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.
- The Examples are there for additional preparation, practice, and revision.
- 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 *. 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) \( \sigma_{\text{class}='Business'}(\text{Seat}) \)

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

<table>
<thead>
<tr>
<th>seatNo</th>
<th>flightNo</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>12D</td>
<td>AF1232</td>
<td>Business</td>
</tr>
<tr>
<td>10A</td>
<td>BA2944</td>
<td>Business</td>
</tr>
<tr>
<td>5D</td>
<td>BA4060</td>
<td>Business</td>
</tr>
</tbody>
</table>

(b) \( \pi_{\text{nationality}}(\text{Booking}) \)

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

<table>
<thead>
<tr>
<th>nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>British</td>
</tr>
<tr>
<td>French</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>airportId</th>
<th>name</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>LHR</td>
<td>Heathrow</td>
<td>London</td>
</tr>
<tr>
<td>LGW</td>
<td>Gatwick</td>
<td>London</td>
</tr>
<tr>
<td>CDG</td>
<td>Charles de Gaulle</td>
<td>Paris</td>
</tr>
<tr>
<td>ORY</td>
<td>Orly</td>
<td>Paris</td>
</tr>
</tbody>
</table>

### Flight

<table>
<thead>
<tr>
<th>flightNo</th>
<th>flightCompany</th>
<th>depAirport</th>
<th>arrAirport</th>
</tr>
</thead>
<tbody>
<tr>
<td>AF1231</td>
<td>Air France</td>
<td>LHR</td>
<td>CDG</td>
</tr>
<tr>
<td>AF1232</td>
<td>Air France</td>
<td>CDG</td>
<td>LHR</td>
</tr>
<tr>
<td>AF1234</td>
<td>Air France</td>
<td>LGW</td>
<td>CDG</td>
</tr>
<tr>
<td>AF1235</td>
<td>Air France</td>
<td>CDG</td>
<td>LGW</td>
</tr>
<tr>
<td>BA2943</td>
<td>British Airways</td>
<td>LGW</td>
<td>ORY</td>
</tr>
<tr>
<td>BA2944</td>
<td>British Airways</td>
<td>ORY</td>
<td>LGW</td>
</tr>
<tr>
<td>BA4059</td>
<td>British Airways</td>
<td>LHR</td>
<td>CDG</td>
</tr>
<tr>
<td>BA4060</td>
<td>British Airways</td>
<td>CDG</td>
<td>LHR</td>
</tr>
</tbody>
</table>

### Booking

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

### Seat

<table>
<thead>
<tr>
<th>seatNo</th>
<th>flightNo</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>12D</td>
<td>AF1232</td>
<td>Business</td>
</tr>
<tr>
<td>30E</td>
<td>AF1232</td>
<td>Economy</td>
</tr>
<tr>
<td>10A</td>
<td>BA2944</td>
<td>Business</td>
</tr>
<tr>
<td>5D</td>
<td>BA4060</td>
<td>Business</td>
</tr>
<tr>
<td>14B</td>
<td>BA4059</td>
<td>Economy</td>
</tr>
<tr>
<td>30D</td>
<td>BA2944</td>
<td>Economy</td>
</tr>
</tbody>
</table>

Figure 2: Table Contents
Retrieves all information about all combinations of bookings by French passengers and possible business class seats. The output table is as follows:

<table>
<thead>
<tr>
<th>ticketNo</th>
<th>name</th>
<th>nationality</th>
<th>flightNo</th>
<th>seatNo</th>
<th>seatNo</th>
<th>flightNo</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS958332</td>
<td>Mathilde Duval</td>
<td>French</td>
<td>BA2944</td>
<td>10A</td>
<td>12D</td>
<td>AF1232</td>
<td>Business</td>
</tr>
<tr>
<td>ABS958332</td>
<td>Mathilde Duval</td>
<td>French</td>
<td>BA2944</td>
<td>10A</td>
<td>10A</td>
<td>BA2944</td>
<td>Business</td>
</tr>
<tr>
<td>ABS958332</td>
<td>Mathilde Duval</td>
<td>French</td>
<td>BA2944</td>
<td>10A</td>
<td>5D</td>
<td>BA4060</td>
<td>Business</td>
</tr>
<tr>
<td>EAG348595</td>
<td>Pierre Fontaine</td>
<td>French</td>
<td>BA2944</td>
<td>30D</td>
<td>12D</td>
<td>AF1232</td>
<td>Business</td>
</tr>
<tr>
<td>EAG348595</td>
<td>Pierre Fontaine</td>
<td>French</td>
<td>BA2944</td>
<td>30D</td>
<td>10A</td>
<td>BA2944</td>
<td>Business</td>
</tr>
<tr>
<td>EAG348595</td>
<td>Pierre Fontaine</td>
<td>French</td>
<td>BA2944</td>
<td>30D</td>
<td>5D</td>
<td>BA4060</td>
<td>Business</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>ticketNo</th>
<th>name</th>
<th>nationality</th>
<th>flightNo</th>
<th>seatNo</th>
<th>class</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS958332</td>
<td>Mathilde Duval</td>
<td>French</td>
<td>BA2944</td>
<td>10A</td>
<td>Business</td>
</tr>
</tbody>
</table>

