TSBE Bern
Höhere Fachschule

ORACLE®
DATABASE

Agenda

TSBE Bern
Höhere Fachschule

1. Datenbankobjekte
2. Data Dictionary
3. Transaktionen

Datenbankobjekte

TSBE Bern
Höhere Fachschule

- Welche Datenbankobjekte gibt es?

```
SQL> select distinct object_type
  2  from dba_objects
  3  order by 1;

OBJECT_TYPE
-------------------
CLUSTER
CONSUMER GROUP
CONTEXT
DESTINATION
DIRECTORY
EDITION
EVALUATION CONTEXT
FUNCTION
INDEX
INDEX PARTITION
JOB
JOB CLASS
LIBRARY
LOB
LOB PARTITION
OPERATOR
PACKAGE
PACKAGE BODY
PROCEDURE
PROGRAM
QUEUE
RESOURCE PLAN
RULE
RULE SET
SCHEDULE
SCHEDULER GROUP
SEQUENCE
SYNONYM
TABLE
TABLE PARTITION
TABLE SUBPARTITION
TRIGGER
TYPE
TYPE BODY
UNDEFINED
VIEW
WINDOW

37 rows selected.
```

Table 1/3

TSBE Bern
Höhere Fachschule

- In den Tabellen werden die effektiven Daten gespeichert
- Data Dictionary View:

```
DBA_TABLES
```

- Tabellen enthalten mehrere Attribute mit unterschiedlichen Datentypen
- Gängige Datentypen:
  - NUMBER
  - VARCHAR2
  - DATE
  - TIMESTAMP
  - CLOB
  - BLOB

Table 2/3

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create table testtab (
id number not NULL,
firstname varchar2(35) not NULL,
lastname varchar2(35) default 'Meier' not NULL)
[tablespace [tablespace name]];
```

- Drop Statement

```
drop table testtab;
```

Table 3/3

TSBE Bern
Höhere Fachschule

- → ORACLE spezifisch:
- Spezialfall Tabelle DUAL:
- DUAL ist eine 1-Row Systemtabelle mit einem Attribut “Dummy”, des Datentyp’s varchar2(1) mit dem Wert “X”
- Abfragen gegen die Datenbank müssen immer gegen eine Tabelle gemacht werden
- DUAL wird verwendet, wenn die Abfrage eigentlich nichts mit Daten in einer Tabelle zu Tun hat

```
SQL> select 1+1 from dual;
SQL> select sysdate from dual;
SQL> select systimestamp from dual;
```

Lob

TSBE Bern
Höhere Fachschule

- Ein LOB ist ein “Large OBject”
- Ein LOB ist ein Datentyp, welcher speziell behandelt wird
- Es gibt zwei verschiedene Arten von LOB’s:
  - BLOB: Binary Large Object
  - CLOB: Character Large Object
- LOB’s werden in eigenen Segmenten abgespeichert, darum werden sie von ORACLE als eigenes DB Objekt ausgewiesen
  - Dies läuft transparent im Hintergrund
- Data Dictionary View:

```
DBA_LOBS
```

Index 1/2

TSBE Bern
Höhere Fachschule

- Indexe werden benötigt um schneller auf die Daten zuzugreifen
- Eine Tabelle kann mehrere Indexe haben
- Indexe können mehrere Attribute beinhalten
- Per Default werden Indexe als B-Tree Bäume angelegt
- Data Dictionary View:

```
DBA_INDEXES
```

Index 2/2

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create index testtab_idx
on testtab (
lastname, firstname
)
[tablespace [tablespace name]];
```

- Drop Statement

```
drop index testtab_idx;
```

- Wird die Tabelle gelöscht, werden die Indexe auch gelöscht

Sequence 1/3

TSBE Bern
Höhere Fachschule

- Sequenzen werden benötigt um Werte sequenziell hochzuzählen
- Sie werden benutzt um Primary Key Werte abzufüllen
- Es kann definiert werden wie viele Werte im Memory gecached werden sollen
  - Die Werte im Memory gehen beim Neustarten der DB verloren
- → Sequenzen generieren eindeutige Nummern
- → Die Nummern sind nicht lückenlos
- Data Dictionary View:

```
DBA_SEQUENCES
```

