DPR B.I.

DPR

From Components to Solutions

DPR BI/DW Solution Data Model Guide

DOCUMENT INFORMATION

Change Record

Version	Date	Author	Reason for Changes
1.0	11/11/2015	Colin Hardie	Document created

Document Properties

Name	Description
Document Title	DPR BI/DW Solution Data Model Guide
Author	Colin Hardie
Last Updated	11/11/2015
Version	1.0
Document status	Draft

Distribution List

Recipient Name	Title	Date
Raj Mittal	DPR Director	???
David Aylmer	BI Project Manager	???

Principal Contacts – DPR Consulting

Company Name	**DPR Consulting**	
Address	**International House**	
	1, St. Katherine's Way	
	London E1W 1UN	
Website	**www.dpr.co.uk**	
Name	**Title**	**Email**
Raj Mittal	Director	raj.mittal@dpr.co.uk
David Aylmer	BI Project Manager	david.aylmer@dpr.co.uk

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Document Purpose

This document describes at a conceptual level the DPR data models in both Origination and Servicing and their areas of commonality. From there it moves on to describe the Data Warehouse model in general terms, highlighting the different types of tables present and presents general principles for querying data. From there it proceeds to dive into a greater level of detail, moving table by table through the warehouse highlighting where each entity fits within the overall model, any specific points that need to be taken into consideration when querying that entity, and how it can be connected to the results of other queries.

1.2. How To Use This Document

This document should be read in conjunction with the Data Warehouse Data Dictionary and ER Diagrams. Rather than attempting to read it cover to cover, it should be used on an "as needed" basis. A typical scenario might be something like the creation of a new report for which you would need to:

1. Identify the data items that are required in the report at a conceptual level
2. Map these conceptual requirements to the Data Dictionary (with help from DPR as appropriate)
3. Identify the table that each of the mapped fields reside in and consult this document for specific guidance on querying those tables and joining them together into a single result set (with guidance from the ER diagrams if necessary).

Another use case might be for someone new to the DPR data model and the Data Warehouse who wishes for a broad overview, in which case the overview sections could be read without needing to dip into the finer detail present in the table based sections.

1.3. Document Overview

The following sections make up the rest of this document:

- Section 2 – DPR Origination Model Overview – this section gives a high level overview of the main concepts from of the Origination data model.
- Section 3 – DPR Servicing Model Overview – this section gives a high level overview of the main concepts from of the Servicing data model.
- Section 4 – Date Warehouse Model Overview – this section is more technical in nature and describes the different entity types present in the Date Warehouse, their naming conventions, and general principles for querying. Specific query principles for each table are broken down in succeeding sections.
- Section 5 – Origination Tables – this section take a more detailed look at all of the tables in the main dbo schema of the Data Warehouse that relate to Origination data, including any tables that are shared with Servicing.
- Section 6 – Servicing Tables - this section repeats the methodology of Section 5 and details the tables that are pertinent to Servicing. It also goes over the tables that are common to both Origination and Servicing again but from a Servicing perspective.

2. DPR ORIGINATION MODEL OVERVIEW

2.1. ORIGINATION CONCEPTUAL MODEL

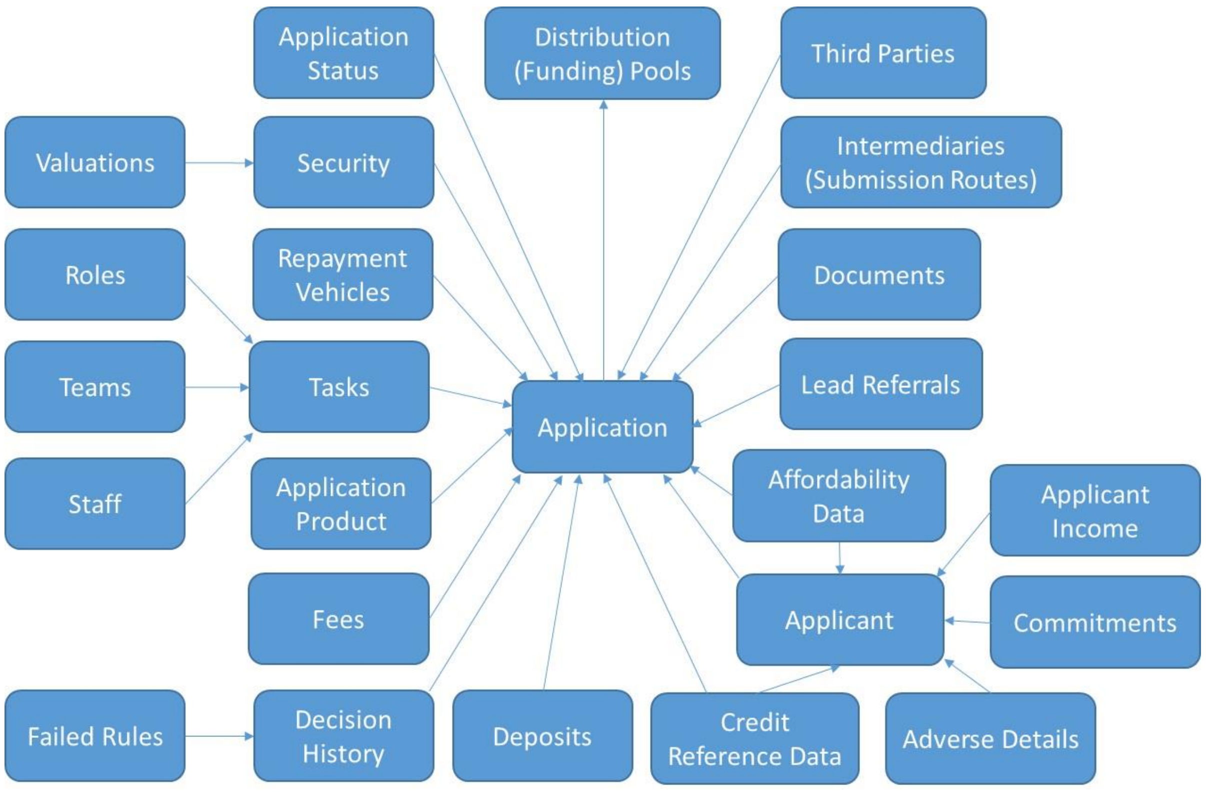


Figure 2.1: *The Origination Conceptual Model*

2.2. APPLICATIONS

Applications are the central object within the DPR Origination system to which everything else ultimately links back. Whether applying for a loan or a savings account, the same application data model is used. Any attributes that affect the application as a whole (e.g. Account Category) are linked to this level of information (as opposed to those attributes which may only reflect a particular product within an application – see Application Products).

In contrast to the Servicing data model, where most operations are governed by a "Process", in the Origination data model there is only one Process – the Application itself.

2.3. APPLICANTS

Applicants are those people who make applications, be that for savings accounts or for loans. Each Applicant record is unique to an Application, so that even if the same physical person makes more than one Application, they will be represented by multiple Applicant records, once for each Application they are associated with. Within the DPR system there is no flag to specifically mark out the "Primary" Applicant. Instead, each Applicant within an Application is assigned an integer customer number, normally 1 or 2. This number is assigned based on the order in which the applicants appear on the application form.

2.4. APPLICATION PRODUCTS

Within Origination, "Products" are actually more like a generic grouping of offerings available. Each "Product" can have more of more "Variants" each of which can have one or more "Permutations" that can in turn have one or more different "Purposes" each of which can have one or more "LTV Bands". These are all held within a number of different tables within the Origination Products database and one or more of these combinations of Product, Variant, Permutation, Purpose and LTV Bands will be selected for an Application. Applications can be made up of many Application Products. An example might be as follows:

- Product – Fixed Rate
    - Variant – 2 year fixed period
        - Permutation – reverts to the standard variable rate (SVR)
            - Purpose – Remortgages
                - LTV Band – 65% - 75% LTV

Within a single application, different types of application can be made. For example, an existing borrower may be in the process of moving house but would like to move their existing mortgage over to the new property – this is known as a Port. In addition, they may also need to borrow a further tranche of money in order to pay for the new house. This would be done as a New loan. Both the Port and New loan would be handled by a single Application but could be attached to two different Application Products.

2.5. SECURITIES

For secured loans of any kind, Securities are those entities that offer collateral as the basis of the loan. The most familiar example of this is a mortgage being taken out on a property. More generally it can be anything of value that in the event of a borrower being unable to repay their loan, could be sold to repay the outstanding debt. Applications can have one or more securities; however they can only have one that is marked as the “Primary” security.

2.5.1. Security Charges

A standard mortgage is normally taken out in a “First Charge” capacity. This means that it is the primary debt secured against the Security. Subsequent mortgages may be taken out in a “Second charge” capacity, which means that should both mortgages go into default and the Security is sold to repay them; the Second Charge will only be paid off after the First Charge mortgage has been cleared.

2.6. VALUATIONS

Each Security can have one of more Valuations. Typically, within a mortgage context the first Valuation will be of the “CustomerProvided” type. Subsequent Valuations can be of various types, including automated (index) valuations, full valuations etc. Within the Origination data model, it is possible to select up to two Valuations as the Active/Selected Valuation, one of CustomerProvided type and one of another type. If both are present, then it is the non-CustomerProvided type that will be used as the basis of relevant calculations, such as the LTV ratio.

2.6.1. Valuation Comparables

A Valuation can be given context by comparing the property being evaluated to recently sold similar properties in the area. This will be noted as a Valuation Comparable.

2.7. INTERMEDIARIES

Applications can either come in directly or via some form of Intermediary submission route. These are external individuals and companies who pass on business to the institution in question for a fee (either paid by the institution or in some cases by the Applicants themselves). The lowest level of Intermediaries are Brokers. These deal directly with the Applicants. A Broker can be a part of a Network, a Mortgage Club, a Packager, some, all or none of these. The implied hierarchy of seniority is Packager > Mortgage Club > Network > Broker.

2.8. THIRD PARTIES

These are other external parties (excluding Intermediaries and Applicants) who are linked in some way to an Application. Typically these will be Solicitors acting on behalf of the Applicant or Lender on a mortgage.

2.9. Repayment Vehicles

In the case of Interest Only or Part and Part mortgages, it is possible to capture one or more Repayment Vehicles i.e. the method that will be used to pay off the capital at the end of the interest only term. Common Repayment Vehicles are the sale of the property, an ISA etc.

2.10. Commitments

Commitments are debts (either secured or unsecured) which the Applicant may already owe. These will be taken into account when assessing the Applicant for suitability to receive the loan. Examples might include loans, mortgages or credit card debt. This data is as reported by the Applicants themselves, rather than by the Credit Reference agencies

2.11. Distribution (Funding) Pools

One of the optional modules within the Origination solution is the Asset Management solution, which facilitates tranche management. Funding or Distribution pools are essentially tranches (buckets) of money that can be used to support a particular application. In other words, when an application comes in, if approved it will be allocated a subset of the total facility within a particular tranche of funding.

2.12. Credit Reference Data

When assessing an Applicant for credit worthiness, the services of a credit reference agency such as Experian or Equifax will normally be utilised. This data encompasses details of know incomes and debts, county court judgements, bankruptcies and the like. The full response of the agency is sent to the Origination solution in the form of an XML Document. This document will normally be several hundred lines long and contain a great deal of information that the Origination solution will not make use of. As a result it will only store a subset of the available fields within its database but will still store the original document within the DPR Filestore.

2.13. Affordability Data

For Applications for lending Products, most will include the use of an automated “Affordability Model” designed by the lender to assess whether an Application can safely be accepted, based on the income and debt levels of the Applicants behind it. Each Affordability model us unique to the lender and can be composed of multiple different formulae. For regulatory purposes, this data has to be stored in order to record the basis for the decisions made and how it was arrived at. The form that this takes within the Origination solution is the output an XML Document and save it in the DPR Filestore.

2.14. Adverse Details

Adverse details are serious events that have taken place within an Applicant’s financial history such as missed payments on existing credit, arrears, defaults, county court judgements etc. This data is taken into account when assessing an Applicant for new credit. This data is as reported by the Applicant themselves, rather than from the Credit Reference agency.

2.15. Decision History

At each stage of the Application Pipeline data a decision can be made on whether to Accept, Decline or Refer an Application. An Accept decision just means that an Application can proceed to the next stage of the pipeline, whilst a Decline normally stops it from proceeding any further. A Refer decision is a placeholder until someone else can look at the Application in more detail before deciding to Accept or Decline it. Accept and Decline decisions can be overridden by a user of the Origination solution, in which case they can optionally note why such an Override was made. The Decision History of an Application is the total set of all Decisions made against it. Related to this history is the concept of Failed Rules.

2.16. Application Statuses (Pipeline)

The Application process, particularly for lending products can be broken down into several discrete stages. For example a typical mortgage application might go through the following stages (though this is customisable):

- Illustrations (KFI)
- Decision in Principle (DIP)
- Full Mortgage Application (FMA)
- Underwriting
- Post Offer
- Completion
- Expired

Within each stage are a collection of sub-statuses that can be applied. For example, within the FMA stage you might have an AcceptFMA, DeclineFMA or ReferFMA status. Within Underwriting you might have Awaiting Valuation, Valuation Received and Offer Made. Each marks a specific point in the Application process. For each Application, the full history of sub-statuses assigned to it is recorded within the Origination solution along with a timestamp as to when it made it to that particular status.

The Expired stage of the process is normally automatically applied 6 months after a "final" status has been posted i.e. after a Cancel/Decline or Completed status.

2.17. Deposits

Deposits, in the case of lending Products like mortgages, are lump sums that are put down as part of the requirements in taking out a loan. For example, a 95% LTV mortgage Product will require a minimum deposit equal to 5% of the value of the property in question. Details of the deposit held by the Origination solution may include its value and its source e.g. inheritance or savings.

2.18. Failed Rules

The Origination solution has the ability to make automated decisions based on a series of rules as defined by the lender. Rules are evaluated in sets called policies and each policy may be evaluated at different stages of the Application Pipeline. If a particular rule fails evaluation, then this can trigger a Decline Decision to be automatically made on an Application. This history of failed rules is kept within the Origination solution and is linked to any Decisions made as a result of that rule failure.

2.19. Documents

Both the Origination and Servicing solutions have the ability to store documents in a central repository called the DPR Filestore. These documents can be of many formats including PDF and XML and cover everything from letters and KFI documents through to the results of calculations. Of particular interest to the BI are those documents that are stored in an XML format. These documents can be brought into the data warehouse and stored in an XML data type column so that they can be queried.

2.20. Tasks

Tasks are discrete units of work that are either automatically or manually created within the DPR system or linked to a particular Application. They are always assigned to a Role that has been created with the User Admin system and optionally may be allocated to a Team and/or Staff Member. Each Task is created with an expected completion time and records when the task is actually completed.

2.21. Fees

Fees are charges to be applied to an Application over the course of its lifetime. Some are for immediate payment whilst others are recorded for "Information Only" purposes i.e. the Origination system does not handle their payment. Typical fees on a Mortgage may include Procuration fees, Valuation fees, Redemption fees etc.

2.22. Applicant Income

When an Applicant applies for a loan, they are required to submit details of their income (along with proof for verification purposes – unless they are self-certified which is becoming increasingly rare). This information is then used in assessing them for Affordability. Multiple incomes can be recorded and can include Employed, Self Employed, Retirement and Other types of income.

2.23. Roles

These are generic titles set up within the User Admin system that describe high level job roles and to which Tasks are always assigned. Examples might be Underwriters, Case Owners etc.

2.24. Teams

These are groups of Staff that can optionally be allocated Tasks. Note that Staff can be allocated to tasks across multiple Teams.

2.25. Staff

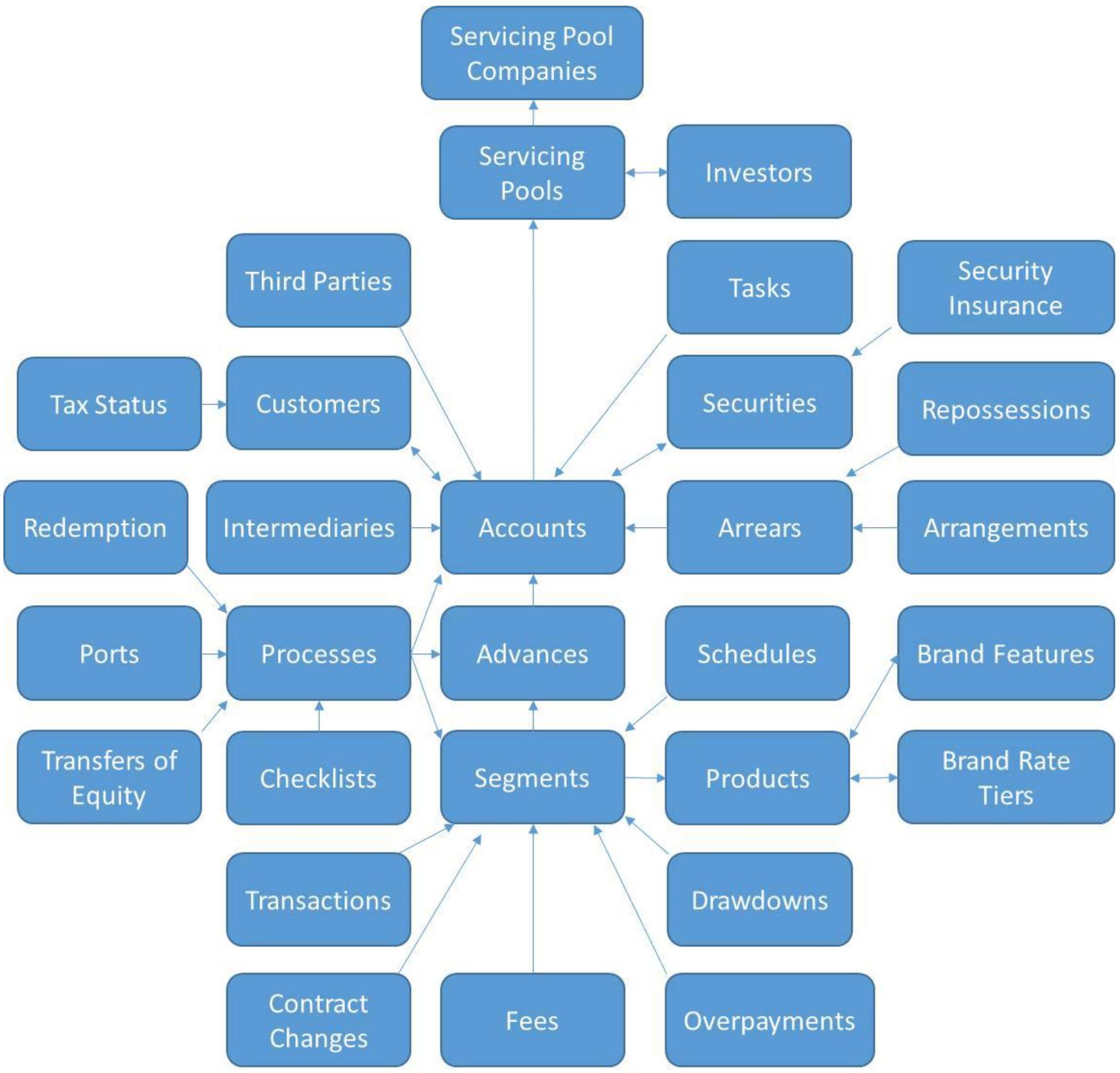
These are individuals set up within the User Admin system that can use the DPR solutions. They can optionally be assigned to Tasks.

2.26. Lead Referrals

In short term lending models (like Payday Loans) it is possible for Applications to be referred from other Lenders. This data is captured as a “Lead Referral” and captures details like the source and type of referral that has been made along with any comments associated with it.

3. DPR SERVICING MODEL OVERVIEW

3.1. Servicing Conceptual Model



3.2. Accounts

In the DPR Servicing model, Accounts are essentially containers that hold the details of the various Products that Customers have taken out with an institution. Each Account belongs to a particular Category and only Products that have meaning within a particular category can be selected within such an Account (e.g. only Bridging Products can be used within a Bridging Account). At the time of writing the possible categories are:

- Lending
  - Bridging
  - CommercialMortgage
  - FirstCharge
  - Lifetime
  - RetailLoan
  - SecondCharge

- Savings
  - RetailSavings

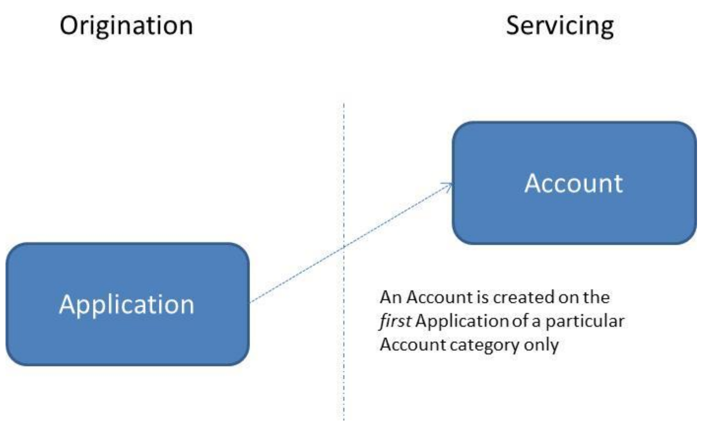
- CommercialSavings

Thus if a Customer has both a mortgage and a savings account, they will have two Accounts of different categories rather than one Account with two Products.

Each Account is identified by an Account Number that is unique within the DPR Servicing system.

3.2.1. Accounts and the Origination Model

An Account will be created by an Application if it is the *first* Application a Customer has made within a particular Account category. Subsequent Applications will create new Advances and Segments only.


Origination
Servicing
Account
Application
An Account is created on the *first* Application of a particular Account category only

3.2.2. Account Events

These are essentially interventions made on a lending account to stop it going into Arrears/Foreclosure. Similar in intent to Arrangements, but done prior to any official proceedings have begun, they would usually take place before an Arrears Episode was created.

Conceptually there is a hierarchy above the Account Events level (or Actions as the Warehouse calls them)

Everything happens within the context of an Episode – this is not the Arrears Episode, but a “container” for all of the Actions that take place to stop an Account moving into Arrears. Within an Episode there will be multiple Actions that are undertaken, each of a particular type – e.g. temporary move to interest only, payment holiday etc.

3.2.3. Account on Hold

An Account can be placed on hold for multiple reasons (e.g. Dormancy, Sanctions etc.) and usually is done for an open ended period until something happens to end the reason for it being on hold.

3.2.4. Account Actions

Account Actions (or Account Status Dates) are a record of actions or events that have taken place or will take place on an Account that affect the status of the Account. These can include things like Dormancy Overrides, the end of an Account Holder being classified as a child etc.

3.2.5. Account Statuses

An Account can have one or more statuses applied to it at a given time. An Account operating normally may simply have a status of “Active” applied to it. Others may be things like “Zero Funding”. Essentially they are descriptors as to the operational health of the Account in question.

3.3. ADVANCES

Historically the DPR Servicing system was created to handle the day to day operations of mortgages, hence the use of Advances. Since the system was expanded to handle savings, Advances have to be considered from two points of view.

3.3.1. Advances within the Savings Account Categories

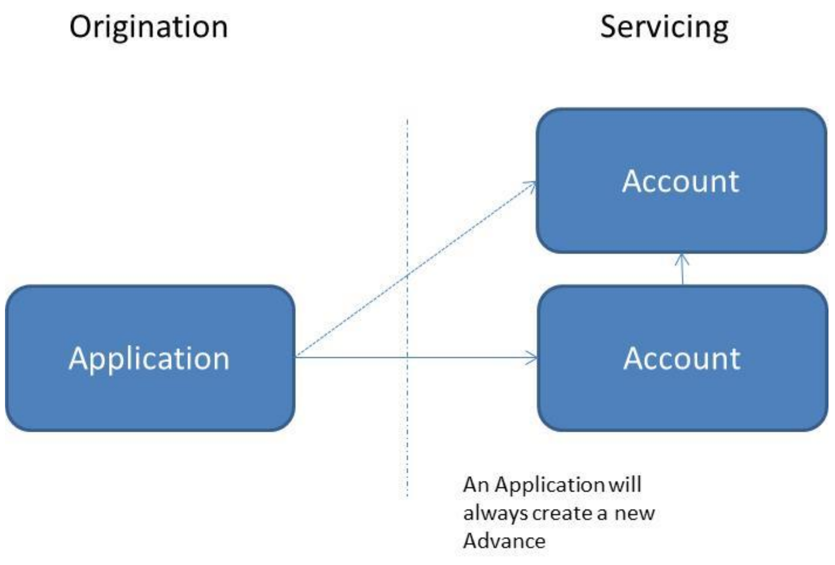
For Accounts within the savings categories, an Advance can be thought of as equivalent to the Account as there is a one-to-one relationship between the two (and conceptually the idea of an Advance within a savings Account is meaningless anyway). Data will still be stored at this level but will be analogous to capturing it at the Account level.

3.3.2. Advances within the Lending Account Categories

For Accounts within the lending categories, an Advance is essentially a tranche of money that a customer has been given. How that tranche is used, depends on the Account category to which it belongs. In most lending categories, the total value of the Advance will be drawn down (i.e. given to the customer) all at once (ignoring any possible effects of retention amounts on the loan). In the Lifetime Account category however, the Advance represents an agreed limit of money that may be drawn down and may be paid out in instalments (Drawdowns)

3.3.3. Advances and the Origination Model

A new Advance will *always* be created by an Application. If it is the first Application within a particular Account category then it will also create an Account.



