUE اجرا شود. در مثال پایین دستور CONTINUE WHEN استفاده شده است.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The window contains the following PL/SQL code:

```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      x NUMBER := 0;
 3  BEGIN
 4      LOOP -- After CONTINUE statement, control resumes here
 5          DBMS_OUTPUT.PUT_LINE ('Inside loop: x = ' || TO_CHAR(x));
 6          x := x + 1;
 7          CONTINUE WHEN x < 3;
 8          DBMS_OUTPUT.PUT_LINE
 9              ('Inside loop, after CONTINUE: x = ' || TO_CHAR(x));
10          EXIT WHEN x = 5;
11      END LOOP;
12      DBMS_OUTPUT.PUT_LINE ('After loop: x = ' || TO_CHAR(x));
13  END;
14 /
Inside loop: x = 0
Inside loop: x = 1
Inside loop: x = 2
Inside loop, after CONTINUE: x = 3
Inside loop: x = 4
Inside loop: x = 4
Inside loop, after CONTINUE: x = 5
After loop: x = 5
PL/SQL procedure successfully completed.
SQL>
```

شکل ۱۸-۶

طوری که در شکل بالا دیده می‌شود، دستور CONTINUE WHEN در سطر ۷ استفاده شده و شرط کم‌بودن قیمت متغول از عدد معین استفاده شده است. نتیجه این مثال، مشابه به نتیجه مثال قبلی است یگانه تفاوت بین‌شان، عبارت از استفاده کردن دستور ادامه مشروط (CONTINUE WHEN) در این پروگرام است.



خلاصه فصل ششم

در فصل ششم به موضوعات مهم و عملی در قسمت ساختارهای کنترولی کد زبان PL/SQL پرداخته شده است. انواع ساختارهای کنترولی در زبان‌های پروسیجری به صورت عمومی در سه دسته تقسیم‌بندی شده که شامل ساختارهای انتخاب‌ها (Selection)، ساختارهای تکرار (Iteration) و ساختارهای ترتیب‌ها (Sequences) می‌شود. دو ساختار ذکر شده در جمله قبلی، در این فصل به بحث گرفته شده‌اند و سومی آن که عبارت از ساختارهای ترتیب‌ها است، در فصل‌های بعدی کتاب توضیح داده خواهد شد.

در گروپ ساختارهای انتخاب‌ها، ساختار و موارد استفاده از دستور IF و حالت‌های مختلف آن مانند IF و نیز دستور CASE با ذکر مثال‌ها توضیح شده‌اند. در گروپ ساختارهای تکرار (Iteration) حلقه‌های FOR و WHILE در عناوین مستقل توضیح داده شده‌اند. ساختارهای کنترولی که به خاطر کنترول کردن حلقه‌ها در زبان PL/SQL استفاده می‌شوند نیز در بخش آخر فصل با مثال‌های لازم تشریح شده‌اند.

سوالات فصل ششم



۱. در زبان‌های پروگرامنویسی پروسیجری مانند PL/SQL ساختارهای کنترولی به صورت عمومی به چند نوع‌اند؟ توضیح دهید.
۲. حلقه‌ها (Loops) در زبان PL/SQL کدام‌ها می‌باشند؟ نام بگیرید.
۳. شکل عمومی دستور IF THEN را بنویسید.
۴. شکل عمومی دستور IF THEN ELSE را بنویسید.
۵. شکل عمومی دستور IF THEN ELSIF را بنویسید.
۶. دستور CASE ساده با دستور CASE جستجوشده چه تفاوت دارد؟ با مثال واضح کنید.
۷. شکل عمومی دستور CASE ساده را بنویسید.
۸. شکل عمومی دستور CASE جستجوشده را بنویسید.
۹. عناصر حلقه‌ها (LOOPS) را در زبان PL/SQL نام ببرید.
۱۰. تفاوت استفاده و یا عدم استفاده از اختیار Reverse FOR را در یک حلقة FOR توضیح دهید.
۱۱. شکل عمومی حلقة اساسی (Basic LOOP) را در زبان PL/SQL بنویسید.
۱۲. شکل عمومی حلقة FOR را در زبان PL/SQL بنویسید.
۱۳. شکل عمومی حلقة WHILE را در زبان PL/SQL بنویسید.
۱۴. کدام دستورها به خاطر ایجاد کردن سکته‌گی در حلقه‌های FOR و WHILE استفاده می‌شوند؟ صرف نام ببرید.
۱۵. دستورهای فرعی LOOP UNTIL و REPEAT UNTIL را در ارتباط با حلقة WHILE در زبان PL/SQL توضیح دهید.
۱۶. تفاوت بین دستورهای EXIT و WHEN EXIT را در قبال حلقه‌های زبان PL/SQL بیان کنید.
۱۷. تفاوت بین دستورهای CONTINUE WHEN و WHEN CONTINUE را در قبال حلقه‌های زبان PL/SQL بیان کنید.

فعالیت های فصل ششم



۱. با استفاده از دستور "sqlplus "/ as sysdba" برنامه سیکویل پلس را از کوماند پرامپت کمپیوترتان به راه بیندازید.
۲. پروگرام Notepad را در کمپیوترتان باز کنید تا دستورهای مورد نظر زبان PL/SQL را در آن نوشته و تنظیم کنید.
۳. در شروع صفحه Notepad، دستور SET SERVEROUTPUT ON را بنویسید تا پروگرام شما خروجی را نشان بدهد.
۴. یک پروگرام ساده PL/SQL را بنویسید که در آن از دستور کنترولی IF THEN استفاده شده باشد. کد نوشته شده را مطابق مثال های کارشده در فصل ششم، کاپی کرده، به صفحه سیکویل پلس انتقال داده و اجراء کنید.
۵. پروگرام نوشته شده در فعالیت قبلی را با نام مشخص و اکستنشن prg1.sql در کمپیوتر ذخیره کنید و با استفاده از دستور START فایل ذخیره شده را از آدرس آن به سیکویل پلس لود کنید؛ طور مثال (START Y:\PLSQL_Practice\prg1.sql)
۶. یک پروگرام ساده PL/SQL را نوشته و (با استفاده از یکی از طرق بالا) اجراء کنید که در آن از دستور کنترولی IF THEN ELSE استفاده شده باشد.
۷. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از دستور کنترولی IF THEN ELSIF استفاده شده باشد.
۸. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از دستور کنترولی CASE ساده استفاده شده باشد.
۹. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از دستور کنترولی CASE جستجو شده استفاده شده باشد.
۱۰. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از حلقة FOR استفاده شده باشد و ۷ بار دستورهای داخل حلقه اجراء شوند.
۱۱. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از حلقة WHILE استفاده شده باشد.
۱۲. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از حلقة (LOOP) استفاده شده و با دستور EXIT WHEN از ادامه اجراء دستورهای داخل حلقه جلوگیری شده باشد.

فصل هفتم

انواع دیتای ترکیبی (Composite Data Types)



هدف کلی: شاگردان با انواع دیتای ترکیبی آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. هدف استفاده از ریکوردهای PL/SQL را شرح دهند.
۲. ایجاد ریکوردهای تعریف شده توسط استفاده کننده در PL/SQL را بیاموزند.
۳. ایجاد ریکوردها به اساس جدول ها با خصوصیات %ROWTYPE را شرح دهند.
۴. نحوه ایجاد کردن INDEX BY table را شرح دهند.

استفاده و به کارگیری دیتای ترکیبی در زبان PL/SQL به قابلیت‌های این زبان در مدیریت دیتا می‌افزاید. دیتای ترکیبی در کنار ستهای دیتا در زمان کار با پروگرام‌های این زبان استفاده‌کننده را قادر می‌سازد تا عملیه‌های پیچیده و مؤثری را در این زمینه اجراء کند. ایجاد ریکوردهای زبان PL/SQL به شکلی پروگرام می‌شوند که از جریان اجرای بلاک‌های این زبان، دیتا به دست آمده و به اشکال قابل پروسس در سیستم، تبدیل و به دسترس استفاده‌کنندگان قرار می‌گیرند.

ریکوردهای PL/SQL می‌توانند توسط استفاده‌کنندگان یا جدول‌های دیتابیس و یا با استفاده از کسرها تعریف و ایجاد شوند. هر طریقه تحت عنوان جداگانه و با نشان‌دادن مثال‌ها توضیح داده شده است. ایجاد کردن اندکس توسط جدول‌ها با تعریف کردن جوهرهای اندکس‌ها و قیمت‌ها و شرایط استفاده از آن‌ها با ذکر مثالی تشریح شده‌اند.

۷.۱ استفاده از ریکوردهای PL/SQL

یک ریکورد PL/SQL عبارت از ساختمانی است که می‌تواند انواع مختلف اجزای دیتا را نشان دهد. ریکوردهای PL/SQL متشکل از فیلدهای مختلف بوده و شباهت به یک سطر جدول در دیتابیس‌ها دارند؛ طور مثال، معلومات راجع به یک کتاب در کتابخانه، معمولاً عبارت از نام کتاب، نام نویسنده یا مترجم و یا هر دو، موضوع، نمبر شناسایی، سال چاپ و غیره در یک جدول دیتابیس ذخیره می‌شود. یک ریکورد زبان PL/SQL مانند این مورد، معلومات منطقی راجع به یک بخش را با جزئیات آن به شکل فیلدهای ریکورد ذخیره کرده و به شکل بهتر آن را نشان می‌دهد. در زبان PL/SQL انواع مختلف ریکوردها استفاده می‌شوند که قرار ذیل‌اند:

۱. ریکوردهای تعریف‌شده توسط استفاده‌کننده (User defined Records)

۲. ریکوردها به اساس جدول‌ها (Table based Records)

۳. ریکوردها به اساس کسرها (Cursor based Records)

هر کدام از انواع بالا در عناوین جداگانه و با مثال‌ها توضیح شده‌اند.

۷.۲ ایجاد ریکوردهای تعریف‌شده توسط استفاده‌کننده در PL/SQL

در زبان PL/SQL نوعی از ریکورد به نام تعریف‌شده توسط استفاده‌کننده (User defined) می‌تواند ایجاد شود. در این نوع ریکوردها ساختارهای مختلف تعریف می‌شوند. در ساختار این نوع ریکوردها فیلدهای مختلف نیز قابل استفاده است. در یک مثال، ایجاد کردن ریکوردهایی از نوع بالا به خاطر ذخیره کردن معلومات در مورد کتاب‌های یک کتابخانه کوچک در نظر گرفته شود که در آن مشخصه‌های لیست‌شده پایین در مورد کتاب‌ها ذخیره شوند.

- نام کتاب (Title)
- نویسنده (Author)
- موضوع (Subject)
- شماره شناسایی کتاب (Book ID)

تعريف کردن یک ریکورد

نوع (Type) ریکورد در زبان PL/SQL به شکل ذیل تعریف می‌شود:

TYPE

Type_name IS RECORD

```
( field_name1 datatype1 [NOT NULL] [:=DEFAULT EXPRESSION],
  field_name2 datatype2 [NOT NULL] [:=DEFAULT EXPRESSION],
  ...
  field_nameN datatypeN [NOT NULL] [:=DEFAULT EXPRESSION];
record-name type_name;
```

ریکورد کتاب به شکل پایین اعلام (Declare) می‌شود:

DECLARE

TYPE books IS RECORD

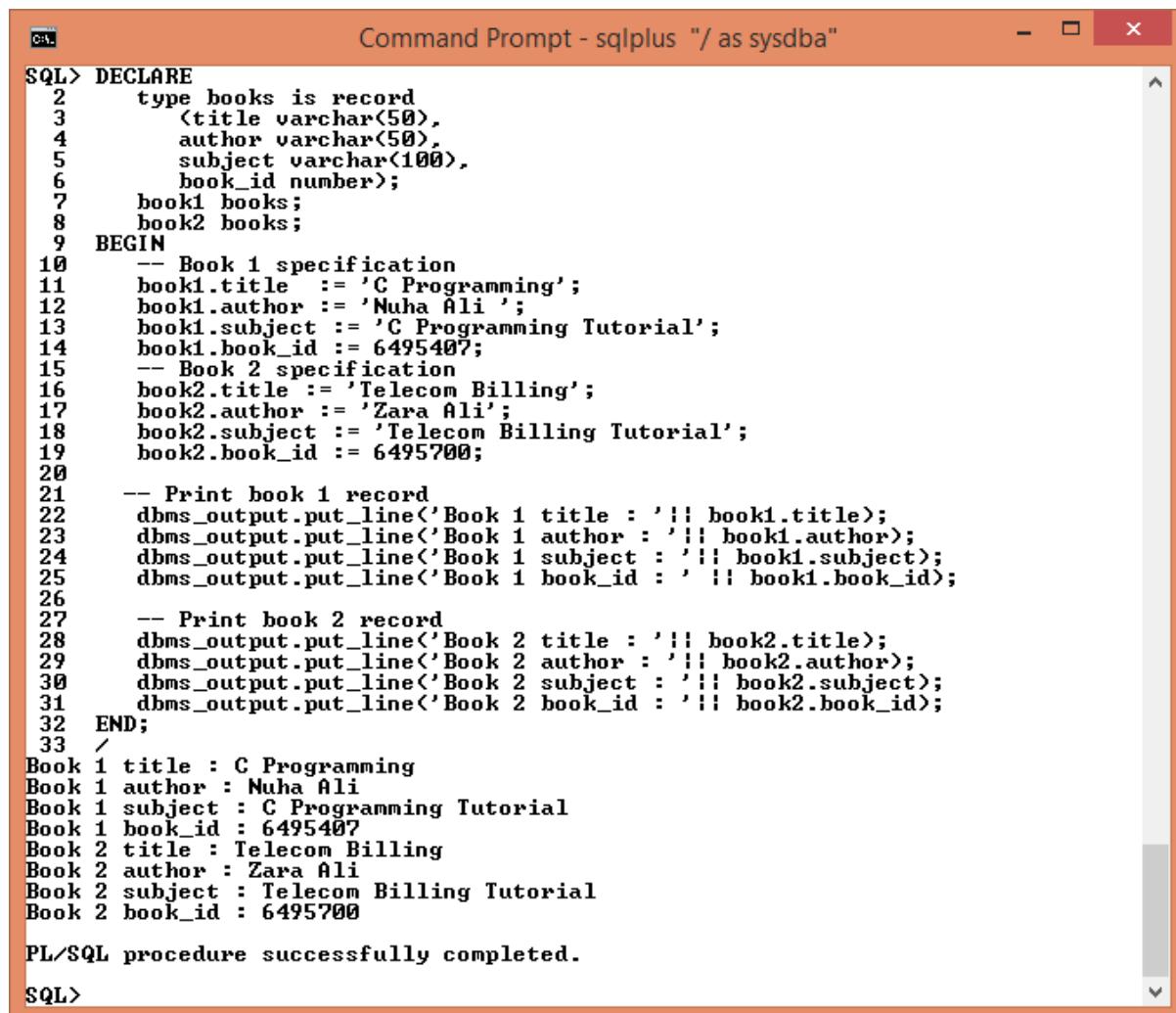
```
(title varchar(50),
  author varchar(50),
  subject varchar(100),
  book_id number);
```

book1 books;

book2 books;

دسترسی به فیلدها

به خاطر دسترسی به هر فیلدی که خواسته شده باشد، از سمبول نقطه(.) استفاده می‌شود. سمبول نقطه در بین نام متتحول ریکورد و نام فیلد مورد نظر به خاطر دسترسی به محتوای آن استفاده می‌شود. در مثال پایین استفاده از نقطه در بین نام متتحول و فیلدهای مشخص نشان داده شده است.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The window contains PL/SQL code and its execution results. The code defines a record type "books" with fields title, author, subject, and book_id. It then declares two variables, book1 and book2, of type books. A BEGIN block sets values for book1 and book2. Another BEGIN block prints the values of book1 and book2 using dbms_output.put_line. The output shows the printed records for both book1 and book2. Finally, it displays a message indicating successful completion of the procedure.

```
SQL> DECLARE
 2      type books is record
 3          <title varchar(50),
 4          author varchar(50),
 5          subject varchar(100),
 6          book_id number;
 7      book1 books;
 8      book2 books;
 9  BEGIN
10      -- Book 1 specification
11      book1.title := 'C Programming';
12      book1.author := 'Nuha Ali';
13      book1.subject := 'C Programming Tutorial';
14      book1.book_id := 6495407;
15      -- Book 2 specification
16      book2.title := 'Telecom Billing';
17      book2.author := 'Zara Ali';
18      book2.subject := 'Telecom Billing Tutorial';
19      book2.book_id := 6495700;
20
21      -- Print book 1 record
22      dbms_output.put_line('Book 1 title : '|| book1.title);
23      dbms_output.put_line('Book 1 author : '|| book1.author);
24      dbms_output.put_line('Book 1 subject : '|| book1.subject);
25      dbms_output.put_line('Book 1 book_id : '|| book1.book_id);
26
27      -- Print book 2 record
28      dbms_output.put_line('Book 2 title : '|| book2.title);
29      dbms_output.put_line('Book 2 author : '|| book2.author);
30      dbms_output.put_line('Book 2 subject : '|| book2.subject);
31      dbms_output.put_line('Book 2 book_id : '|| book2.book_id);
32  END;
33 /
Book 1 title : C Programming
Book 1 author : Nuha Ali
Book 1 subject : C Programming Tutorial
Book 1 book_id : 6495407
Book 2 title : Telecom Billing
Book 2 author : Zara Ali
Book 2 subject : Telecom Billing Tutorial
Book 2 book_id : 6495700

PL/SQL procedure successfully completed.

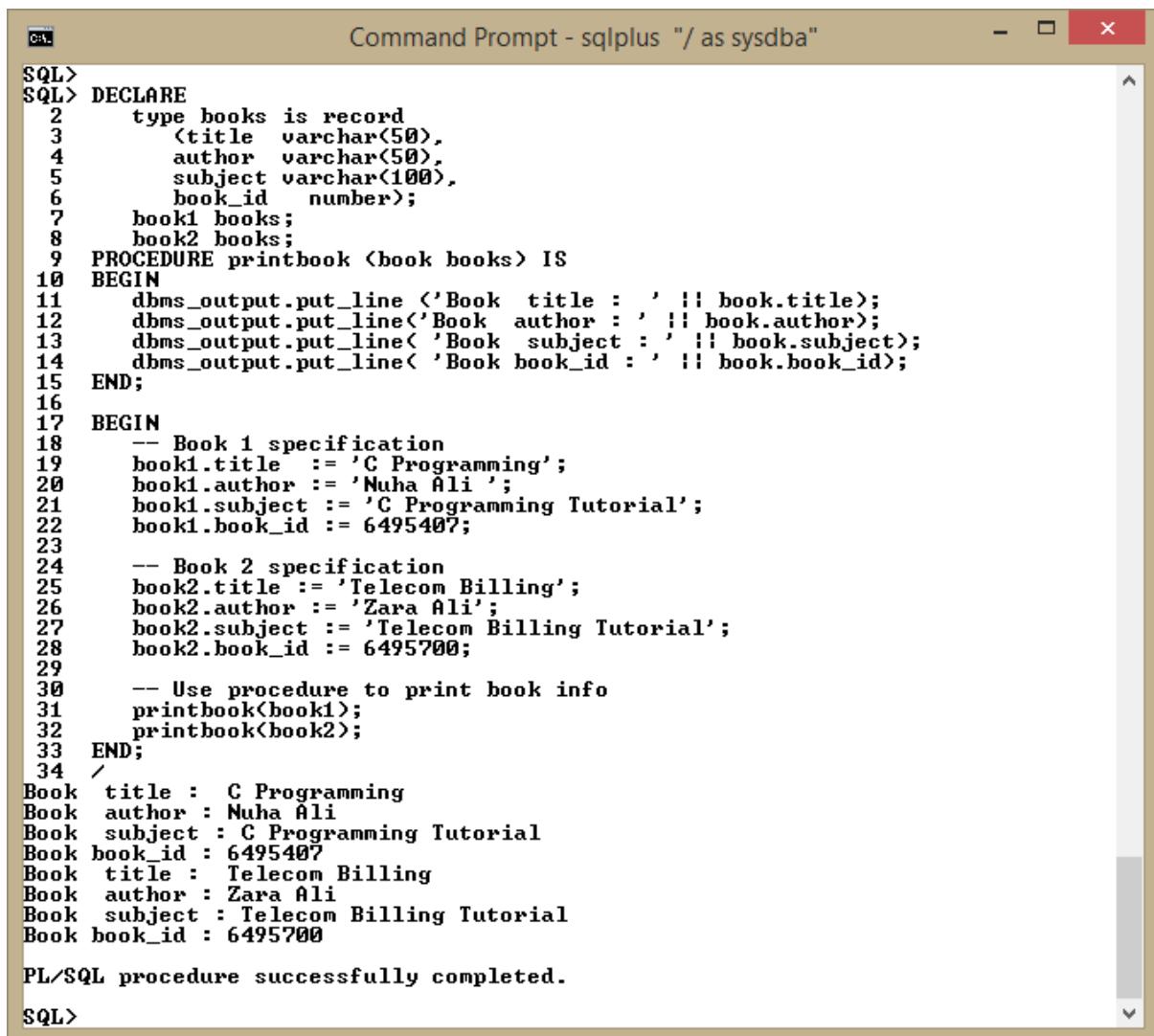
SQL>
```

شکل ۱-۷

طوری که دیده می‌شود، تقریباً در تمامی سطرهای کد بالا، نام متتحولین با استفاده از سمبول نقطه با نام فیلدهای مربوطه ترکیب شده‌اند. در سطرهای 11 الی 19 عناصر دینا به فیلدهای ریکوردها علاوه شده‌اند. در سطرهای 22 الی 31 نتیجه ریکوردهای ثبت‌شده باز هم با ترکیب نام متتحولین (book2 و book1) با استفاده از نقطه با نام فیلدها ترکیب و نتیجه دستورهای خروجی بالای آن‌ها تطبیق شده است. در نهایت پروگرام بالا، خروجی‌های هر دو ریکورد، به شکل فیلدهای جداگانه برای هر کدام چاپ شده‌اند.

ریکوردها به حیث پارامترهای پروگرام فرعی

یک ریکورد در زبان PL/SQL به شکل پارامتر (Parameter) یک پروگرام فرعی استفاده می‌شود. این کار مشابه استفاده از متتحول در ریکورد است. طوری که در مثال قبلی به فیلدهای ریکوردها دسترسی صورت گرفت، به همان شکل فیلدهای ریکوردها در این حالت نیز می‌توانند استفاده شوند؛ به مثال پایین در این زمینه توجه شود:



```
SQL> DECLARE
 2      type books is record
 3          title  varchar(50),
 4          author  varchar(50),
 5          subject  varchar(100),
 6          book_id    number;
 7      book1 books;
 8      book2 books;
 9  PROCEDURE printbook <book books> IS
10  BEGIN
11      dbms_output.put_line ('Book title : ' || book.title);
12      dbms_output.put_line('Book author : ' || book.author);
13      dbms_output.put_line('Book subject : ' || book.subject);
14      dbms_output.put_line('Book book_id : ' || book.book_id);
15  END;
16
17  BEGIN
18      -- Book 1 specification
19      book1.title := 'C Programming';
20      book1.author := 'Nuha Ali';
21      book1.subject := 'C Programming Tutorial';
22      book1.book_id := 6495407;
23
24      -- Book 2 specification
25      book2.title := 'Telecom Billing';
26      book2.author := 'Zara Ali';
27      book2.subject := 'Telecom Billing Tutorial';
28      book2.book_id := 6495700;
29
30      -- Use procedure to print book info
31      printbook(book1);
32      printbook(book2);
33  END;
34 /
Book title : C Programming
Book author : Nuha Ali
Book subject : C Programming Tutorial
Book book_id : 6495407
Book title : Telecom Billing
Book author : Zara Ali
Book subject : Telecom Billing Tutorial
Book book_id : 6495700

PL/SQL procedure successfully completed.

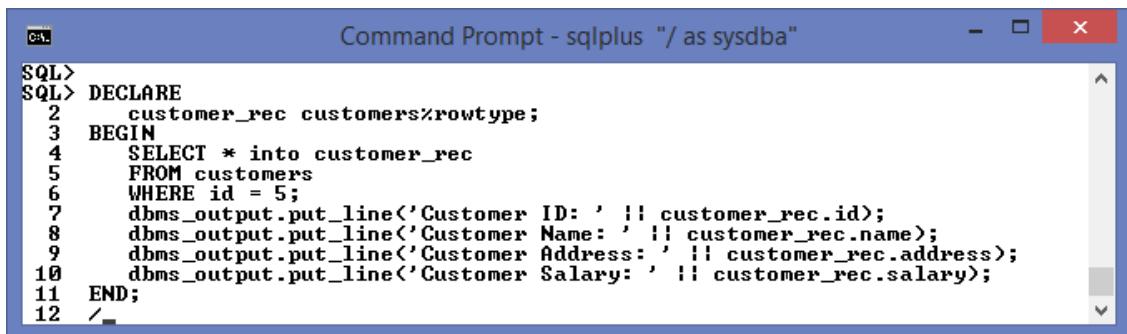
SQL>
```

شکل ۲-۷

۷.۳ ایجاد ریکوردها به اساس جدول‌ها با خصوصیات %ROWTYPE

مشخصه %ROWTYPE یک پروگرام را قادر می‌سازد تا ریکوردهایی را به اساس سطر جدول و یا نمای یک دیتابیس ایجاد کند. این ریکوردها می‌توانند به اساس سطر مکمل یک جدول و یا هم به اساس قسمتی از سطر جدول (تعداد محدودی از فیلدهای یک جدول) ایجاد شود. در چنین مواردی اگر ساختار سطر

جدول تغییر می کند، ساختار ریکورد PL/SQL که به اساس همان سطر مشخص ایجاد شده باشد، نیز تغییر می کند؛ به مثال پایین دقت کنید:



```
SQL> DECLARE
 2   customer_rec customers%rowtype;
 3   BEGIN
 4     SELECT * into customer_rec
 5     FROM customers
 6     WHERE id = 5;
 7     dbms_output.put_line('Customer ID: ' || customer_rec.id);
 8     dbms_output.put_line('Customer Name: ' || customer_rec.name);
 9     dbms_output.put_line('Customer Address: ' || customer_rec.address);
10     dbms_output.put_line('Customer Salary: ' || customer_rec.salary);
11   END;
12 /
```

شکل ۳-۷

دستور فوق در سیکویل پلس اجراء نشده است؛ چون جدولی به نام مشتری باید موجود باشد تا نتیجه به دست آید. در صورت موجودبودن جدول مشتری با دیتای مناسب، اجرای پروگرام بالا این نتیجه را خواهد داشت:

Customer ID: ...

Customer Name: ...