Sequence 2/3

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create sequence testtab_seq;
```

- Drop Statement

```
drop sequence testtab_seq;
```

Sequence 3/3

TSBE Bern
Höhere Fachschule

- Abfrage des aktuellen Wert's einer Sequenz:

```
select testtab_seq.currval from dual;
```

- Abfrage des nächsten Werts einer Sequenz

```
select testtab_seq.nextval from dual;
```

- Nextval erhöht die Sequenz um einen bestimmten Wert (um 1 in diesem Beispiel)
- Nextval wird für Inserts in Primary Key Columns verwendet

```
insert into testtab values (
testtab_seq.nextval,
'Fritz',
'Meier'
);
```

Synonym 1/3

TSBE Bern
Höhere Fachschule

- Synonyme sind Aliasse für DB Objekte
- Die meisten DB Objekte können via Synonym angesprochen werden
  - Tabellen
  - Views
  - Sequenzen
  - DB Links
  - Prozeduren
  - etc.
- Es gibt 2 verschiedene Arten von Synonymen:
  - PUBLIC synonyme
    - Alle DB User sehen dieses Synonym
    - Sollte nicht verwendet werden!
  - Persönliche Synonyme
    - Gehören genau einem User
- Data Dictionary View:

```
DBA_SYNONYMS
```

Synonym 2/3

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create or replace synonym testtab
for user1.testtab;
```

- Drop Statement

```
drop synonym testtab;
```

Synonym 3/3

TSBE Bern
Höhere Fachschule

- Beispiel:
  - user1 hat eine Tabelle address
  - user2 muss auf die Daten von address zugreifen
    - → der Zugriff muss vollqualifiziert gemacht werden

```
SQL> select street from user1.address;
```

  - Dies kann uU zu Problemen führen wenn z.B. die Tabelle address neu einem anderen User gehört
  - Mit einem Synonym kann auf die Tabelle zugegriffen werden wie es die eigene wäre

```
SQL> select street from address;
```

View 1/2

TSBE Bern
Höhere Fachschule

- Views sind verschiedene “Sichten” auf die Daten
- Views sind gespeicherte select Statements
- ORACLE verwendet im Data Dictionary sehr viele Views
  - Im DD werden kaum Abfragen auf die Basis Tabellen gemacht
- Data Dictionary View

```
DBA_VIEWS
```

View 2/2

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create or replace view emp_dep as
select e.first_name, e.last_name, d.department_name
from employees e, departments d
where e.department_id = d.department_id;
```

- Drop Statement

```
drop view emp_dep;
```

Directories 1/2

TSBE Bern
Höhere Fachschule

- Ein Directory ist ein Verweis auf ein physisches Verzeichnis auf dem DB Server
- Data Dictionary View:

```
DBA_DIRECTORIES
```

- Will man aus ORACLE ein File schreiben, muss das über ein Directory gemacht werden
- Wird interessant bei Backup / Recovery bzw. DataPump

Directories 2/2

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create or replace directory dp_export
as '/u00/app/oracle/admin/DAH11203/dpdump';
```

- Drop Statement

```
drop directory dp_export;
```

DB Links 1/2

TSBE Bern
Höhere Fachschule

- Ein Datenbank Link ist ein definierter Zugriff auf eine andere Datenbank
- Mit DB Links ist es möglich ein DB Objekt auf einer anderen Datenbank anzusprechen
- Data Dictionary View:

```
DBA_DB_LINKS
```

DB Links 2/2

TSBE Bern
Höhere Fachschule

- Create Statement:

```
create database link xe112.tsbe.ch
connect to hr identified by hr
using 'xe112.tsbe.ch';
```

