实习5

空间数据库设计

实习5

- 写SQL语句使用数据库→设计数据库 (思考、分析)
  - 实际应用中的函数依赖、数据存储和查询效率
- 主要问题
  - 空间扩展ER图的联系类型和实体象形图
  - 空间属性的函数依赖和主键设计
  - ER图转换为关系时，联系合并和子类设计
  - 关系的函数依赖指定
    - 人的先验知识和假设(人的智能)
    - 从数据中自动发现规律(数据智能)
  - 多源数据融合，即数据ETL
    - 关联分析，不能丢失信息，即保证lossless
  - 多值依赖的判断

数据ETL / Data Wrangling

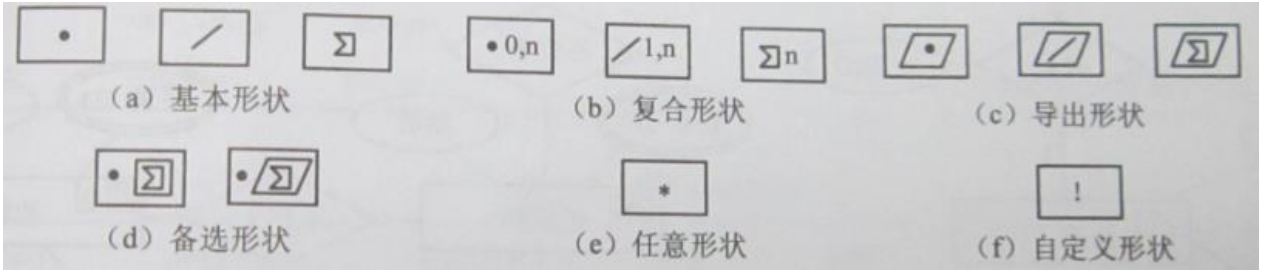
- Enterprise Data Analysis and Visualization: An Interview Study
  - 至少50%的时间在数据ETL上
  - “I spend more than half of my time integrating, cleansing and transforming data without doing any actual analysis. Most of the time I’m lucky if I get to do any analysis. Most of the time once you transform the data you just do an average... the insights can be scarily obvious. It’s fun when you get to do something somewhat analytical.”
- Data Wrangling with MongoDB
  - https://www.udacity.com/course/data-wrangling-with-mongodb--ud032
  - 70% in data wrangling (Gathering, Extracting, Cleaning, Storing)

数据库设计原则

- 尽可能避免设计异常
  - 数据冗余
  - 更新异常
  - 插入异常
  - 删除异常
- 实际使用时数据访问效率
- 解决方法
  - 消除函数依赖和多值依赖，但需避免过分解
  - 考虑常用的查询，尽可能把相关属性存储在同一关系中

空间扩展E-R图

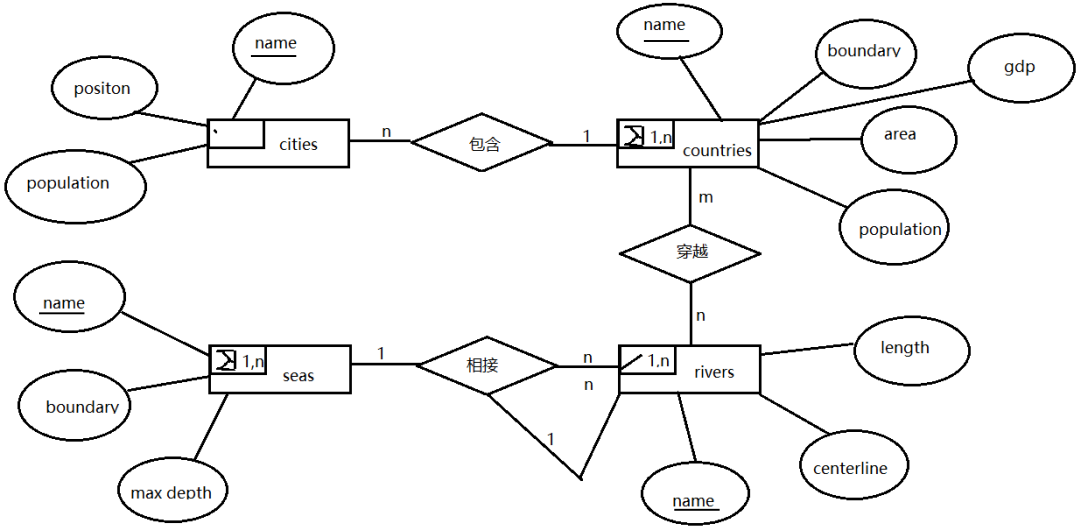
- 实体象形图
  - 点、线、面等
- 实体
  - Countries：MultiPolygon
    - Boundary是几何属性，area是数值属性
  - Cities：Point / MultiPolygon
    - 不同尺度，备选形状
  - Rivers：MultiLineString
    - PostGIS几何类型没有MultiCurve类型
  - Seas：MultiPolygon / Polygon
    - 实际海洋可能是一个联通区域，Polygon可以有多个内边界



