```
---------------------------------------------------------------------------
-- \i C:/Users/Gogolla/Desktop/dblp2sql/sql-intro.txt # skript einfuegen
-- \dt # vorhandene tabellen anzeigen
---------------------------------------------------------------------------
-- Grundlagen von Datenbanken - Martin Gogolla
---------------------------------------------------------------------------
-- SQL: Structured Query Language
-- DDL: Data Definition Language (CREATE TABLE, ...)
-- DML: Data Manipulation Language (INSERT, SELECT, ...)
--
-- diverse Implementierungen: MySQL, SQLite, PostgreSQL, ...
-- hier verwendet: PostgreSQL; google 'postgresql download deutsch'
-- nur Standard-SQL-Sprachmittel in diesem Kurs verwendet
--
---------------------------------------------------------------------------
-- Beispieldaten für eine relationalen Datenbank:
-- Darstellung von wissenschaftlichen Publikationen
---------------------------------------------------------------------------

-- Gogolla, Vallecillo:
--    Modeling principles. EduSym 2012: 28-31
-- Kuhlmann, Gogolla:
--    UML and OCL to Relational Logic. MoDELS 2011 [!]: 415-431
-- Kuske, Gogolla, Kreowski [!]:
--    Graph-Based Semantics for UML. SoSyM 8(3): 403-422 (2009)
--
-- gekürzte bzw. angepasste Version von: google 'gogolla dblp' ...

---------------------------------------------------------------------------
-- Erzeugung von Tabellen und Daten
---------------------------------------------------------------------------

-- Groß-Klein-Schreibweise von Schlüsselworten unerheblich in SQL
-- create table == CREATE TABLE == Create Table

DROP TABLE IF EXISTS autpub CASCADE;
CREATE TABLE autpub(
  authors  VARCHAR,
  title    VARCHAR,
  venue    VARCHAR,
  year     INTEGER,
  pubtype  VARCHAR);

INSERT INTO autpub VALUES ('Gogolla and Vallecillo',
  'Modeling Principles.', 'EduSym 2012:28-31',
  2012, 'inproc');

INSERT INTO autpub VALUES ('Kuhlmann and Gogolla',
  'UML and OCL to Relational Logic.', 'MoDELS 2011:415-431',
  2011, 'inproc');

INSERT INTO autpub VALUES ('Kuske and Gogolla and Kreowski',
  'Graph-Based Semantics for UML.', 'SoSyM 2009:403-422',
  2009, 'article');

-- zentrale Begriffe: Tabelle (Relation), Zeile (Tupel),
-- Spalte (Attribut), Datentyp,
-- relationale Datenbank i.a. bestehend aus mehreren Tabellen

-- SQL-Datentypen: Integer, Varchar (entspricht String), Boolean, Real,
-- Float(n) mit n Nachkommastellen, Decimal(n,m) mit n Stellen davon
-- m Nachkommastellen, Date, Time, ...

------------------------------------------------------------------------
-- Erste Anfragen mittels
-- select <attribut*> from <tabelle*> where <bedingung>
------------------------------------------------------------------------

-- select: * alle Attribute

select * from autpub;

-- where: Vergleichsoperatoren = <> < <= > >=
--        Operator between für Intervalle

select * from autpub where pubtype='inproc';

select * from autpub where pubtype<>'article';

select * from autpub where year<=2000;

select * from autpub where year between 2000 and 3000;

------------------------------------------------------------------------
-- where: Vergleichsoperator like und not like auf Zeichenketten
--        % steht für 0,1,2,... Zeichen und _ für 1 Zeichen
------------------------------------------------------------------------

select * from autpub where title like '%UML%';

select * from autpub where authors not like '%mann %';

select * from autpub where title like '% __L%';

------------------------------------------------------------------------
-- where: and or not
------------------------------------------------------------------------

select * from autpub where pubtype='inproc' and year>=2000;

select * from autpub where pubtype='inproc' or year>=2000;

select * from autpub where not (authors like '% and %');

select * from autpub
where not(pubtype='inproc') or venue like 'MoDELS%';
--    pubtype='inproc' implies venue like 'MoDELS%' -- in SQL verboten!

