4.1 SQL – DDL und DML

- 4.1 SQL – DDL und DML
  - Einführung
  - Datendefinition (DDL)
  - Datenbankmodifikation (DML)
  - Zugriffsüberwachung
- 4.2 SQL-Anfragen 1
- 4.3 SQL-Anfragen 2
- 4.4 Programmiersprachen-Anbindung

Literaturempfehlung

- G. Kuhlmann, F. Müllmerstadt: SQL – Der Schlüssel zu relationalen Datenbanken, Rowohlt Taschenbuch Verlag, 2004
- K.E. Kline: SQL in a nutshell – A Desktop Quick Reference, O'Reilly, 3. Auflage, 2009
- B. Brumm: Beginning Oracle SQL for Oracle Database 18c, Apress, 2019
  als E-Book in HTWG-Bibliothek
- L. de Haan, D. Fink, T. Gorman, I. Jørgensen, K. Morton: Beginning Oracle SQL, Apress, 2009
  als E-Book in HTWG-Bibliothek
- L. de Haan: Mastering Oracle SQL and SQL*Plus, Apress, 2005
  als E-Book in HTWG-Bibliothek
- Oracle 19c SQL Language Reference
  - http://oracle19c.in.htwg-konstanz.de/sqlrf/index.html

Structured Query Language (SQL)
Einführung

- Entwicklung zu Beginn der 70er Jahre, Standardisierung durch ANSI
- 1986 SQL-86, Alias: SQL0
- 1989 SQL-89, Alias: SQL1
- 1992 SQL-92, Alias: SQL2
- 1999 SQL-99, Alias: SQL3
  - Objekt-relationale Erweiterungen
  - Aktive Regeln, etc.
- 2003 SQL:2003, Alias: SQL4
  - SQL/XML: XML-Datentyp, Zugriff auf XML-Daten
  - SQL/MED: Management of External Data
- 2008 SQL:2008
- 20nn: ...

SQL Eigenschaften

- Nichtprozedurale, deskriptive Sprache
  - Fragesteller gibt Frage, aber keinen Algorithmus zur Lösung vor
  - Gegensatz 3GL: Java, C, PASCAL, etc.

1. Generation:	Maschinensprache
2. Generation:	Assembler
3. Generation:	Problemorientierte Sprachen (Java, C, C++, Basic)
4. Generation:	Nichtprozedurale Sprachen (SQL)

SQL als nichtprozedurale Sprache

Problemorientierte Sprache (3GL)	SQL (4GL)

`open(buecher);`
`while ( !EOF (buecher) )`
`{`
`  read(buch);`
`  if (buch.leihfrist > 0)`
`  if (buch.schlagwort = 'SQL')`
`  print(buch.autor, buch.titel);`
`}`
`close(buecher);`
	`SELECT autor, titel`
	`FROM  buecher`
	`WHERE  leihfrist > 0`
	`AND schlagwort = 'SQL';`

Hochschule Konstanz
Fakultät Informatik
Prof. Dr. Oliver Eck

4.1 SQL – DDL und DML

- 4.1 SQL – DDL und DML
  - Einführung
  - Datendefinition (DDL)
  - Datenbankmodifikation (DML)
  - Zugriffsüberwachung
- 4.2 SQL-Anfragen 1
- 4.3 SQL-Anfragen 2
- 4.4 Programmiersprachen-Anbindung

Übersicht SQL

- DDL (Data Definition Language)
  - Datendefinitionssprache zur Beschreibung von
    - Relationen
    - Attribute
    - Wertebereiche
    - Schlüssel
- DML (Data Manipulation Language)
  - Datenmanipulationssprache
  - Mengenorientierte Operationen
- DQL (Data Query Language)
  - Datenabfragesprache
  - Mengenorientierte Operationen

Physischer Entwurf

- Entwurfsschritt physischer Entwurf
  - "Einrichten" des Datenbankschemas mit Hilfe der Datendefinitions-sprache des gewählten Systems
  - Vergabe von Zugriffsrechten
  - Definition von Indexstrukturen für effizienten Datenzugriff
  - Gruppierung von Blöcken zu Cluster
  - Festlegung Datei-Formate

