Chapter 3 The Relational Model

Review Questions

3.1 *Discuss each of the following concepts in the context of the relational data model:*

(a) Relation
(b) Attribute
(c) Domain
(d) Tuple
(e) Intension and Extension
(f) Degree and Cardinality.

Each term defined in Section 3.2.1.

3.2 *Describe the relationship between mathematical relations and relations in the relational data model?*

Let $D_1, D_2, \ldots, D_n$ be n sets. Their Cartesian product is defined as:

$$D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n\}$$

Any set of n-tuples from this Cartesian product is a relation on the n sets. Now let $A_1, A_2, \ldots, A_n$ be attributes with domains $D_1, D_2, \ldots, D_n$. Then the set $\{A_1{:}D_1, A_2{:}D_2, \ldots, A_n{:}D_n\}$ is a relation schema. A relation R defined by a relation schema S is a set of mappings from the attribute names to their corresponding domains. Thus, relation R is a set of n-tuples:

$(A_1{:}d_1, A_2{:}d_2, \ldots, A_n{:}d_n)$ such that $d_1 \in D_1, d_2 \in D_2, \ldots, d_n \in D_n$

Each element in the n-tuple consists of an attribute and a value for that attribute.

Discussed fully in Sections 3.2.2 and 3.2.3.

3.3 *Describe the differences between a relation and a relation schema. What is a relational database schema?*

A relation schema is a named relation defined by a set of attribute and domain name pairs. A relational database schema is a set of relation schemas, each with a distinct name. Discussed in Section 3.2.3.

3.4 *Discuss the properties of a relation.*

A relation has the following properties:

- has a name that is distinct from all other relation names in the relational schema;
- each cell contains exactly one atomic (single) value;
- each attribute has a distinct name;
- the values of an attribute are all from the same domain;
- each tuple is distinct; there are no duplicate tuples;
- the order of attributes has no significance;
- the order of tuples has no significance, theoretically. (However, in practice, the order may affect the efficiency of accessing tuples.)

Discussed fully in Section 3.2.4.

3.5 *Discuss the differences between the candidate keys and the primary key of a relation. Explain what is meant by a foreign key. How do foreign keys of relations relate to candidate keys? Give examples to illustrate your answer.*

The primary key is the candidate key that is selected to identify tuples uniquely within a relation. A foreign key is an attribute or set of attributes within one relation that matches the candidate key of some (possibly the same) relation. Discussed in Section 3.2.5.

3.6 *Define the two principal integrity rules for the relational model. Discuss why it is desirable to enforce these rules.*

Two rules are Entity Integrity (Section 3.3.2) and Referential Integrity (Section 3.3.3).

3.7 *What is a view? Discuss the difference between a view and a base relation.*

View is the dynamic result of one or more relational operations operating on the base relations to produce another relation. Base relation exists as a set of data in the database. A view does not contain any data, rather a view is defined as a query on one or more base relations and a query on the view is translated into a query on the associated base relations.

Exercises

The following tables form part of a database held in a relational DBMS:-

Hotel	(hotelNo, hotelName, city)
Room	(roomNo, hotelNo, type, price)
Booking	(hotelNo, guestNo, dateFrom, dateTo, roomNo)
Guest	(guestNo, guestName, guestAddress)

where Hotel contains hotel details and hotelNo is the primary key;
Room contains room details for each hotel and (roomNo, hotelNo) forms the primary key;
Booking contains details of the bookings and (hotelNo, guestNo, dateFrom) forms the primary key;
and Guest contains guest details and guestNo is the primary key.

3.8 *Identify the foreign keys in this schema. Explain how the entity and referential integrity rules apply to these relations.*

For each relation, the primary key must not contain any nulls.

Room is related to Hotel through the attribute hotelNo. Therefore, the hotelNo in Room should either be null or contain the number of an existing hotel in the Hotel relation. In this case study, it would probably be unacceptable to have a hotelNo in Room with a null value.

Booking is related to Hotel through the attribute hotelNo. Therefore, the hotelNo in Booking should either be null or contain the number of an existing hotel in the Hotel relation. However, because hotelNo is also part of the primary key, a null value for this attribute would be unacceptable. Similarly for guestNo. Booking is also related to Room through the attribute roomNo.