3.3.4. Advance Source

The source information for an Advance is essentially a record of where it came from. Normally this will be from one of three places

1) Origination – this indicates that it came from the DPR Origination system
2) Ingest – this indicates that was an existing advance in another system and came through the Onboarding process
3) Anything else – Originated in some way within the Servicing system, for example as a Port or Further advance.

There will also be additional information captured with the type of source.

3.3.5. Advance Affordability Checks

This is a smaller set of Affordability data that is passed through to the Servicing system on Completion. Historically this information was used in regulatory reporting, but this has now been superseded by the Affordability Data in Origination. This information is now included only for backwards compatibility purposes.

3.4. ARRANGEMENTS

Arrangements are temporary measures that can be put in place on a loan when the customer has entered into a formal Arrears Episode. These arrangements are designed to help the customer get back on track without having to resort to stronger measures like property repossession.

3.5. ARREARS EPISODES

These are formal procedures instigated by a lender when a Customer falls behind in loan payments. There can be several stages within an Episode ranging from initial attempts to get a Customer back on track through to more severe stages like Litigation and Repossession.

3.6. BRAND (PRODUCT) FEATURES

Brand Features describe some of the general functionalities of the Product, for example if it is a Savings Account it may have a minimum deposit feature, or if it is a loan product is may come with cashback. The Servicing system stores the list of features separately to the Products as they can apply to many of them and then uses a many-to-many relationship table to move between the two.

3.7. RATE TIERS

A product (either savings or loan) can have multiple interest rates associated with based on the balance or loan amount. These are known as Rate Tiers.

3.8. CONTRACT CHANGES

Contract Changes are another way of managing a formal Arrears Episode. In this case, they are a permanent change to the contract of a segment. For example, someone in Arrears may have their term increased or their payment amount changed.

3.9. CUSTOMERS

Customers are the Servicing equivalent of Origination Applicants. Unlike their Origination counterparts they are unique and do not duplicate. As a result they have a many-to-many relationship with Accounts (one Account can have many Customers and one Customer can have many Accounts). Customers within Servicing can be broadly categorised by three types of role they can play on an Account.

- Borrower – This is a catch-all term that applies to both lending and savings accounts. In each, these are the customers that actually use the Account i.e. the Account holders.
- Guarantor – These are not Account holders, but provide some form of guarantee on the Account, particularly relevant to loans.
- Other – This tends to be used mostly in Commercial loans where the Account holder is a Company and a named individual is also required to act a primary contact on the account. This “other” is normally a director.

3.9.1. Customer Income

See Origination entry

3.9.2. Customer Source

Similar to the Advance Source, this is the record of where a Customer first entered the system.

3.9.3. Customer Status

A Customer may not have any statuses associated with them, but if they do this can include things like Deceased, Bankrupt, R105 etc. The dates associated with the status are also recorded.

3.9.4. Customer Tax Status

Within savings accounts, Customers can be registered to pay tax in a Net or Gross capacity for certain periods.

3.10. Drawdowns

A Drawdown is the action of withdrawing money from an agreed loan facility. On standard loans, the initial drawdown will normally be for the entire facility (less any retention) but for Lifetime mortgages, this facility can be taken over a period of time using several Drawdowns until the facility is used up. Within normal mortgages, if an amount is retained and subsequently released, then you can also see multiple drawdowns taking place.

3.11. Transactions

These are the atomic financial movements that take place at the Segment level. The transactions exposed within the BI solution come from the LssAccounting database which is the source of the General Ledger.

3.12. Investors

Within certain funding models for loans, investors can directly provide money for the purposes of lending to institutions. This is administered at the Servicing Pool level, whereby an investor will have ownership of a particular pool of Accounts. In turn the investor will earn a yield based on their loan account performance.

3.13. Schedules

There are various types of schedules that apply to different forms of Accounts.

Within lending based Accounts, the most common forms are:

- Repayment Schedules – these layout what is to be paid by a Customer and when in order to pay back their loan
- Fee Schedules – these are less common, but some fees can be scheduled to be paid on a particular date
- Payment Schedules – these are payments out by a lender to a Customer, typically called Cash Releases. Someone on an equity release mortgage may have such a schedule of payments.

Within savings based Account the most common forms of schedules are:

- Bonus – Accounts with a bonus linked to them may have a schedule set up to pay that bonus out at a certain time, if all criteria have been met.

3.14. Ports

A Port is the action of moving an existing mortgage agreement from one security to another i.e. the borrower is moving house and wants to keep his existing mortgage deal. Ports are a Process that can take place within the Servicing system, without resorting to using the full Origination Application Process.

3.15. Processes

Processes are the central Servicing method of completing actions on an Account. They cover a wide variety of scenarios from charging a fee to carrying out a Port or Transfer of Equity. Processes can affect a whole Account, an Advance or just individual Segments.

3.15.1. Process Actions

These are the individuals steps carried out within a Process via Wizards.

3.16. Segments

Segments are the point at which Accounts/Advances are linked to the Products they are associated with. It is at this level that most of the important financial information is held such as balance and Transactions

Again because of the multiple categories of Account that are possible Segments must be thought of from two perspectives.

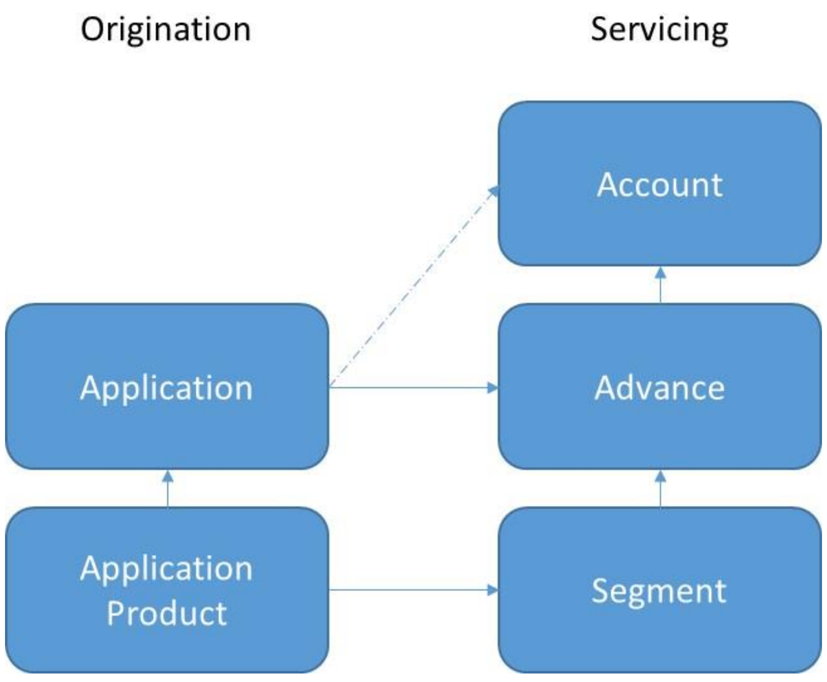
3.16.1. Segments within the Savings Account Categories

As with Advances, Segments within savings Account are synonymous with the Account itself, as there will only be one active Segment on a savings Account at any one time (although there may be closed segments still attached to it).

3.16.2. Segments within the Lending Account Categories

An Advance can be split into multiple Segments within the lending Account categories and an Account can have multiple Advances and Segments active at the same time. For example if a customer has a single Advance, some lenders will allow them to split this loan between a capital repayment Product and an Interest Only Product, thus creating two Segments within the Advance. Another scenario may be that a Customer has an existing mortgage (represented by one Advance with one Segment) and chooses to take out a further advance for some reason, resulting in a second Advance with a second Segment being created.

3.16.3. Segments and the Origination Model



For each new or ported Application Product within an Application, a new Segment will be created within a new Advance.

3.17. Products

Servicing Products are stored in a separate database to those used in Origination. This allows for only those Products that are to be sold to Applicants to be defined in the Origination solution, whereas older Products that are no longer available but are still used by older Customer can be defined in Servicing.

The Products in both Origination and Servicing used a common model to describe them. See Application Products for more details.

3.18. Redemptions

Redemptions are the act of fully paying off a Segment. This can happen naturally at the end of a mortgage term or can be requested earlier, in which case a quote is generated called a Redemption Request. This request projects the final cost of the redemption, including any fees and interest that may be due.

3.19. Pools

Pools are essentially groupings of Accounts. This can be done for one of several reasons including:

- Ownership – Some pools may belong to different investors
- Administration – Some groups of Accounts may be tracked and administered in groups
- System Efficiency – For a large number of Accounts, it is sometimes necessary for the Servicing solution to process them in batches, which can be accomplished through the use of Pools
- Category – Pools containing Savings Accounts and Mortgage Accounts are often kept separately
- Book Purchases – A lender will sometimes purchase a “book” of mortgages from another lender and these are usually kept separately to others

Every Account can only belong to a single pool at any one time.

3.19.1. Split Pools

Split Pools are a feature of the Servicing system used to cope with processing errors on particular Accounts. If an error occurs during the overnight batch run or during the generation of the GL, and the issue is with a particular Account or subset of Accounts, these Accounts can be separated from their main Pool and put in a temporary Split Pool. This allows the batch to be run on the majority of the Accounts while the issues on the problematic ones are solved. They can then be added back into the main pool and their processing progressed.

3.20. Pool Companies

It is possible for a Servicing Pool to be associated with up to three companies:

- Servicer – typically the lender or institution offering a savings Account and who operates the Account on a day to day basis
- Asset Owner – if the Servicer operates the Account on behalf of another institution (similar to an investor) the Asset Owner represent the actual owner of the Account from a non-customer perspective
- Brand -

3.21. Transfers of Equity

When a Customer is removed from a mortgage, the equity that was in their name is passed to someone else, either a new Customer on the mortgage or another existing one. The Transfer of Equity process is what allows this to take place within the Servicing solution.

3.22. Checklists

These are a series of name value pairs that can be associated with a particular instance of a Process. They are stored in this generic way in order to allow full flexibility in the definition of data to be gathered.

3.23. Tax Status

Within Savings Accounts, the Tax Status of a Customer defines whether they should receive interest payments Gross (without tax removed) or Net (with tax removed). All Customers on an Account must be flagged as Gross for tax not to be removed.

3.24. Overpayments

As well as the regular schedule of repayments on a mortgage, a Customer can choose to make overpayments. These overpayments can be used to pay off more capital on the loan, to be used in lieu of "payments holidays" or allow underpayments in the future.

3.25. Repossessions

When an Arrears Episode reaches its most serious point, this may result in a lender beginning a Repossession process. This involves getting a court order to take possession of the security on the loan and to force the sale of it in order to pay off the owed money.

3.26. Security Insurance

Insurance held against the security comes in various forms including buildings cover, contents cover and can also include Mortgage Indemnity Guarantees (MIGs) which cover the lender in case the Customer is unable to pay the mortgage. MIGs tend to only be taken out in high loan to value cases.

3.27. Securities

See Origination entry

3.28. Intermediaries

When a loan is live, the concept of Brokers has less meaning than in Origination. Instead you may have Servicing Agents associated with an Account. These are Servicers who may not be associated directly with the lender but still play some administrative function.

3.29. Third Parties

These can be broadly split into two sorts in Servicing:

- Originating Third Parties – these are essentially just held for information purposes and describe those parties that were used during the application process e.g. the broker, the solicitor etc.
- Servicing Third Parties – these are parties who play a role on the Account but are not defined as one of the Customer roles. These can include Solicitors, Accountants, Beneficial Owners, Parents (on child accounts) etc.

3.30. Documents

Within Servicing, it is also possible to store documents within the DPR Filestore; however from the BI perspective this tends to only include Credit Reference data. See the Origination entry.

3.31. Tasks

See Origination entry

3.32. Fees

See Origination entry

4. GENERAL DATA WAREHOUSE PRINCIPLES

4.1. NAMING CONVENTIONS

4.1.1. Field Naming Conventions

Most fields in the data warehouse have no particular naming convention; however there are some that it is important to note in order to know how to make use of them.

- _DWSK – All of the dimension tables in the data warehouse (and some of the fact tables if appropriate) have a meaningless integer "surrogate key"[1] – a concept used widely in Kimball data warehouses. All foreign key relationships are based on these surrogate keys and their presence allows historic change to be recorded. DWSK stands for Data Warehouse Surrogate Key.

- _SSK – All of the dimensions tables in the data warehouse (and some of the fact tables if appropriate) have an SSK field, which stands for Source System Key. This is the "natural key" of the dimension and in most cases is a GUID that allows you to link back to the equivalent entity or row in the source system table. See the Data Dictionary for which fields to map back to in each case.

- SCD… - SCD stands for Slowly Changing Dimension and is another standard Kimball methodology technique for how change is captured. Each dimension table has an SCDStartDate, SCDEndDate and SCDStatus field which tell you in combination the dates on which a particular "version" of an entity was relevant.

- _ABS – These numeric fields give the true value of a particular measure at the time that it was captured.

- _DELTA – These numeric fields give the change in value of a particular measure since the last time that it was recorded (normally the previous day)

4.1.2. Table Naming Conventions

There are broadly three forms of table in the data warehouse, each type indicated by the appropriate table prefix. These are:

- FACT_ - These are "Fact" tables, following the Kimball approach[2] to dimensional modelling, which contain "measurements" – either numeric data or events. Also in line with Kimball methodology there are three forms of fact table utilised in the data warehouse. These are:

    - Transactional – These populate every time an event happens and then never update. They record a timestamp of when the transaction took place and use the appropriate _DWSK foreign key that was relevant at the time that the transaction took place. An example of this type of fact table is FACT_CO_DecisionHistoryTrx

    - Accumulating Snapshot – These populate a new record every time an event happens but then *do* update those records and will always contain current rather than historic values. An example of this type of table is FACT_CO_ApplicationProductAS

    - Periodic Snapshot – These record a "snapshot" of the same data every day. They record the DWSK in use at that moment in time and allow for historic reporting. An example of this type is the table FACT_CO_ProdSegm

[1] http://www.kimballgroup.com/1998/05/surrogate-keys/
[2] http://www.kimballgroup.com/2008/11/fact-tables/

- DIM_ - These are “Dimension” tables in the Kimball approach[3], describing the context of the measurements expressed in the fact tables. These dimension tables utilise Type 2 Slowly Changing Dimension[4] logic to track change over time.

- IF_ - These are “intermediate fact” tables that allow the modelling of many-to-many relationships. For example an Account can have many Customers and a Customer can have many accounts, therefore we include a table that maps the Customers to the Accounts and include date columns for when this relationship was in place for.

4.2. General Querying Approaches

4.2.1. Dimension tables

Because most of the Dimension tables in the data warehouse adopt Type 2 SCD logic, it is important when querying them to ensure you are only retrieving one “version” of each entity they represent. This can be done in two ways.

The first and easiest method is simply to use the SCDStatus column in each Dimension table. This can contain two values, either ‘H’ for historic (i.e. it is an “old” version of the entity) or ‘C’ for current. If you are only interested in the latest version of the entities, then you can simply retrieve this by filtering on this column i.e.

```
SELECT <Fields>
FROM DIM_CO_<DimensionName>
WHERE SCDStatus = 'C'
```

The problem with this approach comes when you are trying to view either historic data or to do time series analysis that includes but is not limited to the current version. In these cases, if you filtered on the current version only, then any historic data captured against previous versions of the entity would be filtered out. In these cases, when you are trying to report data as of a particular date or across a series of dates then you need to use the SCDStartDate and SCDEndDate fields.

If reporting on a particular known date then you would use a query like:

```
DECLARE @ReportingDate INT = '20151122'

SELECT <Fields>
FROM DIM_CO_<DimensionName>
WHERE SCDStartDate <= @ReportingDate
AND SCDEndDate > @ReportingDate
```

Or if reporting on a series of dates e.g. when looking at status history of an application you would use a query like:

[3] http://www.kimballgroup.com/2003/01/fact-tables-and-dimension-tables/
[4] http://www.kimballgroup.com/2008/08/slowly-changing-dimensions/ & http://www.kimballgroup.com/2008/09/slowly-changing-dimensions-part-2/

```
SELECT
       c.ApplicationRef
       ,a.TransactionDatetime
       ,d.StatusName
FROM FACT_CO_StatusTrx a
       JOIN DIM_CO_Process b
              ON a.DIM_CO_Process_DWSK = b.DIM_CO_Process_DWSK
              AND b.SCDStartDate <= a.TransactionDate
              AND b.SCDEndDate > a.TransactionDate
       JOIN DIM_CO_Application c
              ON c.DIM_CO_Process_DWSK = b.DIM_CO_Process_DWSK
              AND c.SCDStartDate <= a.TransactionDate
              AND c.SCDEndDate > a.TransactionDate
       JOIN DIM_CO_Status d
              ON d.DIM_CO_Status_DWSK = a.DIM_CO_Status_DWSK
ORDER BY c.ApplicationRef, a.TransactionDatetime
```

Note that DIM_CO_Status is one of the few dimension tables in the warehouse that does not use Type 2 SCD logic, hence the lack of filtering on that table in the query above.

4.2.2. Fact Tables

The querying of a Fact table is rarely done in isolation, but in combination with an appropriate set of dimension tables, specific to its local star schema. Because each Fact table behaves in subtly different ways, rather than present a generic guide to querying them here, it is recommended that the reader consults the appropriate table specific guidelines presented in either section 5 or section 6 of this document whenever data from a specific table is required.

4.2.3. Relationship (IF) tables

Similarly to Fact tables, IF tables are rarely queries in isolation but within the context of the Dimension table linked to them. Again, when querying a specific IF table, the reader should consult the table specific entries in section 5 or 6 of this document.

4.2.4. Linking them together (JOINS)

A simple query on the warehouse may only utilise a single Fact or IF table and in those cases it is fairly straightforward to retrieve the required information. In more complicated queries however, it is not uncommon to want to link information from several Fact tables and in those cases things can become complicated unless a structured approach is taken.

In general it is advised that each Fact and IF table be treated as its own separate star schema and that queries which combine information from more than one star schema should be joined together using the appropriate SSK rather than the DWSKs recorded in each. This is based on the fact that each Fact and IF table may behave in different ways and there is no guarantee that the DWSK included in one Fact table will be the same as that used in another.

For example, suppose we want to query the latest balance of all of the Segments in an Account and also to retrieve a summation of all of the overpayments made on those Segments. Because you are retrieving the latest information from FACT_CO_ProdSegm but will be using the historical information from FACT_CO_OverpaymentDetailTrx, you cannot simply join one Fact table to another directly using the DIM_CO_ProdSegm_DWSK column present in both tables. Instead the best approach is to create two subqueries (one per Fact table/star schema) and then join them using the SSK. In other words this query would produce incorrect results…

```
SELECT
       d.AccountNo
       ,c.AdvanceSeqNumber
       ,b.SegmentSeqNo
       ,LatestBalance = MAX(a.Balance_ABS)
       ,TotalOverpayments = SUM(e.Amount)
FROM FACT_CO_ProdSegm a
       JOIN DIM_CO_ProdSegm b
              ON a.DIM_CO_ProdSegm_DWSK = b.DIM_CO_ProdSegm_DWSK
       JOIN DIM_CO_Advance c
              ON c.DIM_CO_Advance_DWSK = b.DIM_CO_Advance_DWSK
       JOIN DIM_CO_Account d
              ON d.DIM_CO_Account_DWSK = c.DIM_CO_Account_DWSK
       LEFT OUTER JOIN FACT_CO_OverpaymentDetailTrx e
              ON e.DIM_CO_ProdSegm_DWSK = b.DIM_CO_ProdSegm_DWSK
WHERE a.AsAt_DATE =  <LatestDate>
GROUP BY d.AccountNo, c.AdvanceSeqNumber, b.SegmentSeqNo
```

…whereas this one would not…

```
WITH cte_BalanceInformation AS
(
       SELECT
              d.AccountNo
              ,c.AdvanceSeqNumber
              ,b.SegmentSeqNo
              ,b.DIM_CO_ProdSegm_SSK
              ,a.Balance_ABS
       FROM FACT_CO_ProdSegm a
              JOIN DIM_CO_ProdSegm b
                     ON a.DIM_CO_ProdSegm_DWSK = b.DIM_CO_ProdSegm_DWSK
              JOIN DIM_CO_Advance c
                     ON c.DIM_CO_Advance_DWSK = b.DIM_CO_Advance_DWSK
              JOIN DIM_CO_Account d
                     ON d.DIM_CO_Account_DWSK = c.DIM_CO_Account_DWSK
       WHERE a.AsAt_DATE =  <LatestDate>
),
cte_Overpayments AS
(
       SELECT
              b.DIM_CO_ProdSegm_SSK
              ,Overpayments = SUM(a.Amount)
       FROM FACT_CO_OverpaymentDetailTrx a
              JOIN DIM_CO_ProdSegm b
                     ON a.DIM_CO_ProdSegm_DWSK = b.DIM_CO_ProdSegm_DWSK
       GROUP BY b.DIM_CO_ProdSegm_SSK
)
SELECT
       a.AccountNo
       ,a.AdvanceSeqNumber
       ,a.SegmentSeqNo
       ,LatestBalance = a.Balance_ABS
       ,TotalOverpayments = b.Overpayments
FROM cte_BalanceInformation a
       LEFT OUTER JOIN cte_Overpayments b
              ON a.DIM_CO_ProdSegm_SSK = b.DIM_CO_ProdSegm_SSK
```

In general, this approach of creating separate queries per Fact table and them linking them together using the appropriate SSK is the advised approach when doing complicated querying like this. The same holds true when combining IF and Fact tables in the same queries as well.

More generally it is also worth bearing in mind the results that specific JOIN types will have on a query.

Any time an INNER JOIN is used, this is equivalent to specifying a WHERE clause with the join criteria in i.e.

```
SELECT *
FROM TableA a
       JOIN TableB b
              ON a.TableBKey = b.TableBKey
              AND b.TableBAttribute = 'Foo'
```

is the equivalent of...

```
SELECT *
FROM TableA a
       JOIN TableB b
              ON a.TableBKey = b.TableBKey
WHERE b.TableBAttribute = 'Foo'
```

…which means that any rows that don't meet the JOIN criteria will be excluded from the result set. Normally this is the desired result, but occasionally it can lead to eliminating rows unintentionally. Sometime the judicious use of an OUTER JOIN is required in order to maintain the presence of all of the required rows. A good example is the query used above involving overpayments. If in the final query that combined the separate subqueries we wrote…

```
SELECT
       a.AccountNo
       ,a.AdvanceSeqNumber
       ,a.SegmentSeqNo
       ,LatestBalance = a.Balance_ABS
       ,TotalOverpayments = b.Overpayments
FROM cte_BalanceInformation a
       JOIN cte_Overpayments b
              ON a.DIM_CO_ProdSegm_SSK = b.DIM_CO_ProdSegm_SSK
```

…we would be filtering out all Segments without overpayments from the result set. If that is the required output then there is no problem. However, if the final result set should include all Segments, irrespective of whether they have received overpayments then an OUTER JOIN must be used…

```
SELECT
       a.AccountNo
       ,a.AdvanceSeqNumber
       ,a.SegmentSeqNo
       ,LatestBalance = a.Balance_ABS
       ,TotalOverpayments = b.Overpayments
FROM cte_BalanceInformation a
       LEFT OUTER JOIN cte_Overpayments b
              ON a.DIM_CO_ProdSegm_SSK = b.DIM_CO_ProdSegm_SSK
```

In general then, when linking Fact tables/IF tables it is generally safer to combine the results using OUTER JOINS and them eliminate rows that are specifically not required in the WHERE clause of the query. The same can occasionally be the case when joining Dimension tables to Fact tables as well. For example, in the simple query…

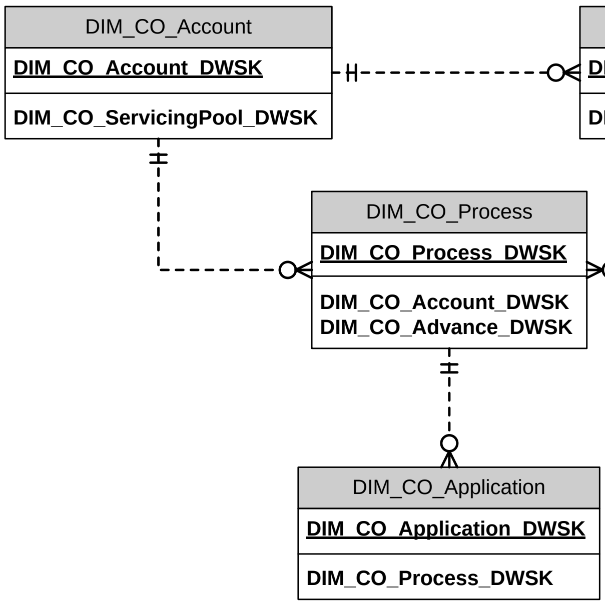
```
SELECT *
FROM FACT_CO_ApplicationProductAS a
       JOIN DIM_CO_Application b
              ON a.DIM_CO_Application_DWSK = b.DIM_CO_Application_DWSK
```

… you are automatically filtering out any Applications that do not currently have any Application Products selected (which can happen in some cases still at the DIP stage). If all Applications should be present in the result set then the query would need to be changed to

```
SELECT *
FROM FACT_CO_ApplicationProductAS a
       RIGHT OUTER JOIN DIM_CO_Application b
              ON a.DIM_CO_Application_DWSK = b.DIM_CO_Application_DWSK
```

4.2.5. Linking Origination & Servicing

The link between Origination and Servicing (i.e. the journey of an Application through to a live Account and Advance) is maintained via the table DIM_CO_Process. This contains instances of “Application Processes” that exist in the following schema:



As an Application completes, a new version of the DIM_CO_Process entry will be created that links to DIM_CO_Account and DIM_CO_Advance. This (due to the parent child relationship between them) will result in a new DIM_CO_Application version being created. Subsequent increments to the versions of DIM_CO_Account and/or DIM_CO_Advance will result in further increments to the DIM_CO_Process and DIM_CO_Application entries. As a result it is possible quite quickly for the data held in the Origination portion of the data warehouse schema (and more specifically the DWSKs linked to this data) to grow out of sync with those being generated by further increments generated by the Servicing portion of the schema. In other words, you cannot simply JOIN through from DIM_CO_Account to FACT_CO_Application and expect to find data based on DWSK joins alone. Instead, as with the general principle of separating Fact table queries, when linking Origination queries to Servicing queries, it is better to join based on the natural keys of the Application i.e. on the SSK.

