

Using Forecasts within Orix

- Not all Apps need the facilities of forecasting, so it is not included in the basic framework, but is added using the standard Orix mechanism: A data-table named "Forecasts" is added as a BusinessObject, and once present this triggers the addition of Forecasting related features in the App.
- Using Forecasting is technically complex to understand to begin with, but once understood and implemented it can be used by ordinary users of an App without further intervention from the Developer.

How Forecasts Work

The Forecasts data-table follows a standard pattern in Orix, using a "LinkTable" and "LinkID" field. This means that you can store values in the Forecasts data-table, and link them either to a whole data-table (via the LinkTable) or to a specific record (via the LinkID).

You add the new Forecasts BusinessObject using a SQL Script. This script is shown at the end of this page. Once it has been added you enable any other BusinessObject in your App to create linked Forecasts records by adding the **DESCRIPTION** AddForecasts=1 to its data-definition, as shown in the SQL below.

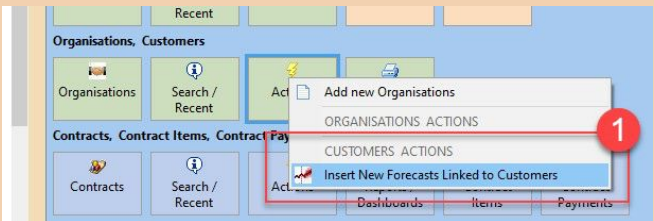
```
CREATE TABLE "Customers"
(
  "ID" INTEGER NOT NULL,

  [OTHER FIELDS AND CONSTRAINT DEFINITIONS WOULD FOLLOW HERE]
)
DESCRIPTION '[Properties]
AddForecasts=1'
```

You can also **ALTER** a data-table to enable Forecasts:

```
ALTER TABLE "Customers" DESCRIPTION '[Properties]
AddForecasts=1'
```

Once this Description is added new features will be visible in Orix.

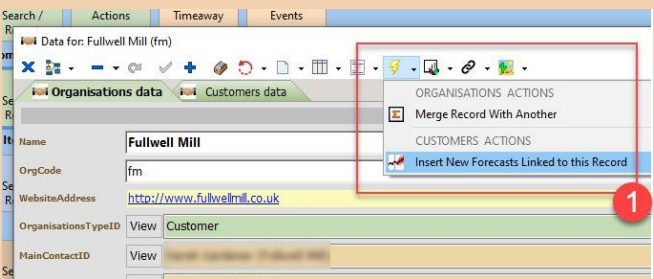


Entity Insert Forecasts

If the above change is made, on the System Entities Screen a new Action "Insert New Forecasts Linked to Customers" will be added, as shown at 1.

Adding Forecasts here sets the LinkTable of the Forecasts data, but leaves the LinkID **blank**. This allows data entered to be returned and compared to **sum** data for the LinkTable.

As an example, "Monthly Sales Forecasts" could be added, and compared to the **sum** of values in a Sales data-table.



Record Insert Forecasts

Also, on the individual Records a new Action is created, as shown at 1.

Adding Forecasts here sets both the LinkTable and LinkID for the Forecasts record. This allows data to be queried and returned **per customer**. Or per any other data-record in any data-table (Products, Projects, People etc.)

Both the above actions open a **Forecasts data-entry Grid** (shown below), which allows the user to add multiple records to the Forecasts data-table.

Forecast Batch Entry Screen

Close Insert New Forecasts Activate Undo

Add forecasts for table: Expenses

Name: Annual Targets Quantity: 0 Start: 0 End: 0

DateStart: 16/03/2021 RecordsToInsert: 12 Value: 500 600

PeriodType: Monthly

Forecasts batch entry. Enter values in the above fields, and Activate. The system will use the data you have added to create a set of data which you can manually edit and update. Click "Post" to actually insert these rows into the database.