3.9 *Produce some sample tables for these relations that observe the relational integrity rules. Suggest some enterprise constraints that would be appropriate for this schema.*

Student should provide some sample tables, observing entity and referential integrity. In particular, ensure the uniqueness for the composite primary keys of the Room and Booking tables.

Some enterprise constraints may be:

- There can be no two bookings for the same room in the same hotel on the same day.
- For the Booking relation, dateFrom must be before dateTo.

- Room price must be greater than 0 and less than £200.

3.10 *Analyze the RDBMSs that you are currently using. Determine the support the system provides for primary keys, alternate keys, foreign keys, relational integrity and views.*

This is a small student project, the result of which is dependent on the system analyzed.

3.11 *Implement the above schema in one of the RDBMSs you currently use. Implement, where possible, the primary, alternate, and foreign keys, and appropriate relational integrity constraints.*

This is a small student project, the result of which is dependent on the RDBMS used. Ensure that keys have been implemented, and that relationships have been implemented if the RDBMS supports this.

Chapter 4 Relational Algebra and Relational Calculus

Review Questions

4.1 *What is the difference between a procedural and non-procedural language? How would you classify the relational algebra and relational calculus?*

Procedural language: a language that allows user to tell the system what data is needed and exactly *how* to retrieve the data.
Non-procedural language: a language that allows user to state *what* data is needed rather than how it is to be retrieved.

Informally, we may describe the relational algebra as a (high-level) procedural language: it can be used to tell the DBMS how to build a new relation from one or more relations in the database. Again, informally, we may describe the relational calculus as a non-procedural language: it can be used to formulate the definition of a relation in terms of one or more database relations.

4.2 *Explain the following terms:*

- *relationally complete;*

  A language that can be used to produce any relation that can be derived using the relational calculus is said to be **relationally complete**.

- *closure of relational operations.*

  The relational algebra is a theoretical language with operations that work on one or more relations to define another relation without changing the original relation(s). Thus, both the operands and the results are relations, and so the output from one operation can become the input to another operation. This allows expressions to be nested in the relational algebra, just as we can nest arithmetic operations. This property is called **closure**: relations are closed under the algebra, just as numbers are closed under arithmetic operations.

4.3 *Define the five basic relational algebra operations. Define the Join, Intersection, and Division operations in terms of these five basic operations.*

Five basic operations are:

- Selection and Projection (Unary)
- Cartesian Product, Union and Set Difference (Binary).

There is also the Join, Intersection, and Division operations:

- Can rewrite θ-Join in terms of the basic selection and Cartesian product operations:

  $R \bowtie_F S = \sigma_F (R \times S)$

- Can express the intersection operation in terms of the set difference operation:

  $R \cap S = R - (R - S)$

- Can express the division operation in terms of the basic operations:

  $T_1 = \Pi_C(R)$
  $T_2 = \Pi_C((S \times T_1) - R)$
  $T = T_1 - T_2$

4.4 *Discuss the differences between the five Join operations: Theta join, Equijoin, Natural join, Outer join, and Semijoin. Give examples to illustrate your answer.*

Theta join	R $\exists_F$ S	Produces a relation that contains tuples satisfying the predicate F from the Cartesian product of R and S.
Equijoin	R $\exists_F$ S	Produces a relation that contains tuples satisfying the predicate F (which only contains equality comparisons) from the Cartesian product of R and S.
Natural join	R $\exists$ S	An Equijoin of the two relations R and S over all common attributes x. One occurrence of each common attribute is eliminated.
(Left) Outer join	R $\exists$ S	A join in which tuples from R that do not have matching values in the common attributes of S are also included in the result relation.
Semijoin	R $\exists_F$ S	Produces a relation that contains the tuples of R that participate in the join of R with S.

4.5 *Compare and contrast the tuple relational calculus with domain relational calculus. In particular, discuss the distinction between tuple and domain variables.*

In the tuple relational calculus, we use variables that range over tuples in a relation. In the domain relational calculus, we also use variables but in this case the variables take their values from *domains* of attributes rather than tuples of relations.

