From the aggregate-oriented logical schema to Cassadra logical/physical schema

Input

- The aggregate-oriented logical schema in meta-notation
- The annotated ER diagram
- One NoSQL system S (for today, Cassandra)

Output

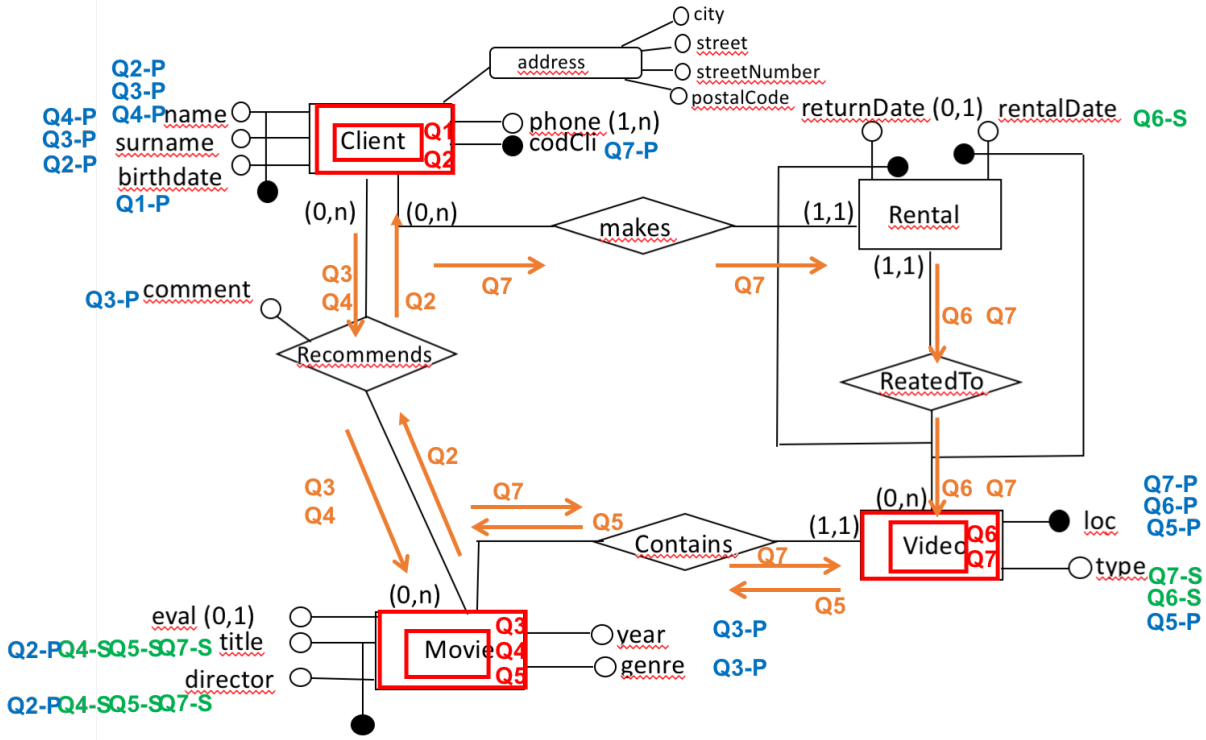
- A schema for S (in metanotation + + partition keys + indexes)

ASSUMPTION: all selection conditions are equality based

Issues

- From the aggregate-oriented logical schemas in meta-notation, the corresponding aggregation entity and the set of associated queries
  to a set of collections for S
- Each collection allows one subset of the queries to be executed
- From the set of selection attributes and the identifiers of the aggregation entity to the partition key and indexes

Input



- client: {<u>name, surname, birthdate</u>, recommends: [{title, director}]}
  - Q1, Q2
- movie: {<u>title, director</u>, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}
  - Q3, Q4, Q5
- video: {<u>loc</u>, type, rentals: [{rentalDate, codCli}], title, director}
  - Q6, Q7

Design in Cassandra

Aggregation entity Client

client: {name, surname, birthdate, recommends: [{title, director}]}

Queries associated with Client: Q1, Q2

Selection attributes for Q1: { }
Selection attributes for Q2: { }

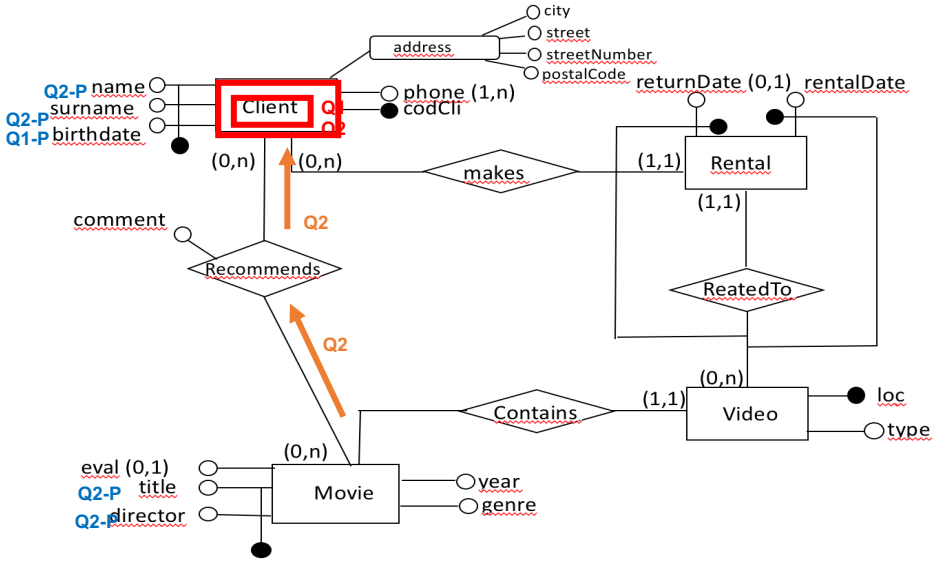
Q1. Average age of clients
Q2. Name and surname of clients and related recommended movies

No selection attribute → no need for a specific partition key

The primary key is taken from the Client identifier (name, surname, birthday)

Partition key = primary key = { name, surname, birthday }

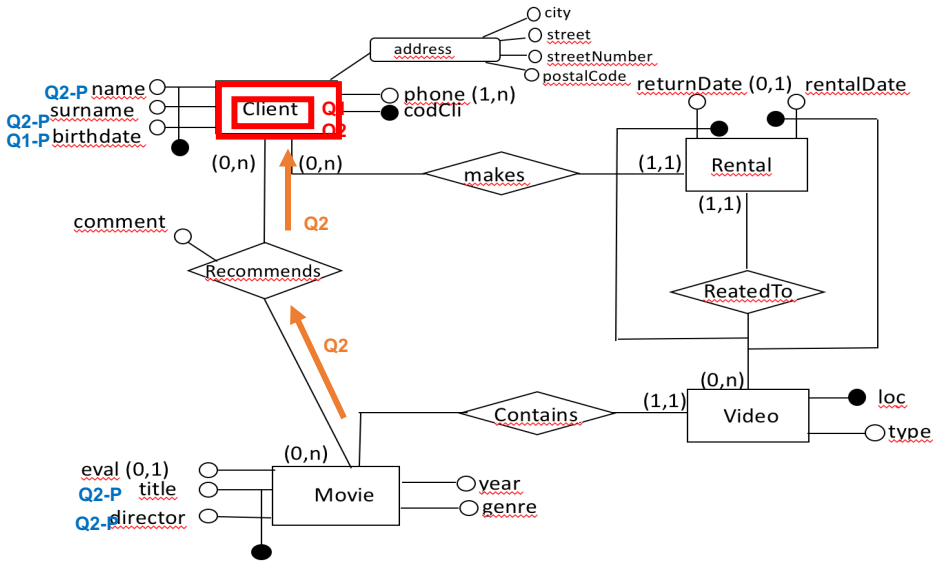
but any other alternative is fine



Aggregation entity Client – option 1 (UDT)