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) Booking $\bowtie$ Flight

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:

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

(f) $\pi_{\text{name}}(\sigma_{\text{depAirport} = \text{CDG}}(\text{Booking} \bowtie\text{Flight}))$
(Tutor Notes) Retrieves the names of all passengers travelling from Charles de Gaulles airport. The output table is as follows:

<table>
<thead>
<tr>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>John Jones</td>
</tr>
<tr>
<td>Fraser McEwan</td>
</tr>
<tr>
<td>Fiona Stewart</td>
</tr>
</tbody>
</table>

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

(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 $\text{Airport}$ table.

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

\[
\{ A \mid A \in \text{Airport} \land A.\text{city} = 'London' \}.
\]

That can be abbreviated slightly in this equivalent expression:

\[
\{ A \in \text{Airport} \mid A.\text{city} = 'London' \}.
\]

Computing this in relational algebra uses a selection operation:

\[
\sigma_{\text{city}='London'}(\text{Airport}).
\]

(b) Retrieve details of all bookings by British and French passengers. The schema of the output table should be same as that of the $\text{Booking}$ table.
Again, this can be rewritten using an abbreviation:

\{ B \in \text{Booking} \mid B.\text{nationality} = 'British' \lor B.\text{nationality} = 'French' \}

Note that this is just a slight alteration of syntax — the underlying expression is the same.

In contrast, the following two terms show two computationally distinct ways to evaluate this query in relational algebra:

\[ \sigma_{\text{nationality} = 'British' \lor \text{nationality} = 'French'} (\text{Booking}) \]

or

\[ \sigma_{\text{nationality} = 'British'} (\text{Booking}) \cup \sigma_{\text{nationality} = 'French'} (\text{Booking}) \]

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

(c) Retrieve the names of all passengers.

\{ P \mid \exists B \in \text{Booking} . P.\text{name} = B.\text{name} \}

\[ \pi_{\text{name}} (\text{Booking}) \]

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

\{ B \mid \exists F \in \text{Flight} . F.\text{flightCompany} = 'British Airways' \land B.\text{flightNo} = F.\text{flightNo} \land B.\text{depAirport} = F.\text{depAirport} \land B.\text{arrAirport} = F.\text{arrAirport} \}

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

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

\{ P \mid \exists B \in \text{Booking}, F \in \text{Flight} . B.\text{flightNo} = F.\text{flightNo} \land P.\text{name} = B.\text{name} \land P.\text{flightNo} = B.\text{flightNo} \land P.\text{flightCompany} = F.\text{flightCompany} \}

\[ \pi_{\text{name}, \text{flightNo}, \text{flightCompany}} (\text{Booking} \bowtie \text{Flight}) \]

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 ⋆. This indicates that they are optional — you are encouraged to attempt all you can, but they are not a requirement for tutorials.

⋆ (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)

\[
\{ \ F \in \text{Flight} \ | \ \exists A \in \text{Airport} . \ F.\text{depAirport} = A.\text{airportId} \land A.\text{city} = 'London' \}
\]

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

Either of these relational algebra expressions will do. Note the use of an explicit comparison in the equijoin “\(\bowtie_{\text{depAirport}=\text{airportId}}\)”, 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.