- Drop Statement

```
drop database link xe112.tsbe.ch
```

- Zugriff auf Remote Objekt

```
SQL> select count(*) from employees@xe112.tsbe.ch;

  COUNT(*)
----------
       107
```

Agenda

TSBE Bern
Höhere Fachschule

1. Datenbankobjekte
2. Data Dictionary
3. Transaktionen

ORACLE Data Dictionary

TSBE Bern
Höhere Fachschule

- ORACLE speichert interne Informationen zu DB Objekten in internen Tabellen
- Die Systemtabellen gehören hauptsächlich dem Schema SYS
- Die Systemtabellen werden “Data Dictionary” genannt
- Der Zugriff auf das Data Dictionary meist via Views
- Das Data Dictionary darf grundsätzlich nicht manuell angepasst werden
- Andere RDBMS Systeme verfügen über ähnliche Konstrukte
  - Unteschiedliche Möglichkeiten:
    - “System Datenbank”
    - Management DB
    - Systemtabellen

Data Dictionary Views

TSBE Bern
Höhere Fachschule

- Es gibt zwei wichtige Gruppen von ORACLE DD Views
  - DBA_* Views
  - V$* Views
- Es gibt einige DBA bzw. V$ Views ☺

```
SQL> select count(*) from dba_views where view_name like 'DBA%';

  COUNT(*)
----------
       706

SQL> select count(*) from dba_views where view_name like 'V_$%';

  COUNT(*)
----------
       619
```

- Man muss nicht alle kennen, man muss wissen wie sie gefunden werden können ☺

DBA Views 1/2

TSBE Bern
Höhere Fachschule

- In den DBA Views stehen Informationen zu Schema Objekten oder generell zu Datenbank Objekten
- Die DBA Views stehen in unterschiedlichen Ausprägungen für unterschiedliche Privilegien zur Verfügung:
  - DBA_* Views
    - Sämtliche Datenbank Objekte
    - Zugriff braucht Privilegien auf einzelne DBA Views oder auf das ganze DD
  - ALL_* Views
    - Alle Objekte, auf welchen ich Berechtigungen habe
    - Zugriff braucht keine speziellen Privilegien
  - USER_* Views
    - Alle meine Datenbank Objekte
      - Wie wenn “where owner = [current User]“ dabei wäre
    - Haben keine Spalte “owner”
    - Zugriff braucht keine speziellen Privilegien

DBA Views 2/2

TSBE Bern
Höhere Fachschule

- Beispiele:
  - DBA_OBJECTS
    - Alle DB Objekte der Datenbank
  - USER_TABLES
    - Alle meine Tabellen
  - ALL_VIEWS
    - Alle Views, auf welche ich zugreifen darf
- Weitere Beispiele folgen später

V$ Views 1/2

TSBE Bern
Höhere Fachschule

- V$ Views werden auch “Performance Views” genannt
- Der Inhalt der V$ Views ist nicht persistent
- Die V$ Views (bzw. die darunterliegenden Tabellen) werden bei jedem Neustart der DB neu geladen
- Die V$ Views stehen oft in zwei unterschiedlichen Ausprägungen zur Verfügung:
  - V$* Views
    - V$ Views zu der Instanz, auf welche ich eingeloggt bin
  - GV$* Views
    - V$ Views mit Inhalten zu allen Instanzen im System
    - Zusätzliches Attribut “INST_ID”

V$ Views 2/2

TSBE Bern
Höhere Fachschule

- Beispiele:
  - V$INSTANCE
    - Informationen zur Instanz
    - Z.B. startup_time
  - V$DATABASE
    - Informationen zur Datenbank
    - Z.B. log_mode, db_name, flashback_on etc
- Weitere Beispiele folgen später