client: {name, surname, birthdate, recommends: [{title, director}]}

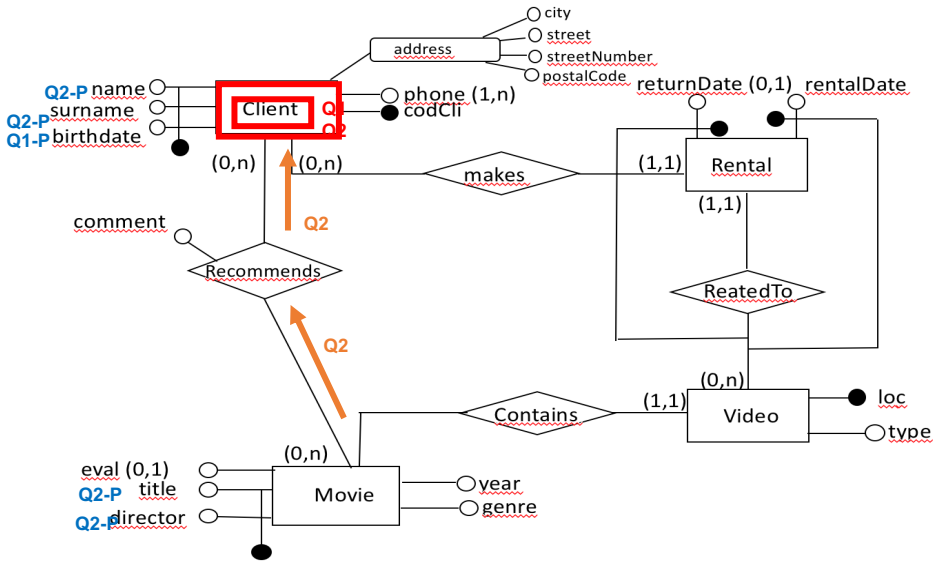
```
CREATE TYPE movie_t(
title text,
director text);
```




```
CREATE TABLE Clients (
name text,
surname text,
birthdate date,
recommends set<frozen<movie_t>>,
PRIMARY KEY (name, surname, birthday));
```

Aggregation entity Client – option 2 (collection types)

client: {name, surname, birthdate, recommends: [{title, director}]}




```
CREATE TABLE Clients (
name text,
surname text,
birthdate date,
recommends set<frozen<map<text, text>>>,
PRIMARY KEY (name, surname, birthday));
```

Cassandra design rules

Let A be the aggregate and Q_A the queries associated with A

1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - You can set, e.g., partition key = primary key = aggregate identifier

Aggregation entity Movie

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}

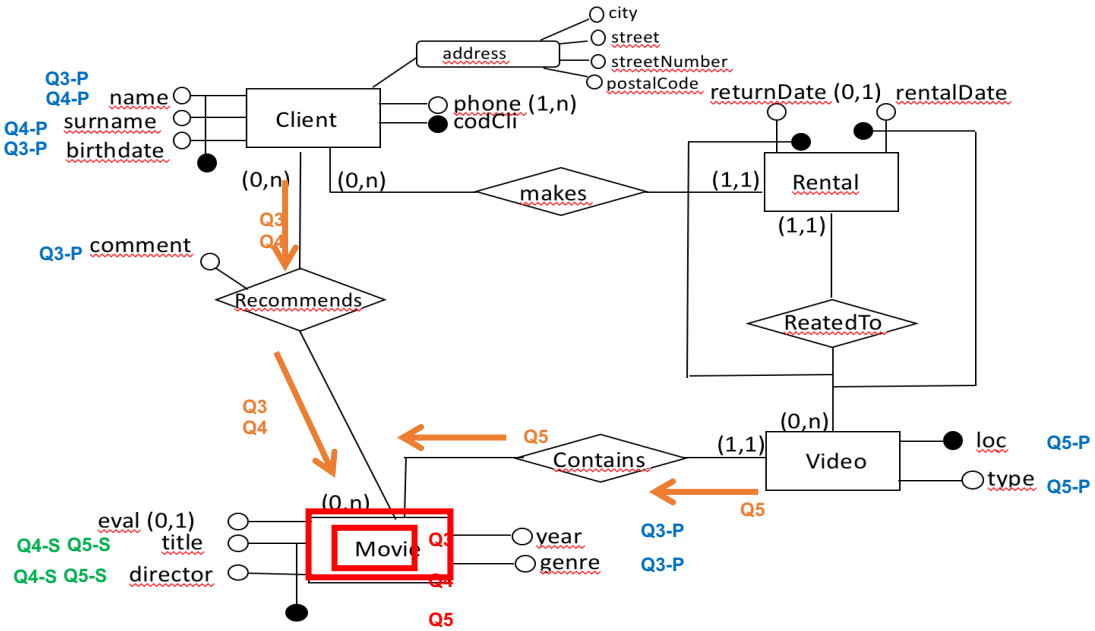
Queries associated with Movie: Q3, Q4, Q5

Selection attributes for Q3: { }
Selection attributes for Q4: { title, director }
Selection attributes for Q5: { title, director }

Q3. Genre and year of the movies and their related recommendations, together with the name and the surname of the client who made them
Q4. Name and surname of clients who recommended the movie 'pulp fiction' by 'quentin tarantino'
Q5. Given a movie, all information of videos that contain it

Title and director must appear in the primary key to execute Q4 and Q5
no other attribute is needed

