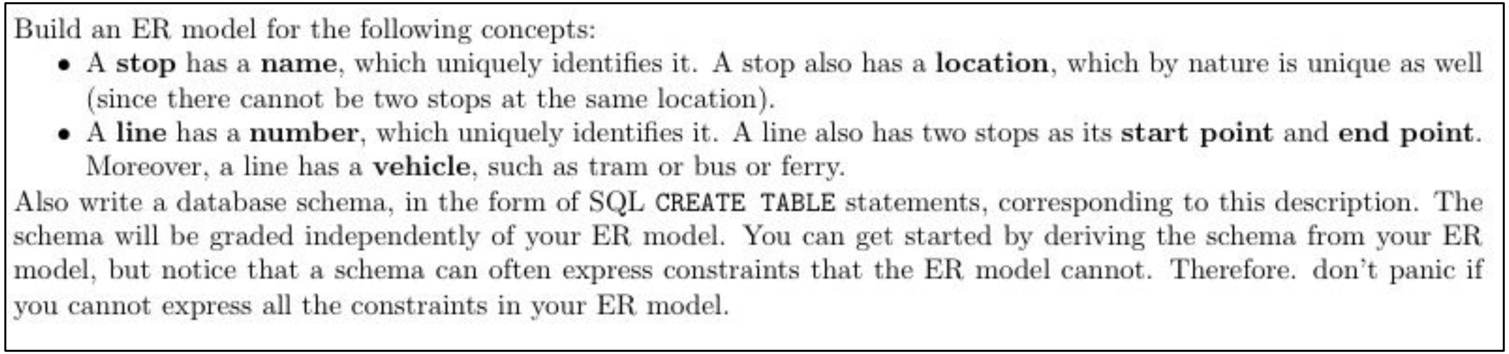
Databases Exam March 2018

TDA357 (Chalmers), DIT620 (University of Gothenburg)

Solutions

Aarne, Alejandro,...

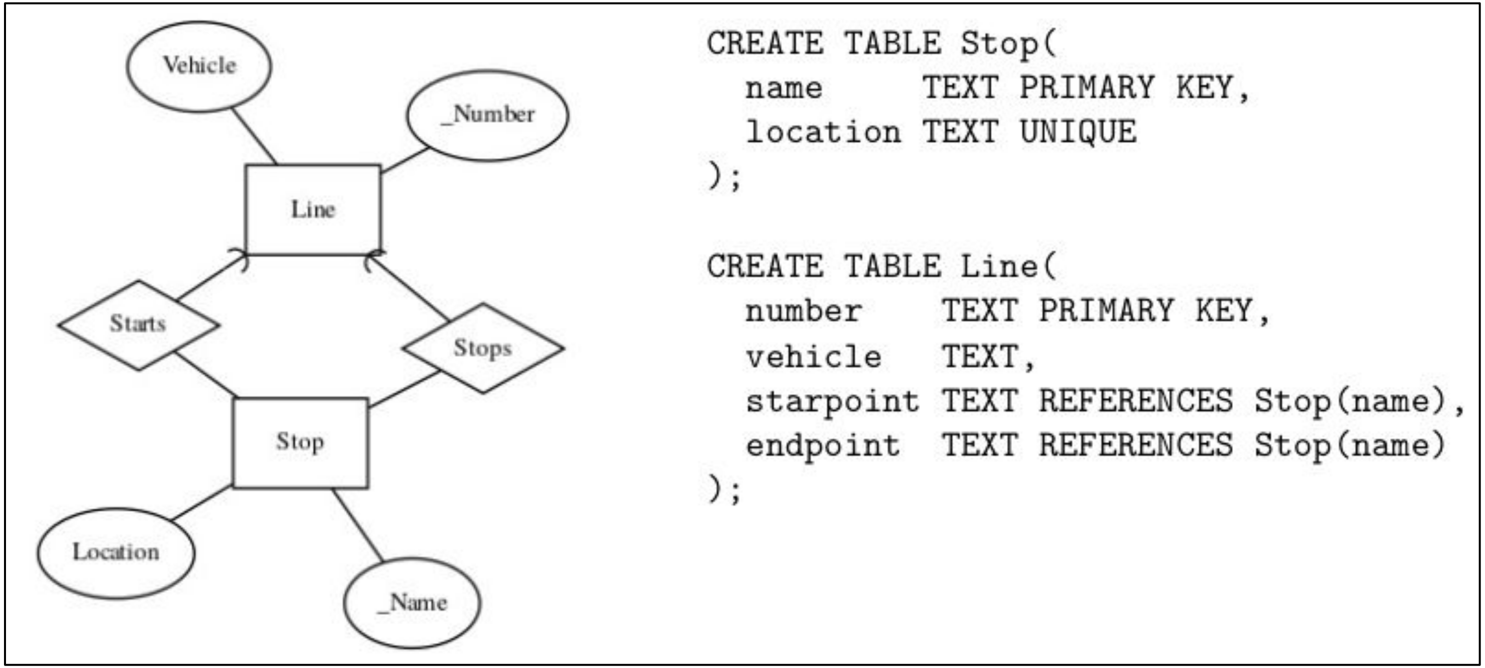
1a (5p)