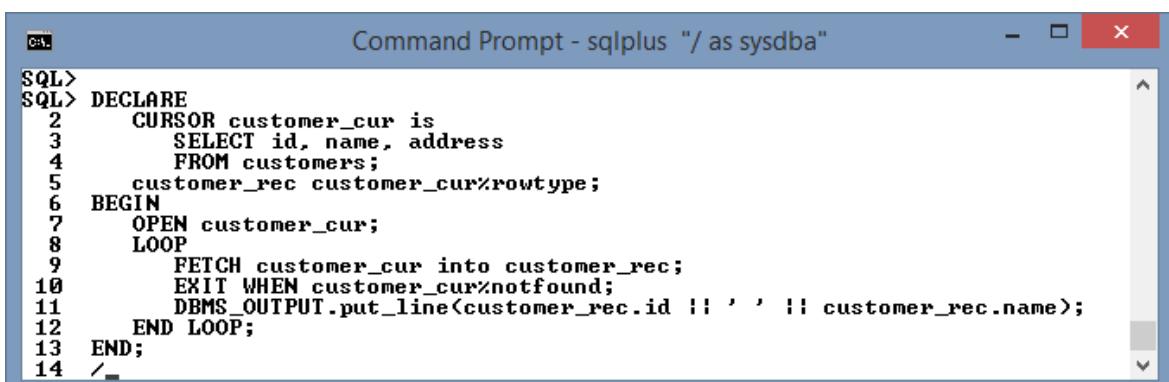
Customer Address: ...

Customer Salary: ...

PL/SQL procedure successfully completed.

۷.۴ ایجاد ریکوردها به اساس کرسوها

مثال پایین مفهوم ایجاد ریکوردها به اساس کرسوها را واضح می سازد. در این مثال نیز ضرورت به جدول مشتری است؛ پس اجرای پروگرام به شکل دستی نشان داده شده است.



```
SQL> DECLARE
 2   CURSOR customer_cur IS
 3     SELECT id, name, address
 4     FROM customers;
 5   customer_rec customer_cur%rowtype;
 6   BEGIN
 7     OPEN customer_cur;
 8     LOOP
 9       FETCH customer_cur INTO customer_rec;
10       EXIT WHEN customer_cur%notfound;
11       DBMS_OUTPUT.put_line(customer_rec.id || ' ' || customer_rec.name);
12     END LOOP;
13   END;
14 /
```

شکل ۴-۷

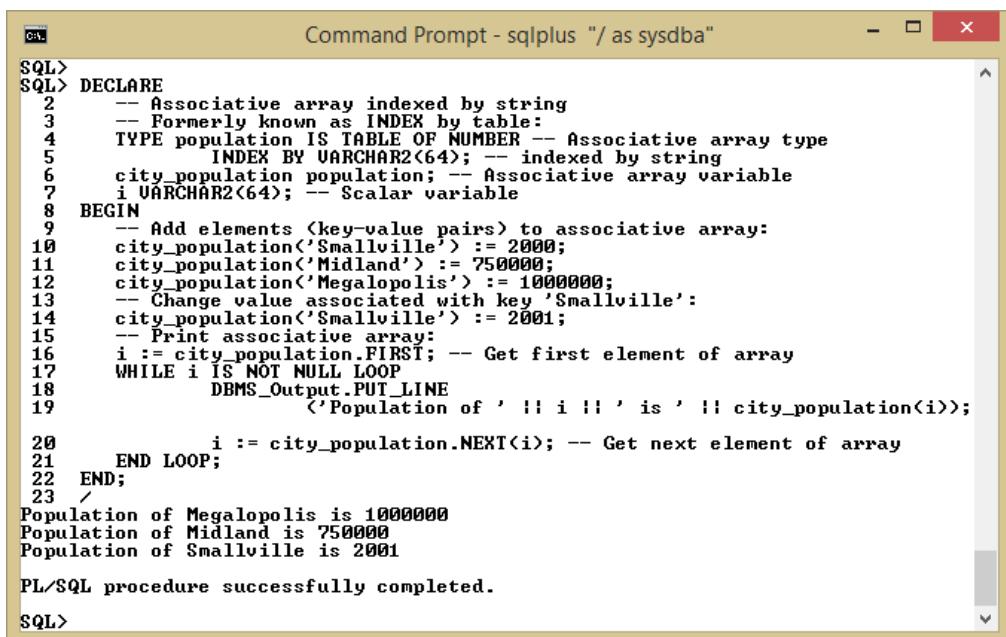
۷.۵ ایجاد کردن INDEX BY table

اصطلاح Index by Table یا هم به نام Associative Array یاد می‌شود و عبارت از سمت جوهرهای کلید و قیمت است. هر کلید یک اندکس غیر تکراری بوده و به خاطر ارتباط به یک قیمت به کار برده می‌شود. شکل نوشتاری این نوع دیتا طور ذیل است:

variable_name(index)

نوعیت دیتای اندکس می‌تواند یکی از انواع رشته‌ها مانند STRING، VARCHAR، VARCHAR2 و یا LONG باشد و یا هم می‌تواند از نوع نمبرها (PLS_INTEGER) باشد. اندکس به اساس زمان ایجاد تنظیم نشده، بلکه به اساس نام‌های شان در سیستم تنظیم می‌شوند.

به مثال پایین توجه شود:



```
SQL> DECLARE
 2   -- Associative array indexed by string
 3   -- Formerly known as INDEX by table:
 4   TYPE population IS TABLE OF NUMBER -- Associative array type
 5       INDEX BY VARCHAR2(64); -- indexed by string
 6   city_population population; -- Associative array variable
 7   i VARCHAR2(64); -- Scalar variable
 8   BEGIN
 9     -- Add elements <key-value pairs> to associative array:
10     city_population('Smallville') := 2000;
11     city_population('Midland') := 750000;
12     city_population('Megalopolis') := 1000000;
13     -- Change value associated with key 'Smallville':
14     city_population('Smallville') := 2001;
15     -- Print associative array:
16     i := city_population.FIRST; -- Get first element of array
17     WHILE i IS NOT NULL LOOP
18       DBMS_Output.PUT_LINE
19         ('Population of ' || i || ' is ' || city_population(i));
20       i := city_population.NEXT(i); -- Get next element of array
21     END LOOP;
22   END;
23 /
Population of Megalopolis is 1000000
Population of Midland is 750000
Population of Smallville is 2001
PL/SQL procedure successfully completed.
SQL>
```

شکل ۷-۷

در شکل بالا یک ایجاد شده است که در آن یک متحول با یک رشته، اندکس شده است. متحول با سه عنصر ترکیب شده، تغییرات به یک عنصر وارد شده و قیمت‌ها چاپ می‌شوند. اگر توجه شود، قیمت‌ها به اساس داخل شدن به سیستم تنظیم نشده، بلکه به اساس نام‌ها تنظیم شده‌اند؛ یعنی نام... Me... در اول، نام... Sm... در دوم و نام... Mi... در قسمت سوم به چاپ رسیده‌اند.



خلاصه فصل هفتم

در این فصل به موضوعات تشریحی در قسمت انواع مختلف دیتای ترکیبی در زبان PL/SQL پرداخته شده است. در عنوان اول، موضوع استفاده از ریکوردهای زبان PL/SQL توضیح شده است و سپس به ترتیب موضوعات ایجاد ریکوردهای تعریف شده توسط استفاده کننده، ایجاد ریکوردها به اساس جدول‌ها با خصوصیت ROWTYPE % و ایجاد ریکوردها به اساس کرسوها در عناوین جداگانه توضیح داده شده‌اند.

در موضوع اول (ایجاد ریکوردهای تعریف شده توسط استفاده کننده) به عناوین فرعی تعریف کردن یک ریکورد، دسترسی به فیلدها و ریکوردها به حیث پارامترهای پروگرام فرعی با ذکر مثال‌ها پرداخته شده است. در قسمت بعدی مسائل مربوط به ایجاد کردن Index by table با ذکر مثال توضیح شده است.



سوالات و فعالیت های فصل هفتم

۱. تفاوت بین ریکوردهای زبان PL/SQL و ریکوردهای جدول‌های دیتابیس را بیان کنید.
۲. در زبان PL/SQL انواع مختلف ریکوردهای قابل استفاده را نام بگیرید.
۳. شکل عمومی نوشتن نوع یک ریکورد را در PL/SQL بنویسید.
۴. سمبل نقطه (.) در دستورهای زبان PL/SQL چه اهمیت دارد و در کجا استفاده می‌شود؟
۵. ایجاد ریکوردها به اساس جدول‌ها با خصوصیات %ROWTYPE را با مثال توضیح دهید.
۶. اصطلاحات معادل برای INDEX BY table کدام‌ها می‌باشند؟ نام بگیرید.

فعالیت ها

۱. مانند فعالیت‌های فصول قبلی، سیکویل پلس و برنامه Notepad را به راه بیندازید.
۲. یک پروگرام ساده PL/SQL بنویسید که در آن با استفاده از دستور TYPE یک ریکورد به نام store با ۳ فیلد نام، شماره و آدرس ایجاد شود.
۳. با استفاده از کد زبان PL/SQL به ریکورد store، دیتا داخل کنید.
۴. با استفاده از دستور خروجی dbms_output.put_line... نتیجه دیتای ثبت شده را دوباره روی صفحه نشان بدهید.

فصل هشتم

کرسرهای آشکار (Explicit Cursors)



هدف کلی: شاگردان با شیوه استفاده کرسرهای آشکار (Explicit Cursors) آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. کرسوها را تعریف کنند.
۲. فرق بین کرسرهای آشکار (Explicit Cursors) و کرسرهای پوشیده یا ضمنی (Implicit Cursors) در سیکویل را شرح دهند.
۳. اهداف استفاده از کرسرهای آشکار را توضیح دهند.
۴. روش ایجاد و کنترول کرسرهای آشکار را بیاموزند.
۵. از دستورهای حلقه (Loop) با کرسها استفاده کنند.
۶. عبارت FOR UPDATE را در قبال کرسرهای آشکار زبان PL/SQL شرح دهند.
۷. عبارت WHERE CURRENT OF را در ارتباط به موضوع قبلی توضیح دهند.

کرسرهای نشانگرها عبارت از ساختمانهایی‌اند که در قسمت کار با دستورهای مربوط به سیکویل استفاده می‌شوند. در یکی از فصول قبلی کتاب به شکل مختصر به این موضوع پرداخته شده است. کرسرهای PL/SQL در زبان PL/SQL به دو شکل تنظیم و استفاده می‌شوند که یک نوع آن‌ها به صورت خودکار توسط PL/SQL در زمان کار با دیتا ایجاد می‌شوند. این نوع کرسرهای پوشیده و یا کرسرهای ضمنی که در انگلیسی به آن‌ها Implicit Cursors می‌گویند، یاد می‌شوند. نوع دومی کرسرهای آشکار (Explicit Cursors) یاد شده و توسط استفاده کنندگان (پروگرامرهای) زبان PL/SQL ایجاد و استفاده می‌شوند. موضوع بحث این فصل بیشتر روی کرسرهای آشکار می‌چرخد.

اهداف استفاده از کرسرهای آشکار و طریقه‌های تعریف کردن، به کارانداختن و گرفتن دیتا از این نوع کرسرهای در این فصل توضیح شده‌اند. استفاده از دستورهای حلقه (LOOP) با کرسرهای آشکار و ایجاد کرسرهای با پارامترها از جمله موضوعات دیگر توضیح داده شده در این فصل‌اند. بعضی عبارت‌های مهم زبان PL/SQL که در ارتباط به تعریف و استفاده از کرسرهای آشکار موجوداند و به غنامندی استفاده از کرسرهای آشکار می‌افزایند نیز مورد بحث قرار گرفته‌اند. از جمله این کتگوری، عبارات WHERE FOR UPDATE و CURRENT OF با ارائه مثال‌هایی توضیح داده شده‌اند.

۸.۱ کرسرهای (Cursors)

طوری که در فصل پنجم کتاب نیز قسمًا به این موضوع پرداخته شده است، کرسرهای (Cursors) عبارت از ساحتی از حافظه کمپیوتراند که در آن‌ها دستورهای کار با دیتا (DML) سیکویل توسط اوریکل تنظیم و اجراء می‌شوند. در برنامه‌نویسی دیتابیس، کرسرهای عبارت از ساختارهای داخلی‌اند که در آن‌ها، اجرای کیوری‌ها صورت گرفته و نتایج متوجه نشان داده می‌شوند. کرسرهای مورد استفاده در یک جلسه (Session) سیکویل به نام کرسرهای جلسه (Session Cursors) یاد می‌شوند. این نوع کرسرهای معلومات را تا زمان ختم جلسه در خود نگه داشته و با بسته‌شدن یک Session استفاده از سیکویل کرسرهای مربوطه آن نیز بسته می‌شوند. کرسرهای PL/SQL به دو نوع کرسرهای آشکار (Explicit Cursors) و کرسرهای پوشیده یا ضمنی (Implicit Cursors) تقسیم می‌شوند.

۸.۲ فرق بین کرسرهای آشکار (Explicit Cursors) و کرسرهای پوشیده (Implicit Cursors)

طوری که گفته شد، در زبان PL/SQL کرسرهای عبارت از ساختمانهایی‌اند که در زمان اجرای دستور کار با دیتا (Data Manipulation Language) به شکل موقت تشکیل شده و جزئیات مربوط به عملیه را در خود ذخیره می‌کنند. کرسرهای در جریان جلسات استفاده از سیستم مدیریت دیتا تشکیل می‌شوند. یک نوع از کرسرهای به صورت خودکار توسط PL/SQL تشکیل می‌شوند و نوع دیگر توسط استفاده کنندگان سیستم طرح شده و استفاده می‌شوند. نوع اول به نام کرسرهای پوشیده و یا ضمنی یاد شده و نوع دوم عبارت از کرسرهای آشکار است.

کرسرهای جلسه که توسط PL/SQL در زمان کار با سیستم ایجاد می‌شوند، به نام کرسرهای پوشیده یا ضمنی (Implicit Cursors) یاد می‌شوند. با اجرای هر دستور DML، زبان PL/SQL یک کرس را به شکل ضمنی ایجاد می‌کند. کرسهایی که توسط استفاده کننده ایجاد می‌شوند، به نام کرسرهای آشکار (Explicit Cursors) یاد می‌شوند. استفاده کردن از محتوای کرسها در هر دو حالت ممکن بوده و توسط پروگرامها به کار برده می‌شوند.

توضیحات در مورد کرسرهای ضمنی یا پوشیده قبلاً در فصل پنجم کتاب گنجانیده شده‌اند. در این فصل بیشتر به کرسرهای آشکار که توسط استفاده کنندگان به شکل دستی (Manual) ایجاد و استفاده می‌شوند، پرداخته شده است.

۸.۳ اهداف استفاده از کرسرهای آشکار

کرسرهای آشکار به شکل دستی (Manual) توسط پروگرامها ایجاد و استفاده می‌شوند. یک کرس آشکار در یک جلسه سیکویل توسط پروگرامر به یک نام مشخص تعریف می‌شود؛ یعنی یک کرس باید نام داشته باشد. هر کرس آشکار در زمان ایجاد باید به یک کیوری ارتباط داده شود. زمانی که این کار به صورت درست انجام شود، نتیجه کیوری به یکی از دو شکل زیر می‌تواند استفاده شود:

- کرس آشکار با دستور OPEN باز شود، سطرهای آن با دستور FETCH گرفته شوند و کرس آشکار با دستور CLOSE بسته شود.
- کرس آشکار در یک حلقة پروگرام مانند FOR LOOP استفاده شود.

هر دو شکل استفاده از نتیجه کیوری در عناوین جداگانه با ذکر مثال‌ها تشریح شده‌اند.

به خاطر گرفتن دیتا از کرسرهای آشکار، از نام آن‌ها استفاده می‌شود. این کار با کرسرهای ضمنی ممکن نیست؛ به همین علت کرسرهای آشکار و یک نوع دیگر از کرسرهای متحول (Variable Cursors) به نام کرسرهای با مسمی (Named Cursors) یاد می‌شوند. کرسرهای متحول خارج از بحث این کتاب‌اند.

مشخصه‌های کرسرهای آشکار نیز مشابه مشخصه‌های کرسرهای ضمنی‌اند. این مشخصه‌ها به خاطر گرفتن معلومات عبارت‌اند از:

- SQL%ISOPEN (آیا کرس باز است؟)
- SQL%FOUND (سطرهای متأثرشده موجوداند؟)
- SQL%NOTFOUND (سطرهای متأثرشده موجود نیستند؟)
- SQL%ROWCOUNT (به چه تعداد سطرهای متأثر شده‌اند؟)
- ...

۸.۴ روش ایجاد و کنترول کرسرهای آشکار

یک پروگرامر در زبان PL/SQL می‌تواند با استفاده از کلمه کلیدی CURSOR یک کرسر از نوع آشکار (Explicit) را ایجاد کند.

```
CURSOR cursor_name IS select_statement;
```

در دستور بالا، قسمت‌هایی که به حروف بزرگ نوشته شده‌اند، عبارت از اجزای دستور بوده و cursor_name نام شناسایی کرسر است. قسمت select_statement عبارت از کیوی انتخاب بوده که کرسر به آن ارتباط داده می‌شود. تمام چهار قسمت بالا به خاطر اجراء کردن کرسر ضروری‌اند. در بلک PL/SQL که دستور ایجاد کردن کرسر آشکار استفاده می‌شود، کرسر باید به همین نام در قسمت اعلامیه پروگرام (Declaration) تعریف شده باشد.

کارکردن با کرسر آشکار شامل چهار اصل پایین می‌شود:

- تعریف کردن کرسرهای اعلامیه پروگرام به خاطر تنظیم حافظه برای آن‌ها: تعریف کردن کرسرهای اعلامیه (Declaration) پروگرام ضروری است. هر کرسر باید به یک نام مسمی شود. کیوی‌بی که کرسر نتیجه آن را استفاده می‌کند نیز باید در این قسمت نوشته شود؛ مثال تعریف کردن یک کرسر به نام c_customers طور ذیل بوده می‌تواند:

```
DECLARE
```

```
CURSOR c_customers IS  
SELECT id, name, address FROM customer;
```

- بازکردن کرسرهای اجرایی پروگرام زبان PL/SQL: بازکردن کرسر (OPEN cursorName) کمک می‌کند که حافظه تخصیص‌داده شده برای کرسر آشکار برای استفاده آماده شود. استفاده از کرسر آشکار، همانا دیدن نتیجه دستورهای DML به اساس کیوی مربوطه و تغییرات وارد شده به جدول است؛ طور مثال سطر کد پایین، کرسر تعریف شده قبلی را باز می‌کند:

```
OPEN c_customers;
```

- دیدن محتوا و گرفتن معلومات از کرسر آشکار که به نام Fetching کرسر یاد می‌شود: توسط دستور Fetching کرسر، پروگرامر می‌تواند در یک زمان تنها یک سطر را مورد دسترسی قرار داده و دیتای آن را استفاده کند؛ طور مثال از کرسر باز شده به نام c_customers مطابق دستور پایین دیتا گرفته می‌شود:

```
FETCH c_customers INTO c_id, c_name, c_addr;
```

- بسته کردن کرسر به خاطر خالی کردن حافظه تعبیه شده برای آن: با بسته کردن کرسر آشکار که قبلاً باز شده باشد، در حقیقت به پروسه بالا خاتمه داده شده و حافظه‌ی که به خاطر کرسر تعریف شده ریزرف شده بود، دوباره به حالت عادی آن بر می‌گردد. مثال بستن کرسر قبلی به نام c_customers طور ذیل است:

```
CLOSE c_customers;
```

البته بستن کرسر یک دستور اختیاری است؛ اگر از آن استفاده نشود، در زمان تکمیل شدن بلاک کد زبان PL/SQL کرسرهای تعریف شده هم بسته شده و حافظه مربوطه تخلیه می‌گردد.

۸.۴.۱ ایجاد کردن کرسرهای آشکار

طوری که گفته شد، کرسرهای آشکار به صورت دستی (Manual) توسط استفاده کنندگان در بلاک کد PL/SQL با یک نام مشخص تعریف و استفاده می‌شوند. شکل کامل تعریف یک کرسر آشکار طور ذیل است:

```
CURSOR cursor_name [parameter_list] [RETURN return_type]
```

```
IS select_statement;
```

در دستور بالا قسمت‌های داخل قوس اختیاری می‌باشند.

یک مثال استفاده از کرسرهای آشکار به اساس کد ذیل در نظر گرفته شده است:

```
DECLARE
```

```
CURSOR c1 RETURN departments%ROWTYPE; -- Declare c1
```

```
CURSOR c2 IS -- Declare and define c2
```

```
SELECT employee_id, job_id, salary FROM employees
```

```
WHERE salary > 2000;
```

```
CURSOR c1 RETURN departments%ROWTYPE IS -- Define c1,
```

```
SELECT * FROM departments -- repeating return type
```

```
WHERE department_id = 110;
```

```
CURSOR c3 RETURN locations%ROWTYPE; -- Declare c3
```

```

CURSOR c3 IS -- Define c3,
SELECT * FROM locations -- omitting return type
WHERE country_id = 'JP';

BEGIN
NULL;

END;
/

```

در مثال بالا، سه کرسر آشکار با نام‌های مشخص تعریف شده‌اند. شکل پایین، صفحه توسعه‌دهنده سیکویل را با کد مذکوره نشان می‌دهد.

```

DECLARE
  CURSOR c1 RETURN departments%ROWTYPE; -- Declare c1
  CURSOR c2 IS -- Declare and define c2
    SELECT employee_id, job_id, salary FROM employees
    WHERE salary > 2000;
  CURSOR c3 RETURN locations%ROWTYPE IS -- Define c1,
    SELECT * FROM departments -- repeating return type
    WHERE department_id = 110;
  CURSOR c3 RETURN locations%ROWTYPE; -- Declare c3
  CURSOR c3 IS -- Define c3,
    SELECT * FROM locations -- omitting return type
    WHERE country_id = 'JP';
BEGIN
  NULL;
END;
/

```

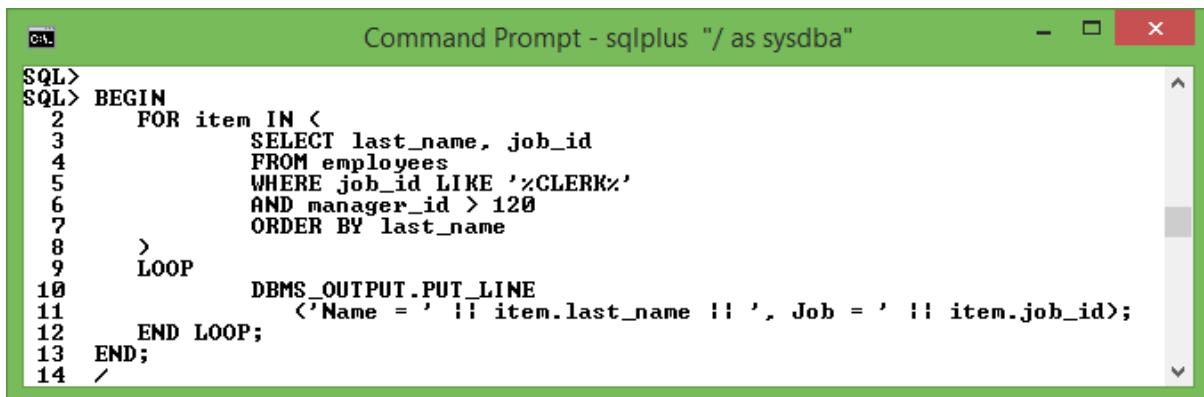
شکل ۱-۸

۸.۴.۲ استفاده از دستورهای حلقه (LOOP) با کرسرهای آشکار

دستوری به شکل CURSOR FOR LOOP در زبان PL/SQL موجود بوده و به خاطر اجراء و تکرار کیوری بالای دیتا در قسمت نتیجه آن استفاده می‌شود. با استفاده از حلقه، با یک دستور، چندین سطر نتیجه یک

کیوری می‌تواند استفاده شود؛ در حالی که در شکل عادی، برای کرسرهای صرف یک سطر مورد استفاده قرار می‌گیرد. استفاده از حلقه‌ها با هر دو شکل کرسرهای پوشیده و کرسرهای آشکار مجاز است.

شکل پایین، کد پروگرام زبان PL/SQL را نشان می‌دهد که در آن، شکل حلقه برای کرسر پوشیده استفاده شده است.



```
SQL> BEGIN
 2   FOR item IN (
 3     SELECT last_name, job_id
 4     FROM employees
 5    WHERE job_id LIKE '%CLERK%'
 6    AND manager_id > 120
 7    ORDER BY last_name
 8  )
 9  LOOP
10    DBMS_OUTPUT.PUT_LINE
11      '<Name = ' || item.last_name || ', Job = ' || item.job_id||';
12  END LOOP;
13 END;
14 /
```

شكل ۲-۸

در شکل بالا، اجرای دستور CURSOR FOR LOOP با یک کرسر ضمنی نشان داده شده است. دستور حلقه در سطرهای 9 الی 12، تخلص‌ها و شماره شناسایی وظیفه کاتب‌های (Clerks) را نشان می‌دهد که شرط سطر 6 برای‌شان تدقیق شود. طوری که دیده می‌شود، به خاطر استفاده کردن کرسر پوشیده در یک حلقه، ضرورت به تعریف قبلی آن در قسمت اعلامیه پروگرام نیست؛ یعنی کرسر ضمنی به صورت خودکار توسط PL/SQL در زمان کار با دستورهای DML ایجاد می‌شوند.

اگر ضرورت باشد تا کیوری SELECT در چندین قسمت یک پروگرام PL/SQL استفاده شود، بهتر است تا از کرسر آشکار در حلقة (LOOP) استفاده شود. کرسر آشکار در قسمت اعلامیه یک پروگرام با جملات کیوری مربوطه آن تعریف می‌شود. به خاطر استفاده کردن کرسر آشکار در حلقه، دستور CURSOR FOR LOOP به کار گرفته می‌شود. دستور استفاده از این نوع کرسر به نام ذیل یاد می‌شود:

Explicit Cursor FOR LOOP statement

مثال دستور حلقه با کرسر آشکار (Explicit Cursor) در ذیل نشان شده است:

```

SQL> DECLARE
  2   CURSOR c1 IS
  3     SELECT last_name, job_id FROM employees
  4     WHERE job_id LIKE '%CLERK%' AND manager_id > 120
  5     ORDER BY last_name;
  6   BEGIN
  7     FOR item IN c1
  8     LOOP
  9       DBMS_OUTPUT.PUT_LINE
 10      ('Name = ' || item.last_name || ', Job = ' || item.job_id);
 11    END LOOP;
 12  END;
 13 /

```

شکل ۳-۸

نتیجه این مثال، مشابه مثل قبلى است؛ با این تفاوت که در اینجا از کرس آشکار استفاده شده است. کرس c1، طوری که دیده می‌شود، در قسمت اعلامیه پروگرام تعریف شده است. همین کرس آشکار در حلقه FOR در سطرهای 7 الی 11 به کار گرفته شده و نتیجه مشابه مثل قبلى را نشان می‌دهد. کرس c1 در قسمت‌های مختلف همین پروگرام نظر به ضرورت می‌تواند استفاده شود.