4.6 *Define the structure of a (well-formed) formula in both the tuple relational calculus and domain relational calculus.*

Tuple relational calculus

A (well-formed) formula is made out of one or more *atoms*, where an atom has one of the following forms:

- $R(S_i)$, where S_i is a tuple variable and R is a relation.
- $S_i.a_1 \; \theta \, S_j.a_2$, where S_i and S_j are tuple variables, a_1 is an attribute of the relation over which S_i ranges, a_2 is an attribute of the relation over which S_j ranges, and θ is one of the comparison operators (<, ≤ , >, ≥ , =, ≠); the attributes a_1 and a_2 must have domains whose members can be compared by θ.
- $S_i.a_1 \; \theta \, c$, where S_i is a tuple variable, a_1 is an attribute of the relation over which S_i ranges, c is a constant from the domain of attribute a_1, and θ is one of the comparison operators.

We recursively build up formulae from atoms using the following rules:

- an atom is a formula;
- if F_1 and F_2 are formulae, so are their conjunction $F_1 \wedge F_2$, their disjunction $F_1 \vee F_2$, and the negation $\sim F_1$;
- if F is a formula with free variable X, then $(\exists X)(F)$ and $(\forall X)(F)$ are also formulae.

Domain relational calculus

A (well-formed) formula is made out of one or more *atoms*, where an atom has one of the following forms:

- $R(d_1, d_2, \ldots, d_n)$, where R is a relation of degree n and each d_i is a domain variable.
- $d_i \; \theta \, d_j$, where d_i and d_j are domain variables and θ is one of the comparison operators (<, ≤ , >, ≥ , =, ≠); the domains d_i and d_j must have members that can be compared by θ.
- $d_i \; \theta \, c$, where d_i is a domain variable, c is a constant from the domain of d_i, and θ is one of the comparison operators.

4.7 *Explain how a relational calculus expression can be unsafe? Illustrate your answer with an example. Discuss how to ensure that a relational calculus expression is safe.*

See end of Section 4.2.1.

Exercises

For the following exercises, use the Hotel schema defined at the start of the Exercises at the end of Chapter 3.

4.8 *Describe the relations that would be produced by the following relational algebra operations:*

a) $\Pi_{hotelNo} (\sigma_{price > 50} (Room))$

This will produce a relation with a single attribute (hotelNo) giving the number of those hotels with a room price greater than £50.

b) $\sigma_{Hotel.hotelNo = Room.hotelNo}(Hotel \times Room)$

This will produce a join of the Hotel and Room relations containing all the attributes of both Hotel and Room (there will be two copies of the hotelNo attribute). Essentially this will produce a relation containing all rooms at all hotels.

c) $\Pi_{hotelName} (Hotel \; 3_{Hotel.hotelNo = Room.hotelNo} (\sigma_{price > 50} (Room)))$

This will produce a join of Hotel and those tuples of Room with a price greater than £50. Essentially this will produce a relation containing all hotel names with a room price above £50.

d) $Guest \; 5 \; (\sigma_{dateTo \geq '1\text{-}Jan\text{-}2002'} (Booking))$

This will produce a (left outer) join of Guest and those tuples of Booking with an end date (dateTo) greater than or equal to 1-Jan-2002. All guests who don't have a booking with such a date will still be included in the join. Essentially this will produce a relation containing all guests and show the details of any bookings they have beyond 1-Jan-2002.

e) $Hotel \; 1_{Hotel.hotelNo = Room.hotelNo} (\sigma_{price > 50} (Room)))$

This will produce a (semi) join of Hotel and those tuples of Room with a price greater than £50. Only those Hotel attributes will be listed. Essentially this will produce a relation containing all the details of all hotels with a room price above £50.

f) $\Pi_{guestName, hotelNo} (Booking \; 3_{Booking.guestNo = Guest.guestNo} Guest) \div \Pi_{hotelNo} (\sigma_{city = 'London'}(Hotel))$

This will produce a relation containing the names of all guest who have booked all hotels in London.

