Brno University of Technology

Faculty of Information Technology

IDS - Databázové systémy 2021/2022

Database system for Hotel

Suvorkin Artur, (xsuvor00)

Contents

Introduction

I decided to do this project myself for a better understanding of the subject. My project demonstrates a very simple information system of the hotel and behavior between entities.

Creating basic database objects

I create tables with **INSERT INTO Statement:**

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

- accommodation
- payment
- room
- reservation
- customer
- receptionist

SQL script with queries over database tables.

For the queries I used **SELECT Statement:**

```
SELECT * FROM table_name;
```

And for some kind of information I used these keywords:

- **LEFT JOIN** (Customers who have not made a reservation; How much each customer paid)
- **RIGHT JOIN** (Reservation status; How many reservations did each of the recipients serve)
- **INNER JOIN** (Which customers worked with which recipients)

Triggers

Trigger is released after the accomodation table is populated and sets the state reservation table on "complete"

```
CREATE OR REPLACE TRIGGER TRIG_RESERV
...
   SET state1.STATE = 'complete'
   WHERE state1.id_reservation = :NEW.reservation_id;
   END;
```

Index

Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

```
PLAN_TABLE_OUTPUT

Plan hash value: 379908182
--------------------------------------------------------------------------------------------------
| Id  | Operation                       | Name            | Rows  | Bytes | Cost (%CPU)| Time     |
--------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                |                 |     2 |   272 |     8  (25)| 00:00:01 |
|   1 |  HASH UNIQUE                    |                 |     2 |   272 |     8  (25)| 00:00:01 |
|   2 |   NESTED LOOPS                  |                 |     2 |   272 |     7  (15)| 00:00:01 |
|   3 |    NESTED LOOPS                 |                 |     2 |   272 |     7  (15)| 00:00:01 |
|   4 |     NESTED LOOPS                |                 |     2 |   162 |     5  (20)| 00:00:01 |
|   5 |      VIEW                       | VW_DTP_4950B5FA |     2 |    52 |     4  (25)| 00:00:01 |
|   6 |       HASH UNIQUE               |                 |     2 |    52 |     4  (25)| 00:00:01 |
|   7 |        TABLE ACCESS FULL        | RESERVATION     |     2 |    52 |     3   (0)| 00:00:01 |
|   8 |      TABLE ACCESS BY INDEX ROWID| RECEPTIONIST    |     1 |    55 |     1   (0)| 00:00:01 |
|*  9 |       INDEX UNIQUE SCAN         | SYS_C008859     |     1 |       |     0   (0)| 00:00:01 |
|* 10 |     INDEX UNIQUE SCAN           | SYS_C008852     |     1 |       |     0   (0)| 00:00:01 |
|  11 |    TABLE ACCESS BY INDEX ROWID  | CUSTOMER        |     1 |    55 |     1   (0)| 00:00:01 |
--------------------------------------------------------------------------------------------------


Plan hash value: 2428459780
------------------------------------------------------------------------------------------------------
| Id  | Operation                             | Name          | Rows  | Bytes | Cost (%CPU)| Time     |
------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT                      |               |     1 |   120 |     4  (25)| 00:00:01 |
|   1 |  SORT GROUP BY                        |               |     1 |   120 |     4  (25)| 00:00:01 |
|   2 |   NESTED LOOPS                        |               |     1 |   120 |     3   (0)| 00:00:01 |
|   3 |    NESTED LOOPS                       |               |     1 |    94 |     3   (0)| 00:00:01 |
|   4 |     TABLE ACCESS BY INDEX ROWID       | CUSTOMER      |     1 |    55 |     1   (0)| 00:00:01 |
|*  5 |      INDEX UNIQUE SCAN                | SYS_C008852   |     1 |       |     1   (0)| 00:00:01 |
|   6 |     TABLE ACCESS BY INDEX ROWID BATCHED| PAYMENT       |     1 |    39 |     2   (0)| 00:00:01 |
|*  7 |      INDEX RANGE SCAN                 | INDEX_PAYM    |     1 |       |     1   (0)| 00:00:01 |
|*  8 |    INDEX RANGE SCAN                   | INDEX_RESERV  |     1 |    26 |     0   (0)| 00:00:01 |
------------------------------------------------------------------------------------------------------
```

