

The University of Leeds

COMP5050 (INF)
Information Modelling (2002–2003)

Coursework One

Deadline 5pm Monday 4th Nov 2002

Table of Contents

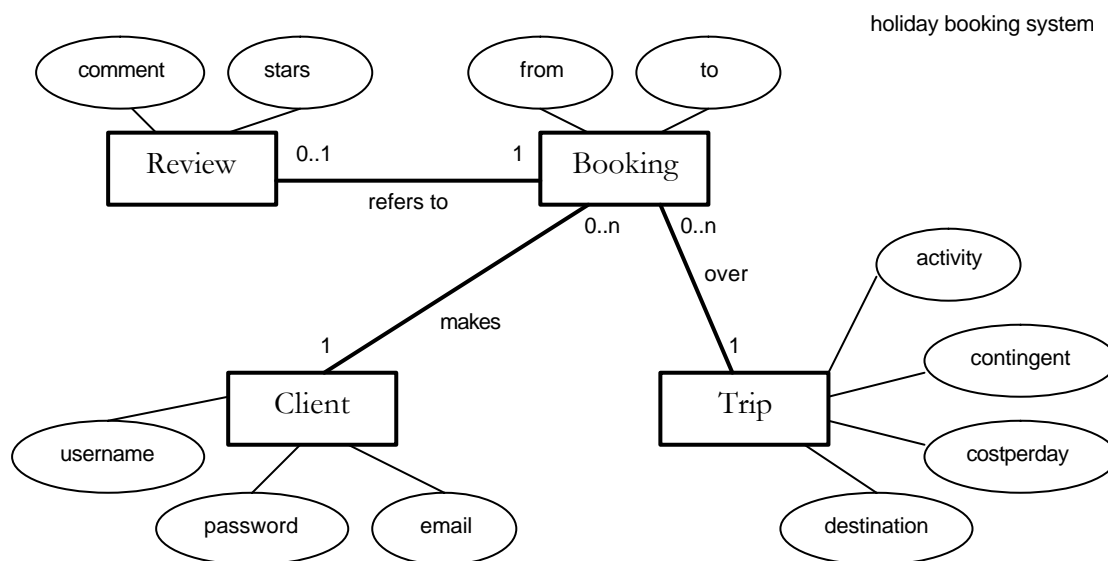
| | |
|---|----|
| Description of the problem..... | 2 |
| The data model for the prototype..... | 2 |
| Entities and their attributes..... | 2 |
| Relationships..... | 3 |
| Summary of the relational schema..... | 3 |
| Normal Forms | 4 |
| Sample set of data in the domain..... | 5 |
| Functionality and queries..... | 6 |
| Browse through catalogue and check availability | 6 |
| Recommendations..... | 8 |
| Extension of holiday-stay..... | 9 |
| Duration of stay and total cost | 9 |
| Filtering reviews depending on content..... | 10 |
| Critique on the adequacy of the data model..... | 10 |
| Conclusion..... | 12 |

David J Aumueller <ics2dja@leeds.ac.uk>
MSc Communication Studies

Description of the problem

The task for this coursework is to design and implement a *simple prototype* of a data-engine for a web-based holiday-booking system for *packaged family holidays*. The aim is to create a design that is simple but functional and scalable. There are some assumptions to be made to define the anticipated design more precisely: The family is treated uniformly as an entity, i.e. for instance the number of children and their age are not taken into account. Also, the prototype booking system is not meant to take care of (various means of) transportation or different types of hotels since holiday trips are supposed to be entire packages, i.e. entities, offering different kinds of activities though. The system being an online booking system the client is expected to pay per credit card to complete the booking-process. The customer should have the possibility to extend his stay and enter a review of his trip. The reviews containing a rating scheme might be taken into account to recommend packages to other clients.

The data model for the prototype



Entity-Relationship Diagram

Entities and their attributes

Entity Client resembles a representative of a family going on holiday. The attributes *username*, *password* and *email* store details of the user like his username and password to log on to the system on the web and his email-address to check his identity and offer a mean of communication.

Entity Trip contains information about the different packaged trips the agency offers. As attributes it holds the type of *activity* of the package, the location of the trip as

destination, the *cost per day* for this trip and finally the amount of 'pre-booked' trips of this kind as *contingent*, i.e. the quantity of trips the agency can offer simultaneously.