Partition key = primary key = { title, director }

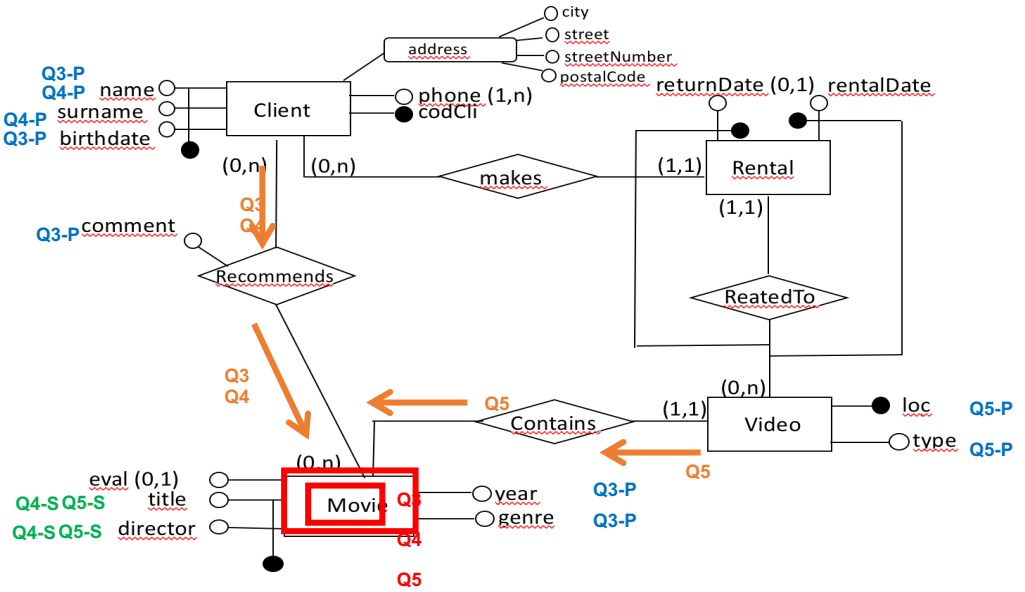


Aggregation entity Movie – option 1 (UDT)

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}

```
CREATE TYPE clientComment_t(
name text,
surname text,
comment text);
```

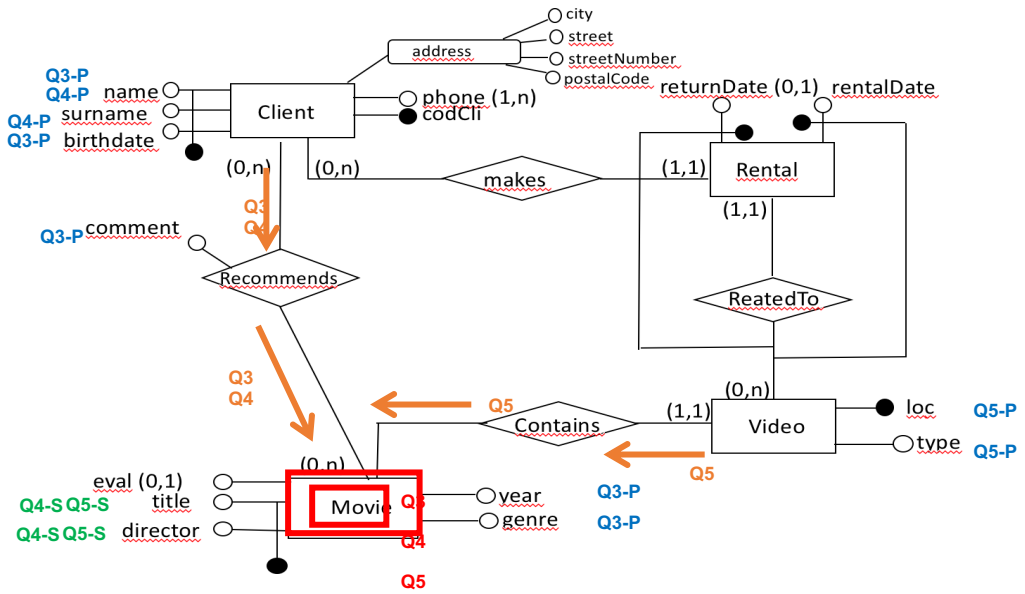
```
CREATE TYPE video_t(
loc integer,
type text);
```




```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by set<frozen<clientComment_t>>,
contained_in set<frozen<video_t>>,
PRIMARY KEY ((title, director));
```

Aggregation entity Movie – option 2 (collection types)

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}




```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by set<frozen<map<text, text>>>,
contained_in set<frozen<map<text,text>>>,
PRIMARY KEY ((title, director));
```

Cassandra design rules

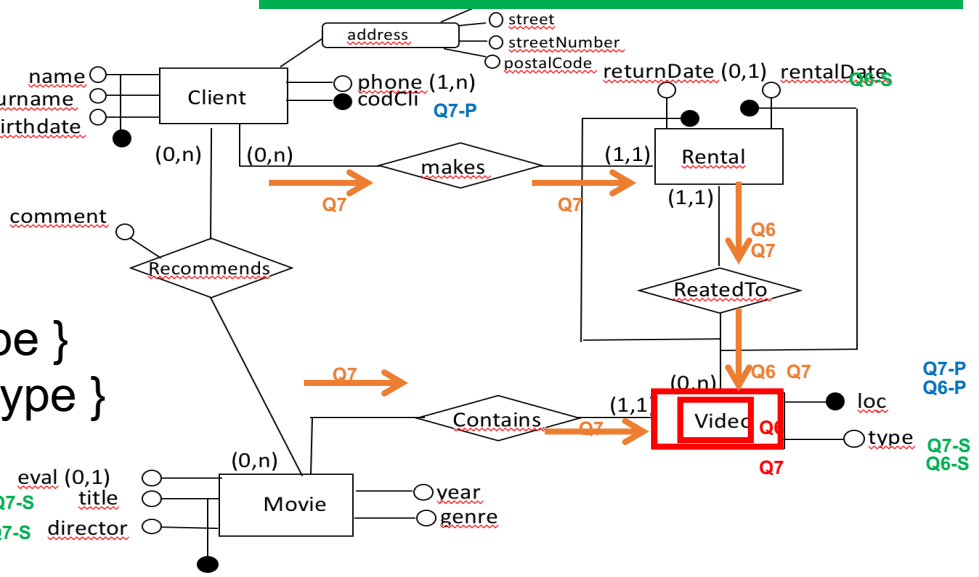
Let A be the aggregate and Q_A the queries associated with A

1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - Partition key = primary key = aggregate identifier
2. If queries in Q_A share the same set of selection attributes or the set of selection attribute is empty
   - The shared set of selection attributes become the partition key
   - If needed, add the aggregate identifier to the partition key to obtain the primary key

Aggregation entity Video

Q6. Videos of type 'DVD', rented from a certain date
Q7. The videos of type 'VHS' containing the movie 'pulp fiction' by 'quentin tarantino' and the clients that rented them

video: {loc, type, rentals: [{rentalDate, codCli}], title, director}



Queries associated with Video: Q6, Q7

Selection attributes for Q6: { rentalDate, type }
Selection attributes for Q7: { title, director, type }

