Tutorial 3: Relational Algebra and Tuple Relational Calculus

Informatics 1 Data & Analysis — Tutorial Notes

Week 5, Semester 2, 2018/19

This worksheet has three parts: tutorial *Questions*, followed by some *Examples* and their *Solutions*.

- Before your tutorial, work through and attempt all of the Questions in the first section. If you get stuck or need help then ask a question on *Piazza* or at *InfBASE*.
- The Examples are there for additional study, practice, and exam preparation.
- Use the Solutions to check your answers, and read about possible alternatives.

You must bring your answers to the main questions along to your tutorial. You will need to be able to show these to your tutor, and may be exchanging them with other students, so it is best to have them printed out on paper.

If you cannot do some questions, write down what it is that you find challenging and use this to ask your tutor in the meeting.

Tutorials will not usually cover the Examples, but if you have any questions about those then write them down and ask your tutor, or post a question on *Piazza*.

It's important both for your learning and other students in the group that you come to tutorials properly prepared. Students who have not attempted the main tutorial questions will be sent away from the tutorial to do them elsewhere and return later.

Some exercise sheets contain material marked with a star $\star$. These are optional extensions.

Data & Analysis tutorials are not formally assessed, but the content is examinable and they are an important part of the course. If you do not do the exercises then you are unlikely to pass the exam.

Attendance at tutorials is obligatory: if you are ill or otherwise unable to attend one week then email your tutor, and if possible attend another tutorial group in the same week.

Please send any corrections and suggestions to Ian.Stark@ed.ac.uk

Introduction

In this tutorial, you will construct queries in *tuple relational calculus* and describe operations to compute their results using *relational algebra*. These systems were introduced, with examples, in the lectures. All questions in this tutorial are based on a set of relational tables dealing with air travel: airports, flights, bookings and seats. You may find this tutorial a bit more difficult than the previous ones. If you need any help, please: look at the sample solutions at the end

of the exercises or ask on *Piazza*. If you are stuck with any question, write down what you are finding difficult and then move on to try the next one.

An Example Relational Model for Flight Bookings

Figures 1 and 2 on the following pages set out some data declarations and tables from a relational model for air travel bookings. This is a very simplified model — in particular, we don't deal with times or dates of individual flights. Notice that we are taking advantage of SQL's case insensitivity for keywords, with **create table** and **primary key** instead of **CREATE TABLE** and **PRIMARY KEY**.

Question 1: Operations in Relational Algebra

For each of the following queries in relational algebra, decide whether it is well-formed; and if it is then calculate the output table and give a brief statement of what information it gives.

(a) $\sigma_{\mathsf{class='Business'}}(\mathsf{Seat})$

(Tutor Notes) Retrieves details of all seats in business class. The output table is as follows.

seatNo	flightNo	class
12D	AF1232	Business
10A	BA2944	Business
5D	BA4060	Business

(b) $\pi_{\mathsf{nationality}}(\mathsf{Booking})$

(Tutor Notes) Retrieves all distinct nationalities of booked passengers. The output table is as follows:

nationality
British
French

Note that relational algebra (unlike SQL) deals in sets rather than multisets, so we don't have duplicate rows.

```
create table Airport (
    airportId   varchar(3),
    name        varchar(50),
    city        varchar(30),
    primary key (airportId)
)

create table Flight (
    flightNo          varchar(6),
    flightCompany     varchar(20),
    depAirport        varchar(3),
    arrAirport        varchar(3),
    primary key (flightNo),
    foreign key (depAirport)  references Airport(airportId),
    foreign key (arrAirport)  references Airport(airportId)
)

create table Booking (
    ticketNo    varchar(9),
    name        varchar(20),
    nationality varchar(20),
    flightNo    varchar(6),
    seatNo      varchar(3),
    primary key (ticketNo),
    foreign key (flightNo) references Flight,
    foreign key (seatNo, flightNo) references Seat
)

create table Seat (
    seatNo    varchar(3),
    flightNo  varchar(6),
    class     varchar(10),
    primary key (seatNo, flightNo),
    foreign key (flightNo) references Flight
)
```

Figure 1: Data Declarations

Airport

airportId	name	city
LHR	Heathrow	London
LGW	Gatwick	London
CDG	Charles de Gaulle	Paris
ORY	Orly	Paris

Flight

flightNo	flightCompany	depAirport	arrAirport
AF1231	Air France	LHR	CDG
AF1232	Air France	CDG	LHR
AF1234	Air France	LGW	CDG
AF1235	Air France	CDG	LGW
BA2943	British Airways	LGW	ORY
BA2944	British Airways	ORY	LGW
BA4059	British Airways	LHR	CDG
BA4060	British Airways	CDG	LHR

