# Erste Schritte mit Oracle

Mit Materialien von Ulf Leser und Sebastian Wandelt

Patrick Schäfer

patrick.schaefer@hu-berlin.de

13. November 2017

Vorlesung: https://hu.berlin/vl_dwhdm17
Übung: https://hu.berlin/ue_dwhdm17

# Arbeiten mit Oracle

- Wir wollen ein Schema erstellen:

```
CREATE TABLE coworkers(
    c_id int          PRIMARY KEY,
    name varchar(25) NOT NULL
)
```

- Daten einfügen:

```
INSERT INTO coworkers
       VALUES (1, 'Herbert')
```

- Und Anfragen stellen:

```
SELECT count(c_id) as count
  FROM coworkers
```

# Überblick

- Verbinden mit Oracle
- JDBC
- PL/SQL
- Oracle SQL Developer

# Verbindungsdaten

- Server: alibaba.informatik.hu-berlin.de
- Port: 1521
- Datenbankinstanz (Oracle SID): orcl
- Version: Oracle Database 12c Enterprise Edition
- Wichtig:
  - Nur aus dem HU-Netz (VPN+WLAN) erreichbar
  - Jede Gruppe erhält eigenen Account und Tablespace. Mir eine Mail schreiben!

# Oracle Datenbankzugriff

- Beim Oracle Technology Network registrieren
  http://www.oracle.com/technology/

a) Alles in JAVA mit JDBC machen
   - JDBC Treiber für Oracle DB 12c herunterladen
     http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
   - Java programmieren

b) GUI-Tool verwenden
   - Empfohlen: Oracle SQL Developer
     http://www.oracle.com/technetwork/developer-tools/sql-developer/overview/index.html
   - Alternative: GUI-Tool (siehe Webseite) und Oracle Instant Client Libraries herunterladen
     http://www.oracle.com/technetwork/database/features/instant-client/index.html