------------------------------------------------------------------------
-- select: *, Attribute, Konstanten, Ausdrücke
-- where: Ausdrücke ebenfalls erlaubt
------------------------------------------------------------------------

select year, title from autpub;

select substr(authors,1,18), 'schrieben', substr(title,1,18), year-1+1,
  'Jahre vor', year*4/2 from autpub;

-- substr(Argumentstring,Position,Laenge) PostgreSQL-spezifisch
-- Standard-SQL: substring(authors from 1 for 18)
-- Konkatenation von Strings in PostgreSQL: ||

select
  '>' || substr('Kuhlmann',5,3) || '#' || substr('Kuhlmann',5,5) || '<';

------------------------------------------------------------------------
-- Kartesisches Produkt
------------------------------------------------------------------------

drop table if exists mt cascade;
create table mt(
  mutter  varchar,
  tochter varchar);

insert into mt values ('Ada','Bel');
insert into mt values ('Bel','Cho');

select * from mt t1, mt t2;

select * from mt t1, mt t2 where t1.tochter=t2.mutter;

select t1.mutter, t1.tochter, t2.tochter
from   mt t1, mt t2
where  t1.tochter=t2.mutter;

select t1.mutter as oma, t1.tochter as muddi, t2.tochter as enkel
from   mt t1, mt t2
where  t1.tochter=t2.mutter;

select ome.enkel, ome.oma
from   ( select t1.mutter as oma, t1.tochter as mum, t2.tochter as enkel
         from   mt t1, mt t2
         where  t1.tochter=t2.mutter) ome
where  ome.oma='Ada';

------------------------------------------------------------------------
-- Spalten- und Tabellen-Umbenennung: as
------------------------------------------------------------------------

select substr(venue,1,6) as venue_SHORT from autpub;

select p1.year, p2.year from autpub as p1, autpub as p2;

select p1.year, p2.year from autpub p1, autpub p2;
-- auch ohne as

------------------------------------------------------------------------
-- distinct: duplikat-erhaltende VS duplikat-freie Anfragen;
--           Multimenge: Kollektion ähnlich einer Menge, in der Elemente
--             auch mehrfach auftreten können
--           Multimengen (Bags) VS Mengen (Sets)
------------------------------------------------------------------------

select pubtype from autpub;

select distinct pubtype from autpub;

select p1.pubtype, p2.pubtype from autpub p1, autpub p2;

select distinct p1.pubtype, p2.pubtype from autpub p1, autpub p2;
```

```
------------------------------------------------------------------------
-- Tabellen können Duplikate enthalten; inserts können nur bestimmte
-- Attribute belegen; nicht genannte Attribute werden mit dem
-- sogenannten null-Wert belegt (mehr dazu später)
------------------------------------------------------------------------

insert into autpub values ('Kuhlmann and Gogolla',
  'UML and OCL to Relational Logic.', 'MoDELS 2011:415-431',
  2011, 'inproc');

insert into autpub(authors,title) values ('Kuhlmann and Gogolla',
  'UML and OCL to Relational Logic.');

select * from autpub;

select * from autpub where year is null;

------------------------------------------------------------------------
-- Datenmodifikation: insert [values | select], delete, update;
-- beziehen sich i.a. Tupelmengen!