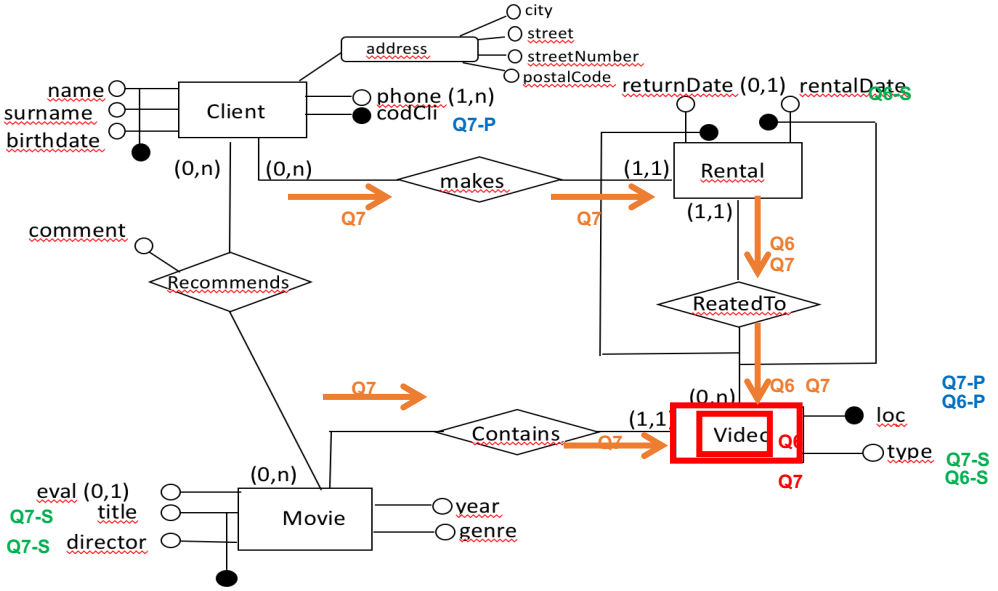
rentalDate is a nested attribute → it cannot belong to the primary key
type appears in both → it becomes the partition key (an equality is always specified)
title and director become clustering columns
To define the primary key, we need a video identifier → loc

Partition key = { type }
Primary key = { type, title, director, loc }

Aggregation entity Video – option 1 (UDT)

video: {loc, type, rentals: [{rentalDate, codCli}], title, director}

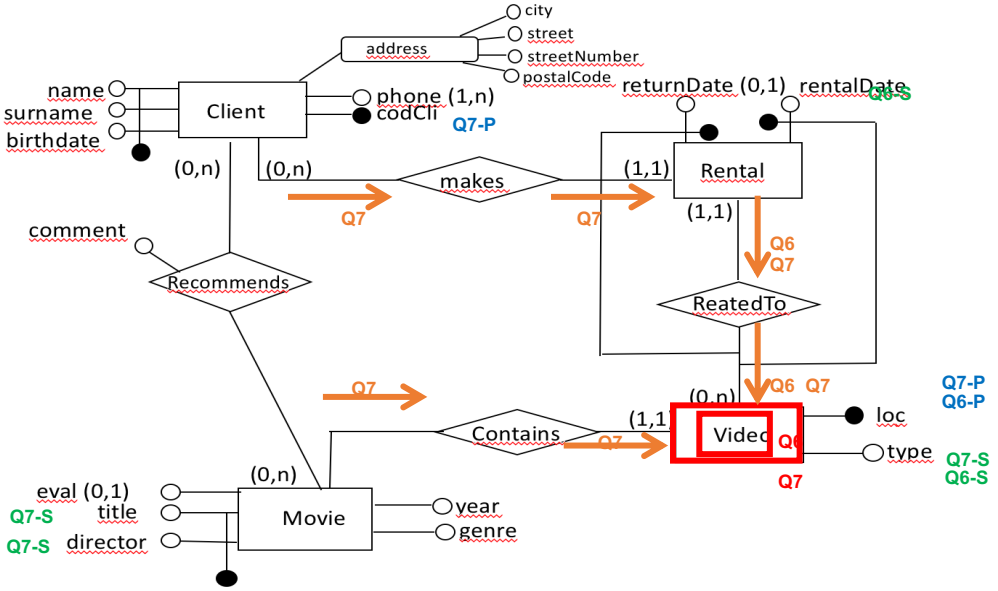
```
CREATE TYPE clientRentals_t(
rentalDate date,
codCli int);
```




```
CREATE TABLE Videos (
loc int,
type text,
rentals set<frozen<clientRentals_t>>,
title text,
director text,
PRIMARY KEY (type, title, director, loc));
```

Aggregation entity Video – option 2 (collection types)

video: {loc, type, rentals: [{rentalDate, codCli}], title, director}




```
CREATE TABLE Videos (
loc int,
type text,
rentals set<frozen<map<text, text>>>,
title text,
director text,
PRIMARY KEY (type, title, director, loc));
```

Aggregation entity Video: problem!

```
CREATE TYPE clientRentals_t(
rentalDate date,
codCli int);
```

```
CREATE TABLE Videos (
loc int,
type text,
rentals set<frozen<clientRentals_t>>,
title text,
director text,
PRIMARY KEY (type, title, director, loc));
```

In both cases,
conditions on rentalDate
cannot be specified, neither
creating some index or specifying the ALLOW FILTERING clause!

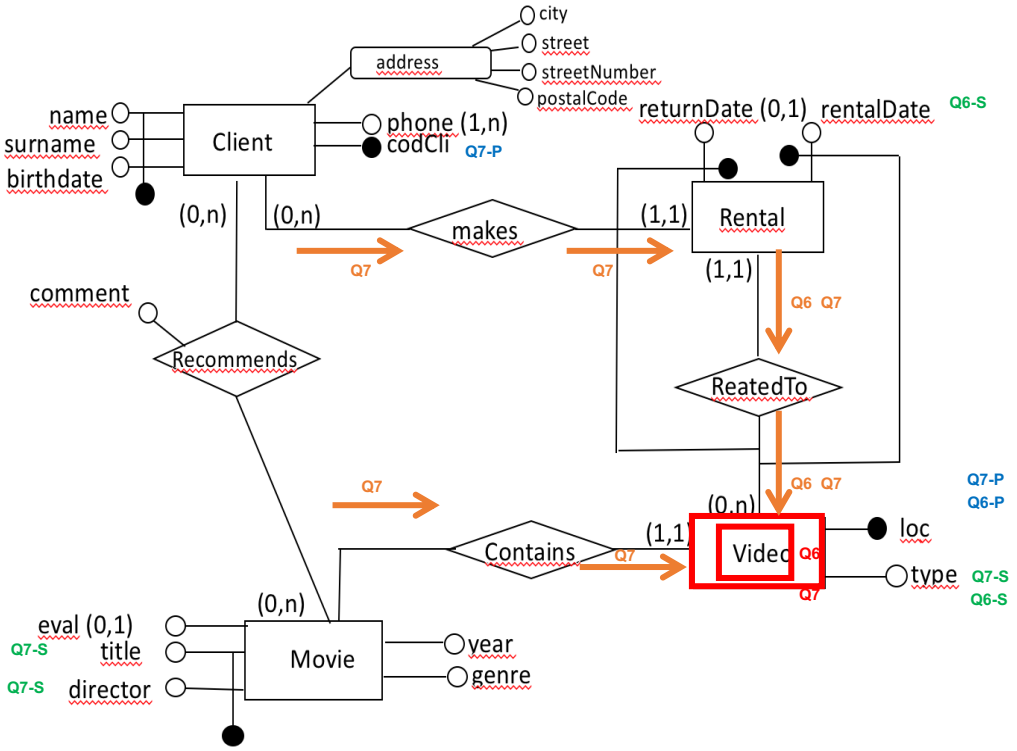
```
CREATE TABLE Videos (
loc int,
type text,
rentals set<frozen<map<text, text>>>,
title text,
director text,
PRIMARY KEY (type, title, director, loc));
```

