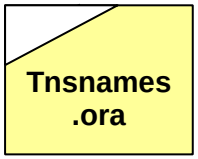
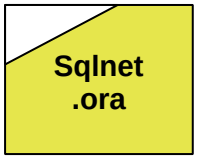
Das Schweizer Taschenmesser für die Oracle-Datenbank

SQL*PLUS

Arbeiten mit SQL*Plus

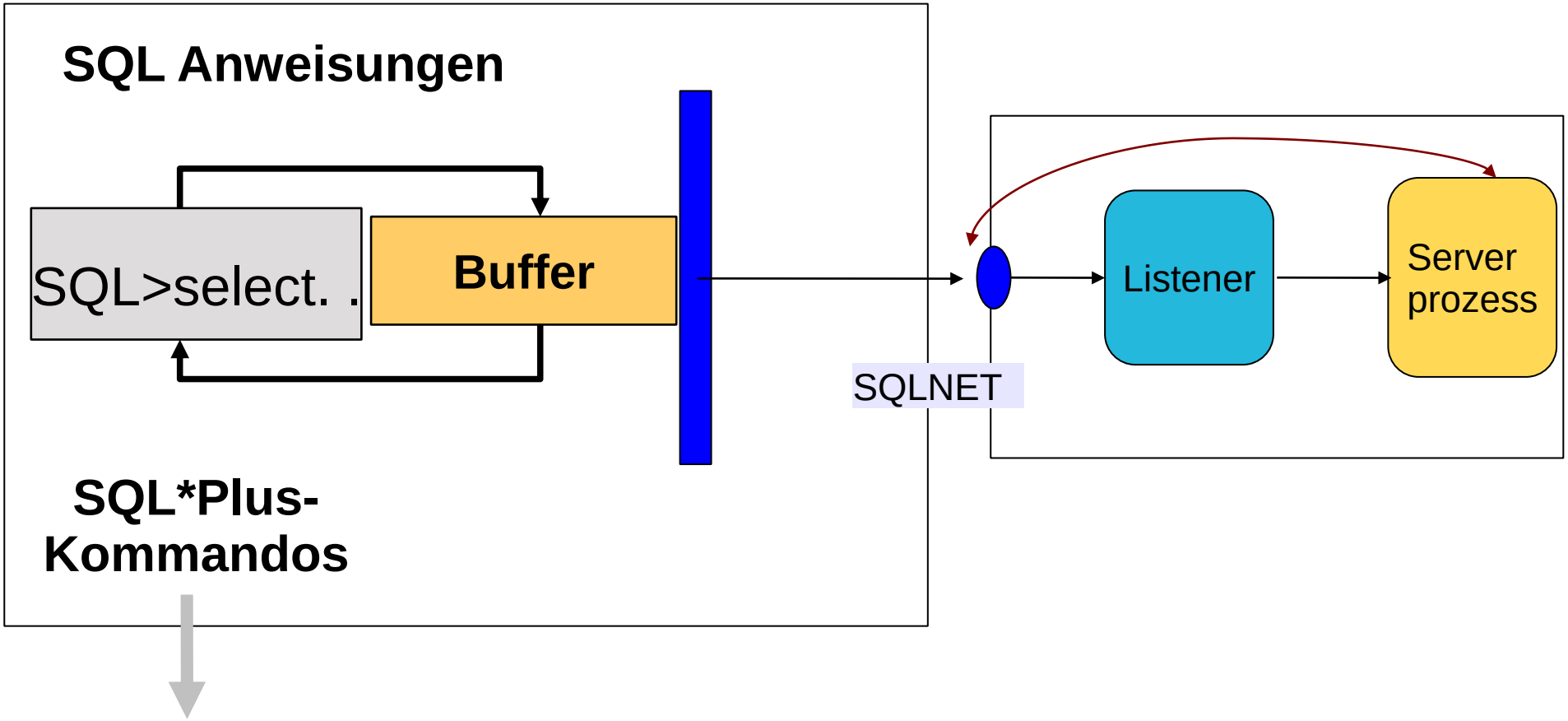
- SQL-Kommando-Interpreter für Oracle DBMS
- Arbeitet mit dem SQLNET-Protokoll
- Voraussetzung:
  - Min. Oracle Client Installation
  - Verbindungsbeschreibung zur Datenbank bekannt und konfiguriert
  - Dateien:
    - Tnsnames.ora
    - sqlnet.ora





