

Database Management Project

Gupta, Ayush

NUID:

001817303

Title: Database Design for Online Hotel Booking Management Database

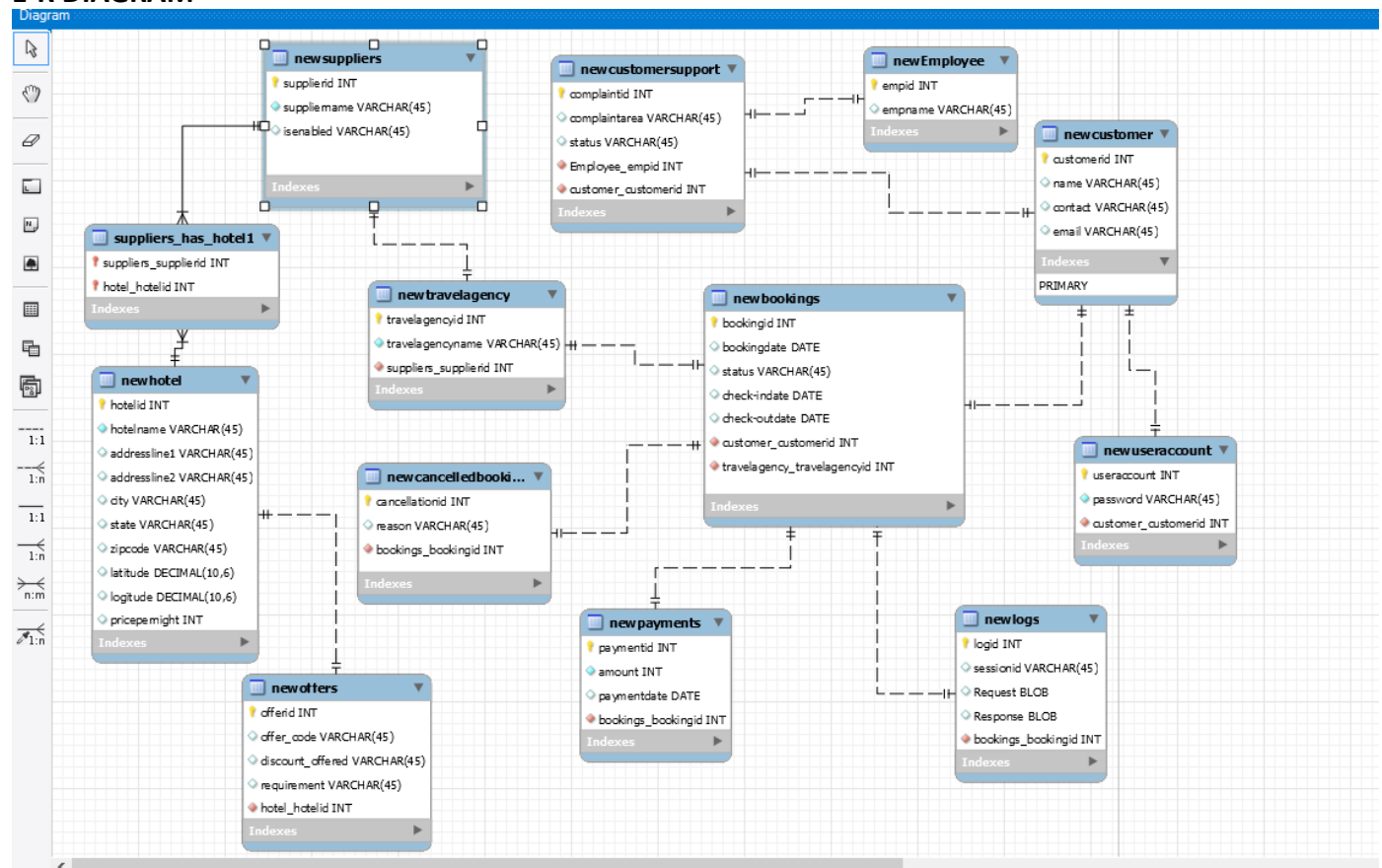
SUMMARY

There are thousands of travel bookings happening every day online. The Online travel booking market is expected to grow exponentially by 2020. The online travel industry should be competent enough to handle such huge amount of data along with maintaining their employee and customers.

SCOPE

The Database model I have designed is scalable and intelligent to meet the needs of today's dynamic market environment. Database uses have far exceeded the typical data storage needs and should be intelligent enough to manipulate the data run-time avoiding mismanagement and unnecessary revenue losses. I have designed to fit the needs of B2B model where multiple organizations come together for achieving success for each other.

E-R DIAGRAM



Triggers/Stored Procedures/Views/Events:

TRIGGERS:-

- 1) cancel_booking : Whenever a booking status gets updated to cancelled this trigger on bookings table adds that booking to the cancelled bookings table.

