8 SQL

- Entwicklung:
  - Vorläufer: SEQUEL (Structured English Query Language), 1973
  - SQL (Structured Query Language), 1979, Fa. ORACLE (teilweise immer noch als 'siquel' ausgesprochen)
  - SQL-86, SQL-87 (ANSI X3.135-1986, ISO 9075:1987), ca. 100 Seiten
    Enthält ein Integrity Enhancement Feature (IEF) als unverbindlichen Vorschlag im Anhang.
  - SQL-89 (ANSI X3.135-1989, ISO 9075:1989), ca. 120 Seiten
    IEF wird als verbindlich erklärt.
  - SQL-92, SQL2 (ANSI X3.135-1992, ISO 9075:1992), über 600 Seiten
    Drei Levels: Full SQL-92, Intermediate SQL-92, Entry SQL-92
  - SQL:1999, SQL3 (ISO 9075:1999), 5 Parts, ca. 2200 Seiten
    Auch objektorientierte Erweiterungen
  - SQL:2003 (ISO/IEC 9075:2003)
    SQL/XML und SQL/MED (Management of External Data)
  - SQL:2006 (ISO/IEC 9075-14:2006)
    Im Besonderen XML mit SQL
  - SQL:2008 (ISO/IEC 9075:2008), 9 Parts

- Die Datenbanksprache (Standardschnittstelle) für praktisch alle relationale Datenbanksysteme:
  - Datendefinitionen
  - Datenzugriffsberechtigungen
  - Datenauswahl, Datenabfrage
  - Datenmanipulationen

- Deskriptiv, nicht prozedural: Benutzer spezifiziert nur, was er haben will und nicht, wie es zu ermitteln ist (teilweise hat der Benutzer sehr wohl Einfluss auf die Art der Ermittlung).

- Verwendung von SQL:
  - interaktiv (iSQL)
  - in Hochsprachen (3GL)
    - Unterprogrammaufrufe
    - Embedded SQL (ESQL) mit Precompiler
    - Objektmodell für den Datenzugriff
  - in 4GL-Sprachen als Datenbankschnittstelle

- Syntax-Prinzipien:
  - nicht case-sensitive
  - nur runde Klammern: ( )
  - nur einfache Anführungszeichen bei String-Literalen: `'Beispiel'`

- Syntax-Notation:
  - BNF-ähnlich
  - Konvention:
```
xyz-commalist ::= xyz | xyz , xyz-commalist
xyz-list      ::= xyz | xyz   xyz-list
```

8.1 Datenbank LT: Lieferanten - Teile - Lieferungen

L

LNR	LNAME	RABATT	STADT
L1	Schmid	20	London
L2	Jonas	10	Paris
L3	Berger	30	Paris
L4	Klein	20	London
L5	Adam	30	Athen

T

TNR	TNAME	FARBE	PREIS	STADT
T1	Mutter	rot	12	London
T2	Bolzen	gelb	17	Paris
T3	Schraube	blau	17	Rom
T4	Schraube	rot	14	London
T5	Welle	blau	12	Paris
T6	Zahnrad	rot	19	London

LT

LNR	TNR	MENGE
L1	T1	300
L1	T2	200
L1	T3	400
L1	T4	200
L1	T5	100
L1	T6	100
L2	T1	300
L2	T2	400
L3	T2	200
L4	T2	200
L4	T4	300
L4	T5	400

MENGE > 0

8.2 Datendefinitionen

- Data Definition Language / Comands, DDL, Schema Definition Language / Commands, Schemaanweisungen

- Datenbank erstellen / löschen:
  Erfolgt im Standard über sogenannte Schema-Definitionen, die meisten SQL-Produkte arbeiten mit eigenen Konstruktionen (CREATE / DROP DATABASE, etc.)

- Tabelle erstellen:

```
CREATE TABLE base-table ( base-table-element-commalist )

base-table-element ::= column-def | table-constraint-def

column-def ::= column data-type
               [ DEFAULT { literal | niladic-function-ref | NULL } ]
               [ column-constraint-def-list ]

column-constraint-def ::= NOT NULL |
                          { UNIQUE | PRIMARY KEY } |
                          CHECK ( search-condition ) |
                          REFERENCES base-table [ ( column ) ]

table-constraint-def ::= { UNIQUE | PRIMARY KEY } ( column-commalist ) |
                         CHECK ( search-condition ) |
                         FOREIGN KEY ( column-commalist )
                                 REFERENCES base-table [ ( column-commalist ) ]
```

  - search-condition aus CHECK muss so formuliert sein, dass nur eine Zeile (in der die Änderung gemacht wird oder die eingefügt wird) isoliert untersucht werden braucht, z.B.
    `CHECK ( RABATT BETWEEN 10 AND 50 )` als column-constraint-def
    `CHECK ( STADT <> 'London' OR RABATT = 20 )` als table-constraint-def
    - Check-Bedingung ist erfüllt, wenn die search-condition TRUE oder NULL (nicht FALSE) liefert
    - Constraints, die mehrere Zeilen betreffen (aus einer oder mehreren Tabellen), können nicht deklarativ definiert werden (siehe Triggers)

  - PRIMARY KEY: Änderung eines Primärschlüsselwertes wird nicht verhindert (außer wenn dadurch eine Verletzung der Referentiellen Integrität erfolgen würde)

- Zusammenfassung Constraints:
  - Unterscheide zwischen Column- und Table-Constraints
  - NOT NULL
    Verwendung bei Fremdsschlüsseln beachten
  - PRIMARY KEY
  - FOREIGN KEY
    Referential Actions
    `ON { DELETE | UPDATE } { NO ACTION | SET NULL | SET DEFAULT | CASCADE }`
  - UNIQUE
    Verwendung bei Fremdschlüsseln beachten
  - CHECK
  - DEFAULT-Clause (kein Constraint im engeren Sinn)

- Datentypen (data-type):

`INTEGER`	Ganze Zahl
`DECIMAL[(precision[,scale])]`	Festkommazahl
`NUMERIC[(precision[,scale])]`	Festkommazahl precision=Gesamtanzahl der Stellen (ohne Dezimalpunkt) scale=Anzahl der Nachkommastellen
`FLOAT[(precision)]`	Gleitkommazahl
`REAL`	Gleitkommazahl
`DOUBLE [PRECISION]`	Gleitkommazahl
`CHARACTER[(length)]`	Zeichenkette fester Länge
`VARCHAR[(length)]`	Zeichenkette variabler Länge

`DATE`	Datum, Format kann angegeben werden (z.B. 24.12.1987 = tt.mm.jjjj), Funktion `current_date`
`TIME`	Zeit, Genauigkeit und Format können angegeben werden (z.B. 13:46 = hh:mm), auch mit Zeitzone, Funktion `current_time`
`TIMESTAMP`	Zeitpunkt, Genauigkeit und Format können angegeben werden (z.B. 24.12.1987 13:46:25 = tt.mm.jjj hh:mm:ss), auch mit Zeitzone, Funktion `current_timestamp`
`INTERVAL`	Zeitintervall, Genauigkeit und Format können angegeben werden (z.B. 17:23:45 = tt:hh:mm)
`LOGICAL`	true / false (optional)
`BLOB[(n)]`	binary large object, Binärdaten (maximal n Bytes Länge)
`CLOB[(n)]`	character large object, Zeichenkette (maximal n Bytes Länge)

  - Teilweise große Unterschiede und Erweiterungen bei den verschiedenen Datenbankprodukten

- Tabelle modifizieren:
```
ALTER TABLE base-table
      [ ADD COLUMN ... | DROP COLUMN ... | { ALTER | MODIFY } COLUMN ... ]
```

- Tabelle löschen:
```
DROP TABLE base-table
```

8.3 Indizes

- Index erstellen:
```
CREATE [ UNIQUE ] INDEX index ON table ( index-specification-commalist )

index-specification ::= column [ ASC | DESC ]
```

- Achtung: keinen Index als Beginnteil eines anderen Index definieren (unnötig)

- Index löschen:
```
DROP INDEX index
```

8.4 Datenzugriffsberechtigungen

- Data Control Language / Comands, DCL

- Der Benutzer, der eine Tabelle erstellt (CREATE TABLE), ist der Eigentümer dieser Tabelle und hat alle Rechte (SELECT, INSERT, UPDATE, DELETE) mit dieser Tabelle

- Der Eigentümer einer Tabelle kann Berechtigungen an andere Benutzer weitergeben (GRANT)

- Zugriffsberechtigungen erteilen:
```
GRANT { ALL [ PRIVILEGES ] | operation-commalist }
      ON table TO grantee-commalist [ WITH GRANT OPTION ]

operation ::= SELECT | INSERT | DELETE | { UPDATE [ ( column-commalist ) ] }

grantee ::= PUBLIC | user
```

  - table: base-table oder viewed-table
  - WITH GRANT OPTION: die erteilten Rechte dürfen an andere Benutzer weitergegeben werden (grundsätzlich kann ein Benutzer nur solche Berechtigungen weitergeben, die er selber hat)

- Zugriffsberechtigungen entziehen:
```
REVOKE { ALL [ PRIVILEGES ] | operation-commalist }
       ON table FROM grantee-commalist
```

  - Dem Eigentümer einer Tabelle können keine Berechtigungen entzogen werden

8.5 Datenauswahl - SELECT

- Data Query Language / Commands, DQL

- Abgeschlossenheit (Closure):
  Abfrageergebnis aus Tabellen (Eingabe) ist wieder eine Tabelle (Ausgabe).
  Mindestens eine Spalte, gegebenenfalls keine Zeile (wenn Abfragebedingung für alle Zeilen falsch ist)

- SELECT-Anweisung (`select-statement`) – prinzipieller Aufbau:

```
SELECT [ ALL | DISTINCT ]
       { * | expression-commalist }   = Projektion
       FROM table-reference-commalist
       [ WHERE search-condition ]     = Restriktion
       [ GROUP BY column-reference-commalist [ HAVING search-condition ] ]
       [ ORDER BY order-specification-commalist ]

table-reference ::= table [ alias ]

column-reference ::= [ { table | alias } . ] column

order-specification ::= { column-reference | integer } [ ASC | DESC ]
```

8.5.1 Projektion

- Auswahl von Spalten aus einer Tabelle

S1	S2	S2	S4	S5

- Alle Spalten, die nach dem SELECT in der expression-commalist angegeben sind

- Alle Spalten: *

- expression:
  - Operatoren: `+ | - | * | / | ( | ) | ** | mod | ||`
  - Operanden: `column-reference | literal | aggregate-function | parameter`
  - Funktionen (z.B. Arithmetische Funktionen, Zeichenkettenfunktionen) stehen zur Verfügung
  - Zuweisung eines Spaltennamens / -alias für die Ergebnistabelle: `expression [ AS ] column`
  - Wenn ein Operand der expression den Wert NULL hat, so ist auch das Ergebnis der expression NULL
  - In neueren Implementierungen kann als Operand auch `(select-statement)` angegeben werden, dieses muss eine skalare Tabelle (eine Zeile, eine Spalte) liefern (siehe Sub-Selects in der Projektion)

- Alle Tabellen nacheinander anzeigen
```
select * from l;
select * from t;
select * from lt;
```
  - falsch: `select * from l,t,lt;`

