Oracle SQL – Data Definition Language

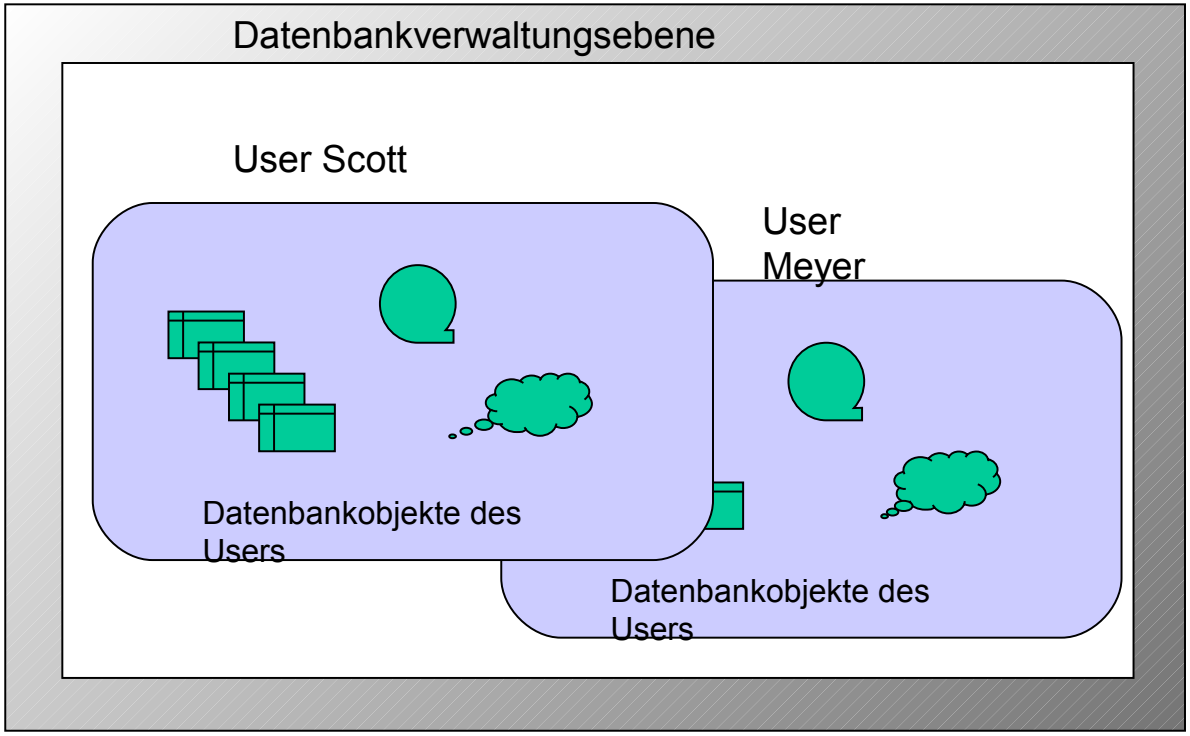
Stephan Karrer

Datenbankobjekte unter Oracle

- Möglichst alle durch Oracle verwalteten Einheiten werden als Datenbankobjekte präsentiert
- Es gibt somit eine Vielzahl von Objekten die mittels der DDL-Anweisungen (CREATE, ALTER, DROP) erzeugt, verändert und gelöscht werden können:
  - Tabellen
  - Views
  - Sequenzen
  - Indizes
  - Schemata
  - Tablespaces
  - ....