یک مثال دیگر مربوط به موضوع در ذیل در نظر گرفته شده است. در این مثال نیز استفاده از کرس آشکار در حلقه پروگرام PL/SQL نشان داده شده است.

```

SQL> DECLARE
  2   c_id customers.id%type;
  3   c_name customers.No.name%type;
  4   c_addr customers.address%type;
  5   CURSOR c_customers IS
  6     SELECT id, name, address FROM customers;
  7   BEGIN
  8     OPEN c_customers;
  9     LOOP
 10       FETCH c_customers INTO c_id, c_name, c_addr;
 11       EXIT WHEN c_customers%notfound;
 12       dbms_output.put_line(c_id || ' ' || c_name || ' ' || c_addr);
 13     END LOOP;
 14     CLOSE c_customers;
 15   END;
 16 /

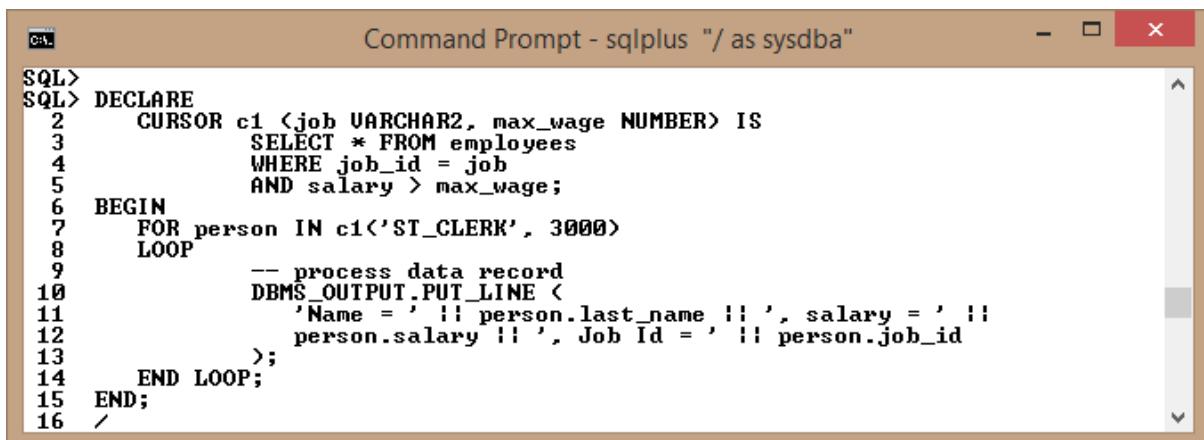
```

شکل ۴-۸

در شکل بالا سه متحول و یک کرس آشکار در قسمت اعلامیه پروگرام تعریف شده‌اند. در قسمت اجرای پروگرام بعد از دستور BEGIN در سطر 7، به طریقه دیگری توسط دستور OPEN کرس تعریف شده، خواسته شده است که باز شود. به تعقیب آن در سطر 9 حلقه (LOOP) باز شده و قیمت‌ها یا محتواهای سه فیلد که در کیوری مربوطه کرس قبلًا خواسته شده‌اند، به فیلد‌های کرس داده می‌شوند. همین پروسه تا آخرین سطر دیتای کیوری شده، ادامه پیدا کرده و در نهایت حلقه به پایان می‌رسد. دستور بعد از حلقه، عبارت از بستن کرس است که در سطر 14 پروگرام بالا نشان داده شده است.

۸.۵ ایجاد کرسوها با پارامترها

کرسوها در PL/SQL با پارامترها طوری ایجاد می‌شوند که در جریان اجراء، قیمت‌ها را از منابع پارامترها به دست آورده و مورد پروسس قرار دهند. این کار با کرسوها آشکار (Explicit Cursors) انجام می‌شود. مثالی در این مورد در نظر گرفته شده است. در این مثال یک کرس آشکار در قسمت اعلامیه پروگرام تعریف شده است. کرس یادشده دو پارامتر را قبول کرده و بعد در قسمت دستور حلقه FOR آن‌ها را به کار می‌برد؛ به مثال پایین توجه شود:



```
SQL> DECLARE
 2    CURSOR c1 (job VARCHAR2, max_wage NUMBER) IS
 3        SELECT * FROM employees
 4        WHERE job_id = job
 5        AND salary > max_wage;
 6    BEGIN
 7        FOR person IN c1('ST_CLERK', 3000)
 8        LOOP
 9            -- process data record
10            DBMS_OUTPUT.PUT_LINE (
11                'Name = ' || person.last_name || ', salary = ' ||
12                person.salary || ', Job Id = ' || person.job_id
13            );
14        END LOOP;
15    END;
16    /
```

شکل ۸-۸

در مثال بالا، کرس آشکار به نام c1 در سطر 2 اعلامیه تعریف شده و به کیوری سطرهای 3 الی 5 ارتباط داده شده است. در قسمت اجرای بلاک PL/SQL در سطر 7 حلقه FOR کرس c1 گرفته شده و به آن قیمت‌های پارامترهای job و max_wage داده شده‌اند. قیمت‌های گرفته شده از پارامترها در کرس آشکار به خاطر نمایش دادن پولی که به کارمندان پرداخته شده است، مورد پروسس قرار می‌گیرد. در نهایت، لیست و مزد (Wages) پرداخته شده برای کارمندانی را چاپ می‌کند که مقدار تعیین شده مزد را در دیپارتمنت‌های مشخص به دست آورده باشند. نتیجه پروگرام بالا، از یک جدول نمونه به نام employees با دیتای معینه آن طور ذیل خواهد بود:

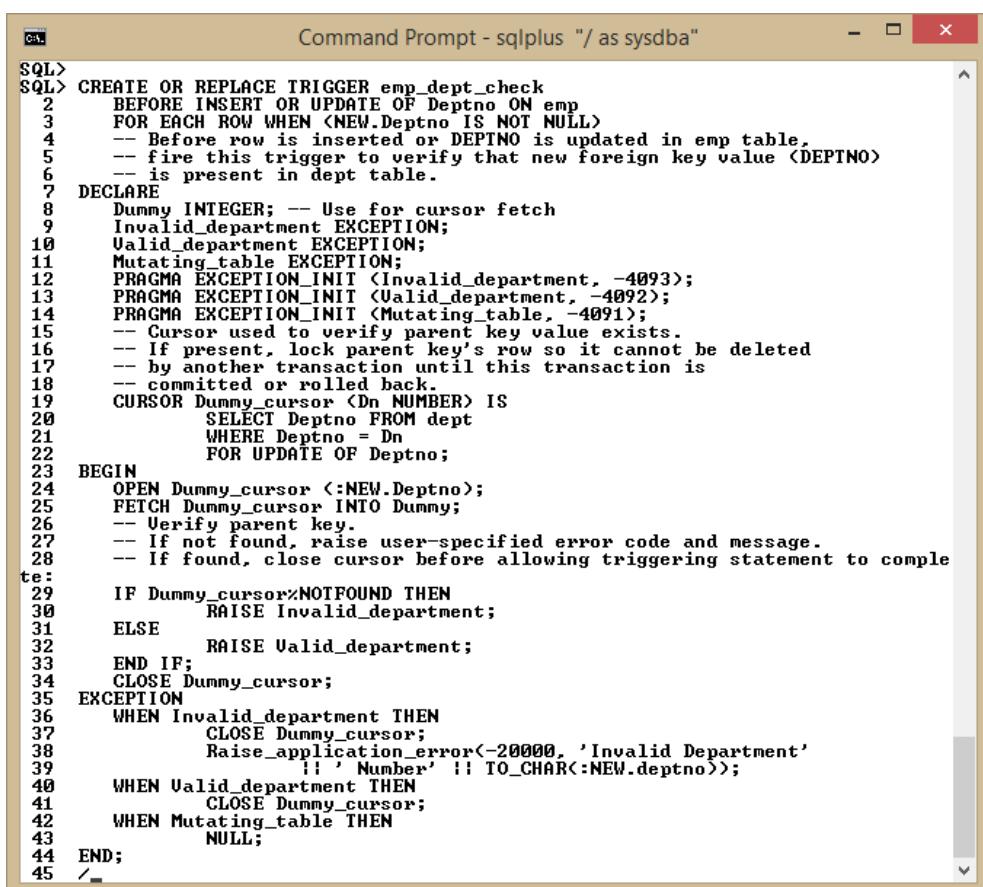
Name = Nayer, salary = 3200, Job Id = ST_CLERK
Name = Bissot, salary = 3300, Job Id = ST_CLERK
Name = Mallin, salary = 3300, Job Id = ST_CLERK
Name = Ladwig, salary = 3600, Job Id = ST_CLERK
Name = Stiles, salary = 3200, Job Id = ST_CLERK
Name = Rajs, salary = 3500, Job Id = ST_CLERK
Name = Davies, salary = 3100, Job Id = ST_CLERK

PL/SQL procedure successfully completed.

۸.۶ استفاده از عبارت FOR UPDATE

دستور SELECT که در کرسوها استفاده می‌شود، با استفاده از عبارت FOR UPDATE سطرهای جدول‌ها را انتخاب کرده و آن‌ها را قفل می‌کند تا استفاده کننده دیگری به اپدیت آن اجازه نداشته باشد. این عملیه تا زمانی که کرس مورد استفاده فعال باشد، قابل تطبیق بوده و بعد از بسته شدن کرس، سطرهای قفل شده دوباره به حالت عادی آن‌ها بر می‌گردند و استفاده کننده دیگر می‌توانند به آن‌ها دسترسی پیدا کنند. این عبارت در بعضی موارد به خاطر قفل کردن محتوای جدول‌ها نیز استفاده می‌شود.

به خاطر وضاحت موضوع به مثال ذیل توجه شود:



```
SQL> CREATE OR REPLACE TRIGGER emp_dept_check
  2>   BEFORE INSERT OR UPDATE OF Deptno ON emp
  3>   FOR EACH ROW WHEN <NEW.Deptno IS NOT NULL>
  4>   -- Before row is inserted or DEPTNO is updated in emp table,
  5>   -- fire this trigger to verify that new foreign key value <DEPTNO>
  6>   -- is present in dept table.
  7>   DECLARE
  8>     Dummy INTEGER; -- Use for cursor fetch
  9>     Invalid_department EXCEPTION;
10>     Valid_department EXCEPTION;
11>     Mutating_table EXCEPTION;
12>     PRAGMA EXCEPTION_INIT (Invalid_department, -4093);
13>     PRAGMA EXCEPTION_INIT (Valid_department, -4092);
14>     PRAGMA EXCEPTION_INIT (Mutating_table, -4091);
15>     -- Cursor used to verify parent key value exists.
16>     -- If present, lock parent key's row so it cannot be deleted
17>     -- by another transaction until this transaction is
18>     -- committed or rolled back.
19>     CURSOR Dummy_cursor (<Dn NUMBER>) IS
20>       SELECT Deptno FROM dept
21>         WHERE Deptno = <Dn
22>           FOR UPDATE OF Deptno;
23>   BEGIN
24>     OPEN Dummy_cursor (:NEW.Deptno);
25>     FETCH Dummy_cursor INTO Dummy;
26>     -- Verify parent key.
27>     -- If not found, raise user-specified error code and message.
28>     -- If found, close cursor before allowing triggering statement to complete;
29>     IF Dummy_cursor%NOTFOUND THEN
30>       RAISE Invalid_department;
31>     ELSE
32>       RAISE Valid_department;
33>     END IF;
34>     CLOSE Dummy_cursor;
35>   EXCEPTION
36>     WHEN Invalid_department THEN
37>       CLOSE Dummy_cursor;
38>       Raise_application_error(-20000, 'Invalid Department'
39>         !! ' Number' !! TO_CHAR(:NEW.deptno));
40>     WHEN Valid_department THEN
41>       CLOSE Dummy_cursor;
42>     WHEN Mutating_table THEN
43>       NULL;
44>   END;
45> /
```

شکل ۸-۸

در سطر 22 شکل بالا، دستور FOR UPDATE OF Deptno قبل از آغاز قسمت اجرای بلاک PL/SQL استفاده شده است. این دستور باعث می‌شود که در جریان اجرای این پروگرام، صلاحیت تغییرآوردن در معلومات Deptno از دیگران سلب شود.

در دستور SELECT FOR UPDATE حالت دیتای مورد نظر طوری تنظیم می‌گردد که الى بسته شدن کرس آشکار که این کیوری با آن ربط دارد، دیتا قفل می‌شود. اگر پروگرامر بخواهد، این دستور را با استفاده از اختیارات آن تغییر داده می‌تواند. اختیارات این دستور عبارت از WAIT, NOWAIT و SKIP LOCKED

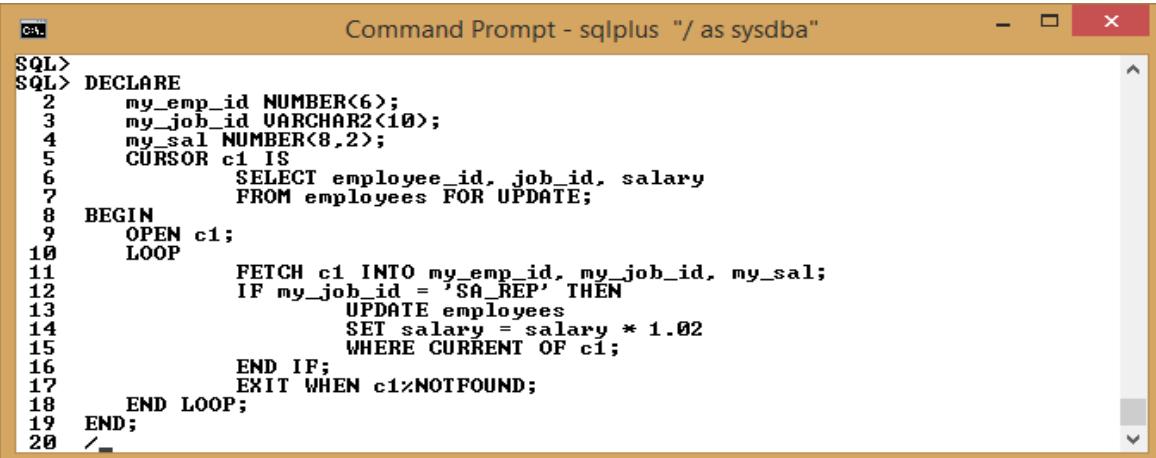
اند؛ یعنی دستور SELECT FOR UPDATE می‌تواند با یکی از اختیارهای توضیح‌داده شده، یک جا استفاده شود.

زمانی که SELECT FOR UPDATE به شکل کیوری ارتباطی با کرسر آشکار استفاده شود، آن را به نام FOR UPDATE CURSOR نیز یاد می‌کنند. در این صورت، تنها همین نوع کرسر می‌تواند در جملات UPDATE و DELETE پروگرام مورد نظر استفاده شود. اگر کیوری SELECT FOR UPDATE با لای چندین جدول اجراء شده باشد، در آن صورت تنها سطرهای متأثرشده از این کیوری قفل می‌شوند. سطرهای خارج از شرایط کیوری قابل به روزآوری و تغییر توسط استفاده کنندگان دیگر می‌باشند.

۸.۷ استفاده از عبارت WHERE CURRENT OF

عبارة بالا، به خاطر به روز کردن (Update) آخرین سطر گرفته شده (Fetched) توسط کرسرهای آشکار به کار برده می‌شود. عبارت WHERE CURRENT OF تنها با عبارت ... FOR UPDATE برای کرسرهای استفاده می‌شود. زمانی که عبارت WHERE CURRENT OF در کرسرهای آشکار استفاده شود، ضرورت استفاده به شرط WHERE از بین می‌رود؛ یعنی از نکات مثبت استفاده کردن آن، جلوگیری از عبارت WHERE در یک دستور اپدیت کرسر است.

جهت وضاحت موضوع به مثال پایین توجه شود:



```
SQL> DECLARE
 2   my_emp_id NUMBER<6>;
 3   my_job_id VARCHAR2<10>;
 4   my_sal NUMBER<8,2>;
 5   CURSOR c1 IS
 6     SELECT employee_id, job_id, salary
 7       FROM employees FOR UPDATE;
 8   BEGIN
 9     OPEN c1;
10    LOOP
11      FETCH c1 INTO my_emp_id, my_job_id, my_sal;
12      IF my_job_id = 'SA_REP' THEN
13        UPDATE employees
14          SET salary = salary * 1.02
15          WHERE CURRENT OF c1;
16      END IF;
17      EXIT WHEN c1%NOTFOUND;
18    END LOOP;
19  END;
20  /
```

شکل ۷-۸

طوری که در شکل بالا دیده می‌شود، در سطر 7 در قسمت اعلامیه، عبارت FOR UPDATE استفاده شده است؛ یعنی با اجرای این پروگرام سطرهای جدول employee قفل شده و الی ختم پروگرام استفاده کنندگان دیگر اجازه آوردن تغییرات در آن‌ها را ندارند. سطر 15 استفاده از دستور WHERE CURRENT OF c1 را نشان می‌دهد. هدف آن خواندن آخرین قیمت کرسر c1 است. در این حالت از شرط WHERE استفاده نشده و این دستور به عین منظور کار می‌کند.



خلاصه فصل هشتم

در این فصل به موضوعاتی در ارتباط با کرسرهای آشکار (Explicit Cursors) در زبان PL/SQL پرداخته شده است. توضیحات مختصری در مورد کرسرهای آشکار، انواع آنها و فرق بین کرسرهای آشکار (Explicit Cursors) و کرسرهای پوشیده (Implicit Cursors) ارائه شده است. اهداف استفاده از کرسرهای آشکار، روش ایجاد و کنترول این نوع کرسرهای مثالهایی توضیح داده شده‌اند. بخش دیگر، مربوط به استفاده کرسرهای آشکار در حلقه‌ها (Loops) بوده و یا به زبان ساده‌تر، استفاده از دستورهای حلقه با کرسرهای آشکار تشریح شده‌اند.

استفاده کردن پارامترها به خاطر دادن دیتا به کرسرهای جهت اجراء در حلقه‌های پروگرام‌های PL/SQL از جمله موضوعات دیگر بحث شده در این فصل اند که با مثالهای لازم به وضاحت آنها پرداخته شده است. بعضی طریقه‌ها به خاطر کنترول دیتایی که بالای آن از طریق کرسرهای کار صورت می‌گیرد و در زمان کار باید استفاده کننده دیگری آن را به کار نگیرد، در PL/SQL موجوداند. این طریقه‌ها با استفاده از عبارت‌های WHERE CURRENT OF FOR UPDATE... و WHERE CURRENT OF FOR UPDATE... کنترول و تنظیم می‌شوند. این دو عبارت با چند مثال، بخش قسمت آخر فصل هشتم هستند.



سوالات و فعالیت های فصل هشتم

۱. از جمله دو نوع کرسر در زبان PL/SQL کدام آن به طور خودکار ایجاد می‌شود؟ واضح سازید.
۲. بر علاوه نام‌گذاری یک کرسر آشکار در زبان PL/SQL، جزء اساسی دیگر آن کدام است؟
۳. کدام دستور PL/SQL به خاطر ایجاد کردن کرسر آشکار استفاده می‌شود؟ شکل عمومی آن را با یک مثال بنویسید.
۴. کدام دستور PL/SQL به خاطر ایجاد کردن کرسر پوشیده استفاده می‌شود؟ شکل عمومی آن را با یک مثال بنویسید.
۵. دستورهای حلقه با کرسرهای CURSOR FOR LOOP را توضیح دهید.
۶. اختیارات قابل استفاده برای دستور SELECT FOR UPDATE در زمان کار کردن با کرسرهای کدامها اند؟ نام ببرید.

فعالیت ها

۱. با مراجعه به فعالیت فصول قبلی، صفحه سیکویل پلس را با استفاده از دستور sqlplus "/ as sysdba" باز کنید.
۲. برنامه Notepad را به خاطر نوشتن و ایدیت کردن کد PL/SQL باز کنید.
۳. پروگرامهای نوشته شده در برنامه Notepad را می‌توانید کاپی و با کلیک راست در صفحه سیکویل Paste کنید و به اجراء در آورید و یا هم می‌توانید با ثبت کردن فایل به یک نام و اکستنشن ".sql" فایل اجرایی آن را ترتیب داده و به اجراء بگذارید.
۴. یک مثال ساده استفاده از کرسر آشکار را نوشته و اجراء کنید.
۵. با استفاده از حلقة (Loop) یک پروگرام ساده بنویسید که در آن از کرسر پوشیده استفاده شده باشد.
۶. با استفاده از حلقة (Loop) یک پروگرام ساده بنویسید که در آن از کرسر آشکار استفاده شده باشد.

فصل نهم

دستورهای بررسی استثناهای (Exception Handling Commands)



هدف کلی: محصلان با سیستم عامل‌های مخصوص سرور آشنایی حاصل کنند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. استثناهای (Exceptions) در زبان PL/SQL را توضیح دهند.
۲. استثناهای بررسی نشده را بشناسند.
۳. انواع مختلف بررسی کننده استثناهای زبان PL/SQL را استفاده کنند.
۴. تأثیر تکثیر کردن استثناهای زبان PL/SQL بر آشیانه‌بیی زبان PL/SQL را شرح دهند.
۵. پیام‌های استثناهای را تنظیم کنند.

پیدا کردن و مدیریت غلطی‌ها در زبان‌های پروگرامنویسی یکی از رازهای کامیابی استفاده‌کنندگان و پروگرامرها به شمار می‌رود. هر زبان پروگرامنویسی طریقه‌هایی را به خاطر انجامدادن این کار مهیا ساخته و در اختیار مردمان مسلکی قرار می‌دهد. زبان PL/SQL نیز به این منظور طریقه‌هایی را دارد که در این فصل به آن‌ها پرداخته شده است. در این زبان به خاطر دیدن، پیدا کردن و مدیریت کردن بهتر غلطی‌ها، دستورهایی موجود است که به نام بررسی استثناهای (Exception Handling Commands) یاد می‌شوند. استثناهای Exceptions در زبان PL/SQL با قواعد مربوطه آن، در این فصل کتاب توضیح داده شده است.

استثناهایی بررسی نشده (Unhandled Exceptions) که در آن مسائل مهمی از قبیل تطبیق بعضی از دستورهای مرتبط شامل می‌شود، به توضیح گرفته شده‌اند. انواع بررسی‌کننده‌های استثناهای، با مثال‌های تحقیکی به بحث گرفته شده‌اند. موضوعات دیگر تحقیکی در ارتباط با استثناهای زبان PL/SQL به ترتیب شامل مدیریت استثناهای در بلاک‌های آشیانه‌یی کد سیکویل و نیز استفاده از تعداد تابع‌ها در این مورد به بحث گرفته شده است. تکثیر و سرایت استثناهای در بلاک‌های آشیانه‌یی زبان PL/SQL به طور اخص و نیز مدیریت پیام‌های استثناهای در این زبان با ذکر مثال‌هایی توضیح داده شده‌اند.

۹.۱ استثناهای PL/SQL (PL/SQL Exceptions)

استثناهای در زبان PL/SQL عبارت از غلطی‌های این زبان در زمان اجرای دستورها است. این غلطی‌ها می‌توانند به اساس دیزاین نادرست پروگرام، کد کردن غیر مسلکی، مشکل سخت‌افزاری و غیره باشد. تمام غلطی‌های یک پروگرام شاید غیر قابل پیش‌بینی باشند؛ اما با استفاده از بررسی‌کننده استثناهای (Exception Handlers) تا اندازه‌یی جلو چنین غلطی‌هایی گرفته می‌شود. قسمت استثناهای در هر بلاک PL/SQL منحیت یک قسمت اختیاری می‌تواند استفاده شود.

با مراجعه به دروس قبلی در فصل‌های اول کتاب، یک بلاک PL/SQL دارای سه قسمت اعلامیه، قسمت اصلی و قسمت استثناهای است. استثناهای به خاطر جلوگیری از غلطی‌های ممکنه به صورت اختیاری در قسمت سوم بلاک PL/SQL استفاده می‌شود. قسمت استثناهای در بلاک PL/SQL می‌تواند یک یا بیشتر از یک بررسی‌کننده استثنای داشته باشد. شکل عمومی بررسی استثناهای در بلاک PL/SQL طور ذیل نوشته می‌شود:

EXCEPTION

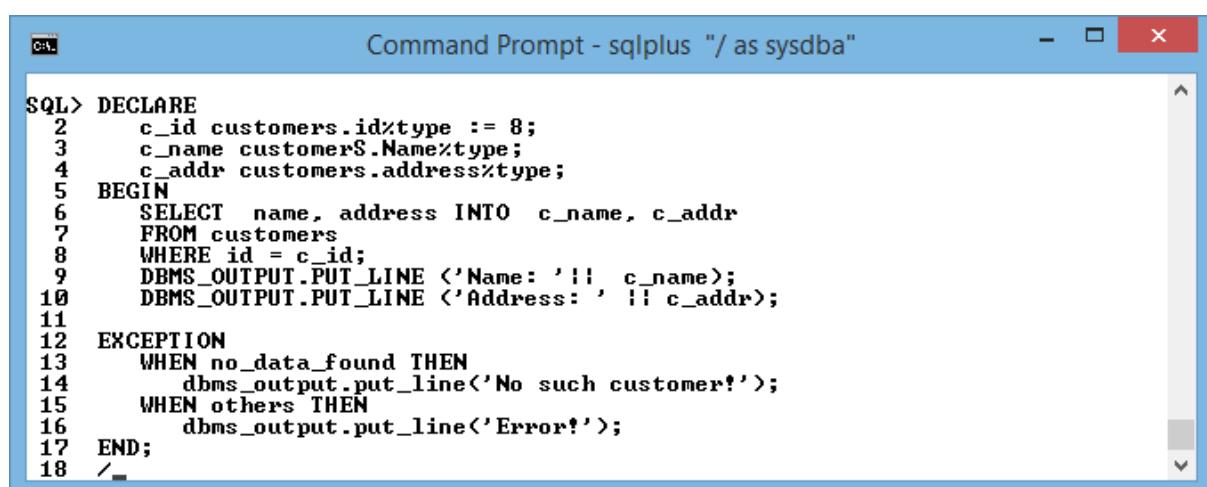
```
WHEN exception_name_1 THEN statements_1 -- Exception handler  
WHEN exception_name_2 OR exception_name_3 THEN statements_2  
WHEN OTHERS THEN statements_3  
END;
```

در دستور بالا که در قسمت استثناهای یک بلاک PL/SQL استفاده می‌شود، کلمه exception_name_n در قسمت اسم استثنای شماره n بوده و کلمه statements_n به ترتیب نام، دستورهای اجرایی را در صورت استثنای لیست شده، به انجام می‌رساند.

