CSCI 620

ASSIGNMENT 5

Abhishek Shah, as5553

Question 1

```
-- Creating a temp table Movies for given conditions
create table Tempmovies(
        id integer PRIMARY KEY,
        type varchar(15),
        title text,
        originalTitle text,
        startYear integer,
        endYear integer,
        runtime integer,
        avgRating float,
        numVotes integer
);


-- Inserting into temp table Movies
insert into Tempmovies
select * from Title
where runtime > 75 and type='movie';
```

```
-- Source 1
-- Non-Materialized view
create view ComedyMovie as
select tm.id, tm.title, tm.startYear as year
from Tempmovies as tm
join
Title_Genre as tg
on tm.id = tg.title
join
Genre as g
on g.id = tg.genre
where g.genre = 'Comedy';

-- Materialized view
create materialized view ComedyMovieMat as
select tm.id, tm.title, tm.startYear as year
from Tempmovies as tm
join
Title_Genre as tg
on tm.id = tg.title
join Genre as g
on g.id = tg.genre
where g.genre = 'Comedy';

-- Source 2
-- Non-Materialized view
create view NonComedyMovie as
select tm.id, tm.title, tm.startYear as year
from Tempmovies as tm
```

```
join
Title_Genre as tg
on tm.id = tg.title
join
Genre as g
on g.id = tg.genre
where g.genre <> 'Comedy' and tm.id not in (select id from ComedyMovie);

-- Materialized view
create materialized view NonComedyMovieMat as
select tm.id, tm.title, tm.startYear as year
from Tempmovies as tm
join
Title_Genre as tg
on tm.id = tg.title
join
Genre as g
on g.id = tg.genre
where g.genre <> 'Comedy' and tm.id not in (select id from ComedyMovie);


-- Source 3
-- Non-Materialized view
create view ComedyActor as
select m.id, m.name, m.birthYear, m.deathYear
from Member as m
where m.id in
(select actor from Title_Actor as ta
 where exists(select 1 from ComedyMovie as cm where cm.id = ta.title));
```

```
-- Materialized view
create materialized view ComedyActorMat as
select m.id, m.name, m.birthYear, m.deathYear
from Member as m
where m.id in
(select actor from Title_Actor as ta where exists(select 1 from ComedyMovie as cm where cm.id = ta.title));


-- Source 4
-- Non-Materialized view
create view NonComedyActor as
select m.id, m.name, m.birthYear, m.deathYear from Member as m where m.id in
(select actor from Title_Actor as ta
where title in (select id from NonComedyMovie as ncm where ncm.id = ta.title));

-- Materialized view
create materialized view NonComedyActorMat as
select m.id, m.name, m.birthYear, m.deathYear
from Member as m
where m.id in
(select actor from Title_Actor as ta
where title in (select id from NonComedyMovie as ncm where ncm.id = ta.title));


-- Source 5
-- Non-Materialized view
create view ActedIn as
select ta.actor, title as movie
from Title_Actor as ta
```

where ta.title in (select id from Movies as m where ta.title = m.id);