4.9 *Provide the equivalent tuple relational calculus and domain relational calculus expressions for each of the relational algebra queries given in Exercise 4.8.*

a) $\Pi_{hotelNo} (\sigma_{price > 50} (Room))$

TRC: $\{R.hotelNo \mid Room(R) \wedge R.price > 50\}$

DRC: $\{hotelNo \mid (\exists rNo, typ, prce) (Room (rNo, hotelNo, typ, prce) \wedge prce > 50)\}$

b) $\sigma_{Hotel.hotelNo = Room.hotelNo}(Hotel \times Room)$

TRC: $\{H, R \mid Hotel(H) \wedge (\exists R)\ (Room(R) \wedge (H.hotelNo = R.hotelNo))\}$

DRC: $\{hNo, hName, cty, rNo, hNo1, typ, prce \mid (Hotel(hNo, hName, cty) \wedge Room(rNo, hNo1, typ, prce) \wedge (hNo = hNo1))\}$

c) $\Pi_{hotelName}\ (Hotel\ \mathbin{3}_{Hotel.hotelNo = Room.hotelNo}\ (\sigma_{price > 50}\ (Room))\ )$

TRC: $\{H.hotelName \mid Hotel(H) \wedge (\exists R)\ (Room(R) \wedge (H.hotelNo = R.hotelNo) \wedge (R.price > 50))\}$

DRC: $\{hotelName \mid (\exists hNo, cty, rNo, hNo1, typ, prce)\ (Hotel(hNo, hotelName, cty) \wedge Room(rNo, hNo1, typ, prce) \wedge (hNo = hNo1) \wedge (prce > 50))\}$

d) $Guest\ \mathbin{5}\ (\sigma_{dateTo \geq \text{'1-Jan-2002'}}\ (Booking))$

TRC: $\{G.guestNo, G.guestName, G.guestAddress, B.hotelNo, B.dateFrom, B.dateTo, B.roomNo \mid Guest(G) \vee (\exists B)\ (Booking(B) \wedge (G.guestNo = B.guestNo) \wedge (B.dateTo > \text{'1-Jan-2002'}))\}$

DRC: $\{guestNo, guestName, guestAddress, hotelNo, dateFrom, dateTo, roomNo \mid (\exists gNo1)\ (Guest(guestNo, guestName, guestAddress) \vee (Booking(hotelNo, gNo1, dateFrom, dateTo, roomNo) \wedge (guestNo = gNo1) \wedge (dateTo \geq \text{'1-Jan-2002'})))\}$

e) $Hotel\ \mathbin{1}_{Hotel.hotelNo = Room.hotelNo}\ (\sigma_{price > 50}\ (Room))\ )$

TRC: $\{H.hotelNo, H.hotelName, H.city \mid Hotel(H) \wedge (\exists R)\ (Room(R) \wedge (H.hotelNo = R.hotelNo) \wedge (R.price > 50))\}$

DRC: $\{hotelNo, hotelName, city \mid (\exists rNo, hNo1, typ, prce)\ (Hotel(hotelNo, hotelName, city) \wedge Room(rNo, hNo1, typ, prce) \wedge (hotelNo = hNo1) \wedge (prce > 50))\}$

f) $\Pi_{guestName,\ hotelNo}$ (Booking $3_{\ Booking.guestNo\ =\ Guest.guestNo}$ Guest) ÷

$\Pi_{hotelNo}$ ($\sigma_{city\ =\ 'London'}$(Hotel))

TRC: {G.guestName | Guest(G) ∧(~ (∃H) (Hotel(H) ∧

(H.city = 'London') ∧ (~(∃B) (Booking(B) ∧

G.guestNo = B.guestNo ∧ H.hotelNo = B.hotelNo))))}

DRC: {guestName | (∃gNo, gName, gAddress, hNo, gNo1, dFrom, dTo, rNo,

hName, cty, hNo1, typ, prce) (~(Hotel(hNo, hName, cty) ∧

(cty = 'London') ∧ Guest(gNo, gName, gAddress) ∧

Booking(hNo1, gNo1, dFrom, dTo, rNo) ∧

(gNo = gNo1) ∧ (hNo = hNo1)))}

4.10 *Describe the relations that would be produced by the following tuple relational calculus expressions:*

(a) {H.hotelName | Hotel(H) ∧ H.city = 'London'}

This will produce a relation containing the names of all hotels in London.

(b) {H.hotelName | Hotel(H) ∧ (∃R) (Room(R) ∧ H.hotelNo = R.hotelNo ∧ R.price > 50)}

This will produce a relation containing the names of all hotels that have a room price above £50.

