

**Submitted to Ins.Worku &**

**Ins.Tsegaw**

**Submitted date 03/10/2005**

**Bahir Dar**

**Ethiopia**

**MEMBERS OF THE GROUP**

**NAME IDNO**

1. WUBSERA MELAKU……………2035/03

2. SIMEGNEW TIZAZU……………1723/03

3. WONDIMAGEGN ALAYU……..2002/03

4. ROBEL TESFAYE………………..2647/03

5. MULUKEN HIRPA………………1481/03

6. ABEBE ALENE…………………..2222/03

THIRD YEAR CSED

Table of Content

Content Page

Chapter One

* 1. Introduction………………………………………………………………………1
     1. Background of Ethiopian Airlines…………………………………………...1
  2. Purpose of the project……………….....................................................................1

1.3. Overview of the Project…………………………………………………………………………………..2

1.4. Objectives of the Project……………………………………………………………2

1.4.1Aim of our project……………………………………………………………2

1.4.2 Specific objective of the project……………………………………………..2

1.4.3 .General objectives of the project……………………………………………3

1.5. Scope of the Project……………………………………………………………………………………3

1.6. Business rule of Ethiopia domestic airline reservation system…………………..3

1.7. Data Dictionary of Entity……………………………………………………………………………4

1.8. Data Dictionary of Relation……………………………………………………………………….5

1.9. ENTITY RELATIONAL DIAGRAM FOR

AIR LINE TICKET RESERVATION SYSTEM……………………………………………..6

1.10. RELATIONAL MAPPING………………………………………………………………………….7

1.11. Normalization………………………………………………………………………………………..8

1.12. Database diagram……………………………………………………………………………………9

Chapter Two

2.1. Function………………………………………………………………………………………………….10

2.2. Stored procedure……………………………………………………………………………………12

2.3. Trigger…………………………………………………………………………………………………….14

2.4. Security Concepts……………………………………………………………..15

* + 1. Grant Privileges………………………………………………………………15
    2. Views and Security…………………………………………………...19

2.5. Transaction ………………………………………………………………………………………..21

2.6. Backup and recovery…………………………………………………………………………...23

2.7. Distributed database……………………………………………………………………………25

**Chapter one**

**Fundamental Database Concepts**

**1.1. Introduction**

1.1.1. Background of Ethiopian Airlines

Ethiopian Airlines is owned by the [country's government](http://en.wikipedia.org/wiki/Government_of_Ethiopia). Ethiopian Air Lines was founded on 21 December 1945 and commenced operations on 8 April 1946 expanding to international flights in 1951..At the beginning, it relied upon American pilots, technicians, administrators and accountants. The firm became a share company in 1965, and changed its name from Ethiopian Air Lines to Ethiopian Airlines. Ethiopian is a member of the [International Air Transport Association](http://en.wikipedia.org/wiki/International_Air_Transport_Association), and, since 1968, of the [African Airlines Association](http://en.wikipedia.org/wiki/African_Airlines_Association) (AFRAA).As of December 2011[[update]](http://en.wikipedia.org/w/index.php?title=Ethiopian_Airlines&action=edit), the airline is also a [Star Alliance](http://en.wikipedia.org/wiki/Star_Alliance) member.

Ethiopian Airways currently has its head office at Bole International Airport, Addis Ababa, but intends to build a new head office facility.

1.2. Purpose of the Project

Airline Ticket Reservation System (ATRS) is developed to help an airline make reservations for passenger. It’s must capable to load and maintain the schedule of flights run by the airline. For each flight, the system maintains the information on origin and destination airports, the arrival and departure times, aircraft type (with its seat capacity), the fare, the flight number, and the flight date.

Besides, the system loads and maintains the seats reserved for every given date of the flight, and the list of passenger reservations. The system interfaces with the ticketing officers and the airline managers. The ticketing officer accesses flight data for the passengers considering a trip, and make reservations according to passengers' preferences.

Implementation of ATRS will definitely brings lots of benefits for the airlines. The system also eases the sales of ticket and increases its revenue.

1.3. Overview of the Project

The following points will give an overview of our project:

* The system is a completely automated system
* The system also generates appropriate reports by a particular PassengerId.
* The system helps the user to retrieve the information as quickly as possible.
* The system provides backup plans to avoid data loss that is of high priority.
* The system is reliable enough to perform in adverse conditions.
* The system provides a user-friendly interface with a realistic view.
* The system provides an error free environment.
* The system is capable enough to reduce the number of registers i.e. system updates the data from one register to another by a single click.
* The system provides search facilities to find a specific entry from the database.
* The system maintains the required details of the passengers. This proposed system will keep the track of the required information for passenger’s entry in the system. Thus, it helps user in efficient searching of information.
* The system will also keep track of information for the payment section, that is, whether the payment is done by the passenger or not.

