Services Online

Demo Database for MS Sql Server

Contents

Introduction

This demo database is intended to be used to promote the sale of different types of services. In this example, I will address it as a potential database to promote and sell excursion services.

The database is designed to allow the business owner to perform several configurations, like new (services) excursion, price changes, front end appearance, to be applied immediately or for the future, all these without the need to modify the front end, since the database will allow it to be dynamic, not static.

For marketing, there a structure that will allow configuring basic rules to promote slow movement products (excursions) automatically.

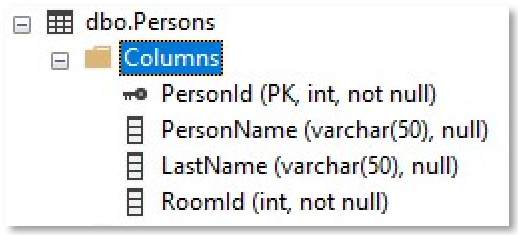
Entities Section

The potential customers can be a person or institutional entities. To consider both, we table Person and Locations.

Persons

We have the Person table, which is designed to hold data related to potential customers and hold data related to other persons, categorizing them by their roles.

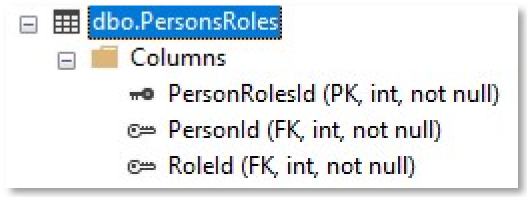
Persons



dbo.Persons
Columns
PersonId (PK, int, not null)
PersonName (varchar(50), null)
LastName (varchar(50), null)
RoomId (int, not null)

PersonId	PersonName	LastName
1	Peter	Smith
2	Paul	Smith
3	Wanda	Smith
4	Ken	Sanchez
5	Terri	Lee
6	Rob	Walters
7	Gail	Ericson
8	Jossef	Goldberg
9	Dylan	Miller
10	Michael	Raheem
11	Janice	Galvin
12	Sharon	Salavaria

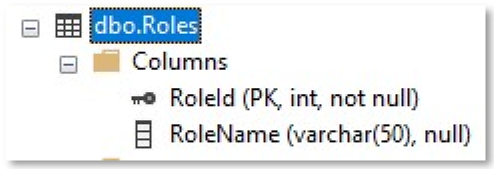
PersonRoles



dbo.PersonsRoles
Columns
PersonRolesId (PK, int, not null)
PersonId (FK, int, not null)
RoleId (FK, int, not null)

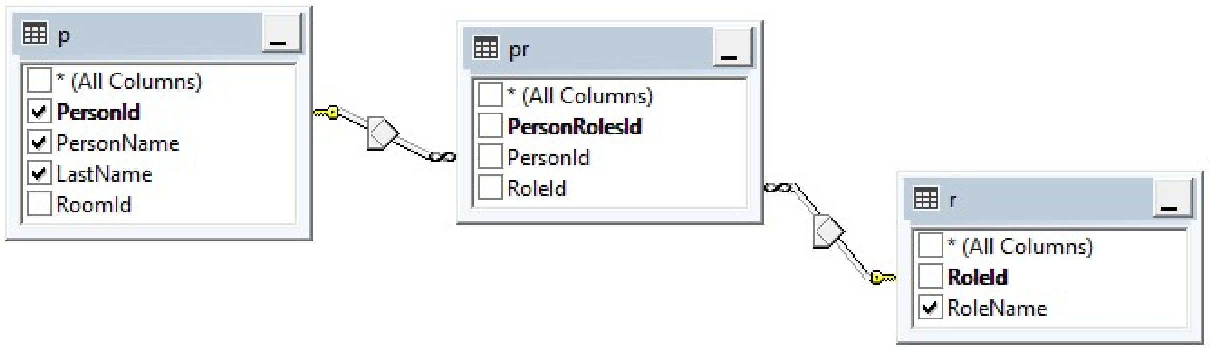
PersonRolesId	PersonId	RoleId
1	1	1
2	2	1
3	3	1
4	4	1
5	5	1
6	6	1
7	7	1
8	8	1
9	9	1
10	10	1
11	11	1
12	12	1
13	1	3
14	2	3

Roles


dbo.Roles
Columns
RoleId (PK, int, not null)
RoleName (varchar(50), null)

RoleId	RoleName
1	Customer
2	Employee
3	VIP
4	Repeater
5	Golden Club Member
6	Travel Agent
7	Inspector

With this structure, a person can have more than one role, so we use a union table to allow such configuration.



Person with their respective role.

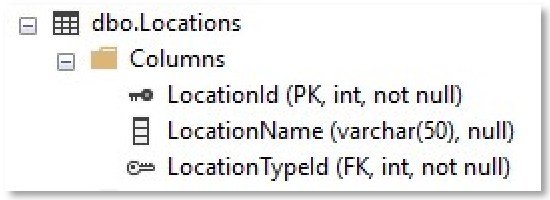
```
SELECT p.PersonId, p.LastName, p.PersonName, r.RoleName
FROM dbo.Roles AS r RIGHT OUTER JOIN dbo.PersonsRoles AS pr
ON r.RoleId = pr.RoleId RIGHT OUTER JOIN dbo.Persons AS p
ON pr.PersonId = p.PersonId
```

PersonId	LastName	PersonName	RoleName
1	Smith	Peter	Customer
1	Smith	Peter	VIP
2	Smith	Paul	Customer
2	Smith	Paul	VIP
3	Smith	Wanda	Customer
4	Sanchez	Ken	Customer
5	Lee	Terri	Customer
6	Walters	Rob	Customer
7	Ericson	Gail	Customer
8	Goldberg	Jossef	Customer
9	Miller	Dylan	Customer
10	Raheem	Michael	Customer
11	Galvin	Janice	Customer
12	Salavaria	Sharon	Customer

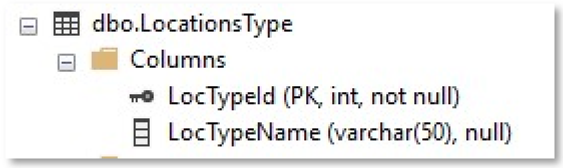
Locations

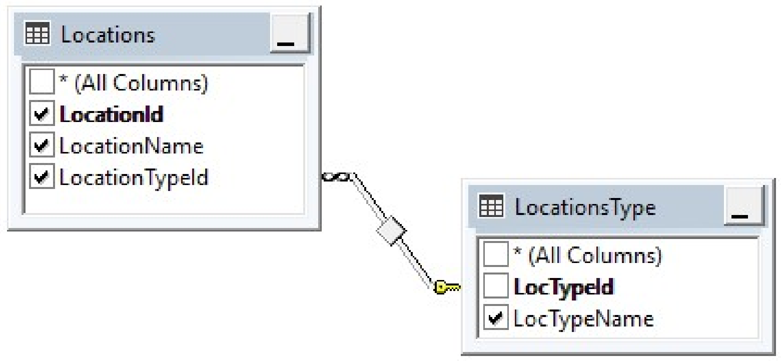
The Location allows us to store information about entities that are not a person. The LocationType object allows us to extend the Location object to any type the business needs:

Locations


dbo.Locations
Columns
LocationId (PK, int, not null)
LocationName (varchar(50), null)
LocationTypeId (FK, int, not null)

LocationsType


dbo.LocationsType
Columns
LocTypeId (PK, int, not null)
LocTypeName (varchar(50), null)



Once we classify the Locations with the LocationType, we have results as shown here:

```
SELECT dbo.Locations.LocationId, dbo.Locations.LocationName,
dbo.Locations.LocationTypeId, dbo.LocationsType.LocTypeName
FROM dbo.LocationsType INNER JOIN dbo.Locations
ON dbo.LocationsType.LocTypeId = dbo.Locations.LocationTypeId
```

Location	LocationName	LocationTypeId	LocTypeName
1	Orion	1	Cruise Ship
2	Cassiopeia	1	Cruise Ship
3	Perseus	1	Cruise Ship
4	Hercules	2	Cargo Ship
5	Ursa Major	3	Hotel
6	Paradise	4	Private Island

As we can see, a Location can be a property like a cruise ship, a hotel ,or even a private island.