SQL- und SQL*Plus-Interaktion

SQL*Plus



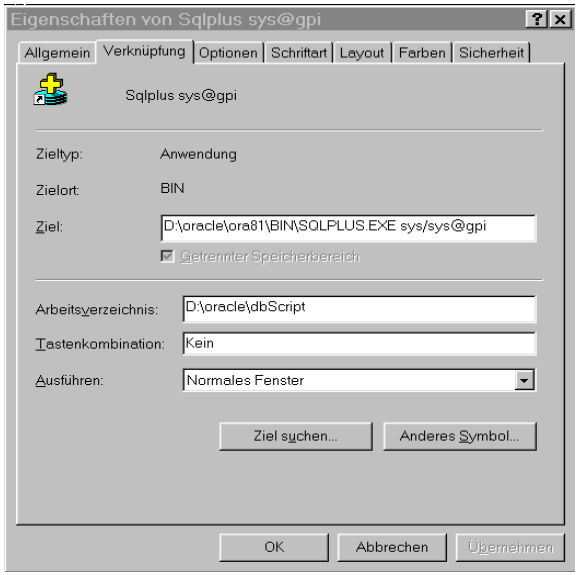
Der Aufruf von SQL*Plus

- Grafische Oberfläche unter Windows:
  - %ORACLE_HOME%/bin/Sqlplusw.exe

- Textoberfläche unter einer Shell
  - %ORACLE_HOME%/bin/Sqlplus.exe
- Empfehlung: Textoberfläche!

Schnelles Arbeiten mit SQL*Plus

- Verknüpfung anlegen auf die Textversion von SQL*Plus auf die am meisten benötigen User sqlplus scott/tiger@oradb1 oder scott@oradb1
- Arbeitsverzeichnis mit Standardskripten setzen
- Copy/Paste in CMD Shell einstellen
- Farben und Font setzen



Quickconnect mit SQL*Plus 10g

- Quickconnect / Easy Connect

  sqlplus system/oracle@'//localhost:1521/GPI'

- URL => 'USER@'//HOST:PORT/SID'

Die Backtaste unter Unix verwenden

- Unter Unix wird nur die Backtaste unterstützt (keine Arrow Keys, kein Kommando-Buffer)
- Setzen der Backtaste über stty

```
sqlplus>stty * ^?^?^?^?v
```

stty erase <Escape_Sequence _Backtaste_shell>

```
~>stty  erase ^?
```

Das Arbeitsverzeichnis setzen SQLPATH

- Mit der Umgebungsvariablen SQLPATH das Arbeitsverzeichnis setzen
- Skripte werden zuerst in diesem Verzeichnis gesucht!

```
~>export SQLPATH=~
```

Verbindung zur Datenbank aufbauen

- Verbindung wird durch SQL*Net aufgebaut
- Angabe des Username/Passwort@TNSalias
  Beispiel:

```
SQL> connect scott/tiger@oradb1
```

- Default TNSALIAS mit der Umgebungsvariablen LOCAL setzen

Ein SQL Statement eingeben

- SQL Statement eingeben
- Zeilenumbruch mit RETURN
- Eingabe abbrechen mit .
- Eingabe ausführen mit ; oder / in neuer Zeile
- Online-Hilfe mit help (rudimentär .-))
- Eingabe löschen mit ESC

Zeilen-Eingaben umbrechen mit -

- Eingabezeilen können mit dem Zeichen - umgebrochen werden (ähnlich \ in der Bash)
- Beispiel:

```
sql>EXECUTE DBMS_OUTPUT.PUT_LINE -
> ('Dies ist eine sehr lange Zeile ..................')

Dies ist eine sehr lange Zeile ......................
```

