Project 2: Database of a conference center

Prepared by: Tetiana Bakai

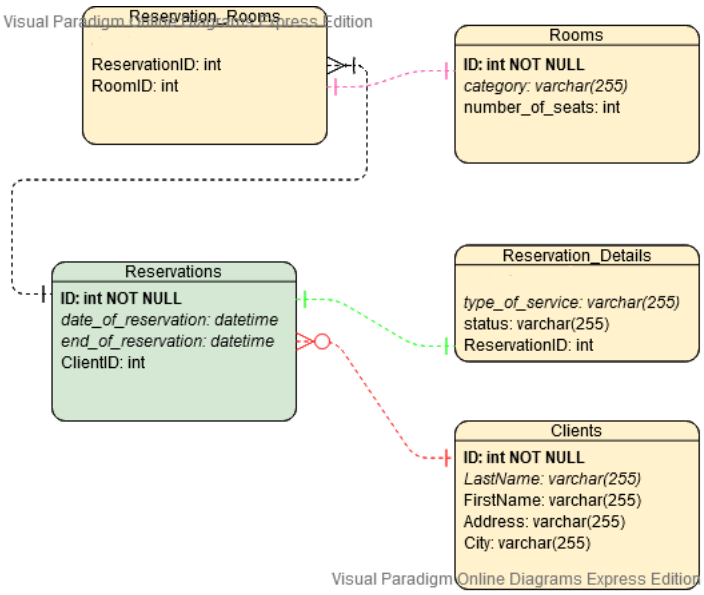
Supervised by: Aleksander Kosicki

2020

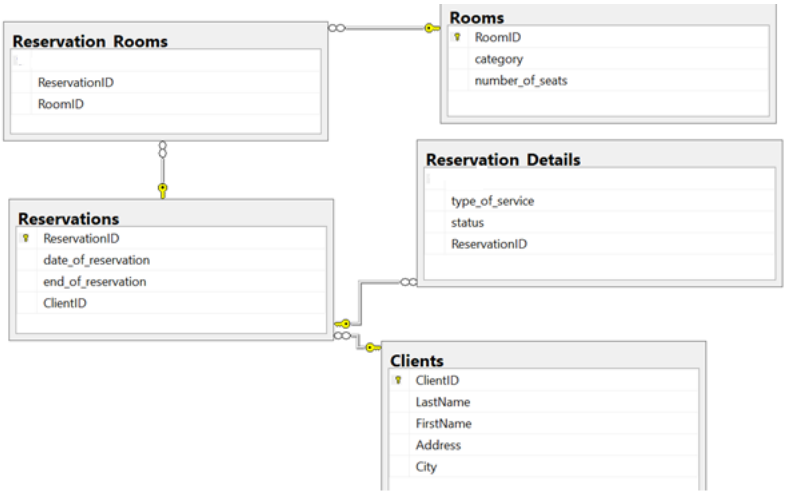
Task #1 – database design

There is a “Mermaid” conference center. It offers its guests medium size meeting rooms and larger seminar rooms. It can also provide catering or organize evening cocktail party (on the terraces by the Vistula river). Each room has a limited number of seats. Clients can reserve separate rooms and seminars, as well as a whole conference center. Room can be reserved for morning or afternoon only, the same with catering. If client decides to reserve the whole center, the reservation needs to be made for the whole days. If the reservation is confirmed, it means the rooms are booked and the catering/party organization is ordered. Single client can made multiple reservations.

ER diagram



Task #2 – data definition and manipulation



Task #3 – indexes

The data rows in a table are stored in sorted order if **Clustered index** was created.
If it is necessary to retrieve all rooms of one particular reservation quickly, we can create a clustered index on the "ReservationID" column of the Reservation_Rooms table. This way the records with the same ReservationID will be physically stored close to each other on disk (clustered) which speeds up their retrieval.

```
create Clustered index CI_Reservation_Details_IDX
        on Reservation_Details(ReservationID)

create Clustered index CI_Reservation_Rooms_IDX
        on Reservation_Rooms(ReservationID, RoomID)
```

Nonclustered indexes are frequently used for JOIN (SEARCH) operations to improve performance while searching for records matching a condition (WHERE and JOIN clause) or sorting them (ORDER BY).