**1.4. Objectives of the Project**

1.4.1Aim of our project

The Aim of our project is to address the limitations of the current system.

The first main goal of this system is to create a function that makes a flight ticket reservation.

1.4.2 Specific objective of the project

The following are specific objectives of our project:

* Implement validation techniques and checks that will help reduce the margin of error in operations.
* Provides adequate data backup facilities in order to ensure system restart even after a calamity.
* Searching and cataloging of data is not a problem in the system.
* The system ensures consistency.
* Should arrive at and obtain a complete automation of all the registers as well as the registers, which are used for smooth working of the firm.
* The system is a reusable and extensible model/code.

1.4.3 .General objectives of the project

The general objective of our project is to replaces the present manual system to web based system and stimulates the need of the passenger. The system also solves many problems besides to Ticket Reservation Such as:

* Scheduling for the passenger within specified flight address.
* Notice to the passenger to prepared them for flight in specified time

1.5. Scope of the Project

* The system will not issue any ticket to the passenger until the full payment has been made.
* The ticket will available for the passenger who registers in the system.
* The system will not available until the passenger has not account.

**1.6. Business rule of Ethiopia domestic airline ticket reservation system**

* Any person, under 10 years of age, cannot reserve ticket.
* The airline will carry a passenger only if the status of the passenger in the reservation table must be confirmed.
* Any passenger will be reserve before 48 hours from flight date.
* If any passenger will pay the price and the status is confirmed the flight is considered fly.
* If the passenger is cancellation of the flight, the airline will allow the passenger to pays the cancelation fee.
* The employee of the airline system is give free flight once per year. Otherwise he/she pay the price to flight.
* One airplane must be flying to one route at once flight.
* One airplane is not assign two routes at one flight.
* If one passenger has not login there is no reserve their ticket.

**1.7. Data Dictionary of Entity**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Entity | Description | Possible Attribute | Domain of Attribute | Primary Key |
| Data type |
| 1 | Passenger | The people that are reserve flight | Passenger Id | Varchar(50) | Passenger Id |
| First name | Varchar(50) |
| Last name | Varchar(50) |
| Sex | Char(6) |
| Date Of Birth | date |
| Email | Varchar(50) |
| Phone Number | Varchar(50) |
| 2 | Routes | The line that has the departure place and destination place | Route Code | Varchar(50) | Route Code |
| Price | Money |
| Departure Place | Varchar(50) |
| Destination Place | Varchar(50) |
| 3 | Airplane | The material that moves passengers from one place to another place | Air Plane Code | Varchar(50) | Air Plane Code |
| Air Plane Name | Varchar(50) |
| Air Plane Model | Varchar(50) |
| Air Plane Capacity | Varchar(50) |
| 4 | Staff Member | The person that gives ticket to the passengers | Staff Member Id | Varchar(50) | Staff Member Id |
| First Name | Varchar(50) |
| Email | Varchar(50) |
| Phone Number | Varchar(50) |
| Sex | Char(6) |

Table 1. Data Dictionary of Entity

**1.8. Data Dictionary of Relation**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| No | Relation | Description | Possible Attribute | Foreign Key | Primary Key |
| 1 | Reserve | Arrange for (a seat, ticket)to be kept for the use of a particular person | Reservation Date  Departure Date  Status | Passenger Id  Route Code | Passenger Id  Route Code |
| 2 | Flight | The process of flying a passenger from one place to anothr place with a plane | Departure Date  Departure Time  Arrival Time | Route Code  Passenger Id  Airplane Code  Staff Member Id | Route Code  Passenger Id  Airplane Code  Staff Member Id |