Build an ER model for the following concepts:

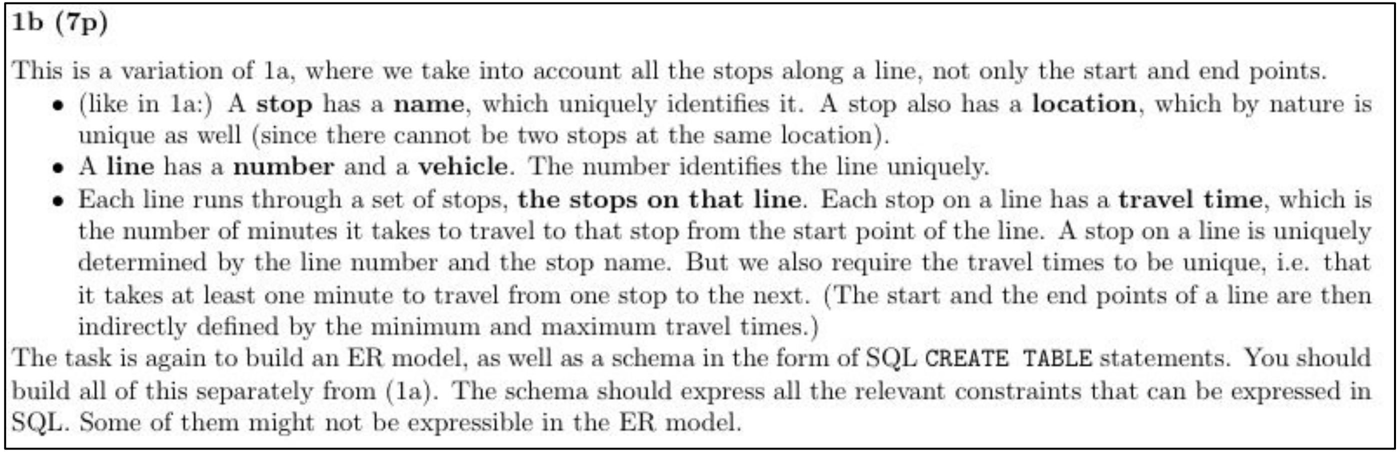
- A **stop** has a **name**, which uniquely identifies it. A stop also has a **location**, which by nature is unique as well (since there cannot be two stops at the same location).
- A **line** has a **number**, which uniquely identifies it. A line also has two stops as its **start point** and **end point**. Moreover, a line has a **vehicle**, such as tram or bus or ferry.