For example, suppose that you had a requirement to bring back the Original loan amount applied for on the Application along with the current balance. For this you would need to retrieve data from the Servicing table FACT_CO_ProdSegm and from the Origination table FACT_CO_Application. To accomplish this, the best way is to split the query into two parts like so:

```
WITH cte_Origination AS
(
       SELECT
              b.DIM_CO_Application_SSK
              ,b.ApplicationRef
              ,a.LoanAmount_ABS
       FROM FACT_CO_Application a
              JOIN DIM_CO_Application b
                     ON a.DIM_CO_Application_DWSK = b.DIM_CO_Application_DWSK
),
cte_Servicing AS
(
       SELECT
              d.AccountNo
              ,c.AdvanceSeqNumber
              ,f.DIM_CO_Application_SSK
              ,Balance = SUM(a.Balance_ABS)
       FROM FACT_CO_ProdSegm a
              JOIN DIM_CO_ProdSegm b
                     ON a.DIM_CO_ProdSegm_DWSK = b.DIM_CO_ProdSegm_DWSK
              JOIN DIM_CO_Advance c
                     ON  c.DIM_CO_Advance_DWSK = b.DIM_CO_Advance_DWSK
              JOIN DIM_CO_Account d
                     ON d.DIM_CO_Account_DWSK = c.DIM_CO_Account_DWSK
              JOIN DIM_CO_Process e
                     ON e.DIM_CO_Account_DWSK = d.DIM_CO_Account_DWSK
                     AND e.SCDStartDate <= a.AsAt_DATE
                     AND e.SCDEndDate > a.AsAt_DATE
              JOIN DIM_CO_Application f
                     ON f.DIM_CO_Process_DWSK = e.DIM_CO_Process_DWSK
                     AND f.SCDStartDate <= a.AsAt_DATE
                     AND f.SCDEndDate > a.AsAt_DATE
       WHERE a.AsAt_DATE = <LatestDate>
       GROUP BY d.AccountNo, c.AdvanceSeqNumber, f.DIM_CO_Application_SSK
)
SELECT
       a.AccountNo
       ,a.AdvanceSeqNumber
       ,b.ApplicationRef
       ,ApplicationAmount = b.LoanAmount_ABS
       ,CurrentBalance = a.Balance
FROM cte_Servicing a
       JOIN cte_Origination b
              ON a.DIM_CO_Application_SSK = b.DIM_CO_Application_SSK
ORDER BY a.AccountNo, a.AdvanceSeqNumber
```

4.2.6. Bringing in other DPR Data (Synonyms)

There will occasionally be the need to bring in data that is not held in the data warehouse, by incorporating fields from the staging databases that are created each day by the ETL process. To make this exercise more straightforward, the data warehouse includes a series of synonyms that point to the tables in the latest copy of each of the main source databases mounted on the BI server. This allows a connection to be made to the DPR_DW database and from that one connection, to issues queries against multiple databases simultaneously.

For example, if you wanted to issue a query that retrieved some additional Application details, let's say the field "IsInsuranceAdvised" contained in the BackOfficeMortgage table dbo.morappfma_FMALoanDetails, then you would use a query like this:

```
SELECT
       a.ApplicationRef
       ,c.IsInsuranceAdvised
FROM DIM_CO_Application a
       JOIN [Latest_Orig_BOM].[dbo_morAppFma_FMAApplication] b
              ON CAST(b.Id AS VARCHAR(50)) = a.DIM_CO_Application_SSK
              COLLATE DATABASE_DEFAULT
       JOIN [Latest_Orig_BOM].[dbo_morAppFma_FMALoanDetails] c
              ON c.ApplicationId = b.Id
WHERE a.SCDStatus = 'C'
```

Note the use of COLLATE DATABASE_DEFAULT on the JOIN between DIM_CO_Application and the first synonym. This is sometime necessary if the collation on one of the databases being joined to differ from that of the data warehouse.

5. ORIGINATION TABLES

5.1. FACT_CO_AddressHistoryAS

5.1.1. Purpose and How it Works

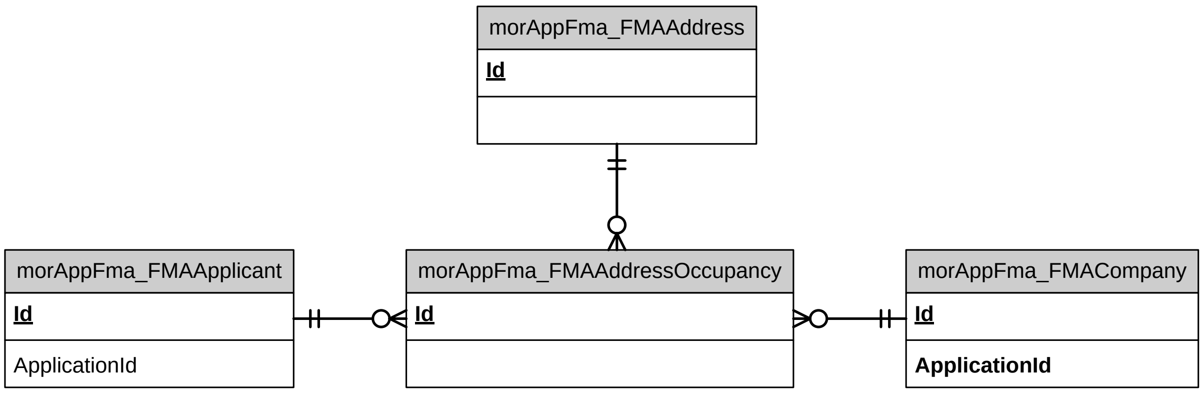
This table is used to track the history of addresses for Applicants and Customers. Each address is given an "Activated" and "Deactivated" date, corresponding to the Start and End date for that address.

It is an Accumulating Snapshot fact table, meaning that all data is updated to the latest version and in this case includes the foreign key DWSKs to be the latest version at the time the load or update is made, so there should only be one Customer DWSK per Customer SSK per Address.

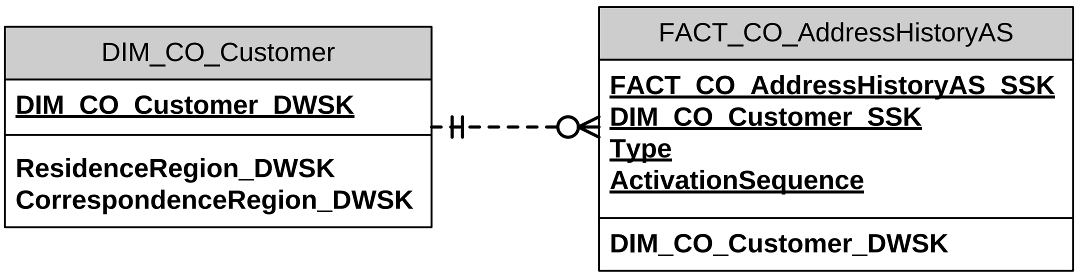
The column FACT_CO_AddressHistoryAS_SSK is a VARCHAR(50) representation of the Id field from the table morappFma_FMAAddressOccupancy and can be used to link out to the wider database using the synonyms.

5.1.2. Place in Conceptual Model

This table takes information from the main address tables in Origination. It links addresses to their respective Applicant records, including commercial Applicants.



5.1.3. Star Schema



5.1.4. Linked Dimension Tables

- DIM_CO_Customer

5.1.5. Query Hints

This table is straightforward to query as there should only be one result per Customer_SSK, per Address as the following shows:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,addr.<Desired Fields>
FROM FACT_CO_AddressHistoryAS addr
       JOIN DIM_CO_Customer cust
              ON addr.DIM_CO_Customer_DWSK = cust.DIM_CO_Customer_DWSK
```

If combining with other Fact table queries, the join should be made based on the DIM_CO_Customer_SSK and not the DWSK.

5.2. FACT_CO_ADVERSEDETAILSAS

5.2.1. Purpose and How it Works

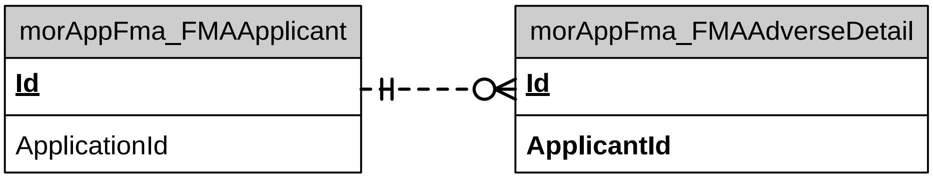
This table gives information supplied by the Applicant or Customer regarding adverse financial situations they may currently or historically have been in, e.g. Bankruptcy or CCJs.

It is an Accumulating Snapshot table and as such contains the latest version of data. This does not include the DWSK versions of the Customer SSK, hence it possible to retrieve "duplicate" results, unless care is taken to filter these out. See Query Hints for how to accomplish this.

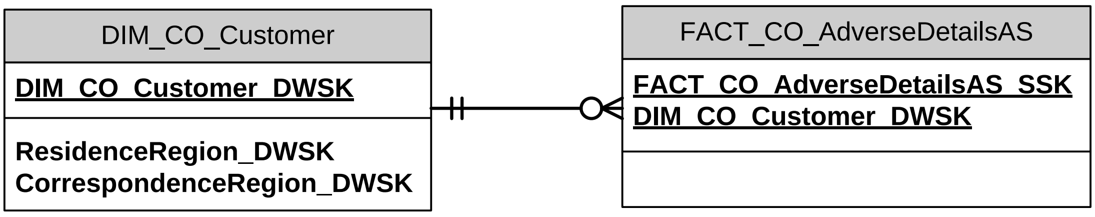
The field FACT_CO_AdverseDetails is a VARCHAR(50) version of the Id field in the Origination table morAppFma_FMAAdverseDetail. This can be used to link out to the BackOfficeMortgage database via the synonyms.

5.2.2. Place in Conceptual Model

This table draws information from the Adverse Details data in Origination and links them to the appropriate Applicant.



5.2.3. Star Schema



5.2.4. Linked Dimension Tables

- DIM_CO_Customer

5.2.5. Query Hints

To retrieve a single result per Customer_SSK and Adverse Detail, there are two methods to be used.

1) To return the "current" Customer version of the data then use the query:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,ad.<Desired Fields>
FROM FACT_CO_AdverseDetailsAS ad
       JOIN DIM_CO_Customer cust
              ON cust.DIM_CO_Customer_DWSK = ad.DIM_CO_Customer_DWSK
WHERE cust.SCDStatus = 'C'
```

2) To return a "historical" Customer version of the data at a particular point in time then use the query:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,ad.<Desired Fields>
FROM FACT_CO_AdverseDetailsAS ad
       JOIN DIM_CO_Customer cust
              ON cust.DIM_CO_Customer_DWSK = ad.DIM_CO_Customer_DWSK
WHERE cust.SCDStartDate <= @ReportDate
AND cust.SCDEndDate > @ReportDate
```

5.3. FACT_CO_AliasHistoryAS

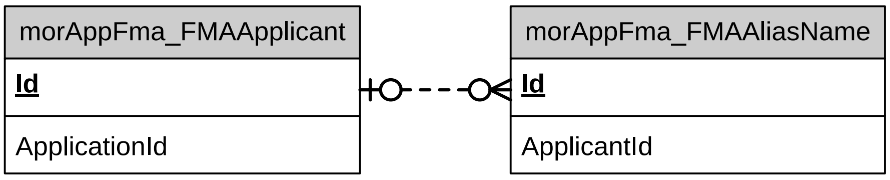
5.3.1. Purpose and How it Works

This table stores the history of aliases for Applicants. Each alias is given an Activation and Deactivation date corresponding to the Start and End dates for when the alias was applicable.

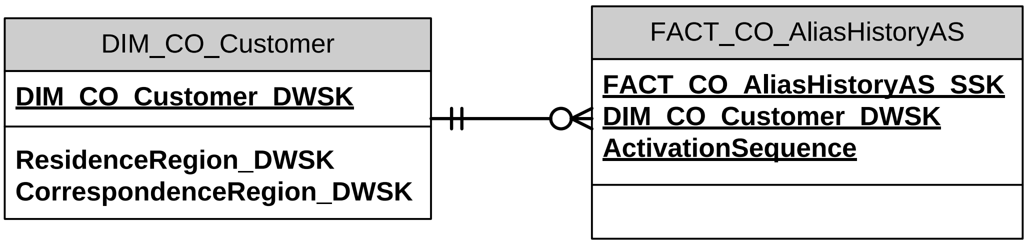
It is an Accumulating Snapshot table and as such only holds the latest version of the data with the exception of the DWSK version of the Customer, so it is possible to return "duplicate" entries in the result set unless care is taken to only return a single record per customer and address. See the Query Hints section on how to accomplish this.

5.3.2. Place in Conceptual Model

This table takes its data from the alias history available at the Applicant level in Origination



5.3.3. Star Schema



5.3.4. Linked Dimension Tables

- DIM_CO_Customer

5.3.5. Query Hints

To ensure that only one record per Customer and Alias is returned, use the either of the following approaches:

1) To return the current Customer data only then use a query like the following:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,ah.<Desired Fields>
FROM FACT_CO_AliasHistoryAS ah
       JOIN DIM_CO_Customer cust
              ON cust.DIM_CO_Customer_DWSK = ah.DIM_CO_Customer_DWSK
WHERE SCDStatus = 'C'
```

2) To return a historic customer version use a query like the following:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,ah.<Desired Fields>
FROM FACT_CO_AliasHistoryAS ah
       JOIN DIM_CO_Customer cust
              ON cust.DIM_CO_Customer_DWSK = ah.DIM_CO_Customer_DWSK
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

5.4. FACT_CO_Application

5.4.1. Purpose and How it Works

This table stores information that is relevant to the whole application, rather than to individual components of it. Data is sourced primarily from the Origination database tables FMAApplication and FMALoandetails. The primary key of the table is the column FACT_CO_Application_SSK which is a copy of the primary key of the FMAApplication table within BackOfficeMortgage - FMAApplication.Id. This allows connections to be made out to the FMAApplication table via the Synonyms.

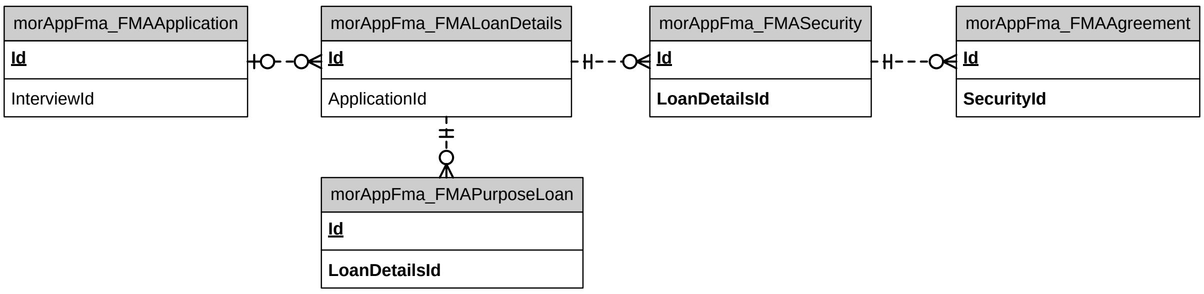
When this table is loaded by the ETL, any completed applications are filtered out of the result set. This is to keep any updates from the Origination system affecting the numeric values of the application once they have been completed (this has query writing implications – see Query Hints).

The table works as an Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture (until the application is complete). Note that this means that any previous values held in this table will be overwritten and no history kept e.g. if the Debt Consolidation portion of the loan changes from £10,000 to £20,000, only the £20,000 will be visible. The table is also always updated to point to the latest SCD “version” of the application, again until the point of completion, after which updates will cease to be registered. This means that historic SCD “versions” of the application will NOT have entries in FACT_CO_Application, only the latest at the moment of Completion.

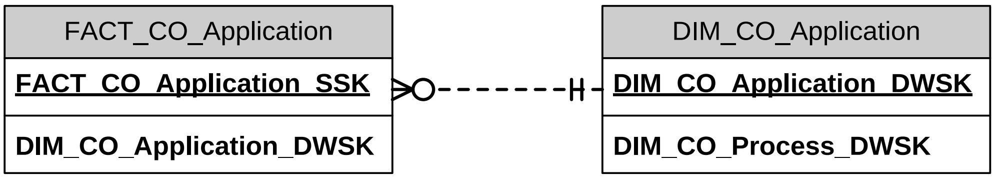
There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

5.4.2. Place in Conceptual Model

This table retrieves most of its data from the Application level of the Origination schema. It also interrogates the multiple possible “loan purposes” to establish if the Application has been made for the purpose of a Remortgage and if so, if any portion of that has been done for reasons of Debt Consolidation. It also checks any Agreements associated with the Security to establish the Shared Ownership Percentage (if any) that will be purchased as a result of the Application.



5.4.3. Star Schema



5.4.4. Linked Dimension Tables

DIM_CO_Application is the sole dimension table in this star schema and contains the descriptive information relevant to the applications. Unlike the FACT_CO_Application table, this table can be updated post completion either by Origination (when the status of the Application is set to Expire after 6 months) or by Servicing when a new Account SCD version is created. See DIM_CO_Application for further details.

5.4.5. Query Hints

When querying FACT_CO_Application there are two main things to bear in mind:

1. The table will have only one entry per application in Origination NOT one per application SCD version in the DIM_CO_Application table. The version linked to will be the latest up until the point of completion, after which any further increments to the DIM_CO_Application SCD versions will not be seen in the table.
2. The table is date independent, thus historical reporting of the values held within it is not possible.

In querying terms this both simplifies and complicates queries, depending on the desired result.

If the current view is all that is of interest (or at least the current view up until completion occurs) then the query is a simple one.

```
SELECT
       dapp.<DesiredColumns>
       ,fapp.<DesiredColumns>
FROM FACT_CO_Application fapp
       JOIN DIM_CO_Application dapp
              ON fapp.DIM_CO_Application_DWSK = dapp.DIM_CO_Application_DWSK
```

Note that no filter is used on DIM_CO_Application i.e. there is no restriction on SCDStatus = ‘C’. This is for two reasons:

1. The filter is unnecessary as there is only one record per application in the table and there is no need to compensate for SCD behaviour in the dimension. This is taken care of by the ETL.
2. By using a filter specifying on DIM_CO_Application that SCDStatus = ‘C’ there is the potential to eliminate records unintentionally e.g. if at the point of completion the latest DIM_CO_Application_DWSK was 5, it would remain at 5 for the lifetime of the table. However, post completion updates may occur to the DIM_CO_Application entry and as a result the “current” DIM_CO_Application_DWSK may be 6, 7 or higher but would not be present in FACT_CO_Application. Thus by filtering on SCDStatus = ‘C’, any applications that have incremented DWSK post completion will be eliminated from the result set of the query.

If for some reason it is necessary to report on some of the descriptive information from prior SCD versions at a specific point in time, then this can become more complicated, but is still possible.

```
DECLARE @ReportingDate INT

SELECT
        dapphistoric.<DesiredColumns>
        ,fapp.<DesiredColumns>
FROM FACT_CO_Application fapp
        JOIN DIM_CO_Application dapp
                ON fapp.DIM_CO_Application_DWSK = dapp.DIM_CO_Application_DWSK
        JOIN
        (
                SELECT
                        dapphistoric.DIM_CO_Application_SSK
                        ,dapphistoric.<DesiredColumns>
                FROM DIM_CO_Application dapphistoric
                WHERE dapphistoric.SCDStartDate <= @ReportingDate
                AND dapphistoric.SCDEndDate > @ReportingDate
        ) dapphistoric
                ON dapphistoric.DIM_CO_Application_SSK = dapp.DIM_CO_Application_SSK
```

The major flaw in this type of reporting is that the descriptive attributes will be relevant to that point in time, but the numeric values will not.

5.5. FACT_CO_ApplicationProductAS

5.5.1. Purpose and How it Works

This table stores information at the individual Product level. The primary key of the table is the column FACT_CO_ApplicationProduct_SSK which is a copy of the primary key of the FMAProduct table within BackOfficeMortgage - FMAProduct.Id. This allows connections to be made out to the FMAProduct table via the Synonyms.

It works in tandem with the table FACT_CO_ApplicationProductArchiveAS in which old Product selection are kept.

When this table is loaded by the ETL, any completed applications are filtered out of the result set. This is to keep any updates from the Origination system affecting the numeric values of the application once they have been completed.

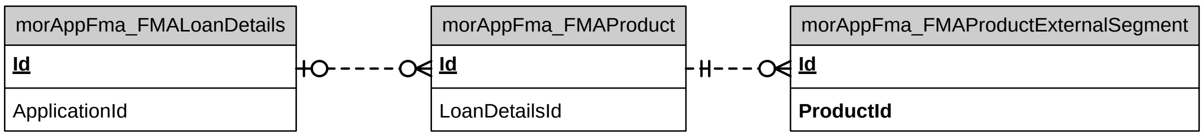
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture (until the application is complete) with the exception of the DIM_CO_Application_DWSKs and DIM_CO_ApplicationProduct_DWSKs. Note that this means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions of Application and Application Product present in the results.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

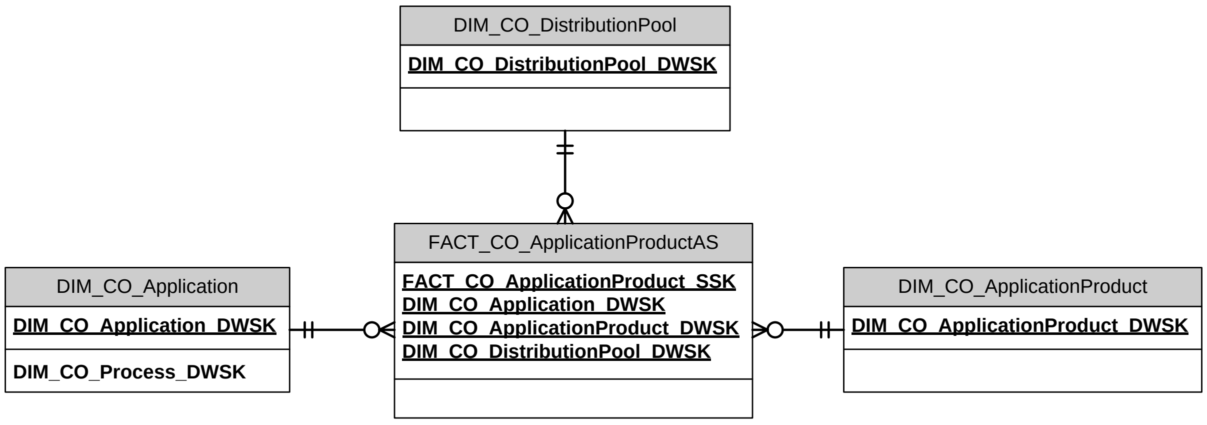
See Query Hints for the best way to obtain values from this table.

5.5.2. Place in Conceptual Model

This table draws its data primarily from the Product level of information held within the BackOfficeMortgage database in Origination. It is the main link between the Application and its Products.