If there is no clustered index created for a table, the records are stored in the form of a heap. Thus, records can be possibly inserted and updated faster as there is no need to preserve any order when inserting and updating records.

```
--select .... from .... join ... where ClientID = 7
-- can be used to sort Table by some column (Order By)

create nonclustered index IX_Reservation_Client
       on Reservations(ClientID)
```

```
create nonclustered index IX_Client
       on Clients(ClientID)

create nonclustered index IX_Client_Name
       on Clients(LastName, FirstName)

create nonclustered index IX_Client_Address
       on Clients(Address, City)

create nonclustered index IX_ReservStatus
       on Reservation_Details(status)

create nonclustered index IX_ReservService
       on Reservation_Details(type_of_service)

create nonclustered index IX_RoomCategory
       on Rooms(category, number_of_seats)

create nonclustered index IX_IDX3 on Reservations(ReservationID, ClientID)

create nonclustered index IX_Reservation_dates
       on Reservations(ReservationID, date_of_reservation, end_of_reservation)
```

Task #4 – data queries

```
-- test script 1 “seminar rooms”
with DateTable as
(
    select R.ReservationID, R.date_of_reservation as Dt, R.end_of_reservation
    from Reservations R
    union all
    select DateTable.ReservationID, DATEADD(D, 1, Dt), end_of_reservation
    from DateTable
    where DATEADD(D, 1, Dt) <= end_of_reservation
)
select concat(
'seminar rooms: ',
   case when
-- in this case, an expected value and an actual value
-- that must be equal for the test to pass
  ( select SUM(T.NumberOfRooms) FROM (select COUNT(RR.RoomID) AS NumberOfRooms, Dt as
Dates
from DateTable JOIN Reservations R ON R.ReservationID = DateTable.ReservationID
JOIN Reservation_Rooms RR ON RR.ReservationID =  R.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
JOIN Rooms RM ON RM.RoomID =  RR.RoomID
WHERE RD.status = 'confirmed' and RM.category = 'seminar room'
GROUP BY Dt) T )
  --expected value,
  = (SELECT SUM(T.AmountOfDays) AS N_SeminarRooms FROM Rooms RM
JOIN (SELECT RR.RoomID, CASE WHEN (DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) > 1 THEN COUNT(RR.RoomID)*(DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) ELSE COUNT(RR.RoomID) END AS AmountOfDays FROM
Reservation_Rooms RR
JOIN Reservations R ON R.ReservationID = RR.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed'
```

```
GROUP BY RR.RoomID, R.end_of_reservation, R.date_of_reservation) T ON
RM.RoomID=T.RoomID
WHERE RM.category = 'seminar room')
  -- actual value
  then 'passed' else 'failed' end
  );
  -- test result. 22 rows



--test script 1 “meeting rooms”
with DateTable as
(
    select R.ReservationID, R.date_of_reservation as Dt, R.end_of_reservation
    from Reservations R
    union all
    select DateTable.ReservationID, DATEADD(D, 1, Dt), end_of_reservation
    from DateTable
    where DATEADD(D, 1, Dt) <= end_of_reservation
)
select concat(
'meeting rooms: ',
   case when
-- in this case, an expected value and an actual value
-- that must be equal for the test to pass
  ( select SUM(T.NumberOfRooms) FROM (select COUNT(RR.RoomID) AS NumberOfRooms, Dt as
Dates
from DateTable JOIN Reservations R ON R.ReservationID = DateTable.ReservationID
JOIN Reservation_Rooms RR ON RR.ReservationID =  R.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