Also write a database schema, in the form of SQL `CREATE TABLE` statements, corresponding to this description. The schema will be graded independently of your ER model. You can get started by deriving the schema from your ER model, but notice that a schema can often express constraints that the ER model cannot. Therefore. don't panic if you cannot express all the constraints in your ER model.



```
CREATE TABLE Stop(
  name     TEXT PRIMARY KEY,
  location TEXT UNIQUE
);

CREATE TABLE Line(
  number    TEXT PRIMARY KEY,
  vehicle   TEXT,
  starpoint TEXT REFERENCES Stop(name),
  endpoint  TEXT REFERENCES Stop(name)
);
```

1b (7p)

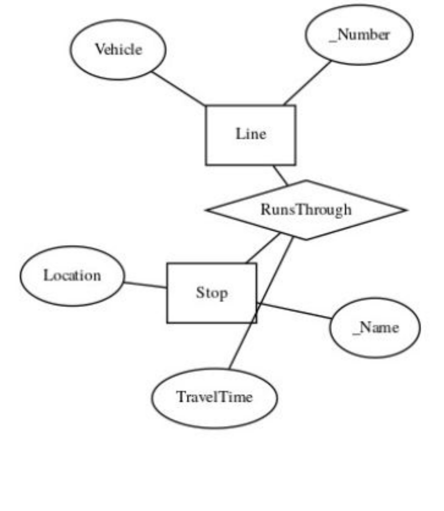


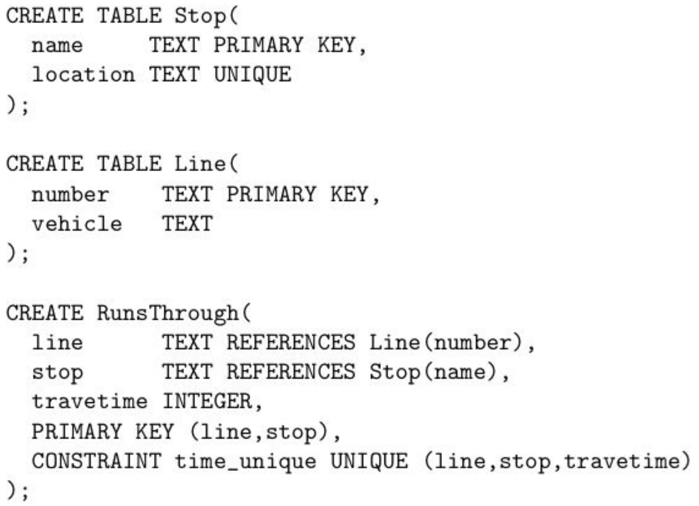
1b (7p)

This is a variation of 1a, where we take into account all the stops along a line, not only the start and end points.

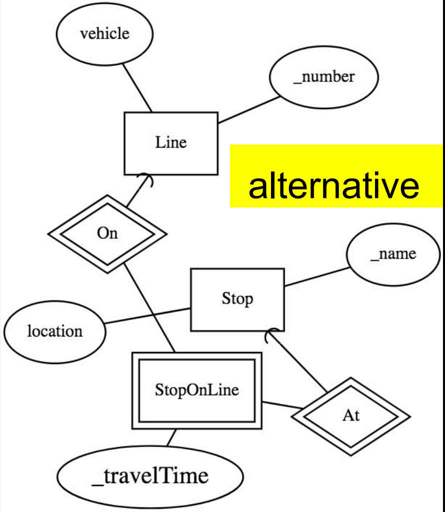
- (like in 1a:) A **stop** has a **name**, which uniquely identifies it. A stop also has a **location**, which by nature is unique as well (since there cannot be two stops at the same location).
- A **line** has a **number** and a **vehicle**. The number identifies the line uniquely.
- Each line runs through a set of stops, **the stops on that line**. Each stop on a line has a **travel time**, which is the number of minutes it takes to travel to that stop from the start point of the line. A stop on a line is uniquely determined by the line number and the stop name. But we also require the travel times to be unique, i.e. that it takes at least one minute to travel from one stop to the next. (The start and the end points of a line are then indirectly defined by the minimum and maximum travel times.)

The task is again to build an ER model, as well as a schema in the form of SQL `CREATE TABLE` statements. You should build all of this separately from (1a). The schema should express all the relevant constraints that can be expressed in SQL. Some of them might not be expressible in the ER model.