5.5.3. Star Schema


DIM_CO_DistributionPool
DIM_CO_DistributionPool_DWSK
FACT_CO_ApplicationProductAS
FACT_CO_ApplicationProduct_SSK
DIM_CO_Application_DWSK
DIM_CO_ApplicationProduct_DWSK
DIM_CO_DistributionPool_DWSK
DIM_CO_Application
DIM_CO_Application_DWSK
DIM_CO_Process_DWSK
DIM_CO_ApplicationProduct
DIM_CO_ApplicationProduct_DWSK

5.5.4. Linked Dimension Tables

- DIM_CO_Application
- DIM_CO_ApplicationProduct
- DIM_CO_DistributionPool

5.5.5. Query Hints

When querying the table the main thing to bear in mind is to only obtain one SCD version per linked Dimension table. Also, because completed applications are filtered out of the results, using a query that specifies SCD status = ‘C’ may cause unintended filtering when further Application_DWSK versions have been created by the ongoing link with Servicing via the Process table. Thus to obtain correct results it is important to filter by a particular reporting date. For example:

```
SELECT
       dap.DIM_CO_Application_SSK
       ,dap.<Desired Fields>
       ,dapp.<Desired Fields>
       ,ddp.<Desired Fields>
       ,fapp.<Desired Fields>
FROM FACT_CO_ApplicationProductAS fapp
       RIGHT OUTER JOIN DIM_CO_Application dap
              ON fapp.DIM_CO_Application_DWSK = dap.DIM_CO_Application_DWSK
              AND dap.SCDStartDate <= @ReportDate
              AND dap.SCDEndDate > @ReportDate
       JOIN DIM_CO_ApplicationProduct dapp
              ON fapp.DIM_CO_ApplicationProduct_DWSK =
dapp.DIM_CO_ApplicationProduct_DWSK
              AND dapp.SCDStartDate <= @ReportDate
              AND dapp.SCDEndDate > @ReportDate
       JOIN DIM_CO_DistributionPool ddp
              ON ddp.DIM_CO_DistributionPool_DWSK = fapp.DIM_CO_DistributionPool_DWSK
              AND ddp.SCDStartDate <= @ReportDate
              AND ddp.SCDEndDate > @ReportDate
```

Note that the date used to filter on will also need to be less than the Completion Date of all Applications be queried but cannot be so far back as to eliminate Applications that took place after that Completion Date. Alternatively use the following approach which will give the latest result per Application:

```
SELECT
       DIM_CO_ApplicationProduct_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dap.DIM_CO_Application_SSK
              ,dap.<Desired Fields>
              ,dapp.<Desired Fields>
              ,ddp.<Desired Fields>
              ,fapp.<Desired Fields>
              ,ResultRank = ROW_NUMBER() OVER
                            (
                                   PARTITION BY
                                          dap.DIM_CO_Application_SSK
                                          , dapp.DIM_CO_ApplicationProduct_SSK
                                          , ddp.DIM_CO_DistributionPool_SSK
                                   ORDER BY
                                          dap.SCDStartDate DESC
                                          , dapp.SCDStartDate DESC
                                          , ddp.SCDStartDate DESC
                            )
       FROM FACT_CO_ApplicationProductAS fapp
              RIGHT OUTER JOIN DIM_CO_Application dap
                     ON fapp.DIM_CO_Application_DWSK = dap.DIM_CO_Application_DWSK
              JOIN DIM_CO_ApplicationProduct dapp
                     ON fapp.DIM_CO_ApplicationProduct_DWSK =
              dapp.DIM_CO_ApplicationProduct_DWSK
              JOIN DIM_CO_DistributionPool ddp
                     ON ddp.DIM_CO_DistributionPool_DWSK =
                     fapp.DIM_CO_DistributionPool_DWSK
) results
WHERE ResultRank = 1
```

5.6. FACT_CO_ApplicationProductArchiveAS

5.6.1. Purpose and How it Works

This table stores information at the individual Product level. The primary key of the table is the column FACT_CO_ApplicationProductArchive_SSK which is a copy of the primary key of the FMAProduct table within BackOfficeMortgage - FMAProduct.Id. Normally this would allow connections to be made out to the FMAProduct table via the Synonyms, however in this case that cannot be done as these records have been deleted in Origination.

It works in tandem with the table FACT_CO_ApplicationProductAS in which the current Product selection is kept.

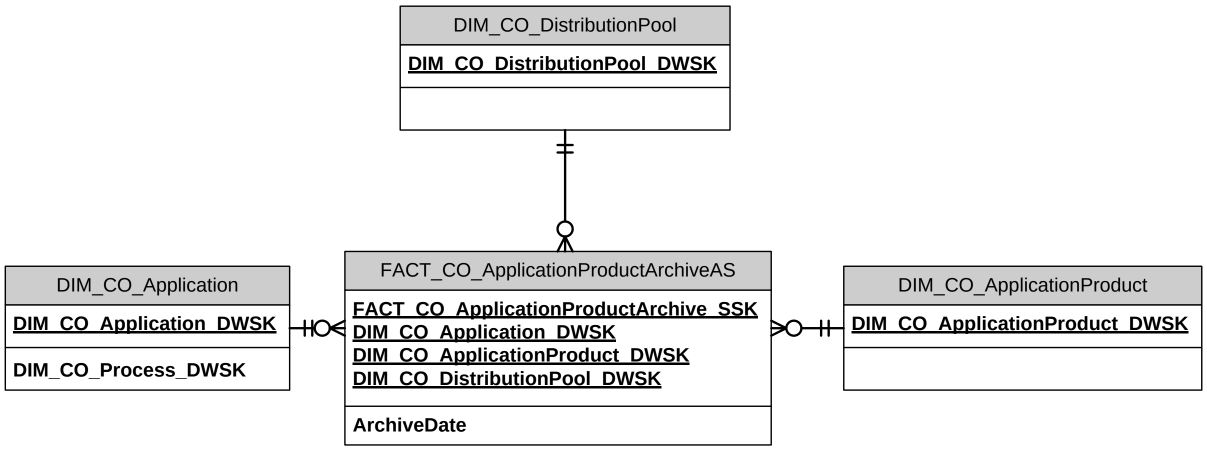
See Query Hints for the best way to obtain values from this table.

5.6.2. Place in Conceptual Model

This table draws purely from the Product level information within BackOfficeMortgage. It checks to see what the current Product information is per Application so that is can archive any that is out of date.

morAppFma_FMAProduct
Id
LoanDetailsId

5.6.3. Star Schema


DIM_CO_DistributionPool
DIM_CO_DistributionPool_DWSK
DIM_CO_Application
DIM_CO_Application_DWSK
DIM_CO_Process_DWSK
FACT_CO_ApplicationProductArchiveAS
FACT_CO_ApplicationProductArchive_SSK
DIM_CO_Application_DWSK
DIM_CO_ApplicationProduct_DWSK
DIM_CO_DistributionPool_DWSK
ArchiveDate
DIM_CO_ApplicationProduct
DIM_CO_ApplicationProduct_DWSK

5.6.4. Linked Dimension Tables

- DIM_CO_Application
- DIM_CO_ApplicationProduct
- DIM_CO_DistributionPool

5.6.5. Query Hints

When querying the table the main thing to bear in mind is to only obtain one SCD version per linked Dimension table. Also, because completed applications are filtered out of the results, using a query that specifies SCD status = ‘C’ may cause unintended filtering when further Application_DWSK versions have been created by the ongoing link with Servicing via the Process table. Thus to obtain correct results it is important to filter by a particular reporting date. For example:

```
SELECT
       dap.DIM_CO_Application_SSK
       ,dap.<Desired Fields>
       ,dapp.<Desired Fields>
       ,ddp.<Desired Fields>
       ,fapp.<Desired Fields>
FROM FACT_CO_ApplicationProductArchiveAS fapp
       RIGHT OUTER JOIN DIM_CO_Application dap
              ON fapp.DIM_CO_Application_DWSK = dap.DIM_CO_Application_DWSK
              AND dap.SCDStartDate <= @ReportDate
              AND dap.SCDEndDate > @ReportDate
       JOIN DIM_CO_ApplicationProduct dapp
              ON fapp.DIM_CO_ApplicationProduct_DWSK =
dapp.DIM_CO_ApplicationProduct_DWSK
              AND dapp.SCDStartDate <= @ReportDate
              AND dapp.SCDEndDate > @ReportDate
       JOIN DIM_CO_DistributionPool ddp
              ON ddp.DIM_CO_DistributionPool_DWSK = fapp.DIM_CO_DistributionPool_DWSK
              AND ddp.SCDStartDate <= @ReportDate
              AND ddp.SCDEndDate > @ReportDate
```

Note that the date used to filter on will also need to be less than the Completion Date of all Applications be queried but cannot be so far back as to eliminate Applications that took place after that Completion Date. Alternatively use the following approach which will give the latest result per Application:

```
SELECT
       DIM_CO_ApplicationProduct_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dap.DIM_CO_Application_SSK
              ,dap.<Desired Fields>
              ,dapp.<Desired Fields>
              ,ddp.<Desired Fields>
              ,fapp.<Desired Fields>
              ,ResultRank = ROW_NUMBER() OVER
                            (
                                   PARTITION BY
                                          dap.DIM_CO_Application_SSK
                                          , dapp.DIM_CO_ApplicationProduct_SSK
                                          , ddp.DIM_CO_DistributionPool_SSK
                                   ORDER BY
                                          dap.SCDStartDate DESC
                                          , dapp.SCDStartDate DESC
                                          , ddp.SCDStartDate DESC
                            )
       FROM FACT_CO_ApplicationProductArchiveAS fapp
              RIGHT OUTER JOIN DIM_CO_Application dap
                     ON fapp.DIM_CO_Application_DWSK = dap.DIM_CO_Application_DWSK
              JOIN DIM_CO_ApplicationProduct dapp
                     ON fapp.DIM_CO_ApplicationProduct_DWSK =
              dapp.DIM_CO_ApplicationProduct_DWSK
              JOIN DIM_CO_DistributionPool ddp
                     ON ddp.DIM_CO_DistributionPool_DWSK =
                     fapp.DIM_CO_DistributionPool_DWSK
) results
WHERE ResultRank = 1
```

5.7. FACT_CO_APPLICATIONREPAYMENTVEHICLEAS

5.7.1. Purpose and How it Works

This table stores information on known Repayment Vehicles being used in tandem with the Application. Part of the primary key of the table is the column FACT_CO_ApplicationRepaymentVehicleAS_SSK which is a copy of the primary key of the FMARepaymentVehicle table within BackOfficeMortgage - FMARepaymentVehicle.Id. This allows connections to be made out to the FMARepaymentVehicle table via the Synonyms.

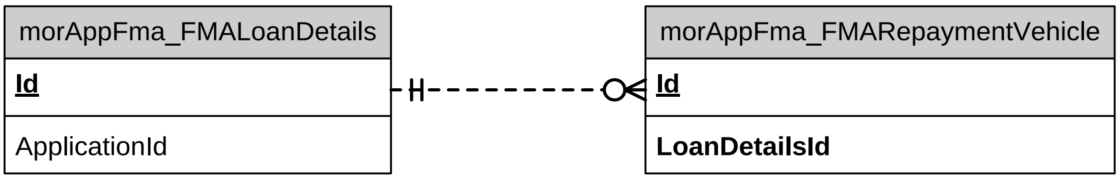
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

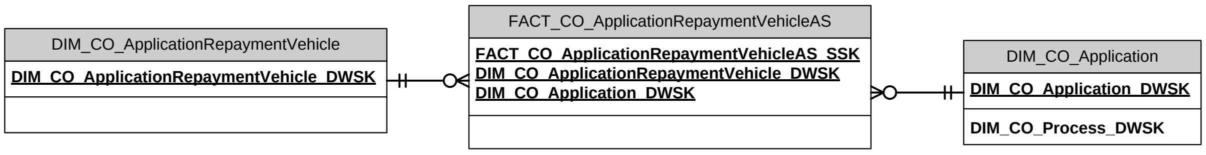
See Query Hints for the best way to obtain values from this table.

5.7.2. Place in Conceptual Model

This table draws from the Repayment Vehicle information from within Origination associated with the Application.



5.7.3. Star Schema


DIM_CO_ApplicationRepaymentVehicle
DIM_CO_ApplicationRepaymentVehicle_DWSK
FACT_CO_ApplicationRepaymentVehicleAS
FACT_CO_ApplicationRepaymentVehicleAS_SSK
DIM_CO_ApplicationRepaymentVehicle_DWSK
DIM_CO_Application_DWSK
DIM_CO_Application
DIM_CO_Application_DWSK
DIM_CO_Process_DWSK

5.7.4. Linked Dimension Tables

- DIM_CO_Application
- DIM_CO_ApplicationRepaymentVehicle

5.7.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dap.DIM_CO_Application_SSK
       ,dap.<Desired Fields>
       ,darv.<Desired Fields>
       ,farv.<Desired Fields>
FROM FACT_CO_ApplicationRepaymentVehicleAS farv
       JOIN DIM_CO_Application dap
              ON dap.DIM_CO_Application_DWSK = farv.DIM_CO_Application_DWSK
              AND dap.SCDStatus = 'C'
       JOIN DIM_CO_ApplicationRepaymentVehicle darv
              ON farv.DIM_CO_ApplicationRepaymentVehicle_DWSK =
              darv.DIM_CO_ApplicationRepaymentVehicle_DWSK
              AND darv.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       dap.DIM_CO_Application_SSK
       ,dap.<Desired Fields>
       ,darv.<Desired Fields>
       ,farv.<Desired Fields>
FROM FACT_CO_ApplicationRepaymentVehicleAS farv
       JOIN DIM_CO_Application dap
              ON dap.DIM_CO_Application_DWSK = farv.DIM_CO_Application_DWSK
              AND dap.SCDStartDate <= @ReportDate
              AND dap.SCDEndDate > @ReportDate
       JOIN DIM_CO_ApplicationRepaymentVehicle darv
              ON farv.DIM_CO_ApplicationRepaymentVehicle_DWSK =
darv.DIM_CO_ApplicationRepaymentVehicle_DWSK
              AND darv.SCDStartDate <= @ReportDate
              AND darv.SCDEndDate > @ReportDate
```

5.8. FACT_CO_AssetSecurityCharges

5.8.1. Purpose and How it Works

This table stores information on existing Charges against. Part of the primary key of the table is the column FACT_CO_AssetSecurityChargesAS_SSK which is a copy of the primary key of the FMASecurityCharge table within BackOfficeMortgage - FMASecurityCharge.Id. This allows connections to be made out to the FMASecurityCharge table via the Synonyms.

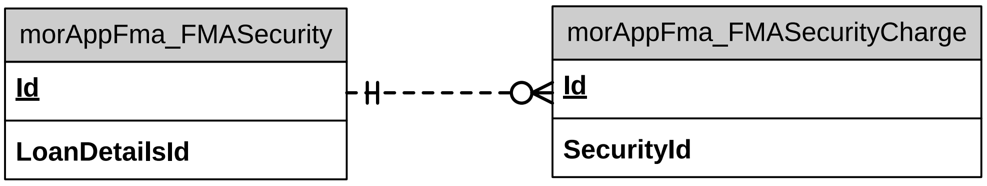
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

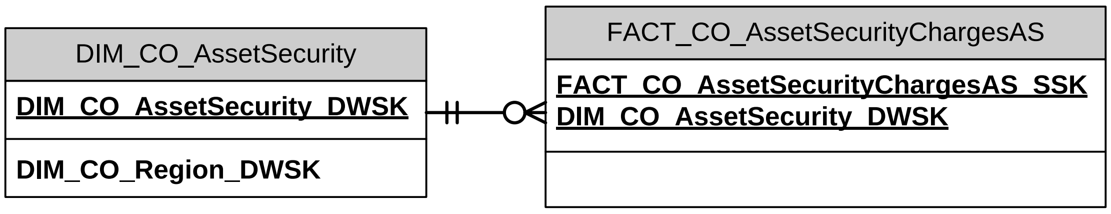
See Query Hints for the best way to obtain values from this table.

5.8.2. Place in Conceptual Model

This table extracts any details of other Charges held against the Security on the Application



5.8.3. Star Schema


DIM_CO_AssetSecurity
DIM_CO_AssetSecurity_DWSK
DIM_CO_Region_DWSK
FACT_CO_AssetSecurityChargesAS
FACT_CO_AssetSecurityChargesAS_SSK
DIM_CO_AssetSecurity_DWSK

5.8.4. Linked Dimension Tables

- DIM_CO_AssetSecurity

5.8.5. Query Hints

The main things to bear in mind when querying this table is to take care to only return one record per Security and Repayment Vehicle. To do this either return the most current records using a query like:

```
SELECT
       das.DIM_CO_AssetSecurity_SSK
       ,das.<Desired Fields>
       ,fasc.<Desired Fields>
FROM FACT_CO_AssetSecurityChargesAS fasc
       JOIN DIM_CO_AssetSecurity das
              ON das.DIM_CO_AssetSecurity_DWSK = fasc.DIM_CO_AssetSecurity_DWSK
              AND das.SCDStatus = 'C'
```

Alternatively retrieve data from a specific moment in time using a query like:

```
SELECT
       das.DIM_CO_AssetSecurity_SSK
       ,das.<Desired Fields>
       ,fasc.<Desired Fields>
FROM FACT_CO_AssetSecurityChargesAS fasc
       JOIN DIM_CO_AssetSecurity das
              ON das.DIM_CO_AssetSecurity_DWSK = fasc.DIM_CO_AssetSecurity_DWSK
              AND das.SCDStartdate <= @ReportDate
              AND das.SCDEndDate > @ReportDate
```

5.9. FACT_CO_CommitmentAS

5.9.1. Purpose and How it Works

This table stores information on existing financial Commitments an Applicant has. Part of the primary key of the table is the column FACT_CO_CommitementAS_SSK which is a copy of the primary key of the FMACommitment table within BackOfficeMortgage - FMACommitment.Id. This allows connections to be made out to the FMASecurityCharge table via the Synonyms.

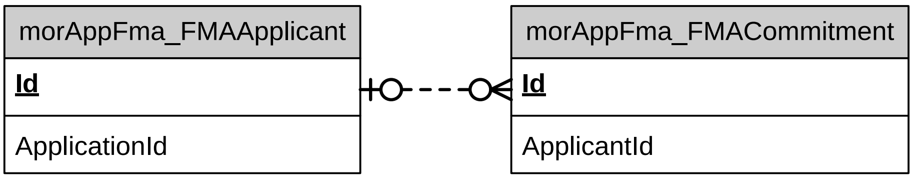
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

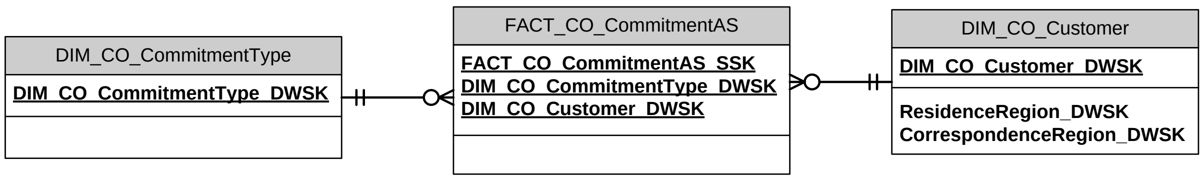
See Query Hints for the best way to obtain values from this table.

5.9.2. Place in Conceptual Model

This table draws information from the Commitments that are linked to the Applicants in Origination



5.9.3. Star Schema



5.9.4. Linked Dimension Tables

- DIM_CO_CommitmentType
- DIM_CO_Customer

5.9.5. Query Hints

The main thing to bear in mind when querying this table is to take care to eliminate multiple records per SSK within the result sets due to SCD versions. This can be done in two ways. Either using the current versions of each dimension, using a query like:

```
SELECT
       dcust.DIM_CO_Customer_SSK
       ,dcust.<Desired Fields>
       ,dcom.<Desired Fields>
       ,fcom.<Desired Fields>
FROM FACT_CO_CommitmentAS fcom
       JOIN DIM_CO_CommitmentType dcom
              ON dcom.DIM_CO_CommitmentType_DWSK = fcom.DIM_CO_CommitmentType_DWSK
              AND dcom.SCDStatus = 'C'
       JOIN DIM_CO_Customer dcust
              ON dcust.DIM_CO_Customer_DWSK = fcom.DIM_CO_Customer_DWSK
              AND dcust.SCDStatus = 'C'
```

Or a query that targets a specific moment in time like this:

```
SELECT
       dcust.DIM_CO_Customer_SSK
       ,dcust.<Desired Fields>
       ,dcom.<Desired Fields>
       ,fcom.<Desired Fields>
FROM FACT_CO_CommitmentAS fcom
       JOIN DIM_CO_CommitmentType dcom
              ON dcom.DIM_CO_CommitmentType_DWSK = fcom.DIM_CO_CommitmentType_DWSK
              AND dcom.SCDStartDate <= @ReportDate
              AND dcom.SCDEndDate > @ReportDate
       JOIN DIM_CO_Customer dcust
              ON dcust.DIM_CO_Customer_DWSK = fcom.DIM_CO_Customer_DWSK
              AND dcust.SCDStartDate <= @ReportDate
              AND dcust.SCDEndDate > @ReportDate
```

5.10. FACT_CO_CoreAffordabilityApplicantDataAS

5.10.1. Purpose and How it Works

This table holds core Affordability values, as dictated by such reports as PSD001 and MLAR. It sources its data from a combination of data already within the Warehouse (see FACT_CO_CustomerIncomeAS and FACT_CO_DocumentXML).

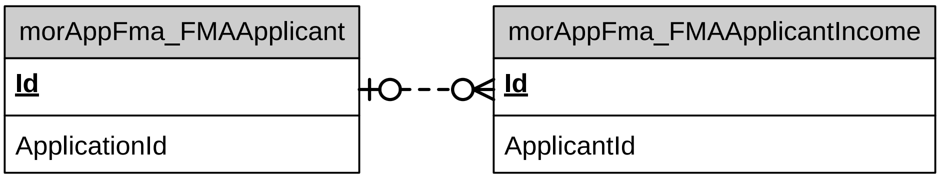
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. This means that any previous values held in this table will be overwritten and no history kept. In practice this will happen rarely with this table as existing XML values are unlikely to change, although income values might. What needs to be thought about when querying this table is the presence of multiple XML results for the same Customer i.e. that multiple Affordability results may have been generated for the Applicants on the Application.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

See Query Hints for the best way to obtain values from this table.

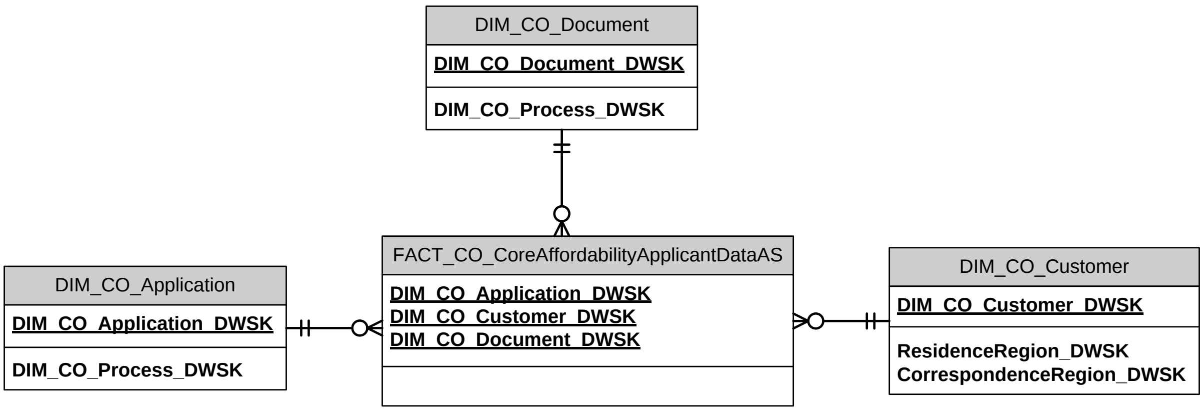
5.10.2. Place in Conceptual Model

This table actually draws information from tables already in the Data Warehouse, however the ultimate source of those tables comes from two place. Applicant Income is drawn from the equivalent table in Origination…



…with the exception of Total Verified Net Income which is always taken from the Afforability Data that is produced when making lending decisions. Each lender will have a different method of calculating this, hence the source of this is the result of a calculation stored in the Affordability XML Document.

5.10.3. Star Schema



5.10.4. Linked Dimension Tables

- DIM_CO_Document
- DIM_CO_Application
- DIM_CO_Customer

5.10.5. Query Hints

The main thing to bear in mind when querying this table is the need to negate the effects of multiple Affordability calculations being run on an Application. The DWSKs are not updated on this table, so there is no need to take into Account the effect of SCD variants. To filter out all but one Affordability calculation result, query the latest available like this:

```
SELECT
       DIM_CO_Application_SSK
       ,DIM_CO_Customer_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dapp.DIM_CO_Application_SSK
              ,dcust.DIM_CO_Customer_SSK
              ,fca.<Desired Fields>
              ,ResultRank = ROW_NUMBER()
                                   OVER
                                   (
                                          PARTITION BY
                                                 dapp.DIM_CO_Application_SSK
                                                 , dcust.DIM_CO_Customer_SSK
                                          ORDER BY
                                                 ddoc.DateCreated DESC
                                   )
       FROM FACT_CO_CoreAffordabilityApplicantDataAS fca
              JOIN DIM_CO_Application dapp
```