Schemata fassen Datenbankobjekte zu logischen Gruppen zusammen

ORACLE Datenbank

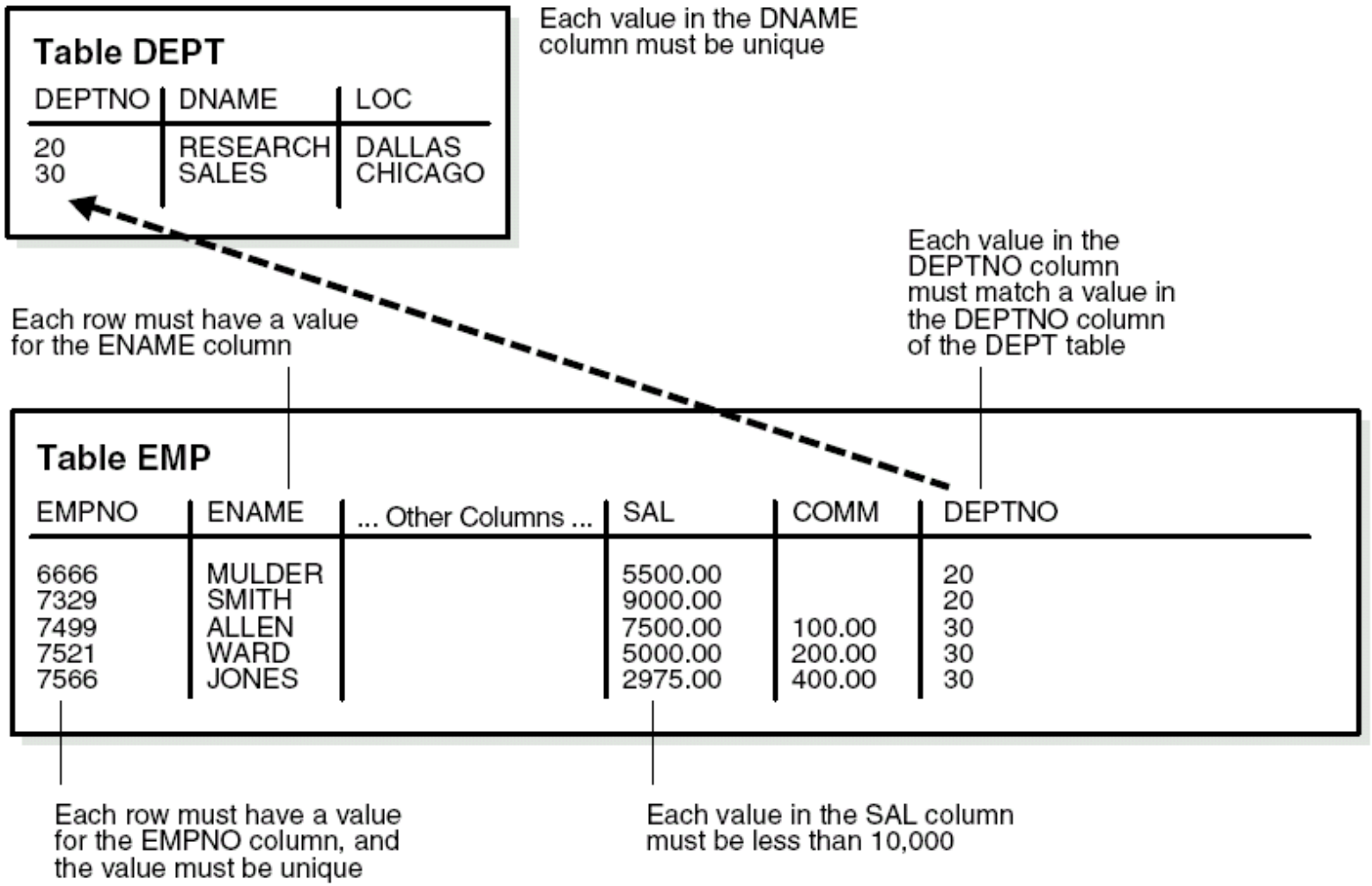


Erzeugen von Tabellen mit der CREATE-Anweisung

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ schema. ]table
   [ relational_properties ]
   [ ON COMMIT { DELETE | PRESERVE } ROWS ]
   [ physical_properties ]
   [ table_properties ] ;
```