(c) {H.hotelName | Hotel(H) ∧ (∃B) (∃G) (Booking(B) ∧ Guest(G) ∧ H.hotelNo = B.hotelNo ∧

B.guestNo = G.guestNo ∧ G.guestName = 'John Smith')}

This will produce a relation containing the names of all hotels that have a booking for a guest called John Smith.

(d) {H.hotelName, G.guestName, B1.dateFrom, B2.dateFrom | Hotel(H) ∧ Guest(G) ∧

Booking(B1) ∧ Booking(B2) ∧ H.hotelNo = B1.hotelNo ∧

G.guestNo = B1.guestNo ∧ B2.hotelNo = B1.hotelNo ∧

B2.guestNo = B1.guestNo ∧ B2.dateFrom ≠ B1.dateFrom}

This will produce a relation containing the names of guests who have more than one booking at the same hotel, along with the hotel number and the dates of the bookings.

4.11 *Provide the equivalent domain relational calculus and relational algebra expressions for each of the tuple relational calculus expressions given in Exercise 4.10.*

(a) {H.hotelName | Hotel(H) ∧ H.city = 'London'}

DRC: {hotelName | (∃hNo, cty) (Hotel(hNo, hotelName, cty) ∧ cty = 'London')}

RA: $\Pi_{hotelName}$ ($\sigma_{city = 'London'}$ (Hotel))

(b) {H.hotelName | Hotel(H) ∧ (∃R) (Room(R) ∧ H.hotelNo = R.hotelNo ∧ R.price > 50)}

DRC: {hotelName | (∃hNo, cty, rNo, hNo1, typ, prce) (Hotel(hNo, hotelName, cty) ∧
Room(rNo, hNo1, typ, prce) ∧ (hNo = hNo1) ∧ (prce > 50)) }

RA: $\Pi_{hotelName}$ (Hotel 3 $_{Hotel.hotelNo = Room.hotelNo}$ ($\sigma_{price > 50}$ (Room)))

(c) {H.hotelName | Hotel(H) ∧ (∃B) (∃G) (Booking(B) ∧ Guest(G) ∧ H.hotelNo = B.hotelNo ∧
B.guestNo = G.guestNo ∧ G.guestName = 'John Smith')}

DRC: {hotelName | (∃hNo, cty, gNo, gName, gAddress, hNo1, gNo1, dFrom, dTo,
rNo) (Hotel(hNo, hotelName, cty) ∧
Guest(gNo, gName, gAddress) ∧
Booking(hNo1, gNo1, dFrom, dTo, rNo) ∧ (gNo = gNo1) ∧
(hNo = hNo1) ∧ (gName = 'John Smith'))}

RA: $\Pi_{hotelName}$ ($\sigma_{guestName = 'John Smith'}$ (Guest) 3 $_{Guest.guestNo = guestNo}$ (
Booking 3 $_{.Booking.hotelNo = Hotel.hotelNo}$ Hotel))

(d) {H.hotelName, G.guestName, B1.dateFrom, B2.dateFrom | Hotel(H) ∧ Guest(G) ∧
Booking(B1) ∧ Booking(B2) ∧ H.hotelNo = B1.hotelNo ∧
G.guestNo = B1.guestNo ∧ B2.hotelNo = B1.hotelNo ∧
B2.guestNo = B1.guestNo ∧ B2.dateFrom ≠ B1.dateFrom}

DRC: {hotelName, guestName, dateFrom1, dateFrom2 | (∃hNo, cty,
gNo, gAddress, hNo1, gNo1, dTo1, rNo1, hNo2, gNo2, dTo2, rNo2)
(Hotel(hNo, hotelName, cty) ∧
Guest(gNo, guestName, gAddress) ∧
Booking(hNo1, gNo1, dateFrom1, dTo1, rNo1) ∧

Booking(hNo2, gNo2, dateFrom2, dTo2, rNo2) ∧

(hNo = hNo1) ∧ (gNo = gNo1) ∧ (hNo2 = hNo1) ∧ (gNo2 = gNo1) ∧

(dateFrom1 ≠ dateFrom2))}

RA: Booking2(hotelNo, guestNo, dateFrom2, dateTo2, roomNo2) ←

$\Pi_{hotelNo, guestNo, dateFrom, dateTo, roomNo}$ (Booking)