Hinweise zur Sprache SQL

- SQL ist nicht case-sensitiv
  - Keine Unterscheidung zwischen Groß- und Kleinschreibung
  - `select name from kunde;`
  - `SELECT NAME FROM KUNDE;`
  - `sEleCT namE frOM kunDE;`
- Zeichenketten in Hochkommata sind case-sensitiv
  - `'Konstanz' ≠ 'KONSTANZ'`
- Attributwerte in Oracle
  - `Zahlen: 100 bzw. 9.55`
  - `Zeichenketten: 'Name'`
  - `Datum: '10-MAR-05'`

Datendefinition (DDL)

- Erzeugen einer neuen Datenbank
  - `CREATE DATABASE databasename;`
- Erzeugen neuer Tabellen (Relationen)
  - `CREATE TABLE tabellenname`
    `(spaltenname datentyp [NOT NULL] [DEFAULT deftyp]`
    `[, spaltenname datentyp] [, ...] )`
- Festlegung von Primär- und Fremdschlüsseln
  - `PRIMARY KEY ( spaltenliste )`
  - `FOREIGN KEY ( spaltenliste )`
    `REFERENCES tabellenname [( spaltenliste )]`
- Modifizieren von Tabellen
  - `ALTER TABLE tabellenname;`
- Löschen von Tabellen
  - `DROP TABLE tabellenname;`

Datendefinition (DDL)
Beispiel

```
CREATE TABLE Studium
( studienfachNr     integer,
  studienfach       varchar(20),
  anzahlSemester    integer,
  abschluss         varchar(20),
  CONSTRAINT Studium_pk PRIMARY KEY(studienfachNr) );


CREATE TABLE Student
( matrikelNr        integer,
  name              varchar(20),
  studienfachNr     integer,
  semester          integer,
  studienort        varchar(20),
  CONSTRAINT Student_pk PRIMARY KEY(matrikelNr),
  CONSTRAINT Student_fk FOREIGN KEY (studienfachNr)
             REFERENCES Studium(studienfachNr)
             ON DELETE SET NULL );
```

Datentypen in SQL

`CHAR(n)`	Zeichenkette der Länge n
`VARCHAR(n)` in Oracle: `VARCHAR2(n)`	Zeichenkette variabler Länge mit maximaler Länge n
`INTEGER oder INT`	Ganzzahl
`NUMERIC(i,n)` in Oracle: `NUMBER(i,n)`	Dezimalzahl mit i Insgesamt-Stellen und n Nachkommastellen, Darstellung exakter numerischer Werte
`FLOAT`	Binäre Gleitkommazahl: Darstellung annähernder numerischer Werte
`DATE`	Datum
`CLOB, BLOB, NCLOB`	große Objekte

- Es gibt in Oracle keinen Datentyp BOOLEAN
  - Ersatz:

```
flag char(1) check (flag in ( 'Y', 'N' ))
```

Datendefinition
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Erzeuge eine neue Tabelle Abt für Abteilung.

Datendefinition
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Erzeuge eine neue Tabelle Pers für Personal.

Schema-Modifikation

- Löschen und Ändern von Spalten
  - ALTER TABLE table_name
    ADD column_name datatype;
  - ALTER TABLE table_name DROP column_name;
  - ALTER TABLE table_name
    MODIFY column_name datatype;
- Nachträgliches Hinzufügen oder Entfernen von Constraints
  - Reihenfolge der Tabellendefinition beliebig
  - Rekursive Fremdschlüssel-Beziehungen möglich

```
ALTER TABLE PERS ADD (
CONSTRAINT fk_abt2 FOREIGN KEY(anr) REFERENCES Abt(anr)
);
ALTER TABLE PERS DROP CONSTRAINT fk_abt2;
```