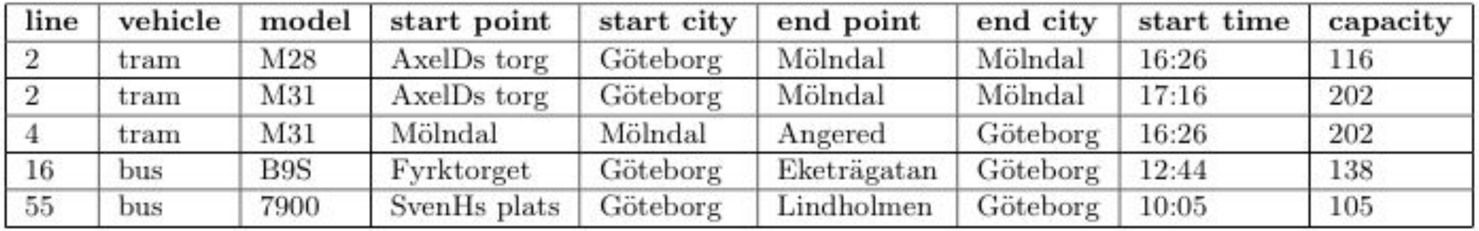
```
CREATE TABLE Stop(
  name     TEXT PRIMARY KEY,
  location TEXT UNIQUE
);

CREATE TABLE Line(
  number    TEXT PRIMARY KEY,
  vehicle   TEXT
);

CREATE RunsThrough(
  line      TEXT REFERENCES Line(number),
  stop      TEXT REFERENCES Stop(name),
  travetime INTEGER,
  PRIMARY KEY (line,stop),
  CONSTRAINT time_unique UNIQUE (line,stop,travetime)
);
```



alternative

Notice: none of the ER models in 1a,b is a precise expression of the UNIQUE constraints.

2a (5p)



line	vehicle	model	start point	start city	end point	end city	start time	capacity
2	tram	M28	AxelDs torg	Göteborg	Mölndal	Mölndal	16:26	116
2	tram	M31	AxelDs torg	Göteborg	Mölndal	Mölndal	17:16	202
4	tram	M31	Mölndal	Mölndal	Angered	Göteborg	16:26	202
16	bus	B9S	Fyrktorget	Göteborg	Eketrägatan	Göteborg	12:44	138
55	bus	7900	SvenHs plats	Göteborg	Lindholmen	Göteborg	10:05	105

Real FDs

startpoint -> startcity
endpoint -> endcity
model -> vehicle capacity
line -> vehicle startpoint endpoint
line starttime -> model

Key:
line starttime

Bogus FDs (no need to list all of these - just a few is enough)

capacity -> vehicle model (*explanation: many models could have the same capacity*)
starttime -> vehicle (*many vehicles could start at the same time*)
starttime capacity -> line startpoint endpoint (*many vehicles of same capacity could start at same time*)
endcity capacity -> line startpoint endpoint starttime (... easy to invent counterexamples to each ...)
endcity starttime -> line model startpoint startcity endpoint capacity
endpoint -> line vehicle startpoint startcity
endpoint capacity -> starttime
endpoint starttime -> model capacity
startcity capacity -> line startpoint endpoint endcity starttime
startcity starttime -> line model startpoint endpoint endcity capacity vehicle
startpoint -> line vehicle endpoint
startpoint capacity -> starttime
startpoint starttime -> model capacity
model starttime -> line startpoint startcity endpoint
model endcity -> line startpoint endpoint starttime
model endpoint -> starttime
model startcity -> line startpoint endpoint starttime
model startpoint -> starttime
vehicle endcity -> startcity
vehicle startcity -> endcity
line capacity -> starttime
line model -> starttime