------------------------------------------------------------------------

delete from autpub where title='UML and OCL to Relational Logic.';

select * from autpub;

update autpub set authors='Kuske and Gogolla and Ziemann'
where 'Kuske and Gogolla and Kreowski'=authors;

-- erstes = Zuweisung, zweites = Gleichheitsabfrage

select * from autpub;

insert into autpub values ('Kuhlmann and Gogolla',
  'UML and OCL to Relational Logic.', 'MoDELS 2011:415-431',
  2011, 'inproc');

update autpub set authors='Kuske and Gogolla and Kreowski'
where authors='Kuske and Gogolla and Ziemann';

select * from autpub;

------------------------------------------------------------------------
-- Unteranfragen im where: in, exists, <vergleich> any, <vergleich> all
-- <vergleich> Vergleichsoperator = <> < <= > >=
------------------------------------------------------------------------

-- älteste Publikationen
select title, year from autpub p1 where
  not exists (select * from autpub p2 where p2.year<p1.year);
  -- korrelierte Unteranfrage, d.h. die Unteranfrage verwendet eine von
  -- 'oben' kommende Variable, hier p1

select title, year from autpub
  where year <= all (select year from autpub);
  -- nicht-korrelierte Unteranfrage, d.h., die Unteranfrage verwendet
  -- keine vone 'oben' kommenden Variable (keine freie Variablen)

-- immer die leere Menge als Ergebnis
```

```
select title, year from autpub
  where year < all (select year from autpub);

select title, year from autpub where not ( year in
  (select p1.year from autpub p1, autpub p2 where p1.year>p2.year));

select title, year from autpub where not ( year in
  (select p1.year from autpub p1, autpub p2 where p1.year>p2.year));

select p3.title, p3.year from autpub p3 where not ( p3.year = any
  (select p1.year from autpub p1, autpub p2 where p1.year>p2.year));

select title, year from autpub where year <> all
  (select p1.year from autpub p1, autpub p2 where p1.year>p2.year);

------------------------------------------------------------------------
-- Sichten
------------------------------------------------------------------------

drop view if exists tagungsartikel cascade;
create view tagungsartikel(jahr, titel, autoren) as
  (select year, title, authors from autpub where pubtype='inproc');

select * from tagungsartikel;

select * from tagungsartikel
where jahr <= all (select jahr from tagungsartikel);

drop view if exists uml cascade;
create view uml as
  (select * from autpub where title like '%UML%');

select * from uml;

------------------------------------------------------------------------
-- order by, group by, having
-- Aggregatfunktionen (count, min, max, sum, avg)
------------------------------------------------------------------------

select title, year, authors from autpub;

select title, year, authors from autpub order by year;

select title, year, authors from autpub order by 2;

select title, year, authors from autpub order by length(title);

select count(*) from autpub;

select count(*), count(pubtype), count(distinct pubtype) from autpub;

select pubtype, count(*) from autpub group by pubtype;

--     authors     |        title      | venue | year | _pubtype_
-- ----------------+-------------------+-------+------+----------
--  Gogolla and Val| Modeling Principle| EduSym| 2012 | inproc   GROUP1
--  Kuhlmann and Go| UML and OCL to Rel| MoDELS| 2011 | inproc
-- ----------------+-------------------+-------+------+----------
--  Kuske and Gogol| Graph-Based Semant| SoSyM | 2009 | article  GROUP2
```

```
select pubtype, min(year), avg(year), max(year), sum(year)
from autpub group by pubtype;

select pubtype from autpub group by pubtype having count(*)>=2;

-- select ... from ... where ... group by ... having ... order by ...