# Zum Entwicklen / Testen

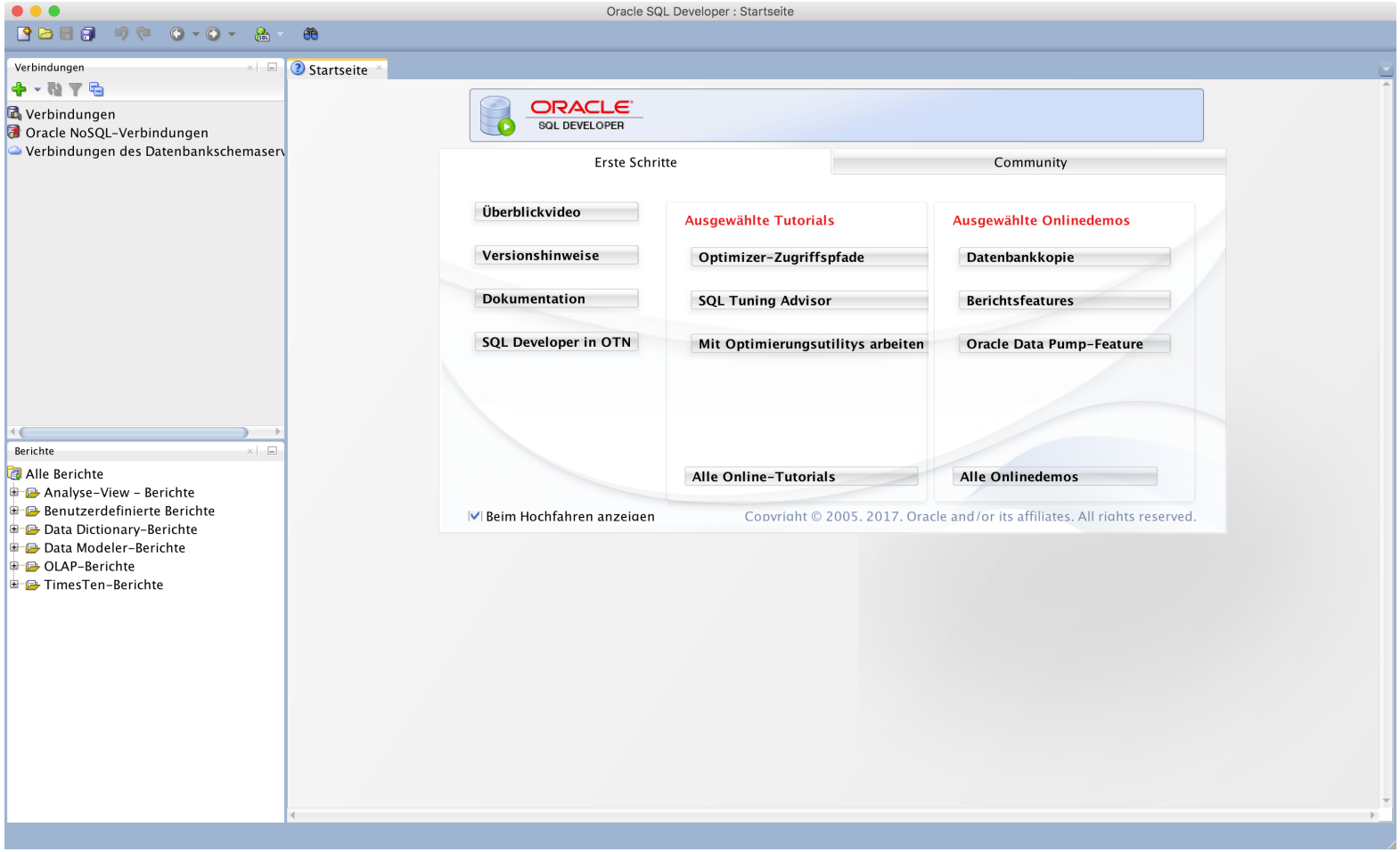
- Lokale Installation von

  Oracle 11g Express Edition (Windows+Linux)

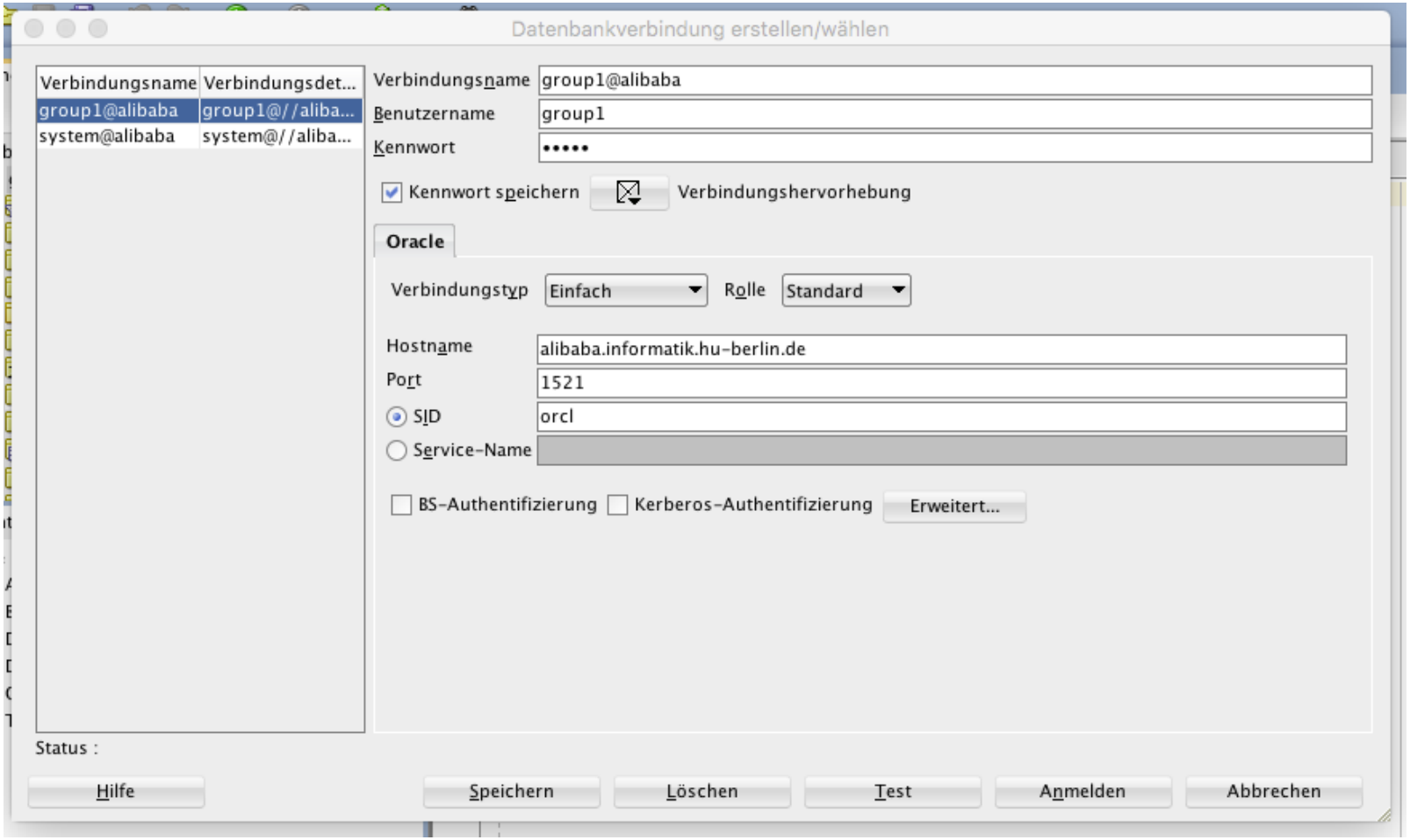
  „Oracle Database 11g Express Edition (Oracle Database XE) is an entry-level, small-footprint database based on the Oracle Database 11g Release 2 code base. It's free to develop, deploy, and distribute; fast to download; and simple to administer.“