6类几何类型： Point, LineString, Polygon, Multixxx

空间扩展E-R图

- 联系类型
  - 1 : 1，1 : n，m : n
- 联系
  - countries : cities = 1 : n
  - countries : rivers = m : n
  - rivers : rivers = 1 : n
  - rivers : seas = 1 : n



空间扩展E-R图

- 实体函数依赖，都属于BCNF

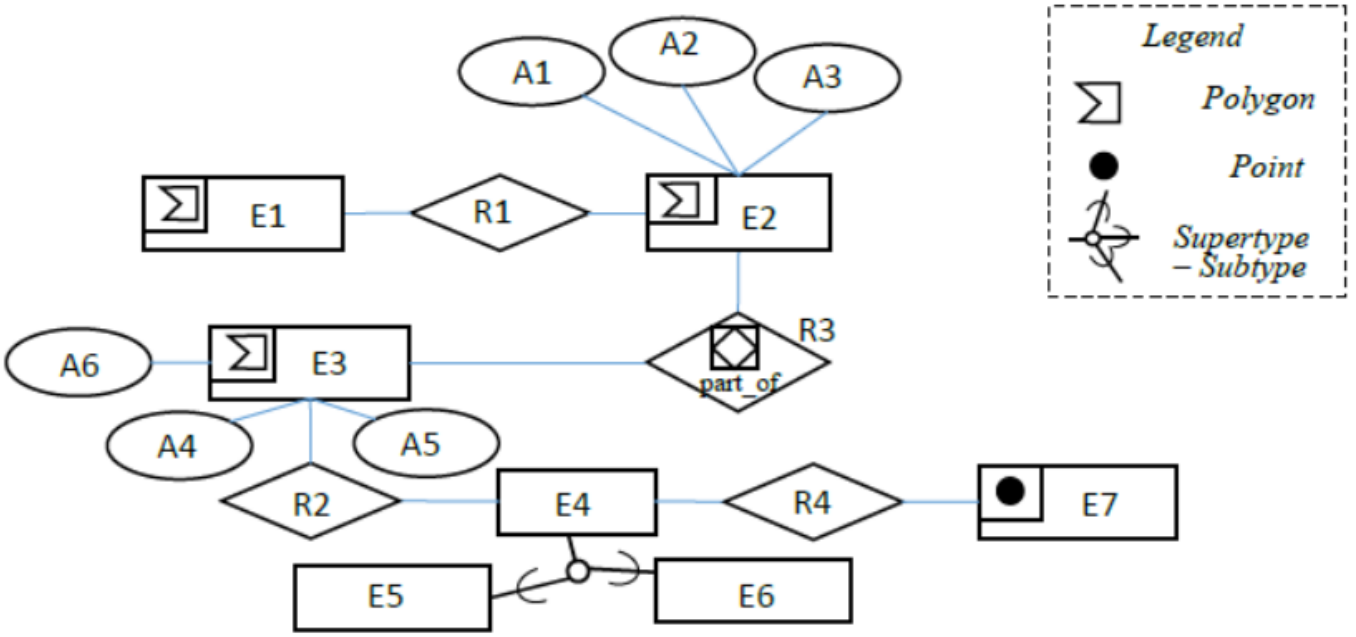
几何属性作为主键？

  - Countries
    - name → boundary, area, population, gdp
    - boundary → name, area, population, gdp
  - Cities
    - name → position, population
      - 取决于name具体含义，可以有重名，该函数依赖不成立
    - position → name, population
  - Rivers
    - name → centerline, length
    - centerline → name, length
  - Seas
    - name → boundary, max depths
    - boundary → name, max depths

name → all attributes,
但boundary → area时，
是否属于BCNF？

National Turfgrass Evaluation Program

- 联系
  - R1(E1(University)→E2(Site)) Administrate 1:N
  - R2(E3(Parcel)→E4(Grass)) Grow N:1
  - R3(E2(Site)→E3(Parcel)) Is_Part_of 1:N
  - R4(E4(Grass)→E7(Sponsor)) Provide N:1