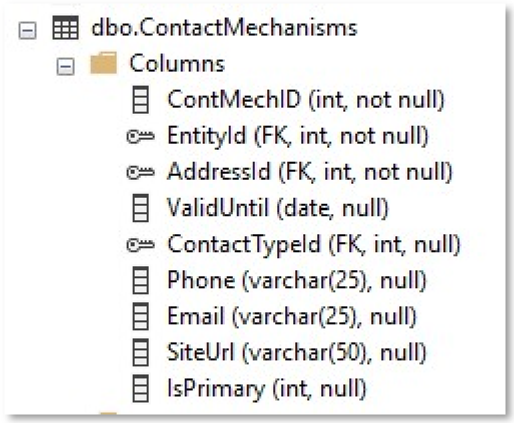
Contact Mechanisms

Customer and Locations can have more than one contact mechanisms, the table ContactType, contains some of the most commons, but there is no limit to add more as needed:

ContactTypeId	ContactTypeName
1	Home Address
2	Home Address
3	Business Address
4	Postal Address
5	Home Phone
6	Portable Phone
7	Email Address
8	Site Url

The table ContactMechanisms contains the configuration of the contact means either for Persons or Locations.

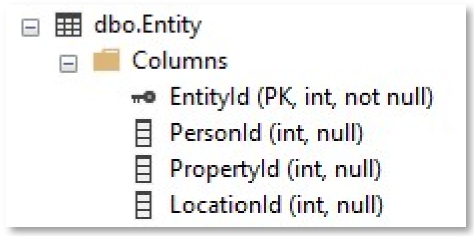
Contact Mechanisms



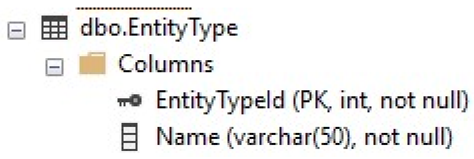
```
dbo.ContactMechanisms
  Columns
    ContMechID (int, not null)
    EntityId (FK, int, not null)
    AddressId (FK, int, not null)
    ValidUntil (date, null)
    ContactTypeId (FK, int, null)
    Phone (varchar(25), null)
    Email (varchar(25), null)
    SiteUrl (varchar(50), null)
    IsPrimary (int, null)
```

Since Person and Locations are heterogeneous objects, we use the union table name Entities to homogenize them into a single identity column:

Entity

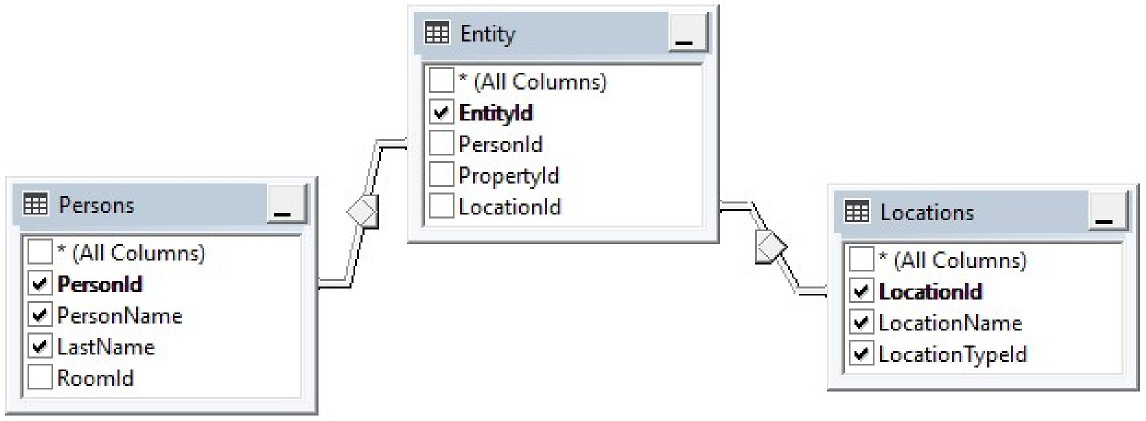


dbo.Entity
Columns
EntityId (PK, int, not null)
PersonId (int, null)
PropertyId (int, null)
LocationId (int, null)



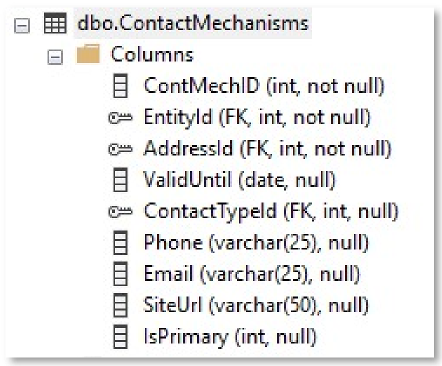
dbo.EntityType
Columns
EntityTypeId (PK, int, not null)
Name (varchar(50), not null)

In this case, the Entity table allows us to homogenize three different tables: Person, Locations, and Properties, but we care only about the Persons and Locations in this demo.



Now we have a single column EntityId, which we can use to create relations to shared objects for different objects like Persons and Locations; for example, both tables share the contact mechanism.

Before we see that in action, let us check the structure for the contact mechanisms first.

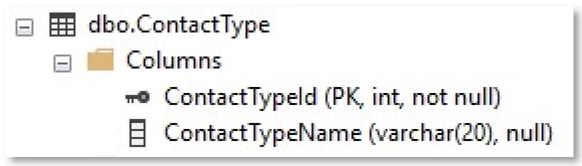


dbo.ContactMechanisms
Columns
ContMechID (int, not null)
EntityId (FK, int, not null)
AddressId (FK, int, not null)
ValidUntil (date, null)
ContactTypeId (FK, int, null)
Phone (varchar(25), null)
Email (varchar(25), null)
SiteUrl (varchar(50), null)
IsPrimary (int, null)

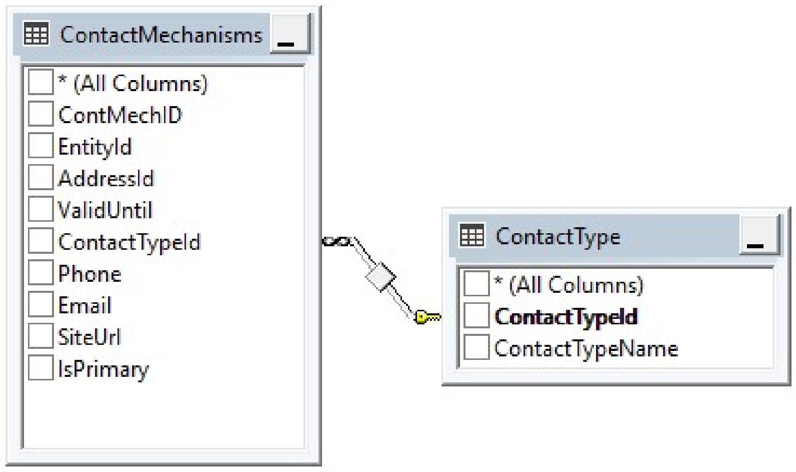
The contact mechanism is another union table with allows us to glue the Persons, Locations, or any new table to the Addresses table. Following this design pattern, I should have a different table for Phones, Emails, and other possible communication means. Still, for the demo, I added those means of communication to the Contact Mechanism table.

The ContactType classifies the contact mechanism, so we can filter out only the information we need.