  http://www.oracle.com/technetwork/database/database-technologies/express-edition/overview/index.html

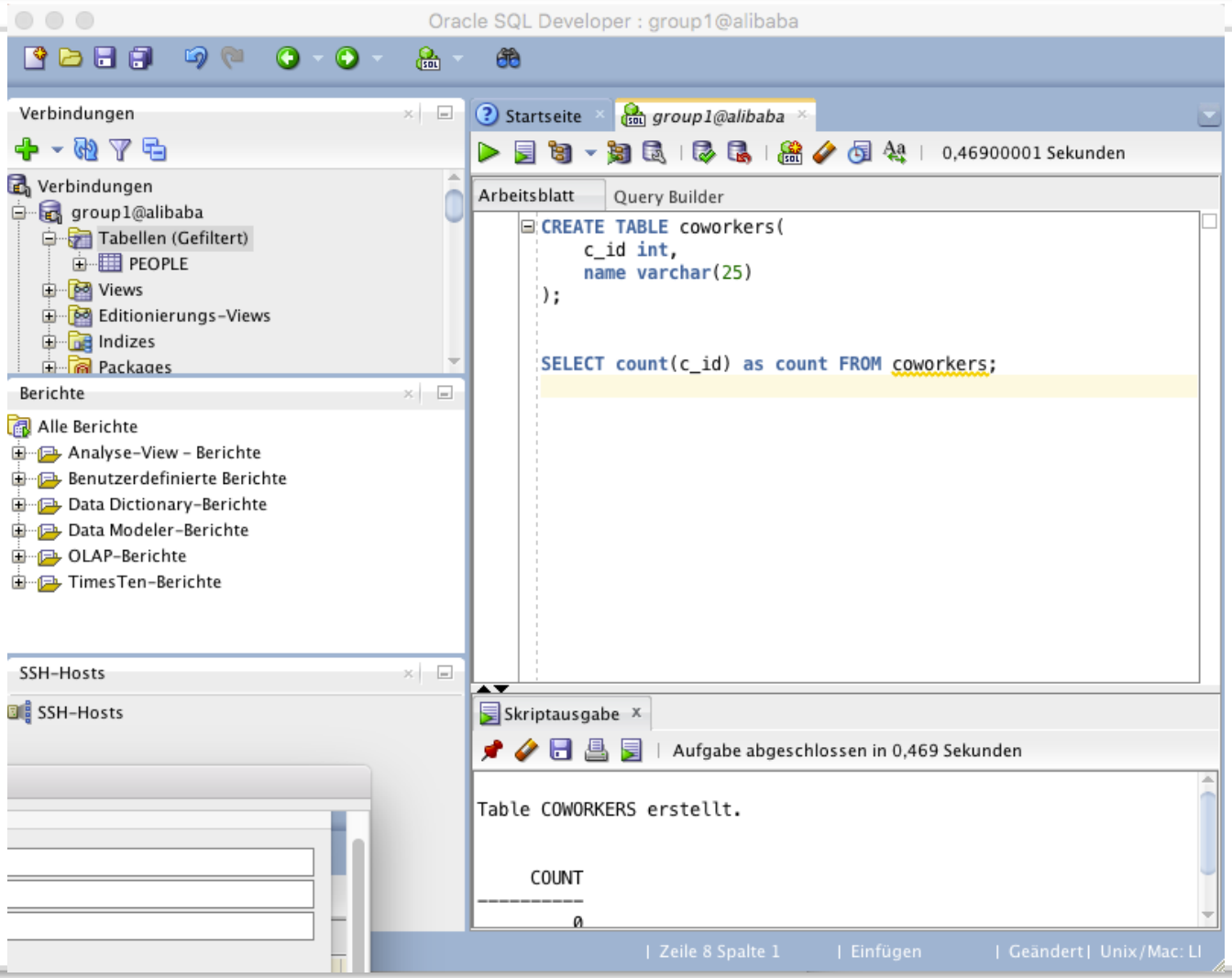
# Oracle SQL Developer



# Verbindungsdaten



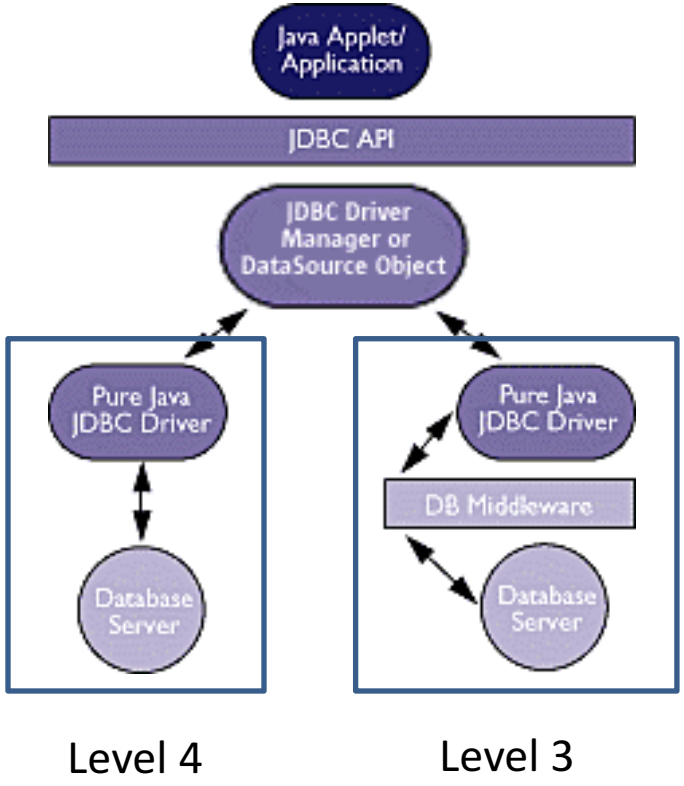
# Skript ausführen (F5)



# JDBC

# Java Database Connectivity (JDBC)- Basics

- Einheitliche Schnittstelle für Zugriff auf RDBMS mittels SQL-basierten Anfragen
- Generische API für eine Vielzahl von Systemen
  - Treiber sind in unterschiedlichen Leveln (1 bis 4) verfügbar
- In Level 3 werden API-Befehle in **generische** DBMS-Befehle übersetzt und an Middleware übertragen
- In Level 4 werden API-Befehle direkt in **native** DBMS-Befehle übersetzt (keine Middleware)



# JDBC - Konzepte

- RDBMS-spezifischer JDBC-Treiber: **`DriverManager`**
- Verbindung zur RDBMS: Interface **`Connection`**
- Anfragen formulieren:
  - **`Statement`** `(einzelne Anfragen),`
  - **`PreparedStatement`** `(mehrere gleichartige Anfragen),`
  - **`CallableStatement`** `(PL/SQL)`
- Ergebnisse auswerten: **`ResultSet`**
- Metadaten auslesen