Forecasts batch-entry grid

1. Give your set of Forecasts a **name**. This will allow you to select them in SQL scripts later.
2. Set a start date for the first Forecast.
3. Choose whether the Forecast should be Weekly, Monthly or Annual.
4. Enter the number of records, records will be added with a start-date separated by the number of days determined by the chosen "PeriodType".
5. Enter Quantity and Value amounts for the start of the period and end of the period.

Forecast Batch Entry Screen

Close Insert New Forecasts Activate Undo

Add forecasts for table: Expenses

Name: Annual Targets Quantity: 0 Start: 0 End: 0

DateStart: 16/03/2021 RecordsToInsert: 12 Value: 500 600

PeriodType: Monthly

Forecasts batch entry. Enter values in the above fields, and Activate. The system will use the data you have added to create a set of data which you can manually edit and update. Click "Post" to actually insert these rows into the database.

ID	DateStart	Name	Value	Quantity
	16/03/2021	Annual Targets	500	0
	16/04/2021	Annual Targets	509.09	0
	16/05/2021	Annual Targets	518.18	0
	16/06/2021	Annual Targets	527.27	0
	16/07/2021	Annual Targets	536.36	0
	16/08/2021	Annual Targets	545.45	0
	16/09/2021	Annual Targets	554.55	0
	16/10/2021	Annual Targets	563.64	0
	16/11/2021	Annual Targets	572.73	0
	16/12/2021	Annual Targets	581.82	0
	16/01/2022	Annual Targets	590.91	0
*	16/02/2022	Annual Targets	600	0

12 6,000 0

12 of 12

Forecasts batch-entry grid 02

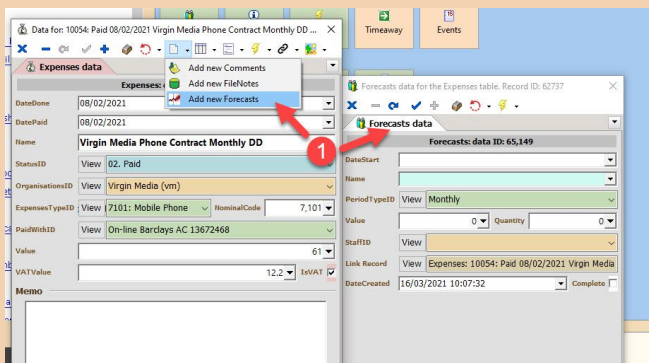
1. Once you have completed setting the values and range for your Forecasts, click **Activate** and a data-grid will open showing the added records.
2. Note that the values across the period will be automatically computed to vary between your start and end value.
3. Once you are happy with the values in the Grid, click **Insert New Forecasts** and data will be added to the Forecasts data-table.

Note: Once the Grid is activated, it is not fixed. You can edit the values to change them to meet your needs. For example you may plan for Sales to grow over the year, but you may expect one month to be below average, due to seasonal issues or an annual holiday. In such cases you can manually correct individual values of the data before you click "Insert"

Adding Individual Forecasts Records

You may not need to add large numbers of Forecasts records at the same time. For example you might just want to set a "Sales Target" for a Customer or Sales-person.

In this case you do not need to use the Insert Forecasts **Action**. Instead, just add individual Forecasts records.



Forecasts single record entry

From any data-record in a BusinessObject that has Forecasts enabled, click the "New" button, then enter Forecasts data, as Shown at 1.

A screenshot of a 'Forecasts data' form for the 'Customers table'. The form has a title bar 'Forecasts data for the Customers table. Record ID: 11472'. Below the title bar is a toolbar with icons for save, delete, add, and other actions. The main form area has fields for 'DateStart', 'Name', 'PeriodTypeID' (set to 'Monthly'), 'Value' (0), 'Quantity' (0), 'StaffID', 'Link Record' (Customers: Fullwell Mill (fm)), and 'DateCreated' (08/04/2021 11:30:26). There is a 'Duplicate this record' button in the top right corner.

