Data-cleaning benchmark

https://github.com/nonan/Data-Cleaning-Benchmark

Denial constraints

Denial constraints provide a more expressive way to define domain rules over the data. Here, we express the dependencies and the domain rules as denial constraints and try to find the violations with respect to these constraints.

C1: $\neg$ *(MP(m, n, g, o), VoteIndiv(*m'*,* n'*, c, v, vv),* $(m = m')$*,* $(n \neq n')$*)*

Semantics. The denial constraint states that a parliamentarian in VoteIndiv table cannot be listed with a name different from the name listed in MP table. The denial constraint is expressed as the following SQL query:

SQL Query1:

```
SELECT DISTINCT * FROM MP, VoteIndiv
WHERE MP.mp_id = VoteIndiv.mp_id AND MP.name ! = VoteIndiv.name;
```

C2: $\neg$ *(MParl(m, p, a, c, e), VoteIndiv(*m'*, n,* c'*, v, vv),* $(m = m')$*,* $(c \neq c')$

Semantics. The denial constraint states that a parliamentarian in VoteIndiv table cannot be listed with a constituency different from the constituecy listed in MParl table.

The denial constraint is expressed as the following SQL query:

SQL Query2:

```
SELECT DISTINCT * FROM MParl, VoteIndiv
WHERE MParl.mp_id = VoteIndiv.mp_id AND MParl.constituency ! = VoteIn-
div.constituency;
```

C3: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), VoteIndiv(m, n, c,* v'*, vv),* $(v = v')$*,* $(y =$ *"True"*$)$*,* $(vv \neq$ *"Yea"))*

Semantics. The denial constraint states that a yea vote in VoteIndiv table cannot be listed with a different value in Vote table.

The denial constraint is expressed as the following SQL query:

SQL Query3:

```
SELECT * FROM VoteIndiv, Vote
WHERE Vote.vote_id = VoteIndiv.vote_id AND VoteIndiv.vote_v = 'Yea' AND Vote.yea
! = 'True';
OR
SELECT * FROM VoteIndiv, Vote
WHERE Vote.vote_id = VoteIndiv.vote_id AND VoteIndiv.vote_v = 'Yea' AND Vote.nay
= 'True';
```

C4: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), VoteIndiv(m, n, c, v′, vv), (v = v′), (n = "True"), (vv* $\neq$ *"Nay"))*

Semantics. The denial constraint states that a nay vote in VoteIndiv table cannot be listed with a different value in Vote table.

The denial constraint is expressed as the following SQL query:

SQL Query4:

```
SELECT * FROM VoteIndiv, Vote
WHERE Vote.vote_id = VoteIndiv.vote_id AND VoteIndiv.vote_v = 'Nay' AND Vote.nay
! = 'True';
OR
SELECT * FROM VoteIndiv, Vote
WHERE Vote.vote_id = VoteIndiv.vote_id AND VoteIndiv.vote_v = 'Nay' AND Vote.yea
= 'True';
```

C5: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), (ys > ns), (d* $\neq$*"Agreed to"))*

Semantics. The denial constraint expresses that a vote with decision division number of yeas more than decision division number of nays in Vote table cannot be marked with a decision value different from "Agreed To".

The denial constraint is expressed as the following SQL query:

SQL Query5:

```
SELECT Vote.vote_id FROM Vote
WHERE Vote.yeas > Vote.nays AND Vote.dec_result ! = 'Agreed To';
```

C6: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), (ns > ys), (d* $\neq$ *"Negatived"))*

Semantics. The denial constraint expresses that a vote with decision division number of nays more than decision division number of yeas in Vote table cannot be marked with a decision value different from "Negatived".

The denial constraint is expressed as the following SQL query:

SQL Query6:

```
SELECT Vote.vote_id FROM Vote
WHERE Vote.nays > Vote.yeas AND Vote.dec_result ! = 'Negatived';
```

C7: $\neg$*(Parl(p, bd, ed), (bd>ed))*

Semantics. The denial constraint expresses that the start date of a parliament in Parl table cannot be greater than its end date.