select pubtype, count(*) from autpub where 2011<=year and year<=2012
group by pubtype having count(*)>=2 order by 1;

-- mit großzügigem Layout und großgeschriebenen SQL-Schlüsselworten
SELECT   pubtype, COUNT(*)
FROM     autpub
WHERE    2011<=year AND year<=2012
GROUP BY pubtype
HAVING   COUNT(*)>=2
ORDER BY 1;

-- select pubtype, year from autpub group by pubtype;
-- FEHLER: Spalte »autpub.year« muss in der GROUP-BY-Klausel erscheinen
-- oder in einer Aggregatfunktion verwendet werden

select pubtype, avg(year) from autpub group by pubtype;

select pubtype, avg(distinct year) from autpub group by pubtype;

---------------------------------------------------------------------------
---------------------------------------------------------------------------
---------------------------------------------------------------------------
-- Daten in mehreren Tabellen
---------------------------------------------------------------------------

-- Gogolla, Vallecillo:
--    Modeling principles. EduSym 2012: 28-31
-- Kuhlmann, Gogolla:
--    UML and OCL to Relational Logic. MoDELS 2011 [!]: 415-431
-- Kuske, Gogolla, Kreowski [!]:
--    Graph-Based Semantics for UML. SoSyM 8(3): 403-422 (2009)

-- Alternative zu 1 Tabelle autpub(authors, title, venue, year, pubtype)
-- 2 Tabellen; statt Autorenliste hier Einzelautoren mit Postition;
-- Verbindung der beiden Tabellen über gemeinsames Attribut citekey;
-- pub(citekey, title, venue, year, pubtype); aut(citekey, author, pos);
-- man beachte die Bezeichnungen der Attribute: authors VS author

DROP TABLE IF EXISTS pub CASCADE;
CREATE TABLE pub(
  citekey VARCHAR,
  title   VARCHAR,
  venue   VARCHAR,
  year    INTEGER,
  pubtype VARCHAR);

DROP TABLE IF EXISTS aut CASCADE;
CREATE TABLE aut(
  citekey VARCHAR,
  author  VARCHAR,
  pos     INTEGER);

INSERT INTO pub VALUES ('GogollaV12',
  'Model Princ', 'EduSym 2012',
  2012, 'inproc');
INSERT INTO aut VALUES ('GogollaV12', 'Gogolla', 1);
INSERT INTO aut VALUES ('GogollaV12', 'Vallecillo', 2);

INSERT INTO pub VALUES ('KuhlmannG11',
  'UML and OCL', 'MoDELS 2011',
  2011, 'inproc');
INSERT INTO aut VALUES ('KuhlmannG11','Kuhlmann', 1);
INSERT INTO aut VALUES ('KuhlmannG11','Gogolla', 2);

INSERT INTO pub VALUES ('KuskeGK09',
  'Graph Seman', 'SoSyM 2009',
  2009, 'article');
INSERT INTO aut VALUES ('KuskeGK09', 'Kuske', 1);
INSERT INTO aut VALUES ('KuskeGK09', 'Gogolla', 2);
INSERT INTO aut VALUES ('KuskeGK09', 'Kreowski', 3);

-------------------------------------------------------------------------
-- Verbunde (Joins)
-------------------------------------------------------------------------

select * from pub;

select * from aut;

-- Vorteil 1 Tabelle: einfache Darstellung des Sachverhalts
-- Vorteil 2 Tabellen: Autorposition direkt zugreifbar
select author, pos from aut where author='Gogolla';

select * from pub, aut where pub.citekey=aut.citekey;

select * from pub natural join aut;

select * from aut natural join pub;

select * from pub join aut on pub.citekey=aut.citekey;

select * from pub join aut using (citekey);

-- outer join, null value

INSERT INTO pub VALUES
  ('OMG14','UML 3.0', 'www.omg',2014, 'article');
INSERT INTO aut VALUES
  ('Koschke14', 'Koschke', 1);
-- citekey: 'OMG14' nicht in aut, 'Koschke14' nicht in pub

select * from pub natural full outer join aut;

select * from aut natural join pub;

-- fehlende korrespondierende Werte werden mit dem null-Wert
-- aufgefüllt; null-Werte werden mit Leerzeichen angezeigt; sind in
-- einer Tabelle aber als Wert null vorhanden; weitere Beispiele zu
-- null unten; Vorsicht bei null-Werten: Manche erwartete Gesetze
-- gelten nicht, wenn null-Werte vorhanden sind,
-- z.B. R =
--      select * from R where B   union   select * from R where not B