Booking

ticketNo	name	nationality	flightNo	seatNo
EAG129489	John Jones	British	AF1232	12D
EAF123456	Fraser McEwan	British	AF1232	30E
ABS958332	Mathilde Duval	French	BA2944	10A
ORE394895	Fiona Stewart	British	BA4060	5D
EYR149583	Karen Woods	British	BA4059	14B
EAG348595	Pierre Fontaine	French	BA2944	30D

Seat

seatNo	flightNo	class
12D	AF1232	Business
30E	AF1232	Economy
10A	BA2944	Business
5D	BA4060	Business
14B	BA4059	Economy
30D	BA2944	Economy

Figure 2: Table Contents

(c) $\sigma_{\text{nationality='French'}}(\text{Booking}) \times \sigma_{\text{class='Business'}}(\text{Seat})$

(Tutor Notes) Retrieves all information about all combinations of bookings by French passengers and possible business class seats. The output table is as follows:

ticketNo	name	nationality	flightNo	seatNo	seatNo	flightNo	class
ABS958332	Mathilde Duval	French	BA2944	10A	12D	AF1232	Business
ABS958332	Mathilde Duval	French	BA2944	10A	10A	BA2944	Business
ABS958332	Mathilde Duval	French	BA2944	10A	5D	BA4060	Business
EAG348595	Pierre Fontaine	French	BA2944	30D	12D	AF1232	Business
EAG348595	Pierre Fontaine	French	BA2944	30D	10A	BA2944	Business
EAG348595	Pierre Fontaine	French	BA2944	30D	5D	BA4060	Business

Note that the seatNo and flightNo fields appear twice: one is the passenger's actual booking, and the other is the possible business class seat. Because this is a cross-product, not a join, there is no connection between these fields.

(d) $\sigma_{\text{nationality='French'}}(\text{Booking}) \bowtie \sigma_{\text{class='Business'}}(\text{Seat})$

(Tutor Notes) Retrieves all information about business-class bookings by French passengers. The output table is as follows:

ticketNo	name	nationality	flightNo	seatNo	class
ABS958332	Mathilde Duval	French	BA2944	10A	Business

Note that the seatNo and flightNo fields no longer appear twice: the natural join projects away those columns, which are guaranteed to be duplicates.

(e) $\text{Booking} \bowtie \text{Flight}$

(Tutor Notes) Retrieves the ticket numbers, passenger names, nationalities, flight numbers, seat numbers, flight company, departure and arrival airports for all bookings. This is a natural join on the flightNo field which appears in both relations. The output table is as follows:

ticketNo	name	nationality	flightNo	seatNo	flightCompany	depAirport	arrAirport
EAG129489	John Jones	British	AF1232	12D	Air France	CDG	LHR
EAF123456	Fraser McEwan	British	AF1232	30E	Air France	CDG	LHR
ABS958332	Mathilde Duval	French	BA2944	10A	British Airways	ORY	LGW
ORE394895	Fiona Stewart	British	BA4060	5D	British Airways	CDG	LHR
EYR149583	Karen Woods	British	BA4059	14B	British Airways	LHR	CDG
EAG348595	Pierre Fontaine	French	BA2944	30D	British Airways	ORY	LGW

(f) $\pi_{\mathsf{name}}(\sigma_{\mathsf{depAirport='CDG'}}(\mathsf{Booking} \bowtie \mathsf{Flight}))$

(Tutor Notes) Retrieves the names of all passengers travelling from Charles de Gaulle airport. The output table is as follows:

name
John Jones
Fraser McEwan
Fiona Stewart

(g) $\mathsf{Airport} \cup \mathsf{Seat}$

(Tutor Notes) This operation is not valid, as the two tables are not compatible, i.e. even though they have the same number of fields, the corresponding fields (from left to right) do not have the same names or domains.

Question 2: Constructing Queries

For each of the following questions, formulate the specified queries in tuple relational calculus and as a computation in relational algebra.

(a) Retrieve all information about airports in London. The schema of the output table should be same as that of the Airport table.

(Tutor Notes) In tuple relational calculus, this is expressed as follows:

$$\{ A \mid A \in \mathsf{Airport} \wedge A.\mathsf{city} = \mathsf{'London'} \} .$$

That can be abbreviated slightly in this equivalent expression:

$$\{ A \in \mathsf{Airport} \mid A.\mathsf{city} = \mathsf{'London'} \} .$$

Computing this in relational algebra uses a selection operation:

$$\sigma_{\mathsf{city='London'}}(\mathsf{Airport}) .$$

(b) Retrieve details of all bookings by British and French passengers. The schema of the output table should be same as that of the Booking table.