Entity Booking stores which client has booked which trip for what period of time, where two dates are kept to resemble start and end of the holiday in the attributes *from* and *to* respectively.

Entity Review holds any reviews or comments made by clients about a trip the client has booked. The attribute *comment* contains the actual text of the review, *stars* contains an overall numerical rating of the holiday in a scale from zero to five stars.

Relationships

Client makes Booking: One client can make several bookings.

Booking over Trip: One trip can be booked by multiple bookings.

Review refers to Booking: One booking (over a trip) might be rated by null or one review.

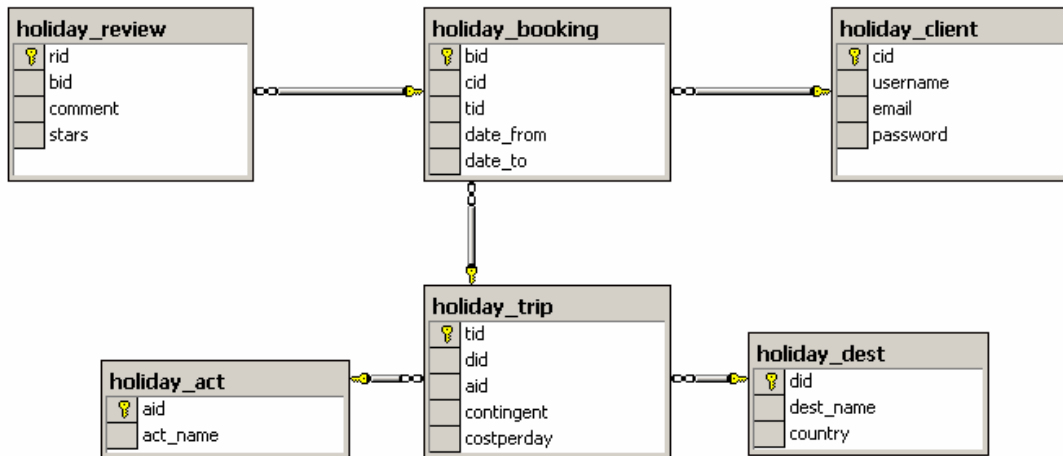
Summary of the relational schema

The database INF_ics2dja is located on MS SQL Server CSMS11.

The following SQL statements can be used to create the schema:

```
CREATE SCHEMA AUTHORIZATION ics2dja

create table holiday_client (
    cid int not null primary key,
    username char(255) not null,
    password char(255) not null,
    email char(255) not null
)
create table holiday_dest (
    did int not null primary key,
    dest_name char(255) not null,
    country char(255) not null
)
create table holiday_act (
    aid int not null primary key,
    act_name char(255) not null
)
create table holiday_trip (
    tid int not null primary key,
    did int not null references holiday_dest,
    aid int not null references holiday_act,
    contingent int not null,
    costperday int not null
)
create table holiday_booking (
    bid int not null primary key,
    cid int not null references holiday_client,
    tid int not null references holiday_trip,
    date_from datetime not null,
    date_to datetime not null,
    check (date_from > date_to + 1)
)
create table holiday_review (
    rid int not null primary key,
    bid int not null references holiday_booking,
    comment not null char(1024),
    stars not null int,
    check (stars >= 0 and stars < 6)
)
```



Relational Schema Diagram

The entities from the Entity-Relationship Model are matched to `holiday_client`, `holiday_trip`, `holiday_booking` and `holiday_review` respectively. Each table has its own unique ID that could have been set to increment automatically upon insert. Compared to the entity-relationship model, the trip-entity is split up into three tables to structure the information of a trip more thoroughly, i.e. the type of activity and the holiday locations/destinations are held in separate tables `holiday_act` and `holiday_dest` to avoid redundancy and to normalize.

Finally there is a constraint check within the table `holiday_booking` to make sure the date for the end of the holiday is later than its beginning and the whole stay lasts at least three days. The amount of stars in the table `holiday_review` is checked to be between 0 and 5 (inclusively) by a second constraint check.