Datenbank-Parameter anzeigen lassen

- Parameter der DB mit SHOW PARAMETER <Parameter> anzeigen
- Zeigt alle Parameter, deren Name ein Teil des Suchstrings ist
  Beispiel:

```
sql>show parameter dump

NAME                           TYPE    VALUE
------------------------------ ------- ------------------------------
background_dump_dest           string  d:\oracle\admin\gpi\bdump
max_dump_file_size             string  10240
user_dump_dest                 string  d:\oracle\admin\gpi\udump
```

- Vor 8i nur im Servermanager (svrmgrl) möglich

SQL*Plus-Datei-Befehle

- Die Ausgabe und den Buffer von SQL*Plus speichern

• SAV[E] filename [CREATE | REPLACE | APPEND]	Buffer in Datei speichern
• GET filename	Datei in Buffer schreiben
• STA[RT] filename	Datei ausführen
• @ filename	Datei ausführen
• ED[IT] filename	Datei editieren
• SPOOL filename	Ausgaben in Datei spoolen

Die SQL*Plus-Editier-Befehle (1)

- Den Buffer von SQL*Plus bearbeiten

• A[PPEND] text	Text anhängen
• C[HANGE] / old / new	aktuelle Zeile im Buffer ändern aktuelle Zeile durch Nummer setzen!
• C[HANGE] / text /	Text im Buffer löschen
• CL[EAR] BUFF[ER]	Buffer löschen
• DEL	Letzte Zeile löschen
• DEL n	Zeile n löschen
• DEL m n	Zeile m bis n löschen

Die SQL*Plus-Editier-Befehle (2)

- Den Buffer von SQL*Plus bearbeiten

• I[NPUT]	Neue Zeile im Buffer hinzufügen
• I[NPUT] text	Neuen Text am Ende anfügen
• L[IST]	Buffer ausgeben
• L[IST] n	Buffer Zeile n ausgeben
• L[IST] m n	Buffer Zeile m bis n ausgeben
• R[UN]	Buffer ausführen (auch / möglich)
• n	Zeile des Buffers anzeigen
• n text	Zeile n des Buffer Text einfügen
• 0 text	erste Zeile text eingeben

Kommando-Shell aus SQL*PLUS aufrufen

- Der HOST-Befehl
  - Kommando-Interpreter des Betriebssystems aufrufen
- Verlassen mit exit

Steuern von SQL*Plus mit SET

- Die SQL*Plus-Umgebung anpassen

  – SET-Befehl => setzt die Eigenschaften der Umgebung

```
SET system_variable value
```

  – Befehl HELP SET => Online-Hilfe für die möglichen Einstellungen

  – SHOW-Befehl => gesetzte Werte überprüfen

Wichtige SET-Kommando-Variablen

- SET-Anweisungen
  - UNDERLINE {_ | text} Linie unter Spaltenbezeichner
  - COLSEP {_ | text} Trennzeichen der Spalten
  - FEEDBACK {6 | n |OFF | ON} SQL*Plus-Angaben im Ergebnis unterdrücken
  - HEADING {OFF | ON} Spaltenname anzeigen
  - LINESIZE {80 | n} Breite der Ausgabe
  - LONG {80 | n} Anzahl der Zeichen von Long
  - PAGESIZE {24 | n} Anzahl Zeilen pro Seite
  - PAUSE {OFF | ON | text} Pause nach jeder Seite
  - TAB {OFF | ON} Tabulatoren für Leerzeichen

Das SPOOL-Kommando

- SPOOL <dateiname>
  Alle Eingaben und Ergebnisse werden in eine Datei <dateiname> geschrieben
- Mit SPOOL OFF ausschalten
- MIT SPOOL ON wird in Default-Datei on.LST im Arbeitsverzeichnis geschrieben

Die Hilfe von SQL*Plus verwenden

- Wie werden die Hilfetabellen für SQL*Plus installiert (vor 8i)?
  - Zur Verwendung des sql>help Kommandos

```
#!/bin/sh
$ORACLE_HOME/bin/svrmgrl << EOF
connect system/manager
@$ORACLE_HOME/sqlplus/admin/help/helptbl.sql;
exit;
EOF
$ORACLE_HOME/bin/svrmgrl << EOF
connect system/manager
@$ORACLE_HOME/sqlplus/admin/help/helpindx.sql;
exit;
EOF
cd $ORACLE_HOME/sqlplus/admin/help
$ORACLE_HOME/bin/sqlldr userid=system/manager control=plushelp.ctl
$ORACLE_HOME/bin/sqlldr userid=system/manager control=plshelp.ctl
$ORACLE_HOME/bin/sqlldr userid=system/manager control=sqlhelp.ctl
```