ContactType



dbo.ContactType
Columns
ContactTypeId (PK, int, not null)
ContactTypeName (varchar(20), null)



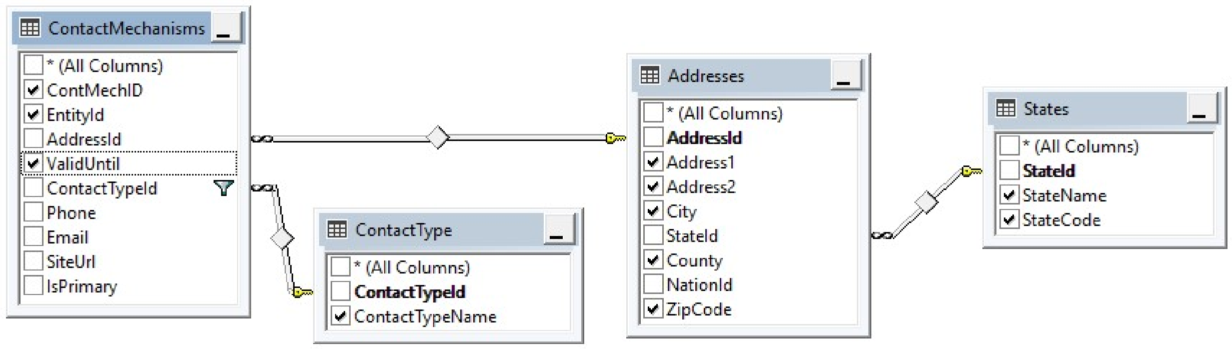
For example, only phones for a given entity id:

ContMechID	EntityId	ContactTypeName	Phone	ContactTypeId
5	1	Home Phone	305 993 4455	5
6	1	Portable Phone	305 999 6666	6
1014	4	Portable Phone	786 473 9999	6
1025	1	Portable Phone	786 473 9999	6

Email address for any given Entity:

ContMechID	EntityId	ContactTypeName	ContactTypeId	Email
7	1	Email Address	7	name@progess.com
1012	3	Email Address	7	name@success.com

Postal addresses for any given Entity:

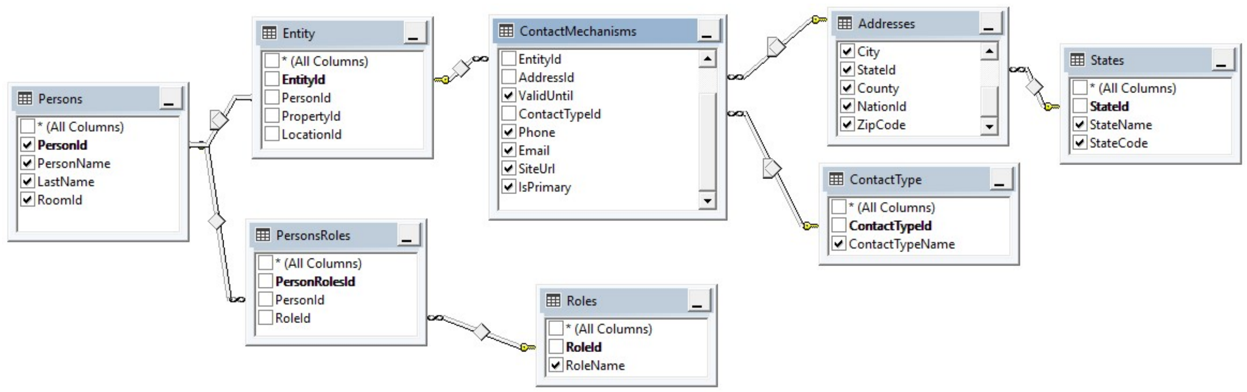


```
SELECT dbo.ContactMechanisms.ContMechID, dbo.ContactMechanisms.EntityId,
dbo.ContactType.ContactTypeName, dbo.ContactMechanisms.ValidUntil,
dbo.Addresses.Address1, dbo.Addresses.Address2, dbo.Addresses.City,
                         dbo.States.StateCode, dbo.States.StateName,
dbo.Addresses.County, dbo.Addresses.ZipCode
FROM dbo.ContactMechanisms INNER JOIN dbo.ContactType
ON dbo.ContactMechanisms.ContactTypeId = dbo.ContactType.ContactTypeId
INNER JOIN dbo.Addresses
ON dbo.ContactMechanisms.AddressId = dbo.Addresses.AddressId INNER JOIN
                         dbo.States ON dbo.Addresses.StateId = dbo.States.StateId
WHERE (dbo.ContactMechanisms.ContactTypeId = 1)
```

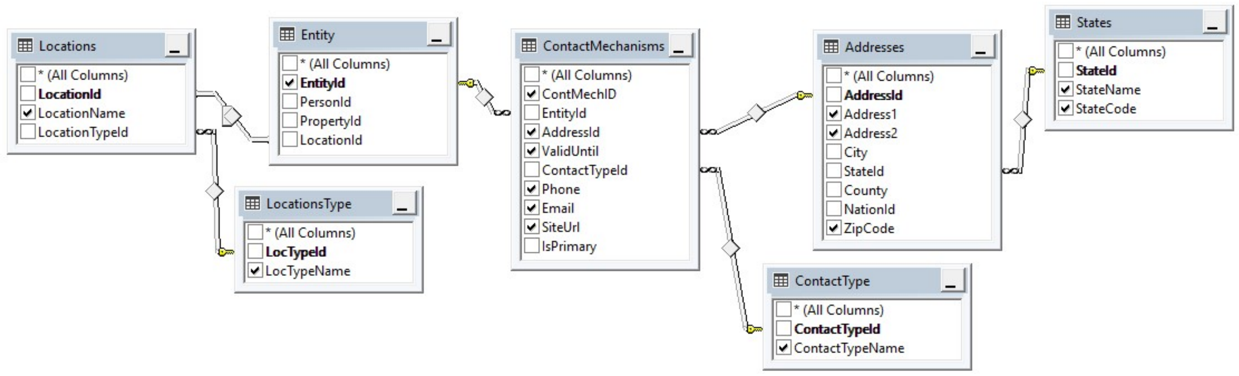
ContMechID	EntityId	ContactTypeName	ValidUntil	Address1	City	StateCode	StateName	ZipCode
1	1	Home Address	2041-03-20	1970 Napa Ct.	Bothell	WA	Washington	98011
2	2	Home Address	2041-03-20	9833 Mt. Dias Blv.	Bothell	WA	Washington	98011
1016	9	Home Address	2041-03-21	1970 Napa Ct.	Bothell	WA	Washington	98011
1017	10	Home Address	2041-03-21	1970 Napa Ct.	Bothell	WA	Washington	98011
1018	11	Home Address	2041-03-21	1970 Napa Ct.	Bothell	WA	Washington	98011
1019	12	Home Address	2041-03-21	9833 Mt. Dias Blv.	Bothell	WA	Washington	98011
1020	13	Home Address	2041-03-21	6657 Sand Pointe Lane	Seattle	WA	Washington	98104
1021	14	Home Address	2041-03-21	80 Sunview Terrace	Duluth	CA	California	55802
1023	16	Home Address	2041-03-21	9833 Mt. Dias Blv.	Bothell	WA	Washington	98011
1024	17	Home Address	2041-03-21	6657 Sand Pointe Lane	Seattle	WA	Washington	98104

Putting all these pieces together, we can construct a similar view for the Locations as shown below:

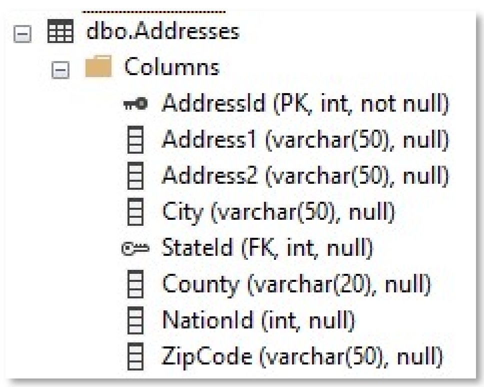
All persons and all contact mechanisms:



All locations and all contact mechanisms:



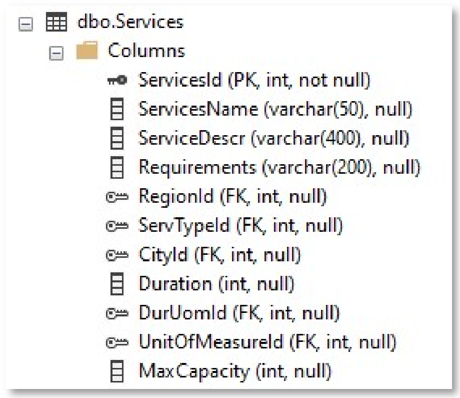
Addresses Table



dbo.Addresses
- Columns
  - AddressId (PK, int, not null)
  - Address1 (varchar(50), null)
  - Address2 (varchar(50), null)
  - City (varchar(50), null)
  - StateId (FK, int, null)
  - County (varchar(20), null)
  - NationId (int, null)
  - ZipCode (varchar(50), null)

Product

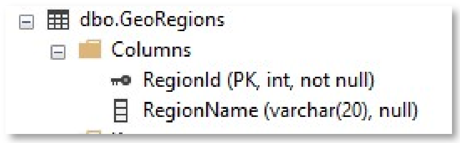
The table name Services is the one used to configure the products, in this case, excursions. Relation to child tables like Georgios, GeoCity, ServicesType, UnitOfMeasures, ServicesMedia helps configure the business's service.



```
dbo.Services
  Columns
    ServicesId (PK, int, not null)
    ServicesName (varchar(50), null)
    ServiceDescr (varchar(400), null)
    Requirements (varchar(200), null)
    RegionId (FK, int, null)
    ServTypeId (FK, int, null)
    CityId (FK, int, null)
    Duration (int, null)
    DurUomId (FK, int, null)
    UnitOfMeasureId (FK, int, null)
    MaxCapacity (int, null)
```

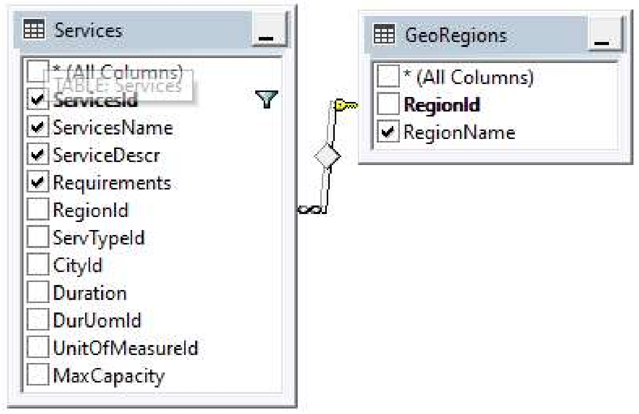
GeoRegions

Geographical regions use to classify where the product is available:



```
dbo.GeoRegions
  Columns
    RegionId (PK, int, not null)
    RegionName (varchar(20), null)
```

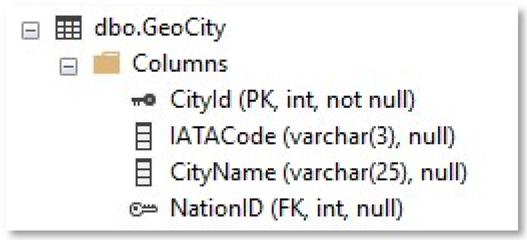
RegionId	RegionName
1	Caribbean
2	North America
3	South America
4	North Europe
5	Mediterranean



ServicesId	ServicesName	ServiceDescr	Requirements	RegionName
1	Whale Watching & Wildlife Quest	This sightseeing cruise features guaranteed whale watching!	Warm clothes	North America

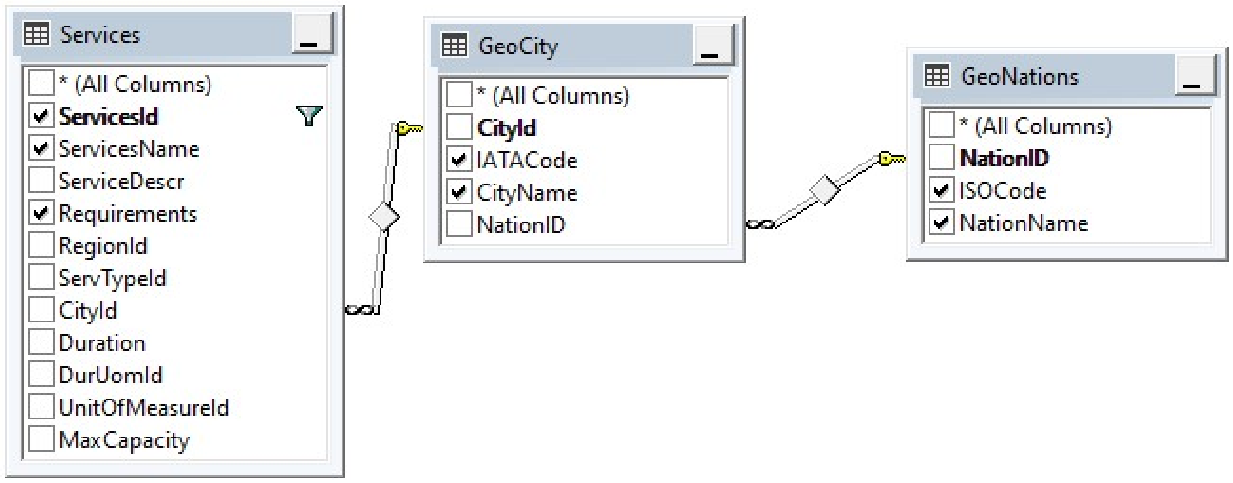
GeoCity

Used to assign in which city the excursion is done.



```
dbo.GeoCity
  Columns
    CityId (PK, int, not null)
    IATACode (varchar(3), null)
    CityName (varchar(25), null)
    NationID (FK, int, null)
```

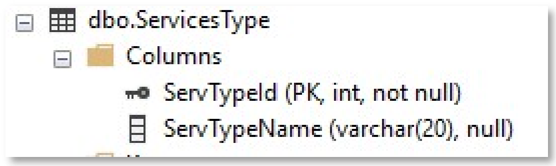
CityId	IATACode	CityName	NationID
1	JNU	Juneau, Alaska	1
2	SFO	San Francisco, California	1
3	SEA	Seattle, Washington	1
4	SGY	Skagway, Alaska	1
5	SIT	Sitka, Alaska	1
6	CZM	Cozumel, México	2
7	CUN	Costa Maya, México	2
8	RTB	Roatán, Honduras	3



ServicesId	ServicesName	Requirements	IATACode	CityName	ISOCode	NationName
1	Whale Watching & Wildlife Quest	Warm clothes	JNU	Juneau, Alaska	US	United States

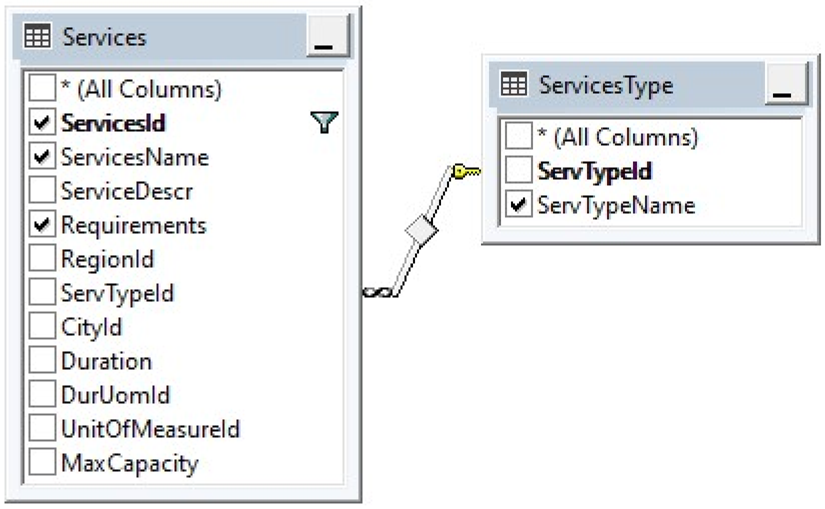
ServicesType

Used to classify the type of service (Product) this is for potential business expansion, Sport Events, Arts Events, etc.



```
dbo.ServicesType
  Columns
    ServTypeId (PK, int, not null)
    ServTypeName (varchar(20), null)
```