(Tutor Notes)

$$\{\, B \mid B \in \mathsf{Booking} \wedge (B.\mathsf{nationality} = \texttt{'British'} \vee B.\mathsf{nationality} = \texttt{'French'}) \,\}$$

Again, this can be rewritten using an abbreviation:

$$\{\, B \in \mathsf{Booking} \mid B.\mathsf{nationality} = \texttt{'British'} \vee B.\mathsf{nationality} = \texttt{'French'} \,\} .$$

This is only a mild change of syntax — the underlying expression is the same. In contrast, the following terms show computationally distinct ways to evaluate this query in relational algebra:

$$\sigma_{(\mathsf{nationality}=\texttt{'British'} \vee \mathsf{nationality}=\texttt{'French'})}(\mathsf{Booking})$$
$$\text{or} \quad \sigma_{\mathsf{nationality}=\texttt{'British'}}(\mathsf{Booking}) \cup \sigma_{\mathsf{nationality}=\texttt{'French'}}(\mathsf{Booking}) .$$

The first of these carries out a single selection operation on the Booking table, using a slightly complex predicate; while the second performs two separate selection operations and then combines the resulting tables. Each method gives the same final result, but when executed may take different amounts of time and storage space.

(c) Retrieve the names of all passengers.

(Tutor Notes)

$$\{\, P \mid \exists B \in \mathsf{Booking} \,.\, P.\mathsf{name} = B.\mathsf{name} \,\}$$
$$\pi_{\mathsf{name}}(\mathsf{Booking})$$

The tuple-relational expression here describes the set of tuples P such that there is a booking record B where P has a name field equal to $B.\mathsf{name}$. Some students suggest the following instead:

$$\{\, B.\mathsf{name} \mid B \in \mathsf{Booking} \,\} .$$

That isn't a statement in TRC: most immediately because a TRC expression describes a set of tuples, and $B.\mathsf{name}$ is not a tuple but a single field. It would be possible to imagine extending TRC with some syntax describing a way to construct tuples from named fields. However, this would miss the point of the tuple relational calculus: it's a pure specification language, stating only what properties the elements of a set must have and not how they are constructed.

Relational algebra, in contrast, is a language stating only how to construct a set, without specifying its properties. The key insight underlying relational databases is that these two languages are equally expressive: anything specified in TRC can be computed in relational algebra; and anything that relational algebra computes can be specified in TRC. That's one of the things these tutorial exercises are exploring: how the same range of things can be expressed in two quite different languages.

The practical database language SQL combines elements of both: with specification information in **where** clauses, projection in the field selectors of **select**, and individual operations of relational algebra like **union**. The results about language expressivity ensure that all of this can always be computed by an underlying relational engine.

(d) Retrieve the flight number, Departure and Arrival airports of all British Airways flights.

(Tutor Notes)

$$\{ B \mid \exists F \in \mathsf{Flight} \,.\, F.\mathsf{flightCompany} = \text{'British Airways'} \wedge B.\mathsf{flightNo} = F.\mathsf{flightNo} \wedge B.\mathsf{depAirport} = F.\mathsf{depAirport} \wedge B.\mathsf{arrAirport} = F.\mathsf{arrAirport} \}$$

$$\pi_{\mathsf{flightNo},\mathsf{depAirport},\mathsf{arrAirport}}(\sigma_{\mathsf{flightCompany}=\text{'British Airways'}}(\mathsf{Flight}))$$

(e) Retrieve the name of every passenger together with their flight number and the associated flight company.

(Tutor Notes)

$$\{ P \mid \exists B \in \mathsf{Booking}, F \in \mathsf{Flight} \,.\, B.\mathsf{flightNo} = F.\mathsf{flightNo} \wedge P.\mathsf{name} = B.\mathsf{name} \wedge P.\mathsf{flightNo} = B.\mathsf{flightNo} \wedge P.\mathsf{flightCompany} = F.\mathsf{flightCompany} \}$$

$$\pi_{\mathsf{name},\mathsf{flightNo},\mathsf{flightCompany}}(\mathsf{Booking} \bowtie \mathsf{Flight})$$

This is a natural join between the two tables, so doesn't need the predicate or projection explicitly specified.

The following questions are all marked with a star $\star$. This indicates that they are optional — you are encouraged to attempt all you can, but they are not a requirement for tutorials.

$\star$ **(f)** Retrieve details of every flight leaving from any airport in London. The output schema should be same as that of Flight table.