```
CREATE TABLE departments_demo
   ( department_id NUMBER(4),
     department_name VARCHAR2(30)
         CONSTRAINT dept_name_nn NOT NULL ,
     manager_id NUMBER(6),
     location_id NUMBER(4),
     dn VARCHAR2(300)
    ) ;
```

Integritätsbedingungen



Constraints: Bedingungen auf Tabellen- bzw. Spalten-Ebene

Folgende Constraints sind in Oracle zulässig:
(in Klammern ist der Constraint-Typ aus der View des Data Dictionary angegeben)

- NOT NULL (C): erlaubt keine NULL-Werte
- UNIQUE (U): erlaubt nur eindeutige oder NULL-Werte
- PRIMARY KEY (P): Kombination aus NOT NULL und UNIQUE
- FOREIGN KEY (R): legt eine Fremdschlüsselbeziehung fest
- CHECK (C): gibt eine/mehrere Bedingung(en) an, die erfüllt sein müssen

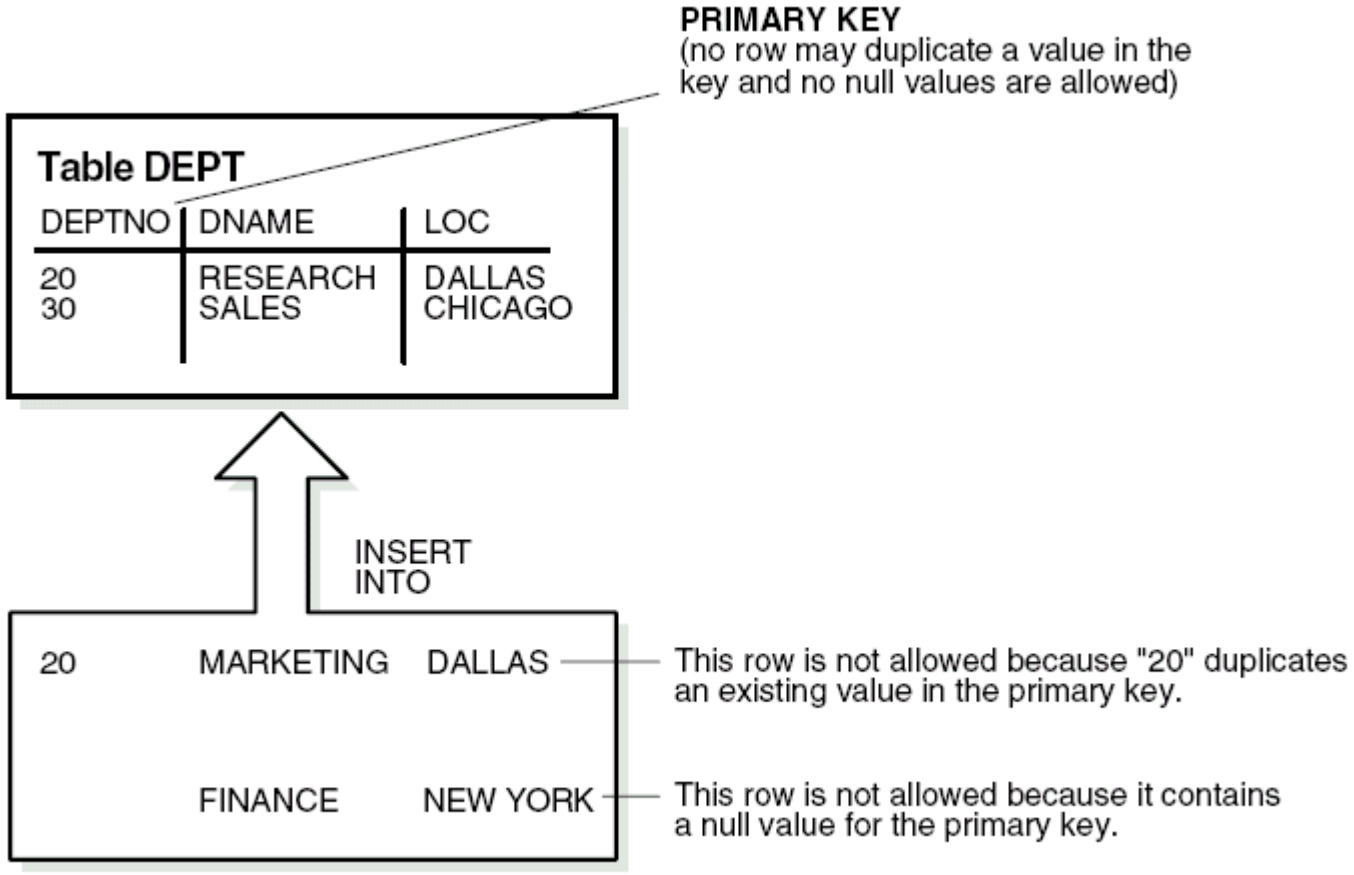
Constraints können entweder beim Anlegen mit CREATE TABLE
oder nachträglich über ALTER TABLE gesetzt werden.

CREATE TABLE:

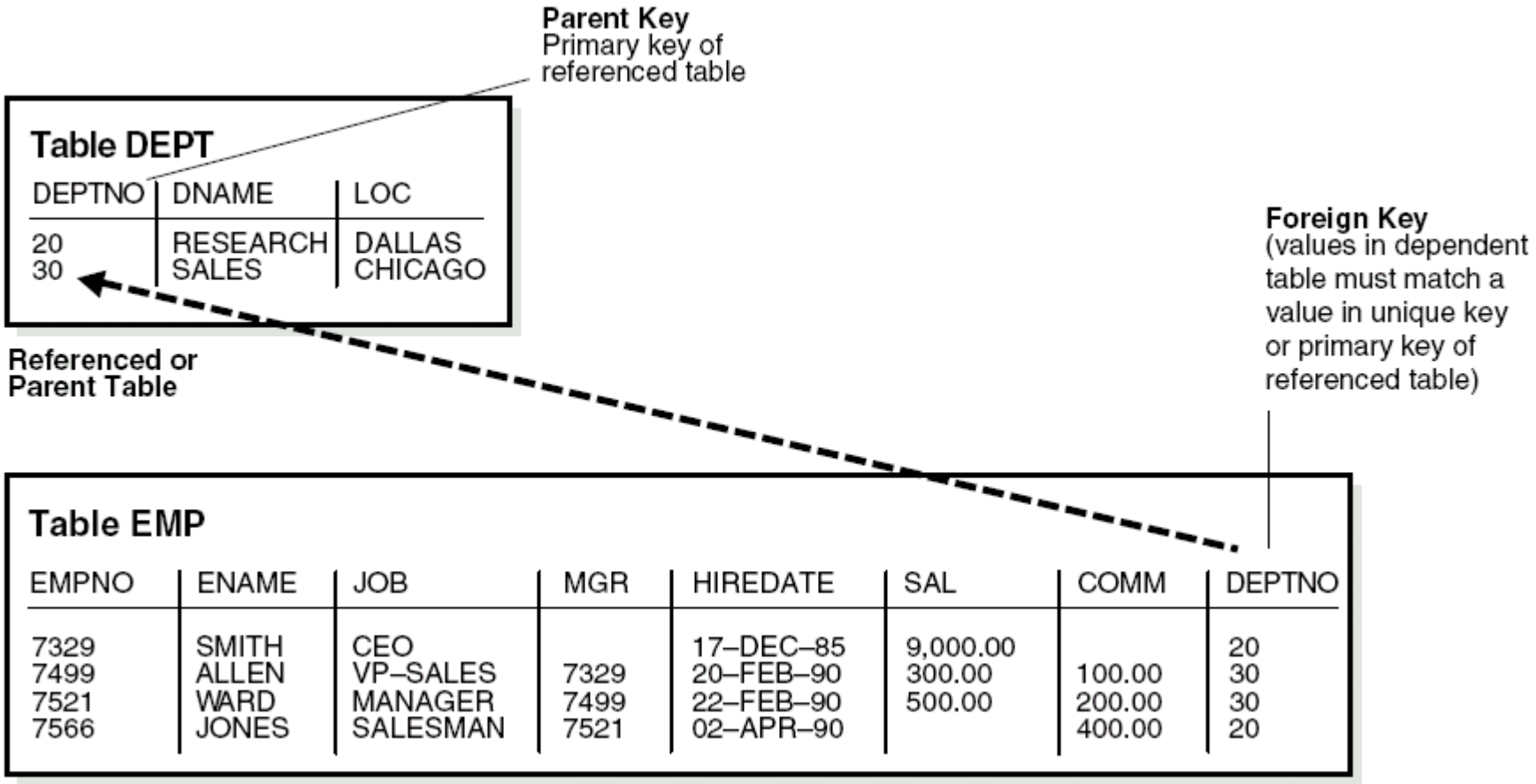
Default-Werte und Constraints