JOIN Rooms RM ON RM.RoomID =  RR.RoomID
WHERE RD.status = 'confirmed' and RM.category = 'meeting room'
GROUP BY Dt) T )
  --expected value,
  = (SELECT SUM(T.AmountOfDays) AS N_SeminarRooms FROM Rooms RM
JOIN (SELECT RR.RoomID, CASE WHEN (DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) > 1 THEN COUNT(RR.RoomID)*(DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) ELSE COUNT(RR.RoomID) END AS AmountOfDays FROM
Reservation_Rooms RR
JOIN Reservations R ON R.ReservationID = RR.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed'
GROUP BY RR.RoomID, R.end_of_reservation, R.date_of_reservation) T ON
RM.RoomID=T.RoomID
WHERE RM.category = 'meeting room')
  -- actual value
  -- the then and else branches of the case statement
  then 'passed' else 'failed' end
  );
  -- test result. 16 rows



-- test script 2 COUNT(Cl)=COUNT(Cl) that have reservations

select concat(
-- the test name
'the clients that actually booked something: ',
-- the case statement
   case when
-- one or more subqueries
```

```sql
-- in this case, an expected value and an actual value
-- that must be equal for the test to pass
  (SELECT COUNT(TD.ClientID) FROM (SELECT C.ClientID, SUM(T.AmountOfDays_perRoom) AS
totalBookedRoomDays FROM Clients C
JOIN (SELECT C.ClientID, CASE WHEN (DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) > 1 THEN COUNT(RR.RoomID)*(DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) ELSE COUNT(RR.RoomID) END AS AmountOfDays_perRoom FROM
Clients C
JOIN Reservations R ON R.ClientID = C.ClientID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
JOIN Reservation_Rooms RR ON RR.ReservationID=R.ReservationID
WHERE RD.status = 'confirmed' GROUP BY C.ClientID, R.end_of_reservation,
R.date_of_reservation) T ON C.ClientID=T.ClientID
GROUP BY C.ClientID) TD)
  --expected value,
  = (SELECT COUNT( DISTINCT C.ClientID) FROM Clients C
JOIN Reservations R ON R.ClientID = C.ClientID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed')
  -- actual value
  -- the then and else branches of the case statement
  then 'passed' else 'failed' end
  -- close the concat function and terminate the query
  );
   -- test result.  7 rows



-- test script 3 COUNT(Cl)=COUNT(Cl) that have no reservations

select concat(
-- the test name
'the clients of unconfirmed room-days reservations: ',
-- the case statement
   case when
-- one or more subqueries
-- in this case, an expected value and an actual value
-- that must be equal for the test to pass
  (SELECT COUNT(TD.ClientID) FROM (SELECT C.ClientID, SUM(T.AmountOfDays_perRoom) AS
totalBookedRoomDays FROM Clients C
JOIN (SELECT C.ClientID, CASE WHEN (DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) > 1 THEN COUNT(RR.RoomID)*(DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) ELSE COUNT(RR.RoomID) END AS AmountOfDays_perRoom FROM
Clients C
JOIN Reservations R ON R.ClientID = C.ClientID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
JOIN Reservation_Rooms RR ON RR.ReservationID=R.ReservationID
WHERE RD.status = 'cancelled' GROUP BY C.ClientID, R.end_of_reservation,
R.date_of_reservation) T ON C.ClientID=T.ClientID
GROUP BY C.ClientID) TD)
  --expected value,
  = (SELECT COUNT( DISTINCT C.ClientID) FROM Clients C
JOIN Reservations R ON R.ClientID = C.ClientID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'cancelled')
  -- actual value
  -- the then and else branches of the case statement
