Find the drinkers who frequent **only** bars serving a beer they like

$\forall x \in R\ \varphi(x) \equiv \neg\exists x \in R\ \neg\varphi(x)$

$$\{d: drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \wedge \\ \forall y \in frequents[y(drinker) = f(drinker) \rightarrow \\ \exists s \in serves \exists l \in likes(s(bar) = y(bar) \wedge \\ s(beer) = l(beer) \wedge l(drinker) = y(drinker))]\}$$

$$\{d: drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \wedge \\ \neg\exists y \in frequents[y(drinker) = f(drinker) \wedge \\ \neg\exists s \in serves \exists l \in likes(s(bar) = y(bar) \wedge \\ s(beer) = l(beer) \wedge l(drinker) = y(drinker))]\}$$

```
select f.drinker
from frequents f
where not exists
        (select *
        from frequents y
        where y.drinker = f.drinker and not exists
                (select *
                from serves s, likes l
                where s.bar= y.bar
                and s.beer= l.beer
                and l.drinker = y.drinker))
```

```
select drinker
from frequents  where drinker not in
        (select f.drinker
        from frequents f
        where f.bar not in
                (select bar
                from serves, likes
                where serves.beer = likes.beer
                and likes.drinker = f.drinker))
```

List the days when no red boat is reserved

$$\{d: day \mid weekday(d) \wedge \forall r \in reservation\ \forall b \in boat \\ ((r(day) = d(day) \wedge r(bname) = b(bname)) \rightarrow b(color) \neq red)\}$$

$$\{d: day \mid weekday(d) \wedge \neg\exists r \in reservation\ \exists b \in boat \\ (r(day) = d(day) \wedge r(bname) = b(bname) \wedge b(color) = red)\}$$

```
select d.day from weekday d
where not exists
        (select * from reservation r, boat b
        where r.day = d.day and r.name = b.bname AND b.color= 'red')
```

Find actors playing in every movie by Berto

$$\{a\text{: actor} \mid \exists y \in \text{movie}\ [a(\text{actor}) = y(\text{actor}) \wedge \\ \forall m \in \text{movie}\ [m(\text{director}) = \text{“Berto”} \rightarrow \exists t \in \text{movie}\ (m(\text{title}) = \\ t(\text{title}) \wedge t(\text{actor}) = y(\text{actor}))]]\}$$

$$\{a\text{: actor} \mid \exists y \in \text{movie}\ [a(\text{actor}) = y(\text{actor}) \wedge \\ \neg\exists m \in \text{movie}\ [m(\text{dir}) = \text{“Berto”} \wedge \neg\exists t \in \text{movie}\ (m(\text{title}) = t(\text{title}) \\ \wedge t(\text{actor}) = y(\text{actor}))]]\}$$

```
CREATE VIEW berto-movie AS
SELECT title FROM movie WHERE director = 'bertolucci'

CREATE VIEW not-all-berto AS
SELECT m.actor FROM movies m, berto-movies
WHERE berto-movies.title NOT IN
      (SELECT title FROM movies
       WHERE actor = m.actor)

SELECT actor FROM movies WHERE actor NOT IN
      (SELECT * FROM not-all-berto)
```

List the directors that every actor is cast in one of his movies

$$\{d: director \mid \ \exists m \in movie\ [d(director) = m(director)\ \wedge \\ \forall a \in movie\ \exists x \in movie\ \exists y \in movie(x(director) = m(director) \wedge x(title) = y(title) \\ \wedge\ y(actor) = a(actor))]\}$$

$$\{d: director \mid \ \exists m \in movie\ [d(director) = m(director)\ \wedge \\ \neg\exists a \in movie\ \neg\exists x \in movie\ \exists y \in movie(x(director) = m(director) \wedge x(title) = y(title) \\ \wedge\ y(actor) = a(actor))]\}$$

Propositional logic

- $\neg(\neg p) = p$
- $p \rightarrow q = \neg p \vee q$ (definition of implication)
- $\neg(p \wedge q) = \neg p \vee \neg q$ (De Morgan's Law 1)
- $\neg(p \vee q) = \neg p \wedge \neg q$ (De Morgan's Law 2)
- $\neg(p \rightarrow q) = \neg(\neg p \vee q) = p \wedge \neg q$

Relational calculus

Below R is a relation and Q, Q_1, Q_2 are calculus formulas.