-- Materialized view

create materialized view ActedInMat as

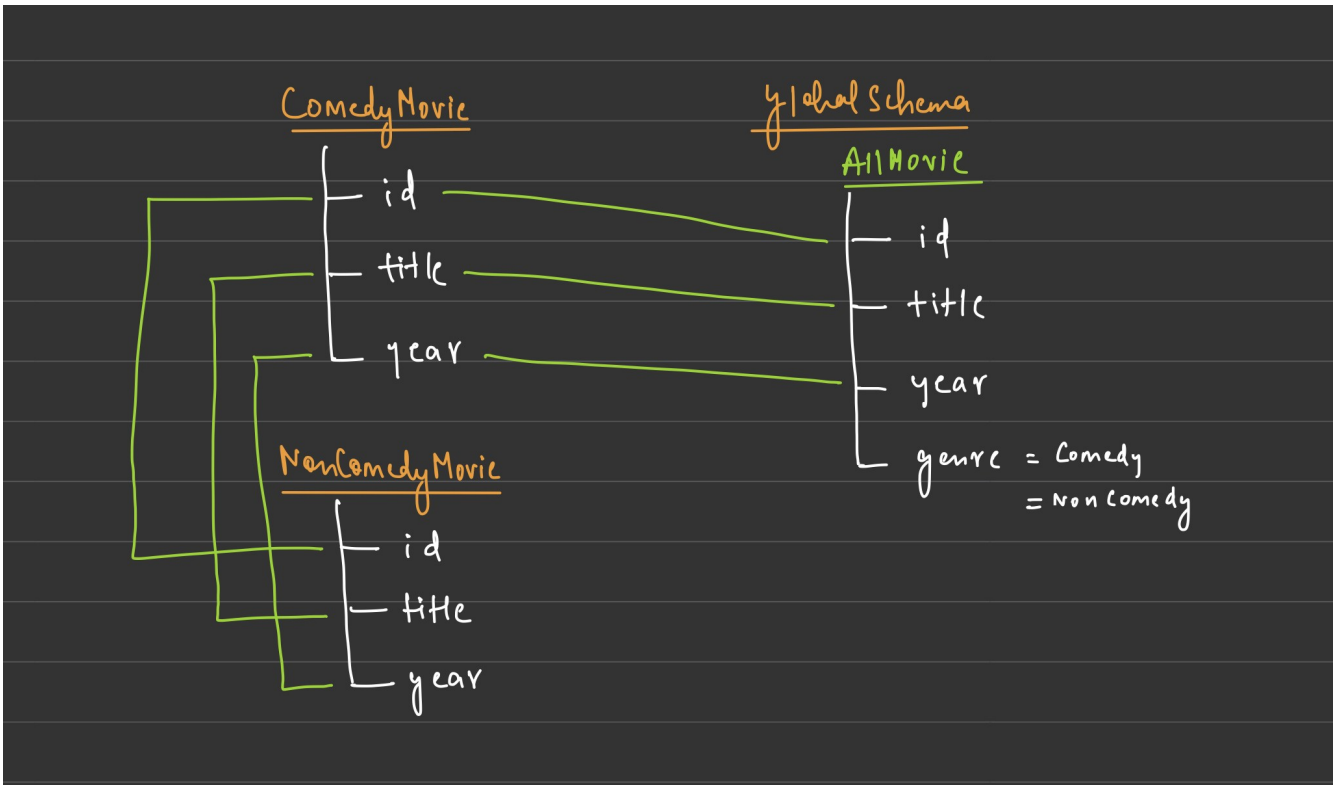
select ta.actor, title as movie

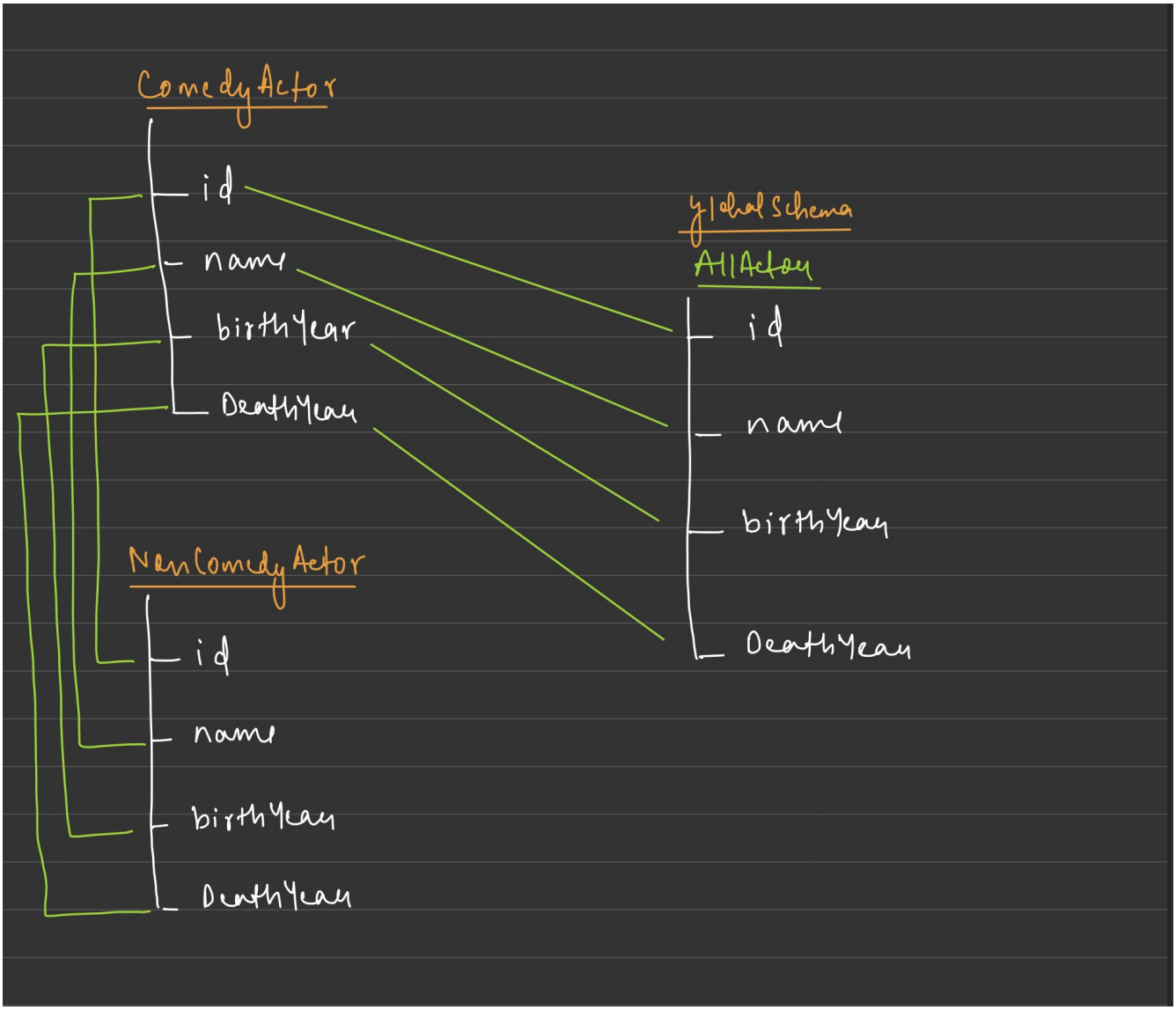
from Title_Actor as ta

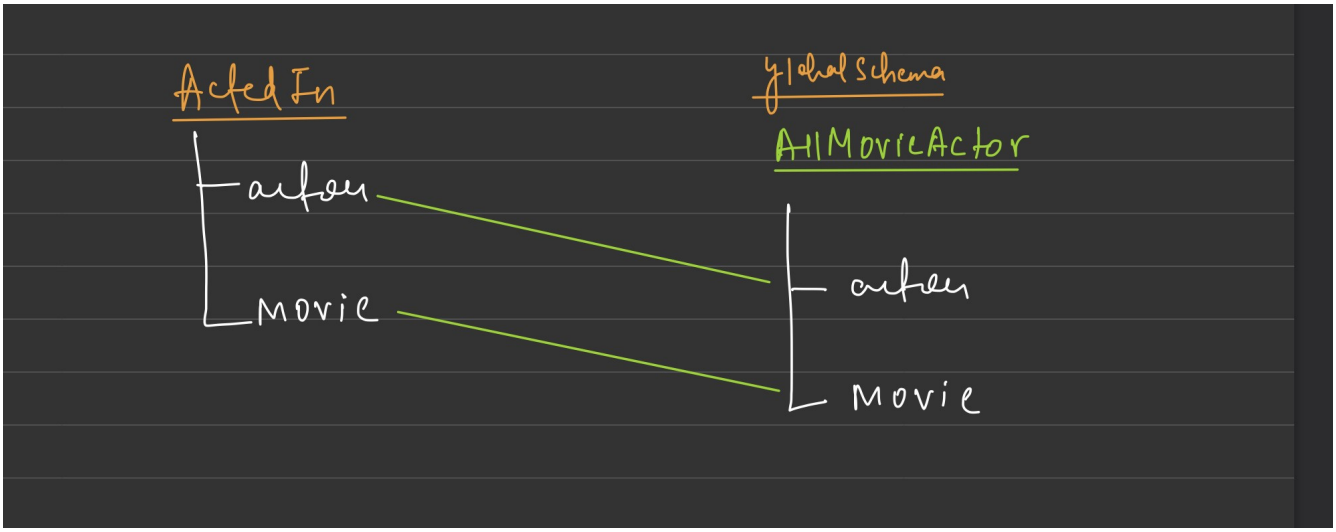
where ta.title in (select id from Movies as m where ta.title = m.id);

Question 2

GAV mappings using non-materialized views




ComedyActor
id
name
birthYear
DeathYear
NonComedyActor
id
name
birthYear
DeathYear
global Schema
AllActor
id
name
birthYear
DeathYear



-- Non Materialized Global Schema

create view All_Movie as

select id,title,year,'Comedy' as genre from ComedyMovie

Union

select id,title,year,'Non-Comedy' as genre from NonComedyMovie;

create view All_Actor as

select id,name,birthyear,deathyear from ComedyActor

Union

select id,name,birthyear,deathyear from NonComedyActor;

create view All_Movie_Actor as

select actor,movie from ActedIn;

-- Materialized Global Schema

create materialized view All_MovieMat as

select id,title,year,'Comedy' as genre from ComedyMovieMat

Union

```
select id,title,year,'Non-Comedy' as genre from NonComedyMovieMat;

create materialized view All_ActorMat as
select id,name,birthyear,deathyear from ComedyActorMat
Union
select id,name,birthyear,deathyear from NonComedyActorMat;

create materialized view All_Movie_ActorMat as
select actor,movie from ActedInMat;
```