Syntax:

```
6
7   delimiter =|
8   • create trigger cancel_booking after update on dataproject.newbookings
9     for each row
10
11   begin
12     set @booking=NEW.bookingid;
13     IF NEW.status = 'cancelled' THEN
14       insert into dataproject.newcancelledbookings (reason,bookings_bookingid)values ('Reason',@booking);
15     end IF;
16   end#
17
18   delimiter ;
19
20
```

- 2) valid_dt: This trigger is used for validation of dates such that booking date needs to be greater than the current date and the check-in date cannot be later than the checkout date.

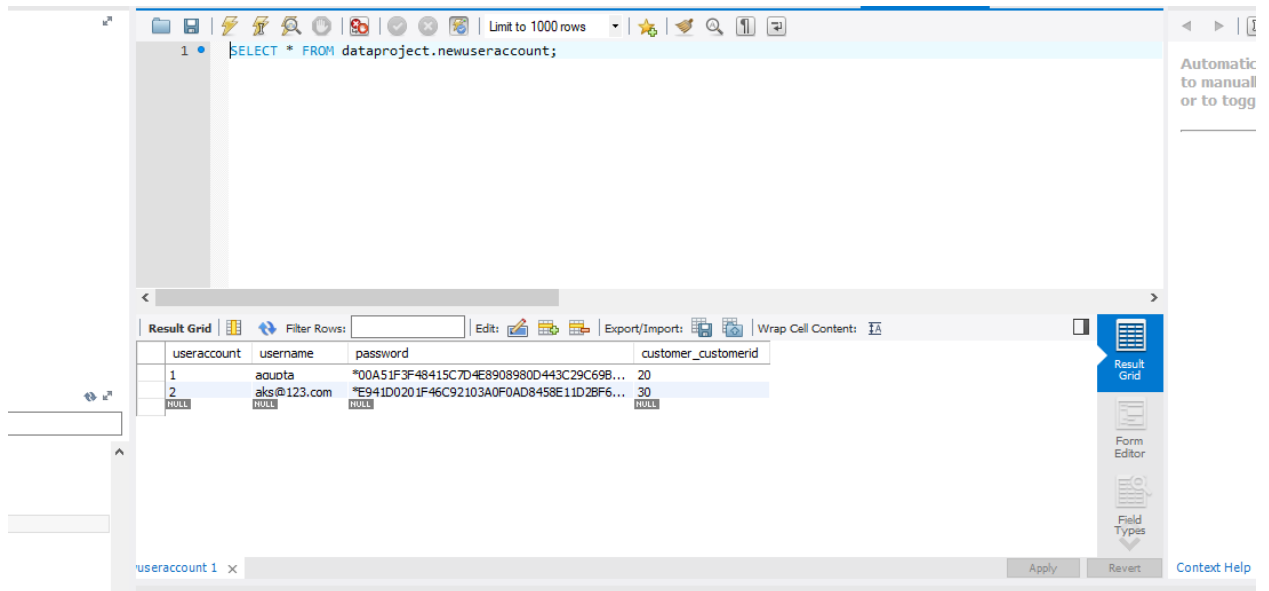
Syntax:

```
1  DELIMITER $$
2  • CREATE TRIGGER valid_dt before INSERT ON
3    dataproject.newbookings
4    FOR EACH ROW
5  BEGIN
6    if bookingdate< curdate() OR checkoutdate < checkindate
7  then
8    SIGNAL SQLSTATE '45000'
9    SET MESSAGE_TEXT = 'Please select a future booking date or a checkout date later then checkin';
10  end if;
11  END;
12  $$
13  • drop trigger valid_dt;
```

- 3) apply_offer: Trigger is used to update new offers on the hotels. If a new offer is inserted the pricepernight of the hotel is revised and updated accordingly

```
1
2  • CREATE TRIGGER apply_offer before INSERT ON
3    dataproject.newoffers
4    FOR EACH ROW
5    update newhotel set pricepernight=(NEW.discount_offered/100*pricepernight) where hotelid=New.hotel_hotelid;
6
7
8
9
```

- 4) sec_pass: Trigger is used to store the user password as a hash and not the actual password increasing the security



