SQL – Data Definition Language

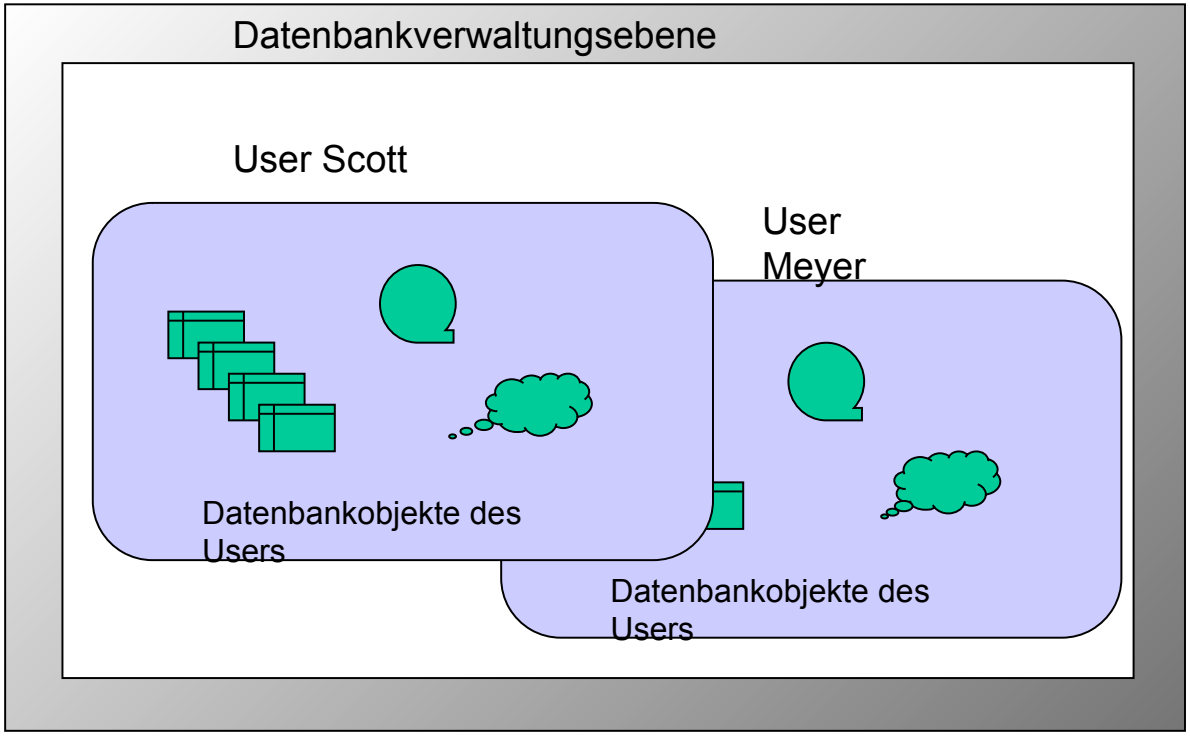
Stephan Karrer

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

Datenbank

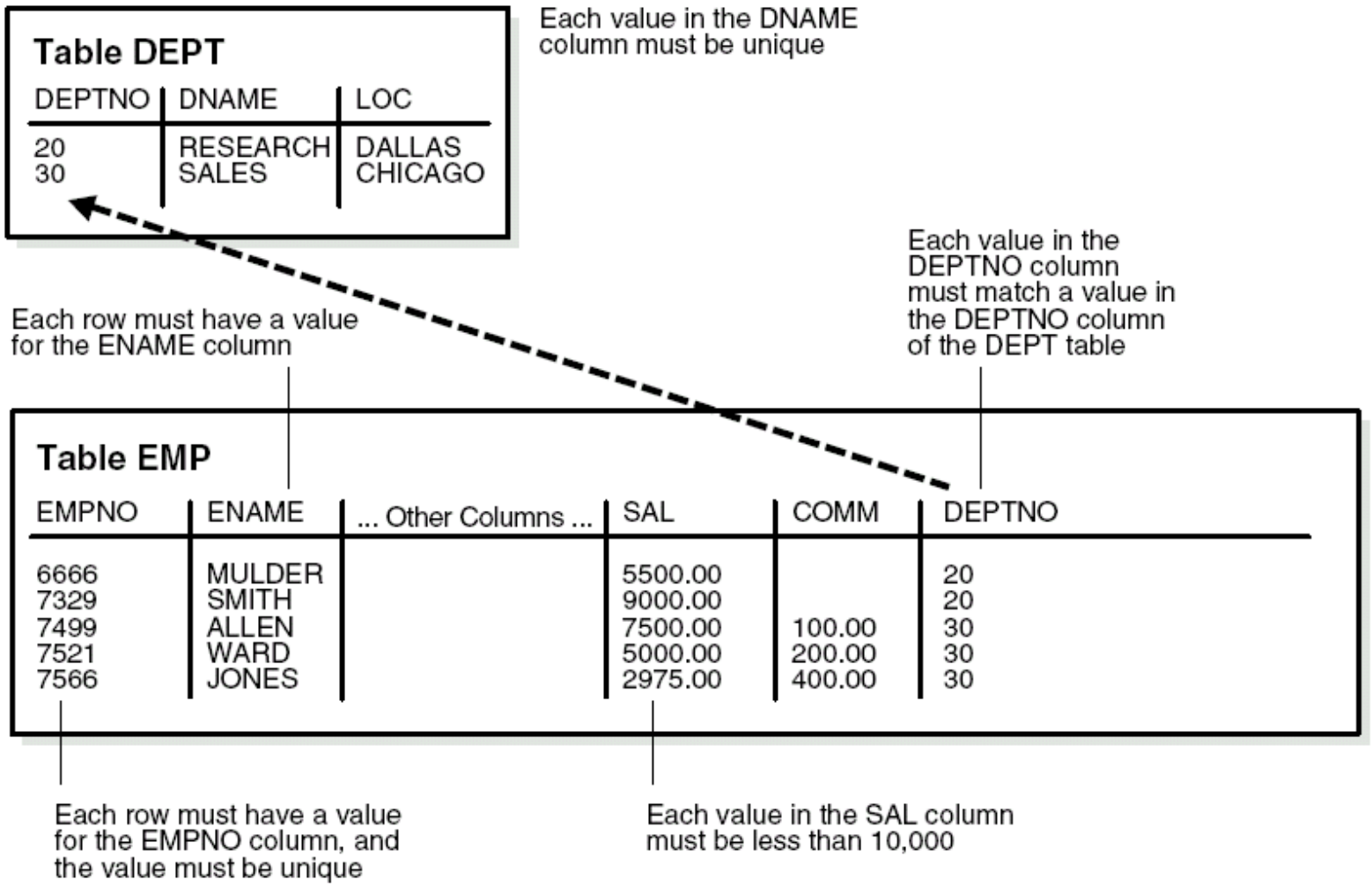


Erzeugen von Tabellen mit der CREATE-Anweisung

```
CREATE TABLE [ schema. ]table
   [ relational_properties ]
   [ vendor-specific 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),
     description VARCHAR2(300)
    ) ;
```

Integritätsbedingungen



Constraints: Bedingungen auf Tabellen- bzw. Spalten-Ebene

Folgende Constraints sind in Oracle zulässig (ANSI-konform):
(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 (Verwendung von Oracle Typen)