Normal Forms

The schema is at least in the first normal form (1NF), since all attribute values are single atomic values. It is at least in 2NF because every non-prime attribute is fully functionally dependent on the primary key of its table. It is in 3NF since no non-prime attribute is transitively dependent on the primary key.

The schema is in the Boyce-Codd normal form (BCNF) because every attribute is dependent on a key-attribute, i.e. every determinant is a (candidate) key.

Sample set of data in the domain

To present the sample data, the results of SELECT * FROM-statements over every table are listed below.

Table holiday_trip

| tid | did | aid | contingent | costperday |
|-----|-----|-----|------------|------------|
| 1 | 1 | 1 | 5 | 55 |
| 2 | 2 | 2 | 5 | 52 |
| 3 | 3 | 3 | 5 | 53 |
| 4 | 3 | 3 | 5 | 54 |
| 5 | 3 | 2 | 5 | 55 |
| 6 | 2 | 2 | 5 | 56 |
| 7 | 1 | 1 | 5 | 57 |
| 8 | 4 | 4 | 5 | 58 |
| 9 | 5 | 4 | 5 | 59 |

Table holiday_client

| cid | username | email | password |
|-----|----------|-------------------|----------|
| 1 | Andreas | andreas@mail.org | hidden1 |
| 2 | Beatrice | beatrice@mail.org | hidden2 |
| 3 | Clement | clement@mail.org | hidden3 |
| 4 | Doris | doris@mail.org | hidden4 |
| 5 | Eric | eric@mail.org | hidden5 |
| 6 | Felix | felix@mail.org | hidden6 |

Table holiday_booking

| bid | cid | tid | date_from | date_to |
|-----|-----|-----|------------|------------|
| 1 | 1 | 1 | 2002-12-01 | 2002-12-17 |
| 2 | 2 | 2 | 2003-01-01 | 2003-01-20 |
| 3 | 3 | 3 | 2003-01-03 | 2003-01-13 |
| 4 | 2 | 5 | 1999-04-21 | 1999-05-07 |
| 5 | 5 | 5 | 2001-01-01 | 2001-01-14 |
| 6 | 6 | 6 | 2002-01-01 | 2002-01-03 |
| 7 | 3 | 7 | 2002-11-01 | 2002-11-21 |
| 8 | 6 | 2 | 2003-01-01 | 2003-01-10 |
| 9 | 5 | 3 | 2003-01-13 | 2003-01-17 |
| 10 | 1 | 2 | 2002-07-14 | 2002-07-21 |
| 11 | 1 | 1 | 2002-12-20 | 2002-12-24 |

Table holiday_dest

| did | dest_name | country |
|-----|-----------|-----------|
| 1 | Edinburgh | Schotland |
| 2 | London | England |
| 3 | Vienna | Austria |
| 4 | Bordeaux | France |
| 5 | Kopenhagn | Denmark |
| 6 | Munich | Germany |
| 7 | Miami | Florida |
| 8 | Malaga | Spain |

| | | |
|----|-----------|---------------|
| 9 | Paris | France |
| 10 | Harvard | Massachusetts |
| 11 | Cambridge | Massachusetts |
| 12 | Harvard | England |
| 13 | Cambridge | England |

Table holiday_act

| aid | act_name |
|-----|----------------|
| 1 | Skiing |
| 2 | Culture |
| 3 | Beach |
| 4 | Night-Clubbing |
| 5 | Climbing |
| 6 | Reading |
| 7 | Theme Park |

Table holiday_review

| rid | bid | comment | stars |
|-----|-----|--|-------|
| 1 | 1 | Very nice place. | 2 |
| 2 | 2 | Will definitely come back! | 3 |
| 3 | 3 | Unrecommendable! | 0 |
| 4 | 4 | Wonderful! | 4 |
| 5 | 5 | Not bad... | 1 |
| 6 | 6 | Fantastic! Best holiday ever! | 5 |
| 7 | 7 | Holiday??? Never seen such a shitty place... | 0 |
| 8 | 11 | Getting better. | 3 |
| 9 | 9 | Not really to recommend... | 1 |

Example: Eric has booked a cultural trip to Vienna for his family from 2001-01-01 to 2001-01-14 for 55 monetary units a day and rated the trip with only one star saying “Not bad...”.

Functionality and queries

This section demonstrates the functionality of the system in presenting some queries with natural language description, the SQL-statement and the results.