# JDBC - kompakt

```
try (Connection con = DriverManager.getConnection(
       "jdbc:oracle:thin:@alibaba:1521:orcl",
       "login", "password")) {
     con.setAutoCommit(true);

     try (Statement stms = con.createStatement()) {
       String sql = „...";
       try (ResultSet rs = stmt.executeQuery(sql)) {
         while (rs.next()) {
           String name = rs.getString("name");
           System.out.println(name);
         }
       }
     }
   }
   catch (SQLException e) {
     e.printStackTrace();
   }
```

Verbindung

Statement

ResultSet

# JDBC - Verbindung

- Im Classpath:
  - Oracle JDBC Library **ojdbc7.jar**

- Dann

```
try (Connection con = DriverManager.getConnection(
      "jdbc:oracle:thin:@alibaba:1521:orcl",
      "login", "password")
    ) {
```

# JDBC – Statements

- Sind gewöhnliche SQL-Statements, die JDBC an das RDBMS weiterleitet

- Erzeugt wird ein Statement über das Interface **Connection**
  - In Java >7: mit **try-with-resource**:

```
try (Statement stmt = con.createStatement()) {
    stmt.executeQuery(SQL);
    // kein stmt.close() notwendig
}
```

  - In Java <7:

```
Statement stmt = con.createStatement();
stmt.execute(...);
stmt.close(); // wichtig!
```

# JDBC – Statements

- Ausführung mittels:
  - `ResultSet executeQuery()`
  - `int executeUpdate()`
  - `boolean execute()`

- `Statement.executeQuery` gibt einen Iterator auf die Ergebnisliste mit Interface `ResultSet` zurück

```
ResultSet result = stmt.executeQuery(
    "SELECT c_id, name FROM coworkers ORDER BY c_id"
);
```

- Ergebnisse können elementweise durchlaufen werden über

```
while (result.next()) { ... }
```

- Oder ein einzelnes Tupel abfragen

```
if (result.next()) { ... }
```

# JDBC – Statements – Beispiele

- Returns **true** if the first result is a ResultSet or **false** if it is an update count or there are no results, zB

```
boolean result = stmt.execute(
     "CREATE TABLE coworkers(
            c_id int,
            name varchar(25))”);
```

- Returns either the **row count** for SQL DML statements or **0** for SQL statements that return nothing, zB

```
int result = stmt.executeUpdate(
     "INSERT INTO coworkers VALUES (1, 'Herbert')"
);
```

- Returns a **ResultSet** object that contains the data produced by the given query; never null, zB

```
ResultSet result = stmt.executeQuery(
     "SELECT count(c_id) as count FROM coworkers”
);
```

# JDBC – ResultSet

- Zugriff auf Werte über die Methoden

```
ResultSet.getXXX("<attrib>")
```

- also getString(), getInt(), getObject() …

```
ResultSet result = stmt.executeQuery("SELECT c_id ...")
while(result.next()) {
    int c_id = result.getInt("c_id");
    String name = result.getString("name");
}
```

- Zugriff auf Datentyp ist auch über Metadaten möglich

```
while (result.next()) {
  int numColumns = result.getMetaData().getColumnCount();
  for ( int i = 1 ; i <= numColumns ; i++ ) { // 1 is first index
    System.out.println(result.getObject(i));
  }
}
```

# JDBC – PreparedStatements

- Normale SELECT Statements werden im RDBMS jedes Mal neu geparst, optimiert und kompiliert
- Ineffektiv bei mehrfacher Ausführung mit geänderten Parametern
- Besser:

```
try (PreparedStatement pstmt=con.prepareStatement(
          "INSERT INTO coworkers (c_id, name) VALUES (?,?)“)) {
   {LOOP}
     // prepare tuples:
     pstmt.setInt(1, anInt);
     pstmt.setString(2, aString);
     // add prepared statement
     pstmt.addBatch();
   {ENDLOOP}
   // execute all operations at once
   ps.executeBatch();
 }
```

# JDBC - Quellen

- Tutorial
  http://www.jdbc-tutorial.com/
- Oracle JDBC Driver (Oracle 12c!)
  http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html
- Google ☺

# Wichtige Typen

- Zeichenketten:
  - Fixed-length (padded): CHAR, NCHAR (Unicode)
  - Variable-length: VARCHAR2, NVARCHAR2 (Unicode)
- Zahlen:
  - INTEGER (32 bit), LONGINTEGER (64 bit),
  - Floating-Point:
    DECIMAL(precision, scale), NUMBER(precision, scale)
- Datum:
  - DATE (default format is DD-MON-YY)
- RAW and LONG RAW
  - BLOB, CLOB, NCLOB, LONG (!!)