select * from pub natural left outer join aut;
select * from pub natural right outer join aut;

-- Klassifikation von Joins / Syntaktische Varianten:
-- - Join-Bedingung
--   natural / using / on
--   = auf allen gemeinsamen Attr. / = auf using Attr. / Bedingung
-- - Behandlung von null-Werten
--   inner join / full outer / left outer / right outer
--     keine null-Werte / null-Werte für fehlende Korrespondenzen l+r /
--     alle Tupel aus linker Tabelle / alle Tupel aus rechter Tabelle

-- In älteren Versionen von SQL, z.B. SQL-86, gab es das Schlüsselwort
-- JOIN noch nicht; man kann aber jeden JOIN auch über äquivalente
-- Anfragen (ohne JOIN) ausdrücken; z.B. Anfragen die äquivalent
-- sind zum FULL OUTER JOIN:
-- select * from pub natural full outer join aut;

select * from pub, aut where pub.citekey=aut.citekey
union
  select *, null, null, null from pub
  where citekey not in (select citekey from aut)
union
  select null, null, null, null, null, * from aut
  where citekey not in (select citekey from pub);

delete from pub where citekey='OMG14';
delete from aut where citekey='Koschke14';

------------------------------------------------------------------------
-- mengentheoretische Operationen: Vereinigung, Differenz, Durchschnitt
------------------------------------------------------------------------

drop view if exists author_article cascade;
create view author_article as select distinct author from aut
where citekey in (select citekey from pub where pubtype='article');

drop view if exists author_inproc cascade;
create view author_inproc as select distinct author from aut
where citekey in (select citekey from pub where pubtype='inproc');

select * from author_article;

select * from author_inproc;

select * from author_article
union
select * from author_inproc;

select * from author_article
except
select * from author_inproc;

select * from author_article
intersect
select * from author_inproc;

------------------------------------------------------------------------
-- null-Werte
------------------------------------------------------------------------
```

```
select * from aut; select * from pub;

update aut set pos=null where citekey='GogollaV12';
update pub set year=null where title like 'UML%';

select * from aut; select * from pub;

select author from aut where pos is null;

select author from aut where pos=null;

select avg(year) from pub;

select avg(year) from pub where year is not null;

select 0 is null, 0.0 is null, '' is null, ' ' is null;

select 0=null, 0.0=null, ''=null, ' '=null;

select (0=null) is null, (0.0=null) is null,
       (''=null) is null, (' '=null) is null;

select not(false), not(true), not(null),
       null=null, null=false, null=true;

select * from pub where year<2000
union
select * from pub where year>=2000;

select * from pub;

update aut set pos=1 where citekey='GogollaV12' and author='Gogolla';
update aut set pos=2 where citekey='GogollaV12' and author='Vallecillo';
update pub set year=2011 where title like 'UML%';

------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------
DROP TABLE IF EXISTS pub CASCADE;
CREATE TABLE pub(
  citekey VARCHAR,
  title   VARCHAR,
  venue   VARCHAR,
  year    INTEGER,
  pubtype VARCHAR);

DROP TABLE IF EXISTS aut CASCADE;
CREATE TABLE aut(
  citekey VARCHAR,
  author  VARCHAR,
  pos     INTEGER);

------------------------------------------------------------------------
\i C:/Users/Gogolla/Desktop/dblp2sql/Gogolla.sql.txt;
------------------------------------------------------------------------

select count(*) from pub;

select count(*) from aut;
select title from pub where citekey in
  (select citekey from aut where author='Antonio Vallecillo');

select title from pub where pub.citekey in