```
                        ON dapp.DIM_CO_Application_DWSK = fca.DIM_CO_Application_DWSK
                JOIN DIM_CO_Customer dcust
                        ON dcust.DIM_CO_Customer_DWSK = fca.DIM_CO_Customer_DWSK
                JOIN DIM_CO_Document ddoc
                        ON ddoc.DIM_CO_Document_DWSK = fca.DIM_CO_Document_DWSK
) a
WHERE ResultRank = 1
```

5.11. FACT_CO_CoreAffordabilityApplicationDataAS

5.11.1. Purpose and How it Works

This table holds core Affordability values, as dictated by such reports as PSD001 and MLAR. It sources its data from data already within the Warehouse (see FACT_CO_DocumentXML).

The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. This means that any previous values held in this table will be overwritten and no history kept. In practice this will happen rarely with this table as existing XML values are unlikely to change. What needs to be thought about when querying this table is the presence of multiple XML results for the same Application i.e. that multiple Affordability results may have been generated for the Applicants on the Application.

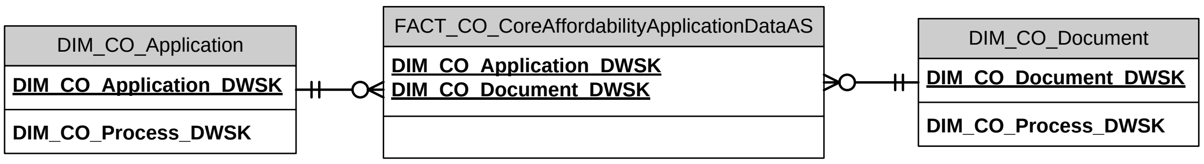
There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

See Query Hints for the best way to obtain values from this table.

5.11.2. Place in Conceptual Model

As with the Applicant level Afforability Data, this table sources its information from the data produced from the Affordability model used when making lending decisions. Each lender will have a different model, hence the source of this is the Affordability XML Document.

5.11.3. Star Schema



5.11.4. Linked Dimension Tables

- DIM_CO_Application
- DIM_CO_Document

5.11.5. Query Hints

The main thing to bear in mind when querying this table is the need to negate the effects of multiple Affordability calculations being run on an Application. The DWSKs are not updated on this table, so there is no need to take into Account the effect of SCD variants. To filter out all but one Affordability calculation result, query the latest available like this:

```
SELECT
       DIM_CO_Application_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dapp.DIM_CO_Application_SSK
              ,fca.<Desired Fields>
              ,ResultRank = ROW_NUMBER()
                                          OVER
                                          (
                                                 PARTITION BY
                                                        dapp.DIM_CO_Application_SSK
                                                 ORDER BY
                                                        ddoc.DateCreated DESC
                                          )
       FROM FACT_CO_CoreAffordabilityApplicationDataAS fca
              JOIN DIM_CO_Application dapp
                     ON dapp.DIM_CO_Application_DWSK = fca.DIM_CO_Application_DWSK
              JOIN DIM_CO_Document ddoc
                     ON ddoc.DIM_CO_Document_DWSK = fca.DIM_CO_Document_DWSK
) a
WHERE ResultRank = 1
```

5.12. FACT_CO_CreditReference

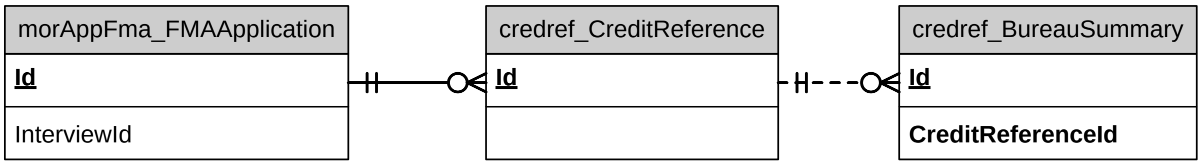
5.12.1. Purpose and How it Works

This table stores information sourced from the Credit Reference search. It works as a Transactional Fact table i.e. new records are added as new credit searches take place. Although based on the Credit Reference data held in the BackOfficeMortgage database, it only contains a subset of it. For the full Credit Reference response, the documents held in FACT_CO_DocumentXML should be used.

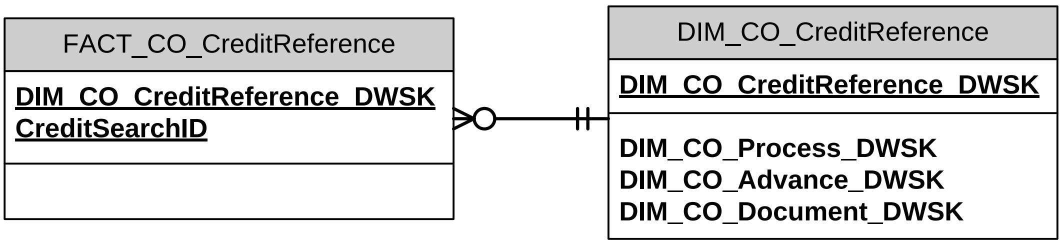
See Query Hints for the best way to obtain values from this table.

5.12.2. Place in Conceptual Model

This table sources it data from the Bureau Summary data from the Origination Credit Reference model.



5.12.3. Star Schema



5.12.4. Linked Dimension Tables

- DIM_CO_CreditReference

5.12.5. Query Hints

The latest Credit Reference data can be retrieved by using a query like:

```
SELECT
       DIM_CO_Application_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dapp.DIM_CO_Application_SSK
              ,fcr.<Desired Fields>
              ,ResultRank = ROW_NUMBER()
                                          OVER
                                          (
                                                 PARTITION BY
                                                        dapp.DIM_CO_Application_SSK
                                                 ORDER BY
                                                 dcr.CreditReferenceStartDate DESC
                                          )
       FROM FACT_CO_CreditReference fcr
              JOIN DIM_CO_CreditReference dcr
                     ON dcr.DIM_CO_CreditReference_DWSK =
                     fcr.DIM_CO_CreditReference_DWSK
              JOIN DIM_CO_Process dproc
                     ON dproc.DIM_CO_Process_DWSK = dcr.DIM_CO_Process_DWSK
              JOIN DIM_CO_Application dapp
                     ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
                     AND dapp.SCDStartDate <= CreditReferenceStartDate
                     AND dapp.SCDEndDate > CreditReferenceStartDate
) a
WHERE ResultRank = 1
```

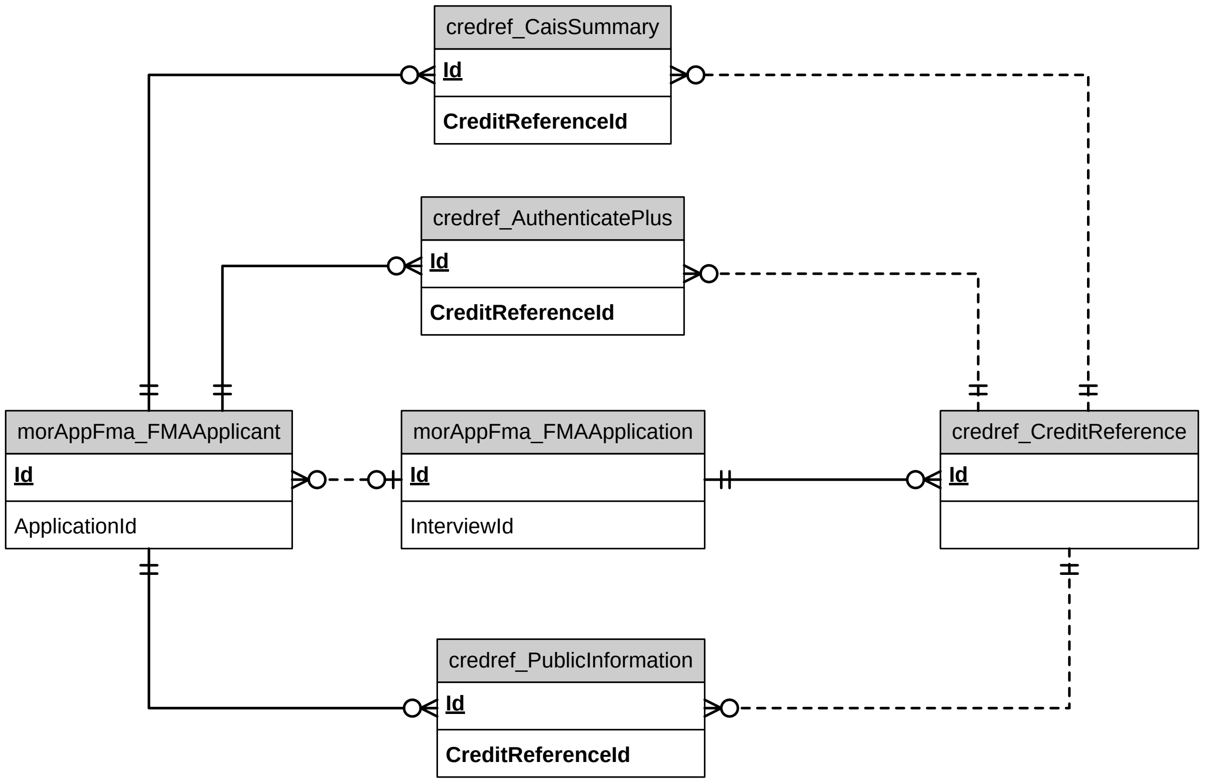
5.13. FACT_CO_CreditReferenceCustomer

5.13.1. Purpose and How it Works

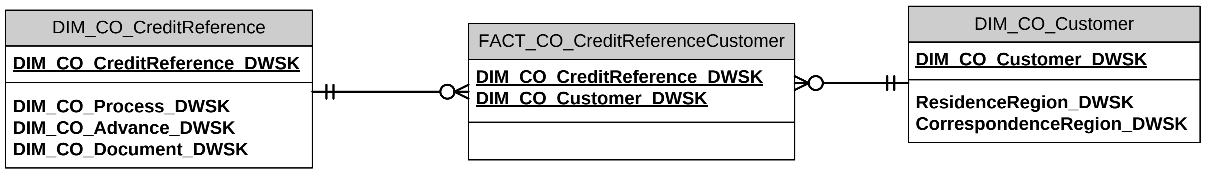
This table stores information sourced from the Credit Reference search. It works as a Transactional Fact table i.e. new records are added as new credit searches take place. Although based on the Credit Reference data held in the BackOfficeMortgage database, it only contains a subset of it. For the full Credit Reference response, the documents held in FACT_CO_DocumentXML should be used.

5.13.2. Place in Conceptual Model

This table stores Applicant level Credit Reference Data sourced from various areas of the Origination Credit Reference model, but with most of it coming from the CAIS summary data (or the equivalent information from other credit reference agencies).



5.13.3. Star Schema



5.13.4. Linked Dimension Tables

- DIM_CO_CreditReference
- DIM_CO_Customer

5.13.5. Query Hints

As with FACT_CO_CreditReference, the main thing to bear in mind when querying this table is to only retrieve the results of one Credit Search per Application and Applicant. This can be done using a query like the following which retrieves the latest result:

```
SELECT
       DIM_CO_Application_SSK
       ,DIM_CO_Customer_SSK
       ,<Desired Fields>
FROM
(
       SELECT
              dapp.DIM_CO_Application_SSK
              ,dcust.DIM_CO_Customer_SSK
              ,fcr.<Desired Fields>
              ,ResultRank = ROW_NUMBER()
                                          OVER
                                          (
                                                 PARTITION BY
                                                        dapp.DIM_CO_Application_SSK
                                                        ,dcust.DIM_CO_Customer_SSK
                                                 ORDER BY
                                                        dcr.CreditReferenceStartDate
DESC
                                          )
       FROM FACT_CO_CreditReferenceCustomer fcr
              JOIN DIM_CO_CreditReference dcr
                     ON dcr.DIM_CO_CreditReference_DWSK =
fcr.DIM_CO_CreditReference_DWSK
              JOIN DIM_CO_Customer dcust
                     ON dcust.DIM_CO_Customer_DWSK = fcr.DIM_CO_Customer_DWSK
              JOIN DIM_CO_Process dproc
                     ON dproc.DIM_CO_Process_DWSK = dcr.DIM_CO_Process_DWSK
              JOIN DIM_CO_Application dapp
                     ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
                     AND dapp.SCDStartDate <= CreditReferenceStartDate
                     AND dapp.SCDEndDate > CreditReferenceStartDate

) a
WHERE ResultRank = 1
```

5.14. FACT_CO_CustomerIncomeAS

5.14.1. Purpose and How it Works

This table stores information on the Customer income reported during the Application process. Part of the primary key of the table is the column FACT_CO_CustomerIncomeAS_SSK which is a copy of the primary key of the FMAApplicantIncome table within BackOfficeMortgage - FMAApplicantIncome.Id. This allows connections to be made out to the FMAApplicantIncome table via the Synonyms.

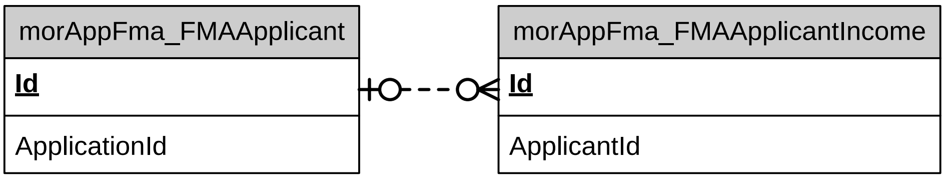
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture with the exception of the DWSKs. Note that this means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions of Application and Application Product present in the results.

There is no date dependency in this table i.e. only the current view is shown, thus historical reporting in its strictest sense is not possible from this table.

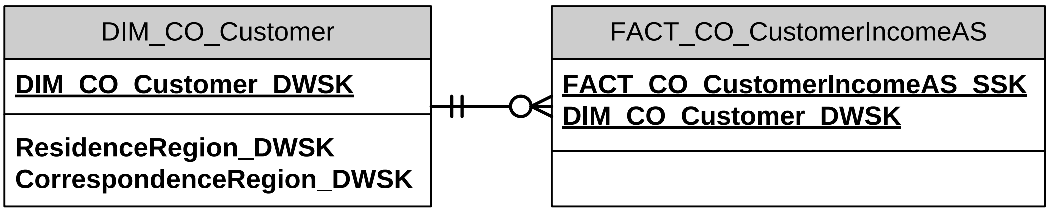
See Query Hints for the best way to obtain values from this table.

5.14.2. Place in Conceptual Model

This table sources its data from the Applicant Income information in Origination.



5.14.3. Star Schema



5.14.4. Linked Dimension Tables

- DIM_CO_Customer

5.14.5. Query Hints

The main thing to bear in mind when querying this table is to take care to filter out multiple records per Customer_DWSK. To do this you can either query on the current SCD versions like this:

```
SELECT
       dcust.DIM_CO_Customer_SSK
       ,fci.<Desired Fields>
FROM FACT_CO_CustomerIncomeAS fci
       JOIN DIM_CO_Customer dcust
              ON dcust.DIM_CO_Customer_DWSK = fci.DIM_CO_Customer_DWSK
              AND dcust.SCDStatus = 'C'
```

Or at a specific point in time like this:

```
SELECT
       dcust.DIM_CO_Customer_SSK
       ,fci.<Desired Fields>
FROM FACT_CO_CustomerIncomeAS fci
       JOIN DIM_CO_Customer dcust
              ON dcust.DIM_CO_Customer_DWSK = fci.DIM_CO_Customer_DWSK
              AND dcust.SCDStartDate <= @ReportDate
              AND dcust.SCDEndDate > @ReportDate
```

5.15. FACT_CO_DecisionHistoryTrx

5.15.1. Purpose and How it Works

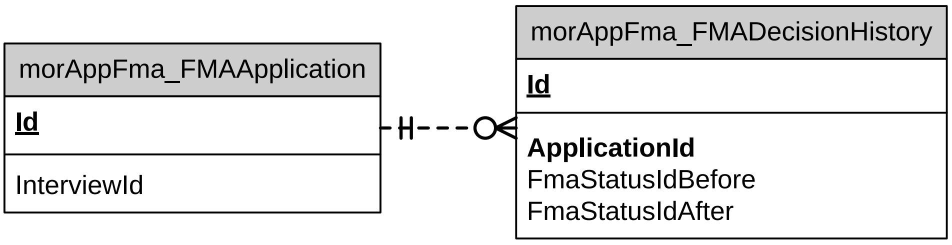
This table stores the record of the Decision taken on an Application. It acts as a Transaction fact table and enters new data as and when a decision is made. The DWSKs at the time of the transaction are the ones used.

Part of the primary key of the table is the column FACT_CO_DecisionHistoryTrx_SSK which is based on the primary key of the BackOfficeMortgage table FMADecisionHistory – FMADecisionHistory.Id. This can be used to connect back to the source system via the synonyms.

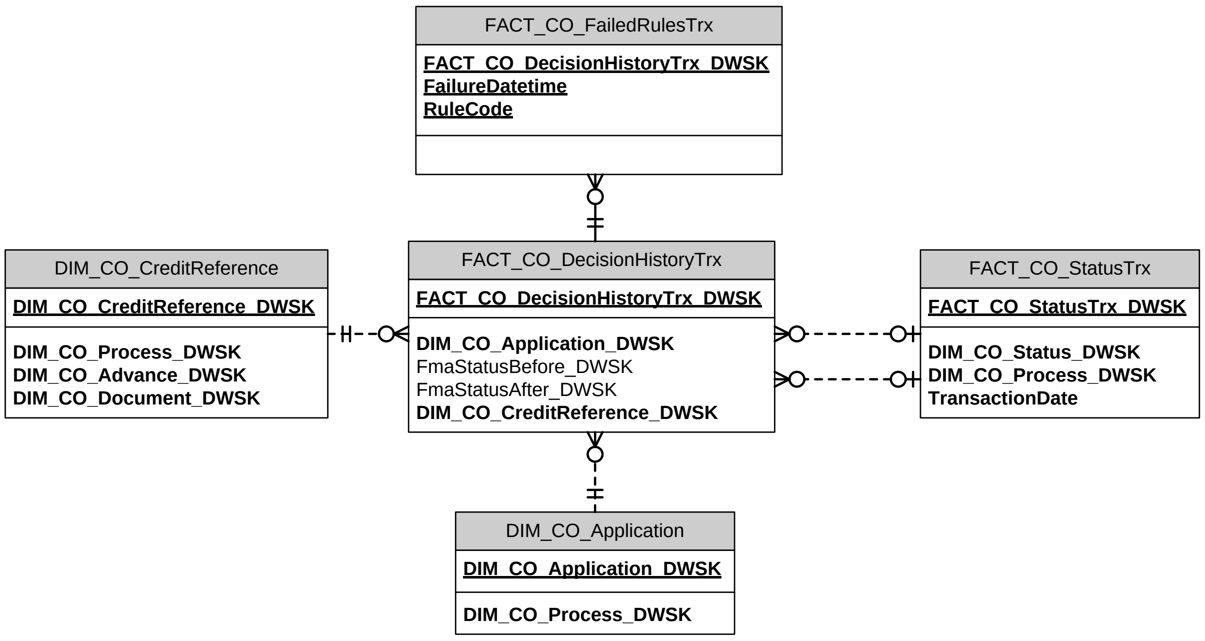
The table allows you to link to the appropriate Status action that was related to the Decision via the links to the Status before and after. The Failed Rules table also links to this table so that the rule failures related to a decision can be easily seen.

5.15.2. Place in Conceptual Model

This table takes all of its information from the Decision History record in Origination.



5.15.3. Star Schema



5.15.4. Linked Dimension Tables

- DIM_CO_CreditReference
- DIM_CO_Application

5.15.5. Query Hints

This table is very straightforward to query as only one DWSK version per decision is recorded. As with all fact tables, results should be collated based on the SSK.

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,fdh.<Desired Fields>
FROM FACT_CO_DecisionHistoryTrx fdh
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Application_DWSK = fdh.DIM_CO_Application_DWSK
```

5.16. FACT_CO_Deposit

5.16.1. Purpose and How it Works

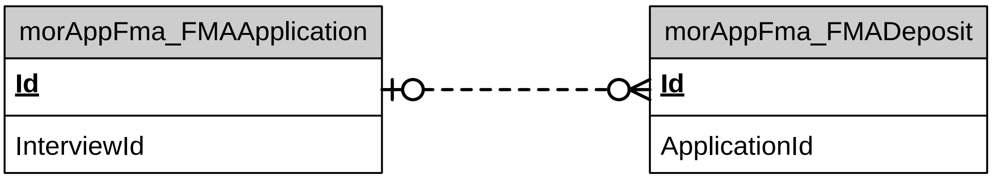
This table stores information on deposits put down by Applicants during the mortgage Applications. The primary key of the table is the column FACT_CO_Deposit_SSK which is a copy of the primary key of the FMADeposit table within BackOfficeMortgage - FMADeposit.Id. This allows connections to be made out to the FMAApplication table via the Synonyms.

When this table is loaded by the ETL, any completed applications are filtered out of the result set. This is to keep any updates from the Origination system affecting the numeric values of the application once they have been completed (this has query writing implications – see Query Hints).

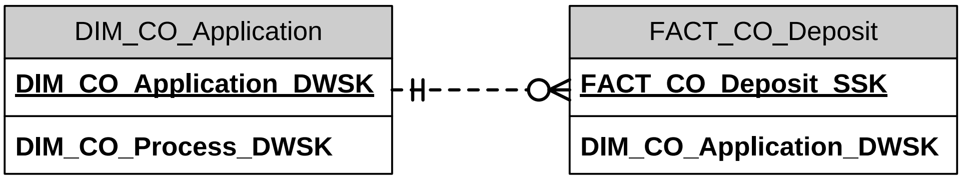
The table works as an Transactional Fact table i.e. new records are added as new Deposits are added and DWSKs are only stored once and not updated or multiplied.

5.16.2. Place in Conceptual Model

This table sources its data from the Deposit data in Origination



5.16.3. Star Schema



5.16.4. Linked Dimension Tables

- DIM_CO_Application

5.16.5. Query Hints

This table is very straightforward to query as only one DWSK version per decision is recorded. As with all fact tables, results should be collated based on the SSK.