Table 2.Data Dictionary of Relation

1.9. ENTITY RELATIONAL DIAGRAM FOR AIR LINE TICKET RESERVATION SYSTEM

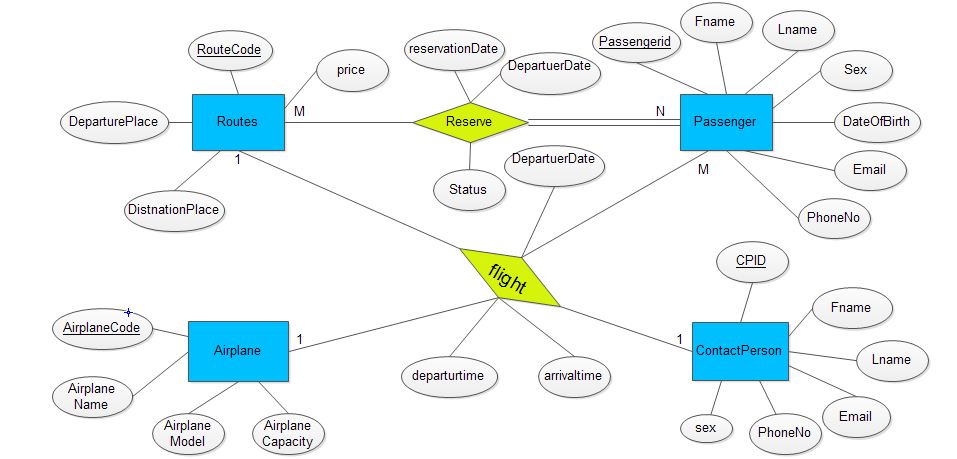


Figure 1. ER Diagram

1.10. RELATIONAL MAPPING

Passenger

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PassengerId | FName | LName | Sex | DOB | E-mail | PhoneNo |

Route

|  |  |  |  |
| --- | --- | --- | --- |
| RouteCode | DeparturePlace | DistinationPlace | price |

Reserve

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PassengerId | RouteCode | ReservationDate | DepartufreDate | Status |

Airplane

|  |  |  |  |
| --- | --- | --- | --- |
| AirplaneCode | Name | Model | Capacity |

ContactPerson

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CPID | FName | LName | Sex | E-mail | PhoneNo |

Flight

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| PassengerId | RouteCode | AirplaneCode | CPID | DepartureTime | DepartureDate | ArivalTime |

1.11. Normalization

A logical design method which minimizes data redundancy and reduces design flows

* Consists of applying various “normal” forms to the database design.
* The normal forms break down large tables into smaller subsets.

FIRST NORMAL FORM (1NF)

* Each attribute must be atomic
* No repeating columns within a row.
* No multi-valued columns.

SECOND NORMAL FORM (2NF)

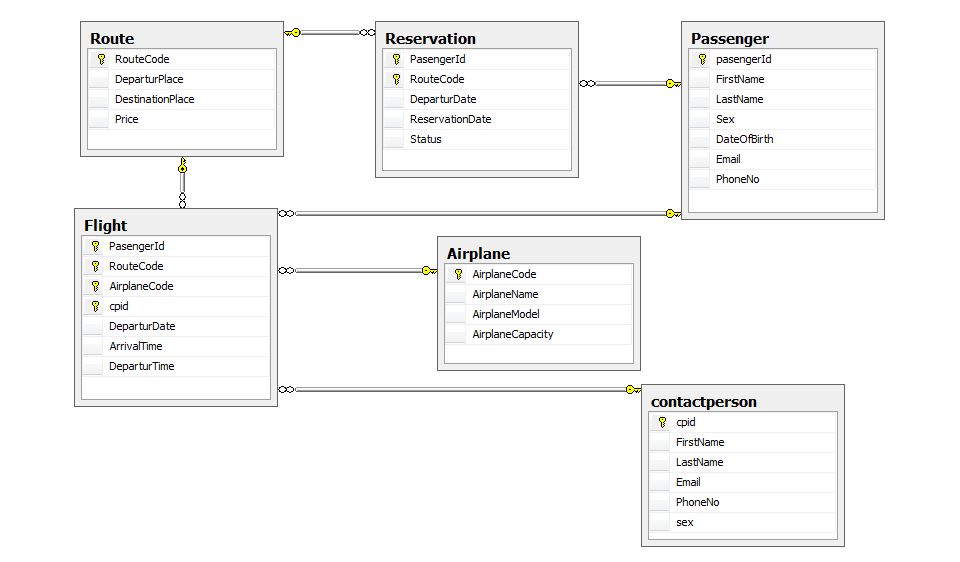
* Each attribute must be functionally dependent on the primary key.
* Functional dependence - the property of one or more attributes that uniquely determines the value of other attributes.
* Any non-dependent attributes are moved into a smaller (subset) table.

THIRD NORMAL FORM (3NF)

* Remove transitive dependencies.
* Transitive dependence - two separate entities exist within one table.
* Any transitive dependencies are moved into a smaller (subset) table.

In this case all the table i.e. Passenger, Routes, Airplane, Staff Member, Reserve, and Flight satisfies the normalization form up to third normal form.

**1.12. Database diagram**

****

**Chapter two**

**Advanced Database Concept**

* 1. **Function**

Function is used to do a specific task from a database table.

In this concept we write a function from our project.

