Normalization of tables

A Case Study

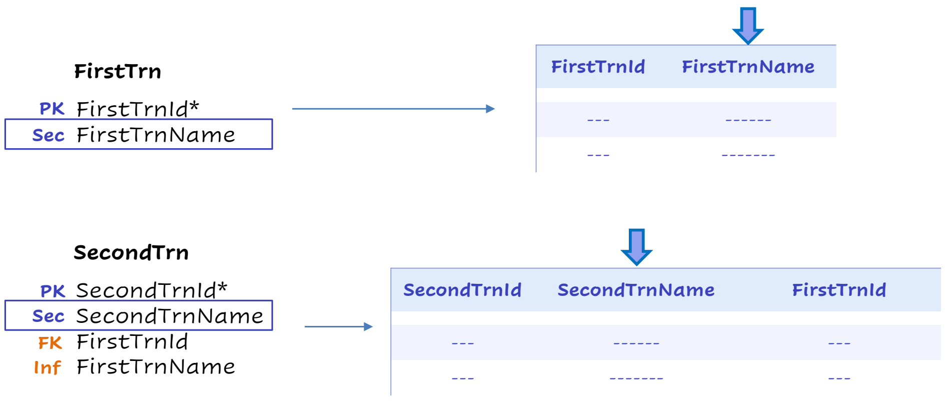
GeneXus™

Throughout the course we have been discussing how GeneXus determines the database structure from the transaction design.

We've also seen that it automatically makes referential integrity checks, and to do so it uses the primary and foreign indexes that it creates in each table.

We will now look at a case study to analyze how GeneXus normalizes the database and determines the table structure from a given transaction design.

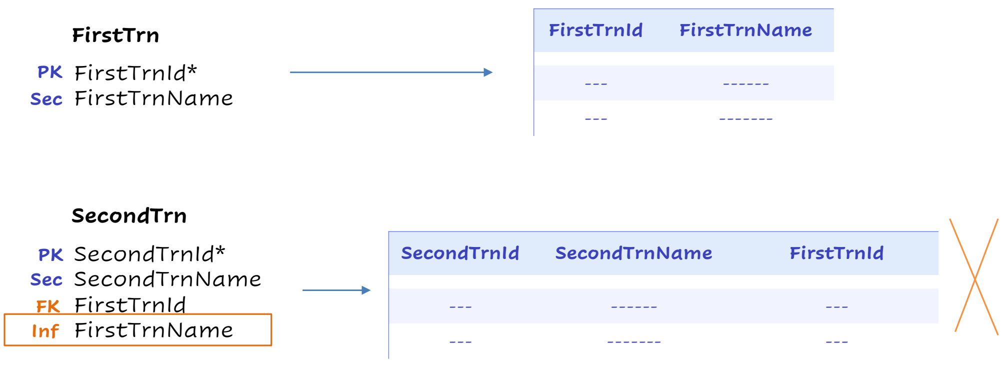
Normalized database: Concept



Let's start then with the concept of a normalized database. What does it mean?

It means there are no duplicated data, no redundancies. Secondary attributes are present in a single table...

Normalized database: Concept



FirstTrn

PK FirstTrnId*
Sec FirstTrnName

FirstTrnId	FirstTrnName
---	------
---	-------

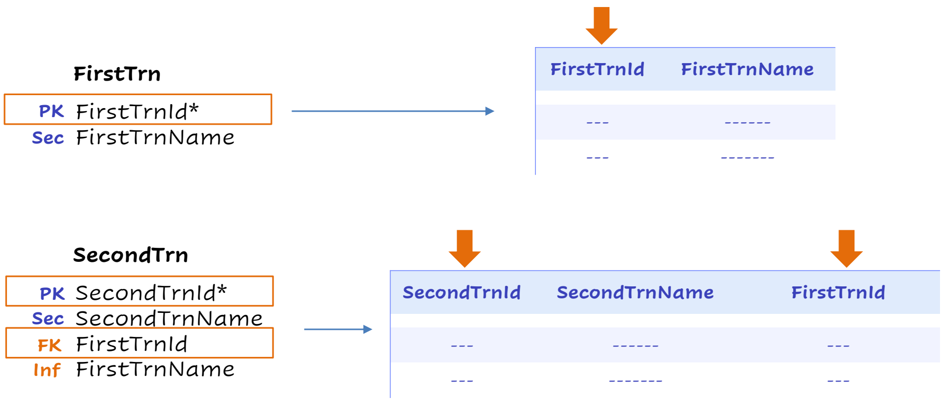
SecondTrn

PK SecondTrnId*
Sec SecondTrnName
FK FirstTrnId
Inf FirstTrnName

SecondTrnId	SecondTrnName	FirstTrnId
---	------	---
---	-------	---

…inferred attributes are not stored…

Normalized database: Concept



..and the only attributes that can be included in more than one table are primary keys, since they are also foreign keys in other tables.