(Tutor Notes)

$$\{ F \in \mathsf{Flight} \mid \exists A \in \mathsf{Airport} \,.\, F.\mathsf{depAirport} = A.\mathsf{airportId} \wedge A.\mathsf{city} = \texttt{'London'} \}$$

$$\pi_{\mathsf{flightNo},\mathsf{flightCompany},\mathsf{depAirport},\mathsf{arrAirport}}(\sigma_{\mathsf{city}=\texttt{'London'}}(\mathsf{Flight} \bowtie_{\mathsf{depAirport}=\mathsf{airportId}} \mathsf{Airport}))$$

$$\pi_{\mathsf{flightNo},\mathsf{flightCompany},\mathsf{depAirport},\mathsf{arrAirport}}(\mathsf{Flight} \bowtie_{\mathsf{depAirport}=\mathsf{airportId}} (\sigma_{\mathsf{city}=\texttt{'London'}}(\mathsf{Airport})))$$

Either of these relational algebra expressions will do. Note the use of an explicit comparison in the equijoin "$\bowtie_{\mathsf{depAirport}=\mathsf{airportId}}$", rather than a natural join, because we are joining on two fields with different names.

Notice also that this is only looking for flights out of London, not into London: several students included both, although that is not in the question.

Some students have asked about nesting in the tuple relational calculus, suggesting expressions like this:

$$\{ F \in \mathsf{Flight} \mid \exists A \in \{ A \in \mathsf{Airport} \mid A.\mathsf{city} = \texttt{'London'} \} \,.\, F.\mathsf{depAirport} = A.\mathsf{airportId} \} \,.$$

Here the inner set specifies airports in London, with the main expression describing flights leaving from any such airport.

This kind of expression is not part of TRC as presented in the course, but it could be a reasonable extension. Unlike the notes earlier about adding operations to construct tuples, this is still a pure specification language. It might also be easier to follow when reading.

However, it's worth directly comparing with the first TRC expression given above. Note that both expressions contain exactly the same components: an $A \in \mathsf{Airport}$ with tests $A.\mathsf{city} = \texttt{'London'}$ and $F.\mathsf{depAirport} = A.\mathsf{airportId}$. Using nested expressions may feel like it is expanding the language, but in fact it doesn't seem to be adding much at all — perhaps surprisingly, you can end up writing the same thing but with a few more characters.

⋆ **(g)** Find out the ticket numbers and names of all passengers departing from London.

(Tutor Notes)

$$\{\, P \mid \exists B \in \mathsf{Booking}, F \in \mathsf{Flight}, A \in \mathsf{Airport} \;.\; B.\mathsf{flightNo} = F.\mathsf{flightNo} \wedge F.\mathsf{depAirport} = A.\mathsf{airportId} \wedge A.\mathsf{city} = \text{'London'} \wedge P.\mathsf{ticketNo} = B.\mathsf{ticketNo} \wedge P.\mathsf{name} = B.\mathsf{name} \,\}$$

One possible implementation of this is

$$\pi_{\mathsf{ticketNo,name}}(\sigma_{\mathsf{city}=\text{'London'}}((\mathsf{Booking} \bowtie \mathsf{Flight}) \bowtie_{\mathsf{depAirport}=\mathsf{airportId}} \rho_{(\mathsf{name}\to\mathsf{airportName})}(\mathsf{Airport})))$$

which uses renaming to avoid a field name clash. Alternatively, we can give fully-specified field names:

$$\pi_{\mathsf{ticketNo,Booking.name}}(\sigma_{\mathsf{city}=\text{'London'}}((\mathsf{Booking} \bowtie \mathsf{Flight}) \bowtie_{\mathsf{depAirport}=\mathsf{airportId}} \mathsf{Airport}))\;.$$

We could also move the selection inwards to first pick out the airportId of every airport in London, dropping airport names at an early stage:

$$\pi_{\mathsf{ticketNo,name}}(\mathsf{Booking} \bowtie (\pi_{\mathsf{airportId}}(\sigma_{\mathsf{city}=\text{'London'}}(\mathsf{Airport})) \bowtie_{\mathsf{airportId}=\mathsf{depAirport}} \mathsf{Flight}))\;.$$

This version rearranges the order of join operations to extract only flights from London before linking up to the Booking relation.

⋆ **(h)** Retrieve the flight number and company of all flights from London to Paris.

(Tutor Notes)

$$\{ R \mid \exists F \in \mathsf{Flight}, A1 \in \mathsf{Airport}, A2 \in \mathsf{Airport} \,.\; F.\mathsf{depAirport} = A1.\mathsf{airportId} \wedge F.\mathsf{arrAirport} = A2.\mathsf{airportId} \wedge A1.\mathsf{city} = \text{'London'} \wedge A2.\mathsf{city} = \text{'Paris'} \wedge R.\mathsf{flightNo} = F.\mathsf{flightNo} \wedge R.\mathsf{flightCompany} = F.\mathsf{flightCompany} \}$$

This query also has several possible realisations as expressions of relational algebra, each describing different possible routes to calculate the same result.