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

(Tutor Notes)

\[
\{ \ P \ | \ \exists B \in \text{Booking}, F \in \text{Flight}, A \in \text{Airport} . \\
B.\text{flightNo} = F.\text{flightNo} \land F.\text{depAirport} = A.\text{airportId} \\
\land A.\text{city} = 'London' \land P.\text{ticketNo} = B.\text{ticketNo} \land P.\text{name} = B.\text{name} \}
\]

One possible implementation of this is

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

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

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

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:

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

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.
\{ R \mid \exists F \in \text{Flight}, A1 \in \text{Airport}, A2 \in \text{Airport} . \}

\begin{align*}
F.\text{depAirport} &= A1.\text{airportId} \land F.\text{arrAirport} = A2.\text{airportId} \\
\land A1.\text{city} &= \text{’London’} \land A2.\text{city} = \text{’Paris’} \\
\land R.\text{flightNo} &= F.\text{flightNo} \land R.\text{flightCompany} = F.\text{flightCompany} \}
\end{align*}

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

\[ \pi_{\text{flightNo}, \text{flightCompany}} \left( \sigma_{\text{depCity}=\text{’London’,}\text{arrCity}=\text{’Paris’}} \left( (\text{Flight} \bowtie\bowtie \rho_{\text{airportId}\rightarrow\text{depAirport},\text{city}\rightarrow\text{depCity},\text{name}\rightarrow\text{depAN}}(\text{Airport})) \bowtie\bowtie \rho_{\text{airportId}\rightarrow\text{arrAirport},\text{city}\rightarrow\text{arrCity},\text{name}\rightarrow\text{arrAN}}(\text{Airport})) \right) \right) \]

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.

\[ \pi_{\text{flightNo}, \text{flightCompany}}(\sigma_{\text{city}=\text{’London’}}(\text{Flight} \bowtie\bowtie_{\text{airportId}=\text{depAirport}} \text{Airport})) \cap \pi_{\text{flightNo}, \text{flightCompany}}(\sigma_{\text{city}=\text{’Paris’}}(\text{Flight} \bowtie\bowtie_{\text{airportId}=\text{depAirport}} \text{Airport})) \]

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.

\[ \pi_{\text{flightNo}, \text{flightCompany}}((\sigma_{\text{city}=\text{’London’}}(\text{Airport}) \bowtie\bowtie_{\text{airportId}=\text{depAirport}} \text{Flight}) \bowtie\bowtie_{\text{airportId}=\text{arrAirport}} (\sigma_{\text{city}=\text{’Paris’}}(\text{Airport}))) \]
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) $\sigma_{\text{age}>45}(\text{Actor})$

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

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

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

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

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

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

(h) $\sigma_{\text{yr} \geq 2000}(\text{Film}) \cap \sigma_{\text{yr} \leq 2010}(\text{Film})$

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

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

### Film

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

### Performance

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

### Director

<table>
<thead>
<tr>
<th>directorId</th>
<th>name</th>
<th>nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>CN345</td>
<td>Christopher Nolan</td>
<td>British</td>
</tr>
<tr>
<td>JC212</td>
<td>James Cameron</td>
<td>Canadian</td>
</tr>
<tr>
<td>SM521</td>
<td>Sam Mendes</td>
<td>British</td>
</tr>
<tr>
<td>SCO78</td>
<td>Martin Scorsese</td>
<td>American</td>
</tr>
<tr>
<td>BUR34</td>
<td>Tim Burton</td>
<td>American</td>
</tr>
</tbody>
</table>

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) $\sigma_{\text{age}>45}(\text{Actor})$

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

<table>
<thead>
<tr>
<th>actorId</th>
<th>name</th>
<th>nationality</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>MKE12</td>
<td>Michael Keaton</td>
<td>American</td>
<td>63</td>
</tr>
<tr>
<td>HBC54</td>
<td>Helena Bonham Carter</td>
<td>British</td>
<td>48</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
</tr>
<tr>
<td>Titanic</td>
</tr>
<tr>
<td>Revolutionary Road</td>
</tr>
<tr>
<td>Skyfall</td>
</tr>
<tr>
<td>Shutter Island</td>
</tr>
<tr>
<td>The Dark Knight</td>
</tr>
<tr>
<td>The Dark Knight Rises</td>
</tr>
<tr>
<td>Batman Returns</td>
</tr>
<tr>
<td>Big Fish</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Titanic</td>
</tr>
<tr>
<td>Batman Returns</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>filmId</th>
<th>title</th>
<th>yr</th>
<th>directorId</th>
<th>directorId</th>
<th>name</th>
<th>nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKF12</td>
<td>Skyrim</td>
<td>2012</td>
<td>SM521</td>
<td>CN345</td>
<td>Christopher Nolan</td>
<td>British</td>
</tr>
<tr>
<td>SKF12</td>
<td>Skyrim</td>
<td>2012</td>
<td>SM521</td>
<td>JC212</td>
<td>James Cameron</td>
<td>Canadian</td>
</tr>
<tr>
<td>SKF12</td>
<td>Skyrim</td>
<td>2012</td>
<td>SM521</td>
<td>SM521</td>
<td>Sam Mendes</td>
<td>British</td>
</tr>
<tr>
<td>DKR12</td>
<td>The Dark Knight Rises</td>
<td>2012</td>
<td>CN345</td>
<td>CN345</td>
<td>Christopher Nolan</td>
<td>British</td>
</tr>
<tr>
<td>DKR12</td>
<td>The Dark Knight Rises</td>
<td>2012</td>
<td>CN345</td>
<td>JC212</td>
<td>James Cameron</td>
<td>Canadian</td>
</tr>
<tr>
<td>DKR12</td>
<td>The Dark Knight Rises</td>
<td>2012</td>
<td>CN345</td>
<td>SM521</td>
<td>Sam Mendes</td>
<td>British</td>
</tr>
</tbody>
</table>
(e) \( \sigma_{yr=2012}(\text{Film}) \bowtie \sigma_{\text{nationality} \neq \text{American}}(\text{Director}) \)