1. When we calculate a total price of one flight: - The name of the function is calculatetotalprice that takes an argument departure place, destination place, departure date and departure time and returns total price that gets in one flight from departure place to destination place with a specific date and time. This means first of all select the rout code from route table using departure place and destination place. Then count the passenger id from flight table with departure date and departure time. Then sum the price of the route from route table.

create function calculatetotalprice(@deppplace varchar(50),@desplace varchar(50),@depdate varchar(50),@deptime varchar(50))--scalar function

returns float

as

begin

Declare @total float

set @total=(select sum(r.price) from Route r ,Flight f

where r.RouteCode=f.RouteCode and r.DeparturPlace=@depplace and r.DestinationPlace=@desplace and

f.DeparturDate=@depdate and f.DepartureTime=@deptime)

return @total

end

1. When we see the price of a specific place:- The name of the function is specificprice that takes an argument departure place and destination place and returns the price of the specific route.

create function priceselect(@dpplace varchar(50),@dsplace varchar(50))

returns table

as

return select r.price

from Route r where r.DeparturPlace=@dpplace and r.DestinationPlace=@dsplace

1. When we calculate the age of the passenger from date of birth:- the name of the function is age that takes an argument passenger id and then calculate the age of the passenger. That means select date of birth of the passenger and then subtracts the date of birth from the current date. Then we get the age of the passenger then return the age.

create function age(@passid varchar(50))

returns int

as

begin

declare @age int

declare @datofbirth date

declare @date date

set @date=(getDate())

set @datofbirth=(select DateOfBirth from Passenger p where p.pasengerId=@passid)

IF cast(datepart(m,@date) as int) > cast(datepart(m,@datofbirth) as int)

SET @age = cast(datediff(yyyy,@datofbirth,@date) as int)

else

IF cast(datepart(m,@date) as int) = cast(datepart(m,@datofbirth) as int)

IF datepart(d,@date) >= datepart(d,@datofbirth)

SET @age = cast(datediff(yyyy,@datofbirth,@date) as int)

ELSE

SET @age = cast(datediff(yyyy,@datofbirth,@date) as int) -1

ELSE

SET @age = cast(datediff(yyyy,@datofbirth,@date) as int) - 1

RETURN @age

end

* 1. **Stored procedure**

A stored procedure is an SQL server routine that is compiled and saved in the database. A stored procedure is to update, insert and delete the database table. From this concept we write a stored procedure.

1. When we select the passenger that is reserve and pay the price and insert in to flight table. The name of the procedure is inserttoflight it takes seven parameter that passes as an argument. The first step is selecting the route code from Route table using departure place and destination place. Then select the passenger id from reserve table using status (the status must be 1). Then insert in to flight table to pass the arguments passenger id from selection, Route code from selection, Airplane code from Airplane table, CPID from contactperson table, departure date, departure time and arrival time. Then execute the procedure to insert in to flight table.

go

alter procedure inserttoflight(@from varchar(50),@to varchar(50),@aircode varchar(50),@cpid varchar(50),@depdate varchar(50),@arvtime varchar(50),@deptime varchar(50))

as

declare @select varchar(50)

declare @passid varchar(50)

set @select=(select r.RouteCode from Route r where r.DeparturPlace=@from and r.DestinationPlace=@to)

set @passid=(select rs.pasengerid from Route r, Reserve rs

where r.RouteCode=rs.RouteCode and r.DeparturPlace=@from and r.DestinationPlace=@to and rs.Status=1)

insert into Flight(PasengerId,RouteCode,AirplaneCode,CPID,DeparturDate,ArrivalTime,DepartureTime)

values(@passid,@select,@aircode,@cpid,@depdate,@arvtime,@deptime)

go

1. When we select the route code depends from departure place and destination place.

The name of the procedure is that selectroutcode it takes two parameters passes as an argument. Then select the route code from Route table using as parameter departure place and destination place. Then execute the procedure to get the route code of the line (Route).

go

create procedure selectroutcode(@from varchar(50),@to varchar(50))

as

select r.RouteCode from Route r where r.DeparturPlace=@from and r.DestinationPlace=@to

go

1. When we display the schedule of airline reservation system. The name of the procedure is that schedule it works to display the full schedule of the all routes with airplane code, departure date, departure time ,departure place ,destination place of all systems.

go

create procedure schedule

as

SELECT r.departurplace, r.destinationplace,

a.airplanename, a.airplanemodel, f.departurdate,

f.DepartureTime ,f.ArrivalTime FROM Airplane a INNER JOIN flight f

ON a.airplanecode = f.airplanecode INNER JOIN Route r

ON f.RouteCode= r.RouteCode

go

1. When we reserve the route first of all select route code in terms of departure place and destination place and insert in to reserve table. the name of the procedure is that inserttoreserve it takes four parameters that passes as an argument .then select route code from Route table with departure place and destination place .then insert in to reservation table to pass an argument passenger id route code from selection, departure date and status.

go

create procedure inserttoreserve(@pasid varchar(50),@dpdate varchar(50),@from varchar(50),@to varchar(50))

as

declare @select varchar(50)

declare @date varchar(50)

set @select=(select r.RouteCode from Route r where r.departurplace=@from and r.destinationplace=@to)

set @date=(GETDATE())

insert into reserve(pasengerid,RouteCode,departurdate,ReservationDate) values(@pasid,@select,@dpdate,@date)

go

* 1. **Trigger**

The statement specifies the table on which a trigger is defined, the events for which the trigger executes, and the particular instructions for the trigger. The trigger is preventing from insertion, deletion and update times or before or after modification of a database table.