$$\pi_{\mathsf{flightNo},\mathsf{flightCompany}}(\sigma_{(\mathsf{depCity}=\text{'London'} \wedge \mathsf{arrCity}=\text{'Paris'})}((\mathsf{Flight} \bowtie \rho_{(\mathsf{airportId}\to\mathsf{depAirport},\mathsf{city}\to\mathsf{depCity},\mathsf{name}\to\mathsf{depAN})}(\mathsf{Airport})) \bowtie \rho_{(\mathsf{airportId}\to\mathsf{arrAirport},\mathsf{city}\to\mathsf{arrCity},\mathsf{name}\to\mathsf{arrAN})}(\mathsf{Airport}))))$$

The renaming here lets us use a natural join: otherwise the name and city fields of departure and arrival airports would become confused.

An alternative approach is to take the intersection of two simpler queries: all flights leaving from London and all flights going to Paris.

$$\pi_{\mathsf{flightNo},\mathsf{flightCompany}}(\sigma_{\mathsf{city}=\text{'London'}}(\mathsf{Flight} \bowtie_{\mathsf{airportId}=\mathsf{depAirport}} \mathsf{Airport})) \cap \pi_{\mathsf{flightNo},\mathsf{flightCompany}}(\sigma_{\mathsf{city}=\mathsf{Paris}}(\mathsf{Flight} \bowtie_{\mathsf{airportId}=\mathsf{depAirport}} \mathsf{Airport}))$$

Another method is to use selections deep inside the expression to pick out airports in London and airports in Paris, before joining these with the main Flight table.

$$\pi_{\mathsf{flightNo},\mathsf{flightCompany}}((\sigma_{\mathsf{city}=\text{'London'}}(\mathsf{Airport}) \bowtie_{\mathsf{airportId}=\mathsf{depAirport}} \mathsf{Flight}) \bowtie_{\mathsf{airportId}=\mathsf{arrAirport}} (\sigma_{\mathsf{city}=\mathsf{Paris}}(\mathsf{Airport})))$$

Examples

This section contains further exercises on constructing queries in tuple relational calculus and relational algebra. These examples are similar to the main tutorial questions: there is a relational model for a given domain with exercises to carry out some operations in relational algebra and construct queries in the tuple relational calculus.

After these questions there are solutions and notes on all the examples.

A Relational Model for Films

Figures 3 and 4 on the following pages describe a very small relational model of films, their actors and directors.

Example 1: Operations in Relational Algebra

For each of the following queries in relational algebra, calculate the output table and give a brief statement of what query it answers.

(a) $\sigma_{\mathsf{age}>45}(\mathsf{Actor})$

(b) $\pi_{\mathsf{title}}(\mathsf{Film})$

(c) $\pi_{\mathsf{title}}(\sigma_{\mathsf{yr}<2000}(\mathsf{Film}))$

(d) $\sigma_{\mathsf{yr}=2012}(\mathsf{Film}) \times \sigma_{\mathsf{nationality}\neq\text{'American'}}(\mathsf{Director})$

(e) $\sigma_{\mathsf{yr}=2012}(\mathsf{Film}) \bowtie \sigma_{\mathsf{nationality}\neq\text{'American'}}(\mathsf{Director})$

(f) $\pi_{\mathsf{title}}(\mathsf{Film} \bowtie \sigma_{\mathsf{nationality}=\text{'British'}}(\mathsf{Director}))$

(g) $\sigma_{\mathsf{yr}<2000}(\mathsf{Film}) \cup \sigma_{\mathsf{yr}>2010}(\mathsf{Film})$

(h) $\sigma_{\mathsf{yr}\geq2000}(\mathsf{Film}) \cap \sigma_{\mathsf{yr}\leq2010}(\mathsf{Film})$

Example 2: Constructing Queries

For each of the following questions, formulate the specified queries in tuple relational calculus and as a computation in relational algebra.

(a) Retrieve details of all films that were released in 2010. The output schema should be the same as that of the Film table.

(b) Retrieve details of all actors that are not in their thirties. The output schema should be the same as that of the Actor table.

(c) Retrieve the names of all directors.

(d) Retrieve the names of all American directors.

(e) Find out the names of all British actors above the age of 40.