Back to the selection of the aggreg. entity

- The problem is due to the selection of the aggregation entity during the logical design phase

  Q6 (Video, [Video(type)_!, Rental(rentalDate)_Rt], [Video(loc)_!])
  Q7 (Video, [Video(type)_!, Movie(title, director)_C] [Video(loc)_!, Client(CodCli)_MRt]))

- Q6: the aggregation entity is an entity from the (0,n) side of an association, this leads to a selection condition on a nested attribute



Back to the selection of the aggreg. entity

- Suppose we change the Q6 aggregation entity

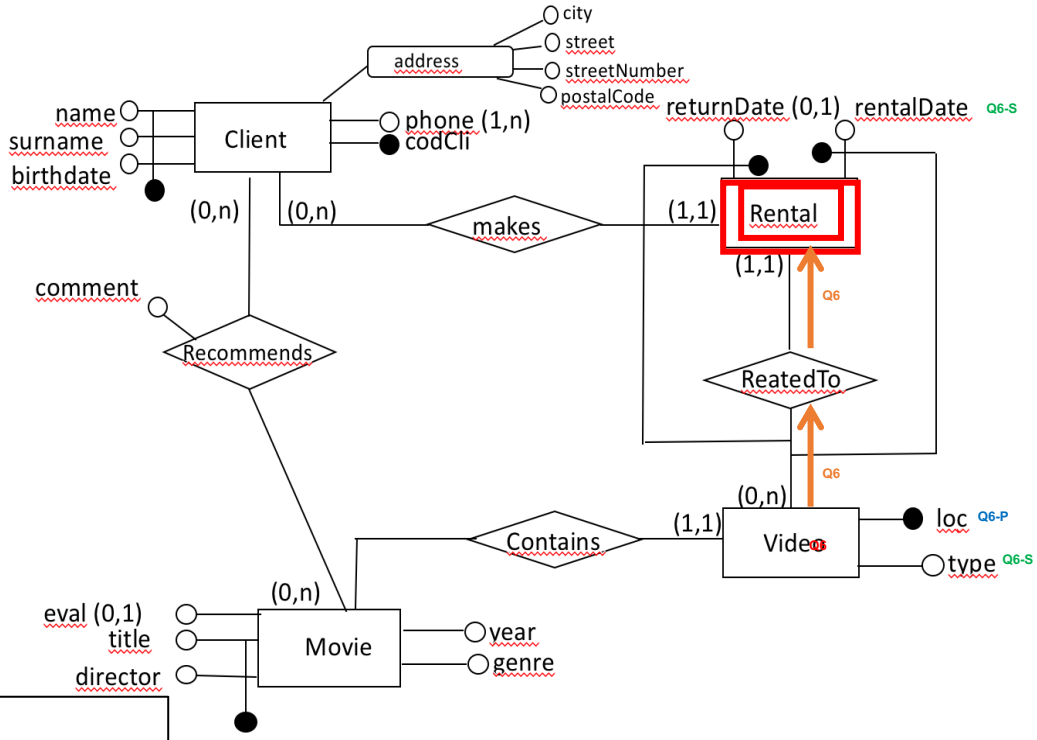
  Q6 (Video, [Video(type)_!, Rental(rentalDate)_Rt], [Video(loc)_!])

  →

  Q6 (Rental, [Video(type)_Rt, Rental(rentalDate) !], [Video(loc) Rt])

- Q6: now the aggregation entity is an entity from the (1,1) side of an association

rental: {rentalDate, loc, type}



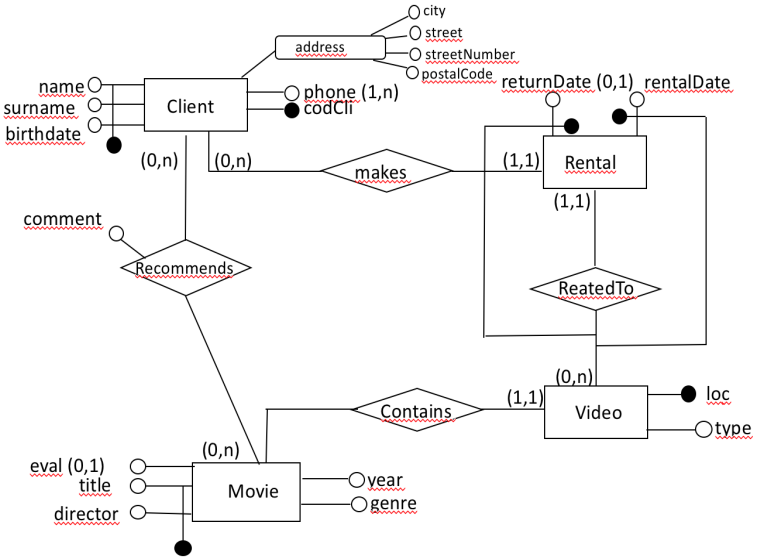

```
CREATE TABLE Rentals(
rentalDate date,
loc int,
type text,
PRIMARY KEY (rentalDate, type, loc));
```

Cassandra design rules

Let A be the aggregate and Q_A the queries associated with A

1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - Partition key = primary key = aggregate identifier
2. If queries in Q_A share the same set of selection attributes or the set of selection attribute is empty
   - The shared set of selection attributes become the partition key
   - If needed, add the aggregate identifier to the partition key to obtain the primary key
3. If one selection attribute is nested, it cannot be included in the primary key → the selection might not be allowed
   - During the aggregate design, favour aggregation entities from the one-side of one-to-many associations

Does it always work?



Determine the surname of clients with name «John» that recommended one film produced in 1997

Q8(Client, [Client(name)_!, Movie(year)_R], [Client(surname)_!])

Q6-P

Q6-S

OR

Q8(Movie, [Client(name)_R, Movie(year)_!], [Client(surname)_R])

Does it always work?

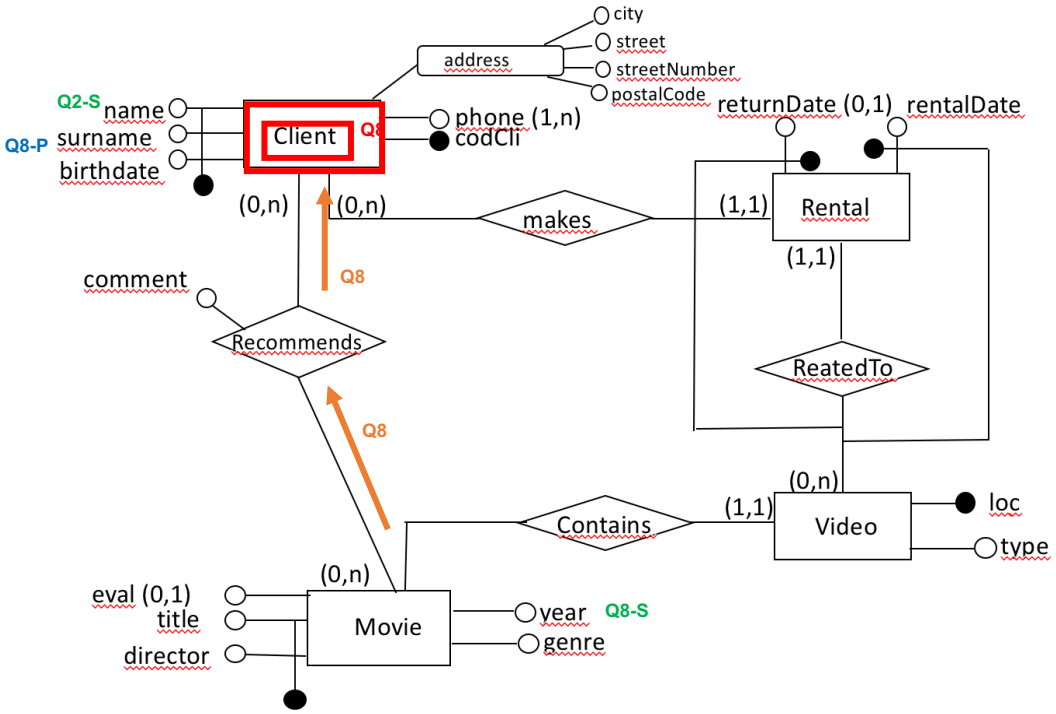
Determine the surname of clients with name «John» that recommended one film produced in 1997