In our project we write the trigger

1. The departure place and the destination place are not the same. If the departure place and the destination place of one passenger is the same the system is raise error out of capacity.

go

create trigger cannoinsertthesameplace on Route for insert

as

if( 'insert into Route(DeparturPlace) values'='insert into Route(DestinationPlace) values' )

begin

Raiserror('You can not insert the same place at one route',16,1)

Rollback transaction

end

go

1. The passenger and contact person is not update the price from route table.

create trigger updateprice on Route for update

as

if(update(price))

begin

Raiserror('You can not update the price',16,1)

Rollback transaction

end

go

* 1. **Security Concepts**

Implementing the security system and enforcing the required restrictions are the responsibility of the DBMS. The SQL language implements a security framework and provides a rather basic syntax used to specify security restrictions. SQL *grants* access to objects by users; it does not *restrict* access. In other words, a user cannot perform any action without first being granted the power to do so.

SQL security is centered around three concepts or ideas:

❑ **Users:** Users, or more correctly user IDs, represent people or programs performing actions on

Objects in the database. Whenever the DBMS creates a new table, inserts or deletes records in that table, or modifies the records in that table, it does so on behalf of a user ID. The DBMS grants user IDs privileges to perform specific actions on specific tables.

❑ **Objects:** The SQL standards define specific objects in the database that users can manipulate. Originally the only objects were tables and views, but the standard has been expanded to include specific columns in tables as well as domains and character sets, and the individual DBMS often adds its own objects that it allows users to manipulate.

❑ **Privileges:** The third leg of SQL security comprises privileges that allow users to manipulate objects. These privileges start with SELECT, INSERT, DELETE, and UPDATE for tables and views but have been expanded to include privileges appropriate to non-data objects. Setting up security on a database involves creating objects such as tables and views, creating users, and finally granting authority to perform specific actions on specific objects to specific users. Previous chapters have described the SQL syntax that creates the tables and fields; this chapter focuses on the process of creating users and granting authority.

* + 1. ***Granting Privileges***

The basic GRANT statement is used to grant security privileges on database objects to specific users or, in some DBMS implementations, to groups. When you create a table, you are the owner of that table and you can use the GRANT statement to grant privileges to any user you want. With views, you can use the GRANT ALL statement to grant all privileges on the view. However, if you don’t own the underlying tables, the GRANT you give is only as good as the privileges you hold on the underlying tables; furthermore, you have to hold a WITH GRANT OPTION on the privileges you are granting, as explained later in this section.

The GRANT statement creates an entry in the security system that allows a user in the current database to work with data in the current database or execute specific Transact-SQL statements.

**As an example:-**In Figure below administrator has been granted full privileges on the Route table, but passenger has been granted only SELECT privileges on a columns departure place ,destination place and price and contact person has been granted only select privileges in all columns.

**User**

Administrator contact person passenger

C:\Users\wonde\Desktop\man.JPGC:\Users\wonde\Desktop\man.JPGC:\Users\wonde\Desktop\man.JPG

Select Data

Select data

Full Access

**Route**

|  |  |  |  |
| --- | --- | --- | --- |
| Route code | Departure place | Destination place | price |
|  |  |  |  |
|  |  |  |  |

**Insert Privilege: in the time of insert operation the administrator is grant in to all tables during insert, the contactperson is grant full access from passenger , reserve, flight and the passenger is grant full access from passenger table but from reserve specific columns that are passenger id, departure date.**

Grant and revoke

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Table | | | | | |
| Passenger | Reserve | Route | Flight | ContactPerson | Airplane |
| Administrator | Full access | Full access | Full access | Full access | Full access | Full access |
| Contact Person | Full access | Full access | x | Full access | x | x |
| Passenger | Full access | Access passenger id, departed date | x | x | x | x |

**Update Privilege: in the time of update operation the administrator is grant in to all tables during update, the contactperson is grant full access from passenger , reserve, flight and the passenger is grant full access from passenger table but from reserve specific columns that are passenger id, departure date.**

Grant and revoke

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Table | | | | | |
| Passenger | Reserve | Route | Flight | ContactPerson | Airplane |
| Administrator |  |  |  |  |  |  |
| Contact Person |  |  |  |  |  |  |
| Passenger |  | x | x | x | x | x |