(f) Retrieve the name of each actor together with the titles of the films she/he has performed in.

```
create table Actor (
    actorId      varchar(5),
    name         varchar(50),
    nationality  varchar(20),
    age          integer,
    primary key (actorId)
)

create table Film (
    filmId       varchar(5),
    title        varchar(50),
    yr           integer,
    directorId   varchar(5),
    primary key (filmId),
    foreign key (directorId) references Director
)

create table Performance (
    actorId      varchar(5),
    filmId       varchar(5),
    part         varchar(50),
    primary key (actorId, filmId),
    foreign key (actorId) references Actor,
    foreign key (filmId) references Film
)

create table Director (
    directorId   varchar(5),
    name         varchar(50),
    nationality  varchar(20),
    primary key (directorId)
)
```

Figure 3: Data Declarations

Actor

actorId	name	nationality	age
LDC21	Leonardo DiCaprio	American	40
KW871	Kate Winslet	British	39
CB379	Christian Bale	British	40
MKE12	Michael Keaton	American	63
JGL81	Joseph Gordon-Levitt	American	33
EMG32	Ewan McGregor	British	43
HBC54	Helena Bonham Carter	British	48

Film

filmId	title	yr	directorId
INC10	Inception	2010	CN345
TIT97	Titanic	1997	JC212
RR008	Revolutionary Road	2008	SM521
SKF12	Skyfall	2012	SM521
SHI10	Shutter Island	2010	SCO78
DK008	The Dark Knight	2008	CN345
DKR12	The Dark Knight Rises	2012	CN345
BAT92	Batman Returns	1992	BUR34
FISH4	Big Fish	2003	BUR34

Performance

actorId	filmId	part
LDC21	INC10	Dominic Cobb
LDC21	TIT97	Jack Dawson
KW871	TIT97	Rose DeWitt Bukater
LDC21	RR008	Frank Wheeler
KW871	RR008	April Wheeler
LDC21	SHI10	Teddy Daniels
CB379	DK008	Bruce Wayne
CB379	DKR12	Bruce Wayne
JGL81	INC10	Arthur
MKE12	BAT92	Bruce Wayne
EMG32	FISH4	Ed Bloom
HBC54	FISH4	Jenny

Director

directorId	name	nationality
CN345	Christopher Nolan	British
JC212	James Cameron	Canadian
SM521	Sam Mendes	British
SCO78	Martin Scorsese	American
BUR34	Tim Burton	American

Figure 4: Table Contents

(g) Find out the names of all actors that have played the part of Bruce Wayne (Batman).

(h) Retrieve the names of all actors that have played the part of Bruce Wayne, together with the year the corresponding films were released.

(i) Retrieve all actors from the film Inception. The output schema should be the same as that of the Actor table.

(j) Find out the names of all actors that have performed in a film directed by Christopher Nolan.

(k) Retrieve the titles of all films in which Leonardo Di Caprio and Kate Winslet have co-acted.

(l) Assuming that the actorId and directorId values for actors and directors are consistent across the tables, retrieve details of all actors that have directed a film.

Solutions to Examples

These are not entirely "model" answers; instead, they indicate a possible solution. Remember that not all of these questions will have a single "right" answer. There can be multiple appropriate ways to formulate a query.

If you have difficulties with a particular example, or have trouble following through the solution, please raise this as a question in your tutorial.

Solution 1

(a) $\sigma_{\mathsf{age}>45}(\mathsf{Actor})$

Retrieves details of all actors above the age of 45. The output table is as follows:

actorId	name	nationality	age
MKE12	Michael Keaton	American	63
HBC54	Helena Bonham Carter	British	48

(b) $\pi_{\mathsf{title}}(\mathsf{Film})$

Retrieves all distinct film titles. The output table is as follows:

title
Inception
Titanic
Revolutionary Road
Skyfall
Shutter Island
The Dark Knight
The Dark Knight Rises
Batman Returns
Big Fish

(c) $\pi_{\mathsf{title}}(\sigma_{\mathsf{yr}<2000}(\mathsf{Film}))$

Retrieves all distinct titles of films that were released before 2000. The output table is as follows:

title
Titanic
Batman Returns

(d) $\sigma_{\mathsf{yr}=2012}(\mathsf{Film}) \times \sigma_{\mathsf{nationality}\neq\text{'American'}}(\mathsf{Director})$

Retrieves all information about all combinations of films released in 2012 and non-American directors. The output table is as follows:

filmId	title	yr	directorId	directorId	name	nationality
SKF12	Skyfall	2012	SM521	CN345	Christopher Nolan	British
SKF12	Skyfall	2012	SM521	JC212	James Cameron	Canadian
SKF12	Skyfall	2012	SM521	SM521	Sam Mendes	British
DKR12	The Dark Knight Rises	2012	CN345	CN345	Christopher Nolan	British
DKR12	The Dark Knight Rises	2012	CN345	JC212	James Cameron	Canadian
DKR12	The Dark Knight Rises	2012	CN345	SM521	Sam Mendes	British

(e) $\sigma_{\mathsf{yr}=2012}(\mathsf{Film}) \bowtie \sigma_{\mathsf{nationality}\neq\text{'American'}}(\mathsf{Director})$