National Turfgrass Evaluation Program

联系为什么和实体合并？

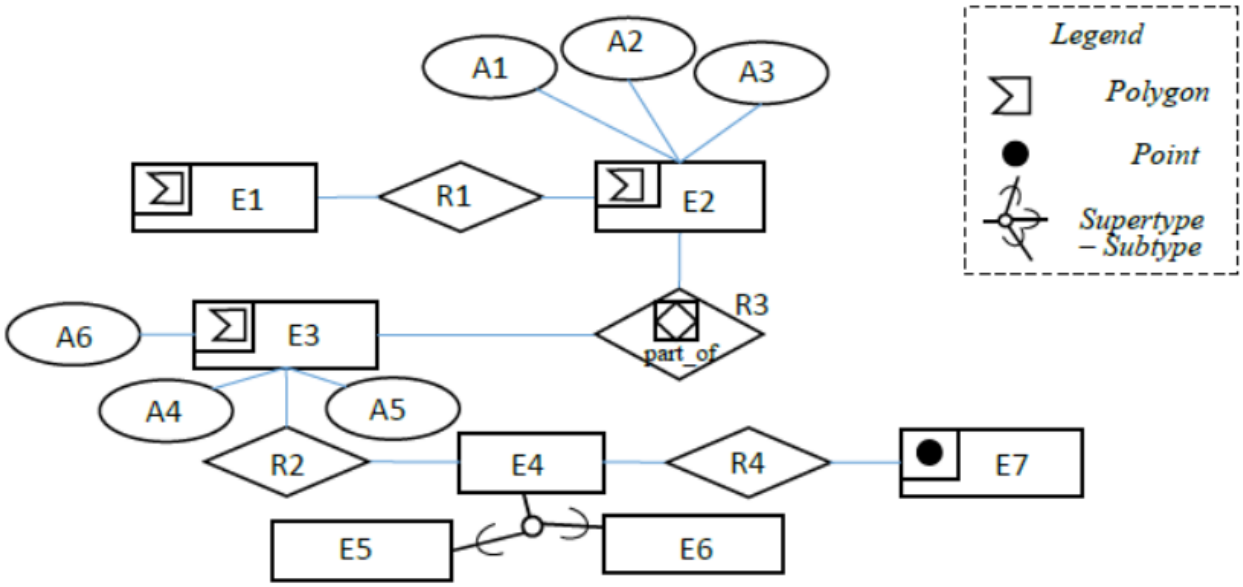
- 实体 → 关系
  - 主码 → 主码
  - 属性 → 属性
  - 复合属性 → 若干原子属性
  - 多值属性 → 关系 或 转化为联系
  - 实体象形图 → Geometry类型
- 联系 (如何确定主码？)
  - 1:1 → 关系 或 与非强制性的实体合并
  - 1:n → 关系 或 与n端的实体合并
  - m:n → 关系
  - IsA联系 → 三种方法
  - part-of联系 → ???

IsA to Relation Schema

- IsA联系描述了实体型之间的继承关系。在关系模型中仍然使用关系表示IsA联系。一般情况下，父实体和各子实体分别用独自的关系表示，表示父实体的关系属性包括所有父实体的属性，子实体对应的关系除了包含各自的属性外，还必须包含父实体的主键
- 如果IsA联系满足不相交约束，也可以用一个关系表示父实体和所有的子实体
- 如果IsA联系满足完备性约束，也可以去除表示父实体的关系，但是父实体的所有属性在每个子实体的关系中都必须出现

National Turfgrass Evaluation Program

- 联系合并
  - R1与E2，R2与E3，R3与E3，R4与E4
  - Site(code, location(Polygon), precipitation, university_id)
  - E3需要E4的主键，E3需要E2的主键，E4需要E7的主键
- 子类定义
  - Grass(grass_id, type, name, sponsor_id)