Datenintegrität

- Statische Integritätsbedingung (Zustandsbedingungen)
  - Bedingung, die von jedem Zustand der Datenbank erfüllt werden muss
  - Beispiele:
    - Kilometerzahl eines Autos ≥ 0
    - Professoren haben Besoldungsgruppe C2, C3, C4, W2, W3, W4
  - Definition in SQL: CHECK
- Dynamische Integritätsbedingung (Übergangsbedingungen)
  - Beispiele:
    - Kilometerzahl eines Autos kann nur größer werden
    - Professoren können nur befördert, nicht degradiert werden
  - Definition in SQL: Trigger

Datenintegrität

- Spaltenbedingungen (column constraints)
  - Spezialfall einer Tabellenbedingung
  - Definition in Spalten- oder Tabellendefinition
  - Wertebereichseinschränkungen (domain constraints)
  - Bsp: Gehalt > 20.000
- Tabellenbedingungen (table constraints)
  - Integritätsbedingungen über der betreffenden Tabelle
  - Definition in Tabellendefinition
  - Bsp: Gehalt > 30.000 bei 20-jähriger Firmenzugehörigkeit
- General Constraint
  - Integritätsbedingungen über beliebige Spalten beliebiger Tabellen
  - Syntax: `CREATE ASSERTION`
  - In SQL-Standard vorgesehen, nicht in allen DBMS realisiert
  - Bsp: Programmierer arbeiten in Erlangen

Spalten- und Tabellenbedingungen

- PRIMARY KEY
  - Definition einer oder mehrerer Spalten als Primärschüssel
- UNIQUE
  - Verhindern, dass in einer Spalte doppelte Werte eingegeben werden
- DEFAULT
  - Definition eines Default-Werts
- NOT NULL
  - Kein NULL-Wert erlaubt
- CHECK
  - Definition einer Bedingung, der jeder Datensatz genügen muss
  - Bereichsangaben `CHECK (a>=0 AND a<10)`
  - Konkrete Werte `CHECK (anrede IN ('Herr', 'Frau'))`
  - Werterelationen `CHECK (kundennr <= 4711)`

Check-Bedingung

- Inline-Definition
  - Innerhalb der Attribut-Definition

```
CREATE TABLE Pers(
  ...
  gehalt integer NOT NULL CHECK (gehalt > 0),
... )
```

Out-of-line Definition

  - Ausserhalb Attributdefinition als Teil der Tabellendefinition

```
CREATE TABLE Pers(
  ...
  gehalt integer NOT NULL,
  ...
  CONSTRAINT PersGehalt CHECK (gehalt > 0),
  ...);
```

Inline vs. Out-of-line Definition

- Inline-Definition
  - Mit oder ohne Constraintname möglich
  - NUT NULL kann nur inline definiert werden
- Vorteile bei Definition eines Constraintnamens
  - Verständlichere Fehlermeldungen
  - Einfacheres Löschen oder Modifizieren des Constraints

```
CREATE TABLE Pers(
   ...
   gehalt integer NOT NULL CHECK (gehalt > 0),
   jahrg  integer CONSTRAINT persjahrg CHECK(jahrg > 1900),
   ...
);
```

Spezifikation von Constraints

- Beispiele für CHECK

```
CREATE TABLE Pers(
  pnr                integer NOT NULL,
  name               varchar2(20) NOT NULL,
  jahrg              integer NOT NULL,
  eindat             date NOT NULL,
  gehalt             integer NOT NULL,
  beruf              varchar2(20),
  anr                char(3) NOT NULL,
  vnr                integer,
  CONSTRAINT pers_pk PRIMARY KEY(pnr),
  CONSTRAINT pers_fk FOREIGN KEY (anr) REFERENCES abt(anr),
  CONSTRAINT persjahrgang CHECK (jahrg > 1900
                                 AND jahrg < 2100),
  CONSTRAINT PersGehalt CHECK (gehalt > 0),
);
```

Überprüfung von Regular Expressions

- Regular Expressions zur Definition von Integritätsconstraints
  - Verwendung von REGEXP_LIKE in Oracle
  - Beispiel: HTWG-Telefonnummern sollen im folgenden Format definiert sein:
    XXXXX-XXX-XXX
  - Beispielnummer: 07531-206-630

```
CREATE TABLE Person
(
  name        VARCHAR2(30),
  telnummer   VARCHAR2(30),
    CONSTRAINT person_telnummer_format
      CHECK ( REGEXP_LIKE
                ( telnummer, '^\d{5}-\d{3}-\d{3}$' ) )
);
```

Semantische Datenintegrität
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Es existieren nur die Orte Nürnberg, Erlangen und Konstanz.