- Welche Namen haben die Teile?
```
select tname from t;
```

- Teile (TNR, TNAME) mit verdoppeltem Preis anzeigen
```
select tnr,tname,preis*2 from t;
```

- Teile (TNR, TNAME) mit Quadratwurzel des Preises (auf zwei Nachkommastellen gerundet) anzeigen
```
select tnr,tname,round(sqrt(preis),2) from t;
select tnr,tname,cast(round(sqrt(preis),2) as decimal(4,2)) from t;
```

- Teile und deren Farben anzeigen, in der Form: MUTTER in rot
```
select upper(tname)||' in '||farbe from t;
select rtrim(upper(tname))||' in '||farbe from t;
```

- Teile (TNR, TNAME) mit verdoppeltem Preis anzeigen (Spaltenname des doppleten Preises: preis_mal_2)
```
select tnr,tname,preis*2 as preis_mal_2 from t;
select tnr,tname,preis*2    preis_mal_2 from t;
```

- Was liefern die folgenden Anweisungen?
```
select 27 'immer 27' from t;
select 27 immer 27   from t;
select 27 immer, 27  from t;
```

- Funktionen
  - Mathematische Funktionen, Zahlen-Funktionen
  - Zeichenketten-Funktionen
  - Datum- und Zeit-Funktionen
  - Typ-Konvertierungs-Funktionen
  - System-Funktionen
  - Sonstige Funktionen

8.5.2 DISTINCT

- Mehrfache Zeilen mit gleichem Inhalt (Duplikatzeilen) werden eleminiert

- Welche verschiedenen Namen haben die Teile?
```
select distinct tname from t;
```

8.5.3 Restriktion (Selektion)

- Auswahl von Zeilen aus einer Tabelle

S1	S2	S2	S4	S5

- Alle Zeilen, bei denen in der WHERE-Klausel die search-condition wahr ist
- search-condition:
  - Operatoren: `OR | AND | NOT | ( | )`
  - Operanden (simple predicate):
    - `expression comparison expression`
    - `expression [ NOT ] BETWEEN expression AND expression`)
      BETWEEN: x BETWEEN y AND z ist gleichbedeutend mit y<=x AND x<=z) 'Syntaktischer Zucker'
    - `expression [ NOT ] IN ( literal-commalist )`) oder
      IN: x IN (a,b,...,z) ist gleichbedeutend mit x=a OR x=b OR ... OR x=z) 'Syntactic Sugar'
    - `column-reference [ NOT ] LIKE character-literal [ ESCAPE escape-character ]`
      character-literal: % beliebige Zeichenkette, _ ein beliebiges Zeichen
      (in Dialekten auch andere Zeichen, z.B. in ACCESS die üblichen * und ?)
      Immer ein Wildcard-Zeichen im character-literal, sonst = (Aufwand!)
    - `column-reference IS [ NOT ] NULL`
    - `comparison ::= = | <> | < | > | <= | >=`
  - Wenn ein Operand (expression) des predicates NULL ist, hat (außer bei NOT, AND, OR) das predicate auch den Wert NULL

- Welche Lieferanten sind aus Paris?
```
select * from l where stadt='Paris';
```

- Welche Lieferanten (LNR, LNAME, RABATT) sind aus Paris?
```
select lnr,lname,rabatt from l where stadt='Paris';
```

- Welche roten Teile haben einen Preis zwischen 13 und 23?
  ```
  select tnr from t where farbe='rot' and preis between 13 and 23;
  ```
  - Kurzform für
    ```
    select tnr from t where farbe='rot' and preis >= 13 and preis <= 23;
    ```

- Welche Teile zu einem Preis von 12 oder 14 sind nicht aus London oder Rom?
  ```
  select tnr from t where preis in (12, 14) and stadt not in ('London', 'Rom');
  ```
  - Kurzform für
    ```
    select tnr from t
    where (preis=12 or preis=14) and not (stadt='London' or stadt='Rom');
    select tnr from t
    where (preis=12 or preis=14) and stadt<>'London' and stadt<>'Rom';
    ```

- Alle Lieferanten (LNR), die das Teil T1 oder das Teil T2 liefern
  ```
  select distinct lnr from lt where tnr='T1' or tnr='T2';
  select distinct lnr from lt where tnr in ('T1','T2');
  ```

- Welche Teile haben ein 'e' im Namen und sind aus einer Stadt mit 3 Zeichen?
  ```
  select * from t where tname like '%e%' and stadt like '___';
  ```

- Entwertung von Wildcard-Zeichen
  ```
  insert into t values ('T8','Zang%','blau',20,'Wien');
  insert into t values ('T9','Weller''l','rot',10,'Wi_en');
  select * from t where tname like '%''%';
  select * from t where stadt like '%_%';
  select * from t where stadt like '%\_%' escape '\';
  select * from t where tname like '%%%';
  select * from t where tname like '%!%%' escape '!';
  delete from t where tnr='T8';
  delete from t where tnr='T9';
  ```

- Welche Teile haben keinen Preis?
  ```
  select * from t where preis is null;
  ```
  - falsch: `... preis=null;`

8.5.4 ORDER BY

- Die Ergebnistabelle wird sortiert zur Verfügung gestellt
- Column-reference von ORDER BY muss sich auf die Projektion beziehen und wird daher auch ein Element der expression-commalist nach SELECT sein
- Bei Verwendung von Spaltenalias kann auch ein solcher angegeben werden
- Ein integer n bezieht sich auf das n-te (mit 1 beginnend) Element der Projektion, damit kann nach dem Ergebnis beliebiger Ausdrücke sortiert werden
- NULL-Werte sind implementierungsabhängig entweder alle größer oder alle kleiner als Nicht-NULL-Werte
- Achtung: Ohne Verwendung von ORDER BY wird das Ergebnis (Tabelle ist eigentlich eine Menge) in 'irgendeiner' Reihenfolge geliefert, nicht unbedingt nach dem Primärschlüssel oder der Reihenfolge des Einfügens. Unterschied zu einer Sequentiellen Datei, wo die Reihenfolge des Schreibens auch die Reihenfolge des Lesens ist.

- Welche Namen haben die Teile (sortiert nach Namen)?
  ```
  select tname from t order by tname;
  ```

- Welche verschiedenen Namen haben die Teile (sortiert nach Namen)?
  ```
  select distinct tname from t order by tname;
  ```

- Alle Teile mit dem Quadrat des um 16 erniedrigten Preises anzeigen, sortiert in erster Linie nach diesem Ausdruck absteigend, in zweiter Linie nach dem Teilenamen
  ```
  select tnr, (preis - 16)*(preis-16),tname from t order by 2 desc, tname;
  select tnr, (preis - 16)*(preis-16) ausdruck,tname from t
  order by ausdruck desc, tname;
  ```

8.5.5 Mengenoperation Vereinigung - UNION

- Die Zeilen zweier SELECT-Ergebnisse kommen ('untereinander') in eine Ergebnistabelle:
  `SELECT ... UNION [ ALL ] SELECT ...`
  - Zu vereinigende Ergebnistabellen müssen vereinigungsverträglich sein, d.h.
    syntaktisch: gleiche Spaltenanzahl, gleiche Datendefinitionen (Typ, Länge) der Spalten
    semantisch: gleiche Bedeutung der Spalten
  - ALL: Duplikatzeilen werden nicht eliminiert
  - ORDER BY darf nur nach dem letzten SELECT angegeben werden

- Welche Lieferanten (LNR) sind aus London oder liefern das Teil T2?
```
select lnr from l where stadt='London'
union
select lnr from lt where tnr='T2';

select lnr from l where stadt='London'
union all
select lnr from lt where tnr='T2'
order by lnr;
```

- Was liefern die folgenden Anweisungen?
```
select lnr from l
union
select * from t;

select lnr,lname from l
union
select tnr,preis from t;

select lnr,lname from l
union
select tnr,tname from t
order by 2;
```

- Nicht Vereinigung statt Restriktion verwenden:
```
select * from t where farbe='blau'
union
select * from t where stadt='Paris';

select * from t where farbe='blau' or stadt='Paris';
```

8.5.6 Cross-Join

- Kreuz-Verbund, (Karthesisches) Produkt

- Die Zeilen zweier Tabellen kommen ('nebeneinander') in eine Ergebnistabelle:
  Jede Zeile der einen Tabelle wird mit jeder Zeile der anderen Tabelle zusammengehängt (konkateniert), alle kommen in die Ergebnistabelle

- Übung: Wieviele Spalten und Zeilen (auf die Spalten- und Zeilenanzahl der Operanden-Tabellen bezogen) hat die Ergebnistabelle eines Cross-Joins?

- Beispiel
```
select * from lt, l;
select * from lt cross join l;
```

8.5.7 Equi-Join

- Gleichheits-Verbund

- Analog Cross-Join, jedoch kommen nur jene Zeilen in die Ergebnistabelle, bei denen eine Gleichheitsbedingung (Spalteninhalte der zwei Operanden-Tabellen) wahr ist.
  Bedingung wird formuliert in
  - der Restriktion (SQL-89) – eigentlich CROSS-Join + Restriktion
  - der FROM-Klausel (SQL-92) – eigene JOIN-Operation

- Häufigster Anwendungsfall: Zeilen, die einen Fremdschlüssel enthalten, werden jeweils mit der Zeile, die den entsprechenden Primärschlüsselwert hat, verbunden

- Spalten-Qualifikation mit Tabellennamen oder Tabellenalias meist notwendig (da Spaltennamen von Primär- und Fremdschlüssel üblicherweise gleich sind)
  Spalte einer Tabelle: `table-reference.column-reference`
  Alle Spalten einer Tabelle: `table-reference.*`

- Lieferungen mit Mengen über 100 samt Lieferantennamen anzeigen (LNR, LNAME, TNR, MENGE)
```
select l.lnr, l.lname, lt.tnr, lt.menge
from lt, l
where lt.lnr = l.lnr and menge>100;

select l.lnr, l.lname, lt.tnr, lt.menge
from lt join l on lt.lnr=l.lnr
where menge>100;
```

- Wo (STADT) ist Lieferant und Teil aus der derselben Stadt?
```
select distinct l.stadt
from lt, l, t
where lt.lnr=l.lnr and lt.tnr=t.tnr and t.stadt=l.stadt;

select distinct l.stadt
from (lt join l on lt.lnr=l.lnr) join t on lt.tnr=t.tnr
where t.stadt=l.stadt;
```
  - Klammerung nicht notwendig, erleichtert Lesbarkeit

  - möglich, aber strukturell falsch:
```
select distinct l.stadt
from (lt join l on lt.lnr=l.lnr) join t on lt.tnr=t.tnr and t.stadt=l.stadt;
```

- Equi-Join über mehrere Spalten (z.B. bei zusammengesetztem Fremdschlüssel)
```
select *
from t1,t2
where t1.t1c1=t2.t1c1 and t1.t1c2=t2.t1c2;

select *
from t1 join t2 on t1.t1c1=t2.t1c1 and t1.t1c2=t2.t1c2;
```

- Verwendung von (sinnvollen) Tabellenalias erspart bei langen Tabellennamen Schreibarbeit