BC范式

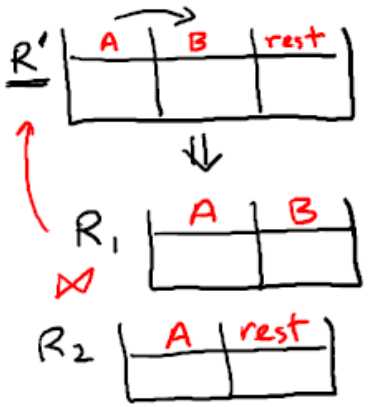
- 关系R(A, B, C, D, E)具有以下函数依赖：AB→C, BC→D, CD→E, DE→A
- A key is a minimal set of attributes that uniquely identifies an entity
  - R($\bar{A}$, $\bar{B}$), suppose $\bar{A}$ → all attributes，$\bar{A}$是关系R的码
- R的码: AB和BC
  - $(AB)^+$ = ABCDE
  - $(BC)^+$ = ABCDE
  - $(CD)^+$ = ACDE
  - $(DE)^+$ = ADE
  - BDE不是关系R的码

BC范式

BCNF decomposition algorithm

Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"



Compute keys for R (using FDs)

Repeat until all relations are in BCNF:

- Pick any R' with $\bar{A} \rightarrow \bar{B}$ that violates BCNF
- Decompose R' into $R_1(\bar{A}, \bar{B})$ and $R_2(\bar{A}, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2

BC范式

- 关系R(A, B, C, D, E)具有以下函数依赖：AB→C, BC→D, CD→E, DE→A
- 违背BC范式：CD→E, DE→A
- 选择CD→E进行分解
  - R1(C, D, E)
    - R1的函数依赖为CD→E，且码为CD，R1属于BC范式
  - R2(A, B, C, D)
    - R2的函数依赖为AB→C, BC→D, CD→A, R2的码AB和BC
    - R2基于违背BC范式的CD→A进行分解
      - R3(C, D, A)和R4(C, D, B)
      - R3的函数依赖CD→A，且码为CD，R3属于BC范式
      - R4的函数依赖BC→D，且码为BC，R4属于BC范式

BC范式

- 关系R(A, B, C, D, E)具有以下函数依赖：AB→C, BC→D, CD→E, DE→A
- 违背BC范式：CD→E, DE→A
- 选择DE→A进行分解
  - R1(D, E, A)
    - R1的函数依赖为DE→A，且码为DE，R1属于BC范式
  - R2(D, E, B, C)
    - R2的函数依赖为BC→D, CD→E，且码为BC
    - R2基于违背BC范式的CD→E进行分解
      - R3(C, D, E)和R4(C, D, B)
      - R3的函数依赖为CD→E，且码为CD，R3属于BC范式
      - R4的函数依赖为BC→D，且码为BC，R4属于BC范式

BC范式

- 关系R(A, B, C, D, E)具有以下函数依赖：AB→C, BC→D, CD→E, DE→A
- 一种BC范式分解
  - R1 (C, D, E)，R3(C, D, A)和R4(C, D, B)
- 另一种BC范式分解
  - R1(D, E, A)，R3(C, D, E)和R4(C, D, B)
- 第8章 P66 选择不同R'进行分解，BC范式分解结果可能不同

函数依赖

- 关系R中，属性集A和属性集B存在函数依赖的定义(t,u是关系R的元组)：

$$\forall t, u \in R:$$
$$t[A_1, \dots A_n] = u[A_1, \dots A_n] => \ t[B_1, \dots B_m] = u[B_1, \dots B_m]$$

- 函数依赖分类
  - 平凡函数依赖 (Trivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \subseteq \bar{A}$
  - 非平凡函数依赖 (Nontrivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \subseteq \bar{A}$
  - 完全非平凡函数依赖 (Completely nontrivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \cap \bar{A} = \emptyset$