زمانی که یک استثنا (Exception) واقع می‌شود، اجرای پروگرام متوقف شده و کنترول پروگرام به این قسمت واگذار می‌شود. اگر exception_name_1 واقع شود، پس دستور statements_1 مطابق متن بالا اجراء می‌شود. در صورتی که exception_name_2 و یا exception_name_3 واقع می‌شود، در این صورت دستور statements_2 اجراء می‌شود. در صورت واقع شدن شکل‌های دیگر استثناهای دستور statements_3 اجراء خواهد شد.

زمانی که یک دستور بررسی کننده استثنا (Exception Handler) اجرا می‌شود، کنترول به جمله اجرایی بعدی به تعقیب بلاک انتقال پیدا می‌کند.

به مثال پایین در این مورد توجه شود:



```
SQL> DECLARE
  2    c_id customers.id%type := 8;
  3    c_name customers.Name%type;
  4    c_addr customers.address%type;
  5  BEGIN
  6    SELECT name, address INTO c_name, c_addr
  7    FROM customers
  8    WHERE id = c_id;
  9    DBMS_OUTPUT.PUT_LINE ('Name: '|| c_name);
10    DBMS_OUTPUT.PUT_LINE ('Address: '|| c_addr);
11
12  EXCEPTION
13    WHEN no_data_found THEN
14      dbms_output.put_line('No such customer!');
15    WHEN others THEN
16      dbms_output.put_line('Error!');
17  END;
18 /
```

شکل ۱-۹

در صورت موجود بودن جدول customers با ساختار و دیتای مناسب، دستور استثنای فوق در قسمت سوم بلاک PL/SQL بین سطرهای 12 الی 16 تطبیق خواهد شد. در صورتی که مشتری با شماره شناسایی 8 در جدول موجود نباشد، قسمت استثناء، اجراء شده و پیام ذیل را چاپ خواهد کرد:

No such customer!

PL/SQL procedure successfully completed.

۹.۲ استثنای بررسی نشده (Unhandled Exceptions)

زمانی که برای کنترول استثنای از بررسی کننده (Handler) استفاده نشده باشد، PL/SQL یک غلطی استثنای بررسی نشده (Unhandled Exception) را به محیط بیرونی نشان می‌دهد. این غلطی، خروجی کد PL/SQL را مشخص می‌کند. اگر یک پروگرام فرعی با استثنای بررسی نشده موجود باشد، PL/SQL دیتابیس را به حالت اول آن Rollback نکرده و تغییرات وارد در دیتابیس حفظ خواهد شد.

استفاده از شرط FORALL، دستورهای DML را برای چندین بار به شکل متواتر با قیمت‌های متفاوت و شرط‌های متفاوت (Different WHEREs) به راه می‌اندازد. اگر یکی از قیمت‌های استفاده شده، باعث غلطی استثنای بررسی نشده (Unhandled Exception) شود، تمام دستورهای اجراء شده DML در این ست، Rollback شده و دیتابیس به حالت اول خود بر می‌گردد.

شرط FORALL به دو شکل تطبیق می‌شود:

- تطبیق FORALL به صورت فوری (Immediate)
- تطبیق FORALL بعد از تکمیل دستورهای آن

هر دو حالت استفاده از دستور FORALL در استثنای بررسی نشده به صورت مختصر توضیح و با مثال‌های در عناوین پایین نشان داده شده‌اند.

۹.۲.۱ تطبیق FORALL به صورت فوری (Immediate)

به خاطر تطبیق دستور FORALL به شکل فوری، کلمه Save Exceptions از دستور دور می‌شود. در این حالت اگر یکی از دستورهای DML باعث غلطی می‌شود، PL/SQL تنها همان دستور را Rollback کرده و دستورهای ماقبل آن را با تغییرات وارد شده حفظ می‌کند.

مسئله تطبیق FORALL به صورت فوری (Immediate) در مثال پایین نشان داده شده است.

```

SQL> DROP TABLE emp_temp;
Table dropped.

SQL> CREATE TABLE emp_temp (
 2      deptno NUMBER(2),
 3      job VARCHAR2(18)
 4  );
Table created.

SQL>
SQL> CREATE OR REPLACE PROCEDURE p AUTHID DEFINER AS
 2      TYPE NumList IS TABLE OF NUMBER;
 3      depts NumList := NumList<10, 20, 30>;
 4      error_message VARCHAR2<100>;
 5  BEGIN
 6      -- Populate table:
 7      INSERT INTO emp_temp (deptno, job) VALUES (10, 'Clerk');
 8      INSERT INTO emp_temp (deptno, job) VALUES (20, 'Bookkeeper');
 9      INSERT INTO emp_temp (deptno, job) VALUES (30, 'Analyst');
10      COMMIT;
11      -- Append 9-character string to each job:
12      FORALL j IN depts.FIRST..depts.LAST
13          UPDATE emp_temp SET job = job || ' Senior';
14          WHERE deptno = depts(j);
15  EXCEPTION
16      WHEN OTHERS THEN
17          error_message := SQLERRM;
18          DBMS_OUTPUT.PUT_LINE (error_message);
19          COMMIT; -- Commit results of successful updates
20      RAISE;
21  END;
22 /

```

Procedure created.

SQL>

شکل ۲-۹

طوری که در شکل دیده می‌شود، در دستور اول یک جدول به نام emp_temp با دو فیلد ساخته شده است. دستور بعدی پروسیجری به نام p ساخته که در آن متغولین depts و error_message از دو نوع مختلف تعریف شده‌اند.

دستور FORALL در سطر 12 شکل بالا، به خاطر به روزکردن (Update) سه جمله استفاده شده است. جمله دومی یک استثناء را به میان می‌آورد. بررسی کننده استثناء (Exception Handler) این مسئله را دنبال کرده و یک پیام غلطی ارائه می‌دارد. همین غلطی در جمله دوم باعث می‌شود تا تغییر جمله اول به صورت فوری (Immediate) تطبیق شده و عملیه به روزکردن، بالای جمله سوم اجراء نشود.

به خاطر به دست آوردن نتیجه، پروسیجر p که قبلاً ایجاد شده است، مطابق شکل پایین به راه انداخته می‌شود.

The screenshot shows a Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The SQL code entered is:

```

SQL> BEGIN
 2   p;
 3 END;
 4 /
BEGIN
*
ERROR at line 1:
ORA-12899: value too large for column "SYS"."EMP_TEMP"."JOB" (actual: 19,
maximum: 18)
ORA-06512: at "SYS.P", line 20
ORA-06512: at line 2

```

Following this, a SELECT query is run:

```

SQL> SELECT * FROM emp_temp;
DEPTNO JOB
-----
 10 Clerk <Senior>
 20 Bookkeeper
 30 Analyst

```

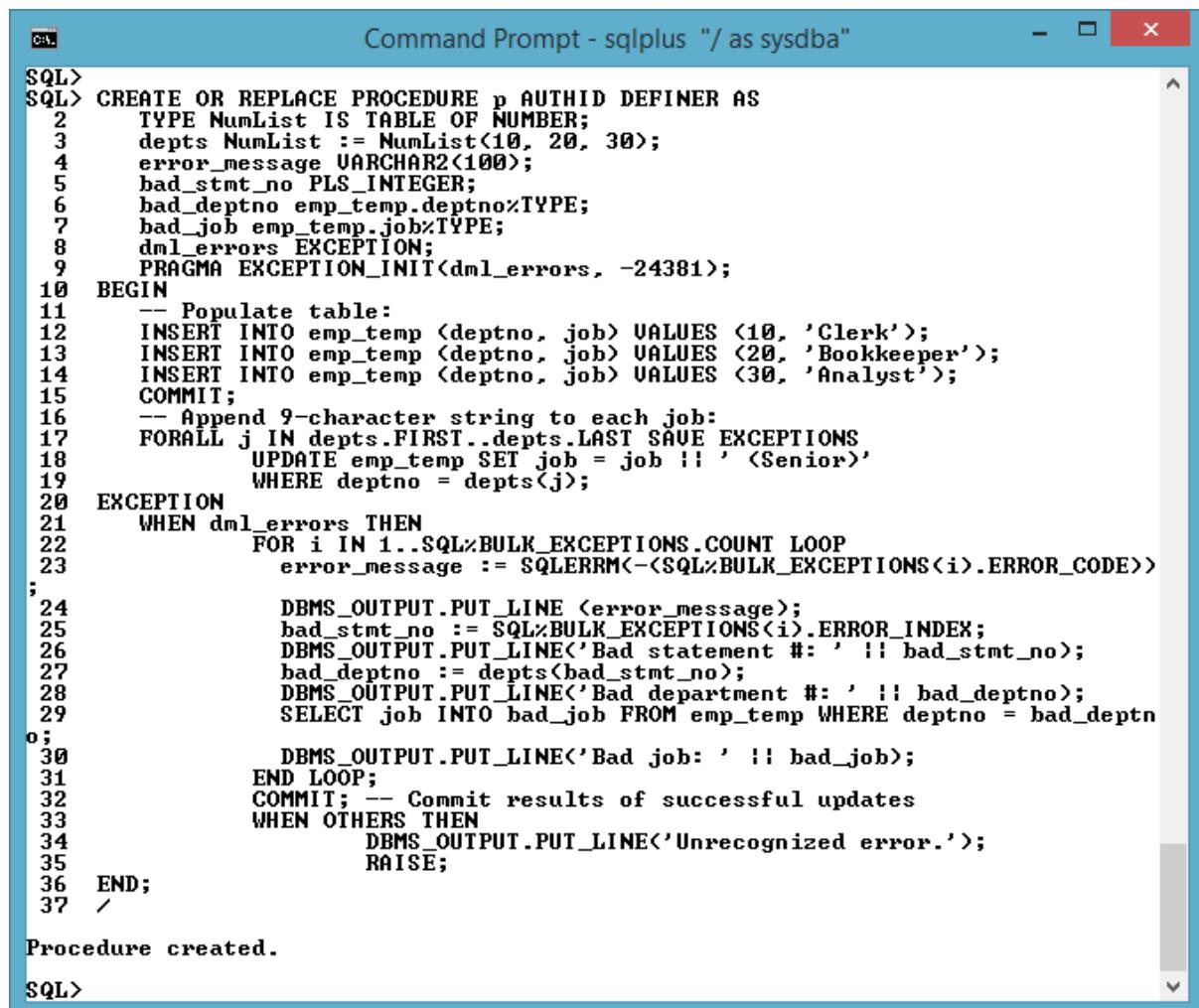
شکل ۳-۹

بعد از اجرای پروسیجر p، کیفیت غلطی نشان داده شده و دیتابیس جدول emp_temp کیوری شده و در نتیجه سه ریکورد نشان داده می‌شود. ریکورد اول اپدیت شده و کلمه Senior به آخر آن اضافه شده است. دو ریکورد دیگر بدون بهروزشدن به حالت خود نشان داده می‌شوند؛ یعنی زمانی که در ریکورد دوم استثناء واقع شده، تغییرات در ریکورد قبلی آن به طور فوری ثبت شده است و ریکورد بعدی بدون واردشدن تغییرات در جدول، به حالت اول آن گذاشته شده است. این مسئله در حالت دیگر استفاده از دستور FORALL که در درس بعدی تشریح شده است، فرق می‌کند.

۹.۲.۲ تطبیق FORALL بعد از تکمیل دستورهای آن

اگر خواسته شود تا دستور FORALL با وجود اجراء نشدن بعضی از دستورهای DML باز هم ادامه پیدا کند، از عبارت SAVE EXCEPTIONS در دستور مورد نظر استفاده می‌شود. در چنین حالت، زمانی که یک دستور DML ناکام می‌شود، PL/SQL استثناء را به صورت فوری اجراء نکرده و اجازه می‌دهد تا تمام عملیه‌های شامل دستور FORALL تکمیل شود. بعد از تکمیل شدن دستورها، PL/SQL یک راپور استثناء (Exception) ارائه می‌کند.

جهت وضاحت موضوع یک مثال دیگر در نظر گرفته شده است:



```
SQL> CREATE OR REPLACE PROCEDURE p_AUTHID DEFINER AS
 2  TYPE NumList IS TABLE OF NUMBER;
 3  depts NumList := NumList<10, 20, 30>;
 4  error_message VARCHAR2<100>;
 5  bad_stmt_no PLS_INTEGER;
 6  bad_deptno emp_temp.deptno%TYPE;
 7  bad_job emp_temp.job%TYPE;
 8  dml_errors EXCEPTION;
 9  PRAGMA EXCEPTION_INIT(dml_errors, -24381);
10 BEGIN
11  -- Populate table:
12  INSERT INTO emp_temp (deptno, job) VALUES <10, 'Clerk'>;
13  INSERT INTO emp_temp (deptno, job) VALUES <20, 'Bookkeeper'>;
14  INSERT INTO emp_temp (deptno, job) VALUES <30, 'Analyst'>;
15  COMMIT;
16  -- Append 9-character string to each job:
17  FORALL j IN depts.FIRST..depts.LAST SAVE EXCEPTIONS
18    UPDATE emp_temp SET job = job || '<Senior>' 
19    WHERE deptno = depts(j);
20 EXCEPTION
21  WHEN dml_errors THEN
22    FOR i IN 1..SQL%BULK_EXCEPTIONS.COUNT LOOP
23      error_message := SQLERRM(<-SQL%BULK_EXCEPTIONS(i).ERROR_CODE>)
24    ;
25      DBMS_OUTPUT.PUT_LINE (error_message);
26      bad_stmt_no := SQL%BULK_EXCEPTIONS(i).ERROR_INDEX;
27      DBMS_OUTPUT.PUT_LINE('Bad statement #: ' || bad_stmt_no);
28      bad_deptno := depts(bad_stmt_no);
29      DBMS_OUTPUT.PUT_LINE('Bad department #: ' || bad_deptno);
30      SELECT job INTO bad_job FROM emp_temp WHERE deptno = bad_deptn
31      DBMS_OUTPUT.PUT_LINE('Bad job: ' || bad_job);
32    END LOOP;
33    COMMIT; -- Commit results of successful updates
34  WHEN OTHERS THEN
35    DBMS_OUTPUT.PUT_LINE('Unrecognized error.');
36    RAISE;
37 END;
Procedure created.
```

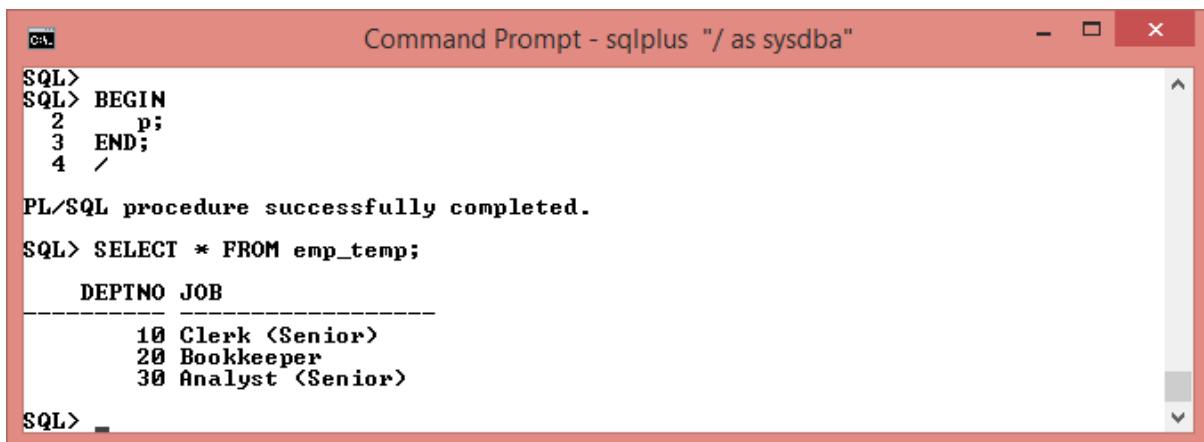
SQL>

شکل ۴-۹

مثال شکل بالا در مواردی مشابه مثال درس قبلی در قسمت استفاده از دستور FORALL با نتایج فوری است؛ ولی در چند مورد ذیل از هم‌دیگر تفاوت دارند:

- در این مثال دستور FORALL عبارت SAVE EXCEPTIONS را در سطر 17 دارد.
- قسمت بررسی استثناء (Exception Handling) در این مثال از بررسی کننده استثناء به نام ORA-24381 استفاده می‌کند. این بررسی کننده توسط استفاده‌کننده تعریف شده و در این مثال به نام dml_errors مسمی ساخته شده است.
- بررسی کننده استثناء برای dml_errors از دو نوع استثناء به نام‌های SQLERRM و SQL%BULK_EXCEPTIONS (Local Variables) بعضی از متحولین محلی (Local Variables) به خاطر نشان دادن پیام‌های غلطی و علت‌های پیام‌های غلطی استفاده می‌کند.

در این مرحله، پروسیجر p در یک بلاک PL/SQL تطبیق شده و نتیجه ذیل را نشان می‌دهد:



```
SQL> BEGIN
 2   p;
 3 END;
4 /
PL/SQL procedure successfully completed.

SQL> SELECT * FROM emp_temp;

DEPTNO JOB
----- -----
 10 Clerk <Senior>
 20 Bookkeeper
 30 Analyst <Senior>

SQL>
```

شکل ۵-۹

طوری که دیده می‌شود، در شکل بالا دو ریکورد اول و سوم جدول emp_temp به روز شده و کلمه Senior به آن‌ها در ستون JOB اضافه شده است؛ یعنی دستور FORALL در این حالت بعد از تکمیل کردن تمام پروسه‌ها و حفظ کردن تغییرات به دیتای جدول، فقط ریکوردی را اپدیت نکرده که در آن غلطی رخ داده است.

۹.۳ انواع بررسی‌کننده‌های استثناهای (PL/SQL Exception Handlers)

بررسی‌کننده‌های استثناهای به سه نوع تقسیم می‌شوند:

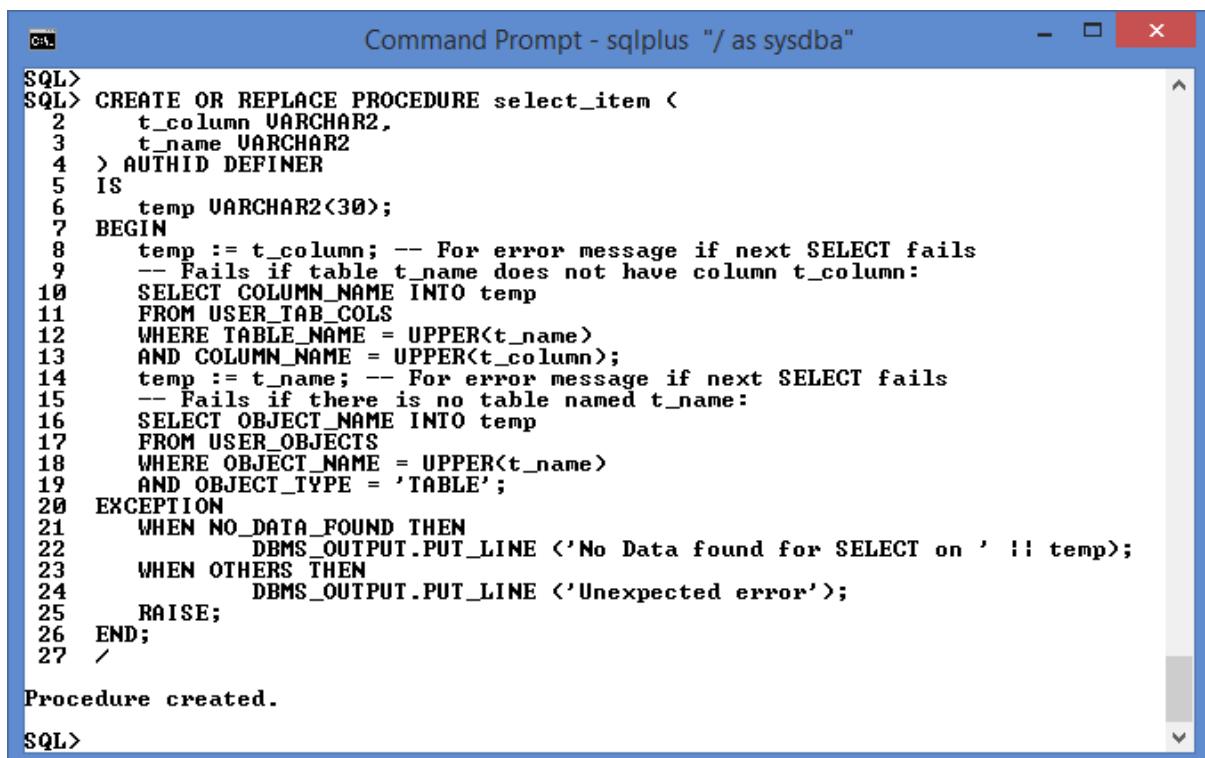
- تعریف شده توسط سیستم (System Defined): این نوع بررسی‌کننده‌ها به صورت خودکار توسط سیستم به شکل پوشیده و یا ضمنی ایجاد می‌شوند. مثال‌های این بررسی‌کننده‌ها عبارت از ORA-00060 و ORA-27102 اند. یک بررسی‌کننده داخلی همیشه یک کد غلطی را نشان می‌دهد. این نوع بررسی‌کننده‌ها در حالت عادی نام ندارند؛ ولی در بعضی حالات امکان دارد از طرف PL/SQL به آن نام داده شود و یا هم استفاده کننده به آن نام بدهد. (به خاطر توضیحات بیشتر به کتاب‌های لیست شده در ریفرنس مراجعه شود).
- از قبل تعریف شده (Predefined): بررسی‌کننده استثناهای نوع از قبل تعریف شده، امکان دارد از نوع قبلی (System defined) بوده و با نام مسمی شده باشد؛ طور مثال ORA-06500 در PL/SQL که به نام storage error یاد می‌شود، اسم از قبل تعریف شده آن عبارت از STORAGE_ERROR است.
- تعریف شده توسط استفاده کننده (User Defined): کاربران سیستم می‌توانند بررسی‌کننده‌های استثناهای (Exception Handlres) مربوط خود را در قسمت اعلامیه (Declaration) پروگرام تعریف کنند؛ طور مثال، یک استثنا به نام insufficient_funds تعریف شود که توسط آن باقی‌داری حساب‌های بانکی نشانی شود. تعریف استثناهای توسط استفاده کننده‌گان باید به صورت آشکار (Explicit) نوشته شده باشند.

دسته‌های استثناهای در جدول پایین تقسیم‌بندی و نشان داده شده‌اند.

Category	Definer	Has Error Code	Has Name	Raised Implicitly	Raised Explicitly
Internally defined	Runtime system	Always	Only if you assign one	Yes	Optionally ¹
Predefined	Runtime system	Always	Always	Yes	Optionally ¹
User-defined	User	Only if you assign one	Always	No	Always

به خاطر وضاحت بیشتر موضوع، مثال‌هایی در نظر گرفته شده‌اند.

در مثال پایین یک بررسی کننده استثناء به خاطر استفاده با چندین استثنا بررسی شده و قرار ذیل است:



```

SQL> CREATE OR REPLACE PROCEDURE select_item (
  2   t_column VARCHAR2,
  3   t_name VARCHAR2
  4 ) AUTHID DEFINER
  5 IS
  6   temp VARCHAR2(30);
  7 BEGIN
  8   temp := t_column; -- For error message if next SELECT fails
  9   -- Fails if table t_name does not have column t_column:
10   SELECT COLUMN_NAME INTO temp
11   FROM USER_TAB_COLS
12   WHERE TABLE_NAME = UPPER(t_name)
13   AND COLUMN_NAME = UPPER(t_column);
14   temp := t_name; -- For error message if next SELECT fails
15   -- Fails if there is no table named t_name:
16   SELECT OBJECT_NAME INTO temp
17   FROM USER_OBJECTS
18   WHERE OBJECT_NAME = UPPER(t_name)
19   AND OBJECT_TYPE = 'TABLE';
20 EXCEPTION
21   WHEN NO_DATA_FOUND THEN
22     DBMS_OUTPUT.PUT_LINE ('No Data found for SELECT on ' || temp);
23   WHEN OTHERS THEN
24     DBMS_OUTPUT.PUT_LINE ('Unexpected error');
25   RAISE;
26 END;
27 /
Procedure created.
SQL>

```

شکل ۶-۹

در مثال بالا، پروسیجری استفاده شده است که در آن یک بررسی کننده استثناء به کار رفته است. در این بررسی کننده، یک استثنای از قبل تعریف شده به نام NO_DATA_FOUND به خاطر واقع شدن در یکی از دو دستور داخل کردن دیتا (SELECT INTO) به کار رفته است.

در مثال بالا، پروسیجر select_item در حالی جدول departments بالای جدول last_name به نام departments موجود باشد؛ ولی این جدول فیلدی به نام last_name نداشته باشد.

BEGIN

```

select_item('departments', 'last_name');

END;
/

```

نتیجه اجرای پروسیجر فوق طور ذیل خواهد بود:

NO Data found for SELECT on departments

مثال بعدی: مطابق شکل پایین، متحولین برای دستورهایی استفاده شده‌اند که در آن‌ها بررسی کننده استثناء شریک ساخته می‌شود.

```

SQL> CREATE OR REPLACE PROCEDURE loc_var AUTHID DEFINER IS
  2      stmt_no POSITIVE;
  3      name_ VARCHAR2(100);
  4  BEGIN
  5      stmt_no := 1;
  6      SELECT table_name INTO name_
  7      FROM user_tables
  8      WHERE table_name LIKE 'ABC%';
  9      stmt_no := 2;
10      SELECT table_name INTO name_
11      FROM user_tables
12      WHERE table_name LIKE 'XYZ%';
13  EXCEPTION
14      WHEN NO_DATA_FOUND THEN
15          DBMS_OUTPUT.PUT_LINE ('Table name not found in query ' || stmt_n
o);
16  END;
17 /
Procedure created.

SQL> CALL loc_var();
Call completed.

SQL>

```

شکل ۷-۹

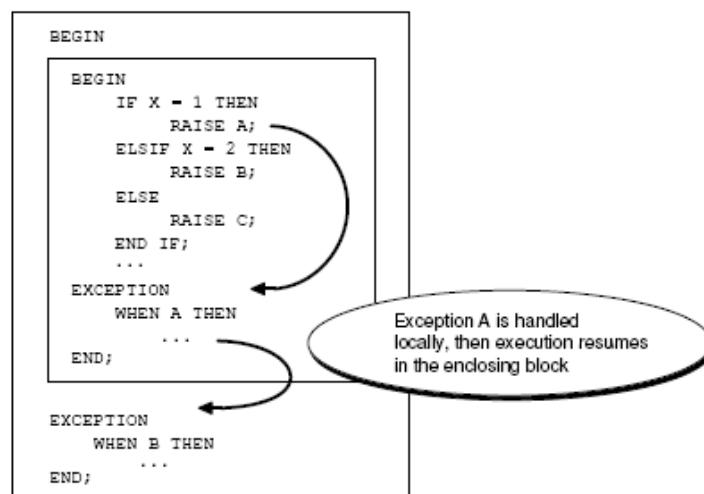
در مثال بالا، موقعیت‌دهنده متحولین به خاطر پیداکردن دستوری که تکمیل نشده است، استفاده می‌شود. در این حالت، دقت پیداکردن غلطی در کد به طریقه‌هایی قابل تنظیم است؛ طور مثال، بررسی کننده استثناء به خاطر مشخص کردن غلطی‌های تقسیم بر صفر، اندرس‌های نادرست رشته‌ها وغیره استفاده شود.

