MONASH UNIVERSITY

The Relational Data Model - Sample Solutions

1. Discuss the following terms:

- Relation
  - *A named set of attributes, consisting of a heading and a body. The heading is the schema, the body the instance (the state at a point in time)*
- Attribute
  - *A characteristic of an object/entity that we wish to record eg. customer_balance*
- Domain
  - *A set of atomic (indivisible) values from which an attribute's values are drawn. Consists of a name, data type and data format eg. gender domain: one character string with allowed values of M and F*
- Tuple
  - *A set of related attributes describing a particular instance of the relation /entity - in file terminology called a row*
- Degree and Cardinality of a Relation
  - *Degree: number of attributes in a relation*
  - *Cardinality: number of tuples*

2. Consider the CUSTOMER and ORDER relations below:

CUSTOMER (CUSTOMER-ID, NAME, ADDRESS)
ORDER (ORDER-ID, DATE, CUSTOMER-ID)

Assume a single customer may have any number of orders.

(a) Identify the primary key and foreign key attributes in these relations.

CUSTOMER Primary Key: CUSTOMER-ID
ORDER Primary Key: ORDER-ID, Foreign key: CUSTOMER-ID

(b) Can you think of a reason why we would not just store all the customer and order information in one relation so that we would not have to perform the join operation?

Doing so would result in substantial data redundancy and would lead to insert, update and delete anomalies.

3. Choosing Primary Key.

(a) In any relation, tuples must be unique. However, in many cases the set of all the attributes in a relation is not considered a candidate key. Why not?

*Although all of the attributes in a relation is a superkey, the candidate key is selected on the basis of a **minimum** superkey. For a given relation there is usually a smaller set of attributes that provide a superkey so there is no need to consider the full set of attributes as a starting point.*

On the other hand, suppose we do have a relation where the set of all attributes is a candidate key. In this case, show that this set must therefore be the only candidate key and hence the primary key.

If the set of all attributes is a candidate key (ie. a minimum superkey), there can be no other superkey and hence it must be the only candidate key and the primary key.

(b) Consider a relation that depicts a tutorial room booking in a university. Each faculty assigns a person to handle the booking for all tutorial classes for that faculty. The person's email address is given to the university's booking system as a contact person.

BOOKING (b_date, b_starttime, b_endtime, unit_code, contact_person, room_no, tutor_id)

(i) Identify candidate key(s) and primary key for the relation if the following business rules are applicable:

a. More than one tutorial classes of the same unit may run at the same time (parallel sessions are possible).

b. A tutor may teach several classes of the same unit.

c. All tutorial classes are 2 hours long.

Candidate Keys:

room_no, b_date, b_starttime

room_no, b_date, b_endtime

tutor_id, b_date, b_starttime

tutor_id, b_date, b_endtime

Primary Key:

room_no, b_date, b_starttime

*Here we would consider a **surrogate** PK of class_no or unit_code, class_no due to the complexity (number of attributes and range of data types) of the natural key*

(ii) Identify candidate key(s) and primary key for the relation if the following business rules are applicable:

a. Tutorial classes can be either 1 hour or 2 hours long.

b. A tutor can only teach one tutorial class in a given unit.

c. There are no parallel sessions of tutorial classes.

Candidate Keys (some of them, not the exhaustive list):

room_no, b_date, b_starttime

room_no, b_date, b_endtime

b_date, unit_code, tutor_id

Primary Key:

b_date, unit_code, tutor_id

4. (Adapted from Exercise 3.6 of Connolly, Begg and Strachan)

Suppose we have the following 4 relations:

HOTEL(HOTEL-NO, NAME, ADDRESS)
ROOM(ROOM-NO, HOTEL-NO, TYPE, PRICE)
BOOKING(HOTEL-NO,GUEST-NO, DATE-FROM, DATE-TO, ROOM-NO)
GUEST(GUEST-NO, NAME, ADDRESS)

Generate the relational algebra for the following queries:

(a) List the names and addresses of all hotels

AnswerA = $\pi_{\text{name, address}}$ HOTEL

(b) List all single rooms with a price below $50

AnswerB = $\sigma_{\text{type='single' and price < 50}}$ ROOM

(c) List the names and addresses of all guests

AnswerC = $\pi_{\text{name, address}}$ GUEST

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

GrosvenorNo = $\pi_{\text{hotel-no}}$ ($\sigma_{\text{name = 'Grosvenor')}}$ HOTEL)
AnswerD = $\pi_{\text{price, type}}$ (GrosvenorNo $\bowtie$ ($\pi_{\text{hotel-no, price, type}}$ ROOM))
or
AnswerD = $\pi_{\text{price, type}}$ (($\pi_{\text{hotel-no}}$ ($\sigma_{\text{name = 'Grosvenor')}}$ HOTEL)) $\bowtie$
($\pi_{\text{hotel-no, price, type}}$ ROOM))

(e) List all names and addresses of guests currently staying at the Grosvenor Hotel (assume that if the guest has a tuple in the BOOKING relation, then they are currently staying in the hotel)

$$GrosvenorNo = \pi_{hotel\text{-}no}(\sigma_{name = 'Grosvenor')} HOTEL)$$
$$GrosvenorBookings = \pi_{guest\text{-}no}(GrosvenorNo \bowtie BOOKING)$$
$$AnswerE = \pi_{name,\ address}(GrosvenorBookings \bowtie GUEST)$$
or
$$AnswerE = \pi_{name,\ address}((\pi_{guest\text{-}no}((\pi_{hotel\text{-}no}(\sigma_{name = 'Grosvenor')} HOTEL)) \bowtie BOOKING)) \bowtie GUEST)$$

5. In the readings we have looked at 7 relational algebra operators, namely:

SELECTION, PROJECTION, JOIN, UNION, INTERSECTION, DIFFERENCE and CARTESIAN PRODUCT. In fact, 5 of these operators may be considered primitive operators in the sense that the others may be expressed in terms of the primitive operators. The primitive operators are:

SELECTION, PROJECTION, UNION, DIFFERENCE and CARTESIAN PRODUCT

Using the sample tables below, show how the JOIN operation can be expressed in terms of the fundamental operators by showing the process to do a natural join of customer and order.

– **CUSTOMER table:**

Cust_ID	Name
1	Green
2	Blue

ORDER table:

Ord_ID	Date	Cust_ID
1	23-Feb-2014	1
2	26-Feb-2014	1
3	26-Feb-2014	2

NATURAL JOIN of ORDER and CUSTOMER

– **CUSTOMER table:**

Cust_ID	Name
1	Green
2	Blue

ORDER table:

Ord_ID	Date	Cust_ID
1	23-Feb-2014	1
2	26-Feb-2014	1
3	26-Feb-2014	2

Step 1: Cartesian Product

Name	C.Cust_ID	O.Cust_ID	Ord_ID	Date
Green	1	1	1	23-Feb-2014
Blue	2	1	1	23-Feb-2014
Green	1	1	2	26-Feb-2014
Blue	2	1	2	26-Feb-2014
Green	1	2	3	26-Feb-2014
Blue	2	2	3	26-Feb-2014

Step 2: Select C.Cust_ID = O.Cust_ID (note this is an Equi Join)

Name	C.Cust_ID	O.Cust_ID	Ord_ID	Date
Green	1	1	1	23-Feb-2014
Green	1	1	2	26-Feb-2014
Blue	2	2	3	26-Feb-2014

Step 3: Project away one of the Cust_ID columns

Final result (Natural Join):

Cust_ID	Ord_ID	Name	Date
1	1	Green	23-Feb-2014
1	2	Green	26-Feb-2014
2	3	Blue	26-Feb-2014