函数依赖

- 函数依赖规则(Armstrong's Rules)
  - 分解规则(Splitting rule)
    - $\bar{A} \rightarrow B_1, \ldots B_m$
    - $\bar{A} \rightarrow B_1, \bar{A} \rightarrow B_2, \ldots, \bar{A} \rightarrow B_m$
  - 合并规则 (Combining rule)
  - 平凡依赖规则 (Trivial-dependency rules)
  - 传递规则 (Transitive rule)

分解规则→右边单属性

- 如何为关系指定函数依赖？
  - Minimal set of completely nontrivial FDs such that all FDs that hold on the relation follow from the dependencies in the set
- How to test - Does $\bar{A} \rightarrow \bar{B}$ follow from S?
  - Compute $\bar{A}^+$ base S check if $\bar{B}$ in set

函数依赖判断

- A → B
  - R(A, B, C, D)，函数依赖与其他属性无关
  - 基于函数依赖的定义判断

```
select R1.*
from R R1, R R2
where R1.A = R2.A and R1.B <> R2.B

select *
from R
group by A
having count(B) > 1
```

函数依赖判断

```
columns = ['A', 'B', 'C', 'D']

template = 'select R1.* from R R1, R R2 where R1.% = R2.%
and R1.% <> R2.%'

for A in columns:
    for B in columns:
        query = template % (A, A, B, B)
        result = %sql $query
        if A != B and len(result) == 0:
            print A + ' → ' + B
```

函数依赖判断

```
colums = ['A', 'B', 'C', 'D']
template = 'select * from R R1,R R2 where R1.%s=R2.%s and R1.%s=R2.%s and R1.%s!=R2.%s'

for A in colums:
    for B in colums:
        for C in colums:
            if A < B  and A !=C and B != C:
                query = template % (A, A, B, B, C, C)
                result = %sql $query
                if len(result) == 0:
                    print A + B + '->' + C
```

函数依赖判断

- R(A, B, C, D)，28个→14个
  - 单属性之间的函数依赖，12个
    - 获得A → B, C → D两个函数依赖
  - 双属性与单属性之间的函数依赖，12个→ 4个→ 2个
    - AB → C, AB → D
    - AC → B, AC → D
    - AD → B, AD → C
    - BC → A, BC → D
    - BD → A, BD → C
    - CD → A, CD → B

    A → C, A → D
    A → B, C → D
    A → B
    , C → D

    BC→A,AD→C
    C → A, C → B
  - 三属性与单属性之间的函数依赖，4个→ 0个
    - ABC → D, ABD → C, ACD → B, BCD → A
  - 与右边双属性/三属性之间的函数依赖 (已隐含在前面)
    - A → BC 等价于 A → B和A→ C

函数依赖判断

- R(A, B, C, D)的函数依赖
  - A→B, C→D
    - 关系R的最小函数依赖集合
- 设计异常
  - 基于A→B, C→D分析是否包含数据冗余、更新异常、插入异常、和删除异常
- BCNF
  - R1(A, B), R2(C, D)
    - 非BCNF分解，非join lossess，即行数不同
  - R1(A, B), R2(A, C, D)
    - R2存在函数依赖C→D，但C^+={C,D}
  - R1(A, B), R3(C, D), R4(C, A)

CurrentTrack视图

```
create view CurrentTrack as
select carid, position, b.id as roadID
from track as a, road as b
where  ST_Distance(position, geom) <= all(select
ST_Distance(position, c.geom) from road as c)
and time = (select max(time) from track as d where
d.carid = a.carid);

select roadID, count(carID)
from currentTrack
group by roadID
order by count(carID) desc
```

CurrentTrack视图

```
create view CurrentTrack as
select carid, position, b.id as roadID
from track as a, road as b
where  b.id = (select c.id from road as c order by
ST_Distance(a.position, c.geom) limit 1)
and time = (select max(time) from track d where
d.carid = a.carid);

select roadID, count(carID)
from currentTrack
group by roadID
order by count(carID) desc
```

CurrentTrack视图

- SQL standard for “updatable views” (SQL标准)
  - Select (no Distinct) on single table T
  - Attributes not in view can be ‘NULL’ or have default value
  - Subqueries must not refer to T
  - No Group by or aggregation
- 不同数据库系统有自己的标准
  - SQL Server
  - PostgreSQL