Case Study: Transaction design

```
Country
CountryId* (PK)
CountryName (Sec)

TouristGuide
TouristGuideId* (PK)
TouristGuideName (Sec)
```

```
Trip
TripId*
TripDate
TripPrice
TouristGuideId
TouristGuideName
CountryId
CountryName
```

```
Customer
CustomerId*
CustomerName
CustomerTripsQty
Trip
(
   TripId*
   TripDate
   TripPrice
   CountryId
   CountryName
   CustomerTripMiles
)
```

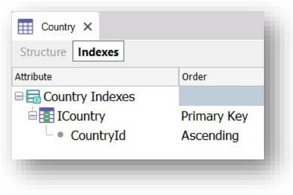
Let's look at the case study then. Consider the following transaction design, where you can see that we are modeling Countries, Trips or tours to a certain country with a Tour Guide in charge, and the Clients who can take several of these trips or tours.

The first thing to remember and keep in mind is that we cannot look at each transaction independently but must analyze the entire data model. Each transaction generates an impact on that data model.

Note that Country and TouristGuide transactions are simple, single-level transactions, each with its own identifier and name, and without any foreign keys. Each one then has its primary key and a secondary attribute.

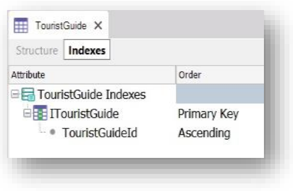
Case Study: Normalized tables

COUNTRY
CountryId*
CountryName



Attribute	Order
Country Indexes	
ICountry	Primary Key
CountryId	Ascending

TOURISTGUIDE
TouristGuideId*
TouristGuideName



Attribute	Order
TouristGuide Indexes	
ITouristGuide	Primary Key
TouristGuideId	Ascending

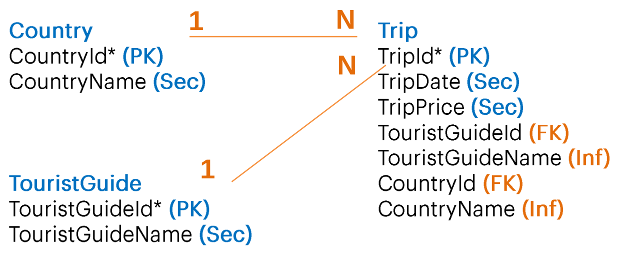
Therefore, we already know that from them GeneXus will create the tables COUNTRY and TOURISTGUIDE with the structure shown here:

The COUNTRY table has CountryId as the primary key and CountryName as the secondary attribute. Over this table then, GeneXus will create the primary index by CountryId.

Remember that GeneXus automatically creates the primary indexes to control the uniqueness of the primary key and to efficiently perform the reference integrity checks.

The TOURISTGUIDE table has TouristGuideId as primary key and TouristGuideName as secondary attribute. In this table, GeneXus will create the corresponding primary index by the TouristGuideId attribute.

Case Study: Transaction design



Country
CountryId* (PK)
CountryName (Sec)

TouristGuide
TouristGuideId* (PK)
TouristGuideName (Sec)

Trip
TripId* (PK)
TripDate (Sec)
TripPrice (Sec)
TouristGuideId (FK)
TouristGuideName (Inf)
CountryId (FK)
CountryName (Inf)

```
Customer
CustomerId*
CustomerName
CustomerTripsQty
Trip
(
    TripId*
    TripDate
    TripPrice
    CountryId
    CountryName
    CustomerTripMiles
)
```

Let’s take a look at the Trip transaction: It has TripId as a primary attribute, and TripDate and TripPrice as secondary attributes. Then TouristGuideId is the foreign key in this transaction. And TouristGuideName is inferred from that value. There is a 1-N relationship between TouristGuide and Trip.

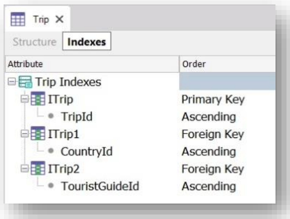
Something similar happens with the CountryId attribute which is a foreign key here, and CountryName which is inferred from that value. There is also a 1-N relationship between Country and Trip.

Case Study: Normalized tables

COUNTRY
CountryId*
CountryName

TOURISTGUIDE
TouristGuideId*
TouristGuideName

TRIP
TripId*
TripDate
TripPrice
TouristGuideId
CountryId



Attribute	Order
Trip Indexes	
ITrip	Primary Key
TripId	Ascending
ITrip1	Foreign Key
CountryId	Ascending
ITrip2	Foreign Key
TouristGuideId	Ascending

We already know that the inferred attributes are not stored because GeneXus gets their value from the corresponding foreign keys; so, what will be the structure of the table associated with the Trip transaction?

Its primary key TripId, and the attributes TripDate, TripPrice, TouristGuideId and CountryId.

If we now think about the indexes, which indexes will be created by GeneXus over the TRIP table?

The primary index by TripId, and the foreign indexes by CountryId and TouristGuideId

This means that, for example, when inserting a trip, the primary index by TripId will control the uniqueness of its value, i.e. that there is no longer another trip with the same primary key value.
The primary index by TouristGuideId in the TOURISTGUIDE table will allow performing the referential integrity checks by making sure that the value of the foreign key TouristGuideId indicated in Trip previously exists as primary key in the TOURISTGUIDE table.