Retrieves the details of all films released in 2012 and directed by a non-American director, along with the details of the corresponding director. The output table is as follows:

filmId	title	yr	directorId	name	nationality
SKF12	Skyfall	2012	SM521	Sam Mendes	British
DKR12	The Dark Knight Rises	2012	CN345	Christopher Nolan	British

(f) $\pi_{\mathsf{title}}(\mathsf{Film} \bowtie \sigma_{\mathsf{nationality}=\text{'British'}}(\mathsf{Director}))$

Retrieves all distinct titles of films directed by a British director. The output table is as follows:

title
Inception
Revolutionary Road
Skyfall
The Dark Knight
The Dark Knight Rises

(g) $\sigma_{\mathsf{yr}<2000}(\mathsf{Film}) \cup \sigma_{\mathsf{yr}>2010}(\mathsf{Film})$

Retrieves details of all films released before 2000 or after 2010. The output table is as follows:

filmId	title	yr	directorId
TIT97	Titanic	1997	JC212
SKF12	Skyfall	2012	SM521
DKR12	The Dark Knight Rises	2012	CN345
BAT92	Batman Returns	1992	BUR34

(h) $\sigma_{\mathsf{yr}\geq2000}(\mathsf{Film}) \cap \sigma_{\mathsf{yr}\leq2010}(\mathsf{Film})$

Retrieves details of all films released between 2000 and 2010. The output table is as follows:

filmId	title	yr	directorId
INC10	Inception	2010	CN345
RR008	Revolutionary Road	2008	SM521
SHI10	Shutter Island	2010	SCO78
DK008	The Dark Knight	2008	CN345
FISH4	Big Fish	2003	BUR34

Solution 2

(a) Retrieve details of all films that were released in 2010. The output schema should be the same as that of the Film table.

$$\{\, F \in \mathsf{Film} \mid F.\mathsf{yr} = 2010 \,\}$$

$$\sigma_{\mathsf{yr}=2010}(\mathsf{Film})$$

(b) Retrieve details of all actors that are not in their thirties. The output schema should be the same as that of the Film table.

$$\{\, A \in \mathsf{Actor} \mid A.\mathsf{age} < 30 \vee A.\mathsf{age} > 39 \,\}$$
$$\sigma_{(\mathsf{age}<30)\vee(\mathsf{age}>39)}(\mathsf{Actor})$$
$$\text{or} \quad \sigma_{\mathsf{age}<30}(\mathsf{Actor}) \cup \sigma_{\mathsf{age}>39}(\mathsf{Actor})$$

(c) Retrieve the names of all directors.

$$\{\, T \mid \exists D \in \mathsf{Director} \,.\, T.\mathsf{name} = D.\mathsf{name} \,\}$$
$$\pi_{\mathsf{name}}(\mathsf{Director})$$

(d) Retrieve the names of all American directors.

$$\{\, T \mid \exists D \in \mathsf{Director} \,.\, D.\mathsf{nationality} = \text{'American'} \wedge T.\mathsf{name} = D.\mathsf{name} \,\}$$
$$\pi_{\mathsf{name}}(\sigma_{\mathsf{nationality}=\text{'American'}}(\mathsf{Director}))$$

(e) Find out the names of all British actors above the age of 40.

$$\{\, T \mid \exists A \in \mathsf{Actor} \,.\, A.\mathsf{nationality} = \text{'British'} \wedge A.\mathsf{age} > 40 \wedge T.\mathsf{name} = A.\mathsf{name} \,\}$$
$$\pi_{\mathsf{name}}(\sigma_{(\mathsf{nationality}=\text{'British'}\wedge\mathsf{age}>40)}(\mathsf{Actor}))$$

(f) Retrieve the name of each actor together with the titles of the films she/he has performed in.

$$\{\, T \mid \exists A \in \mathsf{Actor}, P \in \mathsf{Performance}, F \in \mathsf{Film} \,.$$
$$A.\mathsf{actorId} = P.\mathsf{actorId} \wedge P.\mathsf{filmId} = F.\mathsf{filmId}$$
$$\wedge\, T.\mathsf{name} = A.\mathsf{name} \wedge T.\mathsf{title} = F.\mathsf{title} \,\}$$
$$\pi_{\mathsf{name},\mathsf{title}}(\mathsf{Actor} \bowtie (\mathsf{Performance} \bowtie \mathsf{Film}))$$

(g) Find out the names of all actors that have played the part of Bruce Wayne (Batman; see also Marshall et al., Physics Special Topics 10(1):2011).

$$\{\, T \mid \exists A \in \mathsf{Actor}, P \in \mathsf{Performance} \,.$$
$$A.\mathsf{actorId} = P.\mathsf{actorId} \wedge P.\mathsf{part} = \text{'Bruce Wayne'} \wedge T.\mathsf{name} = A.\mathsf{name} \,\}$$
$$\pi_{\mathsf{name}}(\mathsf{Actor} \bowtie (\sigma_{\mathsf{part}=\text{'Bruce Wayne'}}(\mathsf{Performance}))$$