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,fd.<Desired Fields>
FROM FACT_CO_Deposit fd
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Application_DWSK = fd.DIM_CO_Application_DWSK
```

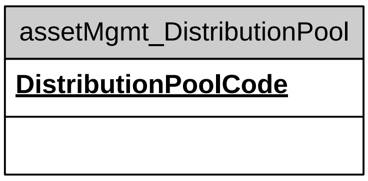
5.17. FACT_CO_DistributionPool

5.17.1. Purpose and How it Works

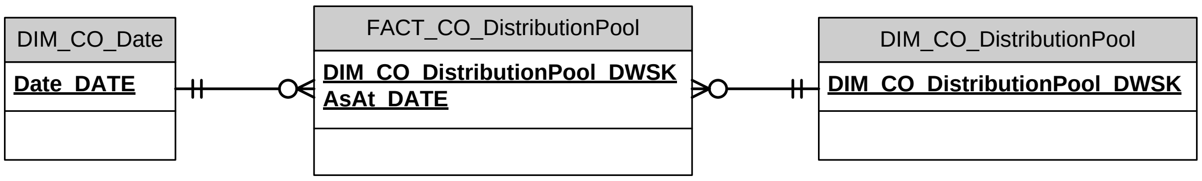
This table stores information on the Distribution Pools recorded in the Asset Management database. It works as a Snapshot Fact table i.e. new records are added on a daily basis and DWSKs are only stored once and not updated or multiplied.

5.17.2. Place in Conceptual Model

This table sources its data from the Distribution Pool data in the Asset Management database.



5.17.3. Star Schema



5.17.4. Linked Dimension Tables

- DIM_CO_DistributionPool

5.17.5. Query Hints

This table is straightforward to query and the only thing that needs to be remembered is to filter by a specific date:

```
SELECT
       ddp.DIM_CO_DistributionPool_SSK
       ,fdp.<Desired Fields>
FROM FACT_CO_DistributionPool fdp
       JOIN DIM_CO_DistributionPool ddp
              ON ddp.DIM_CO_DistributionPool_DWSK = fdp.DIM_CO_DistributionPool_DWSK
WHERE AsAt_DATE = @ReportDate
```

5.18. FACT_CO_DocumentXML

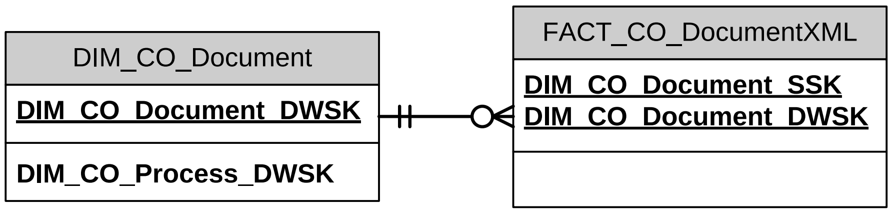
5.18.1. Purpose and How it Works

This table is unique among the tables in the warehouse in that it does not store information in a relational format, but instead stores it as an XML datatype, based on documents held in the DPR Filestore. It gives the opportunity to interrogate the raw form of the documents using XPath/XQuery.

5.18.2. Place in Conceptual Model

This table takes its data from the DPR Filestore. It links XML Documents to their appropriate Application.

5.18.3. Star Schema



5.18.4. Linked Dimension Tables

- DIM_CO_Document

5.18.5. Query Hints

Each query on this table will be unique to the document type being parsed, but most will involve linking back to the Applications linked to the document and then seeking for some specific values within the document using XPath and XQuery. For example

```
WITH cte_XMLSource AS
(
       SELECT
              DIM_CO_Application_SSK
              ,DocumentXML
       FROM
       (
              SELECT
                     dapp.DIM_CO_Application_SSK
                     ,fx.DocumentXML
                     ,ResultRank = ROW_NUMBER()
                                                        OVER
                                                        (
                                                               PARTITION BY
       dapp.DIM_CO_Application_SSK
                                                               ORDER BY dd.DateCreated DESC
                                                        )
              FROM FACT_CO_DocumentXML fx
                     JOIN DIM_CO_Document dd
                            ON dd.DIM_CO_Document_DWSK = fx.DIM_CO_Document_DWSK
                            AND dd.DocumentType = 'AffordabilityXML'
                     JOIN DIM_CO_Process dproc
                            ON dproc.DIM_CO_Process_DWSK = dd.DIM_CO_Process_DWSK
                     JOIN DIM_CO_Application dapp
                            ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
                            AND dapp.SCDStartDate <= DateCreated
                            AND dapp.SCDEndDate > DateCreated

       ) a
       WHERE ResultRank = 1
)
SELECT
       DIM_CO_Application_SSK
       ,TotalRecognisedIncome =
DocumentXML.value('(/AffordabilityData/Products/Product/Input/Applicants/Applicant/TotalRecognisedIncome) [4]','money')
FROM cte_XMLSource
```

5.19. FACT_CO_FAILEDRULESTRX

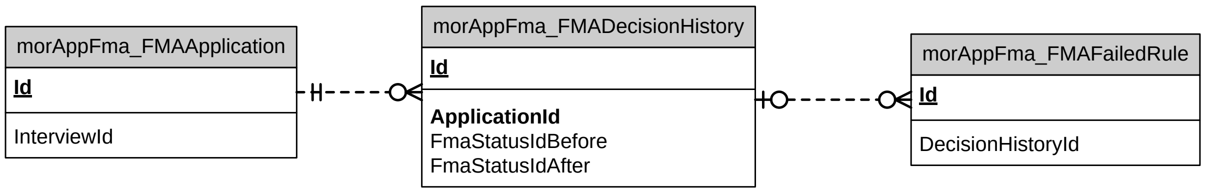
5.19.1. Purpose and How it Works

This table stores the record of the failed rules on an Application. It acts as a Transaction fact table and enters new data as and when a failure takes place. The DWSKs at the time of the transaction are the ones used.

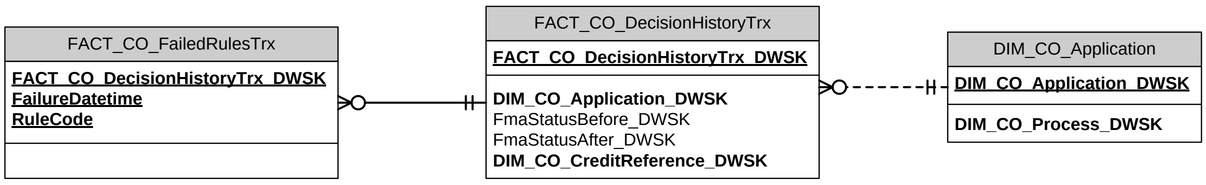
The table allows you to link to the Decision via the links to FACT_CO_DecisionHistoryTrx.

5.19.2. Place in Conceptual Model

This table sources its data from the Failed Rules information in Origination and links it back to the Application via the linked Decision History.



5.19.3. Star Schema



5.19.4. Linked Dimension Tables

None

5.19.5. Query Hints

This is an unusual table to use, the sense that it only makes sense when viewed in the context of another table. All queries will probably involve joining out to the FACT_CO_DecisionHistoryTrx table and from there onto the DIM_CO_Application table.

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,fd.<Desired Fields>
       ,fr.<Desired Fields>
FROM FACT_CO_FailedRulesTrx fr
       JOIN FACT_CO_DecisionHistoryTrx fd
              ON fr.FACT_CO_DecisionHistoryTrx_DWSK =
fd.FACT_CO_DecisionHistoryTrx_DWSK
       JOIN DIM_CO_Application dapp
              ON fd.DIM_CO_Application_DWSK = dapp.DIM_CO_Application_DWSK
```

5.20. FACT_CO_FEESAS

5.20.1. Purpose and How it Works

This table stores information on the Fees charged against the Application. Part of the primary key of the table is the column FACT_CO_FeeAS_SSK which is a copy of the primary key of the FMAFee table within BackOfficeMortgage - FMAFee.Id. This allows connections to be made out to the FMAFee table via the Synonyms.

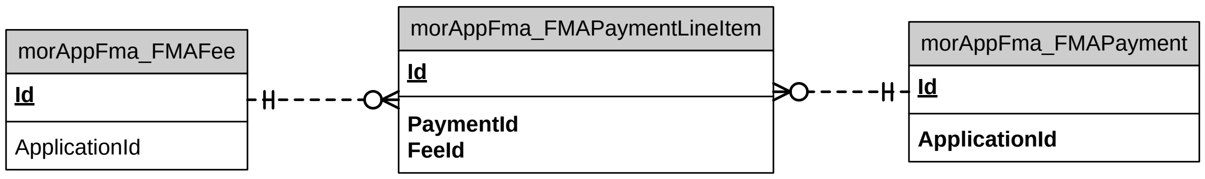
The table works as an Accumulating Snapshot Fact table i.e. new records are added as new applications are added but existing ones are updated to reflect the current picture, including the DIM_CO_Process_DWSKs. Note that this means that any previous values held in this table will be overwritten and no history kept.

Note that records can be deleted from this table as records are deleted from FMAFee. They will normally be replaced by new rows.

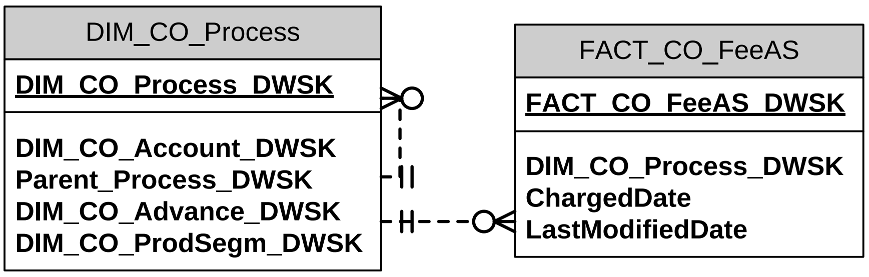
See Query Hints for the best way to obtain values from this table.

5.20.2. Place in Conceptual Model

This table retrieves most of its Fee data from the related table in Origination. It also takes information on Fee payments from the payments tables.



5.20.3. Star Schema



5.20.4. Linked Dimension Tables

- DIM_CO_Process

5.20.5. Query Hints

This table is fairly straightforward to query, the main complication coming when linking out to DIM_CO_Application via DIM_CO_Process. Because more than one DIM_CO_Application_DWSK can have the same parent DIM_CO_Process_DWSK, it is necessary to filter out the potential duplicates that could occur in the result set if this is not taken into account. Such a query would look something like this:

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,ff.<Desired Fields>
FROM FACT_CO_FeeAS ff
       JOIN DIM_CO_Process dproc
              ON dproc.DIM_CO_Process_DWSK = ff.DIM_CO_Process_DWSK
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
              AND dapp.SCDStartDate <= ff.ChargedDate
              AND dapp.SCDEndDate > ff.ChargedDate
```

5.21. FACT_CO_LeadReferral

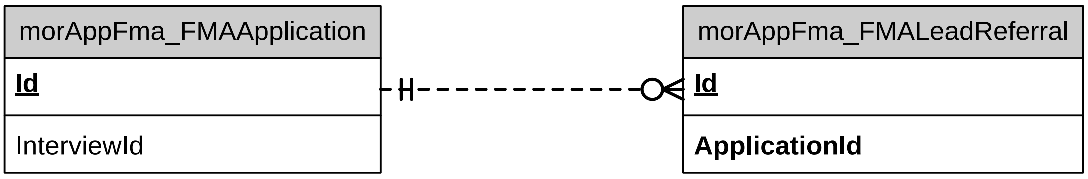
5.21.1. Purpose and How it Works

This table stores the record of any leads on Applications that have been referred by another source. It acts as a Transaction fact table and enters new data as and when a lead is referred. The DWSKs at the time of the referral are the ones used.

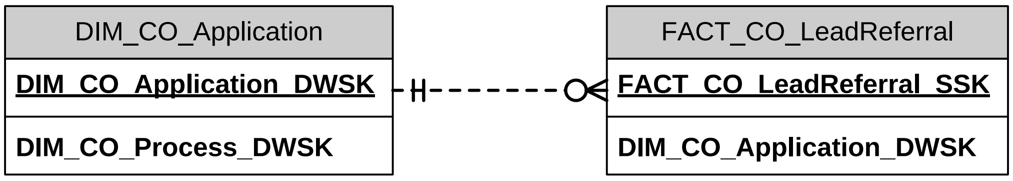
Part of the primary key of the table is the column FACT_CO_LeadReferral_SSK which is based on the primary key of the BackOfficeMortgage table FMALeadReferral – FMALeadReferral.Id. This can be used to connect back to the source system via the synonyms.

5.21.2. Place in Conceptual Model

This table sources its data from the Lead Referral table in Origination



5.21.3. Star Schema



5.21.4. Linked Dimension Tables

- DIM_CO_Application

5.21.5. Query Hints

This is a straightforward table to query and can be done simply by joining between the fact and dimension tables:

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,flr.<Desired Fields>
FROM FACT_CO_LeadReferral flr
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Application_DWSK = flr.DIM_CO_Application_DWSK
```

5.22. FACT_CO_RiskScoreCustomer

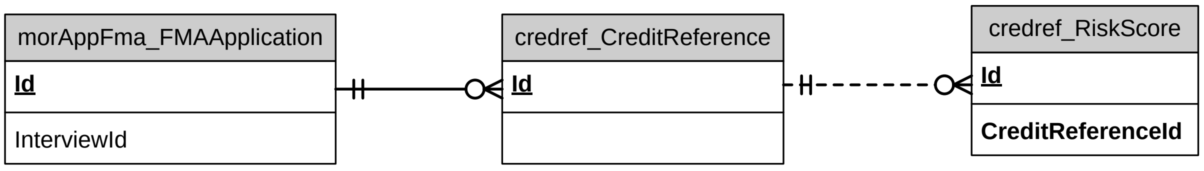
5.22.1. Purpose and How it Works

This table stores scores given to Applicants during the Credit Reference process. It works as an adapted Transactional Fact table i.e. new records are added as new scores are generated but existing ones are updated to reflect corrections. DWSKs remain as they were at the moment of capture.

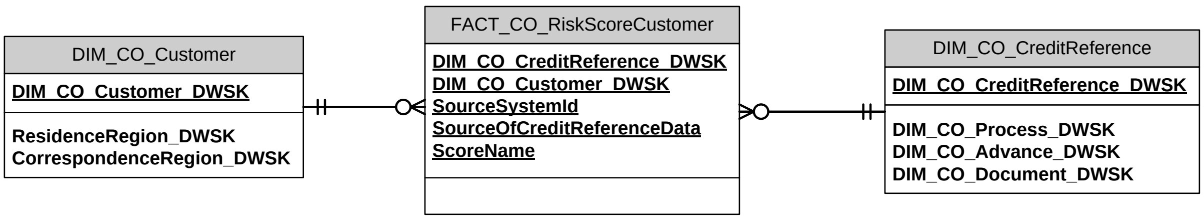
The table is modelled as an open-schema table due to the varying nature of scores that can be returned from each Credit Reference agency.

5.22.2. Place in Conceptual Model

This table sources its data from the risk score data in the Credit Reference schema of Origination



5.22.3. Star Schema



5.22.4. Linked Dimension Tables

- DIM_CO_CreditReference
- DIM_CO_Customer

5.22.5. Query Hints

When querying this table, the important thing to take care on is only returning one value per score per customer, because multiple version could exist if multiple credit searches have taken place. A query like the following will accomplish this through picking the latest version of each score:

```
SELECT
       DIM_CO_Customer_SSK
       ,ScoreName
       ,ScoreValue
FROM
(
       SELECT
              dcust.DIM_CO_Customer_SSK
              ,frs.ScoreName
              ,frs.ScoreValue
              ,ResultRank = ROW_NUMBER() OVER
                                        (
                                               PARTITION BY
                                                      DIM_CO_Customer_SSK
                                                      ,frs.ScoreName
                                               ORDER BY
                                                      dcr.CreditReferenceStartDate
DESC
```

```
                        )
    FROM FACT_CO_RiskScoreCustomer frs
        JOIN DIM_CO_Customer dcust
            ON dcust.DIM_CO_Customer_DWSK = frs.DIM_CO_Customer_DWSK
        JOIN DIM_CO_CreditReference dcr
            ON dcr.DIM_CO_CreditReference_DWSK =
frs.DIM_CO_CreditReference_DWSK
        JOIN DIM_CO_Process dproc
            ON dproc.DIM_CO_Process_DWSK = dcr.DIM_CO_Process_DWSK
        JOIN DIM_CO_Application dapp
            ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
            AND dapp.SCDStartDate <= dcr.CreditReferenceStartDate
            AND dapp.SCDStartDate > CreditReferenceStartDate
) a
WHERE ResultRank = 1
```

Optionally the results of this query could then be pivoted if the desire is to have a row per customer:

```
WITH cte_pivotsource AS
(
    SELECT
        DIM_CO_Customer_SSK
        ,ScoreName
        ,ScoreValue
    FROM
    (
        SELECT
            dcust.DIM_CO_Customer_SSK
            ,frs.ScoreName
            ,frs.ScoreValue
            ,ResultRank = ROW_NUMBER() OVER
                            (
                                PARTITION BY
    dcust.DIM_CO_Customer_SSK
                                    ,frs.ScoreName
                                ORDER BY
    dcr.CreditReferenceStartDate DESC
                            )
        FROM FACT_CO_RiskScoreCustomer frs
            JOIN DIM_CO_Customer dcust
                ON dcust.DIM_CO_Customer_DWSK = frs.DIM_CO_Customer_DWSK
            JOIN DIM_CO_CreditReference dcr
                ON dcr.DIM_CO_CreditReference_DWSK =
frs.DIM_CO_CreditReference_DWSK
            JOIN DIM_CO_Process dproc
                ON dproc.DIM_CO_Process_DWSK = dcr.DIM_CO_Process_DWSK
            JOIN DIM_CO_Application dapp
                ON dapp.DIM_CO_Process_DWSK = dproc.DIM_CO_Process_DWSK
                AND dapp.SCDStartDate <= dcr.CreditReferenceStartDate
                AND dapp.SCDStartDate > CreditReferenceStartDate
    ) a
    WHERE ResultRank = 1
)
SELECT
    DIM_CO_Customer_SSK
    ,ScoreName1
    ,ScoreName2
    ,ScoreName3
FROM cte_pivotsource src
```

```
PIVOT
(
        MAX(ScoreValue)
        FOR ScoreName IN ([ScoreName1],[ScoreName2],[ScoreName3])
) AS pvt
```

5.23. FACT_CO_StatusTrx

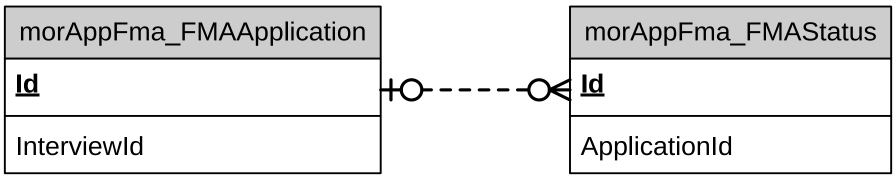
5.23.1. Purpose and How it Works

This table stores the record of the Status changes that have taken place on an Application. It acts as a Transaction fact table and enters new data as and when a Status change takes place. The DWSKs at the time of the transaction are the ones used.

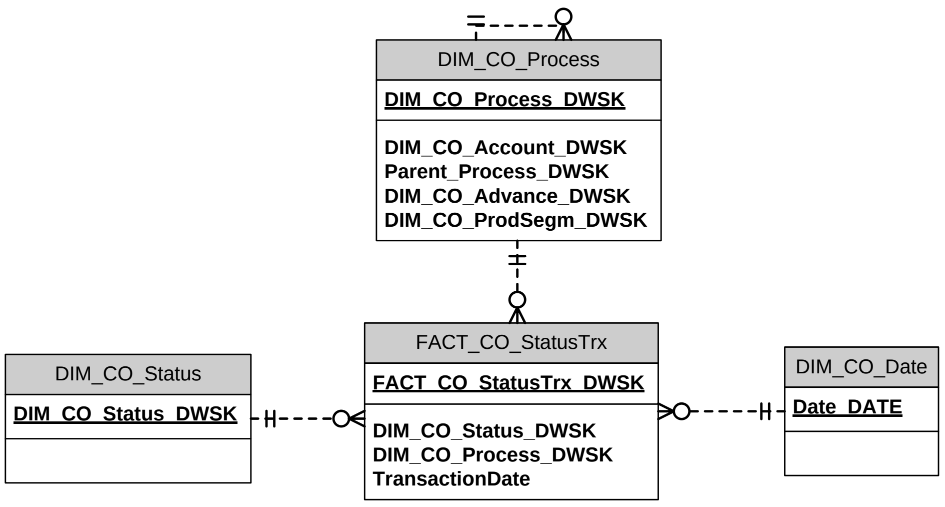
Part of the primary key of the table is the column FACT_CO_StatusTrx_SSK which is based on the primary key of the BackOfficeMortgage table FMAStatus – FMAStatus.Id. This can be used to connect back to the source system via the synonyms.

5.23.2. Place in Conceptual Model

This table sources its data from the Status history in Origination



5.23.3. Star Schema



5.23.4. Linked Dimension Tables

- DIM_CO_Date
- DIM_CO_Process
- DIM_CO_Status

5.23.5. Query Hints

The main thing to bear in mind when querying this table is the need to filter our potential multiple SCD versions of the Application which may appear through the link to DIM_CO_Process. This is best done via the SCDStartDate and SCDEndDate fields as in the query below:

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,fs.TransactionDate
       ,ds.StatusName
FROM FACT_CO_StatusTrx fs
       JOIN DIM_CO_Status ds
              ON ds.DIM_CO_Status_DWSK = fs.DIM_CO_Status_DWSK
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = fs.DIM_CO_Process_DWSK
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Process_DWSK = dp.DIM_CO_Process_DWSK
              AND dapp.SCDStartDate <= fs.TransactionDate
              AND dapp.SCDEndDate > fs.TransactionDate
```

5.24. FACT_CO_TASK

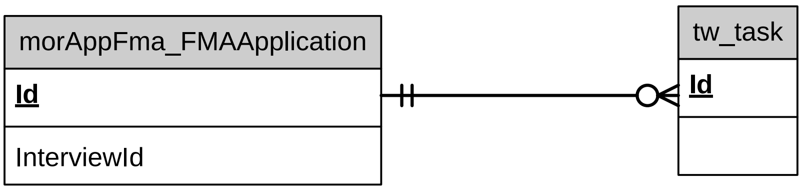
5.24.1. Purpose and How it Works

This table stores the history of Tasks for Applications. Each Task is given a Start date and an Expire date to indicate the SLA on the task. When the task is completed the Completed date is filled in. It will always be assigned a Role and can optionally be assigned a Team and/or Staff Member.

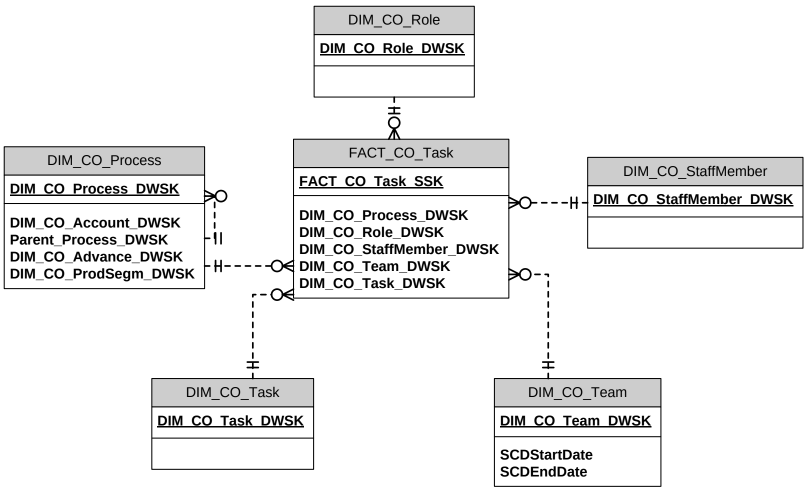
The table works as an Accumulating Snapshot table and as such only holds the latest version of the data, including the DWSKs.

5.24.2. Place in Conceptual Model

This table takes its data from the Task data held in the Thinking Works database and links this back to the Application.



5.24.3. Star Schema



5.24.4. Linked Dimension Tables

- DIM_CO_Process
- DIM_CO_Role
- DIM_CO_Task
- DIM_CO_Team

- DIM_CO_StaffMember

5.24.5. Query Hints

The main thing to bear in mind when querying this table is the need to filter our potential multiple SCD versions of the Application which may appear through the link to DIM_CO_Process. This is best done via the SCDStartDate and SCDEndDate fields as in the query below:

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,<Other Desired Fields>
FROM FACT_CO_Task ftsk
       JOIN DIM_CO_Role dr
              ON dr.DIM_CO_Role_DWSK = ftsk.DIM_CO_Role_DWSK
       JOIN DIM_CO_Task dtsk
              ON dtsk.DIM_CO_Task_DWSK = ftsk.DIM_CO_Task_DWSK
       JOIN DIM_CO_Team dt
              ON dt.DIM_CO_Team_DWSK = ftsk.DIM_CO_Team_DWSK
       JOIN DIM_CO_StaffMember ds
              ON ds.DIM_CO_StaffMember_DWSK = ftsk.DIM_CO_StaffMember_DWSK
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = ftsk.DIM_CO_Process_DWSK
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Process_DWSK = dp.DIM_CO_Process_DWSK
              AND dapp.SCDStartDate <= CONVERT(VARCHAR,ftsk.StartDateTime,112)
              AND dapp.SCDEndDate > CONVERT(VARCHAR,ftsk.StartDateTime,112)
```

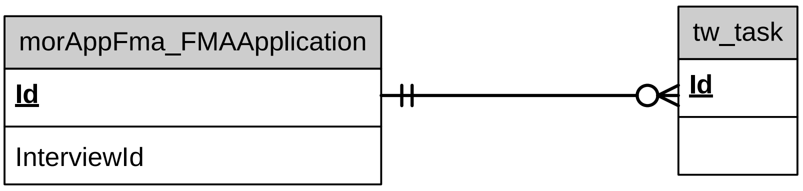
5.25. FACT_CO_TaskStatusTrx

5.25.1. Purpose and How it Works

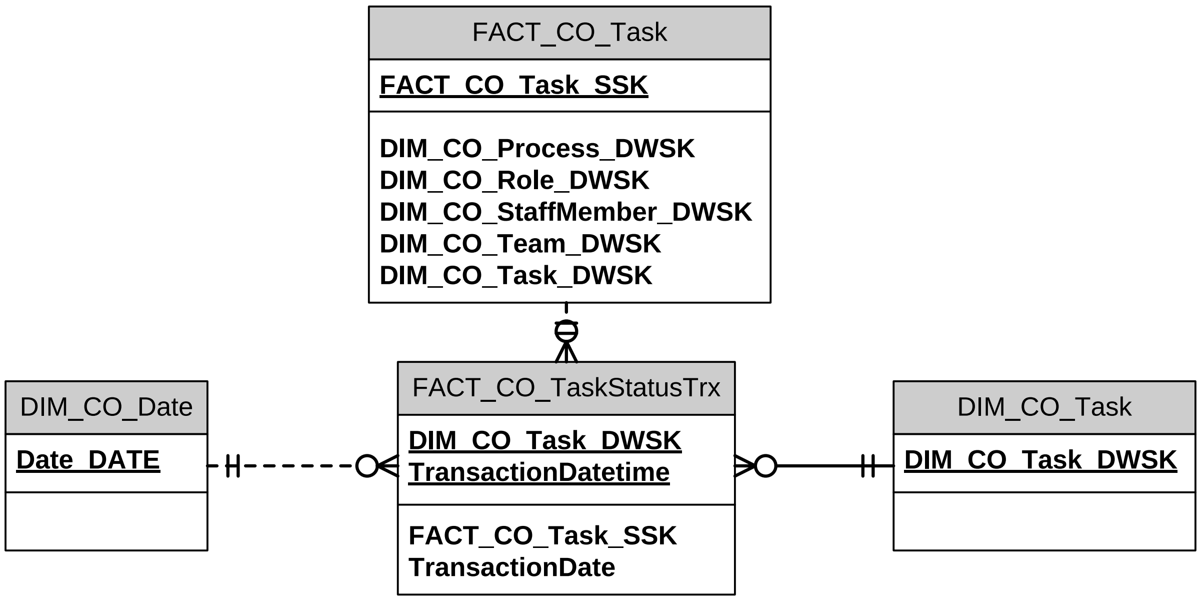
This table stores the record of the Status changes that have taken place on a Task. It acts as a Transaction fact table and enters new data as and when a Status change takes place. The DWSKs at the time of the status change are the ones used. A relationship to the main task table (FACT_CO_Task) is maintained as a way to associate status changes to the correct task instance and to link back to the wider schema.

5.25.2. Place in Conceptual Model

This table takes its data from the Task data held in the Thinking Works database and links this back to the Application.