  - Wie werden die Hilfetabellen für SQL*Plus installiert (8i/9i)?

```
~>Cd $ORACLE_HOME/sqlplus/admin/help/
~>sqlplus system/manger
system>@helpbld.sql . helpus.sql
```

Ein Datenbank-Objekt beschreiben

- Die DESCRIBE-Anweisung
  Beispiel:

```
sqlpus>desc emp
```

```
Name                  Null?    Type
--------------- -------- --------------
EMPNO                 NOT NULL NUMBER(4)
ENAME                          VARCHAR2(10)
JOB                            VARCHAR2(9)
MGR                            NUMBER(4)
HIREDATE                       DATE
SAL                            NUMBER(7,2)
COMM                           NUMBER(7,2)
DEPTNO                         NUMBER(2)
```

Die login.sql-Datei

- wird aus dem Arbeitsverzeichnis gelesen
- wird immer beim Start gelesen
- Platz für die gewünschten SET-Kommandos oder anderen SQL-Befehlen, die immer beim Start ausgeführt werden sollen

Beispiel LOGIN.SQL

- automatisch bei Start von SQL*Plus ausführen

```
set head off
set termout on

select 'User: '|| user || ' on database ' || global_name,
       '(Terminal='||USERENV('TERMINAL')||
       ', Session-Id='||USERENV('SESSIONID')||')'
from global_name;

REM Set prompt
column _user    noprint new_value user_name
column _sid     noprint new_value sid
column _session noprint new_value session_id

select  user                    "_user"   ,
        global_name             "_sid" ,
        USERENV('SESSIONID') "_session"
from global_name;



set sqlprompt &SESSION_ID:&USER_NAME:&SID>
set head on
```

Spalten mit „noprint“ werden nicht ausgeben

```
set sqlprompt "_user 'on' _date 'at' _connect_identifier >"
```

Ausführungspläne in SQL*Plus anzeigen (1)

- Ausführungspläne erzeugen mit AUTOTRACE
- Vorbereitung
  - Plus-Rollen erzeugen (global bzw. individuell)
    - Rolle

```
sys> @?/sqlplus/admin/plustrce.sql
sys> grant plustrace to scott;
```

    - Persönliche Tabelle für die Execution Pläne anlegen

```
scott> @?/rdbms/admin/utlxplan.sql
```

Ausführungspläne in SQL*Plus anzeigen (2)

- Plan-Tabelle für alle Entwickler anlegen
  - Owner für Plantabelle

```
sys> create user tools
     identified by tool123 default tablespace tools
     temporary tablespace temp;
sys> grant connect to tools;
sys> grant create table to tools;
```

  - Plan-Tabelle als „GLOBAL TEMPORARY TABLE“ mit „ON COMMIT PRESERVE ROWS“ anlegen

```
SQL> connect tools/tool123@oragpi
SQL>CREATE GLOBAL TEMPORARY TABLE PLAN_TABLE  (
       statement_id  varchar2(30), .......
       ,filter_predicates varchar2(4000)) ON COMMIT PRESERVE ROWS ;

SQL>grant all on plan_table to public;
```

Ausführungspläne in SQL*Plus anzeigen (3)

- Plan-Tabelle für alle Entwickler anlegen
  - Public Synonym für diese Tabelle anlegen

```
sys>create public synonym plan_table for tools.plan_table;
```

  - Alle Entwickler können sich nun diese Plan-Tabelle teilen, nur Ihre Daten sind für Sie sichtbar, am Ende der Session werden die Daten automatisch gelöscht.