۹.۴ تکثیرشدن استثناء در بلاک‌های آشیانه‌یی (Exception Propagation)

اگر یک استثناء در بلاک PL/SQL طوری به وجود می‌آید که در آن بررسی کننده استثناء در نظر گرفته نشده باشد، در چنین حالتی استثناء تکثیر (Propagate) می‌شود؛ به این معنا که استثناء دوباره به راه افتیده و تا زمانی ادامه پیدا کند که بلاک کد، یک بررسی کننده برای آن در نظر بگیرد و یا در آن کدام محدوده بلاک وجود نداشته باشد. در حالت اول؛ یعنی زمانی که بررسی کننده استثناء موجود نباشد، یک غلطی استثنای بررسی نشده (Unhandled Exception) را نشان خواهد داد.

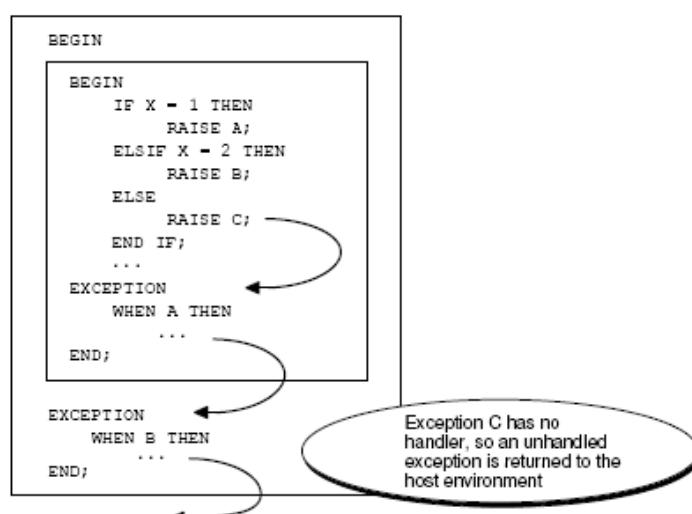
در شکل پایین، نشان داده شده است که یک بلاک در بین بلاک دیگر، آشیانه (Nested) شده است. بلاک داخلی یک استثنای به نام A را صادر می‌کند؛ همین بلاک داخلی یک بررسی‌کننده به خاطر استثنای A دارد، پس استثنای A تکثیر نمی‌شود. زمانی که بررسی‌کننده استثناء (Exception Handler) اجرا می‌شود، کنترول به بلاک بیرونی انتقال پیدا می‌کند.

به شکل پایین توجه شود:



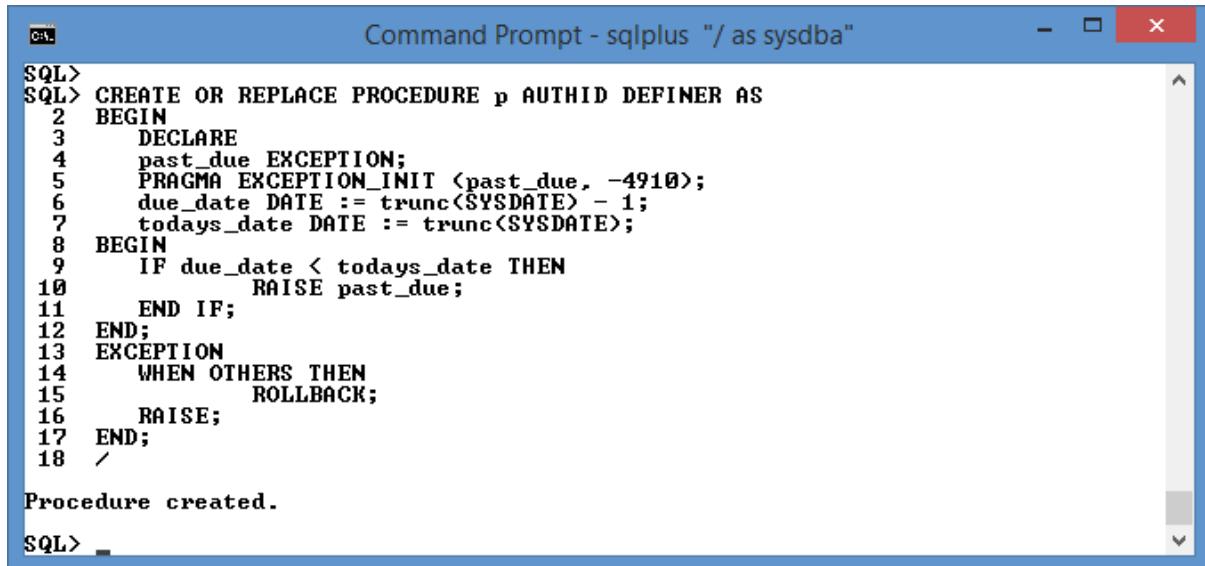
شکل ۸-۹: تکثیر نمی‌شود

در تصویر بعدی، در بلاک داخلی استثنای B واقع می‌شود. این بلاک، بررسی‌کننده استثناء برای استثنای B ندارد؛ پس استثنای B به بلاک بیرونی تکثیر می‌شود (سرایت می‌کند). در بلاک بیرونی یک بررسی‌کننده استثناء برای B موجود است. بعد از آن که بررسی‌کننده استثناء اجراء می‌شود، کنترول به محیط بیرون از PL/SQL می‌رود؛ به شکل پایین در ارتباط به موضوع مراجعه شود.



شکل ۹-۹ PL/SQL ۹-۹ غلطی بررسی استثناء را به محیط بیرونی انتقال می‌دهد.

یک استثناء که توسط استفاده‌کننده تعریف شده باشد، امکان دارد بیشتر از حد تعیین‌شده تکثیر شود (یعنی به بیرون از بلاکی که در آن تعریف شده باشد، سرایت کند)؛ اما نام آن شاید در خارج از بلاک کدی که در آن تعریف شده است، شناخته نشود؛ پس استثناهایی که توسط استفاده‌کننده‌گان تعریف شده باشند، در بلاک‌های بیرونی فقط با بررسی‌کننده استثناء به نام OTHERS شناخته و قابل استفاده هستند. در ارتباط به موضوع به مثال پایین توجه شود.



```

SQL> CREATE OR REPLACE PROCEDURE p AUTHID DEFINER AS
 2  BEGIN
 3    DECLARE
 4      past_due EXCEPTION;
 5      PRAGMA EXCEPTION_INIT (past_due, -4910);
 6      due_date DATE := trunc(SYSDATE) - 1;
 7      todays_date DATE := trunc(SYSDATE);
 8  BEGIN
 9    IF due_date < todays_date THEN
10      RAISE past_due;
11    END IF;
12  END;
13  EXCEPTION
14    WHEN OTHERS THEN
15      ROLLBACK;
16      RAISE;
17  END;
18 /
Procedure created.
SQL>

```

شكل ۱۰-۹

در شکل بالا دیده می‌شود که در بلاک داخلی، یک استثنای به نام past_due تعریف شده است که برای آن بررسی‌کننده‌ی موجود نیست. زمانی که بلاک داخلی استثناء past_due را در سطر 10 بلند (Raise) می‌کند، استثناء به بلاک بیرونی سرایت می‌کند. استثناء past_due در بلاک بیرونی موجود نیست، یعنی شناخته نمی‌شود؛ بناءً این استثناء در بلاک بیرونی به کمک بررسی‌کننده استثناء به نام OTHERS کنترول می‌شود.

در صورتی که بلاک بیرونی، این استثنای تعریف شده توسط استفاده‌کننده را بررسی نتواند، یک غلطی واقع خواهد شد؛ چنین حالتی در مثال پایین نشان داده شده است.

شکل ۱۱-۹

طوری که پیش‌بینی شده بود، در چنین حالتی غلطی واقع شد. در متن صفحه بهوضاحت دیده می‌شود که در سطر 8 کد بالا پیام (unhandled user-defined exception) نشان داده شده است.

۹.۵ پیام‌های استثناهای PL/SQL

در زبان PL/SQL در زمان استفاده کردن از استثناهای غلطی با نیز پیام غلطی‌ها قابل دید است. کد غلطی‌ها در زبان PL/SQL با استفاده از تابع SQLCODE نشان داده می‌شود.

پیام‌های غلطی‌ها در این زبان به طریقه‌های ذیل قابل دیدن هستند:

- استفاده از تابع SQLERRM: این تابع معلوماتی به طول مجموعی 512 بایت را ارائه می‌دارد. طول اعظمی یک پیام غلطی دیتابیس اوریکل نیز همین اندازه است. پیام غلطی می‌تواند معلوماتی در موارد غلطی کد، پیام‌های آشیانه‌یی و پیام‌های داخلی مانند نام جدول‌ها و فیلدها را در بر داشته باشد.
- استفاده از تابع پکیج به نام DBMS_ERROR_STACK: این تابع معلومات مکمل را در مورد غلطی ارائه می‌کند. طول معلومات ارائه شده توسط این تابع به 2000 بایت می‌رسد. در اوریکل در حالات عادی، استفاده از این تابع به خاطر نمایش دادن پیام‌های غلطی توصیه می‌شود. تنها در حالتی که در یک بلاک PL/SQL از دستور FORALL با اختیار SAVE EXCEPTIONS استفاده شده باشد، از به کارگیری تابع DBMS_ERROR_STACK جلوگیری می‌شود. جهت وضاحت موضوع به مثالی که قبلًا در عنوان «تطبیق FORALL بعد از تکمیل دستورهای آن» در همین فصل ارائه شد، مراجعه شود.

مسئله دیگر در قسمت استفاده از پیام‌های غلطی‌ها (استثناهای SQLERRM و SQLCODE) در مورد دستورهای سیکویل است. یک دستور سیکویل با استفاده از تابع‌های SQLERRM و SQLCODE به طور مستقیم قابل دید نیست. به خاطر

انجام این کار، در صورت ضرورت، دستور سیکویل می‌تواند به متحولینی تعریف شوند و بعد آن متحولین در تابع‌های دیدن غلطی کد استفاده شوند. در این مورد نیز مثالی کار شده است که در ذیل نشان داده می‌شود.

```

SQL> DROP TABLE errors;
Table dropped.

SQL> CREATE TABLE errors (
 2   code NUMBER,
 3   message VARCHAR2(64)
 4 );
Table created.

SQL> CREATE OR REPLACE PROCEDURE p_AUTHID DEFINER AS
 2   name EMPLOYEES.LAST_NAME%TYPE;
 3   v_code NUMBER;
 4   v_errm VARCHAR2(64);
 5 BEGIN
 6   SELECT last_name INTO name
 7   FROM EMPLOYEES
 8   WHERE EMPLOYEE_ID = -1;
 9 EXCEPTION
10   WHEN OTHERS THEN
11     v_code := SQLCODE;
12     v_errm := SUBSTR(SQLERRM, 1, 64);
13     DBMS_OUTPUT.PUT_LINE
14       ('Error code ' || v_code || ':' || v_errm);
15   /* Invoke another procedure,
16      declared with PRAGMA AUTONOMOUS_TRANSACTION,
17      to insert information about errors. */
18   INSERT INTO errors (code, message)
19   VALUES (v_code, v_errm);
20   RAISE;
21 END;
22 /

```

شکل ۱۲-۹

طوری که در شکل بالا دیده می‌شود، متحول‌های v_code و v_errm از نوع نمبر و متحول v_errm از نوع کرکتر در سطرهای 3 و 4 پروسیجر تعریف شده‌اند. در سطرهای 11 و 12 تابع‌های SQLCODE و SQLERRM به ترتیب به متحول‌های یادشده معادل ساخته شده‌اند و در نهایت در سطر 19 قیمت‌های متحول‌ها به حیث قیمت‌های عادی به جدول errors داخل شده و قابل استفاده می‌شوند.



خلاصه فصل نهم

در این فصل به موضوعات تخصصی مربوط به غلطی‌های پروگرامنویسی در زبان PL/SQL پرداخته شده است. طوری که از عنوان فصل معلوم می‌شود، دستورهای بررسی استثناهای (Exceptions) در زبان PL/SQL مورد بحث تخصصی قرار گرفته است. در جریان فصل مثال‌های لازم نیز ارائه شده‌اند. عنوانین فصل نهم به ترتیب از توضیح در مورد استثناهای در این زبان آغاز شده و با ارائه معلومات در مورد استثناهای بررسی نشده ادامه پیدا کرده است. دو طریقه تطبیق دستور FORALL در استثناهای بررسی نشده توضیح شده‌اند.

أنواع بررسی‌کننده‌های استثناهای در سیکویل از موضوعات مهم دیگر این فصل به شمار می‌رود که به آن پرداخته شده است. تکثیر کردن استثناهای در شرایط خاص بین بلاک‌های آشیانه‌یی و خطرات احتمالی از دیاد غلطی‌ها در عنوان مستقل با ذکر مثال‌هایی پی گرفته شده‌اند. در نهایت به موضوع پیام‌ها و تنظیم کردن پیام‌های استثناهای با یک مثال توضیحی فصل نهم خاتمه داده شده است.



سوالات و فعالیت های فصل نهم

۱. استثناهای (Exceptions) در زبان PL/SQL به چه مفهوم‌اند؟ توضیح دهید.
۲. شکل عمومی نوشتمن استثناهای را در یک بلاک زبان PL/SQL بنویسید.
۳. شرط FORALL در استثناهای بررسی‌نشده (Unhandled Exceptions) به کدام دو شکل استفاده می‌شود؟ توضیح دهید.
۴. انواع بررسی‌کننده‌های استثناهای را نام ببرید.
۵. در کدام حالت یک استثناء تکثیر (Propagate) می‌شود؟ واضح سازید.
۶. پیام‌های استثناهای در زبان PL/SQL با استفاده از کدام تابع‌ها نمایش داده می‌شوند؟ نام بگیرید.

فعالیت ها

۱. با مراجعه به فعالیت فصول قبلی، صفحه سیکویل پلس را با استفاده از دستور `sqlplus "/ as sysdba"` باز کنید.
۲. برنامه Notepad را به خاطر نوشتن و ایدیت کردن کد PL/SQL باز کنید.
۳. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از دستور FORALL در قسمت Unhadled Exception به شکل فوری (Immediate) استفاده شده باشد.
۴. یک پروگرام ساده PL/SQL را نوشته و اجراء کنید که در آن از دستور FORALL در قسمت Save Exceptions با تکمیل دستورهای آن با استفاده از اختیار Unhadled Exception تطبیق شده باشد.
۵. نتیجه دو فعالیت قبلی را مقایسه کرده و تفاوت را در قسمت به روز شدن محتوای جدول مورد پروسس ببینید.

فصل دهم

پروسیجرها (Functions)، تابع‌ها (Procedures)، تریگرها (Triggers) و مجموعه‌ها (Packages) در زبان (Collections) PL/SQL



هدف کلی: شاگردان با ایجاد کردن پروسیجرها، تابع‌ها، تریگرها، بسته‌ها و مجموعه‌ها در زبان PL/SQL آشنا شوند.

اهداف آموزشی: در پایان این فصل محصلان قادر خواهند شد تا:

۱. برنامه‌های فرعی (Subprograms) را شرح و ایجاد کنند.
۲. بلاک‌های ناشناس (Anonymous Blocks) را در PL/SQL توضیح دهند.
۳. ایجاد کردن یک پروسیجر ساده را با بلاک ناشناخته بیاموزند.
۴. یک تابع ساده (Simple Function) را ایجاد کنند.
۵. یک تابع ساده با پارامترها را ایجاد کنند.
۶. فرق بین پروسیجرها و تابع‌ها را شرح دهند.
۷. تریگرها (Triggers) را در زبان PL/SQL شرح دهند.
۸. بسته‌ها (Collections) را در زبان PL/SQL شرح دهند.
۹. مجموعه‌ها (Collections) را در زبان PL/SQL شرح دهند.

در زبان PL/SQL ساختارهایی موجوداند که به خاطر سهولت کار استفاده‌کنندگان و پروگرامرها ضروری پنداشته می‌شوند. در حالت عادی به خاطر کارکردن با دیتابیس اوریکل، از یک کوماند پرمپت مانند سیکویل پلس، به خاطر اجرای دستورها استفاده می‌شود. این دستورها به صورت مستقیم تایپ شده و یا از مرجع دیگری کاپی و در این صفحه Paste می‌شوند که بعد از آن، دستورهای یادشده با فشاردادن دکمه Enter اجرا می‌شوند؛ اما به خاطر تنظیم و مدیریت بهتر نوشتن کد زبان PL/SQL، به خصوص زمانی که پروگرام‌های طولانی (چند صفحه‌یی) نوشته می‌شوند، وسیله‌هایی در این زبان موجوداند تا پروسه‌های یادشده را به خاطر مدیریت بهتر کمک کنند. نمونه‌های این وسیله‌ها عبارت از پروسیجرها، تابع‌ها، تریگرها، بسته‌ها و مجموعه‌ها در زبان PL/SQL اند.

در این فصل به توضیح، استفاده و به کارگیری وسیله‌ها و طریقه‌های کمکی در قسمت کارکردن با زبان PL/SQL پرداخته شده است. برنامه‌های فرعی که در انگلیسی به آن‌ها Subprograms می‌گویند، همراه با خصوصیات و بخش‌های فرعی آن‌ها تحت عنوانین مختلف توضیح داده شده‌اند. بلاک‌های ناشناس PL/SQL (Anonymous Blocks) و اهمیت استفاده از آن‌ها بعد از موضوعات مربوط به برنامه‌های فرعی شرح شده و ساختن یک پروسیجر ساده با آن توضیح و در مثال نشان داده شده است. ساختن تابع‌های ساده در زبان PL/SQL و استفاده کردن از آن‌ها و نیز فرق بین پروسیجر و تابع در زبان PL/SQL واضح ساخته شده است. در قسمت‌های آخر فصل به توضیحات مختصر و مثال‌هایی در مورد تریگرها، بسته‌ها و مجموعه‌ها پرداخته شده است.

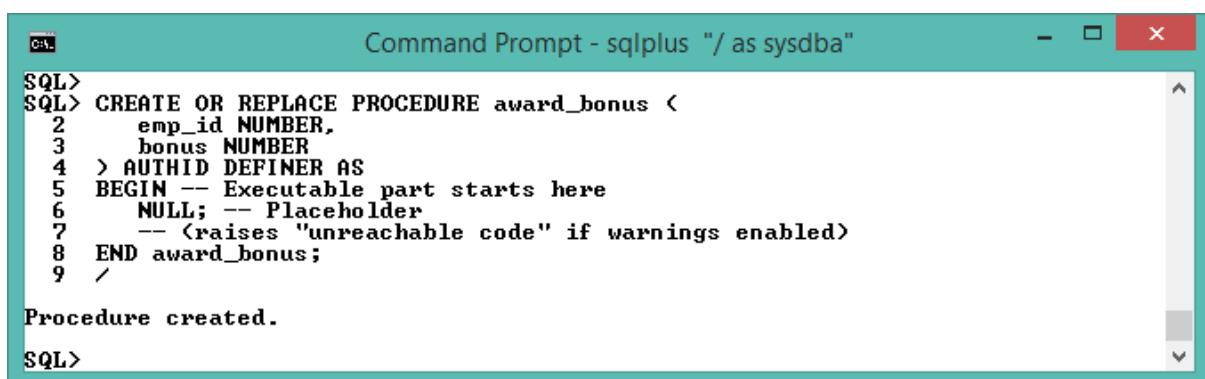
۱۰.۱ برنامه‌های فرعی (Subprograms) در PL/SQL

یک برنامه فرعی در زبان PL/SQL عبارت از یک بلاک این زبان است. در پروگرام فرعی قابلیت اجرای تکراری دستورهای شامل آن مساعد بوده و برنامه فرعی PL/SQL به یک نام مسمی شده و در سیستم ذخیره می‌شود. یک برنامه فرعی می‌تواند از طریق پارامترها دیتا بگیرد، در صورت گرفتن عناصر دیتا از پارامترها، با اجرای هر بار پروگرام، نتیجه متفاوت نشان داده می‌شود.

یک پروگرام فرعی می‌تواند یک پروسیجر و یا یک تابع باشد. یک پروسیجر در زبان PL/SQL معمولاً به خاطر انجام‌دادن یک کار استفاده می‌شود و یک تابع در این زبان به هدف محاسبه و به‌دست‌آوردن یک قیمت به کار برده می‌شود. جزئیات بیشتر در مورد پروسیجرها و تابع‌ها در عنوانین مستقل این فصل ارائه شده‌اند.

پروگرام‌های فرعی به خاطر انکشاف و نگهداشت بهتر کد برنامه‌ها استفاده می‌شوند. این ساختمان‌ها می‌توانند جهت بلندبردن اطمینان از مؤثثیت پروگرام و دوباره استفاده کردن قسمت‌های کد استفاده شوند. به طور کلی، مؤثثیت استفاده از پروگرام‌های فرعی در موارد پایین بیشتر هویداست:

- پیمانه‌بی‌بودن (Modularity): با استفاده از پروگرام‌های فرعی، یک پروگرام بزرگ به قسمت‌های کوچک‌تر تقسیم شده می‌تواند؛ طوری که مدیریت، فهمیدن و استفاده کردن قطعات کوچک‌تر پروگرام‌ها نظر به کدهای طولانی پروگرام‌ها به مرتب سهل‌تر است.
- دیزاین آسان‌تر اپلیکیشن‌ها (Easier Application Design): زمانی که یک اپلیکیشن دیزاین می‌شود، با استفاده از پروگرام‌های فرعی، توضیحات تفصیلی در هر قسمت داده شده و نیز امتحان کردن پروگرام‌های فرعی به صورت واحدهای مستقل پروسه‌های دیزاین کلی اپلیکیشن را آسان‌تر می‌سازد؛ طور مثال در شکل پایین یک پروگرام فرعی (پروسیجری به نام award_bonus) موفقانه ایجاد شده است که به صورت مستقل قابل اجراء و استفاده است.



```

Command Prompt - sqlplus "/ as sysdba"
SQL> CREATE OR REPLACE PROCEDURE award_bonus (
  2   emp_id NUMBER,
  3   bonus NUMBER
  4 ) AUTHID DEFINER AS
  5 BEGIN -- Executable part starts here
  6   NULL; -- Placeholder
  7   -- (raises "unreachable code" if warnings enabled)
  8 END award_bonus;
  9 /
Procedure created.
SQL>

```

شکل ۱-۱۰

- قابلیت نگهداری (Maintainability): یک پروگرام فرعی می‌تواند به صورت مستقل بدون این که به بخش‌های دیگر بستگی داشته باشد، تغییر، اصلاح و انکشاپ داده شود؛ البته تغییرات و انکشاپات باید در هماهنگی با بخش‌های دیگر پروگرام به صورت کلی صورت گیرد.
- قابلیت بسته‌بندی کردن (Packageability): پروگرام‌های فرعی زبان PL/SQL در بسته‌ها قابل دسته‌بندی هستند؛ یعنی هر پروگرام فرعی به صورت یک بسته (Package) بوده و استفاده، انتقال و مدیریت آن به وجه بهتر صورت می‌گیرد.
- قابلیت دوباره استفاده کردن (Reusability): یک پروگرام فرعی (قطعه کد مستقل پروگرام) در پروگرام‌های مختلف و محیط‌های مختلف قابلیت استفاده دوباره را دارد؛ چون هر پروگرام فرعی یک سمت مستقل دستورها را شامل می‌شود که به صورت مستقل قابل اجراء است.
- اجرای بهتر (Better Performance): هر پروگرام فرعی طوری دیزاین می‌شود که شکل اجرایی داشته باشد؛ یعنی دستورهای آغازین، میانی و آخری به خاطر اجرای مستقل عبارت از اجزای تشکیل‌دهنده یک پروگرام فرعی می‌باشند. پروگرام‌های فرعی معمولاً در کمپیوتر سرور دیتابیس‌ها تعبیه می‌شوند و با واردشدن دستورهای اجرایی از طرف استفاده کنندگان، به راه اندخته می‌شوند.

این کار باعث می‌شود تا مقدار کمتر دیتا (تنها دستور به اجراء در آوردن پروگرام فرعی) در شبکه‌ها تبادله شود. مسئله مهم دیگر در قسمت اجرای بهتر پروگرام‌های فرعی، شریک‌ساختن این پروگرام‌ها در بین استفاده‌کنندگان مختلف است. این کار باعث استفاده حداقلی حافظه در یک سیستم شده و اجراء بهتر می‌شود.

۱۰.۱ خصوصیات پروگرام‌های فرعی

به پروگرام‌های فرعی در زبان PL/SQL خصوصیت‌هایی به خاطر مشخص کردن طریقه کارشان قابل تعریف است. این خصوصیات معمولاً در قسمت اعلامیه (Declaration) پروگرام فرعی تعیین می‌شوند. نوشتن خصوصیات به یک پروگرام فرعی به شکل‌های مختلف مجاز است. خصوصیت‌های پروگرام‌های فرعی پیش‌تر از کلمه‌های کلیدی IS و یا AS در قسمت بالایی (Heading) نوشته می‌شوند. در پروگرام‌های فرعی آشیانه‌یی (Nested Subprograms)، استفاده از خصوصیات مجاز نیست. در پروگرام‌های فرعی که در بسته‌بندی‌ها استفاده می‌شوند، تنها به کارگیری خصوصیت ACCESSIBLE BY اجازه بوده و استفاده از متباقی خصوصیات در آن‌ها نیز مجاز نیست.