- $\forall x \in R\ (Q(x)) = \neg\ (\exists x \in R\ (\neg Q(x)))$
- $\exists x \in R\ (Q(x)) = \neg\ (\forall x \in R\ (\neg Q(x)))$
- $\exists x \in R\ (Q_1(x) \vee Q_2(x)) = \exists x \in R\ (Q_1(x)) \vee \exists x \in R\ (Q_2(x))$ ($\exists$ distributes over $\vee$)
- $\forall x \in R\ (Q_1(x) \wedge Q_2(x)) = \forall x \in R\ (Q_1(x)) \wedge \forall x \in R\ (Q_2(x))$ ($\forall$ distributes over $\wedge$)

Note that $\exists$ does not distribute over $\wedge$ and $\forall$ does not distribute over $\vee$ (think of some counter-examples!). However, suppose x occurs in Q_1 but not in Q_2. Then:

- $\exists x \in R\ (Q_1(x) \wedge Q_2) = \exists x \in R\ (Q_1(x)) \wedge Q_2$
- $\forall x \in R\ (Q_1(x) \vee Q_2) = \forall x \in R\ (Q_1(x)) \vee Q_2$

Also note the following useful facts that follow from the previous ones:

- $\forall x \in R\ (Q_1(x) \rightarrow Q_2(x)) = \neg(\exists x \in R\ (Q_1(x) \wedge \neg Q_2(x)))$
- $\forall x \in R\ \forall y \in R\ (Q(x, y)) = \neg(\exists x \in R\ \exists y \in R\ (\neg Q(x, y)))$

If Q_2 does not contain x then:

- $[\exists x \in R\ (Q_1(x)) \rightarrow Q_2] = \forall x \in R\ [Q_1(x) \rightarrow Q_2]$

Basic

- Testing if an attribute A is null: **IS null, IS NOT null**
- Arithmetic operations: involves any null return null
- Comparison: involves any null return ***unknown***

Truth tables involving ***unknown***

- AND: false then must false, else ***unknown***
- OR: true then must true, else ***unknown***

- Aggregate functions
  - Usually ignores tuples with null
  - Returns null if there is no non-null amount
  - All except the COUNT(*) ignore tuples with null values on the aggregate attributes

Foreign key references the **primary key(must have)** of the target or **null**.

```
CREATE TABLE branch(bra_name char(15) not null, bra_city char(30),
    PRIMARY KEY(dnumber),UNIQUE(dname),
    FOREIGN KEY (mgrssn) REFERENCES emp);
DROP TABLE branch; ALTER TABLE r ADD attribute dom
```

<attribute> **LIKE** <pattern>/**(NOT)UNIQUE** <query>/ **ORDER BY** (default asc)

NOT monotonic (CANNOT BE FLATTENED)

- A **op ANY/ALL** <query>, If any/all X of the result of query satisfies A op X.
- **Close world** assumption: If a tuple is missing in database, then it's not true.

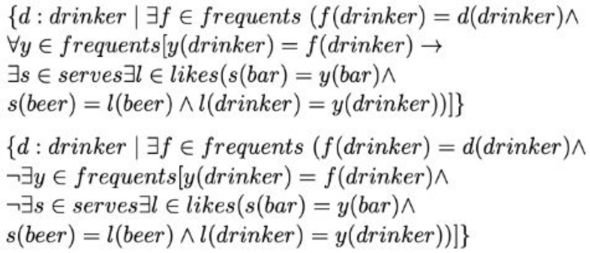
```
INSERT INTO r(attribute, att) VALUES (v1,v2,v3,..)
DELETE FROM r WHERE, UPDATE r SET WHERE
```

update views without aggregates, nesting, group-by or tuple alias, defined on a **single table**