8.5.8 Self-Join

- Alle Lieferanten (LNR), die das Teil T1 und das Teil T2 liefern (mit SQL-89-Join)
```
select lt1.lnr
from lt lt1, lt lt2
where lt1.lnr=lt2.lnr and lt1.tnr='T1' and lt2.tnr='T2';
```
  - Verwendung von Tabellenalias notwendig
  - falsch:
```
select distinct lnr from lt where tnr='T1' and tnr='T2';
```

- Alle Lieferanten (LNR,LNAME), die das Teil T1 und das Teil T2 liefern (mit SQL-92-Join)
```
select l.lnr, l.lname
from (lt lt1 join lt lt2 on lt1.lnr=lt2.lnr) join l on lt1.lnr=l.lnr
where lt1.tnr='T1' and lt2.tnr='T2';
```

- Alle Paare von Lieferanten anzeigen, die aus derselben Stadt sind (nicht Paare mit denselben Lieferanten oder nur zu einem anderen Paar vertauschten Lieferanten)
```
select l1.lnr, l2.lnr from l l1, l l2 where l1.stadt=l2.stadt and l1.lnr<l2.lnr;
```

- ORACLE-Demodatenbank: Nummern und Namen der Mitarbeiter mit den Namen der jeweilige Vorgesetzten, sortiert nach Mitarbeitername (1:n – rekursiv)
```
select e.empno,e.ename,m.ename
from emp e join emp m on e.mgr=m.empno -- nicht: m.mgr=e.empno
order by e.ename;
```

mit dem 'Big Boss'
```
select e.empno,e.ename,m.ename from emp e join emp m on e.mgr=m.empno
union
select e.empno,e.ename,'' from emp e where e.mgr is null
order by e.ename;
```
- siehe auch Outer-Join

- SCHULUNGSFIRMA:
  Welche Kurse (Bezeichnungen) setzt der Kurs 'Komposition' unmittelbar voraus? (m:n – rekursiv)
```
select kv.bezeichn
from setztvor s, kurs k, kurs kv
where s.knr=k.knr and s.knrvor=kv.knr and -- nicht symmetrisch!
      k.bezeichn='Komposition';
```

- SCHULUNGSFIRMA:
  Welche Kurse (Bezeichnungen) setzt der Kurs 'Komposition' alle voraus? siehe Stored Routines

- Ortsentfernungen: Ort (ONr, OName), Entfernt (ONr1, ONr2, km)
```
select o1.oname, o2.oname, km
from entfernt e, ort o1, ort o2
where e.onr1=o1.onr and e.onr2=o2.onr; -- symmetrisch!
```

- Weiteres Anwendungsgebiet: Speicherung und Analyse von Graphen
  (ungerichtet / gerichtet, ohne / mit gewichteten Kanten, ohne / mit Mehrfachkanten, etc.)

8.5.9 Sub-Selects in der Search-Condition

- Unterabfragen, Sub-Queries (non-correlated, correlated)

- search-condition:
  - predicate with subquery (geschachteltes select-statement, sub-select):
    - `expression comparison ( select-statement )` 1)
      Skalare Unterabfrage mit Vergleichsprädikat
    - `expression [ NOT ] IN ( select-statement )` 2)
    - `expression comparison {ALL | ANY | SOME} ( select-statement )` 2)
      Unterabfragen mit quantifiziertem Vergleichsprädikat
      ANY und SOME sind gleichbedeutend
    - `EXISTS ( select-statement )` 3)
    - Notwendige Eigenschaften der Ergebnistabelle des select-statements:
      1) Spaltenanzahl: 1, Zeilenanzahl: 1 (Skalares Ergebnis, Skalare Tabelle)
      2) Spaltenanzahl: 1, Zeilenanzahl: n
      3) Spaltenanzahl: m, Zeilenanzahl: n

- Welche Teile sind teurer als das Teil T4?
```
select * from t where preis > (select preis from t where tnr='T4');
```

- Welche Lieferanten (LNR) liefern ein Teil, das auch von L2 geliefert wird?
```
select distinct lnr
from lt
where tnr in (select tnr from lt where lnr='L2') and (lnr<>'L2');
```

- Alle Lieferanten (LNR,LNAME), die das Teil T1 und das Teil T2 liefern (Alternative)
```
select lnr,lname
from l
where lnr in (select lnr from lt where tnr='T1') and
      lnr in (select lnr from lt where tnr='T2');
```

- Alle Lieferanten (LNR,LNAME), die entweder das Teil T1 oder das Teil T2 (nicht beide) liefern
```
select lnr,lname
from l
where lnr in     (select lnr from lt where tnr='T1') and
      lnr not in (select lnr from lt where tnr='T2') or
      lnr not in (select lnr from lt where tnr='T1') and
      lnr in     (select lnr from lt where tnr='T2');
```

- Welche Lieferanten (LNR) liefern ein Teil in einer Menge, die kleiner ist als jede / eine Liefermenge des Lieferanten L4?
```
select distinct lnr
from lt
where menge < all (select menge from lt where lnr='L4');

select distinct lnr
from lt
where menge < some (select menge from lt where lnr='L4');
```

- Welche Lieferanten (LNR,LNAME) haben Teile geliefert? (mehrere Varianten)
```
select distinct l.lnr, lname from lt, l where l.lnr=lt.lnr;
select lnr, lname from l where lnr in (select lnr from lt);
select lnr, lname from l where exists (select * from lt where lt.lnr=l.lnr);
```
  - EXISTS: In Projektion immer *, da Spalten nicht relevant

- Unterschied:

	References	Processing	Remark
- Non-Correlated Sub-Query	only local references	inside-out	
- Correlated Sub-Query	also non-local references	outside-in, 'nested loop'	always for EXISTS

- Welche Lieferanten (LNR,LNAME) haben keine Teile geliefert?
```
select lnr, lname from l where lnr not in (select lnr from lt);
select lnr, lname from l where not exists (select * from lt where lt.lnr=l.lnr);
```
  - wenn L einen zusammengesetzten Primärschlüssel hätte (und damit LT einen zusammengesetzten Fremdschlüssel), wäre die Variante mit EXISTS zu bevorzugen

- Welche Lieferanten (LNR) haben keine roten Teile geliefert?
```
select lnr from l
where not exists (select * from lt where lt.lnr=l.lnr and lt.tnr in
   (select tnr from t where farbe='rot'));

select lnr from l
where lnr not in (select lnr from lt where tnr in
   (select tnr from t where farbe='rot'));

select lnr from l where not exists
   (select * from lt, t where lt.tnr=t.tnr and lt.lnr=l.lnr and farbe='rot');

select lnr from l where lnr not in
   (select lnr from lt, t where lt.tnr=t.tnr and farbe='rot');
```

  - falsch (welche Fragestellung beantwortet diese Anweisung?):
```
select distinct lnr from lt, t where lt.tnr=t.tnr and farbe<>'rot';
```

- Welche Lieferanten (LNR) liefern ausschließlich gelbe Teile?
```
select lnr from lt
where lnr not in (select lnr from lt join t on lt.tnr=t.tnr where farbe<>'gelb');
```

- Welche Lieferanten (LNR, LNAME) liefern ausschließlich gelbe Teile?
```
select lnr,lname from l
where lnr not in (select lnr from lt join t on lt.tnr=t.tnr where farbe<>'gelb')
      and lnr in (select lnr from lt); -- sonst auch Lieferanten ohne Lieferungen
```

- Welche Lieferanten (LNR, LNAME) haben welche Teile (TNR) noch nicht geliefert?
```
select lnr,lname,tnr from l cross join t
where not exists (select * from lt where lt.lnr=l.lnr and lt.tnr=t.tnr);
```

8.5.10 All-Quantor

- Welche Lieferanten (alle Spalten) haben alle Teile geliefert? (kein Teil nicht geliefert)
```
select * from l where not exists
   (select * from t where not exists
      (select * from lt where lt.tnr=t.tnr and lt.lnr=l.lnr));
```

- Welche Lieferanten (LNR) haben alle Teile geliefert?
```
select distinct lnr from lt where not exists
   (select * from t where not exists
      (select * from lt lt1 where lt1.tnr=t.tnr and lt1.lnr=lt.lnr));
```

- Hinweis: Kann nicht mit zweifachem NOT IN gebildet werden
```
select * from l where lnr not in
   (select lnr from t ???
```
Inneres Sub-Select in diesem Fall mit NOT IN möglich
```
select * from l where not exists
   (select * from t where tnr not in
      (select tnr from lt where lt.lnr=l.lnr))
```

8.5.11 Aggregatfunktionen

- aggregate-function:
```
COUNT(*)
{ COUNT | SUM | AVG | MIN | MAX } ( expression )
{ COUNT | SUM | AVG | MIN | MAX } ( DISTINCT column-reference )
```
  - Aus der Tabelle (COUNT) oder einer Spalte der Tabelle (SUM, AVG, MIN, MAX) wird ein Wert ermittelt
  - NULL-Werte werden vor Anwendung der Funktion eliminiert, außer bei COUNT(*), wo sie mitgezählt werden
  - Wenn die Tabelle leer ist, liefert COUNT 0 zurück, alle anderen Funktionen NULL

- Von wievielen Lieferanten wird das Teil T2 geliefert, wie groß sind Summe, Maximum, Minimum und Durchschnitt der Mengen?
```
select count(*), sum(menge), max(menge), min(menge), avg(menge)
from lt
where tnr='T2';
```

- Aus wieviel verschiedenen Städten sind die Lieferanten?
```
select count(distinct stadt) from l;
```

- Welche / Wieviele Lieferanten liefern mehr als eine Teilenummer?
```
select * from l
where (select count(*) from lt where lt.lnr=l.lnr)>1;

select count(*) from l
where (select count(*) from lt where lt.lnr=l.lnr)>1;
```

- Alle Lieferanten (LNR,LNAME), die entweder das Teil T1 oder das Teil T2 (nicht beide) liefern (Alternative)
```
select lnr,lname from l
where (select count(*) from lt where lt.lnr=l.lnr and tnr in ('T1','T2'))=1;
```

- Welche Lieferanten (alle Spalten) haben alle Teile geliefert? (Alternative)
```
select * from l
where (select count(*) from lt where lt.lnr=l.lnr)=(select count(*) from t);
```
  - so nur möglich, weil LNr und TNr in LT den PK bilden!