پروگرام‌های فرعی مستقل (نه آشیانه‌یی و نه هم بسته‌بندی شده) می‌توانند خصوصیات پایین را در قسمت اعلامیه خود داشته باشند:

- ACCESSIBLE BY Clause
- DEFAULT COLLATION Clause
- Invoker's Rights and Definer's Rights (AUTHID Property)

مثال‌های استفاده از نوع سوم خصوصیات پروگرام‌های فرعی (AUTHID) در دروس فصل‌های قبلی نشان داده شده‌اند. یک مثال از فصل قبلی در ذیل ارائه شده است. در این مثال از خصوصیت AUTHID که بعد از نام پروسیجر آمده، استفاده شده است.

```

SQL> CREATE OR REPLACE PROCEDURE p_AUTHID DEFINER AS
 2 BEGIN
 3   DECLARE
 4     past_due EXCEPTION;
 5     PRAGMA EXCEPTION_INIT(past_due, -4910);
 6     due_date DATE := trunc(SYSDATE) - 1;
 7     todays_date DATE := trunc(SYSDATE);
 8 BEGIN
 9   IF due_date < todays_date THEN
10     RAISE past_due;
11   END IF;
12 END;
13 EXCEPTION
14   WHEN OTHERS THEN
15     ROLLBACK;
16     RAISE;
17 END;
18 /

```

Procedure created.

SQL> _

شکل ۲-۱۰

در شکل بالا دیده می‌شود که در بلاک داخلی یک استثناء به نام `past_due` تعریف شده است که برای آن بررسی‌کننده‌یی موجود نیست. زمانی که بلاک داخلی استثناء `past_due` را در سطر 10 بلند (Raise) می‌کند، استثناء به بلاک بیرونی سرایت می‌کند. استثناء `past_due` در بلاک بیرونی موجود نیست، یعنی شناخته نمی‌شود؛ بناءً این استثناء در بلاک بیرونی به کمک بررسی‌کننده استثناء به نام OTHERS کنترول می‌شود.

۱۰.۱.۲ بخش‌های یک پروگرام فرعی

یک پروگرام فرعی در زبان PL/SQL با عنوان پروگرام فرعی (Subprogram Heading) آغاز می‌شود. عنوان در حقیقت نام پروگرام فرعی بوده که به آن مسمی می‌شود. یک پروگرام فرعی می‌تواند به صورت اختیاری یک لیست از پارامترها را نیز داشته باشد. پروگرام فرعی مانند بلاک PL/SQL دارای سه قسمت است.

- بخش اعلامیه (Declarative Part): این قسمت پروگرام اختیاری است؛ یعنی در صورت ضرورت، استفاده شده و در غیر آن استفاده نمی‌شود. در این بخش Type‌های محلی، کرسوها، ثابت‌ها، متحولین، استثناهای و پروگرام‌های فرعی آشیانه‌یی تعریف و اعلام می‌شوند. تمام عناصر تعریف شده در این بخش، یک پروگرام فرعی محدود به همان پروگرام می‌باشند. با تکمیل شدن اجرای پروگرام فرعی، عناصر تعریف شده در قسمت اعلامیه پروگرام دوباره از حافظه پاک می‌شوند.
- بخش قابل اجرا (Executable Part): این بخش بروگرام حتمی است. در صورت عدم موجودیت قسمت قابل اجرای پروگرام، در حقیقت یک پروگرام فرعی ناقص شمرده شده و برای هدفی که ساخته شده است، قابل استفاده نیست. در بخش اجرایی موجودیت حد اقل یک دستور اجرایی که در آن قیمت‌ها استفاده شود (اجرا کنترول شده و یا با دیتا کار شود) شرط حتمی است. در بعضی

مثال‌ها در فصول قبلی به صورت نمونه یک دستور آن هم NULL استفاده شده است. چنان مثال‌ها، بدون نتیجه (Null) بوده، ولی از نظر منطقی درست‌اند؛ چون حداقل یک دستور اجرایی در آن‌ها استفاده شده‌اند. مثال چنان پروگرام در شکل ذیل نشان داده شده است:

```

SQL> CREATE OR REPLACE PROCEDURE award_bonus (
  2   emp_id NUMBER,
  3   bonus NUMBER
  4 ) AUTHID DEFINER AS
  5 BEGIN -- Executable part starts here
  6   NULL; -- Placeholder
  7   -- <raises "unreachable code" if warnings enabled>
  8 END award_bonus;
  9 /
Procedure created.

SQL>

```

شکل ۳-۱۰

سطرهای ۵ الی ۸ قسمت اجرایی پروسیجر (سرحدات ابتدایی و انتهایی) را در بر دارد. سطر ۸ دستور اجرایی (دستور NULL) را نشان می‌دهد که همین دستور به پروگرام صبغه قانونی داده است.

- بخش بررسی استثناهای (Exception Handling Part): این بخش نیز به شکل اختیاری است که در صورت عدم استفاده از آن، یک پروگرام فرعی قابل اجراء ساخته می‌شود. بخش بررسی استثناهای غلطی‌هایی را بررسی می‌کند که در زمان اجرای پروگرام به میان می‌آیند. توضیحات مکمل راجع به این قسمت در فصل قبلی ارائه شده‌اند. در اینجا صرف یک مثال در نظر گرفته شده است.

```

SQL> DECLARE
  2   c_id customers.id%type := 8;
  3   c_name customers.Name%type;
  4   c_addr customers.address%type;
  5 BEGIN
  6   SELECT name, address INTO c_name, c_addr
  7   FROM customers
  8   WHERE id = c_id;
  9   DBMS_OUTPUT.PUT_LINE ('Name: ' || c_name);
 10   DBMS_OUTPUT.PUT_LINE ('Address: ' || c_addr);
 11
 12 EXCEPTION
 13   WHEN no_data_found THEN
 14     dbms_output.put_line('No such customer!');
 15   WHEN others THEN
 16     dbms_output.put_line('Error!');
 17 END;
 18 /

```

شکل ۴-۱۰

در صورت موجودبودن جدول customers با ساختار و دیتای مناسب، دستور استثنای فوق در قسمت سوم بلاک PL/SQL بین سطرهای 12 الی 16 تطبیق خواهد شد. در صورتی که مشتری با شماره شناسایی 8 در جدول موجود نباشد، قسمت استثناء اجراء شده و پیام ذیل را چاپ خواهد کرد:

No such customer!

PL/SQL procedure successfully completed.

۱۰.۲ بلاک‌های ناشناس (Anonymous Blocks) در PL/SQL

کد زبان PL/SQL با در نظرداشت قواعد و استندردهای این زبان نوشته شده و اجراء می‌شود. قطعات کد می‌توانند مستقیم در صفحه کد کردن این زبان مانند سیکویل پلس تایپ شده و به اجراء گذاشته شوند و یا با استفاده از یک وسیله دیگر مانند ایدیتور کد و یا ایدیتور متن، کد نوشته شده و بعد از کاپی کردن در صفحه کد کردن اوریکل Paste و استفاده شود و یا هم با ذخیره کردن کد در یک فایل متنی با اکستنشن (.sql)، یک فایل اجرایی ایجاد و به اجراء گذاشته شود.

در هر سه شکل، نتیجه مورد نظر بعد از اجرای کد PL/SQL به دست می‌آید. کد اجراشده به این شکل‌ها صرف برای یک بار به اجراء گذاشته می‌شود و به نام بلاک‌های ناشناس (Anonymous Blocks) یاد می‌شوند؛ یعنی بلاک کد در دیتابیس ذخیره نمی‌شود. بلاک‌های ناشناس معمولاً بدون نام می‌باشند، حتاً اگر نام هم برای شان داده شود، باز هم این کد مشخصات بلاک‌های شناخته شده را که در دیتابیس‌ها ذخیره و در PL/SQL اجراء می‌شوند، ندارند.

یک بلاک ناشناس در هر بار اجرای آن توسط سیستم، ترجمه (Compile) می‌شود. ترجمه کد ناشناس در زمان اجراء دارای سه مرحله است:

- چک کردن نحوی (Syntax Checking): در مرحله اول کد نوشته شده توسط PL/SQL دیده شده و تجزیه می‌شود.
- چک کردن معنایی (Semantic Checking): دیدن نوشه‌ها از نظر معنایی توسط سیستم در این مرحله صورت می‌گیرد.
- تولید کد (Code Generation): بعد از طی شدن مراحل اول و دوم، کد قابل اجراء توسط PL/SQL تولید شده و به بخش اجرایی فرستاده می‌شود.

با درنظرداشت جزئیات بالا، بلاک‌های ناشناخته در PL/SQL با بلاک‌های شناخته شده از قبیل پروگرام‌های فرعی (Subprograms)، پروسیجرها (Procedures) و تابع‌ها (Functions) تفاوت‌های بارزی دارد. برخلاف بلاک‌های ناشناس، بلاک‌های شناخته شده در سرورهای دیتابیس ذخیره می‌شوند و با یک دستور به حافظه لود شده و به اجراء گذاشته می‌شوند. در حالی که بلاک‌های ناشناس در هر بار مطابق توضیحات فوق،

توسط PL/SQL تفسیر و ترجمه شده و بعد از آن، کد اجرایی شان تولید و به اجراء گذاشته می‌شود. این مسائل تأثیر مستقیم در قسمت استفاده از منابع اضافی مانند شبکه، پروسیسر و حافظه کمپیوترها دارد.

۱۰.۲.۱ ایجاد یک پروسیجر ساده با بلاک‌های ناشناس

یک پروسیجر (Procedure) در زبان PL/SQL عبارت از یک پروگرام فرعی بوده که به خاطر اجراء کردن یک کار مشخص استفاده می‌شود. یک پروسیجر به واسطه دستور سیکویل به نام (Call) به کار گرفته می‌شود. قبل از استفاده کردن از یک پروسیجر در PL/SQL در قدم نخست باید پروسیجر اعلام (Declare) و تعریف شود.

پروسیجرها به کمک بلاک‌های ناشناس بخش‌های اجرایی کد PL/SQL ایجاد و به اجراء گذاشته می‌شوند. در ذیل یک مثال نشان داده شده است که در آن یک بلاک ناشناس به صورت همزمان یک پروسیجر را اعلام و تعریف کرده و آن را سه مرتبه می‌خواند. در بار سوم یک استثناء بلند (Raise) می‌شود که توسط قسمت بررسی استثناء مورد بررسی قرار می‌گیرد و در نتیجه نشان داده می‌شود.

به مثال توجه شود:

```
SQL> DECLARE
 2   first_name employees.first_name%TYPE;
 3   last_name employees.last_name%TYPE;
 4   email employees.email%TYPE;
 5   employer VARCHAR2(8) := 'AcmeCorp';
 6   -- Declare and define procedure
 7   PROCEDURE create_email < -- Subprogram heading begins
 8     name1 VARCHAR2,
 9     name2 VARCHAR2,
10     company VARCHAR2
11   >           -- Subprogram heading ends
12   IS
13     -- Declarative part begins
14     error_message VARCHAR2(30) := 'Email address is too long.';
15   BEGIN           -- Executable part begins
16     email := name1 || '.' || name2 || '@' || company;
17   EXCEPTION -- Exception-handling part begins
18     WHEN VALUE_ERROR THEN
19       DBMS_OUTPUT.PUT_LINE(error_message);
20   END create_email;
21
22   BEGIN
23     first_name := 'John';
24     last_name := 'Doe';
25     create_email(first_name, last_name, employer); -- invocation
26     DBMS_OUTPUT.PUT_LINE ('With first name first, email is: ' || emai
i1);
27     create_email(last_name, first_name, employer); -- invocation
28     DBMS_OUTPUT.PUT_LINE ('With last name first, email is: ' || emai
i1);
29     first_name := 'Elizabeth';
30     last_name := 'MacDonald';
31     create_email(first_name, last_name, employer); -- invocation
32   END;
33 /
```

شکل ۵-۱۰

نتیجهٔ مثال بالا بعد از سه بار به اجراء گذاشتن پروسیجر در سطرهای 24، 26 و 30 که توسط یک بلاک ناشناس ایجاد شده است، طور ذیل است:

With first name first, email is: John.Doe@AcmeCorp

With last name first, email is: Doe.John@AcmeCorp

Email address is too long.

۱۰.۲.۲ تابع ساده (Simple Function)

تابع‌ها در زبان PL/SQL مانند پروسیجرها ایجاد و استفاده می‌شوند. در پروسیجرها عملیه‌های پروگرام شده اجراه می‌شوند. در تابع‌ها عملیه‌های پروگرام شده، اجراء شده و یک نتیجه (قیمت) را نشان می‌دهند؛ یعنی نشان‌دادن نتیجه با استفاده از دستور RETURN از ویژگی‌های توابع در زبان PL/SQL به شمار می‌رود.

یک تابع ساده با استفاده از دستور CREATE FUNCTION ساخته می‌شود. شکل عمومی دستور ایجاد کردن یک تابع ساده در زبان PL/SQL طور ذیل است:

```
CREATE [ OR REPLACE ] FUNCTION function_name  
    RETURN return_datatype  
    { IS | AS }  
    BEGIN  
        <function_body>  
    END [function_name];
```

در کد بالا، قسمت‌های کلیدی به اختصار طور ذیل شرح می‌شوند:

- function_name: نام تابع به خاطر شناسایی و استفاده تابع‌ها به کار برده می‌شود. طوری که در کد دیده می‌شود، نام تابع در سطرهای اول و آخر ذکر شده است. در سطر اول نام تابع حتمی بوده و در قسمت آخر دستور به شکل اختیاری استفاده می‌شود.
- [OR REPLACE]: این اختیار به این دلیل استفاده می‌شود که اگر از قبل به همین نام کدام تابع موجود باشد، تغییرات درخواست شده به آن تطبیق شود. این قسمت دستور اختیاری است؛ استفاده از آن دارای مزیت‌هایی است و در مواردی می‌تواند باعث ایجاد مشکلات نیز شود. مزیت‌های آن جلوگیری از پیام غلطی توسط سیستم بوده و همچنان تطبیق تابع را (در صورتی که کدام اشتباه دیگر نباشد) حتمی می‌سازد. در قسمت ایجاد مشکلات گفته می‌شود که اگر تابع دیگری به همین نام موجود باشد و آن تابع در قسمت‌های دیگر پروگرام‌ها، پروسیجرها و یا توابع دیگر استفاده شده باشد، با دستور REPLACE تابع جدید، جای‌گزین آن شده و باعث مشکلاتی خواهد شد.
- RETURN: این دستور از جمله قسمت‌های اساسی در یک تابع به شمار می‌رود. داشتن یک نتیجه با محتوای قیمت، یک تابع را از پروسیجرها متمایز می‌سازد.

- در این قسمت که به تعقیب دستور RETURN باید نوشته شود، نوع دیتای خروجی تابع مشخص می‌شود. دیتای خروجی تابع عبارت از دیتایی است که بعد از اجرای تابع نمایش داده شده و قابل استفاده می‌شود.

- {IS | AS} : اگر هدف این باشد که تابع مستقل (Standalone) ساخته شود، از کلمه AS در قسمت قبل از دستور BEGIN استفاده می‌شود و در غیر آن کلمه IS استفاده می‌شود.

- function_body : دستورهای اجرایی محتوای اصلی تابع در این قسمت نوشته می‌شوند. ترتیب منطقی دستورها به اساس قرارگرفتن آن‌ها در کد تابع ضروری بوده و اجرای دستورها به ترتیب از سطر اول شروع می‌شود.

در تعریف تابع‌های PL/SQL اختیارهای جدول ذیل قابل استفاده است:

Option	Description
DETERMINISTIC option	Helps the optimizer avoid redundant function invocations.
PARALLEL_ENABLE option	Enables the function for parallel execution, making it safe for use in slave sessions of parallel DML evaluations.
PIPELINED option	Makes a table function pipelined, for use as a row source.
RESULT_CACHE option	Stores function results in the PL/SQL function result cache.

مثال پایین، ساختن یک تابع ساده را نشان می‌دهد که در آن از کلمه IS استفاده شده و به اساس یک جدول دیتابیس اجرآت می‌کند. خروجی این تابع، مجموع تعداد مشتری‌ها در جدول customers خواهد بود؛ به شکل پایین توجه شود.

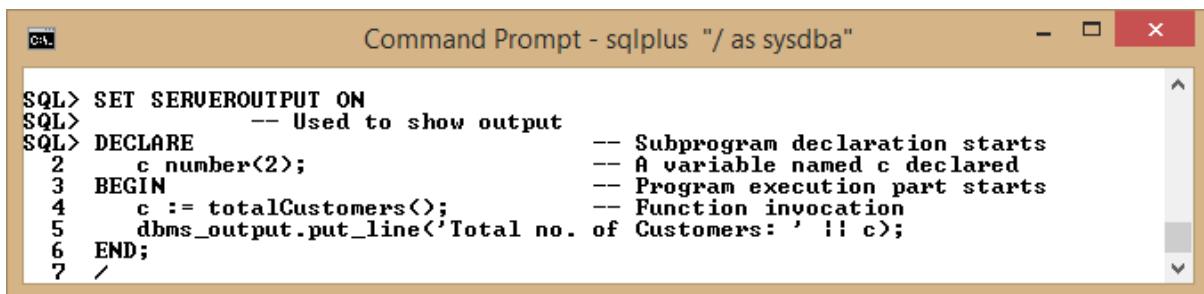
```

SQL> -- Initializing function creation
SQL> CREATE OR REPLACE FUNCTION totalCustomers
  2   RETURN number;          -- RETURN clause
  3   IS;                   -- Related function
  4   total number(2) := 0;  -- Declarative part begins
  5   BEGIN
  6     SELECT count(*) into total;
  7     FROM customers;      -- customers Table must exist
  8     RETURN total;        -- RETURN clause
  9   END;
 10 /
Warning: Function created with compilation errors.
SQL> _
```

شکل ۶-۱۰

طوری که در شکل بالا دیده می‌شود، دستور ایجاد کردن تابع به نام totalCustomers با یک اخطاریه (Warning) به صورت موفقانه اجرا شده و از ایجاد شدن تابع خبر می‌دهد. این اخطاریه به خاطر عدم موجودیت جدولی به نام customers که در سطر 7 خواسته شده است، می‌باشد.

در شکل بعدی همین تابع استفاده (Call) شده است.



```
SQL> SET SERVEROUTPUT ON
SQL>          -- Used to show output
SQL> DECLARE          -- Subprogram declaration starts
2      c number<2>;    -- A variable named c declared
3  BEGIN            -- Program execution part starts
4      c := totalCustomers<>;    -- Function invocation
5      dbms_output.put_line('Total no. of Customers: ' || c);
6  END;
7 /
```

شکل ۷-۱۰

نتیجه اجرا کردن تابع به نام totalCustomers در پروگرام فرعی بالا در صورت موجودیت جدول customers مجموع تعداد کارمندان را به اساس تعداد ریکورد های جدول طور ذیل نشان خواهد داد:

Total no. of Customer: 6

PL/SQL procedure successfully completed.

۱۰.۲.۳ دستور RETURN در PL/SQL

دستور RETURN زمانی که در بخش های کد زبان PL/SQL استفاده شود، اجرای کد به صورت فوري متوقف می شود؛ طور مثال، این دستور وقتی در یک پروگرام فرعی و یا در بلاک ناشناخته استفاده شده باشد، دستورهای قبل از RETURN اجرا شده و دستورهای بعد از آن، اگر موجود باشند، توقف داده می شوند. در یک پروگرام فرعی و یا بلاک ناشناخته زبان PL/SQL چندین بار از دستور RETURN در عین پروگرام می توان استفاده کرد.

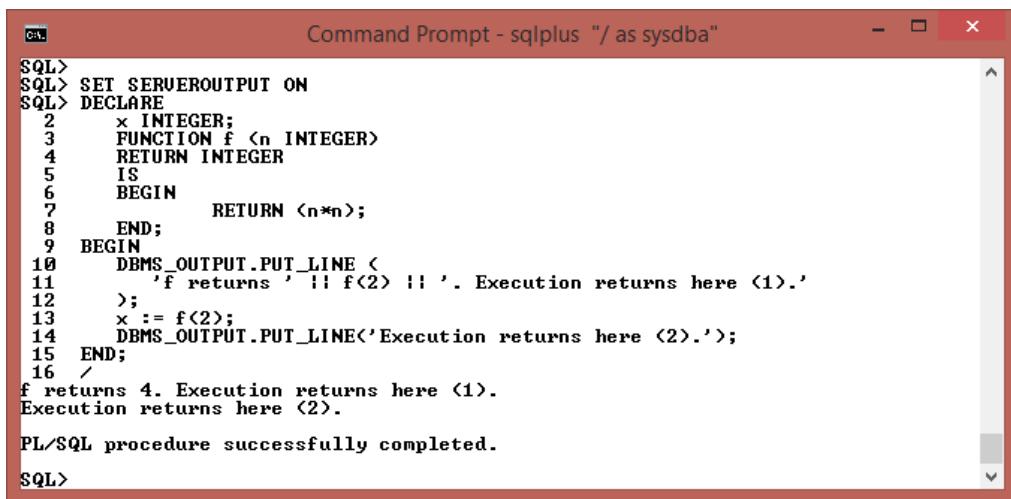
دستور RETURN کارایی زیاد داشته و در زبان PL/SQL در قسمت های ذیل استفاده می شود:

- در تابع ها (Functions)
- در پروسیجرها (Procedures)
- در بلاک های ناشناس (Anonymous Blocks)

۱۰.۲.۴ استفاده از دستور RETURN در تابع

در یک تابع، هر دستور اجرایی باید به نحوی به دستور RETURN خاتمه پیدا کند و هر دستور RETURN تابع ها باید یک خروجی داشته باشد. این دستور قیمت حاصله را به بخش کنترولی تابع سپرده و آن قیمت به شکل خروجی از اجرای تابع استخراج می شود. به خاطر وضاحت موضوع، مثال هایی در نظر گرفته شده اند که در ذیل ارائه می شوند.

به شکل پایین توجه شود:



```
SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      x INTEGER;
 3      FUNCTION f <n INTEGER>
 4          RETURN INTEGER
 5      IS
 6      BEGIN
 7          RETURN <n*n>;
 8      END;
 9      BEGIN
10          DBMS_OUTPUT.PUT_LINE <
11          'f returns ' || f<2> || '. Execution returns here <1>.' 
12      >;
13      x := f<2>;
14      DBMS_OUTPUT.PUT_LINE('Execution returns here <2>.');
15  END;
16 /
f returns 4. Execution returns here <1>.
Execution returns here <2>.

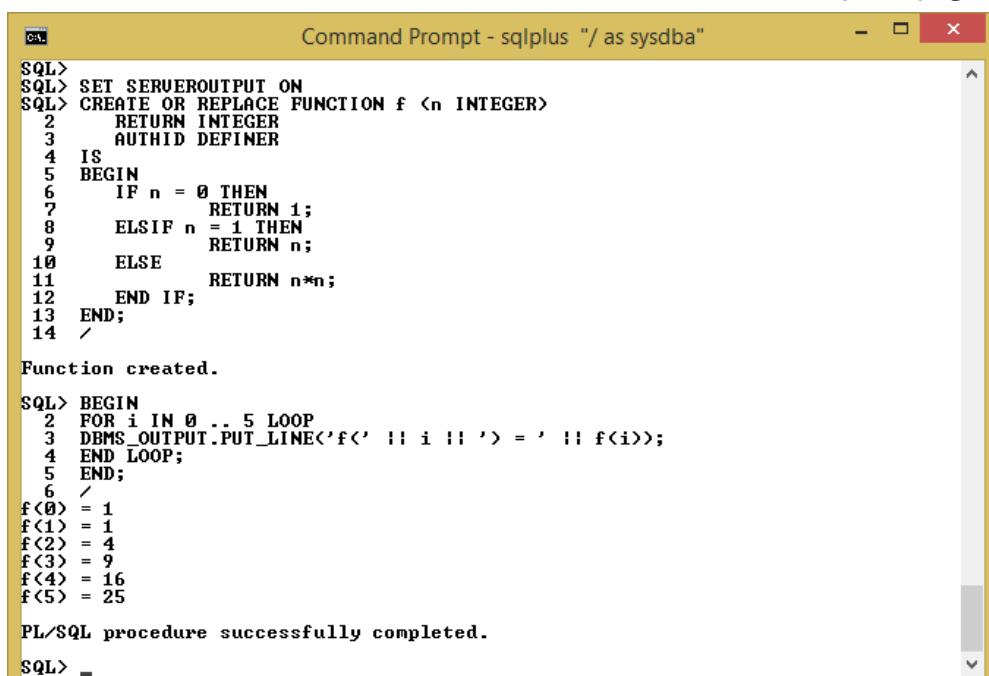
PL/SQL procedure successfully completed.

SQL>
```

شکل ۸-۱۰

طوری که در شکل دیده می‌شود، یک تابع به نام f ایجاد شده است و در آن دستور RETURN دو بار در سطرهای 4 و 7 استفاده شده است. این دستور در سطر 4 به شکل داخلی کار کرده و کنترول را دوباره به داخل تابع حفظ می‌کند. استفاده دوم آن در سطر 7 بعد از اجراء، کنترول را به دستورهای بعد از تابع در سطر 9 انتقال می‌دهد. دستورهای ایجاد کردن تابع و استفاده کردن از آن در بخش اجرایی پروگرام (سطرهای 11 و 13) موفقانه انجام شده و نتایج حاصله از آن‌ها در تصویر به مشاهده می‌رسد.