The same control will be made by the primary index by CountryId defined in the COUNTRY table. It will check that the value indicated here previously exists as the primary key in COUNTRY.

On the other hand, if we try for example to remove a tour guide through the TouristGuide transaction, the foreign index by TouristGuideId defined in Trip will check that there is no tour registered with this TouristGuideId value.

If there is one, GeneXus will warn you that there are records in Trip that have in TouristGuideId the value you are trying to delete, and it will not allow deleting it.

Case Study: Transaction design

Country
CountryId* **(PK)**
CountryName **(Sec)**

TouristGuide
TouristGuideId* **(PK)**
TouristGuideName **(Sec)**

Trip
TripId* **(PK)**
TripDate **(Sec)**
TripPrice **(Sec)**
TouristGuideId **(FK)**
TouristGuideName **(Inf)**
CountryId **(FK)**
CountryName **(Inf)**

Customer
CustomerId* **(PK)**
CustomerName **(Sec)**
CustomerTripsQty **(GLOBAL FORMULA)**
Trip
(
    TripId* **(PK)** - **(FK)**
    TripDate **(Inf)**
    TripPrice **(Inf)**
    CountryId **(Inf)**
    CountryName **(Inf)**
    CustomerTripMiles **(Sec)**
)

Now let's look at the Customer transaction. It is a two-level transaction, with TripId as primary key of the second level. This indicates an N-N relationship between Customer and Trip. And we already know that from this transaction design GeneXus will create two tables: CUSTOMER and CUSTOMERTRIP.

But let's focus on the first level of the transaction.

Let's focus on the second level now. TripId is its primary key, but it is also a foreign key, and we see TripDate, TripPrice, CountryId and CountryName.

If we look again at the structure of the Trip transaction, these attributes are present there; therefore, all of them are obtained –inferred–, from the value of their primary key TripId.

This means that now in the second level of Customer, these attributes will be inferred by the TripId value. So, CountryId was a direct foreign key, stored in Trip, and now is an inferred foreign key on this second level of Customer.

Case Study: Normalized tables

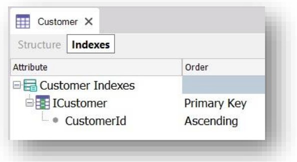
COUNTRY
CountryId*
CountryName

TRIP
TripId*
TripDate
TripPrice
TouristGuideId
CountryId

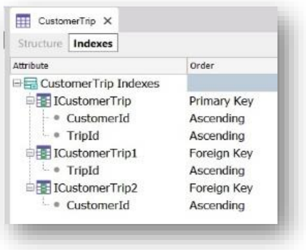
CUSTOMER
CustomerId*
CustomerName

TOURISTGUIDE
TouristGuideId*
TouristGuideName

CUSTOMERTRIP
CustomerId*
TripId*
CustomerTripMiles



Attribute	Order
Customer Indexes	
ICustomer	Primary Key
CustomerId	Ascending



Attribute	Order
CustomerTrip Indexes	
ICustomerTrip	Primary Key
CustomerId	Ascending
TripId	Ascending
ICustomerTrip1	Foreign Key
TripId	Ascending
ICustomerTrip2	Foreign Key
CustomerId	Ascending

How does the structure of the tables based on the Customer transaction look?

The CUSTOMER table associated with the first level, with CustomerId as the primary key and CustomerName. Remember that the attribute CustomerTripsQty, when calculated, is not saved in the associated table. And then the CUSTOMERTRIP table associated with the second level of the Customer transaction, with the pair CustomerId, TripId as primary key and the secondary attribute CustomerTripMiles.

As for the indexes, in CUSTOMER the primary index by CustomerId will be created.
In CUSTOMERTRIP, its corresponding primary index will be created by the attribute pair CustomerId, TripId, and then the corresponding foreign indexes by TripId and CustomerId.

Note that no index is created on this table by CountryId, because as explained before, in this table it is not a direct foreign key but inferred through the value of TripId.

Finally…

Transaction design

Country
CountryId*
CountryName

TouristGuide
TouristGuideId*
TouristGuideName

Trip
TripId*
TripDate
TripPrice
TouristGuideId
TouristGuideName
CountryId
CountryName

Customer
CustomerId*
CustomerName
CustomerTripsQty – Count(TripId)
Trip
(
    TripId*
    TripDate
    TripPrice
    CountryId
    CountryName
    CustomerTripMiles
)

COUNTRY
CountryId*
CountryName

TOURISTGUIDE
TouristGuideId*
TouristGuideName

TRIP
TripId*
TripDate
TripPrice
TouristGuideId
CountryId

CUSTOMER
CustomerId*
CustomerName

CUSTOMERTRIP
CustomerId*
TripId*
CustomerTripMiles

Normalized tables.

In this way then, we have analyzed that from this transaction design, GeneXus creates this normalized structure in the database.

GeneXus™

training.genexus.com
wiki.genexus.com
training.genexus.com/certifications