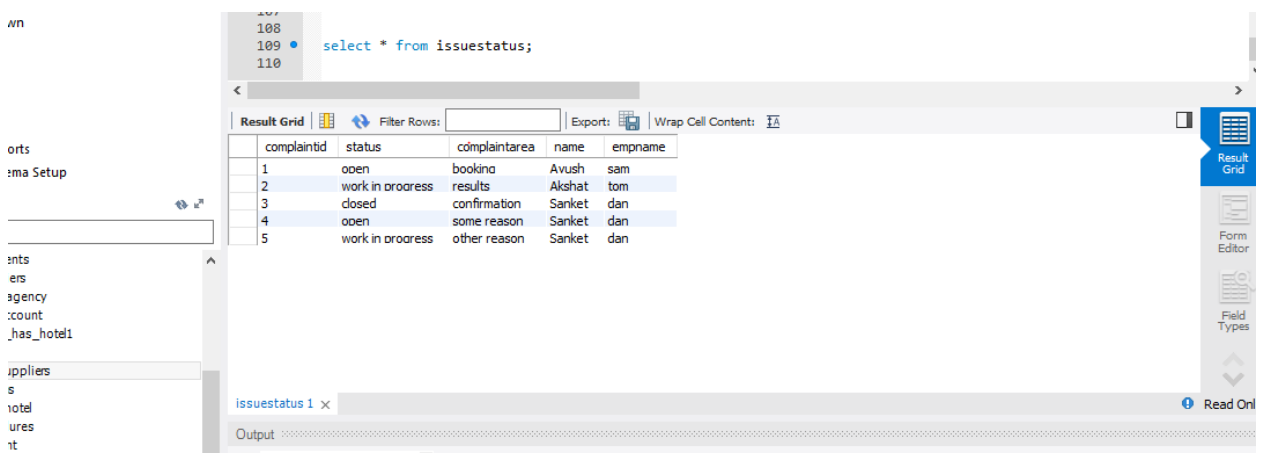
The screenshot shows a database query tool interface. At the top, a SQL query is entered: `SELECT * FROM dataproject.newuseraccount;`. Below the query, the 'Result Grid' displays the following data:

	useraccount	username	password	customer_customerid
1		adوتا	*00A51F3F48415C7D4E8908980D443C29C69B...	20
2		aks@123.com	*E941D0201F46C92103A0F0AD8458E11D2BF6...	30

The interface includes various toolbars for editing, filtering, and exporting data. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, there are 'Apply' and 'Revert' buttons, and a 'Context Help' link.

VIEWS:

- 1) issuestatus: This view returns the insights for the issues reported by the customer which includes the complaint status, employee assigned to it and the customer who reported the issue.



The screenshot shows a database query tool interface. At the top, a SQL query is entered: `select * from issuestatus;`. Below the query, the 'Result Grid' displays the following data:

	complaintid	status	complaintarea	name	empname
1		open	booking	Avush	sam
2		work in progress	results	Akshat	tom
3		closed	confirmation	Sanket	dan
4		open	some reason	Sanket	dan
5		work in progress	other reason	Sanket	dan

The interface includes various toolbars for editing, filtering, and exporting data. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'. At the bottom, there are 'Apply' and 'Revert' buttons, and a 'Context Help' link.

- 2) agency_suppliers: This view returns the travel agency along with the particular supplier they are using.

The screenshot shows a database management interface. On the left, a sidebar contains a tree view with categories like 'NCE', 'Startup / Shutdown', 'Server Logs', 'Options File', 'RMANCE', 'Dashboard', 'Performance Reports', 'Performance Schema Setup', and 'IAS'. Under 'IAS', there's a search bar and a list of objects including 'newpayments', 'newsuppliers', 'newtravelagency', 'newuseraccount', 'suppliers_has_hotel1', 'Views', 'agency_suppliers', and 'issuestatus'. The main area displays a query editor with the SQL statement: `select * from agency_suppliers;`. Below the query editor is a 'Result Grid' showing the following data:

travelagencyname	supplierid	suppliername
makemvtrio	30	exodia
trio2book	42	bookinocom
trivaoo	31	hotelbeds

- 3) This view returns the hotels available with each supplier such that a travel agency can see which supplier to connect to.

The screenshot shows the same database management interface. The query editor now contains the SQL statement: `select * from supplier_hotel;`. The 'Result Grid' displays the following data:

supplierid	suppliername	hotelname
30	exodia	Sheraton
31	hotelbeds	Sheraton
42	bookinocom	Sheraton
30	exodia	Marriot
31	hotelbeds	Marriot
42	bookinocom	Marriot
30	exodia	Hilton
31	hotelbeds	Hilton
42	bookinocom	Hilton

STORED PROCEDURE:

tot_amount – This procedure takes the input as booking id and hotelid. By providing the inputs it calculates the total booking amount based on the hotel pricepernight and the duration of stay and stores the amount in the payment table.

```
1 CREATE DEFINER='root'@'localhost' PROCEDURE `tot_amount`(IN x int,IN y int)
2 BEGIN
3     DECLARE bookvar INT;
4     DECLARE amountvar INT;
5     DECLARE diff INT;
6     declare res INT;
7
8     set @diff= (SELECT DATEDIFF(checkoutdate,checkindate) from newbookings where bookingid=x );
9
10    set @bookvar= (select bookingid from newbookings where bookingid=x);
11    set @amountvar=(select pricepernight from newhotel where hotelid=y);
12    set @res=@diff*@amountvar;
13    Insert into newpayments (amount,paymentdate,bookings_bookingid) values
14    (@res,current_date(),@bookvar);
15 END
```