Arbeiten mit DD Views 1/4

TSBE Bern
Höhere Fachschule

- 1. Anlaufstelle: View DBA_VIEWS
- Sucht man z.B. Informationen zu Datenfiles, könnte sowas weiterhelfen:

```
SQL> select view_name from dba_views where view_name like
'DBA%FILE%' order by 1;

VIEW_NAME
------------------------------
DBA_DATA_FILES
DBA_EXP_FILES
DBA_FILE_GROUPS
…
…
18 rows selected.
```

Arbeiten mit DD Views 2/4

TSBE Bern
Höhere Fachschule

- … was ist DBA_DATA_FILES genau?
- View “dict” kann weiterhelfen

```
SQL> select comments from dict where table_name =
'DBA_DATA_FILES';

COMMENTS
--------------------------------------------------------------------------------
Information about database data files
```

Arbeiten mit DD Views 3/4

TSBE Bern
Höhere Fachschule

- … welche Informationen stehen in DBA_DATA_FILES?

```
SQL> desc dba_data_files
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 FILE_NAME                                          VARCHAR2(513)
 FILE_ID                                            NUMBER
 TABLESPACE_NAME                                    VARCHAR2(30)
 BYTES                                              NUMBER
 BLOCKS                                             NUMBER
 STATUS                                             VARCHAR2(9)
 RELATIVE_FNO                                       NUMBER
 AUTOEXTENSIBLE                                     VARCHAR2(3)
 MAXBYTES                                           NUMBER
 MAXBLOCKS                                          NUMBER
 INCREMENT_BY                                       NUMBER
 USER_BYTES                                         NUMBER
 USER_BLOCKS                                        NUMBER
 ONLINE_STATUS                                      VARCHAR2(7)
```

Arbeiten mit DD Views 4/4

TSBE Bern
Höhere Fachschule

- … und hier die Informationen:

```
SQL> set lines 200
SQL> col file_name format a45
SQL> col tablespace_name format a13
SQL> col mb format 9999
SQL> select file_name, tablespace_name, bytes/1024/1024 MB from
dba_data_files;

FILE_NAME                                     TABLESPACE_NA     MB
--------------------------------------------- ------------- -----
/u02/oradata/XE112/system01XE112.dbf          SYSTEM          250
/u02/oradata/XE112/sysaux01XE112.dbf          SYSAUX          100
/u02/oradata/XE112/undots01XE112.dbf          UNDOTS          200
/u02/oradata/XE112/users01XE112.dbf           USERS            50
/u02/oradata/XE112/tools01XE112.dbf           TOOLS            40
/u02/oradata/XE112/hr01XE112.dbf              HR               50
/u02/oradata/XE112/webshop_data01XE112.dbf    WEBSHOP_DATA     50

7 rows selected.
```

Agenda

TSBE Bern
Höhere Fachschule

1. Datenbankobjekte
2. Data Dictionary
3. Transaktionen

Transaktionen 1/3

TSBE Bern
Höhere Fachschule

- Schreiboperationen (DML Statements) in Datenbanken werden Transaktionen genannt
- Transaktionen müssen manuell gesteuert werden
- Man muss sich gut überlegen wann eine Transaktion beginnt und wann sie endet!
- Transaktionen werden mit dem ersten schreibenden DML Statement gestartet
- Transaktionen werden mit folgenden Befehlen beendet:
  - commit;
    - Ein commit schreibt die Transaktion in die Datenbank
  - rollback;
    - Ein rollback macht die Transaktion rückgängig
- Warum sind Transaktionen wichtig
  - Oft haben mehrere DML Operationen einen direkten Zusammenhang
  - Ist Statement 1 nicht erfolgreich, darf Statement 2 nicht ausgeführt werden

Transaktionen 2/3

TSBE Bern
Höhere Fachschule

- Beispiel:
  - Ein Kunde einer Bank überweist Fr. 100.-- an einen anderen Kunden
  - In der Datenbank könnte diese Transaktion in Etwa so aussehen:

```
update konto
set kontostand = kontostand - 100
where kunde = 'KUNDE1';

update konto
set kontostand = kontostand + 100
where kunde = 'KUNDE2';
```

  - Die Transaktion darf nur geschrieben werden, wenn beide Statements erfolgreich waren.