**Select Privilege**

Grant and revoke

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Table | | | | | |
| Passenger | Reserve | Route | Flight | ContactPerson | Airplane |
| Administrator | Full Access | Full Access | Full Access | Full Access | Full Access | Full Access |
| Contact Person | Full Access | Full Access | Full Access | Full Access | Full Access | Full Access |
| Passenger | Full Access | Access ReservationDate and DepartureDate | Access  DeparturePlace  and DestinationPlace | x | x | Access Airplane Model  Airplane  Name |

**Delete Privilege**

Grant and revoke

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | Table | | | | | |
| Passenger | Reserve | Route | Flight | ContactPerson | Airplane |
| Administrator |  |  |  |  |  |  |
| Contact Person |  |  | x | x | x | x |
| Passenger | X | x | x | x | x | x |

create login adminstration with password='admin'

create user adminstration for login adminstration

sp\_addrolemember db\_owner,adminstration

sp\_addsrvrolemember adminstration,sysadmin

grant insert,update,delete,select on Airplane to adminstration

grant insert,update,delete,select on Flight to adminstration

grant insert,update,delete,select on Passenger to adminstration

grant insert,update,delete,select on Reserve to adminstration

grant insert,update,delete,select on Route to adminstration

grant insert,update,delete,select on contactperson to adminstration

create login contactperson with password=cpid

create user contactperson for login contactperson

create role staffRole authorization contactperson

sp\_addrolemember staffRole, contactperson

grant select on Airplane to contactperson

grant insert,update,select on Flight to contactperson

grant insert,update,delete,select on Passenger to contactperson

grant insert,update,delete,select on Reserve to contactperson

grant select on Route to contactperson

grant select on contactperson to contactperson

create login passenger with password='pass'

create user passenger for login passenger

create role passrole authorization passenger

sp\_addrolemember passrol,passenger

grant select on Airplane to passenger

grant select on Flight to passenger

grant insert,update,select on Passenger to passenger

grant select on Reserve to passenger

grant select on Route to passenger

grant select on contactperson to passenger

* + 1. ***Views and Security***

Views are often used to enforce security on *viewing* data. By creating views that display specific columns

From specific tables, a user can be granted SELECT privileges for that view but not for the underlying

table. Using this strategy makes it possible to determine exactly what information a specific user is

allowed to view.

Vertical and Horizontal Views

*Vertical views* represent select columns of one or more tables such that the user can see only portions of a table or tables. This allows you to hide sensitive data columns from certain users while allowing access to those columns to other users.

* The passenger should be able to see the reservation date and departure date from reserve table, but they have not seen the Status. The name of the view is seepasengerfromreservation, it select the departure date and reservation date of the passenger from reserve table.

go

CREATE VIEW seepasengerfromreservation

AS

SELECT PasengerId, DeparturDate, ReservationDate

FROM Reserve

ORDER BY ReservationDate DESC

go

* The passenger should be able to see the departure place, destination place and price of route table but they have not seen the route code. The name of the view is seepassengerfromrout it selects the departure place, destination place and price of the passenger from route table.

go

CREATE VIEW seepassengerfromrout

AS

SELECT DeparturPlace, DestinationPlace, Price

FROM Route

Go

* The passenger should be able to see full information from passenger table. The name of the view is seethirinfo it selects all from passenger table.

Go

CREATE VIEW seetherinfo

AS

SELECT pasengerId, FirstName, LastName, Sex, DateOfBirth, Email, PhoneNo

FROM Passenger

Go

* The passenger should be able to see the passenger id, airplane code, departure date, departure time and arrival time of flight table but they have not seen the route code, CPID of contact person. the name of the view is seeflightinfo it selects the passenger id ,airplane code, departure date ,departure time ,arrival time from flight table.

go

create view seeflightinfo

as

SELECT AirplaneCode, DeparturDate, DepartureTime, ArrivalTime

FROM Flight

go

* The administration should be able to see full information from all tables. That means from airplane table, contactperson table, flight table, routed table, passenger table. The name of the view seeallinfo it selects all from airplane, contactperson, flight, route, reserve, passenger tables.