Semantische Datenintegrität
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Das Minimalgehalt beträgt 20.000

Semantische Datenintegrität
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Das Minimalgehalt bei 20-jähriger Betriebszugehörigkeit beträgt 30.000

Fremdschlüsselbedingungen
(referentielle Integrität)

- Parent-Tabelle
  - Tabelle, die den Primärschlüssel enthält
- Dependent (Child) -Tabelle
  - Tabelle, die den Fremdschlüssel enthält
- Fremdschlüsselbedingung (referentielle Integrität)
  - Zu jedem von NULL verschiedenen Fremdschlüsselwert der Dependent-Tabelle existiert ein entsprechender Schlüsselwert der Parent-Tabelle
  - Es existieren somit keine "dangling references"

Pers

pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt

anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

HTWIG Hochschule Konstanz
Fakultät Informatik
Prof. Dr. Oliver Eck

Fremdschlüsselbedingungen

- Lösch- und Änderungsregeln
  - Aktionen zur Gewährleistung der referentiellen Integrität
  - Durchführung bei DELETE- und UPDATE
  - Syntax:
```
CONSTRAINT pers-fk FOREIGN KEY (anr)
        REFERENCES abt(anr) ON DELETE SET NULL
```

ON DELETE RESTRICT	Verhindern von Löschungen (Default bei Oracle)
ON DELETE CASCADE	Automatisches Löschen weiterer Sätze
ON DELETE SET NULL	Automatisches auf NULL setzen
ON DELETE SET DEFAULT	Automatisches auf Default setzen

4.1 SQL – DDL und DML

- 4.1 SQL – DDL und DML
  - Einführung
  - Datendefinition (DDL)
  - Datenbankmodifikation (DML)
  - Zugriffsüberwachung
- 4.2 SQL-Anfragen 1
- 4.3 SQL-Anfragen 2
- 4.4 Programmiersprachen-Anbindung

Datenbankmodifikation (DML)

- Einfügen von Tupeln (Insertion)
  - `INSERT INTO tabellenname VALUES (...);`
  - `INSERT INTO tabellenname SELECT-Anweisung;`
- Löschen von Tupeln
  - `DELETE FROM tabellenname [WHERE bedingung];`
- Verändern von Tupeln (Update)

```
UPDATE tabellenname
    SET spaltenname = wert
       [,spaltenname = wert] [, ...]
    WHERE bedingung;
```

Datenbank-Updates
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Füge einen neuen Angestellten mit PNR = 007 und Namen Mayer in Abteilung K55 ein.

Datenbank-Updates
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Erhöhe das Gehalt von Herrn Abel auf 45.000 €.

Datenbank-Updates
Beispiel

Pers = ({pnr, name, jahrg, eindat, gehalt, beruf, anr, vnr})
Abt = ({anr, aname, ort})

Pers							
pnr	name	jahrg	eindat	gehalt	beruf	anr	vnr
406	Coy	1950	01.03.86	80.000	Kaufmann	K55	123
123	Mueller	1958	01.09.80	68.000	Programmierer	K51	
829	Schmidt	1960	01.06.90	74.000	Kaufmann	K53	123
874	Abel		01.05.94	62.000	Softw.Entwickler	K55	829
503	Junghans	1975		55.000	Programmierer	K51	123
...	...	...	...	...	...	...	...

Abt		
anr	aname	ort
K51	Entwicklung	Erlangen
K53	Buchh	Nürnberg
K55	Personal	Nürnberg
...	...	...

Lösche den Angestellten mit PNR = 007.