- Der dritt-kleinste / n-kleinste Primärschlüsselwert ist zu ermitteln
```
select tnr
from t
where (select count(*) from t t1 where t1.tnr<=t.tnr) = 3;
```

- Die drei / n teuersten Teile sind zu ermitteln (ohne: TOP n, FIRST n, LIMIT n, ROWNUM<=n, …)
```
select tnr, tname, preis
from t
where (select count(*) from t t1 where t1.preis>=t.preis) <= 3;
```

8.5.12 Gruppierung

- GROUP BY:
  - Aus jenen Zeilen der Tabelle, die in der (den) bei GROUP BY angegebenen Spalte(n) denselben Wert haben, wird eine Zeile gebildet
  - Column-reference von GROUP BY muss sinnvollerweise auch in der Projektion enthalten sein
  - Die anderen Ausdrücke der Projektion müssen aggregate-functions sein
  - Aggregate-functions bilden in Zusammenhang mit GROUP BY gruppenweise Ergebnisse
  - HAVING: analog WHERE-Klausel, allerdings für Gruppen (in der search-condition muss eine aggregate-functions vorkommen, sonst kann die Bedingung in der WHERE-Klausel effizienter formuliert werden)

- Anzahl der Teile und Durchschnittspreis pro Farbe anzeigen
```
select farbe, count(*) anzahl, avg(preis) durchschnitt from t group by farbe;
```

- nicht
```
select lnr from lt group by lnr;
```
statt
```
select distinct lnr from lt;
```

- In welchen Städten sind mindestens 2 Lieferanten? (Stadt, Anzahl der Lieferanten, durchschnittlicher Rabattwert und Anzahl der verschiedenen Rabattwerte)
```
select stadt, count(*) anzahl, avg(rabatt) durchschnitt,
       count(distinct rabatt) anz_versch
from l
group by stadt
having count(*)>1
order by stadt;
```

- Alle Lieferanten (LNR) mit ihren Umsätzen anzeigen.
```
select lt.lnr, sum(lt.menge * t.preis) umsatz
from lt join t on lt.tnr=t.tnr
group by lnr;
```

- Alle Lieferanten (LNR) mit ihren Umsätzen anzeigen, auch die ohne Umsatz
```
select lt.lnr, sum(lt.menge * t.preis) umsatz
from lt join t on lt.tnr=t.tnr
group by lnr
union
select lnr,0
from l
where lnr not in (select lnr from lt);
```
  - siehe auch Outer-Join

- Alle Lieferanten (LNR, LNAME), die mehr als zwei verschiedene Nummern roter Teile liefern, mit den (unter Berücksichtigung des Rabattsatzes gebildeten) Umsätzen bei diesen Teilen, nach Umsätzen absteigend sortiert, anzeigen
```
select lt.lnr, lname, sum(menge*preis*(1-rabatt/100)) Umsatz
from (lt join t on lt.tnr=t.tnr) join l on lt.lnr=l.lnr
where farbe='rot'
group by lt.lnr, lname
having count(*)>2
order by umsatz desc;
```

- Wieviele (nicht welche) Lieferanten liefern mehr als eine Teilenummer? (Alternative)
```
select count(*)
from l
where lnr in (select lnr from lt group by lnr having count(*)>1);
```
  - falsch, da count(*) pro Gruppe ausgeführt wird
```
select count(*) from lt group by lnr having count(*)>1;
```

8.5.13 Theta-Join

- θ-Join

- Bis jetzt EQUI-JOIN, d.h. Vergleichsoperator in der Join-Bedingung ist 'gleich'. Beim THETA-JOIN ist der Vergleichsoperator in der Joinbedingung nicht 'gleich'.

- ORACLE-Demodatenbank: Name und Gehalt der Mitarbeiter samt Gehaltsstufe anzeigen
```
select ename,sal,grade
from emp join salgrade on sal between losal and hisal;

select ename,sal,grade
from emp,salgrade
where sal between losal and hisal;
```

- siehe auch Self-Join

8.5.14 Outer-Join

- Bisher Inner-Join (Innerer Verbund):
  ```
  ... FROM table-ref [INNER] JOIN table-ref ON cond-expr ...
  ```

- Es müssen nicht alle Zeilen der Tabellen im Ergebnis sein (es fehlen jene, für die die cond-expr nie zutrifft)

- Alle Lieferanten mit jeweils ihren gelieferten Teilen
  ```
  select l.lnr,lname,tnr,menge from l join lt on l.lnr=lt.lnr;
  ```
  - Lieferanten, die nichts geliefert haben, scheinen im Ergebnis nicht auf

- Outer Join (Äußerer Verbund, Verlustfreier Verbund):
  ```
  ... FROM table-ref {LEFT | RIGHT | FULL} [OUTER] JOIN table-ref ON cond-expr ...
  ```

- Es sind alle Zeilen der linken (LEFT), der rechten (RIGHT) oder beider (FULL) Tabelle(n) im Ergebnis, der fehlende Teil der Zeilen wird mit NULL aufgefüllt

- Alle Lieferanten mit jeweils ihren gelieferten Teilen; auch Lieferanten, die nichts geliefert haben, sollen aufscheinen
  ```
  select * from l left join lt on l.lnr=lt.lnr;
  ```
  ohne Outer-Join:
  ```
  select * from l join lt on l.lnr=lt.lnr
  union
  select l.*,NULL,NULL,NULL from l where lnr not in (select lnr from lt);
  ```

- Alle Lieferanten (LNr, LName) mit Anzahl der verschiedenen Teile, die sie liefern (auch Lieferanten, die keine Teile liefern - dort Anzahl gleich 0)
  ```
  select l.lnr, lname, count(lt.lnr) -- Spalte aus lt, nur diese ist NULL
                                     -- count(*) würde 1 statt 0 liefert
  from l left join lt on l.lnr=lt.lnr
  group by l.lnr, lname; -- nicht lt.lnr
  ```

  ohne Outer-Join:
  ```
  select l.lnr, lname, count(*)
  from l join lt on l.lnr=lt.lnr
  group by l.lnr, lname
  union
  select lnr, lname, 0 from l where lnr not in (select lnr from lt);
  ```

- Alle Lieferanten (LNR) mit ihren Umsätzen anzeigen, auch die ohne Umsatz (Alternative)
  ```
  select l.lnr, coalesce(sum(preis*menge),0) umsatz
  from (lt right join l on lt.lnr=l.lnr) left join t on lt.tnr=t.tnr
  group by l.lnr; -- nicht lt.lnr

  select l.lnr, coalesce(sum(preis*menge),0) umsatz
  from (lt join t on lt.tnr = t.tnr) right join l on lt.lnr = l.lnr
  group by l.lnr; -- nicht lt.lnr
  ```

- Welche Lieferanten (LNR,LNAME) haben keine Teile geliefert? (Alternative)
  ```
  select l.lnr, l.lname from l left join lt on l.lnr=lt.lnr where lt.lnr is null;
  ```

- ORACLE-Demodatenbank: Nummern und Namen der Mitarbeiter mit den Namen der jeweilige Vorgesetzten, sortiert nach Mitarbeitername, mit dem 'Big Boss' (Alternative)
  ```
  select e.empno,e.ename,coalesce(m.ename,'') ename_vorgesetzter
  from emp e left join emp m on e.mgr=m.empno
  order by e.ename;
  ```

8.5.15 Natural-Join

- Natürlicher Verbund

- Equi-Join mit Eliminierung jeweils einer der gemeinsamen (doppelten) Spalten

- Variante 1:
  `... FROM table-ref JOIN table-ref USING (column-commalist) ...`
  - wenn table-refs gleich A und B sind und column-commalist aus C1, C2, …, Cn besteht, dann
    `... FROM A JOIN B ON A.C1=B.C1 AND A.C2=B.C2 AND ... AND A.Cn=B.Cn ...`
  - Jede der Spalten C1, C2, …, Cn tritt nur einmal im Resultat auf (Werte ohnehin immer gleich)
  - Die gemeinsamen Spalten scheinen als erste ('links') auf
  - Auch mit Outer-Join kombinierbar

- Variante 2:
  `... FROM table-ref NATURAL JOIN table-ref ...`
  - Wie Variante 1, die column-commalist wird gebildet aus den gemeinsamen Spalten (gleicher Name) der beiden table-refs
  - Wenn es keine gemeinsamen Spalten gibt, dann Degenerierung auf Cross-Join

8.5.16 Semi-Join

- Projektion der Spalten einer Tabelle aus einem Join-Ergebnis:
```
select A.* from A join B on A.Nr=B.Nr;
select B.* from A join B on A.Nr=B.Nr;
```

8.5.17 Mengenoperationen Durchschnitt und Differenz - INTERSECT, EXCEPT

- Tabellen müssen 'vereinigungsverträglich' sein

- Alle Lieferanten (LNR), die das Teil T1 und das Teil T2 liefern (Alternative)
```
select lnr from lt where tnr='T1'
intersect
select lnr from lt where tnr='T2';
```

- Alle Lieferanten (LNR), die das Teil T2 aber nicht das Teil T1 liefern (Alternative)
```
select lnr from lt where tnr='T2'
except
select lnr from lt where tnr='T1';
```

- Welche Lieferanten (LNR) liefern ausschließlich gelbe Teile? (Alternative)
```
select lnr from lt
except
select lnr from lt join t on lt.tnr=t.tnr where farbe<>'gelb';
```

8.5.18 Sub-Selects in der Projektion

- Sub-Select (wie üblich) in Klammern, muss Skalares Ergebnis liefern (Tabelle mit einer Zeile und einer Spalte)

- Beispiel
```
select lnr,(select lname from l where l.lnr=lt.lnr) lname, tnr, menge
from lt;
```
  - nicht sinnvoll, besser Join:
```
select lnr,tnr,menge,
       (select lname from l where l.lnr=lt.lnr) lname,
       (select stadt from l where l.lnr=lt.lnr) stadt,
       (select tname from t where t.tnr=lt.tnr) tname
from lt;
```

- Alle Lieferanten (LNr, LName) mit Anzahl der verschiedenen Teile, die sie liefern (auch Lieferanten, die keine Teile liefern - dort Anzahl gleich 0) (Alternative)
```
select lnr,lname,(select count(*) from lt where lt.lnr=l.lnr) anzahl
from l;
```

- Alle Lieferanten (LNR) mit ihren Umsätzen anzeigen, auch die ohne Umsatz (Alternative)
```
select lnr, (select coalesce(sum(preis*menge),0)
             from lt join t on lt.tnr=t.tnr
             where lt.lnr=l.lnr)
from l;
```

- Ortsentfernungen (Alternative)
```
select o1.oname von,o2.oname nach,
       (select km from entfernt e where e.onr1=o1.onr and e.onr2=o2.onr) km
from ort o1, ort o2
where o1.onr<o2.onr
order by von,nach;
```

8.5.19 Sub-Selects in der From-Klausel (Inline View)

- Sub-Select in der From-Klausel: (wie üblich) in Klammern und unbedingt mit Alias-Namen; kann auch als Join-Operand verwendet werden

- Beispiel
```
select *
from (select * from t where farbe='rot') t1 join
     (select tnr,menge from lt where menge<300) lt1 on t1.tnr=lt1.tnr;
```

- Wieviele (nicht welche) Lieferanten liefern mehr als eine Teilenummer? (Alternative)
```
select count(*) from (select lnr from lt group by lnr having count(*)>1) as erg;
```

8.5.20 NULL und dreiwertige Logik

- Wenn ein Operand NULL ist, dann ist die ganze expression NULL
  Ausnahme: Bei logischen Operatoren (NOT, AND, OR) gilt eine dreiwertige Logik

- Für die Operatoren NOT, AND, OR gilt folgende Verknüpfungstabelle:

	NOT	AND			OR		
		T	NULL	F	T	NULL	F
T	F	T	NULL	F	T	T	T
NULL	NULL	NULL	NULL	F	T	NULL	NULL
F	T	F	F	F	T	NULL	F

- Zeilen, für die die search-condition NULL ist, kommen nicht in das Ergebnis (NULL ist nicht TRUE)

- Abfrage auf NULL mit eigenem Prädikat (IS [NOT] NULL)