$\Pi_{hotelName, guestName, dateFrom, dateFrom2}$ (Hotel 3 $_{Hotel.hotelNo = hotelNo}$

(Guest 3 $_{Guest.guestNo = guestNo.}$ (Booking 3 $_{Booking.hotelNo = Booking2.hotelNo}$ ∧

$_{Booking.guestNo = Booking2.guestNo}$ ∧ dateFrom $^{\neq}$ $_{dateFrom2}$ Booking2)))

4.12 *Generate the relational algebra, tuple relational calculus, and domain relational calculus expressions for the following queries:*

(a) *List all hotels.*

RA: Hotel

TRC: {H | Hotel(H)}

DRC: {hotelNo, hotelName, city | Hotel(hotelNo, hotelName, city)}

(b) *List all single rooms with a price below £20 per night.*

RA: $\sigma_{type='S' \wedge price < 20}$(Room)

TRC: {R | Room(R) ∧ R.type = 'S' ∧ R.price < 20}

DRC: {roomNo, hotelNo, type, price | (Room(roomNo, hotelNo, type, price) ∧

type = 'S' ∧ price < 20)}

(c) *List the names and cities of all guests.*

RA: $\Pi_{guestName, guestAddress}$(Guest)

TRC: {G.guestName, G.guestAddress | Guest(G)}

DRC: {guestName, guestAddress | (∃guestNo)

(Guest(guestNo, guestName, guestAddress))}

(d) *List the price and type of all rooms at the Grosvenor Hotel.*

RA: $\Pi_{price, type}$(Room 3 $_{hotelNo}$ ($\sigma_{hotelName = 'Grosvenor\ Hotel'}$(Hotel)))

TRC: {R.price, R.type | Room(R) ∧ (∃H) (Hotel(H) ∧ (□R.hotelNo = H.hotelNo) ∧

(H.hotelName = 'Grosvenor Hotel'))}

DRC: {price, type | (∃roomNo, hotelNo, hotelNo1, hotelName, city)

(Room(roomNo, hotelNo, type, price) ∧ Hotel(hotelNo1, hotelName, city) ∧

(hotelNo = hotelNo1) ∧ (hotelName = 'Grosvenor Hotel'))}

(e) *List all guests currently staying at the Grosvenor Hotel.*

RA: Guest 3 $_{guestNo}$ ($\sigma_{dateFrom \leq '01\text{-}01\text{-}02' \wedge dateTo \geq '01\text{-}01\text{-}02'}$ (

Booking 3 $_{hotelNo}$ ($\sigma_{hotelName = 'Grosvenor\ Hotel'}$(Hotel))))

(substitute '01-01-02' for today's date).

TRC: {G |Guest(G) ∧ ((∃B)(∃H) (Booking(B) ∧ Hotel(H) ∧ (B.dateFrom ≤ '01-01-02') ∧

(B.dateTo ≥ '01-01-02') ∧ (B.guestNo = G.guestNo) ∧

(B.hotelNo = H.hotelNo) ∧ (H.hotelName = 'Grosvenor Hotel'))))}

DRC: {guestNo, guestName, guestAddress | (∃hotelNo, guestNo1, dateFrom, dateTo,

hotelNo1, hotelName, city)

(Guest(guestNo, guestName, guestAddress) ∧

Booking(hotelNo, guestNo1, dateFrom, dateTo) ∧

Hotel(hotelNo1, hotelName, city) ∧ (guestNo = guestNo1) ∧

(dateFrom ≤ '01-01-02' ∧ dateTo ≥ '01-01-02') ∧

(hotelNo = hotelNo1) ∧ (hotelName = 'Grosvenor Hotel'))}

(f) *List the details of all rooms at the Grosvenor Hotel, including the name of the guest staying in the room, if the room is occupied.*

RA: (Room 3 $_{hotelNo}$ ($\sigma_{hotelName = \text{'Grosvenor Hotel'}}$(Hotel))) 5 // Outer Join

$\Pi_{guestName, hotelNo, roomNo}$(

(Guest 3 $_{guestNo}$ ($\sigma_{dateFrom \leq \text{'01-01-02'} \wedge dateTo \geq \text{'01-01-02'}}$ (

Booking 3 $_{hotelNo}$ ($\sigma_{hotelName=\text{'Grosvenor Hotel'}}$(Hotel))))

(substitute '01-01-02' for today's date).