USER DEFINED FUNCTION:

Distance(): This function takes the hotel and customers location from customer and hotel table. The location is taken in the form of latitude and longitude. By passing the hotel id and customer id it returns the distance between them in kilometers.

```
1 CREATE DEFINER='root'@'localhost' FUNCTION `distance`(x int,y int) RETURNS decimal(10,3)
2 DETERMINISTIC
3 BEGIN
4     declare var1 decimal(10,6);
5     declare var2 decimal(10,6);
6     declare var3 decimal(10,6);
7     declare var4 decimal(10,6);
8
9     set @var1=(select latitude from newhotel where hotelid=x);
10    set @var2=(select logitude from newhotel where hotelid=x);
11    set @var3=(select latitude from newcustomer where customerid=y);
12    set @var4=(select longitude from newcustomer where customerid=y);
13
14    RETURN 6371 * 2 * ASIN(SQRT(
15        POWER(SIN((@var1 - abs(@var3)) * pi()/180 / 2),
16        2) + COS(@var1 * pi()/180 ) * COS(abs(@var3) *
17        pi()/180) * POWER(SIN((@var2 - @var4) *
18        pi()/180 / 2), 2) ));
19
20 END
```

Apply

Revert

The screenshot shows a SQL IDE with a query editor at the top containing the following SQL code:

```

1 SELECT * FROM dataproject.newCustomer;
2
3 select distance(16,20);

```

Below the query editor, the 'Result Grid' is displayed, showing the output of the last query:

distance(16,20)
96.912

The interface includes a toolbar with options like 'Filter Rows', 'Export', and 'Wrap Cell Content'. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'.

EVENTS:

Backup: The backup event is a job that can be configured to be run at schedule intervals and take incremental backups. It is better to automate the backup process rather than the manual work of taking the backup. The event writes the data to an out file of the system so that in case Database crash the sensitive data can be recovered also this can be used for sending bills, invoices to third parties or customers.

The screenshot shows a SQL IDE with a script editor containing the following SQL code:

```

92 DELIMITER $$
93 CREATE DEFINER=`root`@`localhost` EVENT `Backup`
94 ON SCHEDULE EVERY 1 minute
95 STARTS '2017-12-11 18:09:00' ON COMPLETION PRESERVE ENABLE
96 DO
97 BEGIN
98 SET @sql_text = CONCAT("SELECT * FROM dataproject.newbookings INTO OUTFILE 'C:/Users/Ayushkumar/Desktop/backup/bac
99 PREPARE s1 FROM @sql_text;
100 EXECUTE s1;
101 DEALLOCATE PREPARE s1;
102 END $$
103 DELIMITER ;
104 SET GLOBAL event_scheduler = ON;
105 show processlist;

```

Below the script editor, the 'Result Grid' is displayed, showing the output of the last query:

Id	User	Host	db	Command	Time	State	Info
1	root	localhost:3442	dataproject	Sleep	2		NULL
2	root	localhost:3443	dataproject	Query	0	init	show processlist
3	event_scheduler	localhost	NULL	Daemon	2	Waiting for next activation	NULL

The interface includes a toolbar with options like 'Limit to 1000 rows', 'Export', and 'Wrap Cell Content'. On the right side, there are buttons for 'Result Grid', 'Form Editor', and 'Field Types'.

OTHER FUNCTIONS/FEATURES USED:

DATEDIFF() – Used this function to calculate the difference between checkin and checkout dates to calculate the amount in total_amount stored procedure.

PASSWORD()- Used this for hashing the passwords of the new customer sign ups.

BLOB- Used this data type to store hotel images and session xml logs.

ENUM- Used this data type to predefine the status of the complaints filed to ('open', 'work in progress', 'closed')

PRIVILIGES:- created a new user newbie and granted only select functionality on the database.

The screenshot displays a SQL development environment. The main editor window contains the following SQL code:

```
104  
105  
106  
107 • create user 'newbie' identified by '123';  
108 • grant select on dataproject.* to 'newbie';  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132
```

Below the editor, the 'Output' window is visible, showing the results of the executed SQL statements:

#	Time	Action	Message	Duration / Fetch
1	13:59:32	create user 'newbie' identified by '123'	0 row(s) affected	0.000 sec
2	13:59:32	grant select on dataproject.* to 'newbie'	0 row(s) affected	0.000 sec