ServTypeId	ServTypeName
1	Excursions
2	Sport Events



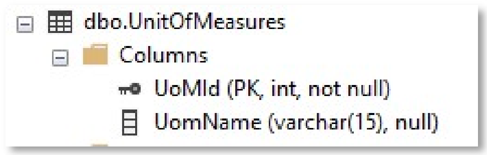
Services	
☐	* (All Columns)
✓	ServicesId
✓	ServicesName
☐	ServiceDescr
✓	Requirements
☐	RegionId
☐	ServTypeId
☐	CityId
☐	Duration
☐	DurUomId
☐	UnitOfMeasureId
☐	MaxCapacity

ServicesType	
☐	* (All Columns)
☐	ServTypeId
✓	ServTypeName

ServicesId	ServicesName	Requirements	ServTypeName
1	Whale Watching & Wildlife Quest	Warm clothes	Excursions

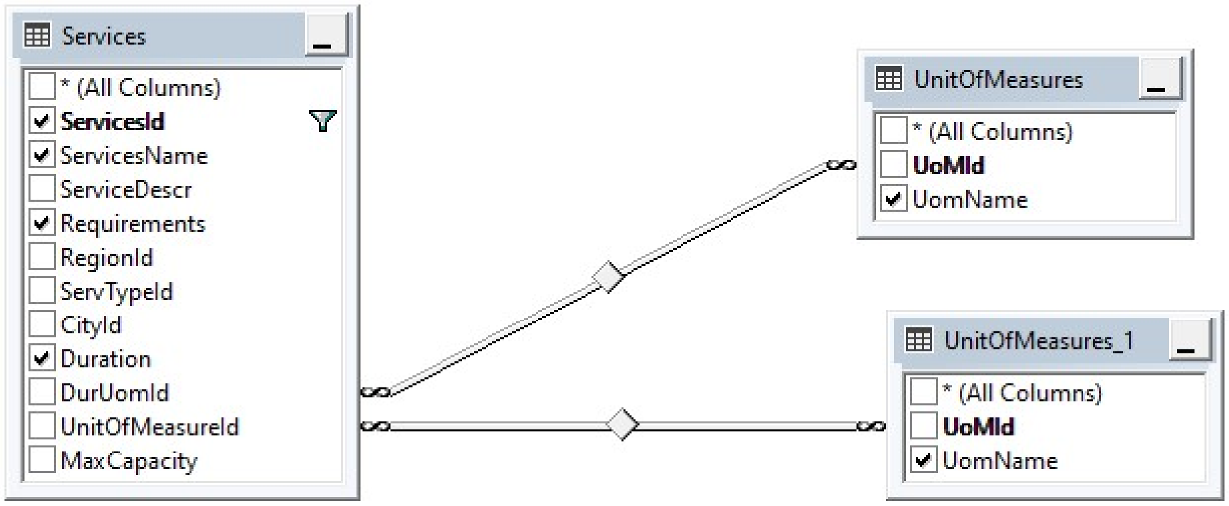
UnitOfMeasures.

To establish the unit in which the service is sold and establish the UOM for the event duration.



dbo.UnitOfMeasures
Columns
UoMId (PK, int, not null)
UomName (varchar(15), null)

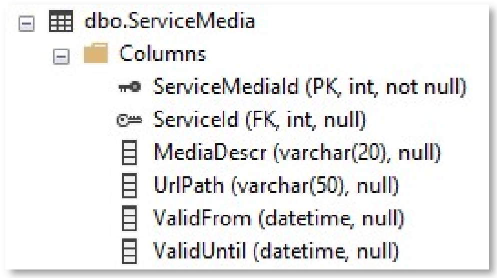
UoMId	UomName
1	each
2	hrs



ServicesId	ServicesName	Requirements	UomName	Duration	DurationUom
1	Whale Watching & Wildlife Quest	Warm clothes	each	4	hrs

ServiceMedia

To configure the URL path to all media related to the excursion or product.

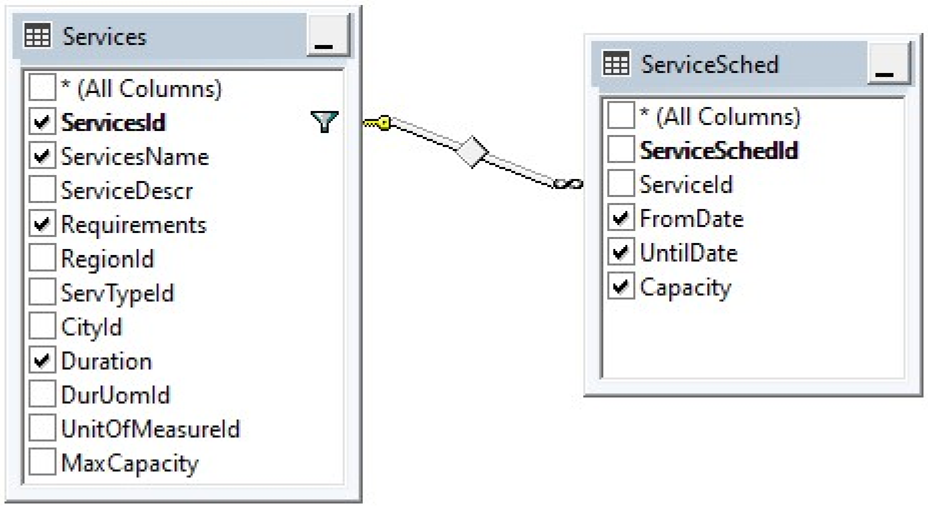


dbo.ServiceMedia
Columns
ServiceMediaId (PK, int, not null)
ServiceId (FK, int, null)
MediaDescr (varchar(20), null)
UrlPath (varchar(50), null)
ValidFrom (datetime, null)
ValidUntil (datetime, null)

ServiceMediaId	ServiceId	MediaDescr	UrlPath	ValidFrom	ValidUntil
1	1	Alaska Scenery	../Images/Alaska1.jfif	2021-01-01 00:00:00.000	2040-01-01 00:00:00.000
2	1	Alaska Scenery	../Images/Alaska2.jfif	2021-01-01 00:00:00.000	2040-01-01 00:00:00.000
3	1	Alaska Scenery	../Images/Alaska3.jfif	2021-01-01 00:00:00.000	2040-01-01 00:00:00.000

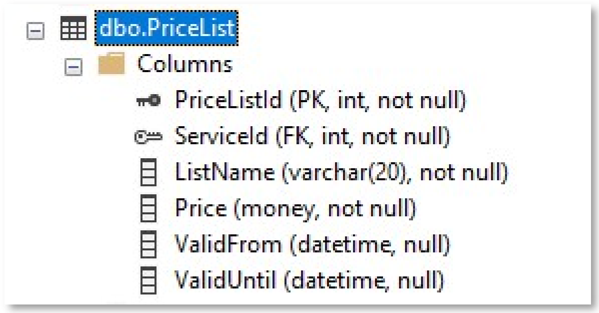
ServiceSched

To schedule the services allows setting different capacities at any given point in time. For the algorithm that calculates if there is a need to market discounts with unsold excursions, the event date is near.