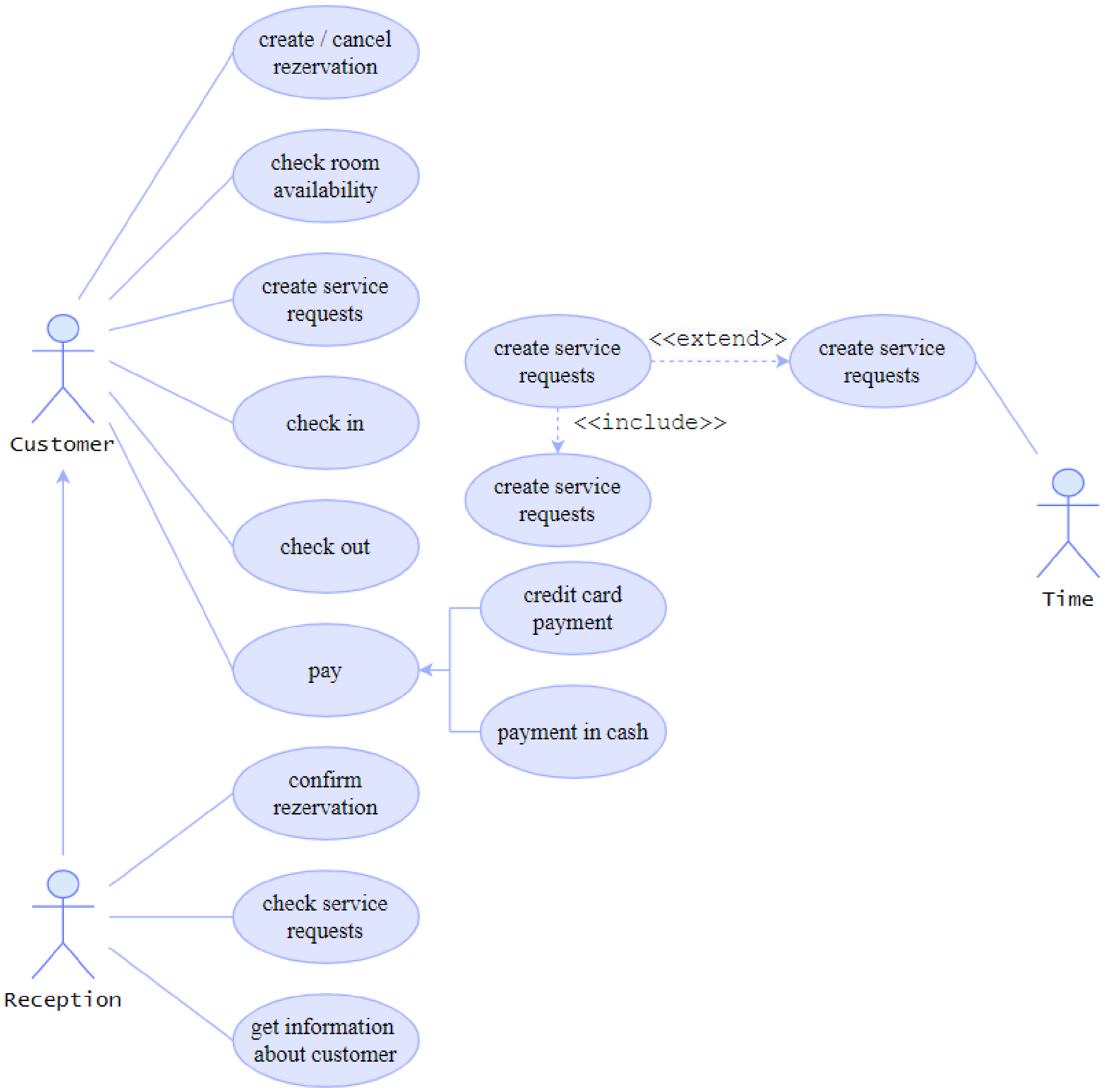
As we can see in the second table, the amount of CPU Cost decreased by 2 times due to indexes