Browse through catalogue and check availability

The following query retrieves a list of all packaged trips that are offered, showing the location, the type of activity and the price/day (as cpd).

```
select T.tid, T.did, D.dest_name, T.aid, A.act_name, contingent, costperday as cpd
from holiday_trip T, holiday_dest D, holiday_act A
where T.did = D.did and T.aid = A.aid
```

| tid | did | dest_name | aid | act_name | contingent | cpd |
|-----|-----|-----------|-----|----------|------------|-----|
| 1 | 1 | Edinburgh | 1 | Skiing | 5 | 55 |
| 2 | 2 | London | 2 | Culture | 5 | 52 |
| 3 | 3 | Vienna | 3 | Beach | 5 | 53 |

| | | | | | | |
|---|---|------------|---|----------------|---|----|
| 4 | 3 | Vienna | 3 | Beach | 5 | 54 |
| 5 | 3 | Vienna | 2 | Culture | 5 | 55 |
| 6 | 2 | London | 2 | Culture | 5 | 56 |
| 7 | 1 | Edinburgh | 1 | Skiing | 5 | 57 |
| 8 | 4 | Bordeaux | 4 | Night-Clubbing | 5 | 58 |
| 9 | 5 | Kopenhagen | 4 | Night-Clubbing | 5 | 59 |

(9 row(s) affected)

The following query returns the availability `avail_curr` of packages (of those where already some trips are booked from) at a specific period of time by entering specific dates on the website by the customer. The user in the example wants to go on holiday from January 2nd 2003 until the 5th.

```
select T.tid, D.dest_name, T.contingent, count(B.bid) as avail_neg, (T.contingent -
count(B.bid)) as avail_curr
from holiday_booking B, holiday_trip T, holiday_dest D
where (date_from <= '2003/01/05' and date_to >= '2003/01/02') and B.tid = T.tid and
T.did = D.did group by B.tid, T.tid, D.dest_name, T.contingent
```

| tid | dest_name | contingent | avail_neg | avail_curr |
|-----|-----------|------------|-----------|------------|
| 2 | London | 5 | 2 | 3 |
| 3 | Vienna | 5 | 1 | 4 |

(2 row(s) affected)

To get a list of all available trip-packages (including destination, availability for chosen dates, type of activity and costs/day) for a specific period of time the following combination (UNION) of the two preceding queries is used (same dates as in the example above):

```
select T.tid, T.did, D.dest_name, T.contingent as avail_curr, T.aid, A.act_name,
T.costperday as cpd
from holiday_booking B, holiday_trip T, holiday_dest D, holiday_act A
where T.did = D.did
and T.aid = A.aid
union
select T.tid, T.did, D.dest_name, (T.contingent - count(B.bid)) as avail_curr, T.aid,
A.act_name, T.costperday
from holiday_booking B, holiday_trip T, holiday_dest D, holiday_act A
where (date_from <= '2003/01/05' and date_to >= '2003/01/02')
and B.tid = T.tid
and T.did = D.did
and T.aid = A.aid
group by B.tid, T.did, T.tid, D.dest_name, T.contingent, T.aid, A.act_name,
T.costperday
```

| tid | did | dest_name | avail_curr | aid | act_name | cpd |
|-----|-----|-----------|------------|-----|----------|-----|
| 1 | 1 | Edinburgh | 5 | 1 | Skiing | 55 |
| 2 | 2 | London | 3 | 2 | Culture | 52 |
| 2 | 2 | London | 5 | 2 | Culture | 52 |
| 3 | 3 | Vienna | 4 | 3 | Beach | 53 |
| 3 | 3 | Vienna | 5 | 3 | Beach | 53 |
| 4 | 3 | Vienna | 5 | 3 | Beach | 54 |
| 5 | 3 | Vienna | 5 | 2 | Culture | 55 |

| | | | | | | |
|---|---|------------|---|---|----------------|----|
| 6 | 2 | London | 5 | 2 | Culture | 56 |
| 7 | 1 | Edinburgh | 5 | 1 | Skiing | 57 |
| 8 | 4 | Bordeaux | 5 | 4 | Night-Clubbing | 58 |
| 9 | 5 | Kopenhagen | 5 | 4 | Night-Clubbing | 59 |

(11 row(s) affected)

Recommendations

To get appropriate recommendations for packaged trips the number of stars in reviewed/rated trips is primarily taken into account, followed by the price (the cheaper the better). The query returns a list (which could be limited to fetch only one or the first view rows) showing the destination, the type of activity, the amount of given reviews including average of given stars and the price/day.