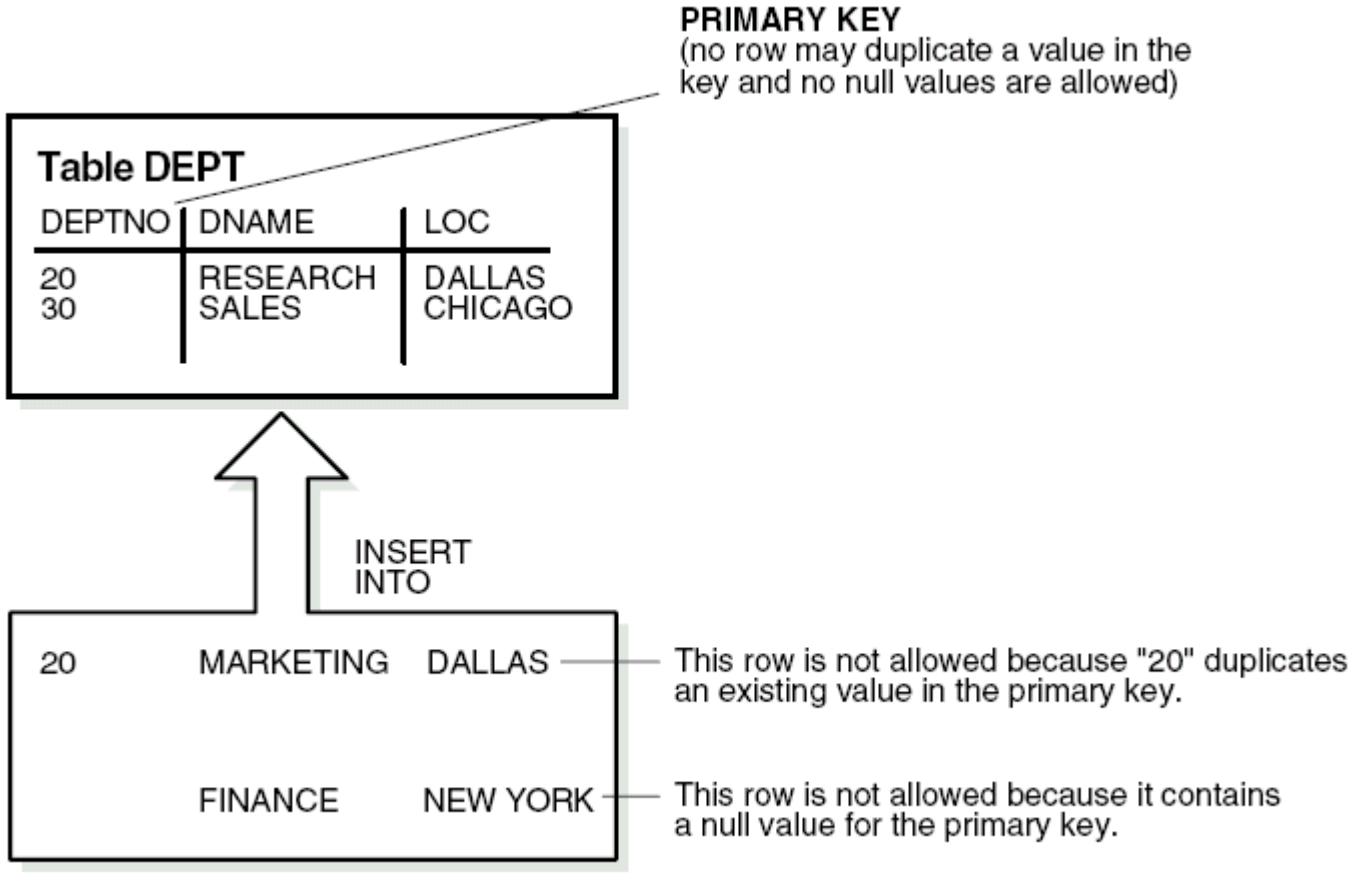
```
CREATE TABLE employees_demo
 ( employee_id NUMBER(6),
   first_name VARCHAR2(20),
   last_name VARCHAR2(25)
         CONSTRAINT emp_last_name_nn_demo NOT NULL,
   email VARCHAR2(25)
         CONSTRAINT emp_email_nn_demo NOT NULL,
   phone_number VARCHAR2(20),
   hire_date DATE DEFAULT SYSDATE
         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),
   CONSTRAINT emp_salary_min_demo CHECK (salary > 0),
   CONSTRAINT emp_email_uk_demo UNIQUE (email)
) ;
```