- Aggregat Functions
  - Zeilen mit NULL-Werten werden vor der Aggregation eliminiert, Ausnahme: count(*) zählt alle
  - Aggregation auf leere Tabelle liefert NULL, Ausnahme: count liefert 0

- Sortierung von NULL siehe ORDER BY

8.5.21 Zusamenfassung der Abarbeitung

- Reihenfolge der Abarbeitung:

  1) Tabellenverknüpfung (nach FROM)
  2) Zeilen auswählen gemäß search-condition (nach WHERE)
  3) Zeilen zusammenfassen gemäß column-reference-commalist (nach GROUP BY)
  4) Zeilen auswählen gemäß search-condition (nach HAVING)
  5) Spalten auswählen gemäß expression-commalist (nach SELECT)

  6) Mengenoperation(en) durchführen (UNION, INTERSECT, EXCEPT)
  7) Zeilen sortieren gemäß order-specification-commalist (nach ORDER BY)

- Merkregel: Abarbeitung erfolgt in der Reihenfolge der Notation der einzelnen Klauseln, außer der Projektion (diese wird am Schluss gemacht)

8.6 Datenmanipulationen

- Data Manipulation Language / Commands, DML, Module Language

8.6.1 INSERT

- Zeile(n) in eine Tabelle einfügen:
```
INSERT INTO table [ ( column-commalist ) ]
       { VALUES ( insert-atom-commalist ) | select-statement | DEFAULT VALUES }

insert-atom ::= literal | NULL | DEFAULT | parameter
```

- Lieferanten L6, namens Maxi aus Graz mit Rabatt 30 einfügen
```
insert into l values ('L6','Maxi',30,'Graz');
```

- Andere Reihenfolge der Spaltenwerte als im CREATE angegeben
```
insert into l (lname,lnr,stadt,rabatt) values ('Moritz','L7','Wien',15);
```

- Standardwerte für nicht angegebene Spalten
```
insert into l (lname,lnr) values ('Pauli','L8');
insert into l values ('L8','Pauli',DEFAULT,DEFAULT);
```

- Lieferungen mit einer Menge kleiner 150 in eine eigene Tabelle auslagern (mit Angabe des Zeitpunktes) und wieder zurückkopieren
```
create table lt_min (lnr char(2), tnr char(2), menge decimal(4), wann timestamp);
-- eventuell auch: wann ... default current_timestamp
insert into lt_min select lt.*,current_timestamp from lt where menge<150;
delete from lt where menge<150;
select * from lt_min;
select * from lt;
insert into lt select lnr,tnr,menge from lt_min;
drop table lt_min;
```

- Sehr schnell sehr viele Zeilen
```
create table x (x integer);
insert into x values (1);
insert into x select * from x; -- öfters!
drop table x;
```

8.6.2 UPDATE

- Zeile(n) in einer Tabelle ändern (searched UPDATE):
```
UPDATE table SET assignment-commalist [ WHERE search-condition ]

assignment ::= column = { expression | NULL | DEFAULT }

search-condition ::= siehe select-statement
```

- Alle Liefermengen um 10% erhöhen
```
update lt set menge=menge*1.1;
```
  - Achtung: keine WHERE-Klausel

- Preise der roten Teile verdoppeln
```
update t set preis=preis*2 where farbe='rot';
```

- Preise der roten Teile verdoppeln und Städte der roten Teile auf Wien setzen
```
update t set preis=preis*2, Stadt='Wien' where farbe='rot';
```

- Alle Athener Lieferanten ziehen nach Rom
```
update l set stadt='Rom' where stadt='Athen';
```

- Rabatt der Lieferanten, die ein gelbes Teil liefern, auf 50 setzen

```
update l set rabatt=50
where lnr in (select lnr from t join lt on t.tnr=lt.tnr where farbe='gelb');

update l set rabatt=50
where exists (select * from t join lt on t.tnr=lt.tnr
              where l.lnr=lt.lnr and farbe='gelb');

update l set rabatt=50
where lnr in (select lnr
              from lt where tnr in (select tnr from t where farbe='gelb'));
```

8.6.3 DELETE

- Zeile(n) aus einer Tabelle löschen (searched DELETE):

```
DELETE FROM table [ WHERE search-condition ]

search-condition::= siehe select-statement
```

- Alle Lieferungen löschen

```
delete from lt;
```

  - Achtung: keine WHERE-Klausel

- Lieferungen mit kleinerer Menge als 200 löschen

```
delete from lt where menge<200;
```

- Die Lieferungen von Londoner Lieferanten löschen

```
delete from lt where lnr in (select lnr from l where stadt='London');
```

8.7 Transaktionssteuerung

- Transaction Control Language / Commands

- Transaktion beginnt: bei erster ausführbarer SQL-Anweisung

- Transaktion normal abschließen:

```
COMMIT [ WORK ]
```

- Transaktion zurücknehmen und abschließen:

```
ROLLBACK [ WORK ]
```

8.8 Virtuelle Tabellen - VIEW

- Werden aus anderen Tabellen, als Ergebnis einer Select-Anweisung, gebildet

- Haben einen eigenen Namen (View, Pseudo-Tabelle, viewed-table zum Unterschied von base-table) und werden wie reale Tabellen benützt

- Werden nicht nur einmal statisch beim Erstellen der View gebildet, sondern sind stets am aktuellen Stand der Tabellen, von denen sie abgeleitet sind

- Umgekehrt sind Datenmanipulationen (INSERT, UPDATE, DELETE) in Views, die sich auf die Ursprungstabellen auswirken, nur eingeschränkt anwendbar (updatability)

- Es können auch Views von Views definiert werden

- Zweck:
  - Vereinfachung sich wiederholender Abfragen (Restriktion, Join, etc.)
    z.B. nur aktive Mitglieder, nicht ausgetretene, nicht verstorbene (Restriktion)
    z.B. nur weibliche Patienten in Geburtsabteilung (Restriktion)
    z.B. immer Name des Referenten bei der Kursveranstaltung notwendig (Join über PNr)
    z.B. pro Sub-Entity-Typ eine View mit allen Attributen definieren (Natural Join)

  - Datenschutzmaßnahmen
  - Änderung der Tabellenstruktur: wenn möglich View bilden, die auf die alten Strukturen abbildet
    z.B. Änderung einer Tabelle mit Person - Gehalt auf
    zwei Tabellen mit Person - Gehaltsstufe und Gehaltsstufe – Gehalt.
    Mit Join kann auf Struktur der ursprünglichen Tabelle abgebildet werden.

- View erstellen:
```
CREATE VIEW view [ ( column-commalist ) ]
            AS select-statement [ WITH CHECK OPTION ]
```
  - column-commalist: Spaltennamen der View. Müssen verwendet werden, wenn die expression-commalist des select-statements Ausdrücke oder gleiche Spaltennamen enthält
  - select-statement: ohne ORDER BY
  - WITH CHECK OPTION: Änderungen in der View dürfen der search-condition der WHERE-Klausel nicht widersprechen

- View löschen:
```
DROP VIEW view
```

- Implementierung: Select-Anweisung jedes Mal ausführen oder transparentes Abspeichern des Select-Ergebnisses (Materialized View)

8.9 Zeilenweise Verarbeitung - CURSOR

- Verarbeitung von Einzelzeilen und Zuweisung an Programmvariable (parameter) kann in Programmiersprachen (Hochsprachen oder 4GL) auf zwei Arten erfolgen:
  - Singleton Select (Ergebnistabelle darf nur eine Zeile haben):
```
SELECT ...
       { * | expression-commalist }
       INTO parameter-commalist
       FROM table-reference-commalist
       ...
```
  - Cursor-Konzept

- Ein Cursor ist ein Zeiger auf eine Zeile einer Tabelle, die als Ergebnistabelle einer Select-Anweisung ermittelt wird

- Cursor definieren:
```
DECLARE cursor CURSOR FOR select-statement [FOR UPDATE]
```
  - FOR UPDATE notwendig für positioned UPDATE und DELETE

- Cursor öffnen:
```
OPEN cursor
```
  - Wertet das select-statement der Cursordefinition aus und stellt den Cursor vor die erste Zeile der Ergebnistabelle

- Zeile lesen:
```
FETCH cursor INTO parameter-commalist
```
  - Der Cursor wird um eine Zeile in der Ergebnistabelle weiterbewegt und danach der Inhalt der Spalten in entsprechender Reihenfolge den Parametern zugewiesen (Anzahl und Datentyp!)
  - Wenn beim Weiterbewegen des Cursors das Tabellenende überschritten wird
    - Variablen wird bestimmter Wert zugewiesen (z.B. sqlcode=100)
    - Funktion liefert bestimmten Wert
    - Eigenschaft des Cursor-Objekts nimmt einen bestimmten Wert an, etc.
    - etc.

- Zeile ändern (positioned UPDATE):
```
UPDATE table SET assignment-commalist WHERE CURRENT OF cursor
```

- Zeile löschen (positioned DELETE):
```
DELETE FROM table WHERE CURRENT OF cursor
```

- Cursor schließen:
```
CLOSE cursor
```

8.10 Übungen zu SQL

1) Erstellen Sie die SQL-Anweisungen zur Definition der Tabellen, die dem in 2.3.3. dargestellten ER-Diagramm entsprechen. Überlegen Sie, in welcher Reihenfolge die Zeilen in die Tabellen Angestellter und Abteilung einzufügen sind. Ergänzen Sie die SQL-Anweisungen auch um das Löschen der Tabellen.
   Abfrage: Namen aller Angestellen, Bezeichnung der Abteilung zu der er gehört und Name des Abteilungsleiters

2) Join-Beispiele

```
drop table a;
drop table b;

create table a (       a1 integer, a2 integer, a3 integer);
insert into  a values ( 1,          1,          1);
insert into  a values ( 2,          1,       NULL);
insert into  a values ( 3,          1,          1);
insert into  a values ( 4,          1,       NULL);

create table b (       b1 integer, b2 integer, b3 integer);
insert into  b values ( 1,          2,          1);
insert into  b values ( 2,          2,       NULL);
insert into  b values ( 3,          2,          1);

select * from a cross join b;

select * from a join b on a.a1=b.b1;
select * from a join b on a.a1=b.b2;
select * from a join b on a.a1=b.b3;
select * from a join b on a.a2=b.b1;
select * from a join b on a.a2=b.b2;
select * from a join b on a.a2=b.b3;
select * from a join b on a.a3=b.b1;
select * from a join b on a.a3=b.b2;
select * from a join b on a.a3=b.b3;

select * from a join b on a.a1=b.b1*2;
select * from a join b on a.a2=b.b2-1;

select * from a join b on a.a1=b.b1 and a.a2=b.b2;
select * from a join b on a.a1=b.b2 and a.a2=b.b1;

select * from a a1 join a a2 on a1.a1=a2.a1;
select * from a a1 join a a2 on a1.a2=a2.a2;
select * from a a1 join a a2 on a1.a3=a2.a3;

select * from a a1 join a a2 on a1.a1=a2.a2;
select * from a a1 join a a2 on a1.a1=a2.a3;
select * from a a1 join a a2 on a1.a2=a2.a3;

select * from a join b on a.a1<=b.b2;
select * from a join b on a.a2<>b.b1;
select * from a join b on a.a1> b.b3;
select * from a join b on a.a3<>b.b3;

select * from a join b on a.a1=b.b1 where a.a3 is not null;
select * from a join b on a.a3=b.b3 where a.a1<4;
```

   Wieviele Zeilen haben die Ergebnistabellen der einzelnen Joins?
   Wie schauen die Ergebnistabellen der einzelnen Joins aus?