N/A	pro	con
virtual	no need to maintain correspondence with base	inefficient for views defined via **complex** queries
materialized	Expect many queries, fast	Cost of space, refresh every time DB update

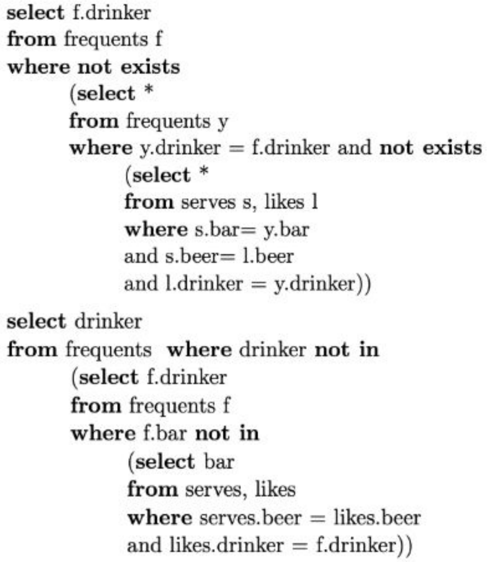
Find the drinkers who frequent **only** bars serving a beer they like

$\forall x \in R\ \varphi(x) \equiv \neg\exists x \in R\ \neg\varphi(x)$



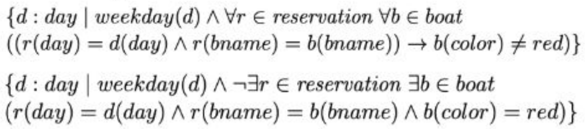
$\{d : drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \land$
$\forall y \in frequents[y(drinker) = f(drinker) \rightarrow$
$\exists s \in serves \exists l \in likes(s(bar) = y(bar) \land$
$s(beer) = l(beer) \land l(drinker) = y(drinker))]\}$

$\{d : drinker \mid \exists f \in frequents\ (f(drinker) = d(drinker) \land$
$\neg\exists y \in frequents[y(drinker) = f(drinker) \land$
$\neg\exists s \in serves \exists l \in likes(s(bar) = y(bar) \land$
$s(beer) = l(beer) \land l(drinker) = y(drinker))]\}$



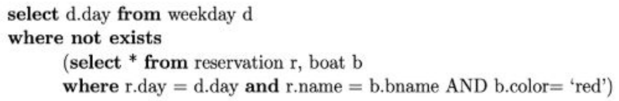
```
select f.drinker
from frequents f
where not exists
        (select *
        from frequents y
        where y.drinker = f.drinker and not exists
                (select *
                from serves s, likes l
                where s.bar= y.bar
                and s.beer= l.beer
                and l.drinker = y.drinker))
select drinker
from frequents  where drinker not in
        (select f.drinker
        from frequents f
        where f.bar not in
                (select bar
                from serves, likes
                where serves.beer = likes.beer
                and likes.drinker = f.drinker))
```

List the days when no red boat is reserved



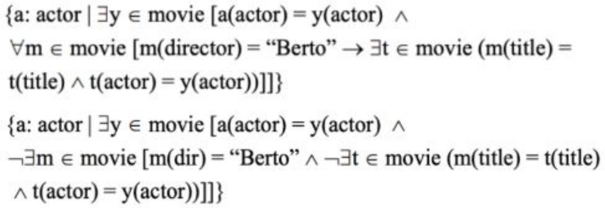
$\{d : day \mid weekday(d) \land \forall r \in reservation\ \forall b \in boat$
$((r(day) = d(day) \land r(bname) = b(bname)) \rightarrow b(color) \neq red)\}$

$\{d : day \mid weekday(d) \land \neg\exists r \in reservation\ \exists b \in boat$
$(r(day) = d(day) \land r(bname) = b(bname) \land b(color) = red)\}$



```
select d.day from weekday d
where not exists
       (select * from reservation r, boat b
       where r.day = d.day and r.name = b.bname AND b.color= 'red')
```