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:

<table>
<thead>
<tr>
<th>filmId</th>
<th>title</th>
<th>yr</th>
<th>directorId</th>
<th>name</th>
<th>nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>SKF12</td>
<td>Skyfall</td>
<td>2012</td>
<td>SM521</td>
<td>Sam Mendes</td>
<td>British</td>
</tr>
<tr>
<td>DKR12</td>
<td>The Dark Knight Rises</td>
<td>2012</td>
<td>CN345</td>
<td>Christopher Nolan</td>
<td>British</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inception</td>
</tr>
<tr>
<td>Revolutionary Road</td>
</tr>
<tr>
<td>Skyfall</td>
</tr>
<tr>
<td>The Dark Knight</td>
</tr>
<tr>
<td>The Dark Knight Rises</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>filmId</th>
<th>title</th>
<th>yr</th>
<th>directorId</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIT97</td>
<td>Titanic</td>
<td>1997</td>
<td>JC212</td>
</tr>
<tr>
<td>SKF12</td>
<td>Skyfall</td>
<td>2012</td>
<td>SM521</td>
</tr>
<tr>
<td>DKR12</td>
<td>The Dark Knight Rises</td>
<td>2012</td>
<td>CN345</td>
</tr>
<tr>
<td>BAT92</td>
<td>Batman Returns</td>
<td>1992</td>
<td>BUR34</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>filmId</th>
<th>title</th>
<th>yr</th>
<th>directorId</th>
</tr>
</thead>
<tbody>
<tr>
<td>INCI0</td>
<td>Inception</td>
<td>2010</td>
<td>CN345</td>
</tr>
<tr>
<td>RR008</td>
<td>Revolutionary Road</td>
<td>2008</td>
<td>SM521</td>
</tr>
<tr>
<td>SHI10</td>
<td>Shutter Island</td>
<td>2010</td>
<td>SCO78</td>
</tr>
<tr>
<td>DK008</td>
<td>The Dark Knight</td>
<td>2008</td>
<td>CN345</td>
</tr>
<tr>
<td>FISH4</td>
<td>Big Fish</td>
<td>2003</td>
<td>BUR34</td>
</tr>
</tbody>
</table>

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.

\[
\{ F \in \text{Film} \mid F.yr = 2010 \} \\
\sigma_{yr=2010}(\text{Film})
\]
(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.

\[
\{ A \in \text{Actor} \mid A.\text{age} < 30 \lor A.\text{age} > 39 \}
\]

\[
\sigma_{(\text{age} < 30) \lor (\text{age} > 39)}(\text{Actor})
\]

or

\[
\sigma_{\text{age} < 30}(\text{Actor}) \cup \sigma_{\text{age} > 39}(\text{Actor})
\]

(c) Retrieve the names of all directors.

\[
\{ T \mid \exists D \in \text{Director} . T.\text{name} = D.\text{name} \}
\]

\[
\pi_{\text{name}}(\text{Director})
\]

(d) Retrieve the names of all American directors.

\[
\{ T \mid \exists D \in \text{Director} . D.\text{nationality} = '\text{American}' \land T.\text{name} = D.\text{name} \}
\]

\[
\pi_{\text{name}}(\sigma_{\text{nationality}= '\text{American}' } (\text{Director}))
\]