go

create view seeallinfo

as

SELECT a.AirplaneCode, a.AirplaneName, a.AirplaneModel, a.AirplaneCapacity, r.RouteCode,

r.DeparturPlace, r.DestinationPlace, r.Price, f.DeparturDate,f.DepartureTime, f.ArrivalTime,

cp.cpid, cp.FirstName, cp.LastName, cp.Email, cp.PhoneNo, cp.sex,

rs.ReservationDate, rs.Status, p.pasengerId, p.FirstName AS pasfname,

p.LastName AS passlname, p.Sex AS passsex, p.DateOfBirth, p.Email AS passemail,

p.PhoneNo AS passphone

FROM Airplane a INNER JOIN

Flight f ON a.AirplaneCode = f.AirplaneCode INNER JOIN

Contactperson cp ON f.cpid = cp.cpid INNER JOIN

Route r ON f.RouteCode = r.RouteCode INNER JOIN

Reserve rs ON r.RouteCode = rs.RouteCode INNER JOIN

Passenger p ON f.PasengerId = p.pasengerId AND rs.PasengerId = p.pasengerId

ORDER BY rs.ReservationDate DESC

go

* Contactperson should be able to see full information from all tables. the name of the view seeinfocp it selects all from airplane, contactperson, flight, route, reserve, passenger tables.

go

create view seeallinfo

as

SELECT a.AirplaneCode, a.AirplaneName, a.AirplaneModel, a.AirplaneCapacity, r.RouteCode,

r.DeparturPlace, r.DestinationPlace, r.Price, f.DeparturDate,f.DepartureTime, f.ArrivalTime,

cp.cpid, cp.FirstName, cp.LastName, cp.Email, cp.PhoneNo, cp.sex,

rs.ReservationDate, rs.Status, p.pasengerId, p.FirstName AS pasfname,

p.LastName AS passlname, p.Sex AS passsex, p.DateOfBirth, p.Email AS passemail,

p.PhoneNo AS passphone

FROM Airplane a INNER JOIN

Flight f ON a.AirplaneCode = f.AirplaneCode INNER JOIN

Contactperson cp ON f.cpid = cp.cpid INNER JOIN

Route r ON f.RouteCode = r.RouteCode INNER JOIN

Reserve rs ON r.RouteCode = rs.RouteCode INNER JOIN

Passenger p ON f.PasengerId = p.pasengerId AND rs.PasengerId = p.pasengerId

ORDER BY rs.ReservationDate DESC

go

**Grant and revoke to view**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| User | VIEW | | | | | |
| seepasengerfromreservation | seepassengerfromrout | seethireinfo | seeflightinfo | seeallinfo | seeallinfocp |
| Administrator |  |  |  |  |  |  |
| Contact Person |  |  |  |  |  |  |
| Passenger |  |  |  |  | x | x |

* 1. **Transaction**

A Transaction is a mechanism for applying the desired modifications/operations (read, retrieval, write, insert or update and delete) to a database. It is evident in real life that the final database instance after a successful manipulation of the content of the database is the most up-to-date copy of the database.

**Transaction used to:**

* prevents database failure
* actions performed sequentially

Basic operations of transactions are read and write.

**Two outcomes for any transaction:**

**Success** - transaction commits and database reaches a new consistent state

Committed transaction cannot be aborted or rolled back.

**Failure** - transaction aborts, and database must be restored to consistent state before it started.

Such a transaction is rolled back or undone.

Aborted transaction that is rolled back can be restarted later.

In our system many transactions occur during

* + - The passenger registers in the flight table when status equals one. The transaction begins the update status from reserve table the status is equal to 1 and then next the insert transaction is begin and insert information in to flight table.
    - Airplane assigns for the flight after count the passenger from reserve table where the status is 1 and then check the capacity of the airplane is less than or equal to the count of the passenger .
  1. **Backup AND Recovery**

In general, **backup and recovery** refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the database after any kind of data loss. The most part of our system is to done a backup and recovery data in database .Instead of this our system is to create backup and when our system could fail in disaster system it can done recover for system database.

**In our system it can use database server is Microsoft SQL Server Database Engine, when firstly step is to create database name our project isARTS to give backup we using this step**

**Step of the Airline ticket reservation system to do backup file in SQL**

1:-After connecting to the appropriate instance of the Microsoft SQL Server Database Engine, in Object Explorer, click the server name to expand the server tree.

2:-Expand **Databases**, and depending on the database, either select a user database or expand **System Databases** and select a system database.

3:-Right-click the database, point to **Tasks**, and then click **Backup**. The **Backup Database** dialog box appears.

4:-In the **Database** list box, verify the database name. You can optionally select a different database from the list.

5:-You can perform a database backup for any recovery model (**FULL**, **BULK\_LOGGED**, or **SIMPLE**).

In the **Backup type** list box, select **Full**.

6:-Optionally, you can select **Copy Only Backup** to create a copy-only backup. A *copy-only backup* is a SQL Server backup that is independent of the sequence of conventional SQL Server backups.

7:-For **Backup component**, click **Database**.

8:-Either accept the default backup set name suggested in the **Name** text box, or enter a different name for the backup set.

9:-Optionally, in the **Description** text box, enter a description of the backup set.