Forecasts Edit Form

The Forecasts Edit Form, shown on the left, contains fields which can be used to set values and quantities over time, linked to a Table or Record.

Users (provided they have adequate security) can add data in these forms.

Note that the Forecasts Edit Window has a built in Action "Duplicate this record", allowing users to easily create multiple records in quick succession.

Using Forecasts data in a Dashboard, Resource, Cube or Chart

Entering Forecasts is simple. Writing the SQL Scripts which will create visualizations of your Forecasts is a bit more technical. Usually the SQL will be written by an Orixia Developer and added as a record to the "Resources" BusinessObject. Once the SQL is complete, any user can open the dashboard and see the result in a location in your App set by fields in the Resources data-record.

Resources data

Resources: data ID: 64,412

LocationID View Entity ComponentID View Chart

LinkTable View WorkItems TargetTable View WorkItems

Name Forecast v Actual WorkItems SecurityLevel 0

Description

SQLStr Description

```

1 SELECT
2   FC.YYMM,
3   FC.Target,
4   WI.Actual,
5   ROUND((WI.Actual - FC.Target) / FC.Target * 100, 2) as "Percentage"
6 FROM
7   (SELECT
8     YearMonth(CAST(DateStart as DATE)) as YYMM,
9     SUM(Quantity) as Target
10    FROM Forecasts
11    WHERE LinkTable = 'WorkItems'
12         AND Name = 'Paid Hours'
13         AND DateStart >= CAST([MinDate] as DATE)
14         AND DateStart <= CAST([MaxDate] as DATE)
15    GROUP BY YYMM) as FC
16 LEFT JOIN
17   (SELECT
18     YearMonth(W.DateDone) as YYMM,
19     SUM(W.HoursWorked) as Actual
20    FROM WorkItems W
21    LEFT JOIN ContractItems C ON C.ID = W.ContractItemsID
22    WHERE C.Chargeable = true
23         AND W.DateDone >= CAST([MinDate] as DATE)
24         AND W.DateDone <= CAST([MaxDate] as DATE)
25    GROUP BY YYMM) as WI ON WI.YYMM = FC.YYMM
26
27

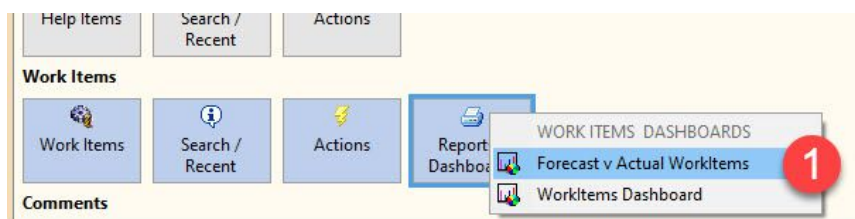
```

Creating a Forecast Resource

In the above example, an App includes features to monitor **Hourly Billing** by staff. Details of this billing are stored in a **WorkItems** data-table. Forecasts have been added with LinkTable = WorkItems and Name = Paid Hours.

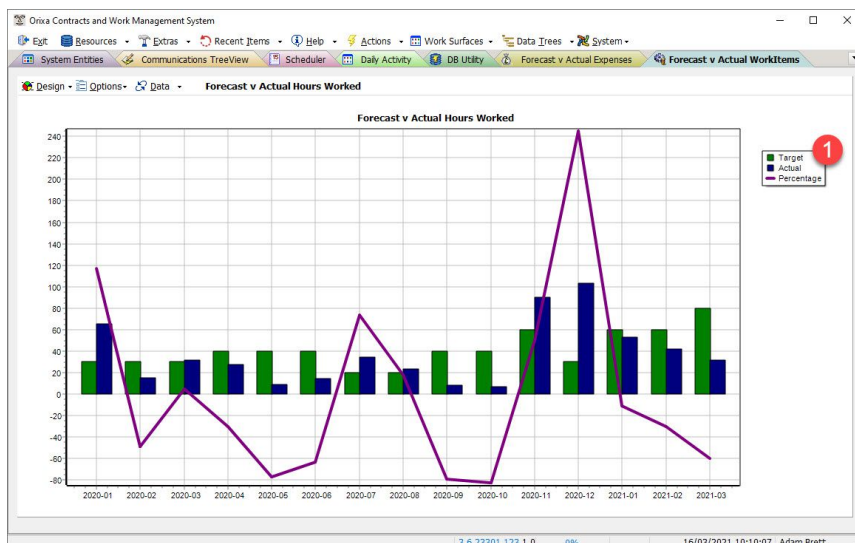
Note how the above SQL relies on **sub-select statements**. That is the data is first returned and **summed** for Forecasts and WorkItems, then this data is pulled together to create a result-set.