5.25.3. Star Schema



5.25.4. Linked Dimension Tables

- DIM_CO_Date
- DIM_CO_Task

5.25.5. Query Hints

This table is essentially an extension of the FACT_CO_Task table and allows you to query a more in depth of the status of the task. As such the query to interrogate it is essentially the same as the one for FACT_CO_Task:

```
SELECT
       dapp.DIM_CO_Application_SSK
       ,<Other Desired Fields>
FROM FACT_CO_TaskStatusTrx fts
       JOIN FACT_CO_Task ftsk
              ON fts.FACT_CO_Task_SSK = ftsk.FACT_CO_Task_SSK
       JOIN DIM_CO_Role dr
              ON dr.DIM_CO_Role_DWSK = ftsk.DIM_CO_Role_DWSK
       JOIN DIM_CO_Task dtsk
              ON dtsk.DIM_CO_Task_DWSK = ftsk.DIM_CO_Task_DWSK
       JOIN DIM_CO_Team dt
              ON dt.DIM_CO_Team_DWSK = ftsk.DIM_CO_Team_DWSK
       JOIN DIM_CO_StaffMember ds
              ON ds.DIM_CO_StaffMember_DWSK = ftsk.DIM_CO_StaffMember_DWSK
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = ftsk.DIM_CO_Process_DWSK
       JOIN DIM_CO_Application dapp
              ON dapp.DIM_CO_Process_DWSK = dp.DIM_CO_Process_DWSK
              AND dapp.SCDStartDate <= CONVERT(VARCHAR,ftsk.StartDateTime,112)
              AND dapp.SCDEndDate > CONVERT(VARCHAR,ftsk.StartDateTime,112)
```

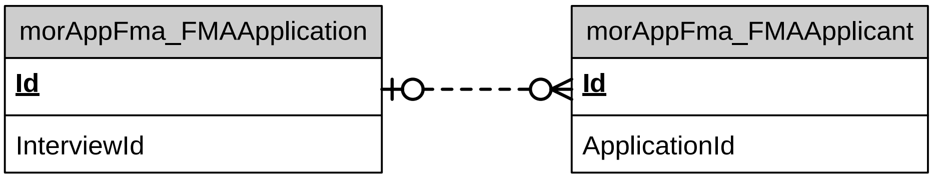
5.26. IF_COAPPLICATION_COCUSTOMER

5.26.1. Purpose and How it Works

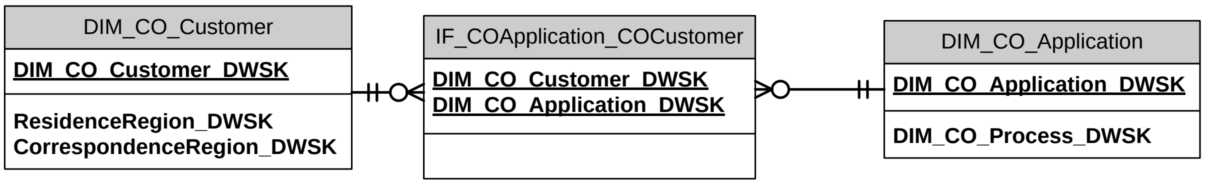
This table holds the relationship between Applicants and Applications. It acts as an adapted transactional fact table and loads entries as new relationships are created but also when new DWSKs are created. No dates are held as the relationship is assumed to last the length of the Application process.

5.26.2. Place in Conceptual Model

This table sources its data from the relationship between Applicant and Application in Origination



5.26.3. Star Schema



5.26.4. Linked Dimension Tables

- DIM_CO_Application
- DIM_CO Customer

5.26.5. Query Hints

Normally, this table will not be used in isolation, but as a conduit to linking Applicant information to Application information. The challenge comes if you are attempting to link data on later DWSKs than were used in the initial creation of the relationship. As with other Fact tables, the key is to link on SSKs rather than DWSKs in this case. For example the following query links data from FACT_CO_Application to Applicant data via the relationship table:

```
WITH cte_FactApplication AS
(
       SELECT
              da.DIM_CO_Application_SSK
              ,fa.LoanAmount_ABS
       FROM FACT_CO_Application fa
              JOIN DIM_CO_Application da
                     ON fa.DIM_CO_Application_DWSK = da.DIM_CO_Application_DWSK
),
cte_Relationship AS
(
       SELECT
              da.DIM_CO_Application_SSK
              ,dc.DIM_CO_Customer_SSK
       FROM IF_COApplication_COCustomer ifac
              JOIN DIM_CO_Application da
                     ON da.DIM_CO_Application_DWSK = ifac.DIM_CO_Application_DWSK
                     AND da.SCDStatus = 'C'
              JOIN DIM_CO_Customer dc
                     ON dc.DIM_CO_Customer_DWSK = ifac.DIM_CO_Customer_DWSK
                     AND dc.SCDStatus = 'C'
),
cte_DimCustomer AS
(
       SELECT
              dc.DIM_CO_Customer_SSK
              ,dc.FirstName
              ,dc.Surname
       FROM DIM_CO_Customer dc
       WHERE SCDStatus = 'C'
)
SELECT
       dc.FirstName
       ,dc.Surname
       ,fa.LoanAmount_ABS
FROM cte_DimCustomer dc
       JOIN cte_Relationship r
              ON r.DIM_CO_Customer_SSK = dc.DIM_CO_Customer_SSK
       JOIN cte_FactApplication fa
              ON fa.DIM_CO_Application_SSK = r.DIM_CO_Application_SSK
```

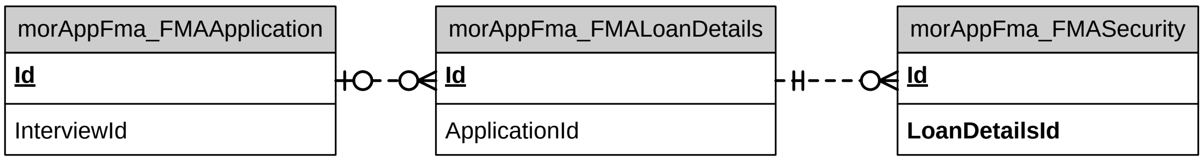
5.27. IF_COAPPLICATION_COSECURITY

5.27.1. Purpose and How it Works

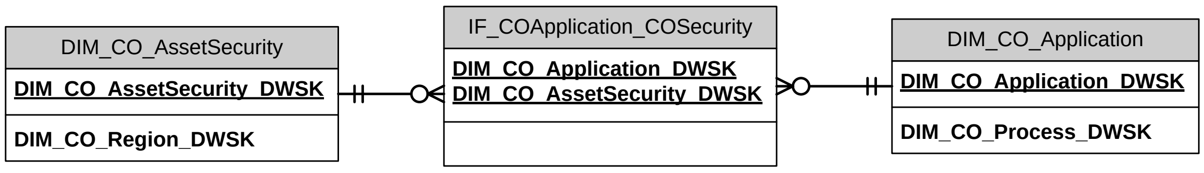
This table holds the relationship between Applicants and Applications. It acts as an adapted transactional fact table and loads entries as new relationships are created but also when new DWSKs are created. No dates are held as the relationship is assumed to last the length of the Application process. Updates do take place on the other values in the table.

5.27.2. Place in Conceptual Model

This table sources its data from the relationship between Security and Application in Origination



5.27.3. Star Schema



5.27.4. Linked Dimension Tables

- DIM_CO_AssetSecurity
- DIM_CO_Application

5.27.5. Query Hints

Normally, this table will not be used in isolation, but as a conduit to linking Security information to Application information. The challenge comes if you are attempting to link data on later DWSKs than were used in the initial creation of the relationship. As with other Fact tables, the key is to link on SSKs rather than DWSKs in this case. For example the following query links data from FACT_CO_Application to Security data via the relationship table:

```
WITH cte_FactApplication AS
(
       SELECT
              da.DIM_CO_Application_SSK
              ,fa.BasePropertyFigure_ABS
       FROM FACT_CO_Application fa
              JOIN DIM_CO_Application da
                     ON fa.DIM_CO_Application_DWSK = da.DIM_CO_Application_DWSK
),
cte_Relationship AS
(
       SELECT
              da.DIM_CO_Application_SSK
              ,das.DIM_CO_AssetSecurity_SSK
       FROM IF_COApplication_COSecurity ifas
              JOIN DIM_CO_Application da
                     ON da.DIM_CO_Application_DWSK = ifas.DIM_CO_Application_DWSK
                     AND da.SCDStatus = ‘C’
              JOIN DIM_CO_AssetSecurity das
                     ON das.DIM_CO_AssetSecurity_DWSK = ifas.DIM_CO_AssetSecurity_DWSK
                     AND das.SCDStatus = ‘C’

),
cte_DimSecurity AS
(
       SELECT
              das.DIM_CO_AssetSecurity_SSK
              ,das.SecurityType
              ,das.SecuritySubType
       FROM DIM_CO_AssetSecurity das
       WHERE SCDStatus = 'C'
)
SELECT
       ds.SecurityType
       ,ds.SecuritySubType
       ,fa.BasePropertyFigure_ABS
FROM cte_DimSecurity ds
       JOIN cte_Relationship r
              ON r.DIM_CO_AssetSecurity_SSK = ds.DIM_CO_AssetSecurity_SSK
       JOIN cte_FactApplication fa
              ON fa.DIM_CO_Application_SSK = r.DIM_CO_Application_SSK
```

5.28. IF_COProcess_COIntermediary

5.28.1. Purpose and How it Works

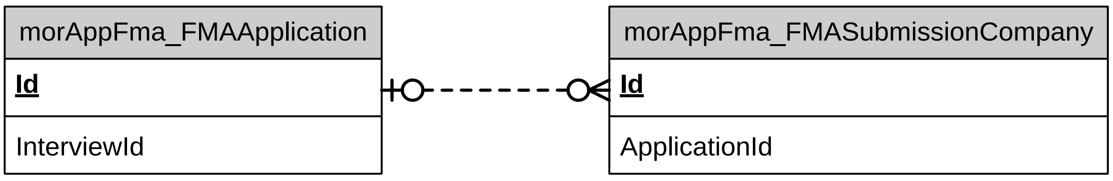
This table is used to track the relationship between Applications (via Process) and Intermediaries. It acts like an adapted Accumulating Snapshot Fact table in that items are updated (the dates associated with the relationship) but new DWSK version are included in the table as they arise without updating existing entries.

To handle this there are two sets of dates on the table. Start Date and End Date represent the period of the relationship between two SSKs i.e. the start and end of the relationship between the two entities. The SCDStartDate and SCDEndDate represent the length of time between two DWSKs i.e. the relationship between two different versions of the entities.

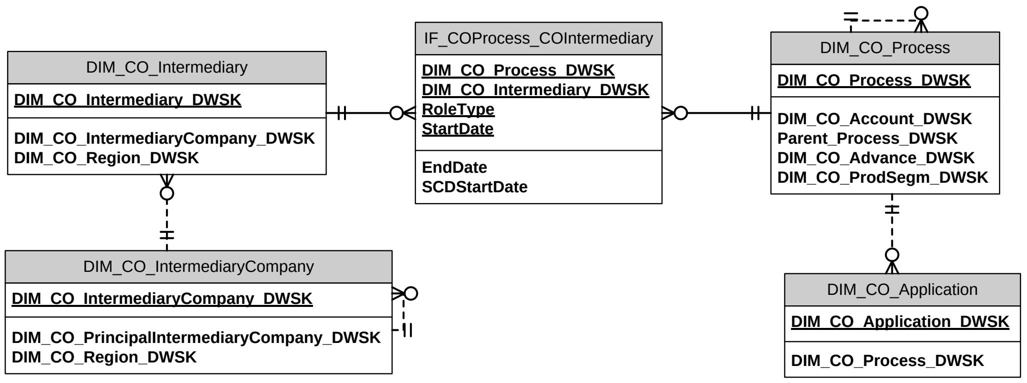
The other main field is the Role Type, which represents the role the intermediary plays on the Application i.e. Broker, Network, Mortgage Club or Packager. This means that more than one Intermediary type can be associated with an Application concurrently – this represents the submission route.

5.28.2. Place in Conceptual Model

This table sources its data from the relationship between Submission Companies (Intermediaries) and Application in Origination.



5.28.3. Star Schema



5.28.4. Linked Dimension Tables

- DIM_CO_Process
- DIM_CO_Intermediary

5.28.5. Query Hints

Usually the reason to use this table will be to extract submission route information for the Applications. To do this the multiple potential relationships need to be accessed whilst also taking care to take into account the multiple DWSKs in place.

A query like the one below can be used for this purpose, which uses the current versions of the dimension records and links them to a single instance of each relationship, anchored on the Start Date of the relationship. A pivot expression is then used to return a single row per application:

```sql
WITH cte_Application AS
(
        SELECT
                DIM_CO_Application_SSK
                ,ApplicationRef
        FROM DIM_CO_Application
        WHERE SCDStatus = 'C'
),
cte_Relationship AS
(
        SELECT
                dapp.DIM_CO_Application_SSK
                ,di.DIM_CO_Intermediary_SSK
                ,ifpi.RoleType
        FROM IF_COProcess_COIntermediary ifpi
                JOIN DIM_CO_Process dp
                        ON dp.DIM_CO_Process_DWSK = ifpi.DIM_CO_Process_DWSK
                        AND dp.SCDStartDate <= ifpi.StartDate
                        AND dp.SCDEndDate > ifpi.StartDate
                JOIN DIM_CO_Application dapp
                        ON dapp.DIM_CO_Process_DWSK = dp.DIM_CO_Process_DWSK
                        AND dapp.SCDStartDate <= ifpi.StartDate
                        AND dapp.SCDEndDate > ifpi.StartDate
                JOIN DIM_CO_Intermediary di
                        ON di.DIM_CO_Intermediary_DWSK = ifpi.DIM_CO_Intermediary_DWSK
                        AND di.SCDStartDate <= ifpi.StartDate
                        AND di.SCDEndDate > ifpi.StartDate
),
cte_Intermediaries AS
(
        SELECT
                di.DIM_CO_Intermediary_SSK
                ,dic.IntermediaryCompanyName
        FROM DIM_CO_Intermediary di
                JOIN DIM_CO_IntermediaryCompany dic
                        ON di.DIM_CO_IntermediaryCompany_DWSK =
                        dic.DIM_CO_IntermediaryCompany_DWSK
                        AND dic.SCDStatus = 'C'
        WHERE di.SCDStatus = 'C'
),
cte_Source AS
(
        SELECT
                dapp.ApplicationRef
                ,r.RoleType
                ,di.IntermediaryCompanyName
        FROM cte_Application dapp
        JOIN cte_Relationship r
                ON dapp.DIM_CO_Application_SSK = r.DIM_CO_Application_SSK
        JOIN cte_Intermediaries di
                ON di.DIM_CO_Intermediary_SSK = r.DIM_CO_Intermediary_SSK
)
SELECT
        ApplicationRef
        ,[Broker]
        ,[Network]
        ,[MortgageClub]
        ,[Packager]
FROM cte_Source
PIVOT
(
        MAX(IntermediaryCompanyName)
        FOR RoleType IN ([Broker],[Network],[MortgageClub],[Packager])
) as pvt
```

5.29. IF_COPROCESS_COTHIRDPARTY

5.29.1. Purpose and How it Works

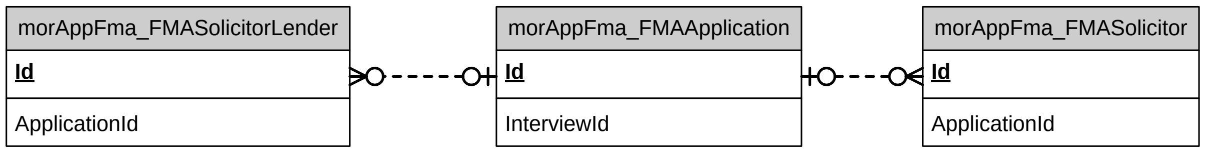
This table is used to track the relationship between Applications (via Process) and Third Parties. It acts like an adapted Accumulating Snapshot Fact table in that items are updated (the dates associated with the relationship) but new DWSK version are included in the table as they arise without updating existing entries.

To handle this there are two sets of dates on the table. Start Date and End Date represent the period of the relationship between two SSKs i.e. the start and end of the relationship between the two entities. The SCDStartDate and SCDEndDate represent the length of time between two DWSKs i.e. the relationship between two different versions of the entities.

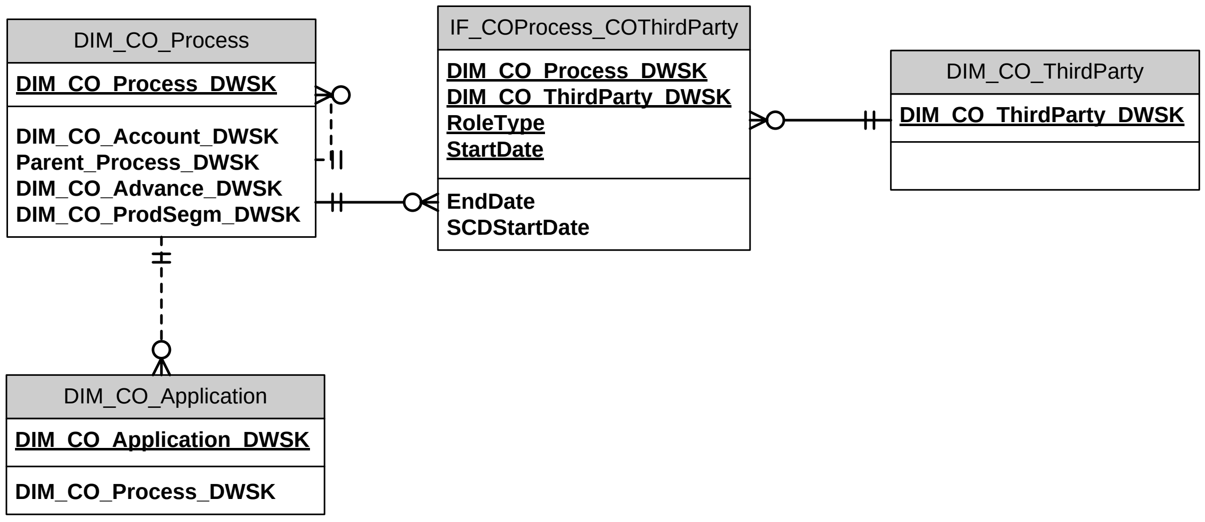
The other main field is the Role Type, which represents the role the Third Party plays on the Application i.e. Solicitor(Borrower), Solicitor(Lender). This means that more than one Party type can be associated with an Application concurrently.

5.29.2. Place in Conceptual Model

This table sources its data from the relationship between Solicitors (Third Parties) and Applications in Origination



5.29.3. Star Schema



5.29.4. Linked Dimension Tables

- DIM_CO_Process
- DIM_CO_ThirdParty

5.29.5. Query Hints

Usually the reason to use this table will be to extract Solicitor information for the Applications. To do this the two potential relationships need to be accessed whilst also taking care to take into account the multiple DWSKs in place.

A query like the one below can be used for this purpose, which uses the current versions of the dimension records and links them to a single instance of each relationship, anchored on the Start Date of the relationship. A pivot expression is then used to return a single row per application:

```
WITH cte_Application AS
(
        SELECT
                DIM_CO_Application_SSK
                ,ApplicationRef
        FROM DIM_CO_Application
        WHERE SCDStatus = 'C'
),
cte_Relationship AS
(
        SELECT
                dapp.DIM_CO_Application_SSK
                ,dt.DIM_CO_ThirdParty_SSK
                ,ifpt.RoleType
        FROM IF_COProcess_COThirdParty ifpt
                JOIN DIM_CO_Process dp
                        ON dp.DIM_CO_Process_DWSK = ifpt.DIM_CO_Process_DWSK
                        AND dp.SCDStartDate <= ifpt.StartDate
                        AND dp.SCDEndDate > ifpt.StartDate
                JOIN DIM_CO_Application dapp
                        ON dapp.DIM_CO_Process_DWSK = dp.DIM_CO_Process_DWSK
                        AND dapp.SCDStartDate <= ifpt.StartDate
                        AND dapp.SCDEndDate > ifpt.StartDate
                JOIN DIM_CO_ThirdParty dt
                        ON dt.DIM_CO_ThirdParty_DWSK = ifpt.DIM_CO_ThirdParty_DWSK
                        AND dt.SCDStartDate <= ifpt.StartDate
                        AND dt.SCDEndDate > ifpt.StartDate
),
cte_ThirdParty AS
(
        SELECT
                dt.DIM_CO_ThirdParty_SSK
                ,dt.CompanyName
        FROM DIM_CO_ThirdParty dt
        WHERE dt.SCDStatus = 'C'
),
cte_Source AS
(
        SELECT
                dapp.ApplicationRef
                ,RoleType = CASE
                                WHEN r.RoleType = 'Solicitor for Customer(s)' THEN SolicitorBorrower'
                                WHEN r.RoleType = 'Solicitor for Lender' THEN 'SolicitorLender'
                            END
                ,dt.CompanyName
        FROM cte_Application dapp
        JOIN cte_Relationship r
                ON dapp.DIM_CO_Application_SSK = r.DIM_CO_Application_SSK
        JOIN cte_ThirdParty dt
                ON dt.DIM_CO_ThirdParty_SSK = r.DIM_CO_ThirdParty_SSK
)
SELECT
        ApplicationRef
        ,[SolicitorBorrower]
        ,[SolicitorLender]
FROM cte_Source
PIVOT
(
        MAX(CompanyName)
        FOR RoleType IN ([SolicitorBorrower],[SolicitorLender])
) as pvt
```

5.30. DIM_CO_Application

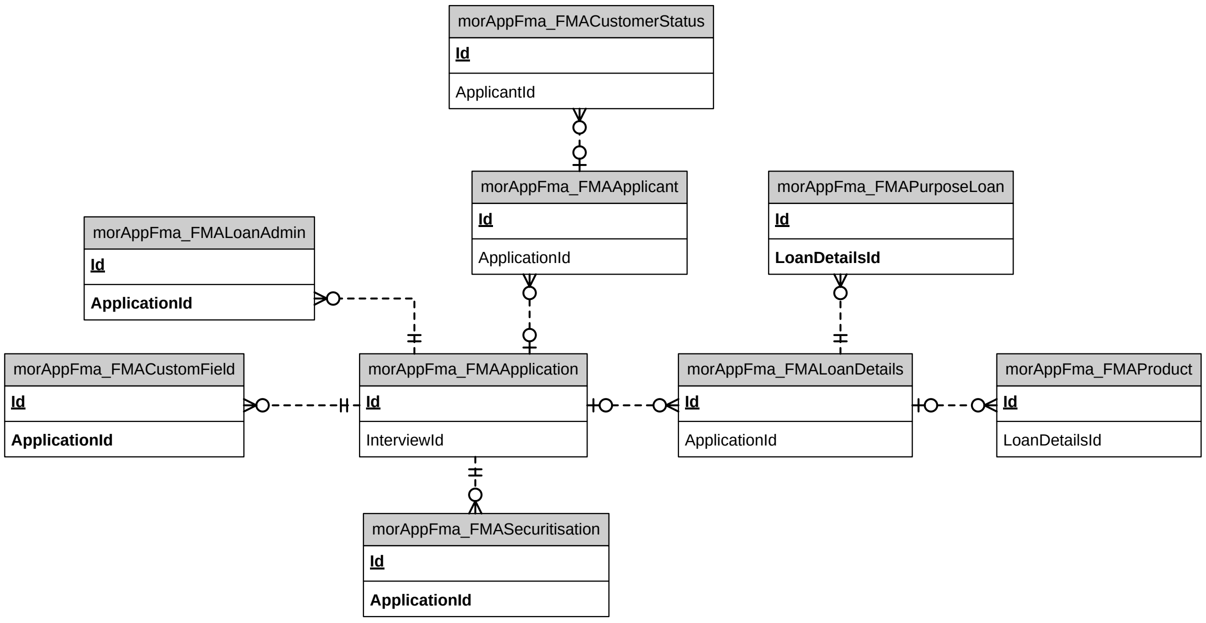
5.30.1. Purpose and How it Works

This table stores the contextual information regarding the Applications. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.30.2. Place in Conceptual Model

This table takes the majority of its data from the Application data in Origination. It also retrieves information from several other parts of the schema to enhance this information. Specifically it gets:

- Funder Pool information from the FMASecuritisation table
- Existing Customer information from FMAApplicant
- Offer Expiry information from FMALoanAdmin
- Branch and Advisor information from FMACustomField
- Currency, Benefit Period and ERC Period information from FMAProduct
- Debt Consolidation, Home Improvement and Capital Raising information from FMAPurposeLoan
- High Net Worth and Mortgage Prisoner information from FMACustomerStatus