# PL / SQL

# PL/SQL - kompakt

- Oracle-eigene Programmiersprache, enge Integration mit SQL
- Sämtliche prozeduralen Konzepte sind verfügbar: Funktionen/ Prozeduren, Abfragen/bedingte Ausführung, Schleifen, etc.
- Wird verwendet für SQL Funktionen, Trigger, Stored-Procedures, etc.
- Code wird in der Datenbank, u.U. während der Ausführung einer SQL-Anfrage ausgeführt
  - Keine Benutzerschnittstellen
  - Keine Ausgabe auf Bildschirm etc.
  - Logging ist gar nicht so einfach (Transaktionen)

# PL/SQL - Programme

```
DECLARE
.. /* Variablen-& Typdeklarationen */
BEGIN
.. /* Programmcode, Funktionen .. */
EXCEPTION
.. /* Optional: Exception-Handling */
END;
```

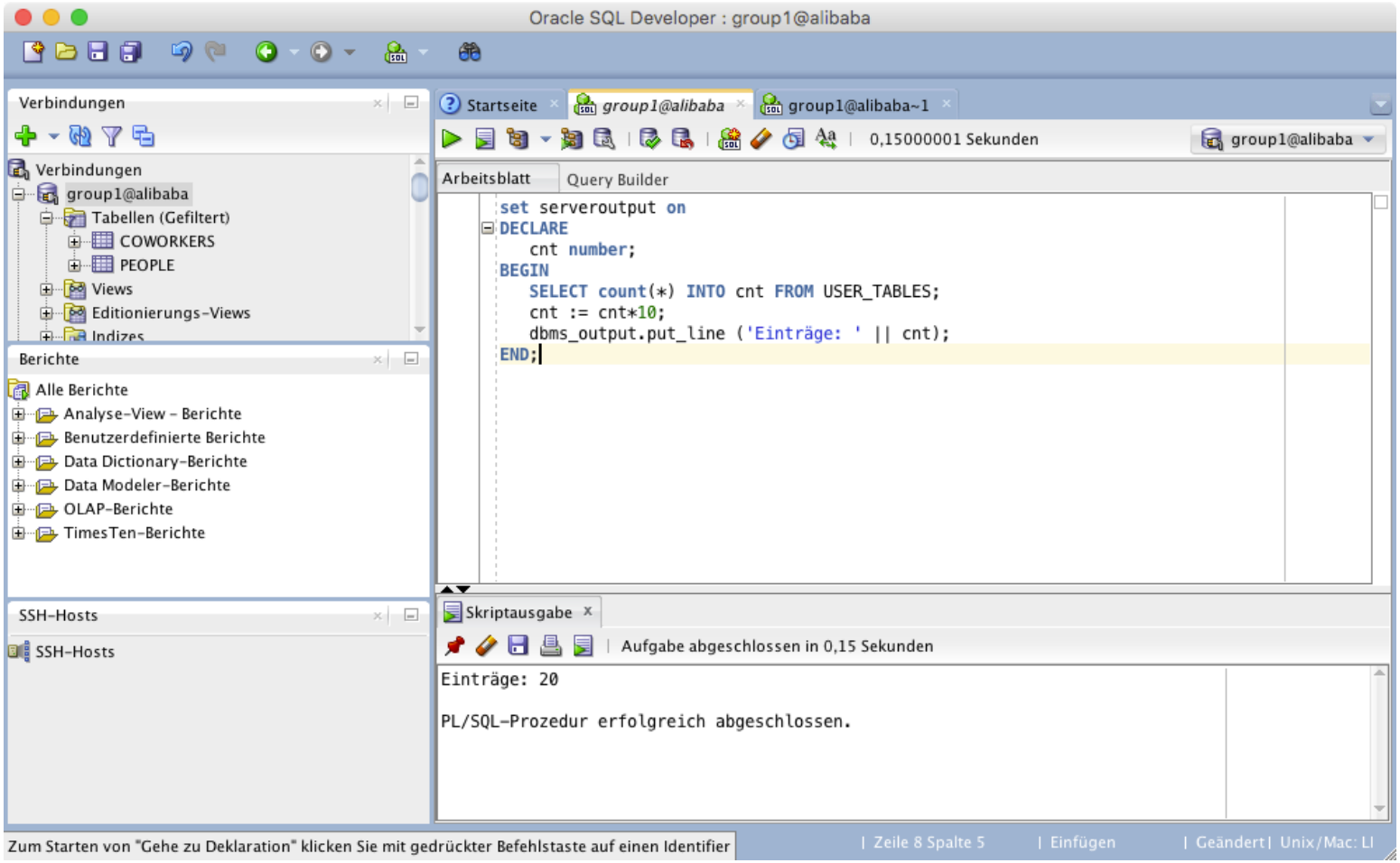
# PL/SQL - Beispiel

- Einfaches Beispiel

```
set serveroutput on
DECLARE
  cnt number := 0;
BEGIN
  SELECT count(*) INTO cnt FROM USER_TABLES;
  cnt := cnt*10;
  dbms_output.put_line ('Einträge: ' || cnt);
END;
```

- Erlaubte SQL-Statements:
  - DML: `select, insert, delete, update`
- Nicht (direkt) erlaubt
  - DDL: `create, drop, alter`, etc.

# PL /SQL - SQL Developer



# PL/SQL - Variablen

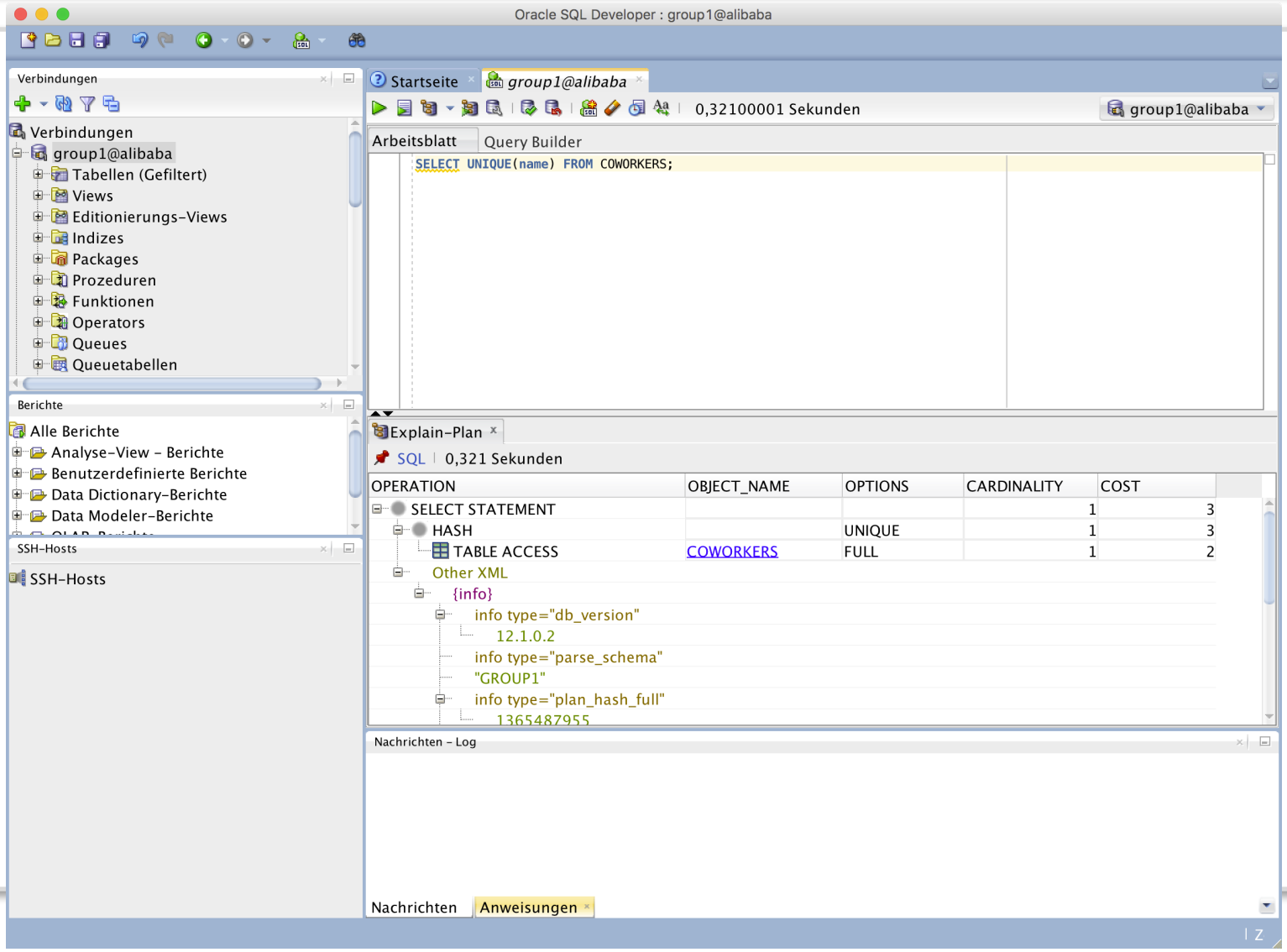
- Alle von Oracle unterstützen Typen + NUMBER, BOOLEAN
- Ermittlung zur Compilezeit möglich

```
DECLARE
   name VARCHAR(20);
   gruppe NUMBER;
   punktezahl table_name.column_name%TYPE;
   …
```