  (select aut.citekey from aut where author='Antonio Vallecillo');

select title from pub p where p.citekey in
  (select a.citekey from aut a where a.author='Antonio Vallecillo');

-------------------------------------------------------------------------
-- group by, having, order by in einem größeren Datenbankzustand
-------------------------------------------------------------------------

select   year, count(*)
from     pub
group by year
order by 2,1;

select   year, count(*)
from     pub
group by year
having   count(*)>3
order by 1;

select   year, count(*)
from     pub
where    citekey in (select citekey from aut where author='Mirco Kuhlmann')
group by year
having   count(*)>3
order by 1;

select   pubtype, year, count(*)
from     pub
group by pubtype, year
having   count(*)>=3
order by year, pubtype;

-------------------------------------------------------------------------
-- komplexe Anfrage (in einem größeren Datenbankzustand)
-- Titelpaare (t1,t2) mit gleicher Autorenmenge [t1<t2] und
--                          |Autorenmenge|>1;
-------------------------------------------------------------------------

-- Man beachte: verbale Formulierung der Anfrage verwendet den Begriff
-- Autorenmenge, nicht Autorenliste; führt zu hoher Komplexität, wenn
-- diese Anfrage in der Ein-Tabellen-Version gestellt wird

select substr(p1.title,1,40), substr(p2.title,1,40)
from   pub p1, pub p2
where  p1.title<p2.title and
       (select count(*) from aut where aut.citekey=p1.citekey)>1 and
       -- Autorenmenge von p1/p2:
       -- AM1 = [select author from aut where aut.citekey=p1.citekey]
       -- AM2 = [select author from aut where aut.citekey=p2.citekey]
       --
       -- [AM1 teilmengeGleich AM2] und [AM2 teilmengeGleich AM1]
       --
       -- [nichtExistiert (a in AM1 and a not in AM2)] und
       -- [nichtExistiert (a in AM2 and a not in AM1)]
       not exists (select author from aut
                   where  aut.citekey=p1.citekey and
                          author not in (select author from aut
                                         where  aut.citekey=p2.citekey))
       and
       not exists (select author from aut
                   where  aut.citekey=p2.citekey and
                          author not in (select author from aut
                                         where  aut.citekey=p1.citekey))
order by 1,2;

select author, pos from aut where citekey in
  (select citekey from pub where title like 'On Constraints and%');

select author, pos from aut where citekey in
  (select citekey from pub where title like 'On Formalizing the%');

-- Für Term T und Menge M gilt:
-- 'T in M' == 'T =any M'; 'T not in M' == 'T <>all M'

select substr(p1.title,1,40), substr(p2.title,1,40)
from   pub p1, pub p2
where  p1.title<p2.title and
       (select count(*) from aut where aut.citekey=p1.citekey)>1 and
       not exists(select author from aut
                  where  aut.citekey=p1.citekey and
                         author <>all(select author from aut
                                      where  aut.citekey=p2.citekey))
       and
       not exists(select author from aut
                  where  aut.citekey=p2.citekey and
                         not author =any(select author from aut
                                         where  aut.citekey=p1.citekey))
order by 1,2;

------------------------------------------------------------------------
------------------------------------------------------------------------
------------------------------------------------------------------------

-- Arten von SQL-Constraints (SQL-Integritätsbedingungen):
--
-- Primärschlüssel (und UNIQUE): definiert Attributmenge, die ein
--                               Tupel eindeutig bestimmt