(h) Retrieve the names of all actors that have played the part of Bruce Wayne, together with the year the corresponding films were released.

$$\{\, T \mid \exists A \in \mathsf{Actor}, P \in \mathsf{Performance}, F \in \mathsf{Film} \,.$$
$$A.\mathsf{actorId} = P.\mathsf{actorId} \wedge P.\mathsf{filmId} = F.\mathsf{filmId}$$
$$\wedge\, P.\mathsf{part} = \text{'Bruce Wayne'} \wedge T.\mathsf{name} = A.\mathsf{name} \wedge T.\mathsf{yr} = F.\mathsf{yr} \,\}$$
$$\pi_{\mathsf{name},\mathsf{yr}}(\mathsf{Actor} \bowtie (\sigma_{\mathsf{part}=\text{'Bruce Wayne'}}(\mathsf{Performance} \bowtie \mathsf{Film})))$$

(i) Retrieve all actors that appeared in Inception. The output schema should be the same as that of the Actor table.

$$\{\, A \in \mathsf{Actor} \mid \exists P \in \mathsf{Performance}, F \in \mathsf{Film} \,.$$
$$A.\mathsf{actorId} = P.\mathsf{actorId} \wedge P.\mathsf{filmId} = F.\mathsf{filmId} \wedge F.\mathsf{title} = \text{'Inception'} \,\}$$
$$\pi_{\mathsf{actorId},\mathsf{name},\mathsf{nationality},\mathsf{age}}(\mathsf{Actor} \bowtie (\mathsf{Performance} \bowtie (\sigma_{\mathsf{title}=\text{'Inception'}}(\mathsf{Film}))))$$

(j) Find out the names of all actors that have performed in a film directed by Christopher Nolan.

$$\{\, T \mid \exists A \in \mathsf{Actor}, P \in \mathsf{Performance}, F \in \mathsf{Film}, D \in \mathsf{Director}\ .$$
$$D.\mathsf{name} = \text{'Christopher Nolan'} \wedge D.\mathsf{directorId} = F.\mathsf{directorId}$$
$$\wedge\, F.\mathsf{filmId} = P.\mathsf{filmId} \wedge P.\mathsf{actorId} = A.\mathsf{actorId} \wedge A.\mathsf{name} = T.\mathsf{name}\,\}$$
$$\pi_{\mathsf{Actor.name}}(\mathsf{Actor} \bowtie \mathsf{Performance} \bowtie \mathsf{Film} \bowtie (\sigma_{\mathsf{name}=\text{'Christopher Nolan'}}(\mathsf{Director})))$$

(k) Retrieve the titles of all films in which Kate Winslet and Leonardo Di Caprio have both performed.

$$\{\, T \mid \exists A1 \in \mathsf{Actor}, A2 \in \mathsf{Actor}, P1 \in \mathsf{Performance}, P2 \in \mathsf{Performance}, F \in \mathsf{Film}\ .$$
$$A1.\mathsf{actorId} = P1.\mathsf{actorId} \wedge A2.\mathsf{actorId} = P2.\mathsf{actorId}$$
$$\wedge\, A1.\mathsf{name} = \text{'Leonardo DiCaprio'} \wedge A2.\mathsf{name} = \text{'Kate Winslet'}$$
$$\wedge\, P1.\mathsf{filmId} = P2.\mathsf{filmId} \wedge F.\mathsf{filmId} = P1.\mathsf{filmId} \wedge T.\mathsf{title} = F.\mathsf{title}\,\}$$
$$\pi_{\mathsf{title}}(\mathsf{Film} \bowtie (\pi_{\mathsf{filmId}}(\sigma_{\mathsf{name}=\text{'Kate Winslet'}}(\mathsf{Actor}) \bowtie \mathsf{Performance})$$
$$\cap\, \pi_{\mathsf{filmId}}(\sigma_{\mathsf{name}=\text{'Leonardo DiCaprio'}}(\mathsf{Actor}) \bowtie \mathsf{Performance})))$$

(l) Assuming that the actorId and directorId values for actors and directors are consistent across the tables, retrieve details of all actors that have directed a film.

$$\{\, A \in \mathsf{Actor} \mid \exists D \in \mathsf{Director}\ .\ A.\mathsf{actorId} = D.\mathsf{directorId}\,\}$$
$$\pi_{\mathsf{actorId,name,nationality,age}}(\mathsf{Actor} \bowtie_{\mathsf{actorId}=\mathsf{directorId}} \mathsf{Director})$$