PriceList

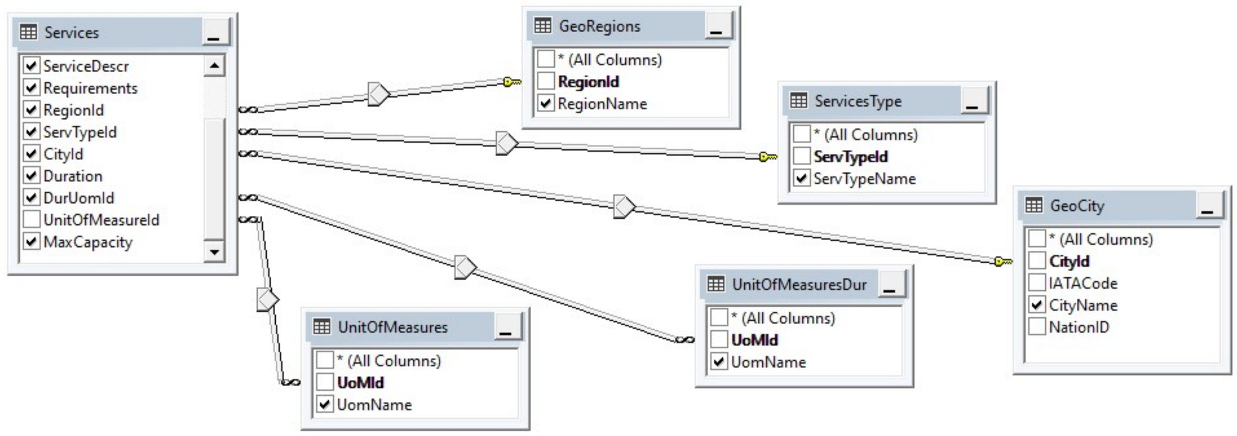
This table allows to configure several pricing schemes, and it can be used to change prices automatically for seasonal changes, to configure different levels of different prices like for adults, kids, students, or elders.



dbo.PriceList
Columns
PriceListId (PK, int, not null)
ServiceId (FK, int, not null)
ListName (varchar(20), not null)
Price (money, not null)
ValidFrom (datetime, null)
ValidUntil (datetime, null)

ServicesId	ServicesName	Requirements	Duration	FromDate	UntilDate	Capacity
1	Whale Watching	Warm clothes	4	2021-03-21 13:37:50.040	2022-03-21 13:37:50.040	50

The entire set of related objects

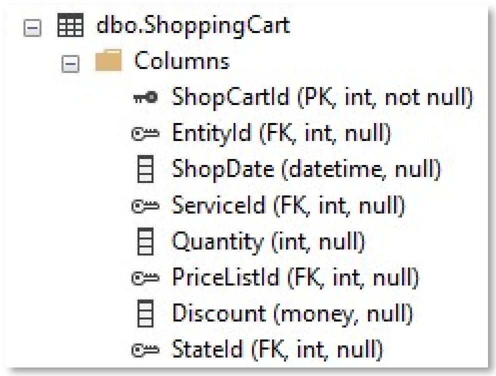


Services ordering

The application should easily allow the customer to place an order. The user adds one or more products or services into the shopping cart, which behaves like a reservations tool.

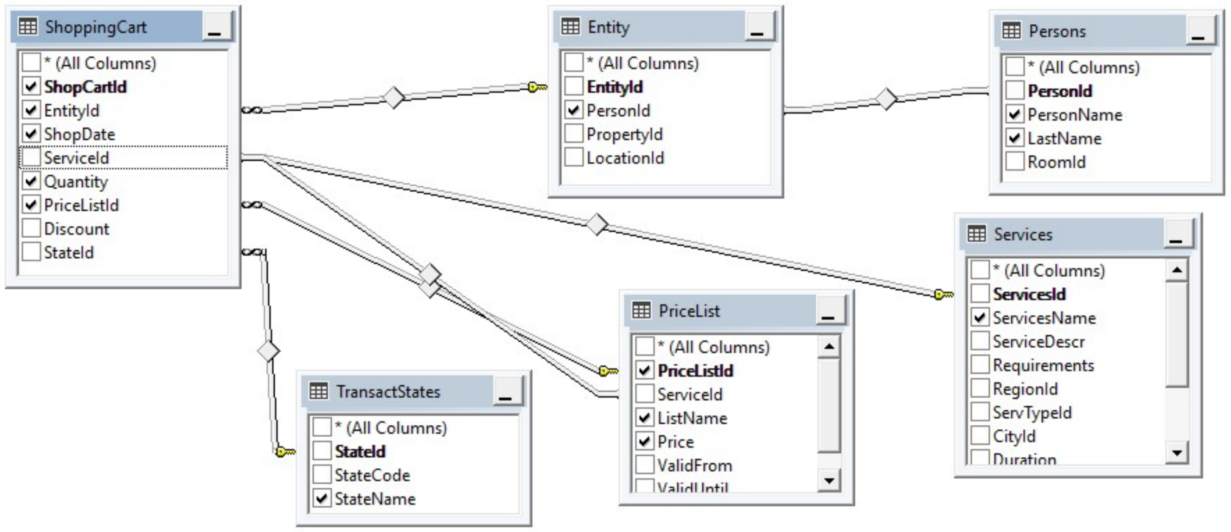
When the user proceeds to pay, the payment method is selected.

ShoppingCart



dbo.ShoppingCart
- Columns
  - ShopCartId (PK, int, not null)
  - EntityId (FK, int, null)
  - ShopDate (datetime, null)
  - ServiceId (FK, int, null)
  - Quantity (int, null)
  - PriceListId (FK, int, null)
  - Discount (money, null)
  - StateId (FK, int, null)

This diagram shows the Shopping cart with all related tables.



Not all possible fields are selected in this case just to have a small result set for illustration purposes.

This illustration shows a limited set of the shopping cart data, because of space limitation:

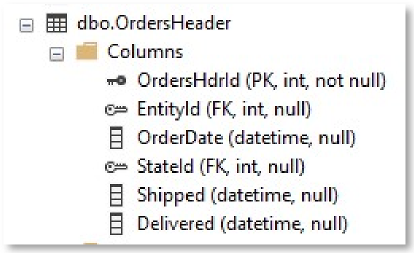
ShopCartId	PersonId	PersonName	LastName	ShopDate	ServicesName	Quantity	LastName	Price	StateName
1	1	Peter	Smith	2021-03-22 02:13:07.947	Whale Watching & Wildlife Quest	2	Adult	50.00	Active in cart

When the user deletes a record from the shipping cart, that will be controlled in the dB by changing the record state.

When the user process to place the order, then the application will proceed to:

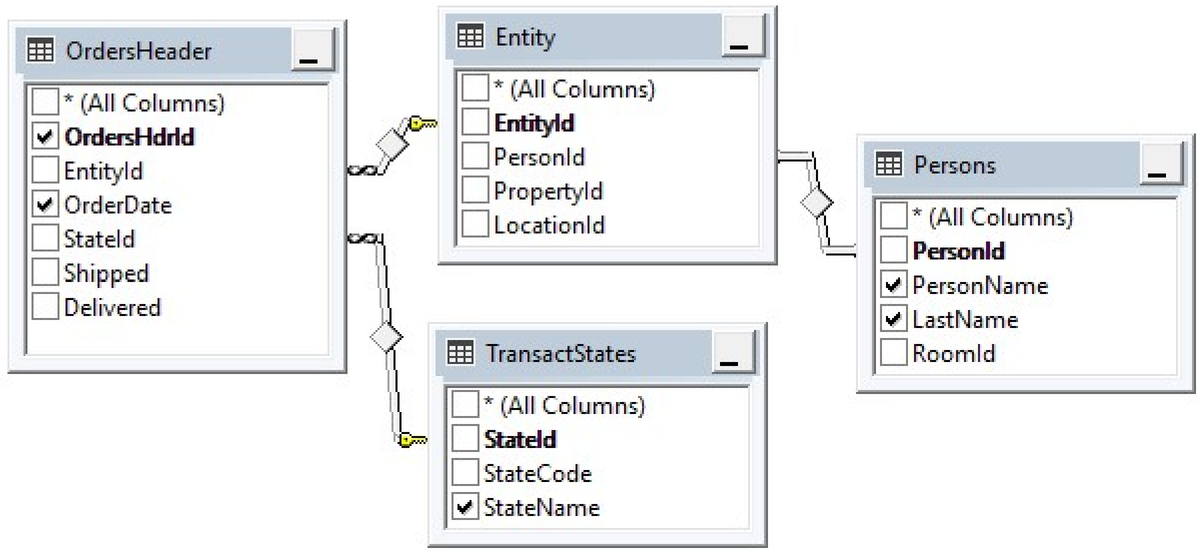
Create an order record in OrdersHeader and add all the items from the cart into the Orders table, and the status of the records in the cart will be changed to Ordered.