The Resource has been created with "ComponentID" = Chart, so a chart will be shown when the Resource is activated.



Accessing Forecast Dashboard

The above image shows how the newly added Forecast Resource shows in the App: Accessible via the "Reports / Dashboards" for the "WorkItems" BusinessObject.



Forecast Dashboard

The above image shows the resulting chart, with "Target" in green, "Actual" in blue, and over / under performance shown as a purple line, marked 1.

Adding forecasting features to your Orixia App involves several stages:

1. Run the SQL script needed to create the Forecasts BusinessObject in your App. Sample SQL doing this is added at the end of this article.
2. Link chosen BusinessObjects in your App to the Forecasting system by adding the correct DESCRIPTION to their data-definition.
3. Add Forecasts records with your target values and quantities, linked to specific BusinessObjects and data-records.
4. Write SQL to query the Orixia database linking the Forecasts to actual business data, and add this as a new "Resources" record.
5. Run the resource and view the resulting data. If the Resource is a Grid it will appear fully formed. If it is a Cube or Chart the user will need to design it to see the final result.

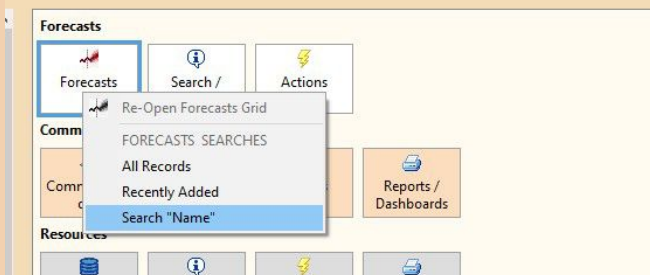
While setting up a Forecasts system is complex, once complete further work will be minimal. Users can add Forecasts for a period, and see the resulting data immediately in linked Resources.

Forecasts BusinessObject

It is important to remember that Forecasts are just an ordinary BusinessObject like any other BusinessObject in Orixia.

This means once Forecasts data has been added it can be viewed, edited, updated or deleted by users exactly like any other records in the system.

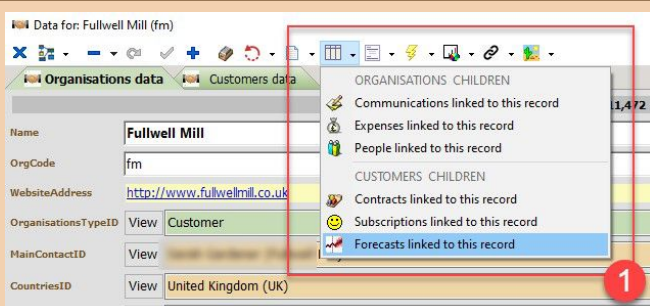
Linked Forecasts can also be seen in a grid directly from records they are connected to.



Forecasts Entity

Once Forecasts have been added users (with the correct Security Level) can access a data Grid from which they can search, edit and update records.

It is possible to add Resources linked to Forecasts, so you can show specialized visualizations of Forecasts data if you want to.