2b (3p)

```
R: line vehicle startpoint startcity endpoint endcity starttime
      model capacity
violation: line -> vehicle startpoint startcity endpoint endcity
Decomposition:
  R1: line vehicle startpoint startcity endpoint endcity
  Violation: endpoint -> endcity
  Decomposition:
    R11: endpoint endcity
    R12: line vehicle startpoint startcity endpoint
    Violation: startpoint -> startcity
    Decomposition:
      R121: startpoint startcity
      R122: line vehicle startpoint endpoint
  R2: model capacity starttime line
  Violation: model -> capacity
  Decomposition:
    R21: model capacity
    R22: line starttime model
```

endpoint	endcity
Mölndal	Mölndal
Angered	Göteborg
Eketrägatan	Göteborg
Lindholmen	Göteborg

startpoint	startcity
Mölndal	Mölndal
AxelDs torg	Göteborg
Fyrktorget	Göteborg
SvenHs plats	Göteborg

model	capacity
M28	116
M31	202
B9S	138
7900	105

line	starttime	model
2	16:26	M28
2	17:16	M31
4	16:26	M31
16	12:44	B9S
55	10:05	7900

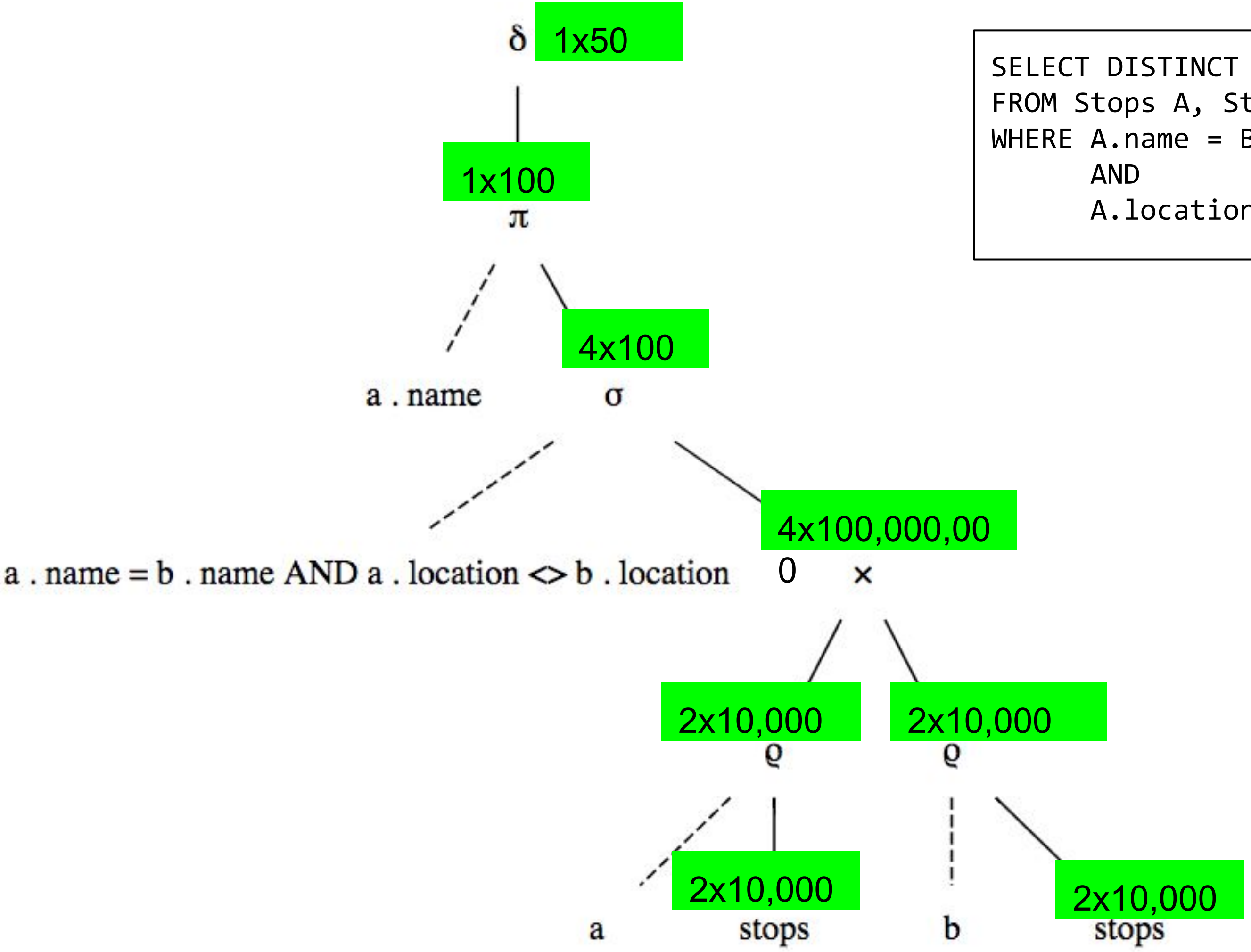
line	vehicle	startpoint	endpoint
2	tram	AxelDs torg	Mölndal
4	tram	Mölndal	Angered
16	bus	Fyrktorget	Eketrägatan
55	bus	SvenHs plats	Lindholmen