مثال بعدی نیز در ارتباط به استفاده از دستور RETURN در تابع‌ها در نظر گرفته شده است؛ در زمینه به شکل پایین توجه شود.



```
SQL> SET SERVEROUTPUT ON
SQL> CREATE OR REPLACE FUNCTION f <n INTEGER>
 2      RETURN INTEGER
 3      AUTHID DEFINER
 4  IS
 5  BEGIN
 6      IF n = 0 THEN
 7          RETURN 1;
 8      ELSIF n = 1 THEN
 9          RETURN n;
10      ELSE
11          RETURN n*n;
12      END IF;
13  END;
14 /

Function created.

SQL> BEGIN
 2  FOR i IN 0 .. 5 LOOP
 3      DBMS_OUTPUT.PUT_LINE(<'f<' || i || '> = ' || f<i>>);
 4  END LOOP;
 5  END;
 6 /
f<0> = 1
f<1> = 1
f<2> = 4
f<3> = 9
f<4> = 16
f<5> = 25

PL/SQL procedure successfully completed.

SQL> _
```

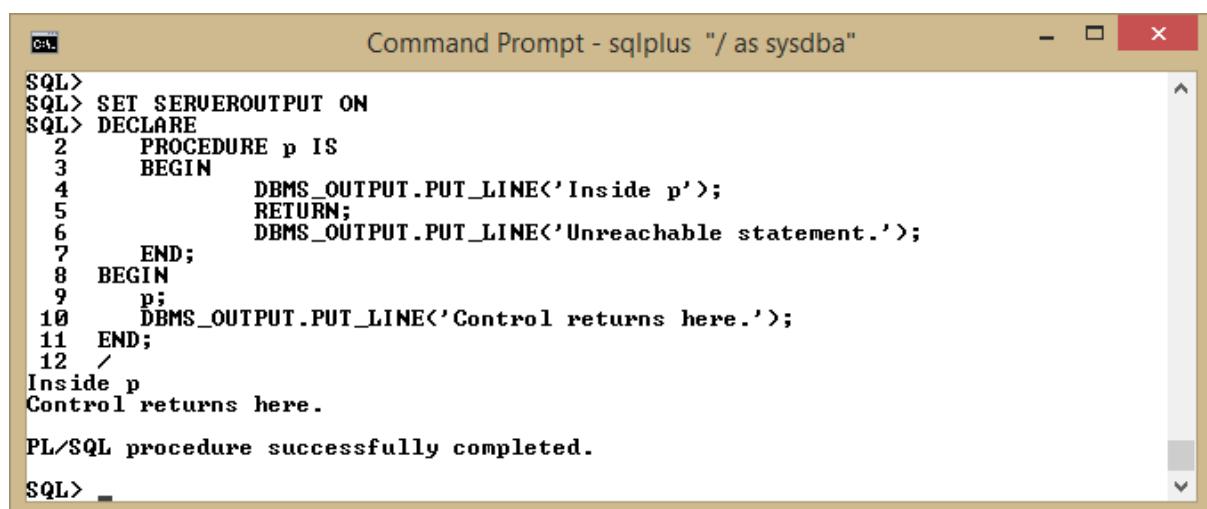
شکل ۹-۱۰

طوری که در شکل بالا دیده می‌شود، باز هم یک تابع متفاوت از مثال قبلی به نام f ساخته شده است که در آن از دستور RETURN چندین بار استفاده شده است. بار اول در سطر 2 مانند مثال قبلی، به شکل داخلی کار کرده و کنترول را در داخل تابع حفظ می‌کند. استفاده‌های بعدی این دستور در قسمت اجرایی تابع (سطرهای 7، 9 و 11) است که در دستور IF ELSIF به کار رفته است.

بعد از ایجاد موقانه تابع f، در قسمت دوم مثال، یک پروگرام نوشته شده است که در آن از تابع یادشده در سطر 3 استفاده شده است. در قسمت نتیجه که به اساس حلقه FOR با قیمت‌های 0 الی 5 تنظیم شده است، شش خروجی در قسمت نتیجه نشان داده شده است.

۱۰.۲.۵ استفاده از دستور RETURN در پروسیجر

برخلاف استفاده حتمی از عبارت RETURN در تابع‌ها، در یک پروسیجر استفاده از دستور RETURN اختیاری است. در صورت استفاده از عبارت RETURN در یک پروسیجر، این دستور کنترول را به محیط بیرونی که پروسیجر از آن جا به راه انداخته شده است، می‌برد. دستور RETURN در پروسیجر یک عبارت (Expression) را مشخص کرده نمی‌تواند. یک مثال در مورد موضوع یادشده مطابق شکل پایین در نظر گرفته شده است.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The SQL*Plus session displays the following code and its execution results:

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   PROCEDURE p IS
 3     BEGIN
 4       DBMS_OUTPUT.PUT_LINE('Inside p');
 5       RETURN;
 6       DBMS_OUTPUT.PUT_LINE('Unreachable statement.');
 7     END;
 8   BEGIN
 9     p;
10   DBMS_OUTPUT.PUT_LINE('Control returns here.');
11 END;
12 /
Inside p
Control returns here.

PL/SQL procedure successfully completed.

SQL> -

```

شکل ۱۰-۱۰

طوری که در شکل دیده می‌شود، یک پروسیجر به نام p در سطرهای 2 الی 7 تعریف شده است. در سطر 5 عبارت RETURN به شکل یک دستور استفاده شده است. با به راه انداختن پروسیجر p، که در سطر 9 شکل بالا تنظیم شده است، بعد از اجرای دستور RETURN کنترول به قسمت بعد از ساختار پروسیجر، یعنی به عوض سطر 6 به سطر 10 می‌رود؛ در نهایت سطر 10 بعد از سطر 4 اجرا شده و در نتیجه نشان داده می‌شود.

در صورت دور کردن دستور RETURN از مثال بالا نتیجه مطابق شکل پایین تغییر می‌کند.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   PROCEDURE p IS
 3     BEGIN
 4       DBMS_OUTPUT.PUT_LINE('Inside p');
 5       -- RETURN;
 6       DBMS_OUTPUT.PUT_LINE('Unreachable statement.');
 7     END;
 8   BEGIN
 9     p;
10   DBMS_OUTPUT.PUT_LINE('Control returns here.');
11 END;
12 /
Inside p
Unreachable statement.
Control returns here.

PL/SQL procedure successfully completed.

SQL>

```

شکل ۱۱-۱۰

با دور کردن دستور RETURN در سطر ۵، پرسیجر به شکل عادی اجرا شده و تمام دستورهای چاپ جمله‌ها (سطرهای ۴، ۶ و ۱۰) در نتیجه پروگرام نشان داده می‌شوند.

۱۰.۲.۶ استفاده از دستور RETURN در بلاک ناشناس

در یک بلاک ناشناس زمانی که عبارت RETURN استفاده شده باشد، با اجرای آن کنترول از بلاک خارج شده و در صورت استفاده از بلاک‌های آشیانه‌بی از تمام بلاک‌ها کنترول خارج شده و به قسمت بعدی می‌رود. استفاده از دستور RETURN مانند استفاده آن در پرسیجرها، یک عبارت Expression را مشخص کرده نمی‌تواند. مثال پایین در این مورد اجرا شده و طریقه استفاده از RETURN را در بلاک ناشناس نشان می‌دهد.

```

SQL> SET SERVEROUTPUT ON
SQL> BEGIN
 2   BEGIN
 3     DBMS_OUTPUT.PUT_LINE('Inside inner block.');
 4     RETURN;
 5     DBMS_OUTPUT.PUT_LINE('Unreachable statement.');
 6   END;
 7   DBMS_OUTPUT.PUT_LINE('Inside outer block. Unreachable statement.');
 8 END;
 9 /
Inside inner block.

PL/SQL procedure successfully completed.

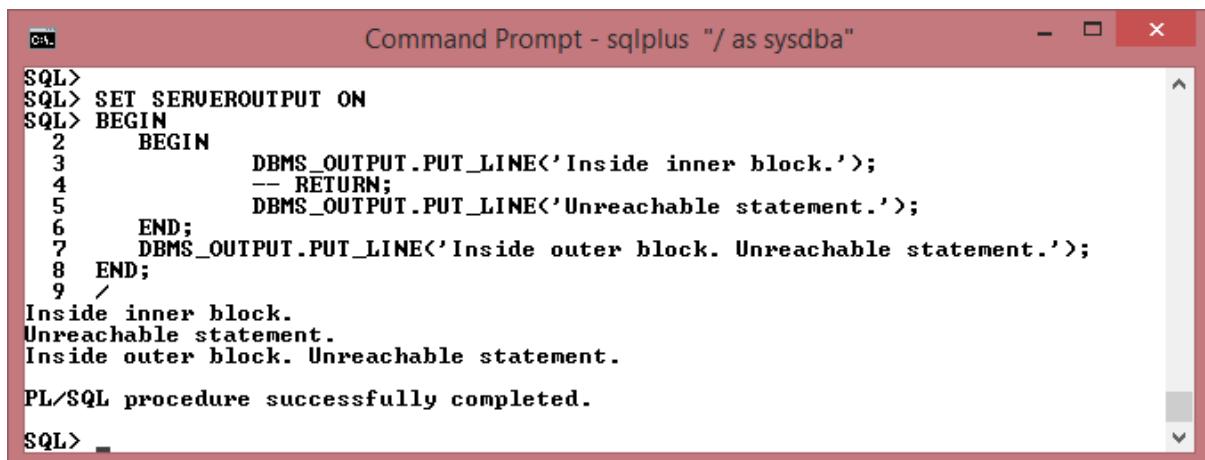
SQL>

```

شکل ۱۲-۱۰

طوری که دیده می‌شود، در نتیجه اجرای پروگرام، تنها دستور قبل از RETURN در سطر ۳ اجرا شده و جمله مورد نظر در قسمت نتیجه چاپ شده است. با اجراء شدن دستور RETURN در مثال بالا، کنترول از تمام بلاک‌های آشیانه‌بی خارج شده و دستورهای نوشته شده در سطرهای ۵ و ۷ اجرا نمی‌شوند.

اگر در همین مثال، عبارت RETURN دور شود، نتیجه طور ذیل تغییر می‌کند:



```
SQL> SET SERVEROUTPUT ON
SQL> BEGIN
 2   BEGIN
 3     DBMS_OUTPUT.PUT_LINE('Inside inner block.');
 4     -- RETURN;
 5     DBMS_OUTPUT.PUT_LINE('Unreachable statement.');
 6   END;
 7   DBMS_OUTPUT.PUT_LINE('Inside outer block. Unreachable statement.');
 8 END;
9 /
Inside inner block.
Unreachable statement.
Inside outer block. Unreachable statement.

PL/SQL procedure successfully completed.

SQL> _
```

شکل ۱۳-۱۰

در شکل بالا، مثال قبلی استفاده شده است طوری که دستور RETURN در سطر چهام به کمنت تبدیل شده است. با دورشدن دستور RETURN تمامی دستورهایی که خروجی دارند، به شکل عادی به اجراء درآمده و در نتیجهٔ پروگرام، جملات خروجی چاپ شده‌اند.

ساختن یک تابع ساده که پارامتر بگیرد.

یک تابع ساده که پارامتر بگیرد نیز با استفاده از دستور CREATE FUNCTION ساخته می‌شود. شکل عمومی دستور ایجادکردن یک تابع ساده با پارامترها در زبان PL/SQL طور ذیل است:

```
CREATE [ OR REPLACE ] FUNCTION function_name
[ (parameter_name [ IN | OUT | IN OUT ] type [, ...]) ]
RETURN return_datatype
{ IS | AS }
BEGIN
< function_body >
END [function_name];
```

در کد بالا، اکثر قسمت‌های کلیدی آن در درس قبلی به تفصیل توضیح شده‌اند. در اینجا به قسمت‌های دیگر به صورت کوتاه پرداخته شده و صرف قسمتی که به استفاده از پارامترها بستگی دارد، با تفصیل بیشتر توضیح می‌شود:

- نام تابع به خاطر شناسایی و استفاده تابع‌ها به کار برد می‌شود. در سطر اول، function_name نام تابع حتمی بوده و در قسمت آخر دستور به شکل اختیاری استفاده می‌شود.
 - ... parameter_name: استفاده از پارامترها در تابع‌ها اختیاری است. زمانی که تابع ساده با پارامترها تعریف می‌شود، در آن باید برای پارامتر نام شناسایی، حالت و نوعیت تعیین شود. به خاطر نوشتن نام باید استندردهای نوشتن شناسه‌ها در زبان PL/SQL در نظر گرفته شوند. حالت پارامترها در یک تابع سه نوع IN OUT و OUT اند؛ یعنی یک پارامتر در یک تابع باید یکی از این سه حالت را داشته باشد. حالت IN قیمتی را نشان می‌دهد که از بیرون به پارامتر داده می‌شود. حالت OUT خروجی پارامتر بوده که بعد از اجراء آن را نشان می‌دهد.
 - [OR REPLACE] : این قسمت اختیاری بوده و به این خاطر استفاده می‌شود که اگر از قبل به همین نام کدام تابع موجود باشد، تغییرات درخواست شده به آن تطبیق شده و از غلطی جلوگیری شود.
 - RETURN: این دستور از جمله قسمت‌های اساسی تابع به شمار رفته و خروجی تابع را حتمی می‌سازد.
 - return_datatype: این قسمت به خاطر مشخص کردن نوع دیتای خروجی تابع استفاده می‌شود. دیتای خروجی تابع عبارت از دیتایی است که بعد از اجرای تابع نمایش داده شده و قابل استفاده می‌شود.
 - { IS | AS }: اگر هدف این باشد که تابع مستقل (Standalone) ساخته شود، باید از کلمه AS در قسمت قبل از دستور BEGIN استفاده شود و در غیر آن کلمه IS استفاده می‌شود.
 - function_body: محتوای اصلی تابع شامل دستورهای اجرایی آن در این قسمت نوشته می‌شوند.
- در تعریف تابع‌های PL/SQL از اختیارهای لیست شده در جدول ذیل نیز استفاده می‌شود:

Option	Description
DETERMINISTIC option	Helps the optimizer avoid redundant function invocations.
PARALLEL_ENABLE option	Enables the function for parallel execution, making it safe for use in slave sessions of parallel DML evaluations.
PIPELINED option	Makes a table function pipelined, for use as a row source.
RESULT_CACHE option	Stores function results in the PL/SQL function result cache.

با در نظرداشت توضیحات فوق، یک تابع ساده می‌تواند با پارامترها طوری دیزاین شود که با تغییر قیمت‌ها نتایج مربوطه نشان داده شوند. در مثال، یک تابع مستقل با استفاده از پارامترها دیزاین و استفاده شده است؛ به شکل پایین در این مورد توجه شود.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
  2      -- Declare and define function
  3      FUNCTION square (original NUMBER) -- parameter list
  4          RETURN NUMBER -- RETURN clause
  5      AS
  6          -- Declarative part begins
  7          original_squared NUMBER;
  8      BEGIN           -- Executable part begins
  9          original_squared := original * original;
 10          RETURN original_squared; -- RETURN statement
 11      END;
 12  BEGIN
 13      DBMS_OUTPUT.PUT_LINE('Square of 25 is: ' || square(25)); -- invocation
 14      DBMS_OUTPUT.PUT_LINE('Square of 100 is: ' || square(100)); -- invocation
 15  END;
 16 /
Square of 25 is: 625
Square of 100 is: 10000
PL/SQL procedure successfully completed.

SQL>

```

شکل ۱۴-۱۰

طوری که در شکل بالا دیده می‌شود، در سطر سوم تعریف یک تابع ساده به نام square آغاز شده است که قیمت را از پارامتر می‌گیرد. سطر چهارم نوعیت خروجی تابع را با استفاده از کلمه RETURN مشخص می‌کند. کلمه AS مستقل بودن این تابع را در سطر ۵ نشان می‌دهد. سطرهای ۸ الی ۱۱ قسمت اصلی تابع (دستورهای اجرایی آن) را در بر دارد. سطرهای ۱۲ الی ۱۵ قسمت اجرایی پروگرام را در بر داشته که در آن دو بار تابع square با پارامترها یا قیمت‌های مختلف استفاده شده و نتیجه اجرای موقانه پروگرام نیز نشان داده شده است.

۱۰.۳ فرق بین پروسیجرها و تابع‌ها

پروسیجرها و تابع‌ها در زبان PL/SQL به هدف تعریف و به راه اندازی مجموعه‌یی از دستورها به خاطر انجام‌دادن یک کار معین استفاده می‌شوند. هر کدام آن‌ها به یک نام مسمی شده، متحولین و عملیه‌ها در آن‌ها گنجانیده شده و در سیستم ذخیره می‌شوند. بعد از اعلام و تعریف، هر کدام آن‌ها با استفاده از نام‌های‌شان به روی دستور اجرایی آمده، (Call) شده و به اجراء گذاشته می‌شوند. تفاوت‌ها بین تابع‌ها و پروسیجرها در چند مورد محدود دیده شده و در ذیل لیست گردیده‌اند:

- یک تابع (Function) در زبان PL/SQL باید دارای عبارت RETURN باشد. این عبارت نوعیت دیتایی را مشخص می‌کند که به حیث خروجی تابع تولید می‌شود. بر عکس، در پروسیجرها در زبان PL/SQL عبارت RETURN استفاده نمی‌شود؛ یعنی ضرورت به استفاده از آن ناست.
- در بخش دستورهای اجرایی یک تابع، هر دستور اجرایی باید به نحوی به دستور RETURN برگرد. اگر چنین پروگرام نشده باشد، PL/SQL غلطی یا اخطاریه زمان تفسیر (Compile Time Warning) را خواهد داد. در پروسیجر استفاده از عبارت RETURN به ترتیب بالا اختیاری بوده و استفاده از این دستور در پروسیجرها توصیه نمی‌شود.

شکل پایین، یک تابع به نام `findMax` را نشان می‌دهد که در آن دو عدد مقایسه شده و نتیجه آن‌ها نشان داده می‌شود.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      a number;
 3      b number;
 4      c number;
 5  FUNCTION findMax<x IN number, y IN number>
 6  RETURN number
 7  IS
 8      z number;
 9  BEGIN
10      IF x > y THEN
11          z:= x;
12      ELSE
13          z:= y;
14      END IF;
15      RETURN z;
16  END;
17  BEGIN
18      a:= 23;
19      b:= 45;
20      c := findMax(a, b);
21      dbms_output.put_line(' Maximum of ' || a || ' and '
22                                || b || ' is: ' || c);
23  END;
24 /
Maximum of 23 and 45 is: 45
PL/SQL procedure successfully completed.
SQL>

```

شکل ۱۵-۱۰

طوری که در شکل بالا دیده می‌شود، دو عدد 23 و 45 توسط تابع با هم مقایسه شده و قیمت بزرگ‌تر آن‌ها نشان داده شده است. همین کار با تغییر علامه 45 دوباره اجرا شده و نتیجه آن مطابق شکل پایین با تغییر نشان داده شده است.

```

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2      a number;
 3      b number;
 4      c number;
 5  FUNCTION findMax<x IN number, y IN number>
 6  RETURN number
 7  IS
 8      z number;
 9  BEGIN
10      IF x > y THEN
11          z:= x;
12      ELSE
13          z:= y;
14      END IF;
15      RETURN z;
16  END;
17  BEGIN
18      a:= 23;
19      b:= -45;
20      c := findMax(a, b);
21      dbms_output.put_line(' Maximum of ' || a || ' and '
22                                || b || ' is: ' || c);
23  END;
24 /
Maximum of 23 and -45 is: 23
PL/SQL procedure successfully completed.
SQL>

```

شکل ۱۶-۱۰

طوری که دیده می‌شود، تابع `findMax` به خاطر پیداکردن مقدار زیاد در بین دو عدد به صورت درست کار کرده و در حالت‌های بالا نتایج دقیق ارائه داشته است

۱۰.۴ تریگرهای PL/SQL

یک تریگر عبارت از بخشی در زبان PL/SQL است که در آن دستورهای اجرایی جایه‌جا شده و در دیتابیس به یک نام ذخیره شده باشد. تریگرهای طوری تنظیم می‌شوند که در زمان اجراء (با اجراشدن) یک عملیه در دیتابیس این‌ها نیز اجراء شوند. اجرای تریگرهای ممکن است قبل از یک عملیه و یا هم بعد از یک عملیه دیتابیس پلان شود؛ طور مثال، زمانی که به یک جدول مشخص شده در یک دیتابیس، دیتا داخل می‌شود و یا قبل از داخل شدن دیتا به جدول مشخص، تریگر به راه افتاد و دستورهای تعیینه شده در آن اجراء شوند.

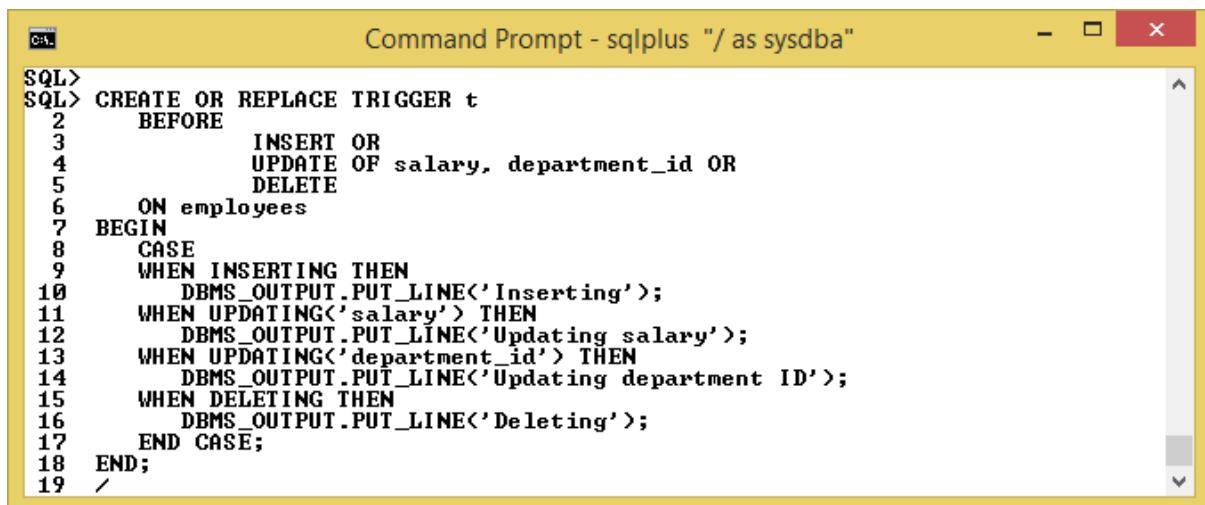
در زبان PL/SQL تریگرهای مشابهت زیاد به پروسیجرها دارند. یکی از تفاوت‌های عمده تریگرهای پروسیجرها، در فعال شدن و غیر فعال شدن است. تریگرهای صرف در حالت فعال بودن اجراء می‌شوند. اگر یک تریگر طوری تنظیم شده باشد که با پاک کردن دیتا از یک جدول باید به راه افتاد، ولی در عین زمان غیر فعال (Disable) ساخته شده باشد؛ در این حالت تریگر مورد نظر، خنثی بوده و در زمان معین اجراء نمی‌شود. نکته دیگری که در قسمت تریگرهای مشهود است، عبارت از تنظیم شدن تریگرهای تنها با فعالیت‌های سیستم است؛ یعنی زمانی که در سیستم دیتابیس کاری انجام می‌شود، تریگرهای خاطر کمک و یا کنترول چنین کارهایی ساخته می‌شوند. کارهایی که مربوط استفاده کنندگان باشد، توسط تریگرهای کمک و یا کنترول نمی‌شوند.

موضوعات ذیل در قسمت استفاده از تریگرهای در نظر گرفته شوند:

- یک تریگر در زبان PL/SQL با استفاده از دستور `CREATE TRIGGER` ایجاد می‌شود.
- Event‌هایی که توسط تریگر در نظر گرفته شده و بالای آن عمل انجام می‌شود، در زمان ایجاد کردن تریگرهای باید مشخص شوند؛ طور مثال، با پاک کردن دیتا از یک جدول مشخص، اضافه کردن دیتا به یک جدول مشخص، اضافه کردن و یا تغییرآوردن ساختمان جدول در یک دیتابیس مشخص و غیره عبارت از Event‌های سیستم‌اند.
- یک تریگر در زبان PL/SQL می‌شود برای یک ساختمان دیتابیس از قبیل جدول و یا نما و یا هم خود دیتابیس ساخته و تعریف شود.
- برای یک تریگر نقطه زمان اجراء نیز تعیین می‌گردد. نقطه زمان اجراء می‌شود قبل از اجرای یک کار و یا هم بعد از اجرای یک کار مشخص در دیتابیس باشد. نقطه زمان اجراء، برعلاوه ایجاد تغییرات در جدول، برای کنترول و راپوردهای تغییرات هر ریکورد یک جدول نیز قابل تعریف است.
- در حالت عادی یک تریگر وقتی ایجاد می‌شود، حالت فعال (Enable) دارد. به خاطر غیر فعال کردن تریگرهای از اختیارات دیگر باید استفاده شود.

- اگر یک تریگر برای جدول و یا نما در یک دیتابیس ساخته شده باشد که در آن جدول و یا نما دستورهای کار با یوزر دیتا (DML) استفاده می‌شوند، آن تریگر از نوع DML Trigger شمرده می‌شود.
- در صورت استفاده کردن یک تریگر به منظور کار با میتادیتا یا DDL (ساختارهای دیتابیس‌ها، جدول‌ها، نماها و غیره)، آن را به نام System Trigger یاد می‌کنند.
- تریگر مشروط (Conditional) نوع دیگری از این ساختمان‌هاست که با درنظرداشت شرط WHEN تنظیم شده و کار می‌کند.

یک مثال استفاده از تریگرها در PL/SQL در نظر گرفته شده و به خاطر وضاحت موضوع در ذیل ارائه شده است. در این مثال یک تریگر مشروط (نوع آخر لیست شده در بالا) به کار رفته است.



```

SQL> CREATE OR REPLACE TRIGGER t
  2    BEFORE
  3      INSERT OR
  4      UPDATE OF salary, department_id OR
  5      DELETE
  6    ON employees
  7    BEGIN
  8      CASE
  9        WHEN INSERTING THEN
 10          DBMS_OUTPUT.PUT_LINE('Inserting');
 11        WHEN UPDATING('salary') THEN
 12          DBMS_OUTPUT.PUT_LINE('Updating salary');
 13        WHEN UPDATING('department_id') THEN
 14          DBMS_OUTPUT.PUT_LINE('Updating department ID');
 15        WHEN DELETING THEN
 16          DBMS_OUTPUT.PUT_LINE('Deleting');
 17      END CASE;
 18    END;
 19  /

```

شکل ۱۷-۱۰

در شکل بالا تریگری به نام t ساخته شده و در آن با استفاده از ساختار CASE چهار حالت یا شرط به کمک کلمه WHEN در نظر گرفته شده است. تمام دستورها مربوط کار با دیتا (DML) بوده و زمانی که در جدول employees، مطابق سطر 6، کاری از قبیل داخل کردن دیتا، به روز کردن دیتا و یا هم پاک کردن بخش‌های مشخص دیتابیس جدول یاد شده صورت گیرد، تریگر به راه افتاده و پیام خروجی مورد نظر را نشان خواهد داد.