OrdersHeader



dbo.OrdersHeader
Columns
OrdersHdrId (PK, int, not null)
EntityId (FK, int, null)
OrderDate (datetime, null)
StateId (FK, int, null)
Shipped (datetime, null)
Delivered (datetime, null)

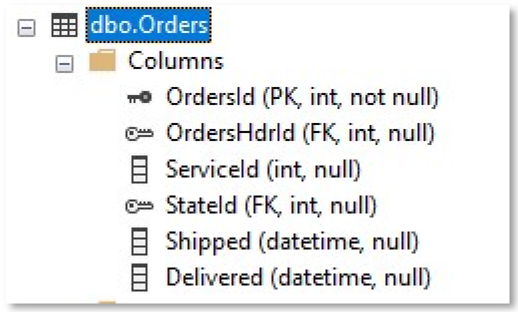
This diagram shows the order header and the corresponding child tables:



OrdersHdrId	PersonName	LastName	OrderDate	StateName
2	**Peter**	**Smith**	**2021-03-22 02:15:00.310**	**Ordered**

Orders

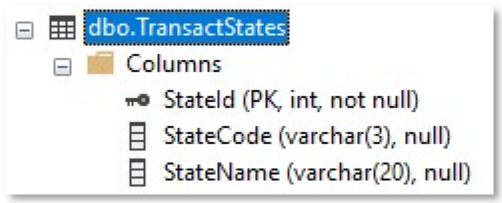
This table contains all the items for the order. The StateId exists to follow the status at the item level in partial deliveries or partial cancelations.



```
dbo.Orders
  Columns
    OrdersId (PK, int, not null)
    OrdersHdrId (FK, int, null)
    ServiceId (int, null)
    StateId (FK, int, null)
    Shipped (datetime, null)
    Delivered (datetime, null)
```

TransactStates

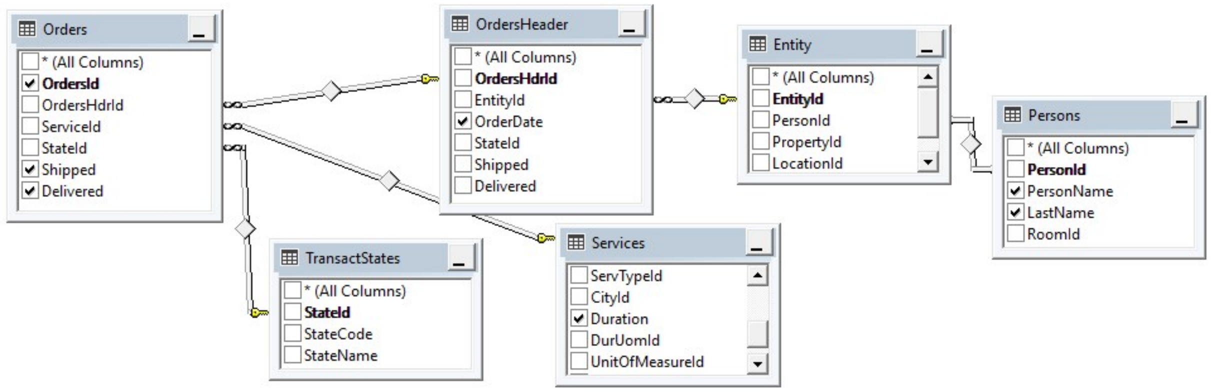
The transaction state use used at the order header and order detail levels to communicate the corresponding status.



dbo.TransactStates
Columns
StateId (PK, int, not null)
StateCode (varchar(3), null)
StateName (varchar(20), null)

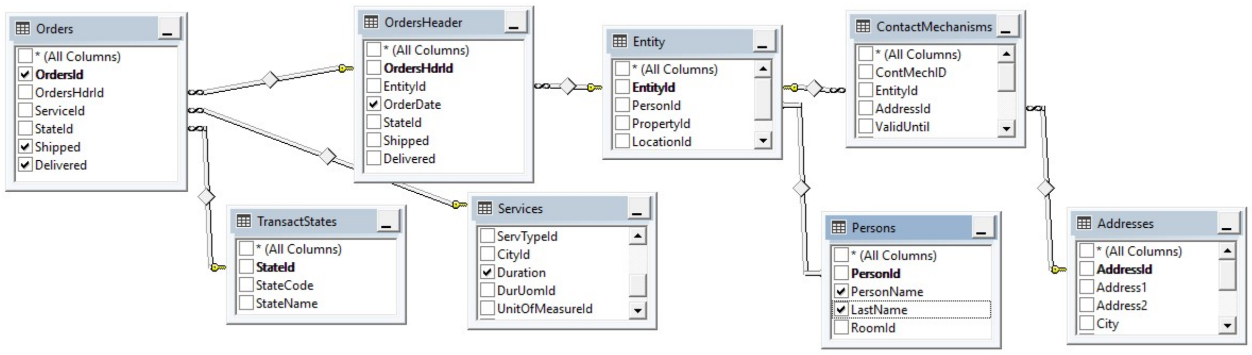
StateId	StateCode	StateName
1	ACT	Active in cart
2	DEL	Deleted from cart
3	ORD	Ordered
4	PRT	Printed from order
5	SHP	Shipped
6	DEL	Delivered
7	CAN	Cancelled from order
8	PP	Payment Pending
9	P	Paid
10	CCA	Credit Card approved
11	CCR	Credit Card rejected

This diagram shows partial data related to the Orders, and we can expand it by adding more tables like the contact mechanism to get the shipment address, for example.



OrdersId	OrderDate	Shipped	Delivered	PersonName	LastName	ServicesName	Requirements	Duration
4	2021-03-22 02:15:00.310	NULL	NULL	Peter	Smith	Whale Watching & Wildlife Quest	Warm clothes	4