Die Ausgabe

```
sql>set autotrace on
sql>select LAST_NAME from EMPLOYEES where EMP_ID=93698;
LAST_NAME
------------------------
Abbott

Execution Plan
----------------------------------------------------------
   0      SELECT STATEMENT Optimizer=CHOOSE                (Cost=1 Card=1 Bytes=13)
   1    0   TABLE ACCESS (BY INDEX ROWID) OF 'EMPLOYEES'   (Cost=1 Card=1 Bytes=13)
   2    1     INDEX (UNIQUE SCAN) OF 'EMP_PK' (UNIQUE)     (Cost=1 Card=2)

Statistics
----------------------------------------------------------
          0  recursive calls    --> Internes SQL
          0  db block gets      --> DB Blöcke gelesen (Current Mode z.B. bei Updates)
          3  consistent gets    --> Konsistente Blöcke gelesen
          0  physical reads     --> Blöcke von Platte gelesen
          0  redo size          --> geschriebene Redo Daten
        372  bytes sent via SQL*Net to client          --> Daten vom Server
        424  bytes received via SQL*Net from client    --> Daten zum Server
          4  SQL*Net roundtrips to/from client
          0  sorts (memory)     --> Sortiervorgänge im Speicher
          0  sorts (disk)       --> Sortiervorgänge auf Platte
          1  rows processed
```

estimate the number of rows

estimate the taking bytes

Ausführungsplan mit utlxpls abfragen

- Ausführungsplan abfragen

```
SQL> explain plan for
  2  select * from emp join dept using(deptno);

Explained.

SQL> @?/rdbms/admin/utlxpls

PLAN_TABLE_OUTPUT
--------------------------------------------------------------------------
| Id  | Operation             |  Name       | Rows  | Bytes | Cost  |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT      |             |    14 |   700 |     5 |
|*  1 |  HASH JOIN            |             |    14 |   700 |     5 |
|   2 |   TABLE ACCESS FULL   | DEPT        |     4 |    72 |     2 |
|   3 |   TABLE ACCESS FULL   | EMP         |    14 |   448 |     2 |
--------------------------------------------------------------------------

Predicate Information (identified by operation id):

PLAN_TABLE_OUTPUT
---------------------------------------------------

   1 - access("EMP"."DEPTNO"="DEPT"."DEPTNO")
```

Ein SQL-Skript enthält

- Kommentare
  - -- Dies ist ein Kommentar
  - /* Dies ist ein mehrzeiliger Kommentar */
- SQL-Anweisungen
  - select * from emp where ename like ‘%1‘
- SQL*Plus-Anweisungen
  - Set Echo on
- SQL*PLus-Variablen (variable)
  - variable wert number
- Austauschvariablen mit dem &-Zeichen
- Anweisungen werden mit dem / oder dem ; abgeschlossen

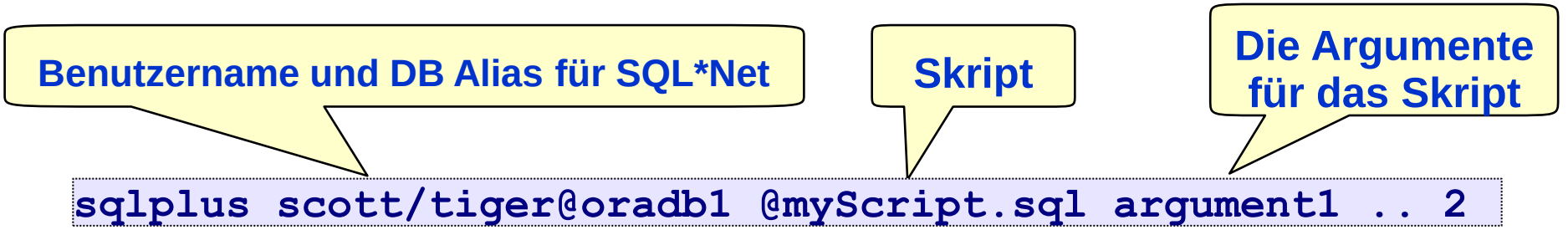
Beispiel:

- Wie oft ist ein Benutzer angemeldet?

```
set ver off
Prompt +------------ Alle aktiven Sessions eines Benutzers ----------+
accept username prompt Benutzername:

Column sid     FORMAT 999   HEADING "SID"
Column serial#       FORMAT 99999 HEADING "Serial#"

column program  FORMAT A35 WRAPPED
Column username FORMAT A9  HEADING "Ora User"
Column machine         FORMAT A9
Column osuser          FORMAT A8   HEADING "OS User"

select s.sid,
       s.serial#,
       s.username,
       s.status,
       s.osuser,
       s.process,
       s.machine,
       s.program
  from v$session s
 where s.username like UPPER('&username');
set ver on
```

Aufruf von SQL*Plus mit einem Skript

- Beispiel



- SQL*Plus wird gestartet und das entsprechende Skript ausgeführt
- Aufruf in SQL*Plus:

```
SQL>@myScript oder sta[rt] myScript
```

  Es wird immer im Arbeitsverzeichnis gesucht
- Mit dem ? kann das aktuelle ORACLE_HOME referenziert werden

```
SQL>@?/rdbms/admin/myScript
```

Aufruf-Argumente beim Start übernehmen

- Ähnlich der %1...n Syntax
- Syntax: &1 --> erster Parameter
  &2 --> zweiter Parameter
  .....
  &n --> n Parameter

  Beispiel:

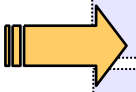
```
select * from emp where empno=&1
```

- Datei sEmp.sql

```
sqlplus scott/tiger@oradb1 @sEmp.sql emp 7777
```

Ersetzungsvariablen

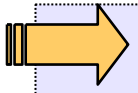
- SQL*Plus-Ersetzungsvariable um temporäre Wert zu speichern.
  - Ampersand (&)
  - Ampersand (&&)
  - DEFINE und ACCEPT
- Übergabe von Variablen an SQL-Anweisungen
- &Wert
  Fordert den Benutzer auf, einen Wert einzugeben
- Achtung !
  Reine Textersetzung! ' bei Literalen beachten!



```
SQL> SELECT   empno, ename, sal, deptno
  2  FROM     emp
  3  WHERE    empno = &employee_num;
Enter value for employee_num: 7369

     EMPNO ENAME             SAL     DEPTNO
---------- ---------- ---------- ----------
      7369 SMITH             800         20
```

Das Kommando SET VERIFY

- Ausgabe der Textersetzung mit on und off einschalten bzw. ausschalten.



```
SQL> SET VERIFY ON
SQL> SELECT    empno, ename, sal, deptno
  2  FROM emp
  3  WHERE     empno = &employee_num;
```

```
Enter value for employee_num: 7369
old   3: WHERE empno = &employee_num
new   3: WHERE empno = 7369

...
```

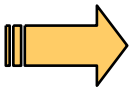
Variablen in SQL*Plus erzeugen

- Nur SQL*PLUS intern, HOST Variablen siehe Kapitel PL/SQL
- Variablen können in SQL*Plus auf zwei Arten erzeugt werden:
  - DEFINE: erzeugt eine Benutzer-Variable vom Typ CHAR
  - ACCEPT: Eingabe eines Benutzers lesen und in einer Variablen speichern
- Variablen sollten keine Leerzeichen enthalten
  - Falls doch, mit einfachen Anführungszeichen versehen

Das ACCEPT Kommando

- Erzeugt einen benutzerdefinierten Prompt für die Eingabe
- Definiert explizit den Datentyp
- Kann für eine Passworteingabe die Eingaben des Benutzers ausblenden

ACCEPT *variable* [*datatype*] [**FORMAT** *format*] [**PROMPT** *text*] [**HIDE**]



```
ACCEPT    dept PROMPT 'Geben Sie den Namen ein: '
SELECT    *
FROM      dept
WHERE     dname = UPPER('&dept')
/
```

```
Geben Sie den Namen ein:         Sales

    DEPTNO DNAME          LOC
---------- -------------- -------------
        30 SALES          CHICAGO
```

Das DEFINE und UNDEFINE Kommando

- Ein Variable bleibt solange definiert, bis:
  - mit UNDEFINE gelöscht wird
  - SQL*Plus beendet wird
- Mit DEFINE überprüfen
- Falls gewünscht, mit DEFINE in der login.sql die Variablen definieren, die stets vorhanden sein sollen.

```
SQL> define Parameter=Wert
SQL> define
DEFINE PARAMETER          = "Wert" (CHAR)
SQL>
```

Variablen anzeigen

Define Tips

- Beispiel:

```
Select * from dept where deptno in ('10','20','30');
```

Drei Treffer !

```
Define dept_string = '10','20','30'
```

```
Select * from dept where deptno in (&dept_string);
```

Ein Treffer !