۱۰.۵ بسته‌ها در PL/SQL (Packages)

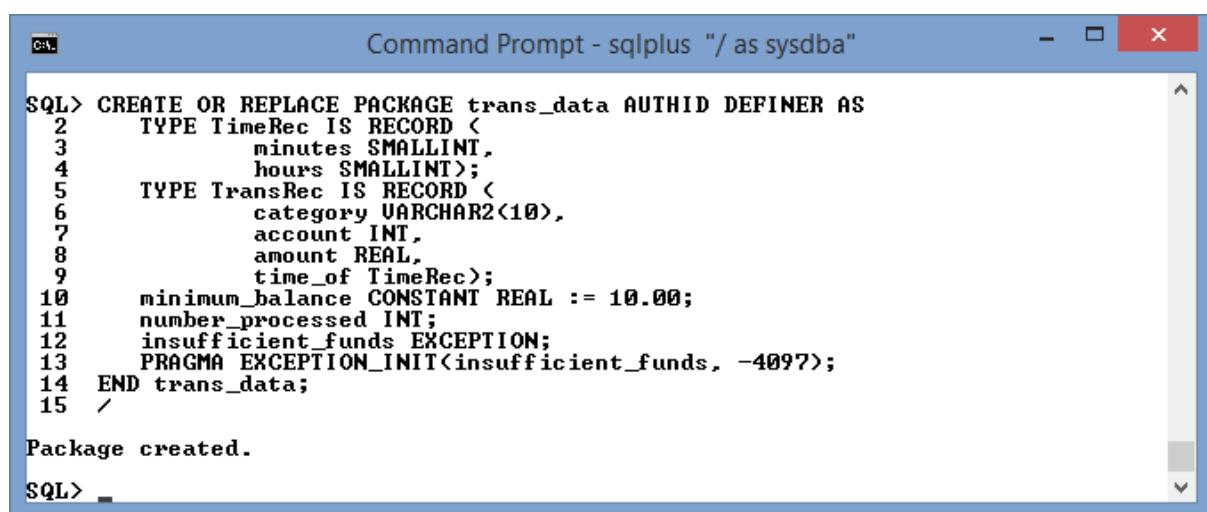
یک بسته (Package) در زبان PL/SQL عبارت از دسته‌یی از نوعیت‌ها (Types)، متغولین (Variables)، ثابت‌ها (Constants)، پروگرام‌های فرعی (Subprograms)، کرسرهای (Cursors) و استثناهای (Exceptions) بوده که به شکل منطقی با هم رابطه داشته باشند. یک بسته در دیتابیس تنظیم و ذخیره می‌شود. همین

امر باعث می شود تا اپلیکیشن های مختلفی که به دیتابیس دسترسی دارند، بتوانند از اجزای بسته استفاده کنند.

یک بسته همیشه دارای خصوصیات است. با استفاده از خصوصیات، امکانات استفاده از عناصر عمومی شامل بسته، تنظیم و برنامه ریزی می شود. در صورتی که یک بسته دارای کرسرهای پروگرامی فرعی باشد، موجودیت بدن (Body) در ساختمان بسته حتمی بوده و در غیر آن وجود بدن در بسته اختیاری می شود. در بدن باید کیوری هایی تعریف شده باشند که توسط آنها کرسرهای عمومی قابل استفاده و کد به خاطر دسترسی به پروگرام های فرعی تنظیم شده باشد.

در قسمت بدن بسته در زبان PL/SQL عناصر خصوصی (Private) به خاطر استفاده داخلی بسته نیز تنظیم می شوند. اجزاء یا عناصر خصوصی جهت کارهای داخلی یک بسته مهم می باشند. قسمت بدن یک بسته، به عین شکل دارای یک بخش به نام آغازسازی (Initialization) است که در آن معرفی متغولین و تنظیم هایی که برای یکبار استفاده می شوند (One time Setups)، تعریف می گردند. قسمت بررسی استثناء (Exception Handling) نیز در بدن بسته زبان PL/SQL اضافه می شود.

در ارتباط به موضوع یک مثال در نظر گرفته شده است:



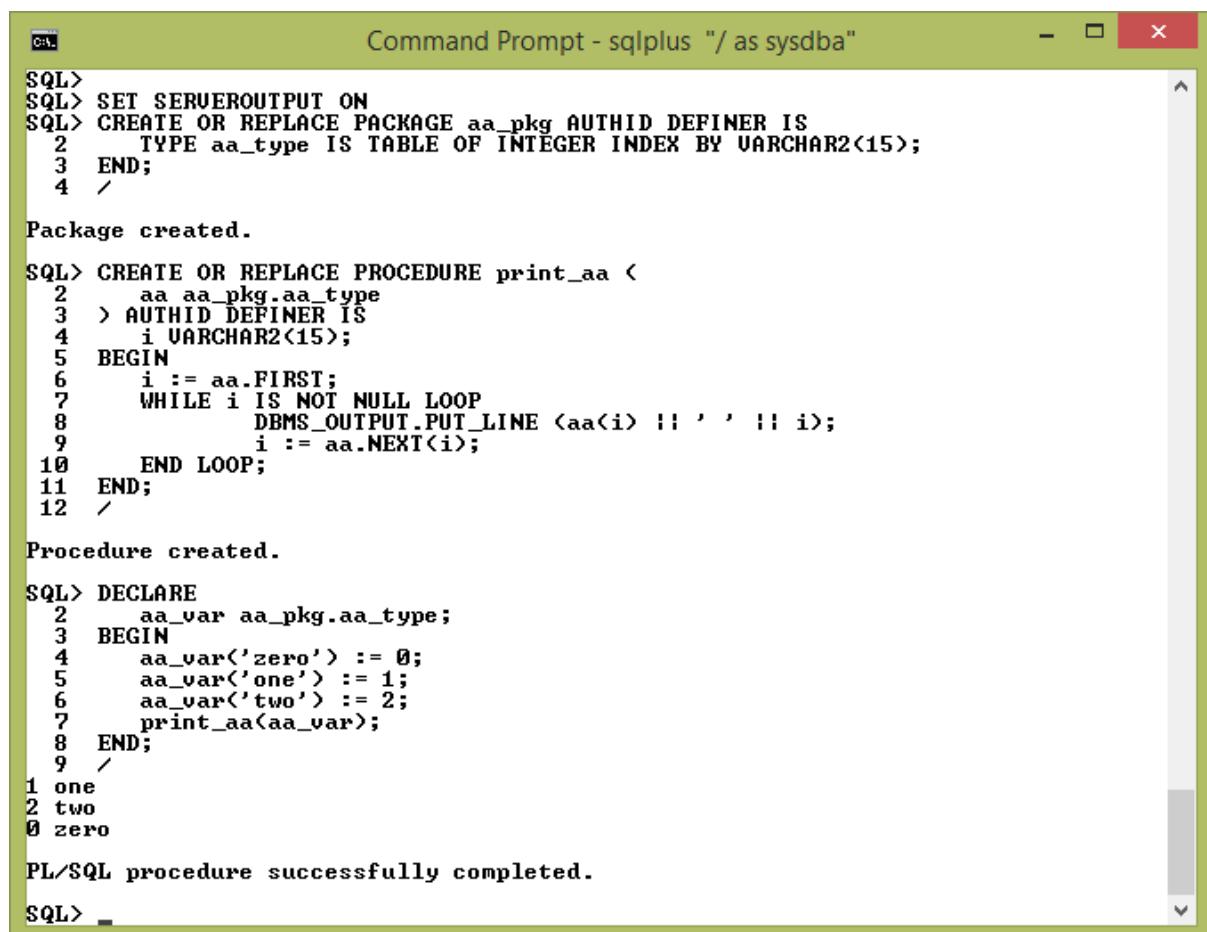
```
SQL> CREATE OR REPLACE PACKAGE trans_data AUTHID DEFINER AS
 2   TYPE TimeRec IS RECORD (
 3     minutes SMALLINT,
 4     hours SMALLINT);
 5   TYPE TransRec IS RECORD (
 6     category VARCHAR2(10),
 7     account INT,
 8     amount REAL,
 9     time_of TimeRec);
10   minimum_balance CONSTANT REAL := 10.00;
11   number_processed INT;
12   insufficient_funds EXCEPTION;
13   PRAGMA EXCEPTION_INIT<insufficient_funds, -4097>;
14 END trans_data;
15 /
Package created.
SQL>
```

شکل ۱۸-۱۰

در مثال بالا با استفاده از ایجاد کردن بسته بیی به نام trans_data، دو نوع عمومی در سطرهای ۳ و ۴ ایجاد شده اند و سه متغول عمومی در سطرهای بعدی تعریف شده اند.

یک مثال دیگر با تفصیل بیشتر به خاطر وضاحت موضوع در نظر گرفته شده است که در آن با رشته ها (Arrays) در زبان PL/SQL کار شده است.

به شکل پایین توجه شود.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - sqlplus "/ as sysdba". The SQL*Plus session displays the creation of a package named aa_pkg with a type aa_type, a procedure print_aa that prints the values of aa_type, and a block of PL/SQL code that declares aa_var of type aa_type and initializes it to 0, 1, or 2. The session concludes with the message "PL/SQL procedure successfully completed."

```
SQL> SET SERVEROUTPUT ON
SQL> CREATE OR REPLACE PACKAGE aa_pkg AUTHID DEFINER IS
 2   TYPE aa_type IS TABLE OF INTEGER INDEX BY VARCHAR2(15);
 3 END;
 4 /
Package created.

SQL> CREATE OR REPLACE PROCEDURE print_aa (
 2   aa aa_pkg.aa_type
 3 ) AUTHID DEFINER IS
 4   i VARCHAR2(15);
 5 BEGIN
 6   i := aa.FIRST;
 7   WHILE i IS NOT NULL LOOP
 8     DBMS_OUTPUT.PUT_LINE (aa(i) || ' ' || i);
 9     i := aa.NEXT(i);
10   END LOOP;
11 END;
12 /
Procedure created.

SQL> DECLARE
 2   aa_var aa_pkg.aa_type;
 3 BEGIN
 4   aa_var('zero') := 0;
 5   aa_var('one') := 1;
 6   aa_var('two') := 2;
 7   print_aa(aa_var);
 8 END;
 9 /
1 one
2 two
0 zero

PL/SQL procedure successfully completed.

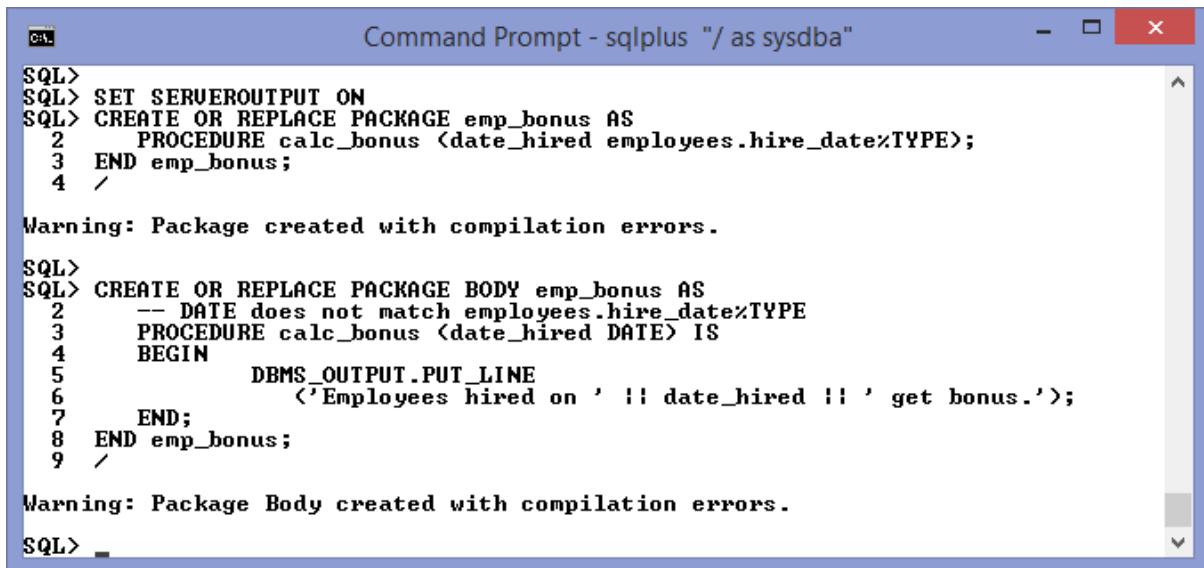
SQL> _
```

شکل ۱۹-۱۰

طوری که در شکل دیده می‌شود، در نخست یک بسته ایجاد شده است که به تعقیب آن پروسیجر تعریف شده و در نهایت، یک پروگرام فرعی به اجراء در آمده است.

در قسمت اول، بسته‌یی به نام aa_pkg ایجاد شده است که در آن یک رشته به نام aa_type تعریف شده است. بعد از ایجادشدن موفقانه بسته، پروسیجر print_aa نیز موفقانه ایجاد شده که در آن با استفاده از اختیار aa_pkg.aa_type پارامتری از بسته قبلی استفاده شده است. در قسمت آخر پروگرام فرعی نیز یک متحول را از نوع aa_type گرفته و آن را به دستور چاپ کردن می‌رساند. بعد از اجرای پروگرام فرعی، نتیجه نهایی به صورت موفقانه در قسمت آخر شکل بالا نشان داده شده است.

یک مثال دیگر به خاطر ایجاد کردن یک بسته با بدن (Body) در نظر گرفته شده است. شکل پایین نمونه کد ایجاد کردن بسته و بدن آن را در زبان PL/SQL نشان می‌دهد.



```
SQL> SET SERVEROUTPUT ON
SQL> CREATE OR REPLACE PACKAGE emp_bonus AS
 2   PROCEDURE calc_bonus <date_hired employees.hire_date%TYPE>;
 3 END emp_bonus;
4 /Warning: Package created with compilation errors.

SQL> CREATE OR REPLACE PACKAGE BODY emp_bonus AS
 2   -- DATE does not match employees.hire_date%TYPE
 3   PROCEDURE calc_bonus <date_hired DATE> IS
 4     BEGIN
 5       DBMS_OUTPUT.PUT_LINE
 6         '<Employees hired on ' || date_hired || ' get bonus.'';
 7     END;
 8 END emp_bonus;
9 /Warning: Package Body created with compilation errors.

SQL> _
```

شکل ۲۰-۱۰

در شکل بالا ایجاد بخش‌های درخواست شده با اخطاریهای همراه است که به خاطر عدم تطابق کلمه‌های استفاده شده است. به هر ترتیب، هدف مثال نشان دادن طریقه ایجاد کردن بدن برای یک بسته در زبان PL/SQL بوده و در شکل بالا نشان داده شده است.

۱۰.۶ مجموعه‌ها (Collections)

مجموعه‌ها در زبان PL/SQL عبارت از نوعی دیتای ترکیبی (Composite Datatype) بوده که در آن عناصر دیتا طوری تنظیم می‌شوند که عین نوعیت دیتا را داشته باشند. هر جزء دیتا که در یک مجموعه استفاده می‌شود به نام عنصر مجموعه یاد می‌شود. نوع دیگر دیتای ترکیبی به نام ریکوردها (Records) یاد می‌شود که در اینجا از تفصیل بیشتر در مورد آن صرف نظر صورت گرفته است.

در یک مجموعه، هر عنصر دارای یک اندکس به خاطر شناسایی و دسترسی به آن است؛ یعنی متحولین مجموعه از طریق اندکس‌های غیر تکراری آن‌ها قابل استفاده‌اند. طریقه استفاده از یک عنصر مجموعه عبارت از variable_name(index) است.

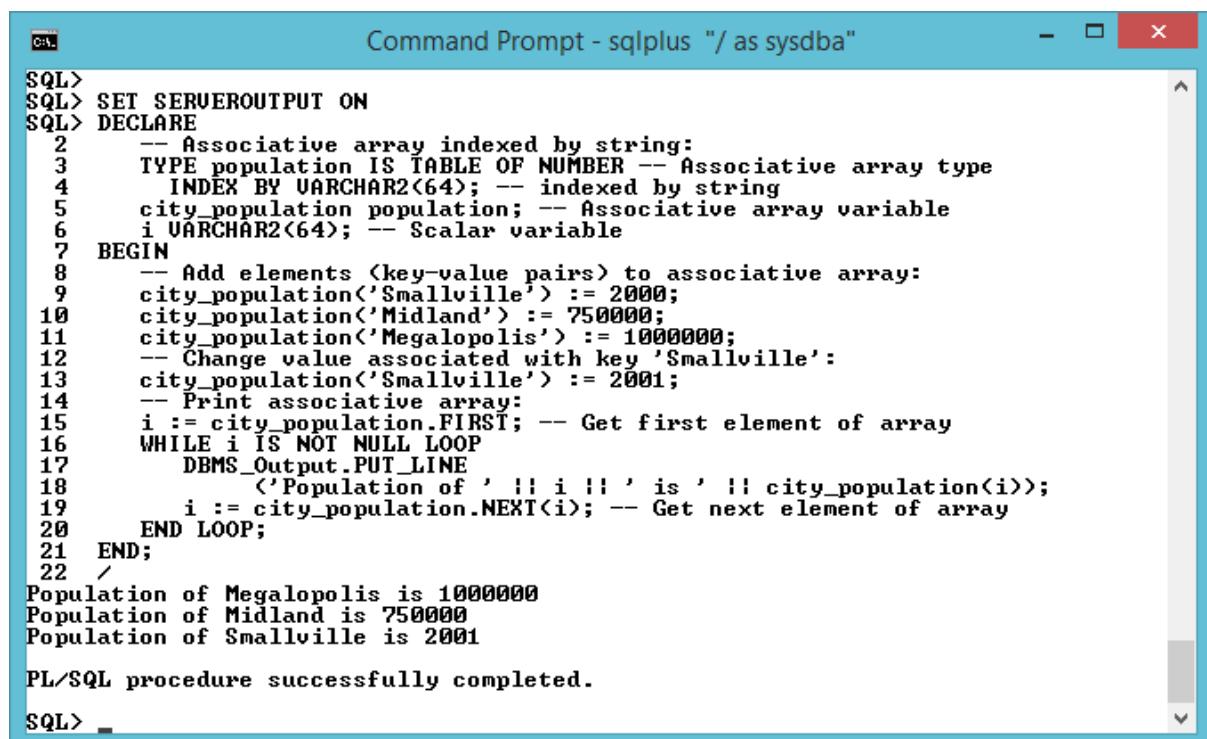
به خاطر ایجاد کردن یک مجموعه متحولین در PL/SQL، در قدم نخست یک مجموعه ایجاد شده و بعد متحولین از همان نوع یا به طور مستقیم ایجاد می‌شوند و یا هم با استفاده از عبارت %TYPE نوعیت‌شان از یک منبع دیگر گرفته می‌شود. متحولین ایجاد شده در یک مجموعه، به شکل پارامترها به پروگرام‌های فرعی به شکل‌های گروپی و یا انفرادی فرستاده و استفاده می‌شوند.

عناصر داخلی یک مجموعه می‌توانند به شکل ترکیبی و یا هم به شکل سکالری تعریف شوند. یک مجموعه در زبان PL/SQL می‌تواند یکی از سه نوع باشد:

- مجموعه رشتہ وابسته (Associative Array)
- مجموعه رشتہ متغیر (Variable Size Array) یا VARRAY
- مجموعه جدول‌های آشیانه‌یی (Nested Table)

به خاطر محدودیت حجم کتاب از تفصیل بیشتر در مورد انواع مجموعه‌ها صرف نظر شده و موضوع با مثالی دنبال می‌گردد. شاگردان علاقه‌مند می‌توانند به کتاب‌های لیست‌شده در منابع و ریفرنس‌ها مراجعه کرده و معلومات حاصل کنند.

به شکل پایین توجه شود.



```
Command Prompt - sqlplus "/ as sysdba"

SQL> SET SERVEROUTPUT ON
SQL> DECLARE
 2   -- Associative array indexed by string:
 3   TYPE population IS TABLE OF NUMBER -- Associative array type
 4     INDEX BY VARCHAR2(64); -- indexed by string
 5   city_population population; -- Associative array variable
 6   i VARCHAR2(64); -- Scalar variable
 7   BEGIN
 8     -- Add elements <key-value pairs> to associative array:
 9     city_population('Smallville') := 2000;
10     city_population('Midland') := 750000;
11     city_population('Megalopolis') := 1000000;
12     -- Change value associated with key 'Smallville':
13     city_population('Smallville') := 2001;
14     -- Print associative array:
15     i := city_population.FIRST; -- Get first element of array
16     WHILE i IS NOT NULL LOOP
17       DBMS_Output.PUT_LINE
18         ('Population of ' || i || ' is ' || city_population(i));
19       i := city_population.NEXT(i); -- Get next element of array
20     END LOOP;
21   END;
22 /
Population of Megalopolis is 1000000
Population of Midland is 750000
Population of Smallville is 2001

PL/SQL procedure successfully completed.

SQL>
```

شکل ۲۱-۱۰

مثال بالا از نوع رشتہ وابسته (Associative Array) کار شده است. در مثال مذکور، یک نوع از این رشتہ تعریف شده و با سه عنصر استفاده شده است. نتیجه ایجاد و استفاده کردن مجموعه زبان PL/SQL به شکل موفقانه نشان داده شده است.



خلاصه فصل دهم

این فصل به موضوعات فرعی ولی مهم در زبان PL/SQL اشاره داشت. برنامه‌های فرعی در زبان PL/SQL، خصوصیات و بخش‌های آن با مثال‌های لازم توضیح داده شده‌اند. بلاک‌های ناشناس که به نام Anonymous Blocks یاد می‌شوند، توضیح داده شده و ایجاد کردن یک پروسیجر PL/SQL توسط این بلاک‌ها شرح شده است. ساختن تابع‌های ساده و تابع‌های ساده با پارامترها، با ذکر مثال‌های جداگانه تحت عناوین جداگانه توضیح داده شده‌اند. بحث روی فرق بین پروسیجرها و تابع‌ها از عنوان‌های دیگر این فصل‌اند.

فصل دهم با توضیحات در مورد تریگرهای (Triggers) و موارد استفاده از آن‌ها با مثال ادامه پیدا کرده است. بسته‌ها (Packages) منحیت ساختمان‌های قابل استفاده به خاطر مدیریت بهتر پروگرام‌ها با توضیح انواع مختلف پکیج‌ها از نظر محتواهای آن‌ها با مثال‌های لازم توضیح شده‌اند. این فصل با ارائه معلومات و مثال‌ها در مورد مجموعه‌ها (Collections) در زبان PL/SQL به پایان رسیده است. انواع مختلف مجموعه‌ها لیست گردیده و از توضیحات بیشتر در این مورد به خاطر کنترول حجم کتاب جلوگیری شده است.



سوالات فصل دهم

۱. یک پروگرام فرعی (Subprogram) را در زبان PL/SQL توضیح دهید.
۲. یک پروسیجر را در زبان PL/SQL توضیح دهید.
۳. یک تابع را در زبان PL/SQL توضیح دهید.
۴. فرق عمدی بین پروسیجر و تابع را در زبان PL/SQL واضح سازید.
۵. یک تریگر (Trigger) را در زبان PL/SQL توضیح دهید.
۶. تفاوت‌های عمدی بین تریگر و پروسیجر را در زبان PL/SQL واضح سازید.
۷. تریگر فعال (Enable) با تریگر غیر فعال شده (Disabled) چه تفاوت دارد؟ با مثال واضح سازید.
۸. یک بسته (Package) را در زبان PL/SQL توضیح دهید.
۹. یک مجموعه (Collection) را در زبان PL/SQL توضیح دهید.
۱۰. یک پروگرام فرعی زبان PL/SQL دارای کدام بخش‌ها است؟ توضیح دهید.
۱۱. هدف از بلاک‌های ناشناس (Anonymous Blocks) در زبان PL/SQL چیست؟ واضح سازید.
۱۲. به چند طریقه کد زبان PL/SQL در یک صفحه کدکردن این زبان مانند سیکویل پلس به اجراء گذاشته می‌شود؟ هر کدام آن را به اختصار توضیح دهید.
۱۳. یک فایل اجرایی کد زبان PL/SQL با کدام اکستنشن ذخیره می‌شود تا قابل اجراء گردد؟ با مثال واضح سازید.
۱۴. آیا امکان دارد که یک بلاک ناشناس (Anonymous Block) نام داشته باشد؛ یعنی با یک نام تعریف شده باشد؟ اگر جواب بلی است، پس توضیح دهید که با وجود داشتن نام، چرا به آن بلاک ناشناس می‌گویند؟
۱۵. تفاوت‌های عمدی بین بلاک‌های ناشناس و بلاک‌های شناخته شده در زبان PL/SQL را با مثال توضیح دهید.
۱۶. کدام دستور در یک تابع باعث می‌شود که تابع یک نتیجه (یک قیمت) را نشان بدهد (تابع را از پروسیجر متمازیز سازد).
۱۷. شکل عمومی نوشتن یک تابع ساده را در زبان PL/SQL بنویسید.
۱۸. شکل عمومی نوشتن یک تابع ساده زبان PL/SQL را که پارامتر داشته باشد، بنویسید.



فعالیت های فصل دهم

۱. با مراجعه به فعالیت فصول قبلی، صفحهٔ سیکویل پلس را با استفاده از دستور `sqlplus "/ as sysdba"` باز کنید.
۲. برنامهٔ Notepad را به خاطر نوشتن و ایدیت کردن کد PL/SQL باز کنید.
۳. یک پروسیجر ساده PL/SQL به نام Procedure1 بنویسید که در آن خصوصیت AUTHID استفاده شده و چند دستور اجرایی داشته باشد.
۴. یک تابع ساده به نام Function1 ایجاد کنید که توسط آن عملیه‌های ساده ریاضی انجام شود.
۵. تابع ایجادشده در دستور قبلی را با استفاده از یک پروگرام فرعی استفاده (Call) کنید.

مأخذ (References)

1. Arakelian, A. et al. (2015). *Oracle Installation Guide*. Boston University, USA.
2. Fogel, S. et al. (2015). *Oracle Database Administrator's Guide: 11g Release 2*. Oracle Corporations, USA.
3. Greenwals, R., Stackowia, R. and Stern, J. (2008). *Oracle Essentials: Oracle Database 11g – 4th Edition*. O'Reilly Press, USA.
4. Jashnani, P. and Williams, D. (2017). *Oracle Database Installation Guide 11g Release 2*. Oracle Corporations, USA.
5. Jashnani, P. and Williams, D. (2017). *Oracle Database Installation Guide 12c Release 1*. Oracle Corporations, USA.
6. Moore, S. et al. (2010). *Oracle Database – PL/SQL Language Reference, 11g Release*. Oracle Corporations, USA.
7. Morin, L. et al. (2017). *Oracle Database – Database PL/SQL Language Reference, 12c Release 2*. Oracle Corporations, USA.
8. Rosenzweig, B. and Rakhimov, E. S. (2009). *Oracle PL/SQL by Example – 4th Edition*. Pearson Education, Inc. Boston, MA, USA.
9. e-Book: *PL/SQL Programming Tutorial*. By Tutorials Point Pvt. Ltd. <http://tutorialspoint.com>. (2018).
10. Official Doc: *Oracle 12c Installation and Configuration Guide*. Hewlett Packard Enterprise Development, HP Co. (2015).
11. Official Doc: *Oracle Database New Features Guide 12c Release 1*. Oracle Corporations, USA. (2015).
12. Official Doc: *Oracle Database PL/SQL User's Guide and Reference 10g Release 2*. Oracle Corporations, USA. (2005).
13. Official Doc: *PL/SQL Enhancements in Oracle Database 11g*. An Oracle White Paper. Oracle Corporations, USA. (2007).