Q8(Client, [Client(name)_!, Movie(year)_R], [Client(surname)_!]

client: {codcli, name, surname, recommends: [{year}] }

```
CREATE TABLE Clients (
codCli int,
name text,
surname text,
recommends set<text>,
PRIMARY KEY (name, codCli));
```

By creating an index on the set, the query can be specified

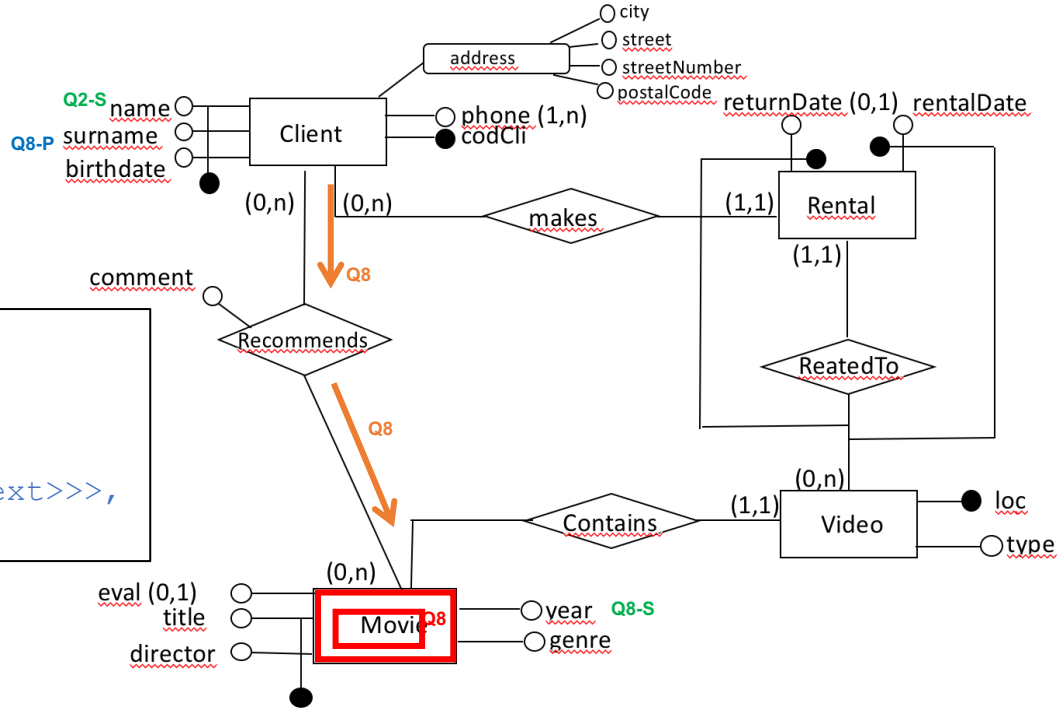


Does it always work?

Determine the surname of clients with name «John» that recommended one film produced in 1997

Q8(Movie, [Client(name)_R, Movie(year)_!], [Client(surname)_R])

movie: {title, director, year, recommended_by: [{name, surname}] }




```
CREATE TABLE Movies (
title text,
director text,
year int,
recommended_by set<frozen<map<text, text>>>,
PRIMARY KEY (year, title, director));
```

The query cannot be specified

Cassandra design rules

Let A be the aggregate and Q_A the queries associated with A

1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - Partition key = primary key = aggregate identifier
2. If queries in Q_A share the same set of selection attributes or the set of selection attribute is empty
   - The shared set of selection attributes become the partition key
   - If needed, add the aggregate identifier to the partition key to obtain the primary key
3. If one selection attribute is nested, it cannot be included in the primary key → the selection might not be allowed
   - During the aggregate design, favour aggregation entities from the one-side of one-to-many associations
4. Indexes allow the selection of atomic values inside sets/lists/maps
   - During the aggregate design, favour nesting of single attributes

Does it always work?

Determine the surname of clients with name «John» that recommended one film produced in 1997, together with the genre of the film

Q8(Client, [Client(name)_!, Movie(year)_R], [Client(surname)_!, Movie(genre)_R])

OR

Q8(Movie, [Client(name)_R, Movie(year)_!], [Client(surname)_R, Movie(genre)_!])

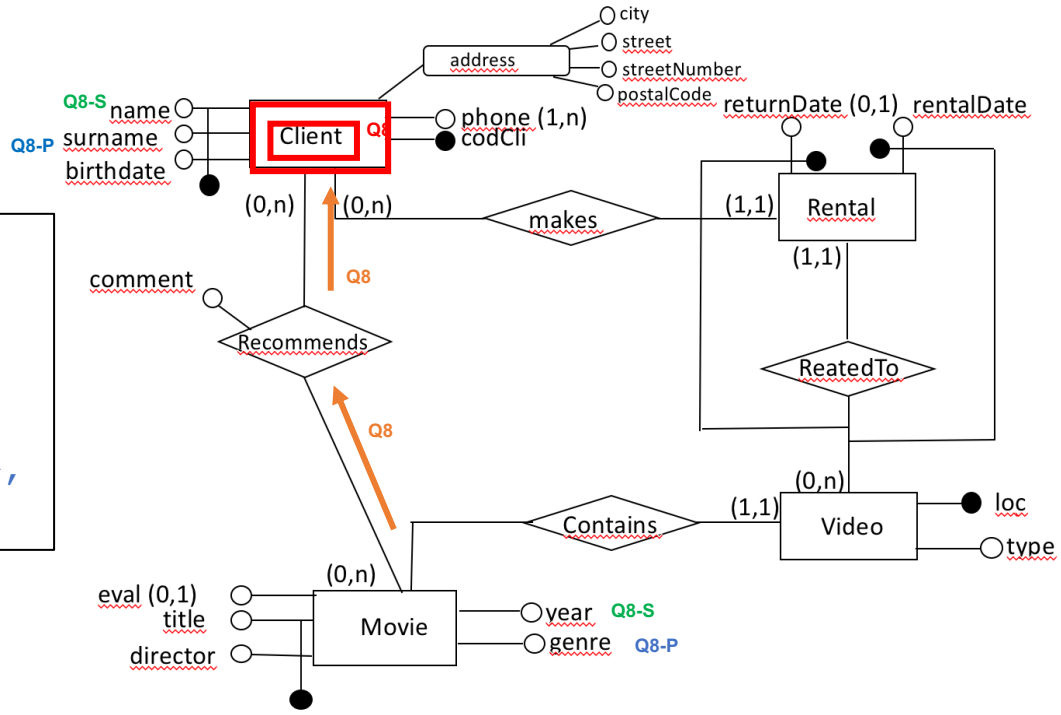
Does it always work?