3) Eine Tabelle a (a1, a2) mit 3 Zeilen sei gegeben (alles INTEGER-Spalten):

```
drop   table a;
create table a (a1 integer primary key, a2 integer not null);
```

	Zeilenanz. mindestens	Zeilenanz. höchstens
`select * from a where a1=7;`		
`select * from a where a2=7;`		
`select * from a where a1=a2;`		
`select * from a x cross join a y;`		
`select * from a x join a y on x.a1=y.a1;`		
`select * from a x join a y on x.a1=y.a2;`		
`select * from a x join a y on x.a2=y.a2;`		

Wie ändern sich die Ergebnisse, wenn bei a2 nicht NOT NULL angegeben ist?

4) Lösen Sie folgende Aufgabenstellungen mit SQL-Anweisungen (Datenbank LT):

   a) Alle Teile (vollständige Information), die von keinem Lieferanten geliefert wurden, der aus derselben Stadt wie das Teil ist.

   b) Alle Lieferanten (LNr, LName), die zumindest alle Teile liefern, die auch L2 liefert.

   c) Alle Lieferanten (LNr, LName), die das Teil T2 in einer Menge liefern, die größer als die durchschnittliche Liefermenge von T2 ist.

   d) Pro Lieferant ist das teuerste Teil, das von Ihm geliefert wurde, anzuzeigen.
   Form: LNr, TNr, TName, Preis

   e) Erhöhen Sie bei allen Lieferungen von Lieferanten aus Paris die Menge um 10%.

   f) Löschen Sie alle Lieferanten, die kein Teil geliefert haben.

   g) Definieren Sie eine View, die nur Lieferungen (samt Lieferantenname und Teilename) enthält, bei denen der Umsatz größer als 2000 ist.

5) Für alle roten und blauen Teile, bei denen die Gesamtliefermenge größer als 350 ist (wobei in der Gesamtliefermenge Lieferungen in einer Menge kleiner oder gleich 200 nicht berücksichtigt werden sollen), sind die Teilenummer, der Preis in Dollar (1 Euro = 1,2 Dollar), die Farbe und die größte Liefermenge (nach dieser absteigend sortiert) auszugeben (Datenbank LT).

6) Gegeben sind folgende Tabellen mit den daneben stehenden Bedeutungen.

BESUCHER	(GAST, WIRTSHAUS)	Welcher Gast besucht welches Wirtshaus?
ANGEBOT	(WIRTSHAUS, BIER)	Welches Wirtshaus bietet welches Bier an?
GESCHMACK	(GAST, BIER)	Welcher Gast mag welches Bier?

Geben Sie jeweils einen SQL-Befehl an, der die folgenden Fragen beantwortet (*=leicht, **=mittel, ***=schwer):

a) * Welche Wirtshäuser bieten ein Bier an, das 'Hans' mag?

b) * Welche Gäste besuchen (mindestens) ein Wirtshaus,
das (mindestens) ein Bier nach ihrem Geschmack anbietet?

c) *** Welche Gäste besuchen nur Wirtshäuser, die (mindestens) ein Bier nach ihrem Geschmack anbieten?

d) ** Welche Gäste besuchen kein Wirtshaus, das (mindestens) ein Bier nach ihrem Geschmack anbietet?

e) * Welche Biere, die in irgendeinem Wirtshaus angeboten werden, mag kein Gast?

f) *** Welche Gäste der 'Braustube' mögen alle Biere, die dort angeboten werden?

g) ** Welche Wirtshäuser bieten alle Biere an, die irgendein Gast mag?

7) ORACLE-Demodatenbank:

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

EMP

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800,00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600,00	300,00	30
7521	WARD	SALESMAN	7698	1981-02-22	1250,00	500,00	30
7566	JONES	MANAGER	7839	1981-04-02	2975,00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-09-28	1250,00	1400,00	30
7698	BLAKE	MANAGER	7839	1981-05-01	2850,00	NULL	30
7782	CLARK	MANAGER	7839	1981-06-09	2450,00	NULL	10
7788	SCOTT	ANALYST	7566	1982-12-09	3000,00	NULL	20
7839	KING	PRESIDENT	NULL	1981-11-17	5000,00	NULL	10
7844	TURNER	SALESMAN	7698	1981-09-08	1500,00	0,00	30
7876	ADAMS	CLERK	7788	1983-01-12	1100,00	NULL	20
7900	JAMES	CLERK	7698	1981-12-03	950,00	NULL	30
7902	FORD	ANALYST	7566	1981-12-03	3000,00	NULL	20
7934	MILLER	CLERK	7782	1982-01-23	1300,00	NULL	10

SALGRADE

GRADE	LOSAL	HISAL
1	700,00	1200,00
2	1201,00	1400,00
3	1401,00	2000,00
4	2001,00	3000,00
5	3001,00	9999,00

8) Ein Fahrtenbuch wird in einer Tablle geführt:
`create table fbuch (datum date primary key, kmstand integer, km integer);`
Am Tagesende wird der Kilometerstand in das Fahrtenbuch geschrieben.
In die Spalte km sollen die am Tag gefahrenen Kilometer eingetragen werden
(CURSOR mit POSITIONED UPDATE).

8.11 Datenbank LTP: Lieferanten - Teile - Projekte - Lieferungen

L

LNR	LNAME	RABATT	STADT
L1	Schmid	20	London
L2	Jonas	10	Paris
L3	Berger	30	Paris
L4	Klein	20	London
L5	Adam	30	Athen

T

TNR	TNAME	FARBE	PREIS	STADT
T1	Mutter	rot	12	London
T2	Bolzen	gelb	17	Paris
T3	Schraube	blau	17	Rom
T4	Schraube	rot	14	London
T5	Welle	blau	12	Paris
T6	Zahnrad	rot	19	London

P

PNR	PNAME	STADT
P1	Flugzeug	Paris
P2	Schiff	Rom
P3	Seilbahn	Athen
P4	Schiff	Athen
P5	Eisenbahn	London
P6	Flugzeug	Oslo
P7	Autobus	London

LTP

LNR	TNR	PNR	MENGE
L1	T1	P1	200
L1	T1	P4	700
L2	T3	P1	400
L2	T3	P2	200
L2	T3	P3	200
L2	T3	P4	500
L2	T3	P5	600
L2	T3	P6	400
L2	T3	P7	800
L2	T5	P2	100
L3	T3	P1	200
L3	T4	P2	500
L4	T6	P3	300
L4	T6	P7	300
L5	T1	P4	100
L5	T2	P2	200
L5	T2	P4	100
L5	T3	P4	200
L5	T4	P4	800
L5	T5	P4	400
L5	T5	P5	500
L5	T5	P7	100
L5	T6	P2	200
L5	T6	P4	500

MENGE > 0

1) Tabellen anlegen
   a) Erstellen der Tabellen
   b) Füllen der Tabellen

2) Einfache Abfragen
   a) Alle Projekte im Detail
   b) Projekte in London im Detail
   c) Nummern der Lieferanten, die das Projekt P2 beliefern
   d) Lieferungen mit einer Menge im Bereich von 300 bis 750
   e) Verschiedene Kombinationen aus Teile-Farben und Teile-Städte
   f) Lieferungen mit einer nichtleeren Menge
   g) Projektnummer und Städte mit einem 'o' als zweiten Buchstaben des Stadtnamens

3) UNION
   a) Liste aller Städte aus denen zumindest ein Lieferant, ein Teil oder ein Projekt ist
   b) Ergebnis von: SELECT FARBE FROM T UNION SELECT FARBE FROM T;
   c) 3b) mit UNION ALL statt UNION
   d) Ergebnis von: SELECT DISTINCT FARBE FROM T UNION ALL SELECT DISTINCT FARBE FROM T;

4) Subqueries
   a) Namen der Projekte, die vom Lieferanten L1 beliefert werden
   b) Farben der Teile, die vom Lieferanten L1 geliefert werden
   c) Nummern der Teile, die an ein Projekt in London geliefert werden
   d) Nummern der Projekte, die zumindest ein Teil beinhalten, das L1 liefert
   e) Nummern der Lieferanten, die zumindest ein Teil liefern, das zumindest von einem Lieferant geliefert wird, der zumindest ein rotes Teil liefert
   f) Nummern der Lieferaten mit einem Rabatt kleiner als der des Lieferanten L1

5) Join
   a) Tripel aus Lieferanten-, Teile- und Projektnummer so, dass die Lieferanten, Teile und Projekte alle aus derselben Stadt sind
   b) Tripel aus Lieferanten-, Teile- und Projektnummer so, dass die Lieferanten, Teile und Projekte nicht alle aus derselben Stadt sind
   c) Tripel aus Lieferanten-, Teile- und Projektnummer so, dass nicht zwei der Lieferanten, Teile und Projekte aus derselben Stadt sind
   d) Nummern der Teile, die von einem Lieferanten aus London geliefert werden
   e) Nummern der Teile, die von einem Lieferanten aus London an ein Projekt in London geliefert werden
   f) Paare von Städtenamen so, dass ein Lieferant aus der ersten Stadt ein Projekt in der zweiten Stadt beliefert
   g) Nummern der Teile, die an ein Projekt von einem Lieferanten geliefert werden, der aus derselben Stadt wie das Projekt ist
   h) Nummern der Projekte, die zumindest von einem Lieferanten beliefert werden, der nicht aus derselben Stadt wie das Projekt ist
   i) Paare von Teilenummern so, dass derselbe Lieferant beide Teile liefert (Self-Join)

6) IN / EXISTS
   a) 4c) mit EXISTS
   b) 4d) mit EXISTS
   c) Nummer der Projekte, die nicht mit roten Teilen von Lieferanten aus London beliefert werden
   d) Nummern der Projekte, die ausschließlich von Lieferant L2 beliefert werden
   e) Nummern der Teile, die an alle Projekte in London geliefert werden
   f) Nummern der Projekte, die zumindest mit allen Teilen beliefert werden, die Lieferant L1 liefert
   g) Welche Lieferanten (alle Spalten) haben alle Teile geliefert?
   h) Welche Lieferanten (alle Spalten) haben alle Teile an ein Projekt geliefert?
   i) Welche Lieferanten (alle Spalten) haben ein Teil an alle Projekte geliefert?
   j) Welche Lieferanten (alle Spalten) haben alle Teile an alle Projekte geliefert?