Lösung:

```
Define dept_string= ''10'',''20'',''30''
```

```
Select * from dept where deptno in (&dept_string);
```

Oder:

```
Define dept_string= '(''10'',''20'',''30'')'
```

```
Select * from dept where deptno in &dept_string;
```

Formatierte Ausgaben mit SQL*Plus

- Formatierungs-Anweisungen
  - COLUMN [column option]
    - Aussehen einer Spalte
  - TTITLE [text | OFF | ON]
    - Setzen eines Titels über der Seite
  - BTITLE [text | OFF | ON]
    - Setzen eines Footer-Texts
  - BREAK [ON report_element]
    - Spaltenumbruch definieren bei geändertem Wert

Die Formatierung von Spalten

- Kontrolliert das Aussehen einer Spalte

COL[UMN] [{***column*|*alias***}] [***option***]]

  - CLE[AR]: löscht zuvor gesetzte Formatierung
  - FOR[MAT] format: setzt die Formatierung über eine Formatangabe
  - HEA[DING] text: setzt die Überschrift der Spalte
    mit | im Text der Überschrift wird ein Umbruch erreicht
  - JUS[TIFY] {align}: richte die Überschrift der Spalte aus (left, center, oder right)

Verwendung des COLUMN-Kommandos

– Überschriften von Spalten erzeugen

```
COLUMN ename HEADING 'Angestellter' FORMAT A20
COLUMN sal JUSTIFY LEFT FORMAT 99,990.00L
COLUMN mgr FORMAT 999999999 NULL 'Kein Chef'
```

– Aktuelle Einstellungen darstellen

```
COLUMN ename
```

– Einstellungen löschen.

```
COLUMN ename CLEAR
```

Das Formatmodell von COLUMN Beispiel „1234“

Format	Beschreibung	Beispiel	Ergebnis
A<n>	Ausgabeweite	-----	1234
9	Nullwerte unterdrücken	9999	1234
0	Nullwerte anzeigen	09999	01234
D und G	Dezimalpunkt und Tausender Punkt	9G999D00	1.234,00

Rechnen in Berichten mit SQL*Plus

- „Compute sum“ und „break on“

```
Sys:gpi>compute sum of bytes on pool
sys:gpi>break on pool skip 1
sys:gpi>select pool
             , name
             , bytes
          from v$sgastat
      order by pool,name;

POOL         NAME                         BYTES
------------ -------------------------- --------
java pool    free memory                 1441792
             memory in use               2560000
************                            --------
sum                                      4001792

large pool   free memory                 7871040
             session heap                 128960
************                            --------
sum                                      8000000

NULL

             db_block_buffers            8192000
             fixed_sga                     70924
             log_buffer                  4999680
************                            --------
sum                                      13262604
```

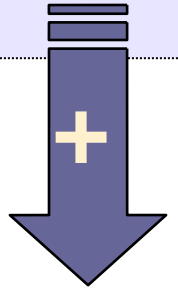
Berechne die Summe auf die Spalte Bytes

pro distinkten Wert der Spalte Pool Umbruch einfügen

SQL verwenden, um SQL zu erzeugen

```
SET HEADING OFF ECHO OFF FEEDBACK
OFF
SET PAGESIZE 0

SPOOL data.sql

SELECT 'INSERT INTO dept VALUES
('||
       deptno||','||
       ''''||dname||''''||',''||
       ''''||loc||''''||');'
FROM   dept;

SPOOL OFF

SET HEADING ON ECHO OFF FEEDBACK ON
SET PAGESIZE 24
```