3

```
-- 3a (3p) “lines that stop at Chalmers”
SELECT line, vehicle
FROM StopsOnLines, Lines
WHERE stop = 'Chalmers' AND line = Lines.number

-- 3b (4p) “connections from Chalmers to Brunnsparken with exactly one change”
WITH Distances AS (
  SELECT A.stop AS startPoint, B.stop AS endPoint, A.line,
         (B.timeFromStart - A.timeFromStart)) AS minutes
  FROM StopsOnLines A, StopsOnLines B
  WHERE A.line = B.line AND B.timeFromStart > A.timeFromStart
  )
  SELECT A.line, B.line, A.endPoint AS change, A.minutes + B.minutes AS duration
  FROM Distances A, Distances B
  WHERE A.startPoint='Chalmers' AND A.endPoint=B.startPoint AND B.endPoint='Brunnsparken'

-- 3c (5p) “classify stops into cities by the first letter of location code”
(SELECT name AS stop , 'Gothenburg' AS city
 FROM Stops
 WHERE location LIKE 'G%'
   UNION
 SELECT name AS stop, 'Mölndal' AS city
 FROM Stops
 WHERE location LIKE 'M%'
) ORDER BY name
```

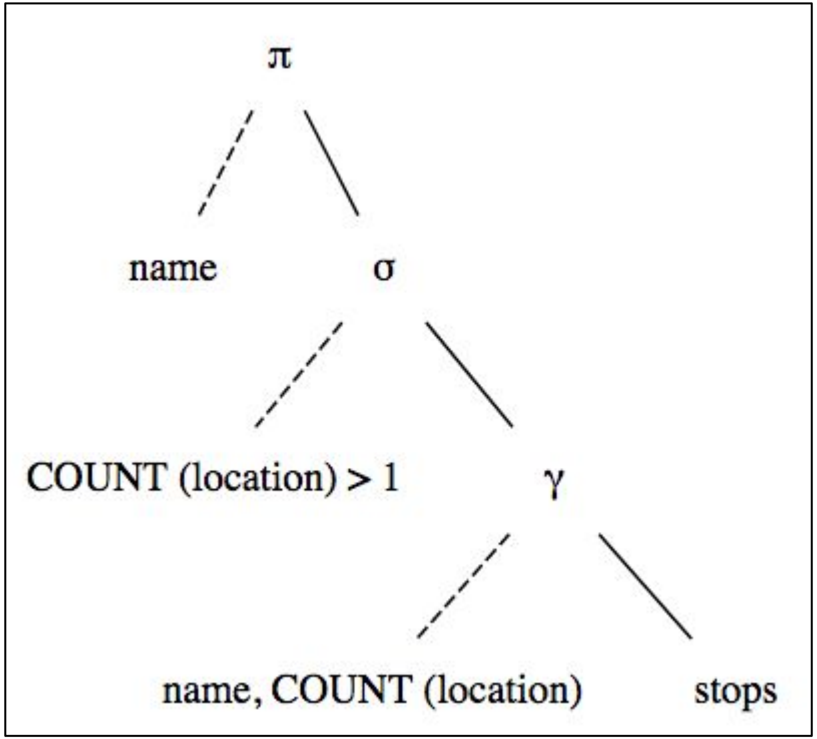
4a,
b

```
SELECT DISTINCT A.name
FROM Stops A, Stops B
WHERE A.name = B.name
      AND
      A.location <> B.location
```