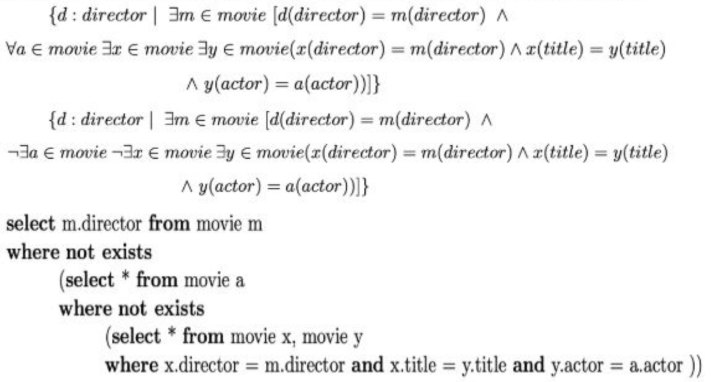
Find actors playing in every movie by Berto



$\{a: actor \mid \exists y \in movie\ [a(actor) = y(actor)\ \land$
$\forall m \in movie\ [m(director) = \text{“Berto”} \rightarrow \exists t \in movie\ (m(title) =$
$t(title) \land t(actor) = y(actor))]]\}$

$\{a: actor \mid \exists y \in movie\ [a(actor) = y(actor)\ \land$
$\neg\exists m \in movie\ [m(dir) = \text{“Berto”} \land \neg\exists t \in movie\ (m(title) = t(title)$
$\land\ t(actor) = y(actor))]]\}$

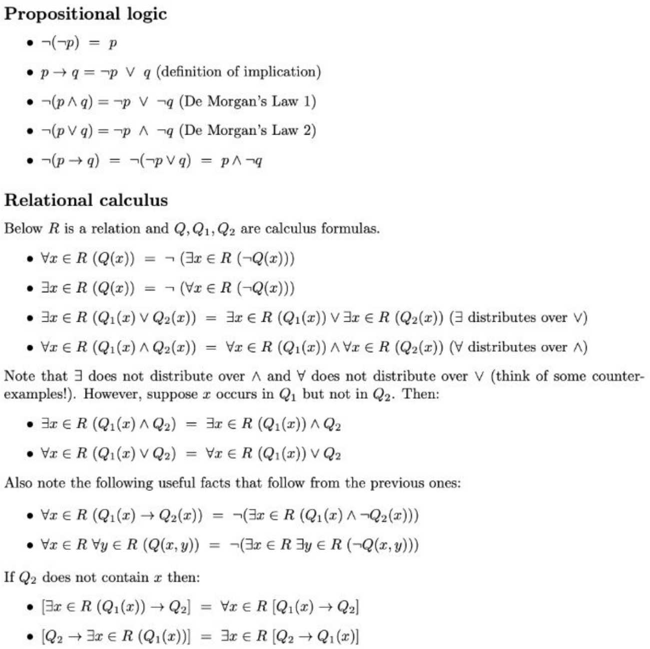
List the directors that every actor is cast in one of his movies



$\{d : director \mid\ \exists m \in movie\ [d(director) = m(director)\ \land$
$\forall a \in movie\ \exists x \in movie\ \exists y \in movie(x(director) = m(director) \land x(title) = y(title)$
$\land\ y(actor) = a(actor))]\}$

$\{d : director \mid\ \exists m \in movie\ [d(director) = m(director)\ \land$
$\neg\exists a \in movie\ \neg\exists x \in movie\ \exists y \in movie(x(director) = m(director) \land x(title) = y(title)$
$\land\ y(actor) = a(actor))]\}$

```
select m.director from movie m
where not exists
      (select * from movie a
      where not exists
            (select * from movie x, movie y
            where x.director = m.director and x.title = y.title and y.actor = a.actor ))
```



Propositional logic

- $\neg(\neg p) = p$
- $p \rightarrow q = \neg p \lor q$ (definition of implication)
- $\neg(p \land q) = \neg p \lor \neg q$ (De Morgan's Law 1)
- $\neg(p \lor q) = \neg p \land \neg q$ (De Morgan's Law 2)
- $\neg(p \rightarrow q) = \neg(\neg p \lor q) = p \land \neg q$

Relational calculus

Below R is a relation and Q, Q_1, Q_2 are calculus formulas.