Vergabe von Schlüsselwerten in Oracle

- Verwendung von Sequenzen
  - Automatische Vergabe von eindeutigen Zahlen

```
CREATE SEQUENCE persID INCREMENT BY 1 START WITH 100;

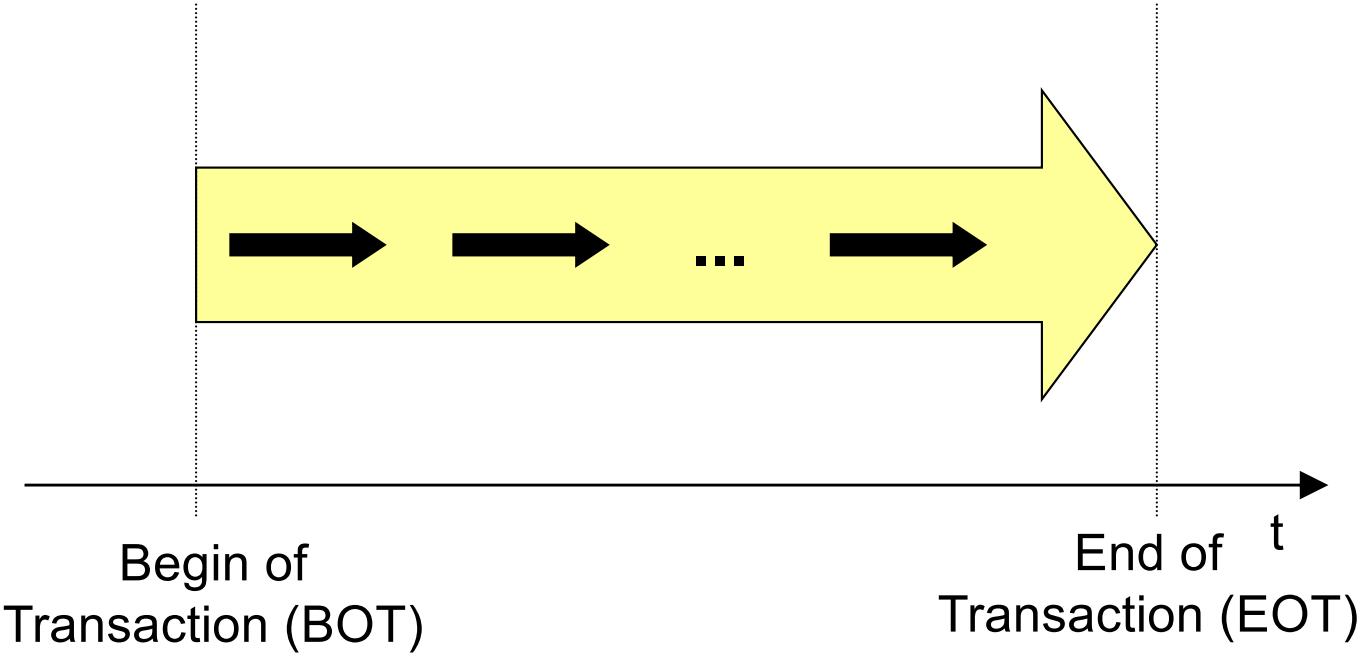
INSERT INTO Pers(pnr, name, jahrg)
  VALUES (persID.NextVal, 'Jens Maier', 1960);
```

- Identity Columns
  - Neu in Oracle19c

```
CREATE TABLE T1 (c1 NUMBER GENERATED BY DEFAULT
                    ON NULL AS IDENTITY,
                 c2 VARCHAR(10));
INSERT INTO T1(c2) VALUES ('Wert 1');
INSERT INTO T1(c1, c2) VALUES (NULL, 'Wert 2');
```

Definition Transaktion

Eine Transaktion ist eine ununterbrechbare Folge von DB-Operationen, die eine DB von einem konsistenten Zustand in einen (nicht notwendigerweise verschiedenen) konsistenten Zustand überführt.