-- Fremdschlüssel: Attributmenge, die auf den Primärschlüssel eines
--                 anderen Tupels verweist
-- Check-Constraint: einfache Einschränkungen für Attribute und Tupel;
--                   Funktionsaufrufe erlaubt in PostgreSQL
-- Not-Null-Constraint: Verbot von null-Werten

-- Kern-Syntax für Constraints in CREATE TABLE
-- - Not-Null-Constraint: <attribut> <datatype> NOT NULL
-- - Unique-Constraint: UNIQUE (<attribut*>)
-- - Primary-Key-Constraint: PRIMARY KEY (<attribute*>)
--                           PRIMARY KEY = UNIQUE + NOT NULL
-- - Foreign-Key-Constraint:
--   FOREIGN KEY (<attribute*>) REFERENCES <table> (<attribute*>)
-- - Check-Constraint: CHECK (<condition-for-tuple>)
--                     bezogen auf ein Tupel

-- Primary-Key-, Unique- und Foreign-Key-Constraints als Formeln:
```

```
--
-- create table R(a dt1, b dt2, primary key(a))
-- forall R r1, R r2 ( r1<>r2 implies r1.a<>r2.a )
-- forall R r ( r.a is not null )
--
-- Notation angelehnt an SQL; forall, implies NICHT in SQL vorhanden
--
-- create table S(e dt3, f dt4, foreign key (f) references R(a))
-- forall S s exists R r ( s.f=r.a )
--
-- create table R(a dt1, b dt2, c dt3, primary key(b,c))
-- forall R r1, R r2 ( r1<>r2 implies (r1.b<>r2.b or r1.c<>r2.c) )
-- forall R r ( r.b is not null and r.c is not null )

-- table R in SQL auch als: create table R(a dt1 primary key, b dt2)

-- Tabellen mit Primärschlüssel-Constraints entsprechen Relationen,
-- d.h. Tupelmengen (dann keine Duplikate in der Tabelle erlaubt)

------------------------------------------------------------------------

DROP TABLE IF EXISTS pub CASCADE;
CREATE TABLE pub(
  citekey VARCHAR PRIMARY KEY,
  title   VARCHAR NOT NULL,
  venue   VARCHAR NOT NULL,
  year    INTEGER NOT NULL,
  pubtype VARCHAR NOT NULL);

DROP TABLE IF EXISTS aut CASCADE;
CREATE TABLE aut(
  citekey VARCHAR NOT NULL,
  author  VARCHAR NOT NULL,
  pos     INTEGER NOT NULL CHECK (pos>0),
  PRIMARY KEY(citekey,author),
  UNIQUE(citekey,pos),
  FOREIGN KEY (citekey) REFERENCES pub(citekey));

------------------------------------------------------------------------

-- forall pub p1, pub p2 ( p1<>p2 implies p1.citekey<>p2.citekey )
-- forall pub p ( p.citekey is not null )
--
-- forall aut a1, aut a2
--   ( a1<>a2 implies (a1.citekey<>a2.citekey or a1.author<>a2.author) )
-- forall aut a ( a.citekey is not null and a.author is not null)
--
-- forall aut a1, aut a2
--   ( a1<>a2 implies (a1.citekey<>a2.citekey or a1.pos<>a2.pos) )
--
-- forall aut a ( a.pos>0 )
-- forall aut a ( exists pub p ( a.citekey=p.citekey ) )

-- pub                                     aut
-- citekey     | title       |  |  |    citekey     | author     | pos
-- ------------+-------------+--+--+--  ------------+------------+-----
-- GogollaV12  | Model Princ |  |  |    GogollaV12  | Gogolla    |  1
-- KuhlmannG11 | UML and OCL |  |  |    GogollaV12  | Vallecillo |  2
-- KuskeGK09   | Graph Seman |  |  |    KuhlmannG11 | Kuhlmann   |  1
--                                      KuhlmannG11 | Gogolla    |  2
--                                      KuskeGK09   | Kuske      |  1
--                                      KuskeGK09   | Gogolla    |  2
--                                      KuskeGK09   | Kreowski   |  3

---------------------------------------------------------------------------

DROP FUNCTION IF EXISTS pos_exists(INTEGER,VARCHAR) CASCADE;
CREATE FUNCTION pos_exists(P INTEGER,CK VARCHAR) RETURNS BOOLEAN AS $$
  SELECT EXISTS ( SELECT * FROM aut a
                  WHERE  a.pos=P AND a.citekey=CK ) $$ LANGUAGE SQL;

ALTER TABLE aut ADD CONSTRAINT no_pos_gaps
  -- CHECK ( pos>=2 IMPLIES pos_exists(pos-1,citekey) );
  -- [A IMPLIES B] <=> [NOT(A) OR B]
  CHECK ( NOT(pos>=2) OR pos_exists(pos-1,citekey) );