Again a union of two queries is used to obtain unrated holiday-packages as well. Trips with an average rating of below 3 stars will not be recommended.

```
(
select B.tid, A.aid, A.act_name, D.dest_name, count(comment) as comments,
avg(convert(decimal, stars)) as stars_average, T.costperday
from holiday_review R, holiday_trip T, holiday_dest D, holiday_act A, holiday_booking
B
where R.bid = B.bid
and T.did = D.did
and A.aid = T.aid
and B.tid = T.tid
group by B.tid, D.dest_name, A.aid, A.act_name, T.costperday
having avg(convert(decimal, stars)) > 2
/* order by stars_average desc */
)
union
(
select T.tid, A.aid, A.act_name, D.dest_name, NULL, NULL, T.costperday as cpd
from holiday_trip T, holiday_act A, holiday_dest D, holiday_booking B
where T.aid = A.aid
and T.did = D.did
/* order by costperday asc */
)
order by stars_average desc, costperday asc
```

| tid | aid | act_name | dest_name | comments | stars_average | costperday |
|-----|-----|----------------|------------|----------|---------------|------------|
| 6 | 2 | Culture | London | 1 | 5.000000 | 56 |
| 2 | 2 | Culture | London | 1 | 3.000000 | 52 |
| 1 | 1 | Skiing | Edinburgh | 2 | 2.500000 | 55 |
| 5 | 2 | Culture | Vienna | 2 | 2.500000 | 55 |
| 2 | 2 | Culture | London | NULL | NULL | 52 |
| 3 | 3 | Beach | Vienna | NULL | NULL | 53 |
| 4 | 3 | Beach | Vienna | NULL | NULL | 54 |
| 1 | 1 | Skiing | Edinburgh | NULL | NULL | 55 |
| 5 | 2 | Culture | Vienna | NULL | NULL | 55 |
| 6 | 2 | Culture | London | NULL | NULL | 56 |
| 7 | 1 | Skiing | Edinburgh | NULL | NULL | 57 |
| 8 | 4 | Night-Clubbing | Bordeaux | NULL | NULL | 58 |
| 9 | 4 | Night-Clubbing | Kopenhagen | NULL | NULL | 59 |

(13 row(s) affected)

This result is also available via the view `holiday_recommendation`. The following query on this view returns only recommendations of trips for 'cultural activities' (`aid=2`), again ordered by the average amount of stars if rated, followed by price for unrated trips.

```
select * from holiday_recommendation
where aid=2
order by stars_average desc, costperday asc
```

| tid | aid | act_name | dest_name | no_of_comments_for_trip | stars_average | costperday |
|-----|-----|----------|-----------|-------------------------|---------------|------------|
| 6 | 2 | Culture | London | 1 | 5.000000 | 56 |
| 2 | 2 | Culture | London | 1 | 3.000000 | 52 |
| 5 | 2 | Culture | Vienna | 2 | 2.500000 | 55 |
| 2 | 2 | Culture | London | NULL | NULL | 52 |
| 5 | 2 | Culture | Vienna | NULL | NULL | 55 |
| 6 | 2 | Culture | London | NULL | NULL | 56 |

(6 row(s) affected)

Extension of holiday-stay

To extend the stay only one attribute value, `date_to` in the table `holiday_booking`, has to be changed using the following SQL-statement that represents the example having Eric prolonged his stay for more three days (booking with `bid=5`):

```
update holiday_booking
set date_to = dateadd(day, 3, date_to)
where bid = 5
```

(1 row(s) affected)