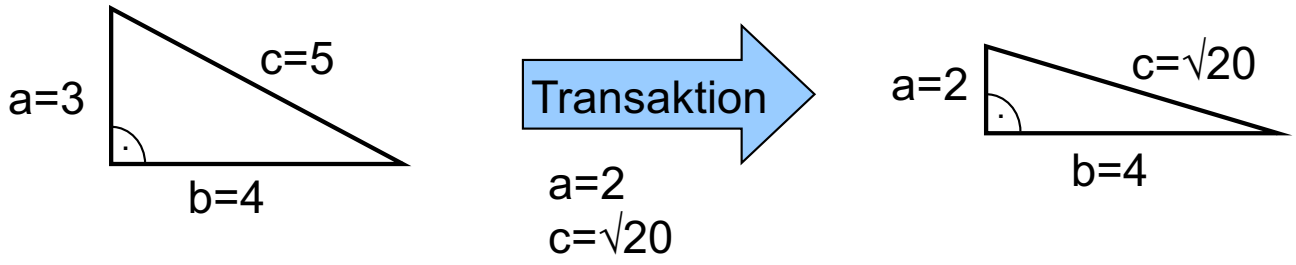


Datenbank-Transaktionen

- Commit
  - Alle Änderungen einer Transaktion werden permanent
  - Rollback ist anschließend nicht mehr möglich
  - Syntax: `COMMIT;` (äquivalent: `COMMIT WORK;`)
- Rollback
  - Einfügen, Aktualisieren und Löschen von Daten kann rückgängig gemacht werden
  - Wichtig bei Fehlern
  - Syntax: `ROLLBACK;`
- DDL-Statements bilden immer eine Transaktion und müssen nicht mit COMMIT abgeschlossen werden
  - `CREATE TABLE...`
  - `DROP TABLE...`

Arten von Konsistenz

- Unverzögerte Integritätsbedingungen
  - Überwachung direkt nach DB-Änderung, z.B. Wertebereichsgrenzen
  - Bsp: Mitarbeitergehalt > 0
- Verzögerte Integritätsbedingungen
  - Überwachung am Transaktionsende
  - Inkonsistente Zustände innerhalb der Transaktion erlaubt
  - Rechtwinkliges Dreieck: $a^2 + b^2 = c^2$



Überprüfung von Constraints in Oracle

```
CREATE TABLE pers
( pnr        integer PRIMARY KEY,
  name       varchar2(20),
  jahrg      integer,
  eindatum   date,
  gehalt     integer,
  beruf      varchar2(20),
  anr        char(3),
  vnr        integer,
  CONSTRAINT pers_fk
             FOREIGN KEY (anr) REFERENCES abt(anr) DEFERRABLE);

SET AUTOCOMMIT OFF;

SET CONSTRAINTS ALL DEFERRED;

INSERT INTO pers
   VALUES (444, 'test', 1980, '10-JAN-1990', 50000,
   'Kaufmann', 'K66', NULL);

INSERT INTO abt VALUES ('K66', 'Entw', 'Konstanz');

COMMIT;
```

4.1 SQL – DDL und DML

- 4.1 SQL – DDL und DML
  - Einführung
  - Datendefinition (DDL)
  - Datenbankmodifikation (DML)
  - Zugriffsüberwachung
- 4.2 SQL-Anfragen 1
- 4.3 SQL-Anfragen 2
- 4.4 Programmiersprachen-Anbindung

Sicherheit in Datenbanken

- Information ist ein wertvolles Gut
  - Vertraulichkeit (z.B. Gehalt)
  - Löschen aller Daten kann das Ende einer Firma bedeuten
  - Verhindern von Manipulationen (z.B. Gehalt)
- Unterschiedliches Schutzbedürfnis
  - Hochschule
  - Betrieb
  - Militärische Anlagen

Zugriffsschutz in SQL

- Zugriffsschutz
  - Kontrolliert Zugriffe von Subjekten auf Objekte
- Benutzerbestimmbare Zugriffskontrolle
  - Eigentümer-Prinzip: Jedes Objekt hat einen Eigentümer
  - Eigentümer ist zuständig für Schutz
  - Interne Kennung eines DB-Benutzer, z.B. User-ID
  - Rolle: Gruppe zusammenhängender Privilegien
- Identifikation und Authentisierung
  - Identifikation von Benutzern vor Zugang zu DBMS (Benutzername)
  - Authentifizierung meist durch Passwort
- Autorisierung und Zugriffskontrolle
  - Prinzip des kleinstmöglichen Privilegs
  - Menge von Regeln, die die erlaubten Arten des Zugriffs bestimmt