View Linked Forecasts

If a BusinessObject is linked to Forecasts, then an item is added to the list of Grids in that BusinessObject's Edit Window (shown at 1. on the left) allowing users to see linked Forecasts records, to allow updating.

Example SQL For Forecasts Resources

The following SQL will return data-sets which can be shown in your Orixia App bringing together Forecasts for a period with data in your App.

```
SELECT
  FC.YYMM,
  FC.Target,
```

```

    WI.Actual,
    ROUND((WI.Actual - FC.Target) / FC.Target * 100, 2) as "Percentage"
FROM
    (SELECT
    YearMonth(CAST(DateStart as DATE)) as YYMM,
    SUM(Quantity) as Target
    FROM Forecasts
    WHERE LinkTable = 'WorkItems'
        AND Name = 'Paid Hours'
        AND DateStart >= CAST([MinDate] as DATE)
        AND DateStart <= CAST([MaxDate] as DATE)
    GROUP BY YYMM) as FC
LEFT JOIN
    (SELECT
    YearMonth(W.DateDone) as YYMM,
    SUM(W.HoursWorked) as Actual
    FROM WorkItems W
    LEFT JOIN ContractItems C ON C.ID = W.ContractItemsID
    WHERE C.Chargeable = true
        AND W.DateDone >= CAST([MinDate] as DATE)
        AND W.DateDone <= CAST([MaxDate] as DATE)
    GROUP BY YYMM) as WI ON WI.YYMM = FC.YYMM

```

The above SQL generates a "Target" field, from a SUM of the Quantity field of Forecasts, and an "Actual" field from a SUM of the "HoursWorked" field of the WorkItems data-table.

Only Forecasts with a LinkTable = WorkItems and Name = Paid Hours are returned, so other Forecasts are not accidentally added to the totals.

Note how the SQL uses the YearMonth Function to create a field "YYMM", and how this Period-field is used to JOIN the two sets of data, so that Target and Actual for each YYMM appear in the same row of the result-set.

```

SELECT
    FC.YYMM,
    FC.Target,
    COALESCE(E.Actual, 0) as Actual,
    ROUND((E.Actual - FC.Target) / FC.Target * 100, 2) as "Percentage"
FROM
    (SELECT
    YearMonth(DateStart) as YYMM,
    SUM(Value) as Target
    FROM Forecasts
    WHERE LinkTable = 'Expenses'
        AND Name = 'Monthly Expenses'
        AND DateStart >= CAST([MinDate] as DATE)
        AND DateStart <= CAST([MaxDate] as DATE)
    GROUP BY YYMM) as FC
LEFT JOIN
    (SELECT
    YearMonth(DateDone) as YYMM,
    COALESCE(SUM(Value), 0.00) as Actual
    FROM Expenses
    WHERE DateDone >= CAST([MinDate] as DATE)
        AND DateDone <= CAST([MaxDate] as DATE)
        AND NOT ExpensesTypeID = 56675
    GROUP BY YYMM) as E ON E.YYMM = FC.YYMM

```

The above SQL generates a "Target" field from a SUM of the Value field of the Forecasts table, and an "Actual" field from the SUM of Monthly expenses. In this case (as with the previous example) the result is totalled per month, but it would also be possible to GROUP BY other fields, such as by StaffID to show totals for each staff member.

```

SELECT
    FC."Year",
    W.Customer,
    FC.Target,
    COALESCE(W.Actual, 0) as Actual,
    COALESCE(W.Planned, 0) as Planned,