Data Dictionary



SQL

SQL Skript

```
INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');
INSERT INTO dept VALUES (30,'SALES','CHICAGO');
INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');
INSERT INTO dept VALUES (91,'DEVELOPMENT','AUGSBURG');
```

Ein Skript dynamisch erzeugen

- Löschen aller Tabellen eines Benutzers

```
SQL> SELECT 'DROP TABLE ' || object_name || ';'
  2  FROM    user_objects
  3  WHERE   object_type = 'TABLE';
```

```
DROP TABLE EMP;
DROP TABLE DEPT;
DROP TABLE SALGRADE;
. . .
```

Die Umgebung einstellen

- Mit SET die SQL*Plus-Ausgabe vorbereiten

Die Umgebung setzen

```
SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SPOOL droptab.sql
  SQL STATEMENT

SPOOL OFF



SET ECHO ON
SET FEEDBACK ON
SET PAGESIZE 24
```

Auf die Standard-Werte zurücksetzen

Das komplette Beispiel

- Ein Drop-Skript erzeugen

```
SET ECHO OFF
SET FEEDBACK OFF
SET PAGESIZE 0

SPOOL dropem.sql

SELECT 'DROP TABLE ' || object_name || ';'
FROM    user_objects
WHERE   object_type = 'TABLE';

SPOOL OFF


SET ECHO ON
SET FEEDBACK ON
SET PAGESIZE 24
```

Den Inhalt einer Tabelle in eine Datei

- Ein Import/Export-Beispiel

```
SET HEADING OFF ECHO OFF FEEDBACK OFF
SET PAGESIZE 0

SPOOL data.sql

SELECT 'INSERT INTO dept VALUES ('||
       deptno||','||
       ''''||dname||''''||',''||
       ''''||loc||''''||');'
FROM   dept;

SPOOL OFF

SET HEADING ON ECHO OFF FEEDBACK ON
SET PAGESIZE 24
```

Hochkommatas erzeugen

- Escapesequenz von Hochkommatas

Sourcecode		Ergebnis
`'''X'''`	→	`'X'`
`''''`	→	`'`
`''''||dname||''''`	→	`'AUGSBURG'`

Passwörter in SQL-Skripten

- Die Aufrufe von Oracle Tools über die Kommandozeile enthalten die Passwörter

```
Sqlplus scott/tiger@db1 @import.sql
```

- Über lokale Umgebungsvariablen maskieren!
  Neuen Shell Prozess temporär erzeugen mit gesetzten Variablen (z.B. Fork aus C-Prog.)

```
Sqlplus scott/$SCOTT_PWD@db1 @import.sql
```

- Vorteil: neben der Sicherheit zentrale Verwaltung aller Passwörter

Die Aufrufparameter von Prozessen

- PS Liste enthält die Aufrufparameter eines Prozesses (-x unter Linux)

```
4555 sqlplus  sqlplus scott/tiger@db1 @test.sql
```

- Passwort nicht in der Umgebung verwenden
- Passwort und Befehle über Pipe an sqlplus ‚übergeben‘

```
>sqlplus << EOF scott/tiger@gpi
> ... Befehle
> EOF
```

BackTick! Richtung beachten!

```
>sqlplus `echo scott/tiger@db1`
```

- Oder führen leider nicht zum gewünschten Erfolg
- Evtl. OPS$ Account benutzen

Zusammenfassung

- SQL*Plus ist DAS Werkzeug für die Oracle-Datenbank
  - Alle administrativen Tasks können mit SQL*Plus durchgeführt werden
  - Für die Entwicklung unentbehrlich
  - Die wichtigsten Umgebungsvariablen
    - TNS_ADMIN => wo finde ich meine SQL*Net-Dateien
    - LOCAL => welchen TNS ALIAS möchte ich als Default
    - SQLPATH => wo sind meine SQL-Skripte
  - Arbeitsverzeichnis sauber setzen
  - mit „login.sql“ arbeiten, um das Verwechseln von Datenbanken zu vermeiden
  - DOS Oberfläche empfehlenswerter als das Grafische Tool


Fragen
&
Antworten