7) Aggregatfunktionen
   a) Anzahl der Projekte, die vom Lieferant L5 beliefert werden
   b) Gesamtliefermenge des Teiles T1 durch Lieferant L1
   c) Für jedes Teil, das an ein Projekt geliefert wird: Teilenummer, Projektnummer und entsprechende Summe der Liefermengen
   d) Nummern der Projekte, deren Stadt die erste in der alphabetischen Reihenfolge der Projekt-Städte ist
   e) Nummern der Projekte, die mit Teil T1 in einer durchschnittlichen Menge beliefert werden, die größer ist als die größte Liefermenge in der ein Teil an Projekt P1 geliefert wird
   f) Nummern der Lieferanten, die ein Projekt mit Teil T1 in einer Menge beliefern, die größer ist als die Durchschnittsmenge in der Teil T1 an dieses Projekt geliefert wird (Correlated Sub-Query)

8) INTERSECT / EXCEPT
   a) Projekte (nur PNr oder PNr und PName), die keine Teile aus London beinhalten
   b) Lieferanten (nur LNr oder LNr und LName), die Projekte in London und Paris beliefern
   c) 8a) und 8b) ohne Mengenoperationen

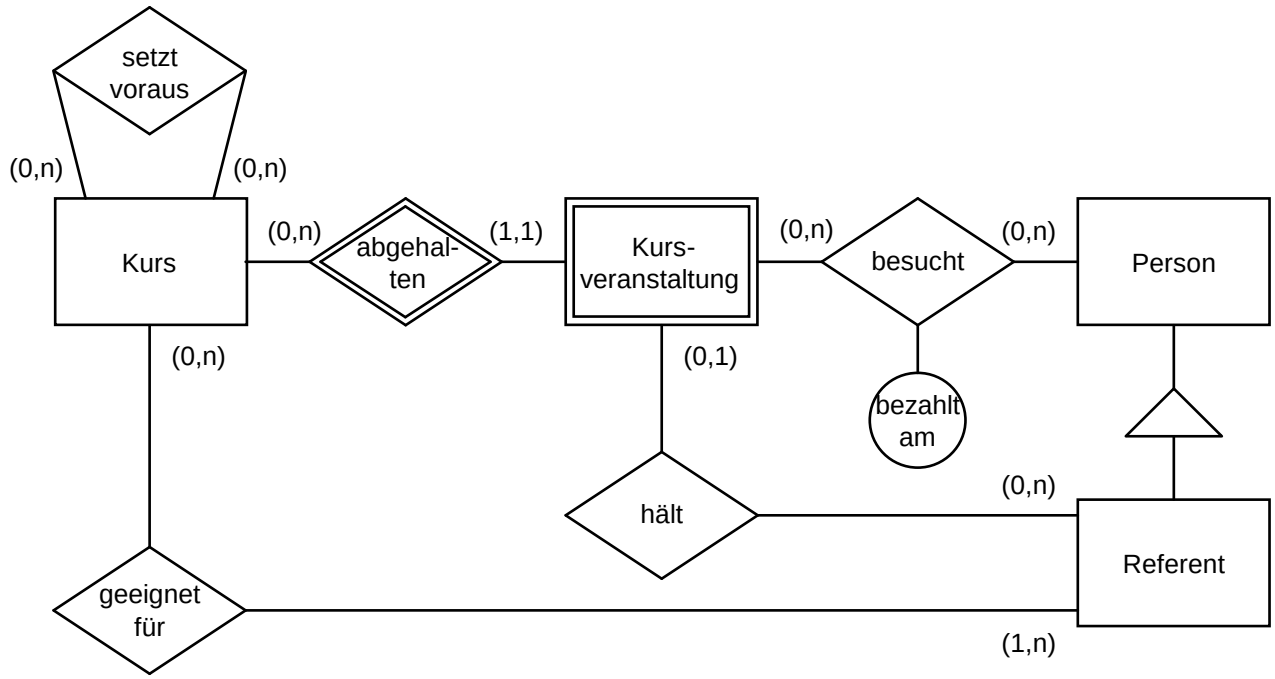
9) Änderungen von Tabelleninhalten
   a) Farbe aller roten Teile in rosa ändern
   b) Alle Projekte, die nicht beliefert werden, löschen
   c) Bei allen Lieferungen von Lieferanten, die ein rotes Teil liefern, die Menge um 10 Prozent erhöhen
   d) Alle Projekte in Rom löschen
   e) Neuen Lieferanten (L9) aus Wien namens Schuh aufnehmen, dessen Rabatt noch nicht bekannt ist
   f) Tabelle erstellen mit allen Nummern von Teilen, die von einem Lieferanten aus London oder an ein Projekt in London geliefert werden
   g) Tabelle erstellen mit allen Nummern von Projekten, die in London sind oder von einem Lieferanten aus London beliefert werden
   h) Bei allen Lieferanten, deren Rabatt kleiner als der von Lieferant L4 ist, den Rabatt um 5 erhöhen

10) Views
   a) Projekte in London (PNR, PNAME, STADT)
   b) Aus LTP abgeleitet: LT (LNR, TNR, MENGE)
   c) Projekte (PNR, STADT), die von Lieferant L1 oder mit Teil T1 beliefert werden
   d) Alle Kombinationen aus Nummern von Teilen und Lieferanten, die nicht aus derselben Stadt sind (LNR, TNR)

11) Tabellen löschen

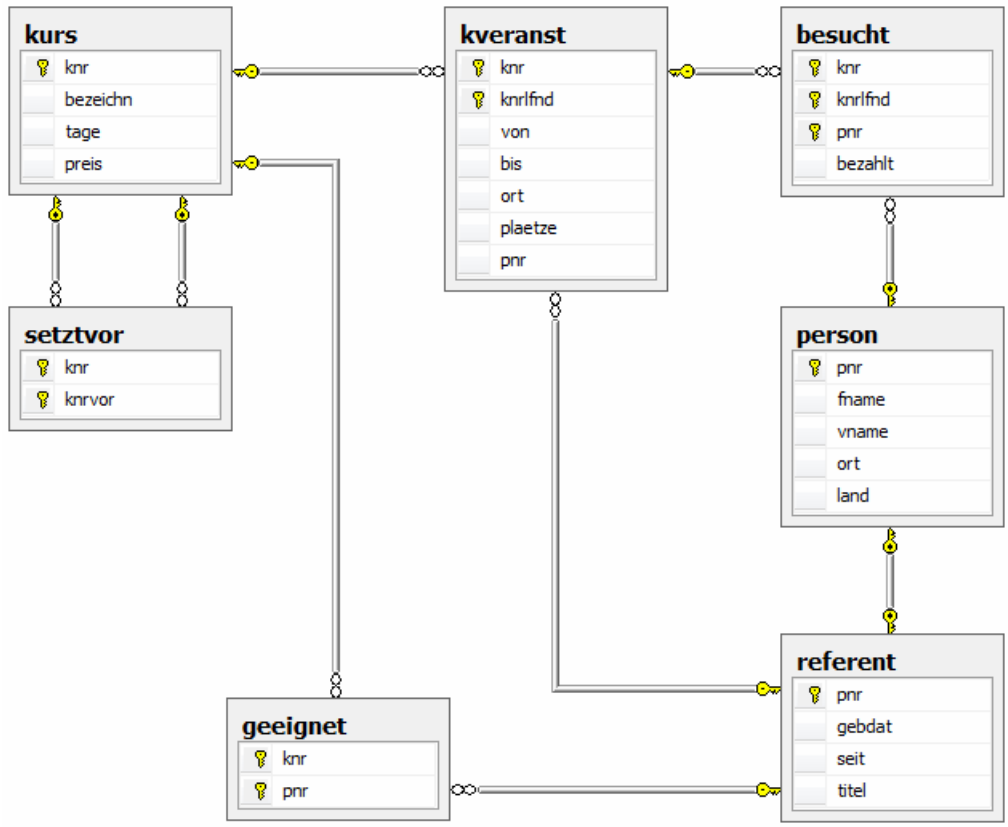
8.12 Datenbank Schulungsfirma

1) Erstellen Sie für folgendes ER-Diagramm einer Schulungsfirma die notwendigen SQL-Anweisungen zur Definition der entsprechenden Tabellen



Attribute
- Person: PNr, FName, VName, Ort, Land (A, D, F, GB, I oder RUS)
- Referent: GebDat, Seit, Titel (GebDat kleiner Seit)
- Kurs: KNr, Bezeichn, Tage (mindestens 1 und höchstens 10), Preis
- KVeranst: KNrLfnd, Von, Bis, Ort, Plätze (von nicht größer als bis, bis kann NULL sein)

2) Die Schulungsfirma hat ihre Datenbank in einem RDBMS implementiert. Das folgende Datenbankdiagramm zeigt die Tabellen (mit ihren Spalten und Primärschlüssel) samt den Beziehungen zwischen den Tabellen (Fremdschlüssel)



PERSON

PNR	FNAME	VNAME	ORT	LAND
101	Bach	Johann Sebastian	Leipzig	D
102	Händel	Georg Friedrich	London	GB
103	Haydn	Joseph	Wien	A
104	Mozart	Wolfgang Amadeus	Salzburg	A
105	Beethoven	Ludwig van	Wien	A
106	Schubert	Franz	Wien	A
107	Berlioz	Hector	Paris	F
108	Liszt	Franz	Wien	A
109	Wagner	Richard	München	D
110	Verdi	Giuseppe	Busseto	I
111	Bruckner	Anton	Linz	A
112	Brahms	Johannes	Wien	A
113	Bizet	Georges	Paris	F
114	Tschaikowskij	Peter	Moskau	RUS
115	Puccini	Giacomo	Mailand	I
116	Strauss	Richard	München	D
117	Schönberg	Arnold	Wien	A

REFERENT

PNR	GEBDAT	SEIT	TITEL
101	21.3.1935	1.1.1980	
103	1.4.1932	1.1.1991	
104	27.1.1956	1.7.1985	
111	4.9.1924	1.7.1990	Mag
114	25.4.1940	1.7.1980	
116	11.6.1964	1.1.1994	Dr

KURS

KNR	BEZEICHN	TAGE	PREIS
1	Notenkunde	2	1400,00
2	Harmonielehre	3	2000,00
3	Rhythmik	1	700,00
4	Instrumentenkunde	2	1500,00
5	Dirigieren	3	1900,00
6	Musikgeschichte	2	1400,00
7	Komposition	4	3000,00

GEEIGNET

KNR	PNR
1	103
1	114
2	104
2	111
3	103
4	104
5	101
5	114
6	111
7	103
7	116

SETZTVOR

KNR	KNRVOR
2	1
3	1
5	2
5	3
5	4
7	5
7	6

KVERANST

KNR	KNRLFND	VON	BIS	ORT	PLAETZE	PNR
1	1	7.4.2003	8.4.2003	Wien	3	103
1	2	23.6.2004	24.6.2004	Moskau	4	114
1	3	10.4.2005	11.4.2005	Paris	3	
2	1	9.10.2003	11.10.2003	Wien	4	104
3	1	17.11.2003	17.11.2003	Moskau	3	103
4	1	12.1.2004	13.1.2004	Wien	2	116
4	2	28.3.2004	29.3.2004	Wien	4	104
5	1	18.5.2004	20.5.2004	Paris	3	101
5	2	23.9.2004	26.9.2004	Wien	2	101
5	3	30.3.2005	1.4.2005	Rom	3	
7	1	9.3.2005	13.3.2005	Wien	5	103
7	2	14.9.2005	18.9.2005	Muenchen	4	116

BESUCHT

KNR	KNRLFND	PNR	BEZAHLT
1	1	108	1.5.2003
1	1	109	
1	1	114	
1	2	110	1.7.2004
1	2	112	3.7.2004
1	2	113	20.7.2004
1	2	116	
1	3	110	
2	1	105	15.10.2003
2	1	109	3.11.2003
2	1	112	28.10.2003
2	1	116	
3	1	101	
3	1	109	
3	1	117	20.11.2003
4	1	102	20.1.2004
4	1	107	1.2.2004
4	1	111	
4	2	106	7.4.2004
4	2	109	15.4.2004
5	1	103	
5	1	109	7.6.2004
5	2	115	7.10.2004
5	2	116	
7	1	109	20.3.2005
7	1	113	
7	1	117	8.4.2005

4) Welche Kurse (KNr) haben einen Kurs als Voraussetzung?