```
CREATE TABLE employees_demo
 ( employee_id NUMBER(6),
   first_name VARCHAR2(20),
   last_name VARCHAR2(25)
     CONSTRAINT emp_last_name_nn_demo NOT NULL,
   email VARCHAR2(25)
     CONSTRAINT emp_email_nn_demo NOT NULL,
   phone_number VARCHAR2(20),
   hire_date DATE DEFAULT SYSDATE
     CONSTRAINT emp_hire_date_nn_demo NOT NULL,
   job_id VARCHAR2(10)
     CONSTRAINT emp_job_nn_demo NOT NULL,
   salary NUMBER(8,2)
     CONSTRAINT emp_salary_nn_demo NOT NULL,
   commission_pct NUMBER(2,2),
  manager_id NUMBER(6),
  department_id NUMBER(4),
  dn VARCHAR2(300),
  CONSTRAINT emp_salary_min_demo
     CHECK (salary > 0),
  CONSTRAINT emp_email_uk_demo
     UNIQUE (email)                    ) ;
```

Primärschlüssel-Beziehung



Fremdschlüssel-Beziehung (referentielle Integrität)



Primär- und Fremdschlüssel festlegen

```
/* Definition auf Tabellenebene (outline) */
CREATE TABLE emp (empno number(3),
                  ename varchar2(10),
                  deptno number(3)
                  CONSTRAINT pk_emp PRIMARY KEY(empno),
                  CONSTRAINT fk_deptno FOREIGN KEY(deptno)
                      REFERENCES dept(deptno)
                  );
```

```
/* Definition auf Spaltenebene (inline) */
CREATE TABLE emp (empno number(3) CONSTRAINT pk_emp PRIMARY KEY,
                  ename varchar2(10),
                  deptno number(3) CONSTRAINT fk_deptno
                      REFERENCES dept(deptno)
                  );
```

Referentielle Integrität: Ändern des Standardverhaltens

```
/* Definition auf Tabellenebene (outline) */
CREATE TABLE emp (empno number(3),
                  ename varchar2(10),
                  deptno number(3)
                  CONSTRAINT pk_emp PRIMARY KEY(empno),
                  CONSTRAINT fk_deptno FOREIGN KEY(deptno)
                      REFERENCES dept(deptno)
                      DELETE ON CASCADE   /* bzw. ON DELETE SET NULL */
                  );
```

Welche Constraints existieren?

Es können folgende Sichten auf das Data Dictionary verwendet werden:

- DBA_CONSTRAINTS, ALL_CONSTRAINTS, USER_CONSTRAINTS (Gesamtübersicht)
- DBA_CONS_COLUMNS, ... (Welche Spalten sind betroffen)

Beispiel (Oracle-eigene Schreibweise für den JOIN):

```
SELECT a.constraint_name, a.constraint_type,
       a.table_name, b.column_name,
       a.search_condition, a.r_constraint_name
     FROM   user_constraints a, user_cons_columns b
     WHERE  a.constraint_name = b.constraint_name;
```

Tabellen durch Unterabfragen erstellen

```
CREATE TABLE dept_80
       ( d80_emplid,
         d80_name,
         d80_jobid DEFAULT 'UNKNOWN' )
AS SELECT employee_id,
         first_name || last_name Name,
         job_id
   FROM employees
   WHERE department_id = 80;
```

Weitere Parameter bei CREATE TABLE

```
CREATE [ GLOBAL TEMPORARY ] TABLE [ schema. ]table
   [ (relational_properties) ]
   [ ON COMMIT { DELETE | PRESERVE } ROWS ]
   [ physical_properties ]
   [ table_properties ] ;
```

- Protokollierung
- Verschlüsselung
- Temporäre/Dauerhafte Speicherung
- Physische Speicherorganisation (Tablespace, ..., Clustering)
- Logische Speicherorganisation (Komplexe Datenstrukturen, Nested Tables, ...)

Beispiel für temporäre Tabelle

```
CREATE GLOBAL TEMPORARY TABLE today_sales
    ON COMMIT PRESERVE ROWS
    AS SELECT * FROM orders WHERE order_date = SYSDATE;
```

- GLOBAL TEMPORARY:
  Tabelle ist temporär und für alle Sessions mit entspr. Privilegien sichtbar
- ON COMMIT PRESERVE ROWS:
  Die in der jeweiligen Session eingefügten Daten (Zeilen) werden zum Ende der Sitzung wieder gelöscht (default: bei jedem Transaktionsende, sprich COMMIT)

CREATE TABLE: Beispiel für Speicher-Parameter

```
CREATE TABLE divisions
        (div_no NUMBER(2),
         div_name VARCHAR2(14),
         location VARCHAR2(13) )
   STORAGE ( INITIAL 100K NEXT 50K
             MINEXTENTS 1 MAXEXTENTS 50 PCTINCREASE 5);
```

ALTER TABLE: Spalten hinzufügen, ändern, umbenennen, löschen

```
ALTER TABLE countries
       ADD (duty_pct NUMBER(2,2) CHECK (duty_pct < 10.5),
            visa_needed VARCHAR2(3));
```

```
ALTER TABLE countries
       MODIFY (duty_pct NUMBER(3,2));

ALTER TABLE product_information
       MODIFY (min_price DEFAULT 10);
```

```
ALTER TABLE supplier
              RENAME COLUMN supplier_name to sname;
```

```
ALTER TABLE supplier
       DROP COLUMN supplier_name;
```

Löschen von Spalten mit Constraints, die andere Spalten einbeziehen: Alle Constraints löschen

```
CREATE TABLE t1
     (  pk NUMBER PRIMARY KEY,
        c1 NUMBER,
        c2 NUMBER,
        CONSTRAINT ck1 CHECK (pk > 0 and c1 > 0),
        CONSTRAINT ck2 CHECK (c2 > 0)    );