Determine the surname of clients with name «John» that recommended one film produced in 1997, together with the genre of the film

Q8(Client, [Client(name)_!, Movie(year)_R], [Client(surname)_!, Movie(genre)_R])

client: {codCli, name, surname, recommends: [{year, genre}] }

```
CREATE TABLE Clients (
codCli int,
name text,
surname text,
recommends
    set<frozen<map<text,text>>>,
PRIMARY KEY (name, codCli));
```



The query cannot be specified (condition on a subcomponent of recommends, frozen)!

Does it always work?

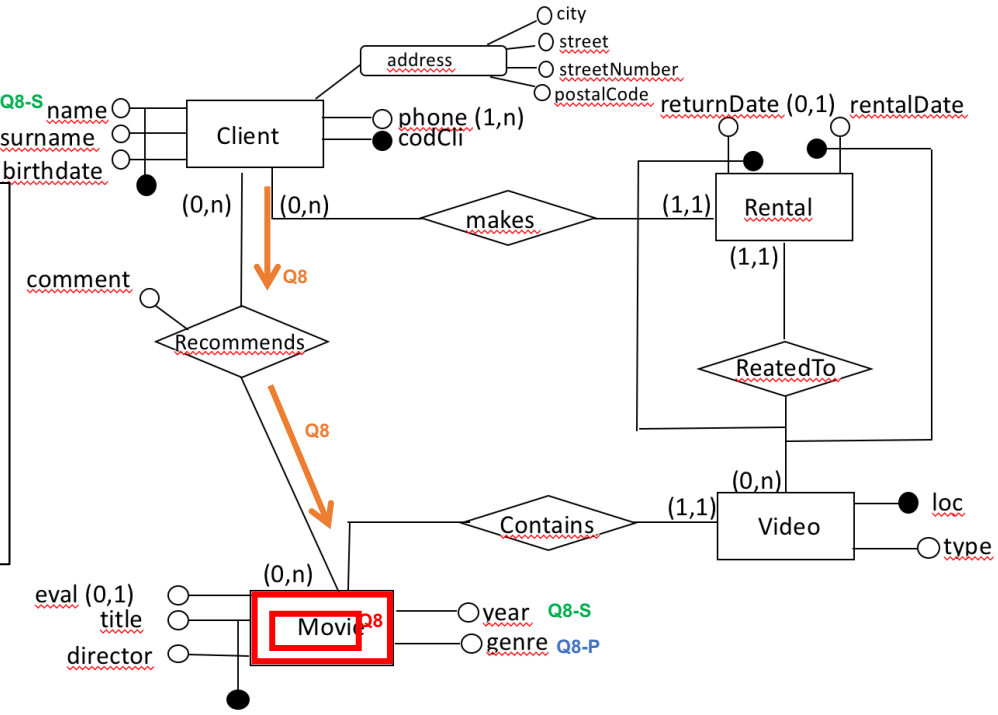
Determine the surname of clients with name «John» that recommended one film produced in 1997 together with the genre of the film

Q8(Movie, [Client(name)_R, Movie(year)_!], [Client(surname)_R])

movie: {title, director, year, recommended_by: [{name, surname}] }

```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by
     set<frozen<map<text, text>>>,
PRIMARY KEY (year, title, director));
```

The query cannot be specified (condition on a subcomponent of recommended_by, frozen)!



Does it work?

- The only possible solution in this case is to change the design approach and create a table corresponding to the many-to-many association

recommendation: {codCli, title, director, name, surname, year, genre}

```
CREATE TABLE Recommendation (
codCli int,
name text,
surname text,
title text,
director text,
year int,
genre text,
PRIMARY KEY ((name, year), codCli, title, director);
```

Cassandra design rules

Let A be the aggregate and Q_A the queries associated with A

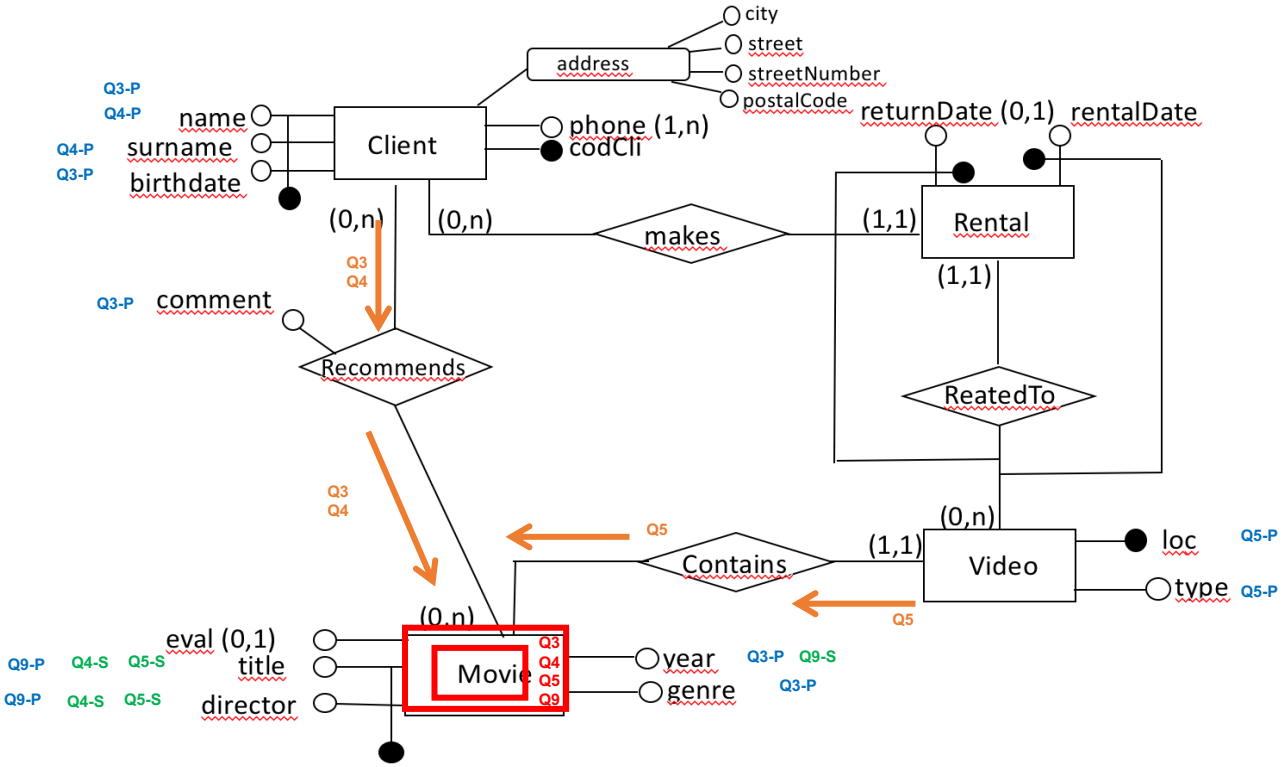
1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - Partition key = primary key = aggregate identifier
2. If queries in Q_A share the same set of selection attributes or the set of selection attribute is empty
   - The shared set of selection attributes become the partition key
   - If needed, add the aggregate identifier to the partition key to obtain the primary key
3. If one selection attribute is nested, it cannot be included in the primary key → the selection might not be allowed
   - During the aggregate design, favour aggregation entities from the one-side of one-to-many associations
4. Indexes allow the selection of atomic values inside sets/lists/maps
   - During the aggregate design, favour nesting of single attributes
5. Sometimes, in presence of many-to-many associations, during the aggregate design, new aggregates corresponding to the association and not to one entity should be taken into account to avoid selections over nested attributes