```

```

ROUND((W.Actual - FC.Target) / FC.Target * 100, 2) as "PercentageActual",
ROUND((W.Planned - FC.Target) / FC.Target * 100, 2) as "PercentagePlanned"
FROM
(SELECT
EXTRACT(Year FROM DateStart) as "Year",
LinkID,
SUM(Value) as Target
FROM Forecasts
WHERE LinkTable = 'Customers'
AND Name = 'Customer Annual'
AND DateStart >= CAST([MinDate] as DATE)
AND DateStart <= CAST([MaxDate] as DATE)
GROUP BY "Year", LinkID) as FC
LEFT JOIN
(SELECT
EXTRACT(Year FROM DateDone) as "Year",
O.ID as LinkID,
O.FullName as Customer,
COALESCE(SUM(WI.HoursPlanned * 60), 0.00) as Planned,
COALESCE(SUM(Value), 0.00) as Actual
FROM WorkItems WI
LEFT JOIN Contracts C ON C.ID = WI.ContractsID
LEFT JOIN Organisations O ON O.ID = C.CustomersID
WHERE DateDone >= CAST([MinDate] as DATE)
AND DateDone <= CAST([MaxDate] as DATE)
GROUP BY Year, LinkID) as W ON ((W.Year = FC.Year)
AND (W.LinkID = FC.LinkID))

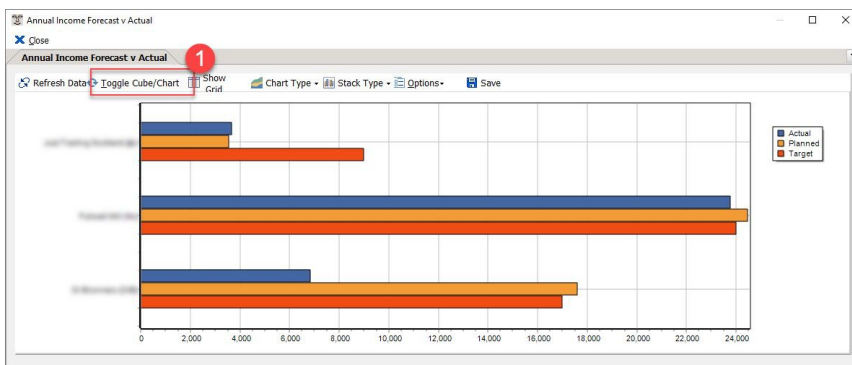
```

The above SQL generates 2 grouping Columns, and is intended to show results in a **Cube**, rather than a chart.

Two sub-select statements return columns for Year, LinkID and Target / Actual. The two grouping Columns are then used in the JOIN Statement. Note that this Forecast also uses the "HoursPlanned" and "Value" columns, allowing Forecasts to be compared with actual planned work which staff have entered into the App, but which has not yet been completed.

Customer	2020			2021		
	Actual	Planned	Target	Actual	Planned	Target
Grand total	34,270.20	45,625.20	50,000.00	23,500.20	23,500.20	16,000.00
Customer A	6,823.80	17,608.80	17,000.00	3,448.80	3,568.80	2,000.00
Customer B	23,771.40	24,461.40	24,000.00	17,726.40	17,696.40	12,000.00
Customer C	3,675.00	3,555.00	9,000.00	2,325.00	2,235.00	2,000.00

Decision Cube output from example SQL



Decision Cube, toggled to show as a Chart

Note that Decision-Cubes can also display chart data.