CREATE TABLE t2
     (  fk NUMBER,
        CONSTRAINT ri FOREIGN KEY (fk) REFERENCES t1(pk) );

/* The next two statements return errors:
ALTER TABLE t1 DROP (pk); -- pk is a parent key
ALTER TABLE t1 DROP (c1); -- c1 is referenced by multicolumn
                          -- constraint ck1

/*Mögliche Lösung: */
ALTER TABLE t1 DROP (pk) CASCADE CONSTRAINTS;
```

Constraints hinzufügen und löschen

```
ALTER TABLE test MODIFY (name CONSTRAINT nn_name NOT NULL);


ALTER TABLE test ADD (CONSTRAINT ck_name
       CHECK (upper(name)=name) AND length(name)>=5));


ALTER TABLE emp_copy ADD (CONSTRAINT fk_copy FOREIGN KEY(deptno)
       REFERENCES dept(deptno) ON DELETE CASCADE);
                        /* alternativ: ON DELETE SET NULL */
```

```
ALTER TABLE employees DROP UNIQUE (email);


ALTER TABLE departments DROP PRIMARY KEY CASCADE;
/* ALTER TABLE departments DROP CONSTRAINT pk_dept CASCADE; */
```

Constraints können deaktiviert/aktiviert werden

```
/* Definition auf Spaltenebene (inline) */

CREATE TABLE emp (empno number(3) CONSTRAINT pk_emp PRIMARY KEY
DISABLE,
                  ename varchar2(10),
                  deptno number(3) CONSTRAINT fk_deptno
                      REFERENCES dept(deptno)
                  );
```

- Beim Erzeugen kann das Constraint deaktiviert werden (Aktivierung ist default)
- Mit ALTER kann das Constraint später aktiviert werden

Constraints aktivieren und deaktivieren

```
ALTER TABLE locations
       MODIFY PRIMARY KEY DISABLE CASCADE;


ALTER TABLE employees ADD CONSTRAINT check_comp
       CHECK (salary + (commission_pct*salary) <= 5000)
       DISABLE;
```

```
ALTER TABLE employees ENABLE VALIDATE CONSTRAINT emp_manager_fk
       EXCEPTIONS INTO exceptions;
```

Constraint-Prüfung verzögern

```
ALTER TABLE dept2
     ADD CONSTRAINT dept2_id_pk PRIMARY KEY (department_id)
       DEFERRABLE INITIALLY DEFERRED;
                                .
                                .
                                .

SET CONSTRAINTS dept2_id_pk IMMEDIATE;
```

TRUNCATE: Löschen aller Zeilen

```
TRUNCATE TABLE copy_emp;
```

- Entfernt alle Zeilen aus der Tabelle
- Ist effizienter als das Löschen aller Zeilen mit DELETE
- Tabellenstruktur verbleibt im Data Dictionary
- Constraints werden beachtet, sofern sie aktiviert sind
- Es werden keine DELETE-Trigger, sofern definiert, ausgelöst
- Es ist kein Rollback möglich

DROP: Löschen von Tabellen

```
DROP TABLE list_customers CASCADE CONSTRAINTS PURGE;
```

- Alle Daten und die Struktur der Tabelle werden gelöscht
- Alle noch offenen Transaktionen werden festgeschrieben
- Alle Indizes für die Tabelle werden gelöscht
- Alle Constraints werden gelöscht
  (CASCADE CONSTRAINTS: Fremdschlüsselbeziehungen. werden ebenfalls zurückgesetzt)
- PURGE: Der Speicherplatz wird freigegeben, d.h. kann nicht rückgängig gemacht werden.
- Es ist kein Rollback möglich

Löschen von Tabellen: Der Papierkorb "RECYCLEBIN" (ab Version 10)

```
DROP TABLE list_customers;

SELECT object_name, droptime FROM user_recyclebin
       WHERE original_name = 'LIST_CUSTOMERS';
```

```
OBJECT_NAME DROPTIME
------------------------------ -------------------
RB$$45703$TABLE$0 2003-06-03:15:26:39
RB$$45704$TABLE$0 2003-06-12:12:27:27
RB$$45705$TABLE$0 2003-07-08:09:28:01
```

```
FLASHBACK TABLE list_customers TO BEFORE DROP;

FLASHBACK TABLE list_customers TO BEFORE DROP
       RENAME TO list_customers_old;
```