CurrentTrack视图

- PostgreSQL标准 (特定数据库标准)
  - The view must have exactly one entry in its FROM list, which must be a table or another updatable view
  - The view definition must not contain WITH, DISTINCT, GROUP BY, HAVING, LIMIT, or OFFSET clauses at the top level
  - The view definition must not contain set operations (UNION, INTERSECT or EXCEPT) at the top level
  - All columns in the view's select list must be simple references to columns of the underlying relation. They cannot be expressions, literals or functions. System columns cannot be referenced, either.
  - No column of the underlying relation can appear more than once in the view's select list
  - The view must not have the security_barrier property

https://www.postgresql.org/docs/current/static/sql-createview.html

CurrentTrack视图

- SQL Server标准 (特定数据库标准)
  - 任何修改（包括 UPDATE、INSERT 和 DELETE 语句）都只能引用一个基表的列。
  - 视图中被修改的列必须直接引用表列中的基础数据。 不能通过任何其他方式对这些列进行派生，如通过以下方式
    - 聚合函数：AVG、COUNT、SUM、MIN、MAX、GROUPING、STDEV、STDEVP、VAR 和 VARP
    - 计算不能从使用其他列的表达式中计算该列。使用集合运算符UNION、UNION ALL、CROSSJOIN、EXCEPT和INTERSECT形成的列将计入计算结果，且不可更新。
  - 被修改的列不受 GROUP BY、HAVING或DISTINCT 子句的影响
  - TOP 在视图的 select_statement 中的任何位置都不会与 WITH CHECK OPTION 子句一起使用

CurrentTrack视图

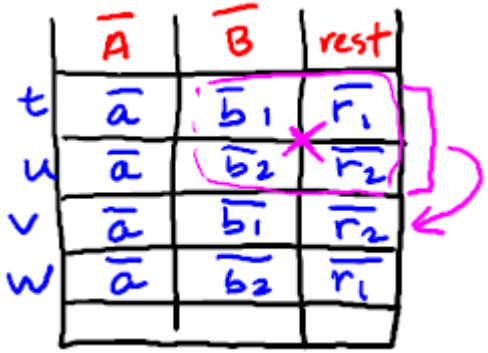
- CurrentTrack在SQL标准和PostgreSQL数据库标准下，都是不可更新视图
- Instead of触发器
  - 不能修改userName，需要使用SESSION_USER

```
create or replace function insertTrack() returns trigger as $$
begin
   insert into track(carid, position) values (new.carid, new.position);
   return new;
end;
$$ language plpgsql;

create trigger insertTrack
instead of insert on CurrentTrack
for each row
execute procedure insertTrack();
```

多值依赖

- 关系R中，属性集A和属性集B存在多值依赖的定义(t, u, v是关系R的元组)：



$$\forall t,u \in R: t[\bar{A}] = u[\bar{A}]\ then$$
$$\exists v \in R: v[\bar{A}] = t[\bar{A}]\ and$$
$$v[\bar{B}] = t[\bar{B}]\ and$$
$$v[rest] = u[rest]$$

- 关系R (A, B, C, D)的实例如下图，对于每个多值依赖，使R满足多值依赖需要增加多少个元组？
  - AB →→ C　0
  - CD →→ A　0
  - D →→ C　4
    - (1, 2, 5, 7), (4, 2, 3, 7)
    - (1, 2, 5, 8), (4, 2, 3, 8)

A	B	C	D
1	2	3	7
1	2	3	8
4	2	5	7
4	2	5	8

多值依赖判断

- AC →→ B
  - R(A, B, C, D)，多值依赖与其他属性有关

```
select *
from R R1, R R2
where R1.A = R2.A and R1.C = R2.C and not exists
        (select *
         from R R3
         where R3.A = R1.A and
                    R3.B = R1.B and
                    R3.C = R1.C and
                    R3.D = R2.D)
```

多值依赖判断

- AC →→ B
  - R(A, B, C, D)，多值依赖与其他属性有关

```
select R1.A, R1.B, R1.C, R2.D
from R R1, R R2
where R1.A = R2.A and R1.C = R2.C

except

select *
from R
```

多值依赖判断

- B→→AC
  - r3.b = r1.b 或 r3.b = r2.b

```
select distinct r3
from R r1,R r2,R r3
where r1!=r2
and r1.b=r2.b
and r3.a=r1.a and r3.c=r1.c and r3.d=r2.d
```