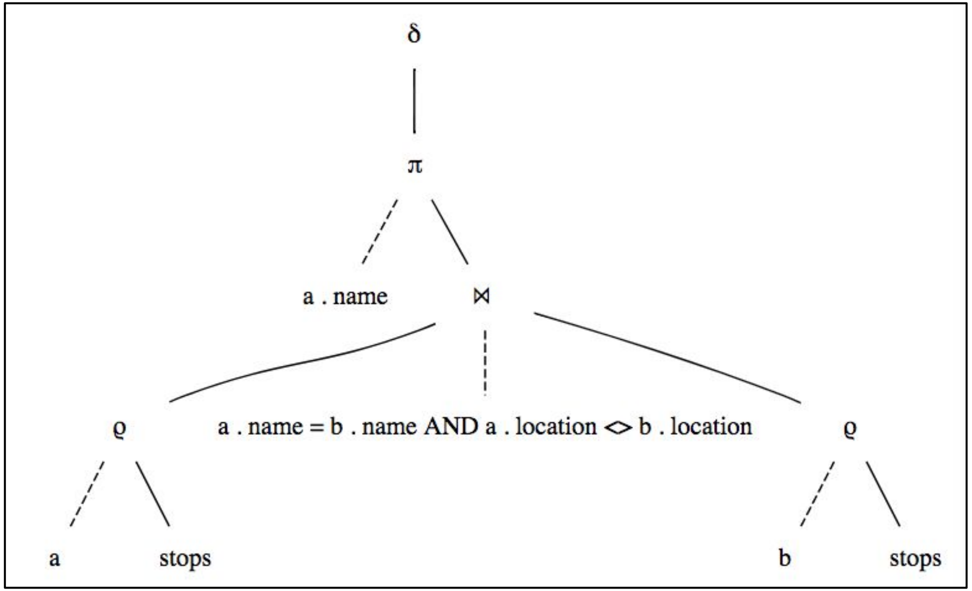


4c

```
SELECT name
FROM stops
GROUP BY name
HAVING COUNT (location)> 1
```



```
-- alternative; avoids the big table
but may need more comparisons

SELECT DISTINCT A.name
FROM Stops A INNER JOIN Stops B ON
  (A.name = B.name
     AND
   A.location <> B.location)
```



5a,b

```
-- 5a “restrict M28,M31 to trams and B9S,7900 to buses”

CREATE TABLE Vehicles (
  vehicletype TEXT,
  model TEXT,
  capacity INT,
  CHECK (model NOT IN ('M28','M31') OR vehicletype = 'tram'),
  CHECK (model NOT IN ('B9S','7900') OR vehicletype = 'bus')
  ) ;
-- the constraints express “if A then B” by “not A or B”

-- 5b “timeFromPrevious is 0 if and only if ordering is 0”

CREATE TABLE StopsAndIntervals (
  line TEXT REFERENCES Lines(number),-- we don’t care about the
referential and key constraints when grading this question
  stop TEXT REFERENCES Stops(name),
  ordering INT,
  timeFromPrevious INT,
  PRIMARY KEY (line,ordering),
  CHECK (ordering <> 1 OR timeFromPrevious = 0),
  CHECK (timeFromPrevious <> 0 OR ordering = 1)
  ) ;
-- the same logic as in 5a, for “if A then B” and “if B then A”
```

5c,d

```
-- 5c “list the times when lines stop at each stop”
CREATE VIEW StopTimes AS (
 SELECT Runs.line AS line, stop, startTime+timeFromPrevious AS time
 FROM StopsOnLines, Runs
 WHERE
  StopsOnLines.line = Runs.line
 ) ;

-- 5d “schedule a line to leave so that it stops at given stop at given time”
CREATE FUNCTION scheduleStopTime () RETURNS TRIGGER AS $$
      BEGIN
       INSERT INTO Runs VALUES (NEW.line, NEW.time -
        (SELECT timeFromPrevious FROM StopsAndIntervals S
         WHERE NEW.stop = S.stop)) ;
       RETURN NEW ;
      END
      $$ LANGUAGE plpgsql ;
CREATE TRIGGER scheduleStopTimeTrigger
       INSTEAD OF INSERT ON StopTimes
       FOR EACH ROW
       EXECUTE PROCEDURE scheduleStopTime () ;
-- alternative solution to 5d forthcoming, corresponding to another
interpretation of the question
```