TRC: {R, G.guestName | (Room(R) $\wedge$ ($\exists$H)(Hotel(H) $\wedge$

(R.hotelNo = H.hotelNo) $\wedge$ (H.hotelName = 'Grosvenor Hotel'))) $\vee$

(Guest(G) $\wedge$ (($\exists$B)($\exists$H) (Booking(B) $\wedge$ Hotel(H) $\wedge$

(G.guestNo = B.guestNo) $\wedge$ (B.hotelNo = H.hotelNo) $\wedge$

(H.hotelName = 'Grosvenor Hotel') $\wedge$

(B.dateFrom $\leq$ '01-01-02' $\wedge$ B.dateTo $\geq$ '01-01-02'))))}

DRC: {roomNo, hotelNo, type, price, guestName |

($\exists$hNo, hName, city, hNo1, gNo1, dFrom, dTo, rNo)

(Room(roomNo, hotelNo, type, price) $\wedge$ Hotel(hNo1, hName, city) $\wedge$

(hotelNo = hNo1) $\wedge$ (hName = 'Grosvenor Hotel')) $\vee$

(Guest(guestNo, guestName, guestAddress) $\wedge$ Hotel(hNo, hName, city) $\wedge$

Booking(hNo1, gNo1, dFrom, dTo, rNo) $\wedge$

(guestNo = gNo1) $\wedge$ (hNo1 = hNo) $\wedge$ (hName = 'Grosvenor Hotel') $\wedge$

(dFrom $\leq$ '01-01-02' $\wedge$ dTo $\geq$ '01-01-02'))))}

(g) *List the guest details (guestNo, guestName, and guestAddress) of all guests staying at the Grosvenor Hotel.*

RA: $\Pi_{guestNo, guestName, guestAddress}(Guest \ 3_{guestNo} (\sigma_{dateFrom \leq '01\text{-}01\text{-}02' \wedge dateTo \geq '01\text{-}01\text{-}02'} ($

$Booking \ 3_{hotelNo} (\sigma_{hotelName='Grosvenor\ Hotel'}(Hotel)))))$

(substitute '01-01-02' for today's date).

TRC: $\{G \mid Guest(G) \wedge ((\exists B)(\exists H)(Booking(B) \wedge Hotel(H) \wedge (B.guestNo = G.guestNo) \wedge$

$(B.hotelNo = H.hotelNo) \wedge (H.hotelName = 'Grosvenor\ Hotel') \wedge$

$(B.dateFrom \leq '01\text{-}01\text{-}02' \wedge B.dateTo \geq '01\text{-}01\text{-}02'))\}$

DRC: $\{guestNo, guestName, guestAddress \mid$

$((\exists hNo, gNo, dFrom, dTo, rNo, hNo1, hName, city)$

$(Guest(guestNo, guestName, guestAddress) \wedge$

$Booking(hNo, gNo, dFrom, dTo, rNo) \wedge Hotel(hNo1, hName, city) \wedge$

$(guestNo = gNo) \wedge (hNo = hNo1) \wedge (hName = 'Grosvenor\ Hotel') \wedge$

$(dFrom \leq '01\text{-}01\text{-}02' \wedge dTo \geq '01\text{-}01\text{-}02'))\}$

4.13 *Using relational algebra, create a view of all rooms in the Grosvenor Hotel, excluding price details. What would be the advantages of this view?*

$\Pi_{roomNo, hotelNo, type}(Room \ 3_{hotelNo} (\sigma_{hotelName='Grosvenor\ Hotel'} (Hotel)))$

Security - hides the price details from people who should not see it.
Reduced complexity - a query against this view is simpler than a query against the two underlying base relations.

4.14 *Analyze the RDBMSs that you are currently using. What types of relational languages does the system provide? For each of the languages provided, what are the equivalent operations for the eight relational algebra operations defined in Section 4.1?*

This is a small student project, the result of which is dependent on the system analyzed. However, it is likely that the supported languages will be based around SQL and QBE, in which case, the student should attempt to map the various SQL clauses to the algebra and calculus. See also Exercise 5.31.