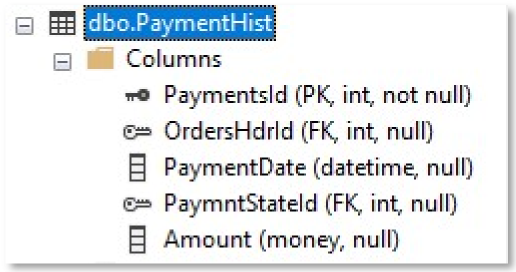
Extended Orders diagram

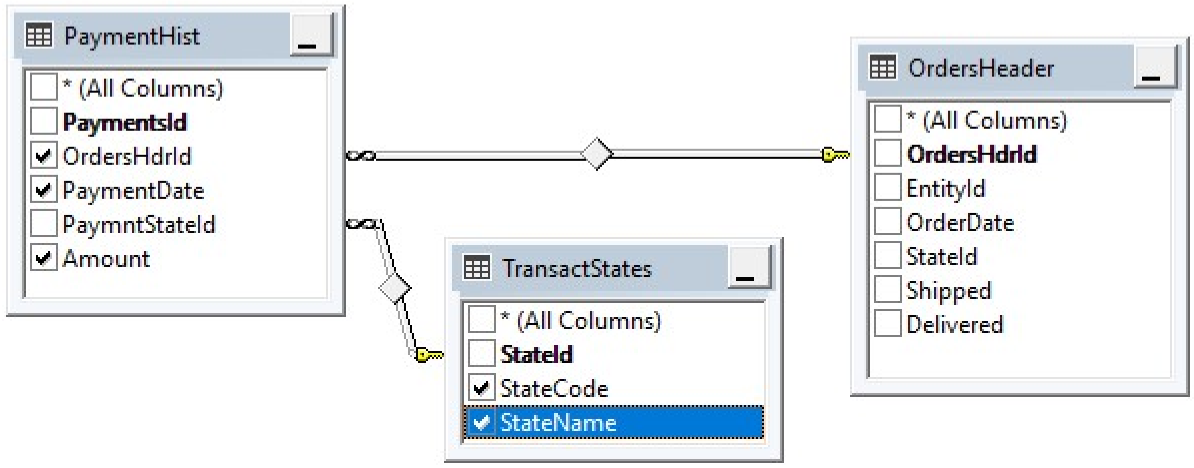


Payments and CCTransactions.

When the user proceeds to pay for the order, such actions are registered in the PaymentHistory, related to the OrderHeader. The table is also linked to the TransactStates table to decode the status of the payment.

PaymentHist


dbo.PaymentHist
Columns
PaymentsId (PK, int, not null)
OrdersHdrId (FK, int, null)
PaymentDate (datetime, null)
PaymntStateId (FK, int, null)
Amount (money, null)



OrdersHdrId	PaymentDate	Amount	StateCode	StateName
2	2021-03-22 02:20:41.113	150.00	P	Paid

The user could decide to charge it to the open tab when the payment is made with a credit card; the following could happen:

The bank rejects the charge.

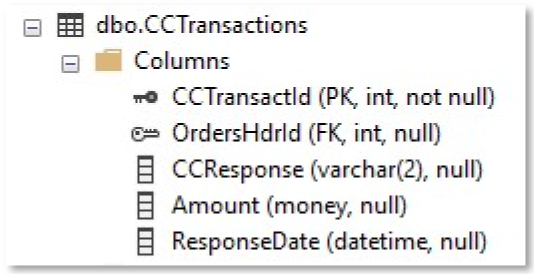
Such an event is stored in the CCTransactions table with the Credit Card rejected status. The status in the order header is also updated.

The bank approves the charge.

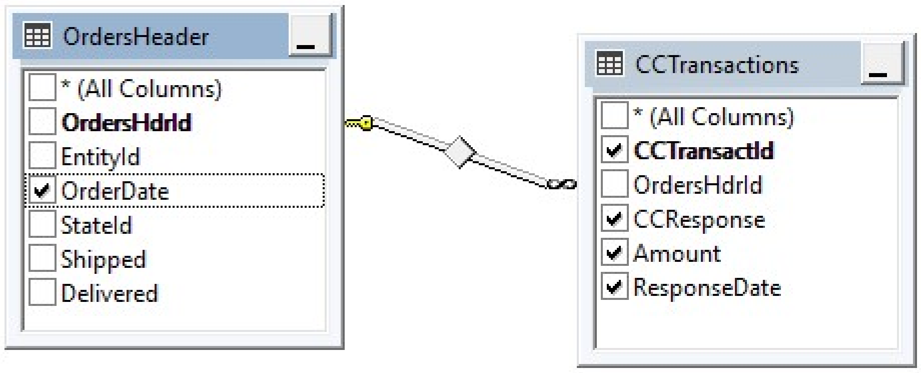
Such an event is stored in the CCTransactions table with the Credit Card rejected status. The status in the order header is also updated.

In both cases, the answer back from the bank is stored in the CCTransactions table.

CCTransactions



dbo.CCTransactions
Columns
CCTransactId (PK, int, not null)
OrdersHdrId (FK, int, null)
CCResponse (varchar(2), null)
Amount (money, null)
ResponseDate (datetime, null)



OrdersHeader
- [ ] * (All Columns)
- [ ] **OrdersHdrId**
- [ ] EntityId
- [x] OrderDate
- [ ] StateId
- [ ] Shipped
- [ ] Delivered

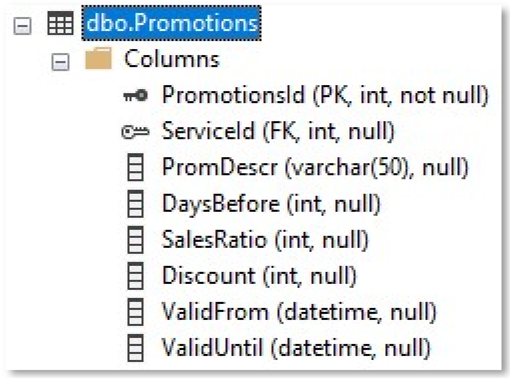
CCTransactions
- [ ] * (All Columns)
- [x] **CCTransactId**
- [ ] OrdersHdrId
- [x] CCResponse
- [x] Amount
- [x] ResponseDate

CCTransactId	OrderDate	Amount	ResponseDate	CCResponse
1	2021-03-22 02:15:00.310	50.00	2021-03-22 02:22:02.047	00

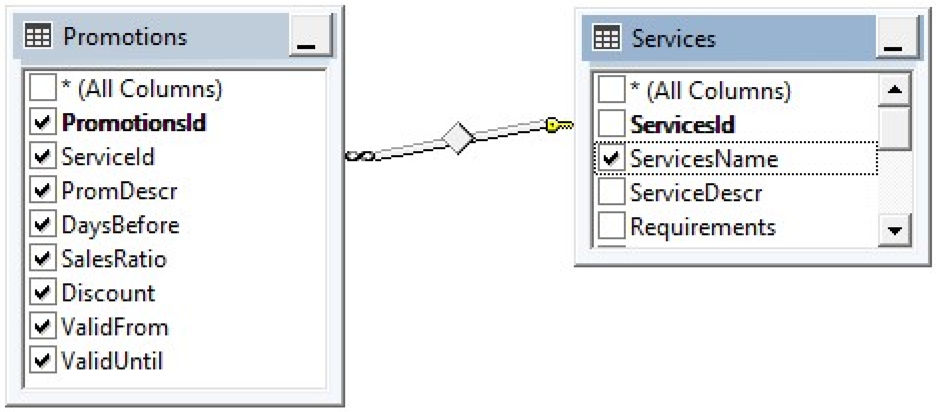
Promotions

This table allows to configure rules to be used for automated discounts of products when they are not being sold, for example, we can state the following:

When we have only two days before the service will occur, and we have sold less than 60% of the total capacity, then provide a 10 percent discount.



dbo.Promotions
Columns
PromotionsId (PK, int, not null)
ServiceId (FK, int, null)
PromDescr (varchar(50), null)
DaysBefore (int, null)
SalesRatio (int, null)
Discount (int, null)
ValidFrom (datetime, null)
ValidUntil (datetime, null)



PromotionsId	ServiceId	ServicesName	PromDescr	DaysBefore	SalesRatio	Discount	ValidFrom	ValidUntil
1	1	Whale Watching & Wildlife Quest	Rule 1	1	80	20	2021-03-22 04:36:10.063	2022-03-22 04:36:10.063
2	1	Whale Watching & Wildlife Quest	Rule 1	2	60	10	2021-03-22 04:36:10.063	2022-03-22 04:36:10.063
3	1	Whale Watching & Wildlife Quest	Rule 1	3	60	20	2021-03-22 04:36:10.063	2022-03-22 04:36:10.063