Duration of stay and total cost

To get an overview of all booked packages with duration and total costs, the following query is proposed, that also could for instance be restricted to return only bookings within one specific year, using `datepart(year, date_from) = '2002'` in the where-clause, for example.

```
SELECT B.bid, DATEDIFF(day, date_from, date_to)+1 AS no_of_days, T.costperday,
(DATEDIFF(day, date_from, date_to)+1) * T.costperday AS cost_total
FROM holiday_booking B, holiday_trip T
WHERE B.tid = T.tid
```

| bid | no_of_days | costperday | cost_total |
|-----|------------|------------|------------|
| 1 | 17 | 55 | 935 |
| 2 | 20 | 52 | 1040 |
| 3 | 11 | 53 | 583 |
| 4 | 17 | 55 | 935 |
| 5 | 14 | 55 | 770 |
| 6 | 3 | 56 | 168 |
| 7 | 21 | 57 | 1197 |
| 8 | 10 | 52 | 520 |
| 9 | 5 | 53 | 265 |
| 10 | 8 | 52 | 416 |

11 5 55 275

(11 row(s) affected)

Filtering reviews depending on content

The following two queries are examples for a subquery and the IN, LIKE, and EXISTS constructs. They both return the same list of clients who might have entered comments with offensive language:

Using IN-construct:

```
select C.cid, C.username from holiday_client C, holiday_booking B
where B.cid = C.cid
and B.bid in (
    select bid from holiday_review R
    where comment like '%shit%'
)
```

```
-----
cid      username
-----
3        Clement
```

(1 row(s) affected)

Using EXISTS:

```
select C.cid, C.username from holiday_client C
where exists (
    select B.cid from holiday_booking B
    where B.cid = C.cid
    and exists (
        select * from holiday_review R
        where R.bid = B.bid
        and comment like '%shit%'
    )
)
```

```
-----
cid      username
-----
3        Clement
```

(1 row(s) affected)

Very often there are multiple approaches to get the equivalent results; some solutions might be faster in processing, others more elegant, shorter or easier 'to read'.

Critique on the adequacy of the data model

As pointed out in the introductory description of the problem the system ought to be scaleable and therefore some of the restricting assumptions made in the beginning might be cancelled to enlarge the model.

- One could think of support for family-size and reduced prices for children. The agency might offer bonuses for clients with bookings or many reviews; the price of the prolonged stay might be lower. What about a correlation between availability and price? If availability goes down, price might go up by some factor...
- The holiday packages could be describable in detail with attached images. Depending on the quantity of pictures per trip additional database-tables would be necessary to structure the information.
- Timestamps in various tables might be introduced to keep track of the booking-date or date of review writing. Especially the latter might be checked not to take place before the actual (end of) holiday.
- At this stage the model also allows multiple bookings of one customer for the same or an overlapping period of time. That is an issue a more sophisticated system could take care of.
- The contingent for availability merely is a constant, i.e. the travel agency has 'pre-booked' a more or less fixed quantity of places, which has the advantage that this data does not need to be changed if a trip is booked and no triggers or the like are to be implemented. Availability is checked whenever a booking is in progress, or whenever the dataset is browsed to present the number of available spaces.
- Building a complete model for airplane traffic/transportation is not considered to be part of the holiday booking system since this type of information is expected to be incorporated on the travel agencies' website from the air travel-companies' result-sets.
- For some queries, IDs are important to know beforehand. They can also be retrieved by queries, for example the client's ID by his username, which therefore has to be unique.
- For booking and payment, date is entered or chosen in the web-form and total cost is calculated as mentioned earlier. Since the system is for an online service booking is only to be accepted if credit-card payment is successful. The same is true for extensions of stay: change of booking (`date_to`) is only to be accepted if extension costs ($=$ new total costs – former total costs) are being paid. Credit card information is sensible data that should not be stored because of security reasons. One could also think about alternative payment methods like cheques and bills sent to client's real name and address which would then be a necessary attribute of client-entity probably resulting in extra table(s) in the relational schema. Since payment would have to be monitored, the system would need to support the status of payment.
- The use of an extra table for the list of activities offers highly **structured information** since the string for each activity is only saved in one place; also it allows adding new activity types easily. For example, if putting destination within trip table, one could spell a destination differently (various languages/different capitalisation);

so having an extra table for that prevents redundancy and helps not to lose overview.

Conclusion

The data model offers a good basis of structured information that can easily be presented, queried and manipulated using a web or other interface. The designed and implemented simple prototype offers the functionalities anticipated in the task description at a most basic but highly scaleable level.