6, putting all data into elements

```
<?xml version="1.0" encoding="utf-8"
standalone="no"?>
<!DOCTYPE Route [
<!ELEMENT Route (Leg+)>
<!ELEMENT Leg (Dep, Arr, Line)>
<!ELEMENT Dep (Time, Stop, City)>
<!ELEMENT Arr (Time, Stop, City)>
<!ELEMENT Line (Number, Vehicle)>
<!ELEMENT Time (#PCDATA)>
<!ELEMENT Stop (#PCDATA)>
<!ELEMENT City (#PCDATA)>
<!ELEMENT Number (#PCDATA)>
<!ELEMENT Vehicle (#PCDATA)>
]>
```

```
<Route>
  <Leg>
    <Dep>
      <Time>17:02</Time>
      <Stop>Almedal</Stop>
      <City>Göteborg</City>
    </Dep>
    <Arr>
      <Time>17:12</Time>
      <Stop>Mölndal station</Stop>
      <City>Mölndal</City>
    </Arr>
    <Line>
      <Number>2</Number>
      <Vehicle>tram</Vehicle>
    </Line>
  </Leg>
  <Leg>
    <Dep>
      <Time>17:23</Time>
      <Stop>Mölndal station</Stop>
      <City>Mölndal</City>
    </Dep>
    <Arr>
      <Time>17:41</Time>
      <Stop>Kungsbacka station</Stop>
      <City>Kungsbacka</City>
    </Arr>
    <Line>
      <Number>3069</Number>
      <Vehicle>train</Vehicle>
    </Line>
  </Leg>
    <Leg>
    <Dep>
      <Time>17:47</Time>
      <Stop>Kungsbacka station</Stop>
      <City>Kungsbacka</City>
    </Dep>
    <Arr>
      <Time>18:14</Time>
      <Stop>Idala</Stop>
      <City>Kungsbacka</City>
    </Arr>
    <Line>
      <Number>744</Number>
      <Vehicle>bus</Vehicle>
    </Line>
  </Leg>
</Route>
```

6, alternative, using attributes

```
<?xml version="1.0" encoding="utf-8"
standalone="no"?>
<!DOCTYPE Route [
<!ELEMENT Route (Leg+)>
<!ELEMENT Leg (Dep, Arr, Line)>
<!ELEMENT Dep EMPTY>
<!ELEMENT Arr EMPTY>
<!ELEMENT Line EMPTY>
<!ATTLIST Dep
time CDATA #REQUIRED
stop CDATA #REQUIRED
city CDATA #REQUIRED
>
<!ATTLIST Arr
time CDATA #REQUIRED
stop CDATA #REQUIRED
city CDATA #REQUIRED
>
<!ATTLIST Line
number CDATA #REQUIRED
vehicle CDATA #REQUIRED
>
]>
```

```
<Route>
  <Leg>
    <Dep time="17:02" stop="Almedal" city="Göteborg" />
    <Arr time="17:12" stop="Mölndal station" city="Mölndal" />
    <Line number="2" vehicle="tram" />
  </Leg>
  <Leg>
    <Dep time="17:23" stop="Mölndal station" city="Mölndal" />
    <Arr time="17:41" stop="Kungsbacka station" city="Kungsbacka" />
    <Line number="3069" vehicle="train" />
  </Leg>
  <Leg>
    <Dep time="17:47" stop="Kungsbacka station" city="Kungsbacka" />
    <Arr time="18:14" stop="Idala" city="Kungsbacka" />
    <Line number="744" vehicle="bus" />
  </Leg>
</Route>
```