Most questions in this exam will be about one and the same domain: flights. This domain makes heavy use of databases, familiar from web applications. The exam will go through different aspects having to do with airlines, flight connections, and bookings.

Table 1. We start with a table listing some flights. It is an example of a relation that links together flight codes, airlines, airports, and aircraft. The airlines and airports are real ones, whereas the codes are fake.

flight code	airline	prime flight	operating airline	departure city	departure airport	destination city	destination airport	aircraft type	seats
SK111	SAS	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
LH555	Lufthansa	SK111	SAS	Gothenburg	GOT	Frankfurt	FRA	B737	140
AF111	Air France	AF111	Air France	Gothenburg	GOT	Paris	CDG	A320	170
LH111	Lufthansa	LH111	Lufthansa	Frankfurt	FRA	Paris	CDG	A321	200
LH222	Lufthansa	LH222	Lufthansa	Frankfurt	FRA	Malta	MLA	A320	170
AF222	Air France	AF222	Air France	Paris	ORY	Malta	MLA	A320	170
AB222	Air Berlin	AB222	Air Berlin	Frankfurt	FRA	Munich	MUC	A320	170
KM111	Air Malta	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
LH333	Lufthansa	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
SK222	SAS	KM111	Air Malta	Munich	MUC	Malta	MLA	A319	140
AF333	Air France	AF333	Air France	Paris	CDG	Frankfurt	FRA	A320	170

We assume the following (slightly simplified) conventions for this domain:

- the “flight code” attribute determines all other attributes on a row
- the “prime flight” is the flight code used by the airline operating the flight; the “flight code” in the first column can thus belong to another airline that has a code sharing agreement with the operating airline
- the “prime flight” appears in the table as a “flight code” as well, having itself as prime flight
- each airport has a unique code
- every aircraft of the same type has the same number of seats

(It is a common practice that one and the same flight can be booked using different airlines. Each airline uses a different "flight code", but the passengers end up in the same plane. The code used by the actual operating airline is called the "prime flight" code. For example, whether you book flight LH333 with Lufthansa or flight SK222 with SAS, you end up in the plane of Air Malta flight KM111.)

Question 5a (2p)
Create a view that lists booking references, passengers, flight codes, and departure and destination cities.

Question 5b (7p)
Create a trigger on the view defined in (5a). This trigger takes care of booking a new passenger to a flight. It is fired by an insertion of a passenger and a flight code to the view, for instance, “book Annie Adams for AF666”. Its effect should be the following:

- if the number of free seats on AF666 is positive, decrement it by one; the booking is successful
- if there are no free seats, the booking fails
- if the booking is successful, add Annie Adams and AF666 to Bookings, with the price given in AvailableFlights when booking her; also add a booking reference which is the

Question 6a (4p)
Bookings are typically made concurrently by many users. Assuming that the booking steps in (5b) are *not* an atomic transaction, show sequences of events (i.e. queries and updates from different customers) where

- a customer is suggested a flight but it turns out to be full
- a customer is told that a flight is full although it has seats
- a customer has to pay overprice i.e. a price that applies to customers booking later

For each sequence, also indicate which isolation level would prevent the undesired outcome from happening.

Exam (20 March 2015)

Question 3. 10 p

A university uses a database to manage information about applications to its Master's programmes. This database has the following relations:

Programmes(code, name, department, numPlaces)
Applicants(name, address, appNumber)
AppliesFor(applicant, programme, choiceNumber, meritScore, status)

Each programme is identified by its code. The programme name, the name of the department that arranges the programme, and the number of available places for students in the programme this year are stored.

Each applicant's name, address and (unique) applicant number are stored.

Each applicant can apply to up to four programmes, and attribute *choiceNumber* records the applicant's prioritisation of the programmes (e.g. someone might apply to two programmes: Computer Science as choice 1 and Physics as choice 2). Possible values for this attribute are 1, 2, 3 and 4. The choice numbers for a particular applicant must be unique (e.g. an applicant cannot apply to two programmes giving both as choice number 1).

When applications to a programme are processed by the university, the applications are ranked, and each application is assigned a merit score between 1 and 1000 (e.g. the top three applications for a programme might be assigned merit scores 920, 890 and 870). No two applicants can get the same merit score for the same programme (you can assume that no programme has more than 1000 applicants). A default merit score of '0' indicates that the application has not been evaluated yet.

Attribute *status* has the value 'unprocessed', 'offered' (a place on the programme has been offered to the applicant), 'accepted' (the applicant received an offer and accepted it), 'declined' (the applicant received an offer but declined it), 'offer withdrawn' (an offer was made, but the applicant did not respond before the deadline for accepting the offer), or 'rejected' (the university decided not to offer a place on this programme to this applicant). The default status is 'unprocessed'.

Question 6. 5 p

Suppose a system for handling applications to a university's Master's programmes (as described in Question 3) has a transaction, T, with the following steps:

T_1: get a programme code from the user (store this in p), and find the unprocessed application to this programme with the highest merit score
T_2: count how many applicants have accepted a place on programme p
T_3: offer the applicant identified in step T_1 a place on programme p, and set the *status* attribute in the relevant row of relation *AppliesFor* to 'offered'
T_4: get response from the applicant ('accepted' or 'declined') and set the *status* attribute in the relevant row of relation *AppliesFor* accordingly
T_5: count how many applicants have accepted a place on programme p

a) In database transactions, what are *phantoms*? Refer to transaction T in your answer. (2p)

b) Discuss how the results of steps T_2 and T_5 could differ if transaction T is run with different isolation levels.
Discuss what isolation level(s) would be most appropriate for running transaction T. (3p)