Transaktionen 3/3

TSBE Bern
Höhere Fachschule

- Die meisten RDBMS Systeme unterstützen Transaktionen
- Bei einigen RDBMS Systemen muss eine Transaktion explizit gestartet werden
- Andere Datenbanken unterstützen keine Transaktionen z.B. MySQL mit MyISAM Storage Engine
  - Werden in MySQL Transaktionen benötigt, muss die Storage Engine InnoDB verwendet werden
  - Vorteil von MyISAM: Hohe Performance weil keine Redo- und keine Undo Informationen geschrieben werden.
- Solange kein Commit ausgeführt wurde, sind die veränderten Werte nur in der Session ersichtlich, in welcher die Daten verändert wurden
  - Andere Sessions sehen den Stand der Daten von vor der Änderung

Autocommit

TSBE Bern
Höhere Fachschule

- Einige RDBMS Systeme unterstützen Autocommit
- Ist Autocommit eingeschalten, wird jedes Statement automatisch commited
- Vorsicht mit der Transaktionssteuerung ☺

Implizites Commit

TSBE Bern
Höhere Fachschule

- Bei verschiedenen Operationen wird ein implizites Commit ausgelöst
- Ein implizites Commit ist ein Commit, welches intern ausgeführt wird ohne dass ein Commit manuell ausgeführt wurde
- Implizite Commits müssen in der Transaktionssteuerung berücksichtigt werden
- In folgenden Fällen wird ein implizites Commit ausgelöst:
  - Ausführen eines DDL Statements
  - Schliessen von SQL*Plus ohne ein Commit abzusetzen

Übungen

TSBE Bern
Höhere Fachschule

- Damit man ein «Gspüri» für Transaktionen bekommt, ist es nützlich ein paar Übungen zu machen:
- Die Lösungen zu den Übungen sind am Ende der Präsentation
  - Erst selber versuchen, nicht einfach die Lösungen abtippen ☺
- Bei jeder Übung die Erkenntnisse notieren

Übung 1

TSBE Bern
Höhere Fachschule

Geänderte Daten sind erst nach dem Commit in anderen Sessions sichtbar

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. Eine zweite Session auf die DB öffnen und den Inhalt der Tabelle mit einem select Statement anzeigen lassen
5. In der ersten Session ein Commit absetzen
6. In beiden Sessions den Inhalt der Tabelle mit einem select Statement anzeigen lassen

Übung 2

TSBE Bern
Höhere Fachschule

Implizites Commit beim Schliessen von SQL*Plus

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. SQL*Plus schliessen ohne ein Commit abzusetzen
5. Inhalt der Tabelle mit select Statement anzeigen lassen

Übung 3

TSBE Bern
Höhere Fachschule

Implizites Commit vor dem Ausführen eines DDL Statements

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. Eine zweite, beliebige Tabelle erstellen
5. Irgendwas inserten (**kein** commit machen!)
6. Inhalt der Tabelle mit select Statement anzeigen lassen
7. Transaktion rückgängig machen (Rollback ausführen)
8. Inhalt beider Tabellen mit select Statement anzeigen lassen

Übung 4

TSBE Bern
Höhere Fachschule

Rollback veranschaulichen

1. Eine beliebige Testtabelle erstellen
2. Eine zweite, beliebige Tabelle erstellen
3. Irgendwas in Tabelle 1 inserten
4. Irgendwas in Tabelle 2 inserten
5. Inhalt beider Tabellen abfragen
6. Transaktion schreiben (commit;)
7. Inhalt von Tabelle 1 ändern
8. Inhalt von Tabelle 2 ändern
9. Inhalt beider Tabellen abfragen
10. Transaktion rückgängig machen (rollback;)
11. Inhalt beider Tabellen abfragen

Übung 5

TSBE Bern
Höhere Fachschule

Truncate ist ein DDL Statement. Korrekten Transaktionsverlauf überprüfen.