Question 3

```
--3.1
select aa.name
from All_Movie as am
join
All_Movie_Actor as alm
on alm.movie = am.id
join
All_Actor as aa
on aa.id = alm.actor
where am.year between 2000 and 2005
group by actor, aa.name
having count(movie) > 10;
-- Total query runtime: 21 secs 362 msecs.

--3.2
select aa.name
from All_Movie as am
```

```
join
All_Movie_Actor as alm
on
alm.movie=am.id
join
All_Actor as aa
on aa.id=alm.actor
where aa.name like 'Ja%' and am.genre='Non-Comedy';
-- Total query runtime: 6 secs 298 msecs.
```

Question 4

```
-- 4.1
-- Non Materialized Views
select aa.name
from
(select id,title,year,'Comedy' as genre from ComedyMovie
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovie) as am
join
(select actor,movie from ActedIn) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActor
Union
select id,name,birthyear,deathyear from NonComedyActor) as aa
on aa.id=alm.actor
where am.year between 2000 and 2005
group by actor,aa.name
having count(movie)>10;
```

```
--  Total query runtime: 19 secs 907 msec.

-- Materialized
select aa.name
from
(select id,title,year,'Comedy' as genre from ComedyMovieMat
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovieMat) as am
join
(select actor,movie from ActedInMat) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActorMat
Union
select id,name,birthyear,deathyear from NonComedyActorMat) as aa
on aa.id=alm.actor
where am.year between 2000 and 2005
group by actor,aa.name
having count(movie) > 10;
-- Total query runtime: 220 msec.


-- 4.2
-- Non-Materialized view
select aa.name
from
(select id,title,year,'Comedy' as genre from ComedyMovie
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovie) as am
join
```

```
(select actor,movie from ActedIn) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActor
Union
select id,name,birthyear,deathyear from NonComedyActor) as aa
on aa.id=alm.actor
where aa.name like 'Ja%'
and am.genre='Non-Comedy';
-- Total query runtime: 5 secs 47 msec.


-- Materialized View
select aa.name
from
(select id,title,year,'Comedy' as genre from ComedyMovieMat
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovieMat) as am
join
(select actor,movie from ActedInMat) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActorMat
Union
select id,name,birthyear,deathyear from NonComedyActorMat) as aa
on aa.id=alm.actor
where aa.name like 'Ja%'
and am.genre='Non-Comedy';
--  Total query runtime: 286 msec.
```

Question 5

```
-- Optimizing 4.1
-- Non Materialized Views
select aa.name
from
(select id,title,year,'Comedy' as genre from ComedyMovie
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovie) as am
join
(select actor,movie from ActedIn) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActor
Union
select id,name,birthyear,deathyear from NonComedyActor) as aa
on aa.id = alm.actor
where am.year between 2000 and 2005
group by aa.name
having count(movie)>10;
--  Total query runtime: 19 secs 907 msec.
```

Before optimizing: 19 secs 907 msecs

After Optimizing: 17 secs 771 msecs

```
-- Materialized
select aa.name
from
```

```
(select id,title,year,'Comedy' as genre from ComedyMovieMat
Union
select id,title,year,'Non-Comedy' as genre from NonComedyMovieMat) as am
Join
(select actor,movie from ActedInMat) as alm
on alm.movie=am.id
join
(select id,name,birthyear,deathyear from ComedyActorMat
Union
select id,name,birthyear,deathyear from NonComedyActorMat) as aa
on aa.id=alm.actor
where am.year between 2000 and 2005
group by aa.name
having count(movie) > 10;
-- Total query runtime: 241 msec.
```

Before optimizing: 241 msec

After Optimizing: 204 msec

I need to optimized the queries by removing redundant join or removing sources not used. However, for this query, I was unable to figure out which join I should remove in order to boost the performance. I find all the joins extremely necessary and even if I remove any one join or remove any one source, I know I'll get an error.

I got a 1-2 seconds by removing unnecessary overhead from group clause.

```
-- Optimizing 4.2
-- 4.2 Non Materialized
select distinct aa.name
from
```

```
(select id,title,year from NonComedyMovie) as am
inner join
(select actor,movie from Actedin) as alm
on alm.movie=am.id
inner join
(select id,name,birthyear,deathyear from NonComedyActor) as aa
on aa.id=alm.actor
where aa.name like 'Ja%';
-- Total query runtime: 4 secs 49 msec.
```

Before optimizing: 5 secs 47 msecs

After Optimizing: 4 secs 49 msecs

```
-- 4.2 Materialized
select distinct aa.name
from
(select id,title,year from NonComedyMovieMat) as am
inner join
(select actor,movie from ActedinMat) as alm
on alm.movie=am.id
inner join
(select id,name,birthyear,deathyear from NonComedyActorMat) as aa
on aa.id=alm.actor
where aa.name like 'Ja%';
-- Total query runtime: 88 msec.
```

Before optimizing: 286 msecs

After Optimizing: 88 msecs

We optimized the queries by removing a redundant join and removing the union by deselecting the ComedyMovie source and ComedyMovieMat source.