  then 'passed' else 'failed' end
  -- close the concat function and terminate the query
  );
   -- test result.  2 rows
```

```sql
-- test script 4 if this date<>Sunday and not exists in table

DECLARE @Avail AS datetime = NULL;


with DateTable as
(
    select R.ReservationID, R.date_of_reservation as Dt, R.end_of_reservation
    from Reservations R
    union all
    select DateTable.ReservationID, DATEADD(D, 1, Dt), end_of_reservation
    from DateTable
    where DATEADD(D, 1, Dt) <= end_of_reservation
)
SELECT @Avail FROM
(select Dt as Dates
from DateTable JOIN Reservations R ON R.ReservationID = DateTable.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed'
GROUP BY Dt
INTERSECT
SELECT m.Closest_free_day FROM
(SELECT TOP 1 c.needed_date,
CASE WHEN (datediff(DAY,  GETDATE(), c.needed_date)>1) THEN GETDATE() ELSE
c.needed_date END as Closest_free_day FROM
(SELECT TOP 1 b.beginning_date+1 as needed_date FROM
(SELECT TOP 1 R.end_of_reservation as beginning_date, LEAD(R.date_of_reservation) over
(order by R.date_of_reservation) as End_day,
DATENAME(WEEKDAY, R.end_of_reservation) AS DAY_OF_WEEK FROM Reservations R
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE R.end_of_reservation >= GETDATE() and RD.status='confirmed' ORDER BY
R.date_of_reservation) b
GROUP BY b.beginning_date, b.End_day
HAVING DAY(b.End_day)-DAY(b.beginning_date)>1 ) c) m
WHERE DATENAME(WEEKDAY, m.Closest_free_day)<>'Sunday') MN;
  -- actual value
  -- the then and else branches of the case statement
IF @Avail is NULL
    SELECT 0 AS Passed
ELSE
    SELECT @Avail AS failed



-- test script 5 count()=count(roomID) in Res_Rooms

DECLARE @Avail AS int = NULL;

SELECT @Avail FROM
  (SELECT TOP 5 TD.RoomID FROM
(SELECT TOP 5 T.RoomID FROM Rooms RM
JOIN (SELECT RR.RoomID, CASE WHEN (DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) > 1 THEN COUNT(RR.RoomID)*(DAY(R.end_of_reservation)-
DAY(R.date_of_reservation)+1) ELSE COUNT(RR.RoomID) END AS AmountOfDays FROM
Reservation_Rooms RR
JOIN Reservations R ON R.ReservationID = RR.ReservationID
JOIN Reservation_Details RD ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed'
GROUP BY RR.RoomID, R.end_of_reservation, R.date_of_reservation) T ON
RM.RoomID=T.RoomID
GROUP BY T.RoomID
ORDER BY SUM(T.AmountOfDays)) TD
```

```
  --expected value,
  EXCEPT SELECT TOP 5 T.RoomID FROM(SELECT TOP 5 RR.RoomID FROM Reservation_Rooms RR
JOIN Reservations R ON R.ReservationID = RR.ReservationID JOIN Reservation_Details RD
ON RD.ReservationID =  R.ReservationID
WHERE RD.status = 'confirmed' GROUP BY RR.RoomID ORDER BY COUNT(RR.RoomID)) T ORDER BY
T.RoomID ) TT;
  -- actual value
  -- the then and else branches of the case statement
IF @Avail is NULL
    SELECT 0 AS Passed
ELSE
    SELECT @Avail as failed
```

Tast #5 – stored procedure

```
-- test script of stored procedure


BEGIN
SELECT * FROM Reservation_Details WHERE ReservationID = 14;

EXEC ConfirmationOfReservations @ReservationID = 14;

SELECT * FROM Reservation_Details WHERE ReservationID = 14;
END
```

Results | Messages

	ReservationID	type_of_service	status
1	14	catering	NULL

	ReservationID	type_of_service	status
1	14	catering	confirmed

My personal data

My Student ID is **302163**. My group is GAI3 and lab group #5.

I have done the work I am submitting by myself.

List of attachments

- SQLBakaiProject2.sql
- ER_ConferenceCenter.png