Zugriffsüberwachung (Data Control Facility)

- Zugriffsberechtigungen
  - `ALL, ALTER, DELETE, INDEX, INSERT, SELECT`
- ```
  GRANT privilege-type
  ON { table-Name | view-Name }
  TO grantees
  ```
- Entziehen von Zugriffsrechten mit `REVOKE`
- Teilweise unterschiedliche Konzepte in versch. DBMS
- Privilege-types in Oracle
  - ALL – erlaubt alle Operationen
  - DELETE, INSERT, SELECT, UPDATE
  ```

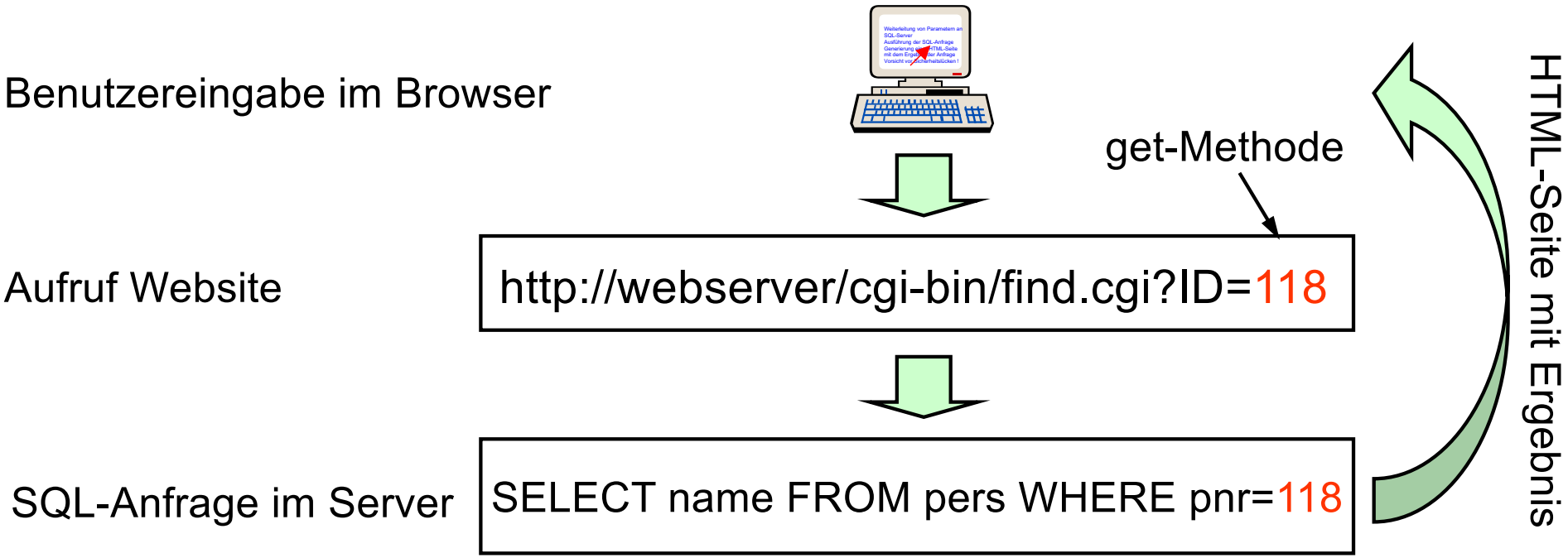
Grant und Revoke

- Zugriffsrechte vergeben
  - `GRANT INSERT, DELETE, SELECT, UPDATE ON pers TO dbsys01;`
  - `GRANT UPDATE(name) ON pers TO dbsys01;`
  - `GRANT SELECT ON sequencename TO dbsys01;`
- Zugriffsrechte vergeben und Erlaubnis Rechte weiter zu geben
  - `GRANT INSERT, DELETE, UPDATE, SELECT ON pers TO dbsys01 WITH GRANT OPTION;`
- Zugriffsrechte entziehen
  - `REVOKE INSERT, DELETE, UPDATE ON pers FROM dbsys01;`

Zugriff auf andere Benutzertabellen

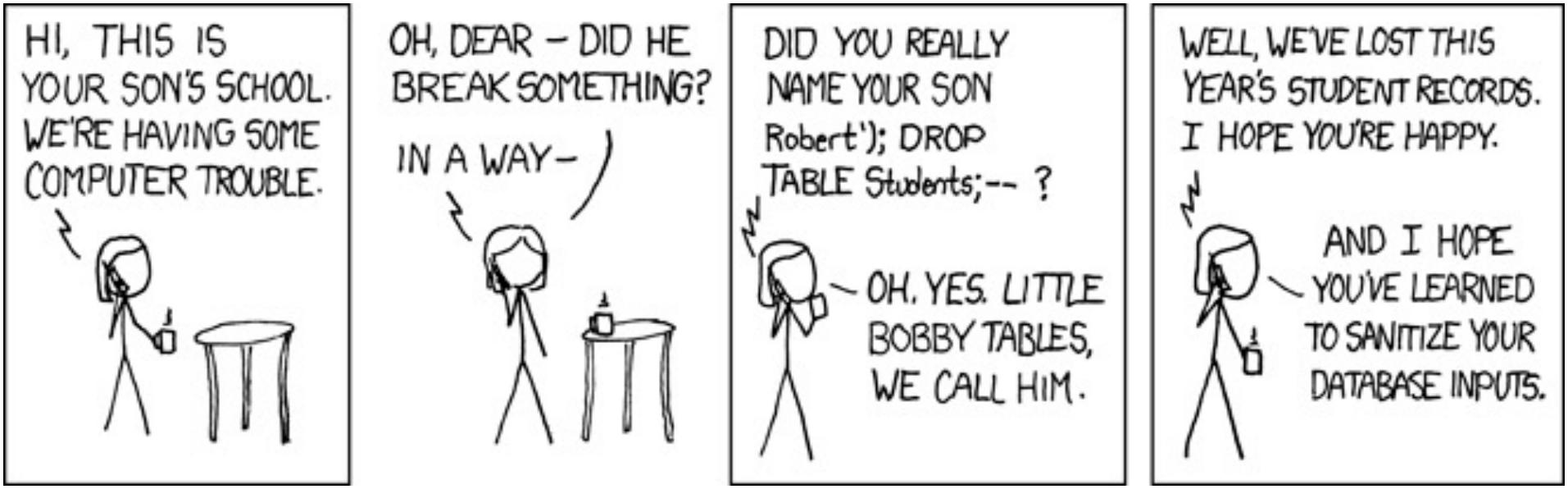
- Schema in Oracle
  - Datenbank kann mehrere Schemas haben
  - In jedem Schema sind alle Objekte, die ein bestimmter Nutzer der Datenbank erstellt hat
- Angabe des Schemanamens beim Zugriff auf Tabellen fremder Benutzer
  - ```
    SELECT name, gehalt
    FROM dbsys99.pers
    WHERE jahrg > 1980;
    ```
- Alternative
  - ```
    ALTER SESSION SET current_schema = dbsys27;
    ```

SQL-Injection in Webseiten



- SQL-Anfragen in Webseiten
  - Weiterleitung von Parametern an SQL-Server
  - Ausführung der SQL-Anfrage
  - Generierung einer HTML-Seite mit dem Ergebnis der Anfrage
  - Vorsicht vor Sicherheitslücken, z.B. durch SQL-Injection !

Don‘t trust user input



Quelle: B. Karwin: SQL Antipatterns

Gegenmaßnahmen gegen SQL-Injections

- Authentifikation, Privilegien zuweisen
  - Prinzip des kleinstmöglichen Privilegs
- Prüfung der Eingabedaten
  - Prüfung von Länge, Sonderzeichen, z.B. durch reguläre Ausdrücke
- Parametrisierte Kommandos, z.B. Prepared Statement in JDBC
  - Daten als Parameter an einen bereits kompilierten Befehl