The denial constraint is expressed as the following SQL query:

SQL Query7:

```
SELECT Parl.p_id FROM Parl WHERE date(Parl.bdate) > date(Parl.edate);
```

C8: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), Yeas(b′, d′, ys′), (b = b′), (d = d′), (ys ≠ ys′)*
Semantics. The denial constraint expresses that total number of yea votes of a specific bill should match yeas attribute in Vote table.
The denial constraint is expressed as the following SQL queries:

SQL Query8:

```
CREATE VIEW yeav(p_id, session_id, dec_div, bill_no, bill_name, doc_name, no) as select
p_id, session_id, dec_div, bill_no, bill_name, doc_name, count(*) FROM Vote
WHERE Vote.yea = "True" AND Vote.bill_name ! = "Null" GROUP BY Vote.p_id,
Vote.session_id, Vote.dec_div, Vote.bill_no, Vote.bill_name, Vote.doc_name;
SELECT * FROM yeav,Vote
WHERE yeav.p_id = Vote.p_id AND yeav.session_id = Vote.session_id AND yeav.dec_div
= Vote.dec_div AND yeav.bill_no = Vote.bill_no AND yeav.bill_name = Vote.bill_name
AND yeav.doc_name = Vote.doc_name AND yeav.no ! = Vote.yeas;
```

C9: $\neg$*(Vote(v, p, s, d, b, bn, dn, y, n, ys, ns, dr), Nays(p′,s′,d′,b′,bn′,dn′, ns′), (p = p′), (s = s′), (d = d′), (b = b′), (bn = bn′), (dn = dn′), (d = d′), (ns ≠ ns′)*
Semantics. The denial constraint expresses that total number of nay votes of a specific bill should match nays attribute in Vote table.
The denial constraint is expressed as the following SQL queries:

SQL Query9:

```
CREATE VIEW nays(p_id, session_id, dec_div, bill_no, bill_name, doc_name, no) as select
p_id, session_id, dec_div, bill_no, bill_name, doc_name, count(*) FROM Vote
WHERE Vote.nay = "True" AND Vote.bill_name != "Null" GROUP BY Vote.p_id,
Vote.session_id, Vote.dec_div, Vote.bill_no, Vote.bill_name, Vote.doc_name;
SELECT * FROM nays,Vote
WHERE nays.p_id = Vote.p_id AND nays.session_id = Vote.session_id AND nays.dec_div
= Vote.dec_div AND nays.bill_no = Vote.bill_no AND nays.bill_name = Vote.bill_name
AND nays.doc_name = Vote.doc_name AND nays.no != Vote.nays;
```

C10: $\neg$*(MP(m, n, g, o), MParl(m′, p, a, c, e), Bill(bid, b, bt, sp, n′, a′, p′, s, ind, bt), (m = m′), (n = n′), (p = p′), (a ≠ a′))*
Semantics. The denial constraint expresses that the affiliation of the sponsor in Bill table should match the affiliation attribute in MParl table.
The denial constraint is expressed as the following SQL queries:

SQL Query10:

```
SELECT * FROM MP, MParl, Bill WHERE MP.mp_id=MParl.mp_id
AND MP.name = Bill.sponsor_name AND MParl.p_id = Bill.p_id AND
MParl.paffiliation! =Bill.sponsor_affiliation;
```

C11:¬*(MParl(m,p,a,c,e), Parl(p′,bd,ed), (p=p′), (e>ed))*

Semantics. The denial constraint expresses that the election date of a parliamentarian in a given parliament cannot be later than the end date of parliament.

The denial constraint is expressed as the following SQL queries:

SQL Query11:

```
SELECT * FROM MParl,Parl WHERE MParl.electiond > Parl.edate and MParl.p_id =
Parl.p_id;
```

C12:¬*(Bill(bid, b, bt, sp, n, a, p, s, ind, bt), Parl(p′,bd,ed), (p=p′), (ind>ed))*

Semantics. The denial constraint expresses that the date that a bill introduced in a parliament cannot be later than the end date of parliament.

SQL Query12:

```
SELECT * FROM Bill, Parl WHERE Bill.p_id = Parl.p_id AND Bill.intro_d > Parl.edate;
```

C13:¬*(Bill(bid, b, bt, sp, n, a, p, s, ind, bt), Parl(p′,bd,ed), (p=p′), (ind<bd))*

Semantics. The denial constraint expresses that the date that a bill introduced in a parliament cannot be before than the beginning date of parliament.

SQL Query13:

```
SELECT * FROM Bill, Parl WHERE Bill.p_id = Parl.p_id AND Bill.intro_d < Parl.bdate;
```

C14:¬*(MParl(m,p,a,c,e), (a="New Party"), (e>"1961"))*

Semantics. The denial constraint expresses that the election date of a parliamentarian cannot be later than 1961.

SQL Query14:

```
SELECT * FROM MParl WHERE paffiliation="New Party" AND electiond >
DATE('1961-01-01');
```