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

\[
\{ T \mid \exists A \in \text{Actor} . A.\text{nationality} = '\text{British}' \land A.\text{age} > 40 \land T.\text{name} = A.\text{name} \}
\]

\[
\pi_{\text{name}}(\sigma_{\text{nationality}= '\text{British}' \land \text{age} > 40 } (\text{Actor}))
\]

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

\[
\{ T \mid \exists A \in \text{Actor}, P \in \text{Performance}, F \in \text{Film} .
\]

\[
A.\text{actorId} = P.\text{actorId} \land P.\text{filmId} = F.\text{filmId} \\
\land T.\text{name} = A.\text{name} \land T.\text{title} = F.\text{title} \}
\]

\[
\pi_{\text{name, title}}(\text{Actor} \bowtie (\text{Performance} \bowtie \text{Film}))
\]

(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).

\[
\{ T \mid \exists A \in \text{Actor}, P \in \text{Performance} .
\]

\[
A.\text{actorId} = P.\text{actorId} \land P.\text{part} = '\text{Bruce Wayne}' \land T.\text{name} = A.\text{name} \}
\]

\[
\pi_{\text{name}}(\text{Actor} \bowtie (\sigma_{\text{part}= '\text{Bruce Wayne}' } (\text{Performance})))
\]

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

\[
\{ T \mid \exists A \in \text{Actor}, P \in \text{Performance}, F \in \text{Film} .
\]

\[
A.\text{actorId} = P.\text{actorId} \land P.\text{filmId} = F.\text{filmId} \\
\land P.\text{part} = '\text{Bruce Wayne}' \land T.\text{name} = A.\text{name} \land T.\text{yr} = F.\text{yr} \}
\]

\[
\pi_{\text{name, yr}}(\text{Actor} \bowtie (\sigma_{\text{part}= '\text{Bruce Wayne}' } (\text{Performance} \bowtie \text{Film})))
\]

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

\[
\{ A \in \text{Actor} \mid \exists P \in \text{Performance}, F \in \text{Film} .
\]

\[
A.\text{actorId} = P.\text{actorId} \land P.\text{filmId} = F.\text{filmId} \land F.\text{title} = '\text{Inception}' \}
\]

\[
\pi_{\text{actorId, name, nationality, age}}(\text{Actor} \bowtie (\text{Performance} \bowtie (\sigma_{\text{title}= '\text{Inception}' } (\text{Film}))))
\]
(j) Find out the names of all actors that have performed in a film directed by Christopher Nolan.

\[
\{ T \mid \exists A \in \text{Actor}, P \in \text{Performance}, F \in \text{Film}, D \in \text{Director} .
\]

\[
D.\text{name} = 'Christopher Nolan' \land D.\text{directorId} = F.\text{directorId}
\land F.\text{filmId} = P.\text{filmId} \land P.\text{actorId} = A.\text{actorId} \land A.\text{name} = T.\text{name}
\}
\]

\[
\pi_{\text{Actor.name}} (\text{Actor} \bowtie \text{Performance} \bowtie \text{Film} \bowtie (\sigma_{\text{name}='Christopher Nolan'} (\text{Director})))
\]

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

\[
\{ T \mid \exists A1 \in \text{Actor}, A2 \in \text{Actor}, P1 \in \text{Performance}, P2 \in \text{Performance}, F \in \text{Film} .
\]

\[
A1.\text{actorId} = P1.\text{actorId} \land A2.\text{actorId} = P2.\text{actorId}
\land A1.\text{name} = 'Leonardo DiCaprio' \land A2.\text{name} = 'Kate Winslet'
\land P1.\text{filmId} = P2.\text{filmId} \land F.\text{filmId} = P1.\text{filmId} \land T.\text{title} = F.\text{title}
\}
\]

\[
\pi_{\text{title}} (\text{Film} \bowtie (\pi_{\text{filmId}} (\sigma_{\text{name}='Kate Winslet'} (\text{Actor}) \bowtie \text{Performance})
\land \pi_{\text{filmId}} (\sigma_{\text{name}='Leonardo DiCaprio'} (\text{Actor}) \bowtie \text{Performance})))
\]

(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.

\[
\{ A \in \text{Actor} \mid \exists D \in \text{Director} . A.\text{actorId} = D.\text{directorId} \}
\]

\[
\pi_{\text{actorId},\text{name},\text{nationality},\text{age}} (\text{Actor} \bowtie \text{actorId}=\text{directorId} \text{ Director})
\]