# Kontrollflüsse

- Bedingungen
  - ```
    IF <B> THEN ...
    ELSIF <B2> THEN ...
    ELSE ... END IF;
    ```
  - ```
    CASE selector
       WHEN 'value1' THEN ...;
       ELSE ...;
    END CASE;
    ```
- Schleifen
  - ```
    LOOP
       ...
       EXIT WHEN <B>
       ...
    END LOOP;
    ```
  - ```
    WHILE <B> LOOP
       ...
    END LOOP;
    ```
  - ```
    FOR <V> IN <C1>..<C2> LOOP
       ...
    END LOOP;
    ```
    ```

# PL / SQL String Funktionen

- CONCAT
- || : Konkatenation
- INSTR: Suche nach Teilsequenz
- REPLACE
- SUBSTR
- LENGTH
- TRIM
- UPPER
- LOWER
- …

# PL / SQL - Procedure

- Gibt keinen Wert zurück (analog zu ’void’)

```
CREATE OR REPLACE PROCEDURE procedure_name(
  parameter [IN | OUT | IN OUT] type
) AS
   /* Variablen-& Typdeklarationen */
BEGIN
   /* Programmcode */
END procedure_name;
```

# PL/SQL - Funktionen

- Gibt einen Wert zurück

```
CREATE OR REPLACE FUNCTION function_name(
  parameter_name [IN | OUT | IN OUT] type
) RETURN type
AS
   /* Variablen-& Typdeklarationen */
BEGIN
   /* Programmcode */
  RETURN …;
END function_name;
```

# PL/SQL - Sonstiges

- Befehl ausführen mittels GUI Tool (z.B. Oracle SQL Developer)

```
CREATE OR REPLACE PROECUDURE myproc
AS
… // Variablen
BEGIN
…
end;
```

- Entweder geht es gut, oder es gibt Fehler

# PL / SQL - Konzepte

- Weitere Konzepte:
  - Arrays (VARARRAY), Collections
  - Cursor: Iterator aller Tupel einer Anfrage
  - Trigger: Programme, die automatisch ausgeführt werden, wenn eine Bedingung erfüllt wird bei
    - z.B: “auto_increment”
  - Exceptions
  - …

# PL/SQL – EXPLAIN PLAN

- ```
  EXPLAIN PLAN FOR
      SELECT * FROM user_tables;
  ```
- Parst und optimiert die Query, ohne sie auszuführen
- Erfasst die Abfolge sämtlicher gewählter Operationen
  - Referenzierte Tabellen
  - Verwendete Zugriffsmethoden
  - Join-Methode
  - Geschätzte Kosten
  - …
- Pläne können direkt in SQL Developer angezeigt werden
  ```

# PL/SQL – EXPLAIN PLAN



# Weiterführende Links

- SQL Developer:
  - https://www.youtube.com/watch?v=U-Iigi2oBUo
- Oracle Dokumentation:
  - http://docs.oracle.com/database/122/index.htm
- PL/SQL
  - Quick Guide https://www.tutorialspoint.com/plsql/plsql_quick_guide.htm
  - Oracle https://docs.oracle.com/database/121/LNPLS/

Fragen, Anregungen?