Check Constraints

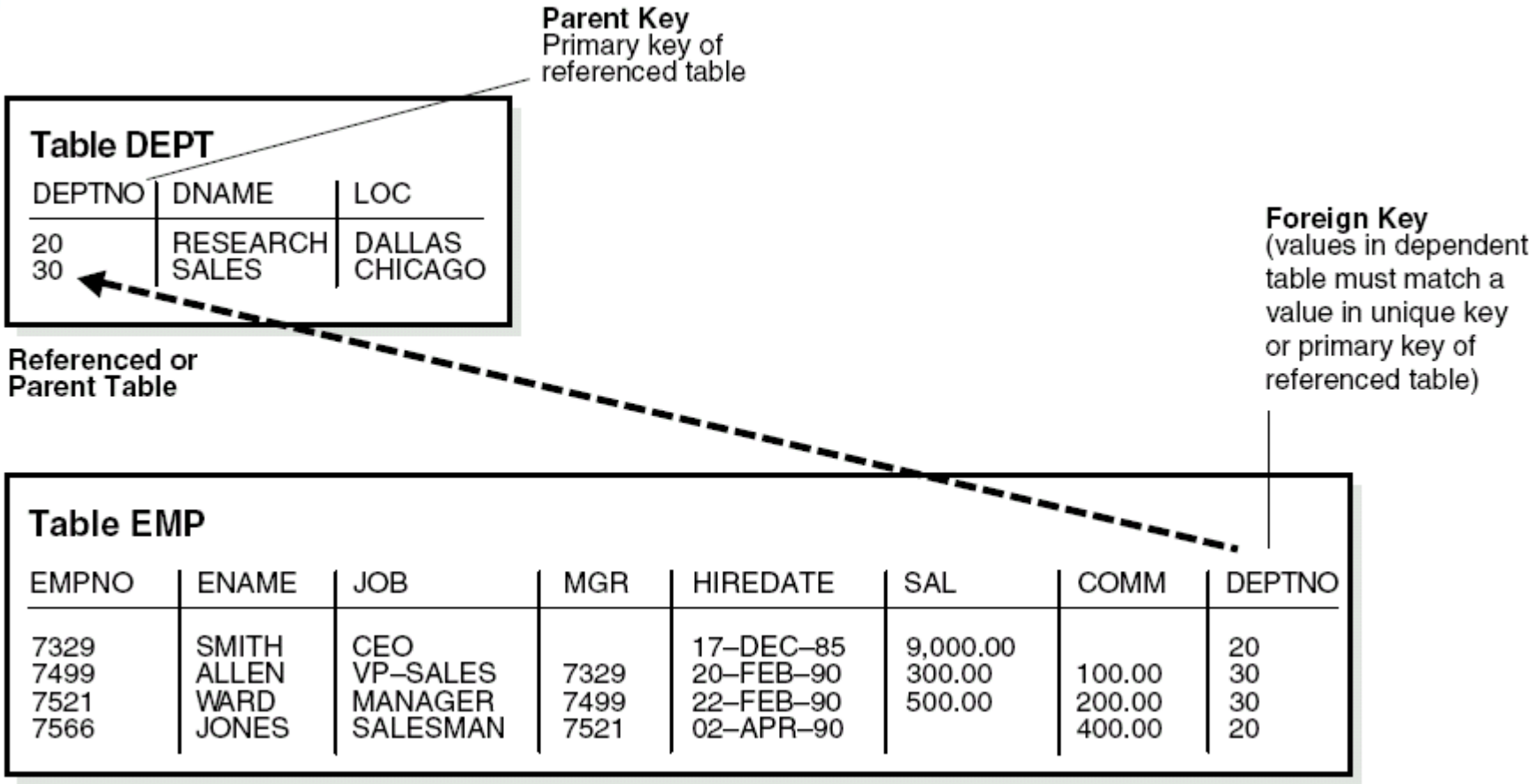
```
CREATE TABLE products (
    product_no integer,
    name varchar(20),
    price numeric CONSTRAINT nc CHECK (price > 0),
    discounted_price numeric,
    CONSTRAINT dc CHECK (discounted_price > 0),
    CHECK (price > discounted_price)  -- uebergreifend
);
```

- Constraints können benannt werden, ansonsten vergibt das System einen spezifischen Bezeichner
- Spaltenübergreifende Constraints müssen seperat definiert werden (outline)

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?

Bei Oracle können folgende Sichten auf das Data Dictionary verwendet werden:

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

Beispiel:

```
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;
```

Automatische Schlüsselgenerierung

- Häufig sollen künstlich generierte eindeutige Werte für Schlüsselspalten verwendet werden

- 2 Verfahren sind üblich und auch ANSI-konform:
  - Auto-Increment: pro Spalte eigene Generierung von Ganzzahlen
  - Sequenzgenerator: eigenes Datenbank-Objekt, das übergreifend genutzt werden kann und Ganzzahlen produziert

- ANSI definiert dafür folgende Syntax:
  "GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY"
  - Leider hält sich kaum ein Hersteller an diese Syntax

Sequenzgenerator bei Oracle

```
CREATE SEQUENCE seq1
    INCREMENT BY 5
    START WITH 10
    MAXVALUE 10000
    CACHE   50;

CREATE TABLE Test1 (  id NUMBER DEFAULT seq1.nextval,
                      name VARCHAR2(99)  );

-- oder automatischer Sequenzgenerator (ab 12c)
CREATE TABLE Test2 (
           id NUMBER GENERATED ALWAYS AS IDENTITY
           (START WITH 100 INCREMENT BY 10));
```

Auto-Increment Spalte bei SQL Server

```
CREATE TABLE new_employees
(
 id_num int IDENTITY(1,1),  -- (seed, increment)
 fname varchar (20),
 minit char(1),
 lname varchar(30)
);
```

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;
```

- Hersteller weichen manchmal von der Syntax ab, z.B. SQL SERVER

```
SELECT Customers.CustomerName, Orders.OrderID
    INTO CustomersOrderBackup2017
FROM Customers LEFT JOIN Orders
    ON Customers.CustomerID = Orders.CustomerID;
```

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;
```

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
- Es ist kein Rollback möglich

DROP: Löschen von Tabellen

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

- Alle Daten und die Struktur der Tabelle werden gelöscht
- Alle Indizes für die Tabelle werden gelöscht
- Alle Constraints werden gelöscht
  (CASCADE CONSTRAINTS: Fremdschlüsselbeziehungen. werden ebenfalls zurückgesetzt)
- Es ist kein Rollback möglich ? (hängt vom Hersteller ab)