Forecasts data-table structure

```

1 CREATE TABLE "Forecasts"
2 (
3   "ID" INTEGER DEFAULT UID() NOT NULL,
4   "DateStart" DATE,
5   "Name" VARCHAR(120) COLLATE "ANSI" DESCRIPTION '[Properties]
6   ReuseLast=1
7   Distinct=1',
8   "PeriodTypeID" INTEGER DEFAULT TypeID('Monthly') DESCRIPTION '[Properties]
9   ReuseLast=1',
10  "Value" DECIMAL(19,4) DEFAULT 0 NOT NULL,
11  "Quantity" FLOAT DEFAULT 0 NOT NULL,
12  "LinkTable" VARCHAR(60) COLLATE "ANSI" NOT NULL,
13  "LinkID" INTEGER,
14  "StaffID" INTEGER DESCRIPTION '[Properties]
15  ReuseLast=1',
16  "DateCreated" TIMESTAMP DEFAULT Current_Timestamp,
17  "Complete" BOOLEAN DEFAULT false NOT NULL,
18  CONSTRAINT "PK_Forecasts" PRIMARY KEY ("ID"),
19  CONSTRAINT "StaffID" FOREIGN KEY ("StaffID") REFERENCES "Staff" ("ID")
20    ON UPDATE NO ACTION ON DELETE NO ACTION,
21  CONSTRAINT "PeriodTypeID" FOREIGN KEY ("PeriodTypeID") REFERENCES "Types" ("ID")
22    ON UPDATE NO ACTION ON DELETE NO ACTION
23 )
24 DESCRIPTION '[Properties]
25 AddDuplicateRecord=1'

```

Forecasts data-table structure

The Forecasts data-table has the following structure:

```

CREATE TABLE "Forecasts"
(
  "ID" INTEGER DEFAULT UID() NOT NULL,
  "DateStart" DATE,
  "Name" VARCHAR(120) COLLATE "ANSI"
  DESCRIPTION '[Properties]
  ReuseLast=1
  Distinct=1',
  "PeriodTypeID" INTEGER DEFAULT
  TypeID('Monthly') DESCRIPTION '[Properties]
  ReuseLast=1',
  "Value" DECIMAL(19,4) DEFAULT 0 NOT NULL,
  "Quantity" FLOAT DEFAULT 0 NOT NULL,
  "LinkTable" VARCHAR(60) COLLATE "ANSI" NOT
  NULL,
  "LinkID" INTEGER,
  "StaffID" INTEGER DESCRIPTION '[Properties]
  ReuseLast=1',
  "DateCreated" TIMESTAMP DEFAULT
  Current_Timestamp,
  "Complete" BOOLEAN DEFAULT false NOT NULL,
  CONSTRAINT "PK_Forecasts" PRIMARY KEY ("ID"),
  CONSTRAINT "StaffID" FOREIGN KEY ("StaffID")
  REFERENCES "Staff" ("ID")
    ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT "PeriodTypeID" FOREIGN KEY
  ("PeriodTypeID") REFERENCES "Types" ("ID")
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
DESCRIPTION '[Properties]
AddDuplicateRecord=1'

```

OrdinalPos	Name	Description	Type	Length	Nulls	GenCompDefault
1	ID		Integer		<input type="checkbox"/>	DEFAULT: UID()
2	DateStart		Date		<input checked="" type="checkbox"/>	
3	Name	[Properties]	Varchar	120	<input checked="" type="checkbox"/>	
4	PeriodTypeID	[Properties]	Integer		<input type="checkbox"/>	DEFAULT: TypeID('Monthly')
5	Value		Decimal		<input type="checkbox"/>	DEFAULT: 0
6	Quantity		Float		<input type="checkbox"/>	DEFAULT: 0
7	LinkTable		Varchar	60	<input type="checkbox"/>	
8	LinkID	[Properties]	Integer		<input checked="" type="checkbox"/>	
9	StaffID		Integer		<input checked="" type="checkbox"/>	
10	DateCreated		Timestamp		<input checked="" type="checkbox"/>	DEFAULT: Current_Timestamp
11	Complete		Boolean		<input type="checkbox"/>	DEFAULT: false

Forecasts data-structure

1. The LinkTable and LinkField allow "promiscuous linking" between any Forecasts data-record and other records or data-tables in your App.
2. Each row contains two number columns: "Value" and "Quantity" these can be used to store the data you want to set as your Forecast for that period.