Procedure

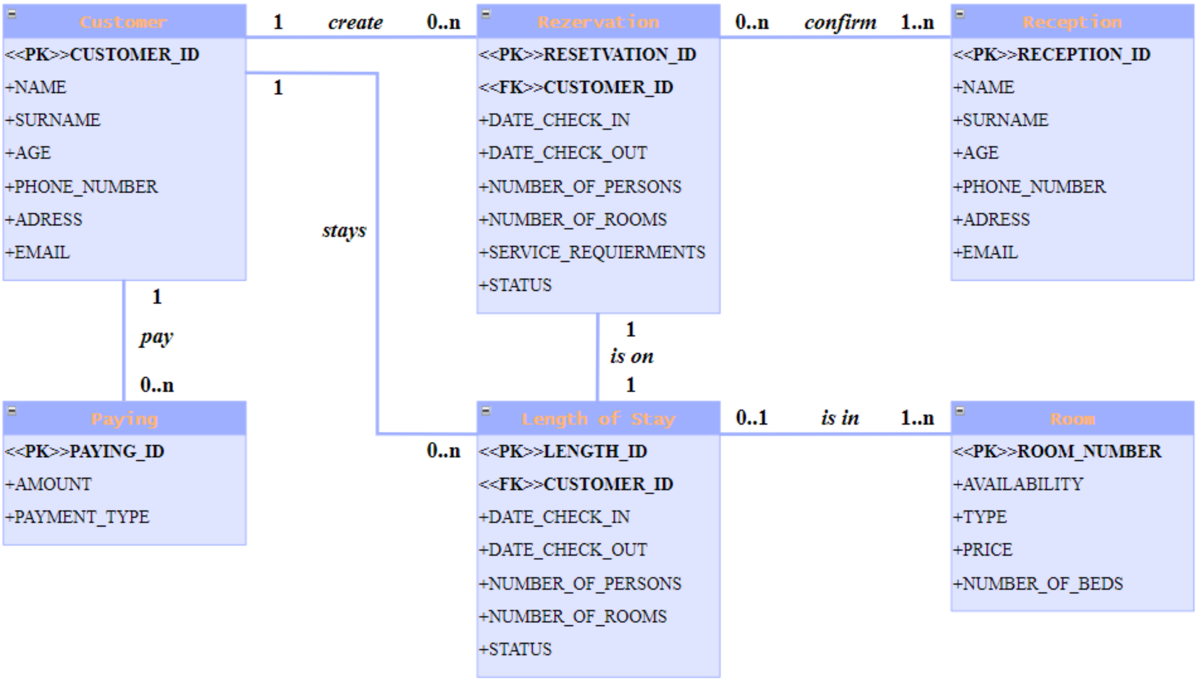
PAY_ROOM procedure compares paid reservations with the room price if PAID < ROOM prices, then email and text are displayed that the amount of money is not enough

```
 EXIT WHEN c1%NOTFOUND;
  FETCH c1 INTO v_name,v_email,v_idres, v_pay;
  EXIT WHEN c1%NOTFOUND;
   OPEN r1;
   LOOP
   EXIT WHEN r1%NOTFOUND;
   FETCH r1 INTO v_number,v_price ;
   EXIT WHEN r1%NOTFOUND;
   IF v_pay < v_price THEN
        dbms_output.put_line(v_email||' Payment is not
sufficient. Was paid = '||v_pay);
       EXIT;
    END IF;
  END LOOP;
  CLOSE r1;
SELECT * FROM demo;
```

Use Case Diagram



Entity Relationship Diagram



Resources

For everything I used https://www.w3schools.com/sql/ it really helps me with understanding