- $\forall x \in R\ (Q(x)) = \neg\ (\exists x \in R\ (\neg Q(x)))$
- $\exists x \in R\ (Q(x)) = \neg\ (\forall x \in R\ (\neg Q(x)))$
- $\exists x \in R\ (Q_1(x) \lor Q_2(x)) = \exists x \in R\ (Q_1(x)) \lor \exists x \in R\ (Q_2(x))$ ($\exists$ distributes over $\lor$)
- $\forall x \in R\ (Q_1(x) \land Q_2(x)) = \forall x \in R\ (Q_1(x)) \land \forall x \in R\ (Q_2(x))$ ($\forall$ distributes over $\land$)

Note that $\exists$ does not distribute over $\land$ and $\forall$ does not distribute over $\lor$ (think of some counter-examples!). However, suppose x occurs in Q_1 but not in Q_2. Then:

- $\exists x \in R\ (Q_1(x) \land Q_2) = \exists x \in R\ (Q_1(x)) \land Q_2$
- $\forall x \in R\ (Q_1(x) \lor Q_2) = \forall x \in R\ (Q_1(x)) \lor Q_2$

Also note the following useful facts that follow from the previous ones:

- $\forall x \in R\ (Q_1(x) \rightarrow Q_2(x)) = \neg(\exists x \in R\ (Q_1(x) \land \neg Q_2(x)))$
- $\forall x \in R\ \forall y \in R\ (Q(x, y)) = \neg(\exists x \in R\ \exists y \in R\ (\neg Q(x, y)))$

If Q_2 does not contain x then:

- $[\exists x \in R\ (Q_1(x)) \rightarrow Q_2] = \forall x \in R\ [Q_1(x) \rightarrow Q_2]$
- $[Q_2 \rightarrow \exists x \in R\ (Q_1(x))] = \exists x \in R\ [Q_2 \rightarrow Q_1(x)]$

Basic

- Testing if an attribute A is null: **IS null, IS NOT null**
- Arithmetic operations: involves any null return null
- Comparison: involves any null return ***unknown***

Truth tables involving ***unknown***

- AND: false then must false, else ***unknown***
- OR: true then must true, else ***unknown***

- Aggregate functions
  - Usually ignores tuples with null
  - Returns null if there is no non-null amount
  - All except the COUNT(*) ignore tuples with null values on the aggregate attributes

Foreign key references the **primary key(must have)** of the target or **null**.

```
CREATE TABLE branch(bra_name char(15) not null, bra_city char(30),
    PRIMARY KEY(dnumber),UNIQUE(dname),
    FOREIGN KEY (mgrssn) REFERENCES emp);
DROP TABLE branch; ALTER TABLE r ADD attribute dom
```

<attribute> **LIKE** <pattern>/**(NOT)UNIQUE** <query>/ **ORDER BY** (default asc)

NOT monotonic (CANNOT BE FLATTENED)

- A **op ANY/ALL** <query>, If any/all X of the result of query satisfies A op X.
- **Close world** assumption: If a tuple is missing in database, then it's not true.

```
INSERT INTO r(attribute, att) VALUES (v1,v2,v3,..)
DELETE FROM r WHERE, UPDATE r SET WHERE
```

update views without aggregates, nesting, group-by or tuple alias, defined on a **single table**

N/A	pro	con
virtual	no need to maintain correspondence with base	inefficient for views defined via **complex** queries
materialized	Expect many queries, fast	Cost of space, refresh every time DB update