---------------------------------------------------------------------------
-- forall aut b ( b.pos>=2 implies
--   exists aut a ( a.pos=b.pos-1 and a.citekey=b.citekey ) )
---------------------------------------------------------------------------

INSERT INTO pub VALUES ('GogollaV12',
  'Model Princ', 'EduSym 2012', 2012, 'inproc');
INSERT INTO aut VALUES ('GogollaV12', 'Gogolla', 1);
INSERT INTO aut VALUES ('GogollaV12', 'Vallecillo', 2);

INSERT INTO pub VALUES ('KuhlmannG11',
  'UML and OCL', 'MoDELS 2011', 2011, 'inproc');
INSERT INTO aut VALUES ('KuhlmannG11','Kuhlmann', 1);
INSERT INTO aut VALUES ('KuhlmannG11','Gogolla', 2);

INSERT INTO pub VALUES ('KuskeGK09',
  'Graph Seman', 'SoSyM 2009', 2009, 'article');
INSERT INTO aut VALUES ('KuskeGK09', 'Kuske', 1);
INSERT INTO aut VALUES ('KuskeGK09', 'Gogolla', 2);
INSERT INTO aut VALUES ('KuskeGK09', 'Kreowski', 3);

---------------------------------------------------------------------------
-- Beispiele für Anweisungen, die die Integrität verletzen;
-- Primärschlüssel, Fremdschlüssel, Check-Constraint, NotNull-Constraint
---------------------------------------------------------------------------

INSERT INTO aut VALUES ('KuskeGK09', 'Kreowski', 3);

---------------------------------------------------------------------------

INSERT INTO pub VALUES (NULL,
  'Intro Rel DB', 'JACM', 1973, 'article');

---------------------------------------------------------------------------

INSERT INTO pub VALUES ('GogollaV12',
  'More Modeling Principles.', 'EduSym 2012:28-31', 2012, 'inproc');

---------------------------------------------------------------------------

INSERT INTO aut VALUES ('GogollaV13', 'Gogolla', 1);

---------------------------------------------------------------------------
```

```
INSERT INTO aut VALUES ('GogollaV12', 'Kuhlmann', 0);

------------------------------------------------------------------------

INSERT INTO aut VALUES ('KuskeGK09', 'Vallecillo', 5);

------------------------------------------------------------------------

UPDATE aut SET pos=NULL WHERE author='Gogolla';

------------------------------------------------------------------------

ALTER TABLE pub ADD CONSTRAINT pubtype_key UNIQUE (pubtype);

------------------------------------------------------------------------

-- Index: Zusätzliche Datenstruktur für den effizienten Zugriff
-- bei Datenmanipulation durch Datenbanksystem aktualisiert
-- Primärschlüssel und Unique-Deklarationen erzeugen implizit Indexe
-- ein eindeutiger Index (unique) verweist auf genau ein Tupel

-- automatisch durch obige Tabellendefinitionen erzeugt:
-- create unique index pub_pkey on pub(citekey);
-- create unique index aut_pkey on aut(citekey,author);
-- create unique index aut_citekey_pos_key on aut(citekey,pos);

------------------------------------------------------------------------

drop index if exists pub_year cascade;
drop index if exists pub_pubtype_year cascade;
drop index if exists aut_pos cascade;

create index pub_year on pub(year);
create index pub_pubtype_year on pub(pubtype,year);
create index aut_pos on aut(pos);

-- aut   citekey   |   author   | pos      aut_pos pos | tuple set
--     ------------+------------+-----            ----+---------------
-- t42 GogollaV12  | Gogolla    |   1             1   | {t42,t48,t54}
-- t45 GogollaV12  | Vallecillo |   2             2   | {t45,t51,t57}
-- t48 KuhlmannG11 | Kuhlmann   |   1             3   | {t60}
-- t51 KuhlmannG11 | Gogolla    |   2
-- t54 KuskeGK09   | Kuske      |   1
-- t57 KuskeGK09   | Gogolla    |   2
-- t60 KuskeGK09   | Kreowski   |   3
--
-- tXY Tupelidentifier, nicht explizit zugreifbar für Benutzer

------------------------------------------------------------------------
```