5) Welche Kurse (Bezeichnung) dauern zwischen 2 und 4 Tagen und haben einen durchschnittlichen Tagespreis von höchstens 700,--? (aufsteigend nach Bezeichnung sortiert)

6) Welche Personen (Familienname) haben ein Leerzeichen im Vornamen und denselben Vokal zweimal im Ort?

7) Welche Personen (PNr, aufsteigend sortiert danach) haben noch nicht alle Kursbesuche bezahlt?

8) Wieviele Tage dauern die Kursveranstaltungen, die in Wien stattfinden und von Referent 103 oder 104 gehalten werden (KNr, KNrLfnd, Tage - danach absteigend)?

9) Welche Referenten (PNr, Alter) sind älter als 75 Jahre?

10) Welche Personen (PNr) halten oder besuchen mindestens eine Kursveranstaltung?

11) Alle Kursveranstaltungen (KNr, Bezeichnung, KNrLfnd, von) bei denen noch kein Referent festgelegt ist

12) Alle Kursveranstaltungen (KNr, Bezeichnung, KNrLfnd, von), die mindestens ein Teilnehmer besucht

13) Alle Kursveranstaltungen (KNr, Bezeichnung, KNrLfnd, von), die mindestens ein Teilnehmer besucht und bei denen schon ein Referent festgelegt ist

14) Referenten zahlen für Besuche von Kursveranstaltungen nichts.
Zeigen Sie Besuche (samt Kursbezeichnng und Familienname) an, wo dies nicht eingehalten ist.

15) Teilnehmerliste pro Kursveranstaltung mit folgenden Spalten (sortiert nach vonDatum und Kursbezeichnung):
- Kursbezeichnung
- vonDatum
- Familien- und Vorname des Referenten
- Familien- und Vorname des Teilnehmers

16) Welche Kursveranstaltungen (Bezeichnung, vonDatum) werden von Referenten besucht?

17) Welche Kursveranstaltungen (Bezeichnung, vonDatum) werden von Referenten besucht und halten diese auch? (Fehlerfall!)

18) Welche Personen (FName) haben den Kurs Nr 1 und den Kurs Nr 5 besucht?

19) Alle Kursveranstaltungen mit durchschnittlichen Tagespreisen zwischen 610,-- und 690,--, die von Referenten ohne Titel gehalten werden (Bezeichnung, von, bis, durchschnittlicher Tagespreis)

20) Welche Kursbesuche wurden vor dem Kursbeginn bezahlt?
Welche Kursbesuche wurden während des Kurses bezahlt?
Welche Kursbesuche wurden nach dem Kursende bezahlt?

21) Welche Personen (Familienname, danach sortiert) besuchen Kursveranstaltungen, die in ihrem Wohnort abgehalten werden und länger als zwei Tage dauern?

22) Welche Kursveranstaltungen (Bezeichnung, laufende Nummer) werden von Referenten gehalten, die für den Kurs auch geeignet sind?

23) Alle Referenten (Nummer und Name), die Kursveranstaltungen gehalten haben bevor / nachdem sie in die Firma eingetreten sind (seit).

24) Alle Personen (PNr, FName), die einen Kurs in 'Wien' besucht oder gehalten haben.

25) Dauer der Kursveranstaltungen im Vergleich mit der im Kurs angegebenen Dauer
(geht die Veranstaltung über ein Wochenende / Sa,So?)

26) Welche Referenten (Nummer und Name), haben Kursveranstaltungen in einem Alter von über 60 Jahren gehalten?

27) Welche Kursveranstaltungen gibt es, zu denen eine (unmittelbar) vorausgesetzte Kursveranstaltung zeitlich davor und am selben Ort abgehalten wird?
(jeweils alle Daten der Kursveranstaltung und der vorausgesetzten Kursveranstaltung)

28) Welche Kursveranstaltungen überschneiden einander terminlich? (je Bezeichnung, laufende Nummer, von, bis)

29) Gibt es Personen (Familien- und Vorname), bei denen Kursbesuche einander terminlich überschneiden?

30) Gibt es Referenten (Familien- und Vorname), bei denen Kursveranstaltungen, die sie halten, einander terminlich überschneiden?

31) Laut ERD muss jeder Referent für mindestens einen Kurs geeignet sein.
Gibt es Referenten (Name), bei denen diese Bedingung nicht eingehalten ist?

32) Welche Kurse (Bezeichnung) kosten nicht mehr als der Kurs 'Dirigieren'?

33) Welche Kurse (Bezeichnung) sind teurer als alle, die in 'Paris' veranstaltet wurden?

34) Welche Kurse (Bezeichnung) setzen keine Kurse voraus?

35) Welche Personen (FName, VName) haben im Jahr 2003 oder 2005 keine Kursveranstaltung besucht?

36) Welche Personen (FName, VName) haben im Jahr 2003 und 2005 keine Kursveranstaltung besucht?

37) Welche Personen (FName, VName) haben einen Kurs besucht, der billiger ist als ein Kurs, der in 'Moskau' veranstaltet wurde?

38) Alle Kursveranstaltungen (KNr, Bezeichnung, KNrlfnd und von), die von einem Referent (PNr) gehalten werden, der nicht geeignet ist

39) Welche Personen (PNr und FName) besuchen Kurse in 'Wien' und in 'Paris'?
Welche Personen (PNr und FName) besuchen Kurse in 'Wien' aber nicht in 'Paris'?

40) Welche Kurse (Bezeichnung) fanden in 'Wien' und in 'Paris' statt?
Welche Kurse (Bezeichnung) fanden in 'Wien' aber nicht in 'Paris' statt?

41) Welche Personen (PNr) haben mindestens zwei Kursveranstaltungen besucht?

42) Alle Kursveranstaltungen (KNr, KNrLfnd), die keiner besucht

43) Pro Kursveranstaltung den tatsächlichen Preis anzeigen: Normalerweise der Kurs-Preis, wenn an einem Wochenende (Sa oder So enthalten), dann 10% (einmal / pro Wochenend-Tag) teurer

44) Alle Kursveranstaltungen (KNr, KNrLfnd) mit vorhandenen, belegten und freien Plaetzen
(nur Kursveranstaltungen, bei denen noch Plätze frei sind)

45) Für wieviele Kurse gibt es noch keinen geeigenten Referenten?
Für wieviele Kursveranstaltungen gibt es noch keinen geeigenten Referenten?

46) Welche Referenten halten keine Kursveranstaltung?

47) Welche Personen besuchen keinen Kurs?

48) Alle Kurse (nach Bezeichnung geordnet) mit jeweils allen Kursen, die unmittelbare Voraussetzung sind (Bezeichnung), anzeigen.

49) Zu welchen Kursen, die länger als einen Tag dauern, gibt es keine Kursveranstaltungen, die in 'Wien' stattfinden?

50) Welche Personen (PNr, FName) haben alle Kurse besucht?

51) Welche Personen feiern während eines Kursbesuches / Kursreferates Geburtstag?

52) Für welche Kurse (KNr, Bezeichnung) sind ausschließlich österreichische Referenten geeignet?

53) Alle Personen (PNr, FName), die einen Kurs in 'Wien' besucht und gehalten haben

54) In welchen Orten wurden alle Kurse abgehalten?

55) Welche Kurse (Bezeichnung) kosten weniger als die Hälfte des teuersten Kurses?

56) Wie viele Tage dauern und was kosten die Kurse im Durchschnitt?

57) Wie viele Tage dauert die längste und die kürzeste Kursveranstaltung?

58) Für jeden Referenten, der mindestens eine Kursveranstaltung gehalten hat, geben Sie an:
Nummer des Referenten, Anzahl der Kursveranstaltungen (nach Anzahl absteigend sortiert)

59) Für jeden Referenten, der mindestens zwei Kursveranstaltungen gehalten hat, geben Sie an:
Familien- und Vorname des Referenten, Anzahl der Kursveranstaltungen (nach Name sortiert)

60) Für jede Person, die mindestens eine Kursveranstaltung besucht hat, geben Sie an:
Familien- und Vorname der Person, Summe der besuchten Kurstage
(nach Summe der Kurstage absteigend sortiert)

61) Wie heißt (Familien- und Vorname) der älteste Referent?

62) Wieviele Kursveranstaltungen haben die Personen aus Österreich, die keine Referenten sind, jeweils besucht?
(Personennummer, Familienname, Anzahl Kursveranstaltungen)

63) Welche Beträge haben die einzelnen Kursteilnehmer in Summe schon bezahlt?
(Familien- und Vorname, Summe der Zahlungen - danach sortiert)
Nur Personen ausgeben, die in Summe schon mehr als 2000,-- bezahlt haben.

64) Geben Sie die Familiennamen der Personen aus, die für mehr als zwei Kurse geeignet sind

65) Alle Kurse (KNr, Bezeichnung) samt Anzahl der geeigneten Referenten für diesen Kurs und Anzahl der Kursveranstaltungen zu diesem Kurs

66) Welche Referenten (PNr, FName) haben Kursveranstaltungen gehalten, die höchstens einen vorausgesetzten Kurs haben?

67) Welche Personen haben eine Kursveranstaltung besucht (Daten der Person und der Kursveranstaltung) zu der sie noch nicht alle (unmittelbar) vorausgesetzten Kurse (terminlich vorher) besucht haben?

68) Welche Kurse (Bezeichnungen) setzt der Kurs 'Komposition' alle voraus? (Stored Routine)

69) Fügen Sie in geeignet die Person 101 für alle Kurse ein

70) Alle Personen, die den Kurs 1 noch nicht besucht haben, sollen in besucht für die Kursveranstaltung KNr 1 und KNrLfnd 3 eingefügt werden

71) Alle ausständigen Bezahlungen der Kursbesuche sind mit aktuellem Tagesdatum zu begleichen

72) Erhöhen Sie die Preise aller Kurse um 100,--, bei denen es höchstens 2 Veranstaltungen gibt.

73) Die Platzanzahl aller Kursveranstaltungen, die voll ausgebucht sind, soll verdoppelt werden

74) Erhöhen Sie die Kursdauer bei allen Kursen, die länger als 2 Tage dauern, um 1 Tag und verlängern Sie auch die entsprechenden Kursveranstaltungen durch Änderung des bis-Datums.

75) Löschen Sie möglichst alle Personen, soweit dies unter Einhaltung der referentiellen Integrität möglich ist.

76) Löschen Sie alle Kursveranstaltungen, für die es keine Besuche gibt und die mehr als 2 Kurse als Voraussetzung haben