Oracle Database 11g: Advanced PL/SQL (D52601)

ID D52601 **Preis** 2.010,- € (exkl. MwSt.) **Dauer** 3 Tage

Voraussetzungen

Erforderlich

- Professioneller Einstieg in Oracle Database 11g SQL
- Oracle Datenbank 11g: PL/SQL und Datenbankprogrammierung

Empfohlen

- Erfahrung mit SQL und PL/SQL
- Vertrautheit mit der Oracle-Datenbank

Kursziele

- Codierungsstandards anwenden, um Sicherheitslücken in Bezug auf SQL-Injections zu vermeiden
- Verschiedene Typen von SQL-Injections-Angriffen kategorisieren und erläutern
- LOB-Datentypen erstellen und verwalten
- Collections erstellen und verwenden
- Subtypen auf Basis der für eine Anwendung vorhandenen Typen erstellen
- Features von SecureFile-LOBs beschreiben
- Prozess der fein granulierten Zugriffskontrolle beschreiben
- Deduplication, Komprimierung und Verschlüsselung für SecureFile-LOBs aktivieren
- Externe Java-Programme über PL/SQL ausführen
- Externe C-Programme über PL/SQL ausführen
- Richtlinien für den Cursor-Entwurf angeben
- Speicherauslastung durch Cachen von SQL-Ergebnismengen verbessern
- Profile von PL/SQL-Anwendungen erstellen
- PL/SQL-Funktionen so einrichten, dass sie PL/SQL-Ergebnis-Caching verwenden
- PL/SQL-Code optimieren
- LOBs mit dem PL/SQL-Package DBMS_LOB steuern

Kursinhalt

Die Teilnehmer dieses Kurses lernen, wie sie mit den fortgeschrittenen PL/SQL-Features PL/SQL-Code entwerfen und optimieren, um eine möglichst effiziente Schnittstelle zur Datenbank und zu anderen Anwendungen bereitzustellen. Sie erfahren, wie sich mit den fortgeschrittenen Features zur Programmentwicklung sowie mit Packages, Cursorn, erweiterten Schnittstellenmethoden, Large Objects und Collections leistungsstarke PL/SQL-Programme erstellen lassen. Weitere Themen dieses Kurses sind die effiziente Programmierung, die Einbindung externer C- und Java-Routinen, der fein granulierte Zugriff sowie das Schützen des Codes vor SQL-Injection-Angriffen. Learn To:

- Code erstellen, der als Schnittstelle zu externen Anwendungen und dem Betriebssystem fungiert
- Code als Schnittstelle zu Large Objects und zur Verwendung von SecureFile-LOBs erstellen
- PL/SQL-Anwendungen entwickeln, die Collections verwenden
- Code vor SQL-Injection-Angriffen schützen
- Virtual Private Databases mit fein granulierter Zugriffskontrolle implementieren
- PL/SQL-Packages und -Programmeinheiten entwerfen, die effizient ausgeführt werden

Entwicklungsumgebungen – Überblick

- SQL Developer
- SQL*Plus

Überlegungen zum Design

- Vordefinierte Datentypen beschreiben
- Subtypen auf Basis der für eine Anwendung vorhandenen Typen erstellen
- Verschiedene Richtlinien für den Cursor-Entwurf angeben

- Cursor-Variablen verwenden
- Cursor-Variablen als Programmparameter übergeben
- Cursor-Variablen mit statischen Cursorn vergleichen

Collections verwenden

- Collections – Überblick
- Assoziative Arrays
- Nested Tables
- Varrays
- PL/SQL-Programme entwickeln, die Collections verwenden
- Collections effektiv verwenden

Weitere Schnittstellenmethoden

- C über PL/SQL aufrufen
- Java über PL/SQL aufrufen

VDP mit fein granulierter Zugriffskontrolle implementieren

- Funktionsweise der fein granulierten Zugriffskontrolle verstehen
- Features der fein granulierten Zugriffskontrolle beschreiben
- Anwendungskontext beschreiben
- Anwendungskontext erstellen
- Anwendungskontext einstellen
- DBMS_RLS-Prozeduren auflisten
- Policys implementieren
- Dictionary View mit den Informationen zur fein granulierten Zugriffskontrolle abfragen

Large Objects bearbeiten

- Large Objects beschreiben
- Interne LOBs verwalten
- BFILEs beschreiben
- DIRECTORY-Objekt erstellen und damit auf BFILEs zugreifen
- DBMS_LOB-Package beschreiben
- LOBs entfernen
- Temporäres LOB mit dem DBMS_LOB-Package programmatisch erstellen

SecureFile-LOBs verwalten

- SecureFile-LOBs – Einführung
- Umgebung für SecureFile-LOBs aktivieren
- Mit SecureFile-LOBs Dokumente speichern
- BasicFile-LOBs in das SecureFile-LOB-Format konvertieren
- Performance von SecureFile-LOBs prüfen
- Deduplication und Komprimierung aktivieren
- Verschlüsselung aktivieren

Performance und Tuning

- Compiler verstehen und beeinflussen
- PL/SQL-Code optimieren
- Inlining innerhalb von Einheiten aktivieren
- Speicherprobleme identifizieren und optimieren

Performance mit SQL- und PL/SQL-Caching verbessern

- Ergebnis-Caching beschreiben
- Ergebnis-Cache für SQL-Abfragen
- Ergebnis-Cache für PL/SQL-Funktionen

PL/SQL-Code analysieren

- Mit bereitgestellten Packages und Dictionary Views Codierungsinformationen suchen
- Identifier-Typen und deren Verwendung mit PL/Scope ermitteln
- Mit dem DBMS_METADATA Package Metadaten aus dem Data Dictionary als XML oder Erstellungs-DDL abfragen, um die Objekte damit erneut zu erstellen

PL/SQL-Code – Tracing und Profilerstellung

- PL/SQL-Programmausführung tracen
- Profile von PL/SQL-Anwendungen erstellen

Code vor SQL-Injection-Angriffen schützen

- SQL-Injections beschreiben
- Angriffsflächen verringern
- DBMS_ASSERT verwenden
- Immunen Code entwickeln
- Code auf SQL-Injection-Schwachstellen testen