10:-Specify when the backup set will expire and can be overwritten without explicitly skipping verification of the expiration data:

* To have the backup set expire after a specific number of days, click **After** (the default option), and enter the number of days after set creation that the set will expire. This value can be from 0 to 99999 days; a value of 0 days means that the backup set will never expire.
* The default value is set in the **Default backup media retention (in days)** option of the **Server Properties** dialog box (Database Settings Page). To access this, right-click the server name in Object Explorer and select properties; then select the **Database Settings** page.
* To have the backup set expire on a specific date, click **on**, and enter the date on which the set will expire.

11:-Choose the type of backup destination by clicking **Disk** or **Tape**. To select the paths of up to 64 disk or tape drives containing a single media set, click **Add**. The selected paths are displayed in the **Backup to** list box.

* To remove a backup destination, select it and click **Remove**. To view the contents of a backup destination, select it and click **Contents**.

12:-To view or select the advanced options, click **Options** in the **Select a page** pane.

13:-Select an **Overwrite Media** option, by clicking one of the following:

**Back up to the existing media set**. For this option, click either **Append to the existing backup set** or **overwrite all existing backup sets**.

* Optionally, select **Check media set name and backup set expiration** to cause the backup operation to verify the date and time at which the media set and backup set expire.
* Optionally, enter a name in the **Media set name** text box. If no name is specified, a media set with a blank name is created. If you specify a media set name, the media (tape or disk) is checked to see whether the actual name matches the name you enter here.
* **Back up to a new media set, and erase all existing backup sets**
* For this option, enter a name in the **New media set name** text box, and, optionally, describe the media set in the **New media set description** text box.

13:-In the **Reliability** section, optionally check:

**Verify backup when finished**.

* **Perform checksum before writing to media**, and, optionally, **Continue on checksum error**. ,

14:-If you are backing up to a tape drive (as specified in the **Destination** section of the **General** page), the **Unload the tape after backup** option is active. Clicking this option activates the **Rewind the tape before unloading** option.

15:-SQL Server 2008 Enterprise and later supports [backup compression](http://msdn.microsoft.com/en-us/library/bb964719.aspx). By default, whether a backup is compressed depends on the value of the **backup-compression default** server configuration option. However, regardless of the current server-level default, you can compress a backup by checking **Compress backup**, and you can prevent compression by checking **Do not** compress backup.

* To view or change the current backup compression default.

**When the system is fail in the case of disaster and so on it can use the backup file to restore (recover) the file, the second step is to see how can restore the backup file.**

**Step for recover airline ticket reservation system to done**

* After connecting to the appropriate instance of the SQL Server Database Engine, in Object Explorer, click the server name to expand the server tree.
* Expand Databases, and, depending on the database, either select a user database or expand System A Databases and select a system database.
* Right-click the database, and then click Properties, which opens the Database Properties dialog box.
* In the Select a page pane, click Options.
* The current recovery model is displayed in the Recovery model list box.
* Optionally, to change the recovery model select a different model list. The choices are Full, Bulk-logged, or Simple. Click OK.
  1. **Distribution database**

In our system we can configure distributions database store metadata and history data for all type of replication transaction for transaction replication. In our system we can use Three-Tier Client-Server Architecture. We can configure this type of architecture because of implementation; our implementation is web based application.

### To configure distribution in Airline reservation system to SQL server

1. In Microsoft SQL Server Management Studio, connect to the server that will be the Distributor (in many cases, the Publisher and Distributor are the same server), and then expand the server node.
2. Right-click the Replication folder and then click Configure Distribution.
3. Follow the Configure Distribution Wizard to:
   * Select a Distributor. To use a local Distributor, select '< MSSQLSERVER >' will act as its own Distributor; SQL Server will create a distribution database and log. To use a remote Distributor, select Use the following server as the Distributor, and then select a server. The server must already be configured as a Distributor, and the Publisher must be enabled to use the Distributor

If you select a remote Distributor, you must enter a password on the Administrative Password page for connections made from the Publisher to the Distributor. This password must match the password specified when the Publisher was enabled at the remote Distributor.

* + Specify a root snapshot folder (for a local Distributor). The snapshot folder is simply a directory that you have designated as a share; agents that read from and write to this folder must have sufficient permissions to access it. Each Publisher that uses this Distributor creates a folder under the root folder, and each publication creates folders under the Publisher folder in which to store snapshot files.
  + Specify the distribution database (for a local Distributor). The distribution database stores metadata and history data for all types of replication and transactions for transactional replication.
  + Optionally enable other Publishers to use the Distributor. If other Publishers are enabled to use the Distributor, you must enter a password on the Distributor Password page for connections made from these Publishers to the Distributor.
  + Optionally script configuration settings.