Back to Movie: new queries, more than one Cassandra table

Add a new query in the workload: determine the film produced in 1997

Q9 = (Movie, [Movie(year)_!], [Movie(title, director)_!])

The aggregate does not change with respect to the previous workload, however …



movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}

Back to Movie: new queries, more than one Cassandra table

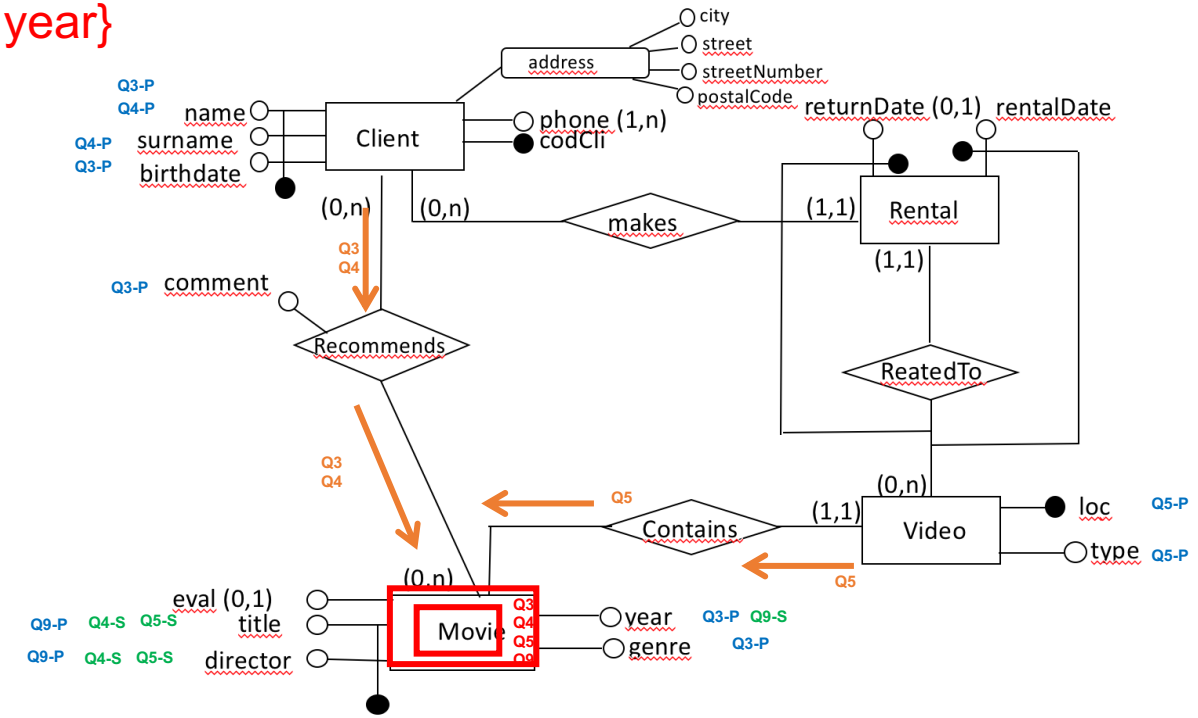
movie: {title, director, year, genre, recommended_by: [{name, surname, comment}],
contained_in: [{loc, type}]}

Queries associated with Movie: Q3, Q4, Q5, Q9

Selection attributes for Q3: { }
Selection attributes for Q4: { title, director }
Selection attributes for Q5: { title, director }
Selection attributes for Q9: { year}

Selection attributes of the queries are disjoint

How to define the partition key and the primary key?



Back to Movie: new queries, more than one Cassandra table

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}],
contained_in: [{loc, type}]}

Queries associated with Movie: Q3, Q4, Q5, Q9

Selection attributes for Q3: { }
Selection attributes for Q4: { title, director }
Selection attributes for Q5: { title, director }
Selection attributes for Q9: { year}

PRIMARY KEY ((title, director), year)	only Q3, Q4, Q5	admitted
PRIMARY KEY (year, title, director)	only Q3, Q9	admitted

The only solution is to split the aggregate into two column-families

Back to Movie: new queries, more than one Cassandra table

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}

```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by set<frozen<map<text, text>>>,
contained_in set<frozen<map<text,text>>>,
PRIMARY KEY ((title, director), year);
```

For queries Q3, Q4, Q5

For queries Q3, Q9

```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by set<frozen<map<text, text>>>,
contained_in set<frozen<map<text,text>>>,
PRIMARY KEY (year, title, director);
```

Back to Movie: new queries, more than one Cassandra table

movie: {title, director, year, genre, recommended_by: [{name, surname, comment}], contained_in: [{loc, type}]}

```
CREATE TABLE Movies (
title text,
director text,
year int,
genre text,
recommended_by set<frozen<map<text, text>>>,
contained_in set<frozen<map<text,text>>>,
PRIMARY KEY ((title, director), year;
```

For queries Q3, Q4, Q5

For queries Q9

```
CREATE TABLE Movies (
title text,
director text,
year int,
PRIMARY KEY (year, title, director);
```

Cassandra design rules

Let A be the aggregate and Q_A the queries associated with A

1. If the set of selection attributes of Q_A is empty, the partition key no matters
   - Partition key = primary key = aggregate identifier
2. If queries in Q_A share the same set of selection attributes or the set of selection attribute is empty
   - The shared set of selection attributes become the partition key
   - If needed, add the aggregate identifier to the partition key to obtain the primary key
3. If one selection attribute is nested, it cannot be included in the primary key → the selection might not be allowed
   - During the aggregate design, favour aggregation entities from the one-side of one-to-many associations
4. Indexes allow the selection of atomic values inside sets/lists/maps
   - During the aggregate design, favour nesting of single attributes
5. Sometimes, in presence of many-to-many associations, during the aggregate design, new aggregates corresponding to the association and not to one entity should be taken into account to avoid selections over nested attributes
6. If the non-empty set of selection attributes for queries in Q_A are disjoint, one aggregate has to be mapped in two or more Cassandra tables.