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten und Transaktion beenden
3. Inhalt der Tabelle anzeigen
4. Inhalt der Tabelle löschen (mit delete)
5. Inhalt der Tabelle anzeigen
6. Transaktion rückgängig machen (rollback;)
7. Inhalt der Tabelle anzeigen
8. Tabelle mit truncate leeren (truncate table test)
9. Inhalt der Tabelle anzeigen
10. Transaktion rückgängig machen (rollback;)
11. Inhalt der Tabelle anzeigen

Lösung Übung 1

TSBE Bern
Höhere Fachschule

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. Eine zweite Session auf die DB öffnen und den Inhalt der Tabelle mit einem select Statement anzeigen lassen
5. In der ersten Session ein Commit absetzen
6. In beiden Sessions den Inhalt der Tabelle mit einem select Statement anzeigen lassen

```
Session 1:
create table test (a number);
insert into test values (100);
select * from test;

Session 2:
select * from test;

Session 1:
Commit;
select * from test;

Session 2:
select * from test;
```

Lösung Übung 2

TSBE Bern
Höhere Fachschule

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. SQL*Plus schliessen ohne ein Commit abzusetzen
5. Inhalt der Tabelle mit select Statement anzeigen lassen

```
create table test (a number);
insert into test values (100);
select * from test;
exit;

select * from test;
```

Lösung Übung 3

TSBE Bern
Höhere Fachschule

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten (**kein** commit machen!)
3. Inhalt der Tabelle mit select Statement anzeigen lassen
4. Eine zweite, beliebige Tabelle erstellen
5. Irgendwas inserten (**kein** commit machen!)
6. Inhalt der Tabelle mit select Statement anzeigen lassen
7. Transaktion rückgängig machen (Rollback ausführen)
8. Inhalt beider Tabellen mit select Statement anzeigen lassen

```
create table test (a number);
insert into test values (100);
select * from test;
create table test2 (a number);
insert into test2 values (200);
select * from test2;
rollback;
select * from test;
Select * from test2;
```

Lösung Übung 4

TSBE Bern
Höhere Fachschule

1. Eine beliebige Testtabelle erstellen
2. Eine zweite, beliebige Tabelle erstellen
3. Irgendwas in Tabelle 1 inserten
4. Irgendwas in Tabelle 2 inserten
5. Inhalt beider Tabellen abfragen
6. Transaktion schreiben (commit;)
7. Inhalt von Tabelle 1 ändern
8. Inhalt von Tabelle 2 ändern
9. Inhalt beider Tabellen abfragen
10. Transaktion rückgängig machen (rollback;)
11. Inhalt beider Tabellen abfragen

```
create table test1 (a number);
create table test2 (a number);
insert into test1 values (100);
insert into test2 values (200);
select a.a, b.a from test1 a, test2 b;
commit;
update test1 set a = 1000 where a = 100;
update test2 set a = a * 10 where a = 200;
select a.a, b.a from test1 a, test2 b;
rollback;
select a.a, b.a from test1 a, test2 b;
```

Lösung Übung 5

TSBE Bern
Höhere Fachschule

1. Eine beliebige Testtabelle erstellen
2. Irgendwas inserten und Ttansaktion beenden
3. Inhalt der Tabelle anzeigen
4. Inhalt der Tabelle löschen (mit delete)
5. Inhalt der Tabelle anzeigen
6. Transaktion rückgängig machen (rollback;)
7. Inhalt der Tabelle anzeigen
8. Tabelle mit truncate leeren (truncate table test)
9. Inhalt der Tabelle anzeigen
10. Transaktion rückgängig machen (rollback;)
11. Inhalt der Tabelle anzeigen

```
create table test (a number);
insert into test values (100);
commit;
select * from test;
delete from test;
select * from test;
rollback;
select * from test;
truncate table test;
select * from test;
rollback;
select * from test;
```

Fragen?

TSBE Bern
Höhere Fachschule