SQL Script to add Forecasts to your Orixia App

Note that the script below requires the following BusinessObjects to already be present in your Orixia App: **People, Staff**.

```

-- ORIXA AUTOMATED SQL TO CREATE BUSINESS-OBJECT --
-- MAIN CREATE SCRIPT FOR Forecasts --
CREATE TABLE "Forecasts"
(
  "ID" INTEGER DEFAULT UID() NOT NULL,
  "DateStart" DATE,
  "Name" VARCHAR(120) COLLATE "ANSI" DESCRIPTION '[Properties]

```

```

ReuseLast=1
Distinct=1',
"PeriodTypeID" INTEGER DEFAULT TypeID('Monthly') DESCRIPTION '[Properties]
ReuseLast=1',
"Value" DECIMAL(19,4) DEFAULT 0 NOT NULL,
"Quantity" FLOAT DEFAULT 0 NOT NULL,
"LinkTable" VARCHAR(60) COLLATE "ANSI" NOT NULL,
"LinkID" INTEGER,
"StaffID" INTEGER DESCRIPTION '[Properties]
ReuseLast=1',
"DateCreated" TIMESTAMP DEFAULT Current_Timestamp,
"Complete" BOOLEAN DEFAULT false NOT NULL,
CONSTRAINT "PK_Forecasts" PRIMARY KEY ("ID"),
CONSTRAINT "StaffID" FOREIGN KEY ("StaffID") REFERENCES "Staff" ("ID")
    ON UPDATE NO ACTION ON DELETE NO ACTION,
CONSTRAINT "PeriodTypeID" FOREIGN KEY ("PeriodTypeID") REFERENCES "Types" ("ID")
    ON UPDATE NO ACTION ON DELETE NO ACTION
)
DESCRIPTION '[Properties]
AddDuplicateRecord=1'
!
-- INSERT BUSINESS OBJECTS DATA RECORD Forecasts --
INSERT INTO "BusinessObjects"
(Name, Color, BusObjType,
AddNewButton, LinkToImages, LinkToComments, LinkToAddresses,
LinkToPhones, SecurityLevel, Icon,
DisplayScript, ListScript, ViewScript, DiaryScript,
SummaryScript, Settings)
VALUES
('Forecasts', 16777215, 'N', True, False, False, False, False, 0, 181,
' SELECT
    Name, ID
FROM "Forecasts"
WHERE ID = %d ', ' SELECT
    Name as FullName, ID
FROM "Forecasts"
ORDER BY FullName ', 'SELECT
    ID,
    DateStart,
    Name,
    T1.Name as PeriodType,
    "Value",
    "Quantity",
    Pe.FullName as Staff,
    DateCreated,
    LinkTable,
    Complete
FROM "Forecasts" F
    LEFT JOIN Types T1 ON T1.ID = F.PeriodTypeID
    LEFT JOIN People Pe ON Pe.ID = F.StaffID
%s ',
    '', '', '')
!
-- INSERT SEARCHES DATA RECORDS --
INSERT INTO Searches
(LinkTable, Name, SQLStr)
VALUES
('Forecasts', 'All Records', '')!
-- INSERT SEARCHES DATA RECORDS --
INSERT INTO Searches
(LinkTable, Name, SQLStr)
VALUES
('Forecasts', 'Recently Added',

```

```

' WHERE DateCreated > Current_Timestamp - INTERVAL '1' MONTH ')!
-- INSERT SEARCHES DATA RECORDS --
INSERT INTO Searches
(LinkTable, Name, SQLStr)
VALUES
('Forecasts', 'Search "Name"',
' WHERE UPPER(Name) LIKE UPPER('%[STR Name]%' ) ')!
-- INSERT TYPES DATA RECORDS --
INSERT INTO Types
(Name, LinkField, LinkTable)
VALUES
('Weekly', 'PeriodTypeID', 'Forecasts') !
INSERT INTO Types
(Name, LinkField, LinkTable)
VALUES
('Monthly', 'PeriodTypeID', 'Forecasts') !
INSERT INTO Types
(Name, LinkField, LinkTable)
VALUES
('Annual', 'PeriodTypeID', 'Forecasts') !

```