5.30.3. Associated Hierarchies

- DIM_CO_Process
    - DIM_CO_Application

5.30.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked Fact table. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per Application. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Application
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating Applications that were made after the specified date:

```
SELECT
        <Desired Fields>
FROM DIM_CO_Application
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

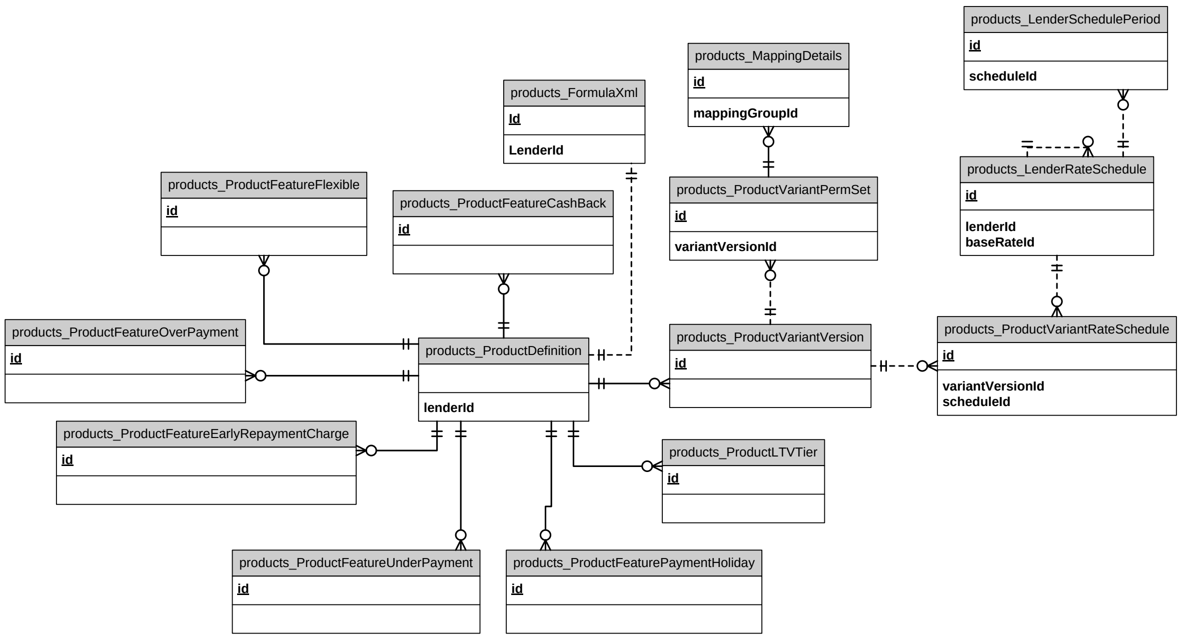
5.31. DIM_CO_APPLICATIONPRODUCT

5.31.1. Purpose and How it Works

This table stores the contextual information regarding the Application Products. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.31.2. Place in Conceptual Model

This table sources data from a wide selection of tables in the Origination Products database.



products_LenderSchedulePeriod
id
scheduleId
products_MappingDetails
id
mappingGroupId
products_FormulaXml
Id
LenderId
products_ProductFeatureFlexible
id
products_ProductFeatureCashBack
id
products_ProductVariantPermSet
id
variantVersionId
products_LenderRateSchedule
id
lenderId
baseRateId
products_ProductFeatureOverPayment
id
products_ProductDefinition
lenderId
products_ProductVariantVersion
id
products_ProductVariantRateSchedule
id
variantVersionId
scheduleId
products_ProductFeatureEarlyRepaymentCharge
id
products_ProductLTVTier
id
products_ProductFeatureUnderPayment
id
products_ProductFeaturePaymentHoliday
id

5.31.3. Associated Hierarchies

None

5.31.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked Fact table. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per Application Product. This can either be done by viewing the current record:

```
SELECT
        <Desired Fields>
FROM DIM_CO_ApplicationProduct
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ApplicationProduct
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

5.32. DIM_CO_ApplicationRepaymentVehicle

5.32.1. Purpose and How it Works

This table stores the contextual information regarding the Repayment Vehicles. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.32.2. Place in Conceptual Model

This table sources its data from the Repayment Vehicle information in Origination.

morAppFma_FMARepaymentVehicle
<u>Id</u>
LoanDetailsId

5.32.3. Associated Hierarchies

None

5.32.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked Fact table. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ApplicationRepaymentVehicle
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ ApplicationRepaymentVehicle
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

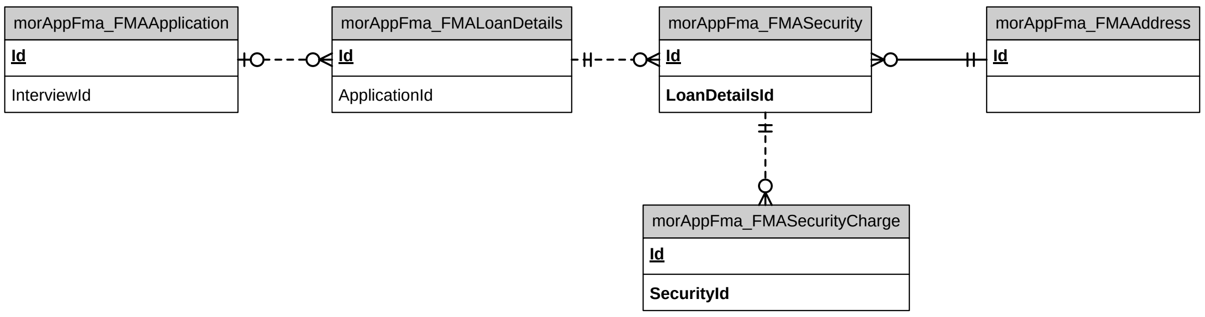
5.33. DIM_CO_AssetSecurity

5.33.1. Purpose and How it Works

This table stores the contextual information regarding the Securities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.33.2. Place in Conceptual Model

This table sources most of its data from the Security information within Origination (including Security Charges). It also links back to the Application to ascertain whether this is a Shared Ownership purchase or not.



5.33.3. Associated Hierarchies

- DIM_CO_AssetSecurity
    - DIM_CO_Valuation
        - DIM_CO_ValComparable

5.33.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

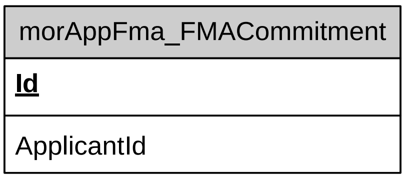
5.34. DIM_CO_COMMITMENTTYPE

5.34.1. Purpose and How it Works

This table stores the contextual information regarding the Commitment Types. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.34.2. Place in Conceptual Model

This table sources its data exclusively from the Commitments information in Origination.



5.34.3. Associated Hierarchies

None

5.34.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_CommitmentType
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_CommitmentType
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

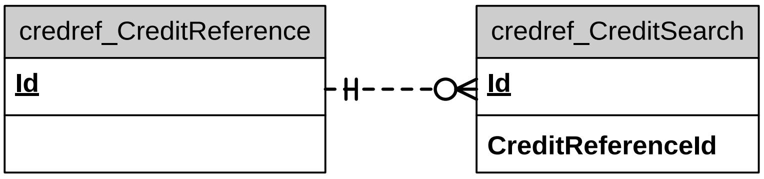
5.35. DIM_CO_CreditReference

5.35.1. Purpose and How it Works

This table stores the contextual information regarding the Credit Reference. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.35.2. Place in Conceptual Model

This table sources its information from the Credit Reference data in Origination



5.35.3. Associated Hierarchies

- DIM_CO_Process
  - DIM_CO_CreditReference

- DIM_CO_Document
  - DIM_CO_CreditReference

5.35.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_CreditReference
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_CreditReference
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

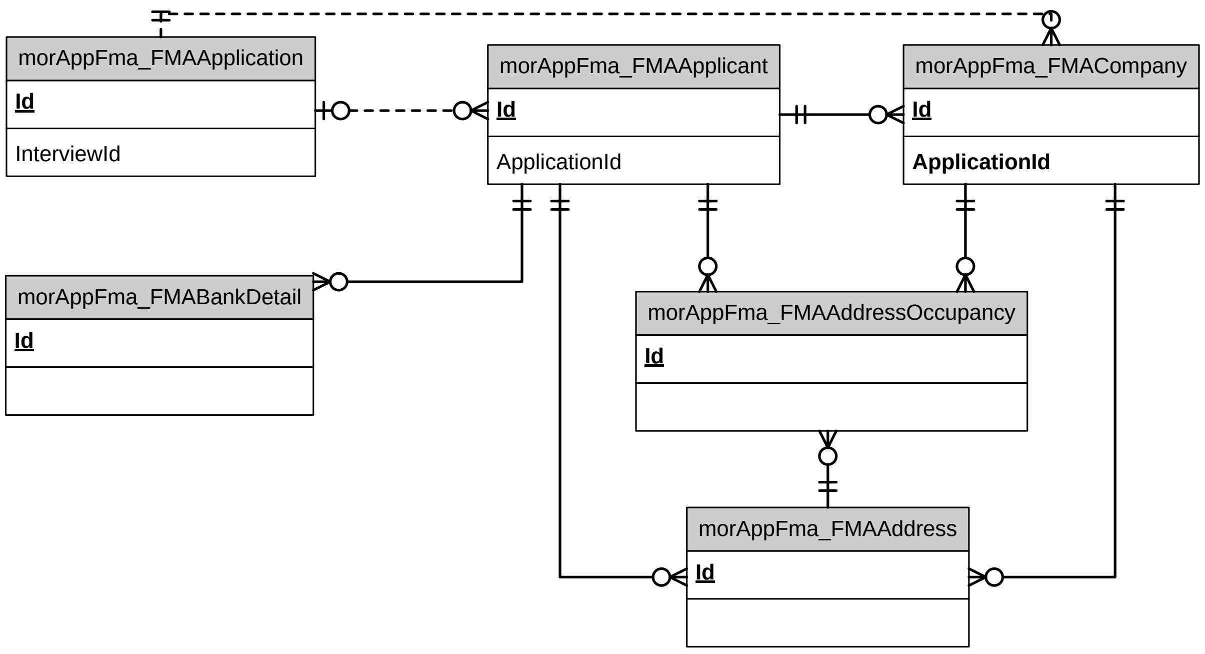
5.36. DIM_CO_CUSTOMER

5.36.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.36.2. Place in Conceptual Model

This table sources its data from the Applicant level information in Origination and its associated Address and Banking Details



5.36.3. Associated Hierarchies

None

5.36.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Customer
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
```

```
       <Desired Fields>
FROM DIM_CO_Customer
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

5.37. DIM_CO_Date

5.37.1. Purpose and How it Works

This is a static helper table that stores date related information

5.37.2. Place in Conceptual Model

This is a fixed list of dates and is not sourced directly from anywhere.

5.37.3. Associated Hierarchies

None

5.37.4. Query Hints

This table should be queried in combination with linked tables

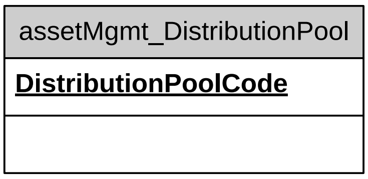
5.38. DIM_CO_DistributionPool

5.38.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.38.2. Place in Conceptual Model

This table takes its data exclusively from the Distribution Pool information in the Asset Management database



5.38.3. Associated Hierarchies

- None

5.38.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_DistributionPool
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
```

```
FROM DIM_CO_DistributionPool
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

5.39. DIM_CO_Document

5.39.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.39.2. Place in Conceptual Model

This table takes its data from the Documents held in the DPR Filestore database

5.39.3. Associated Hierarchies

None

5.39.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Document
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Document
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

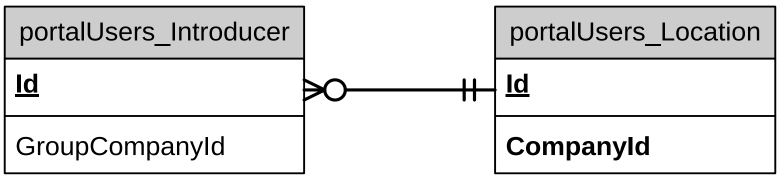
5.40. DIM_CO_Intermediary

5.40.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.40.2. Place in Conceptual Model

This table sources its data from the Introducer (Intermediary) information in the Portal Users database.



It also stores an "Anonymous Individual" for every DIM_CO_IntermediaryCompany entry in the data warehouse. This individual is used whenever an introducer is not recorded, which is the case for Networks, Mortgage Clubs and Packagers.

5.40.3. Associated Hierarchies

- DIM_CO_IntermediaryCompany
    - DIM_CO_Intermediary

5.40.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Intermediary
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Intermediary
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

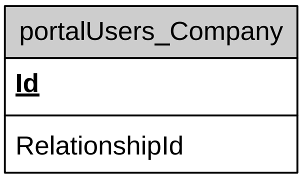
5.41. DIM_CO_IntermediaryCompany

5.41.1. Purpose and How it Works

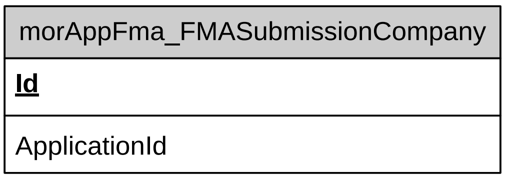
This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.41.2. Place in Conceptual Model

This table sources its data from two sources. First it takes all of the known information from the company records in the Portal Users database.



It also takes a feed of information from the BackOfficeMortgage table FMASubmissionCompany…



…which is used in case a CompanyId is present in the BackOfficeMortgage database that is not recorded in Portal Users.

5.41.3. Associated Hierarchies

- DIM_CO_IntermediaryCompany
    - DIM_CO_Intermediary

5.41.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

5.42. DIM_CO_IntermediaryCompanyLocation

5.42.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.42.2. Place in Conceptual Model

This table is sourced exclusively from the location information held in Portal Users

portalUsers_Location
<u>Id</u>
CompanyId

5.42.3. Associated Hierarchies

- DIM_CO_IntermediaryCompany
    - DIM_CO_IntermediaryCompanyLocation

- DIM_CO_Intermediary
    - DIM_CO_IntermediaryCompanyLocation

5.42.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompanyLocation
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompanyLocation
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

5.43. DIM_CO_Process

5.43.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.43.2. Place in Conceptual Model

This data is sourced from the Application information in Origination.

morAppFma_FMAApplication
Id
InterviewId

5.43.3. Associated Hierarchies

There is a hierarchy within DIM_CO_Process itself, such that Processes can be parents to other Processes

In the context of Origination then the hierarchy at play is

- DIM_CO_Process
  - DIM_CO_Application

5.43.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Process
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Process
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

5.44. DIM_CO_Region

5.44.1. Purpose and How it Works

This is a static table linking postcodes to UK regions.

5.44.2. Place in Conceptual Model

This is a set of information sourced from an external Excel spreadsheet.

5.44.3. Associated Hierarchies

None

5.44.4. Query Hints

This table should be queried in combination with linked tables

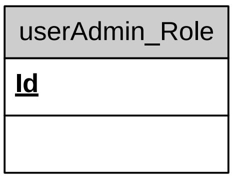
5.45. DIM_CO_ROLE

5.45.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.45.2. Place in Conceptual Model

This table sources its data exclusively from the Role information in the User Admin database



5.45.3. Associated Hierarchies

None

5.45.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Role
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Role
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

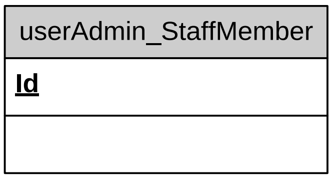
5.46. DIM_CO_STAFFMEMBER

5.46.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.46.2. Place in Conceptual Model

This table sources its data exclusively from the Staff Member information in the User Admin database



5.46.3. Associated Hierarchies

None

5.46.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_StaffMember
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_StaffMember
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

5.47. DIM_CO_Status

5.47.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.47.2. Place in Conceptual Model

This table combines information from Lookups information stored in the Origination Reference Data (of the types FMAStatusType and FMAActivityType) and supplements it with data held in an Excel spreadsheet. The values held will correspond to entries seen in the Origination Status data.

5.47.3. Associated Hierarchies

None

5.47.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Status
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Status
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

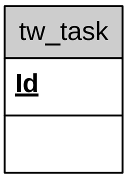
5.48. DIM_CO_TASK

5.48.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.48.2. Place in Conceptual Model

This table sources its data from the Task information in the ThinkingWorks database.



5.48.3. Associated Hierarchies

None

5.48.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Task
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Task
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

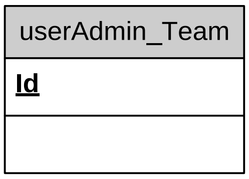
5.49. DIM_CO_TEAM

5.49.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.49.2. Place in Conceptual Model

This table sources its data exclusively from the Team information held in User Admin.



5.49.3. Associated Hierarchies

None

5.49.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Team
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Team
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

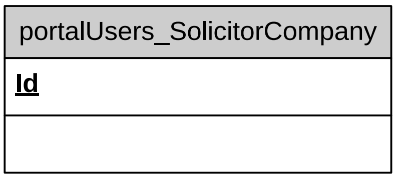
5.50. DIM_CO_THIRDPARTY

5.50.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.50.2. Place in Conceptual Model

This table sources its information from two locations. The primary one is the Solicitor Company information held in Portal Users.



The second is from the tables FMASolicitor and FMASolicitorLender in BackOfficeMortgage, where the Application has been linked to Solicitors that have not been set up in Portal Users.

5.50.3. Associated Hierarchies

None

5.50.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

5.51. DIM_CO_TIME

5.51.1. Purpose and How it Works

This is a static table used to provide intraday information

5.51.2. Place in Conceptual Model

This is a fixed table populated at the time of the warehouse’s creation

5.51.3. Associated Hierarchies

- DIM_CO_Date
    - DIM_CO_Time

5.51.4. Query Hints

This table should not be queried in isolation, but in combination with linked tables.

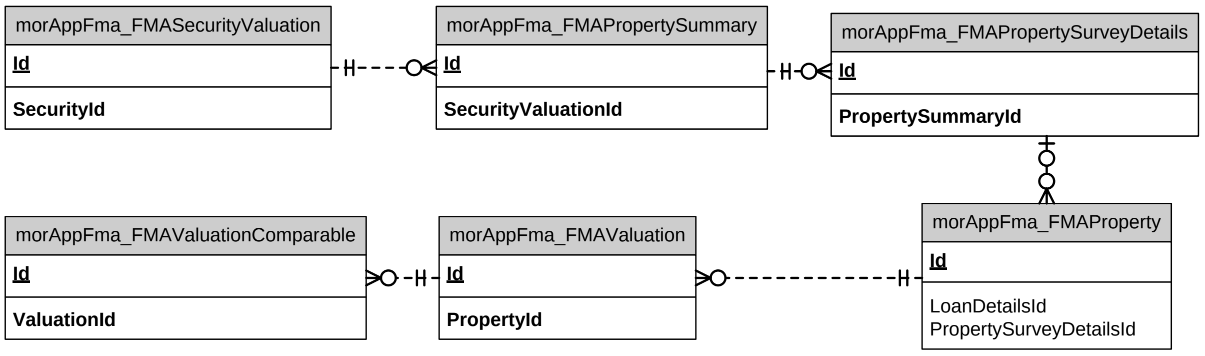
5.52. DIM_CO_VALCOMPARABLE

5.52.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.52.2. Place in Conceptual Model

This table sources its data from the Valuation Comparable information held in Origination.



5.52.3. Associated Hierarchies

- DIM_CO_AssetSecurity
  - DIM_CO_Valuation
    - DIM_CO_ValComparable

5.52.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ValComparable
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ValComparable
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

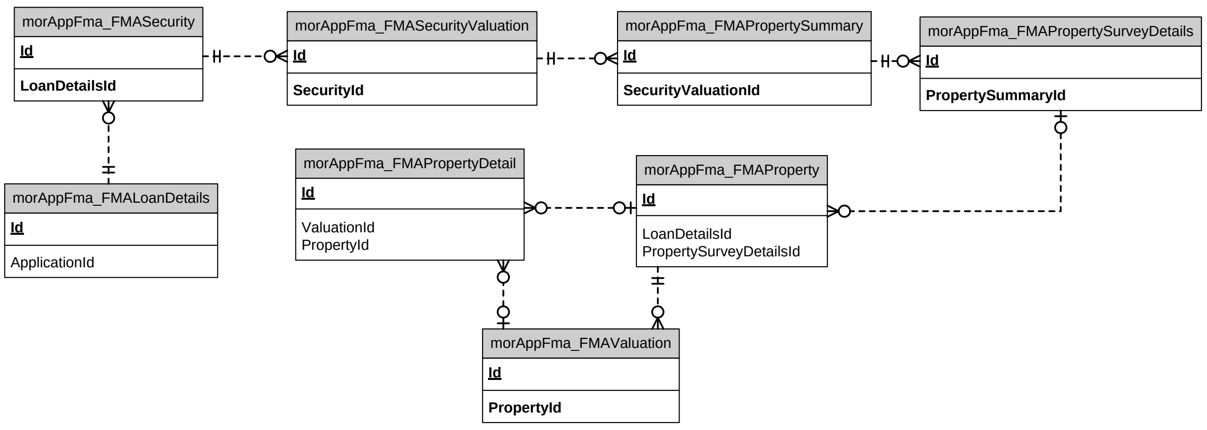
5.53. DIM_CO_VALUATION

5.53.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

5.53.2. Place in Conceptual Model

This table sources its data from the Valuation model held in Origination



5.53.3. Associated Hierarchies

- DIM_CO_AssetSecurity
    - DIM_CO_Valuation
        - DIM_CO_ValComparable

5.53.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6. SERVICING TABLES

6.1. FACT_CO_Account

6.1.1. Purpose and How it Works

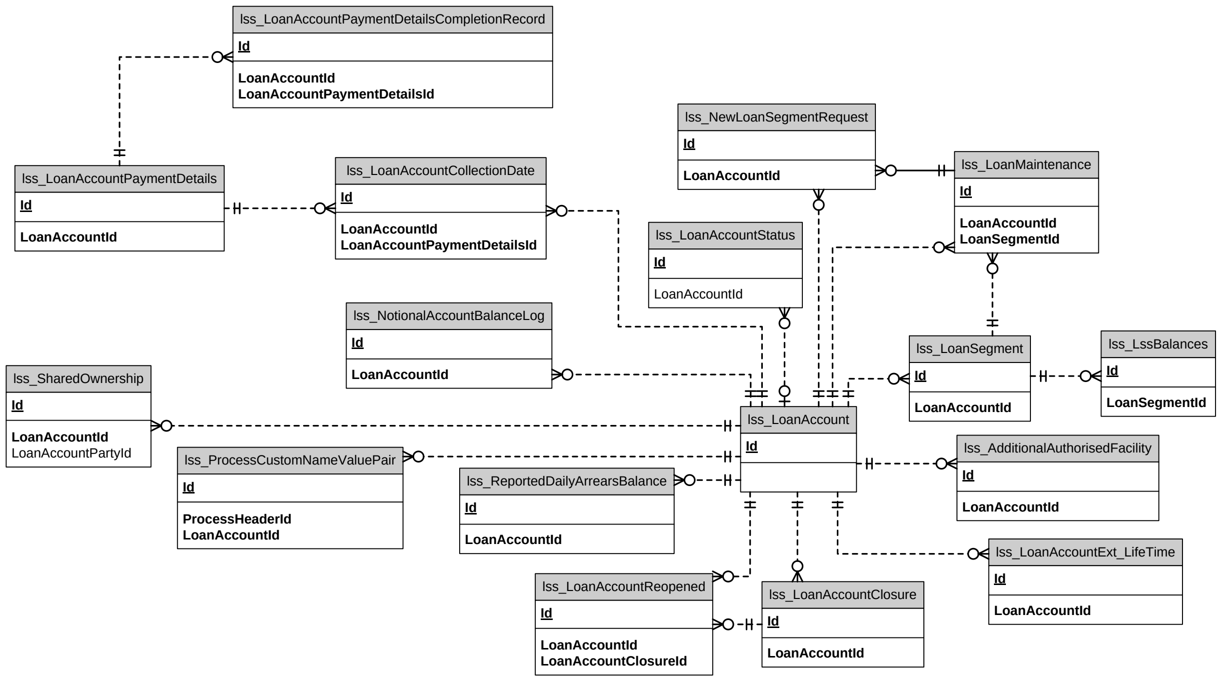
This table stores information at the Account level. It works as a Snapshot Fact table i.e. new records are added on a daily basis and DWSKs are only stored once and not updated or multiplied.

When accounts close, a configurable grace period (typically set to 100 days) is allowed before fact records stop being loaded.

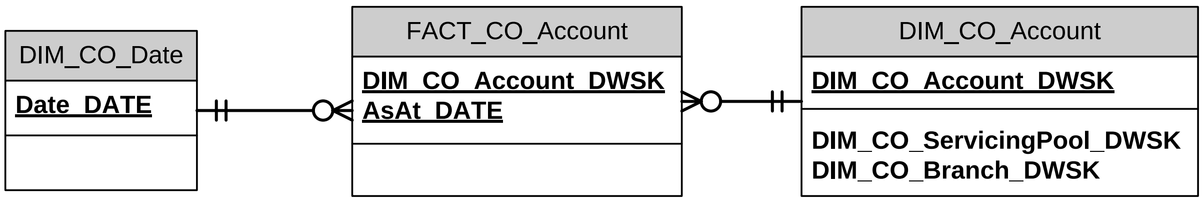
6.1.2. Place in Conceptual Model

This table sources its data from various areas of the Servicing schema linked to the Account information. As well as the main Account table it uses the following:

- Iss_LssBalances and Iss_LoanSegment to source unallocated funds information
- Iss_ReportedDailyArrearsBalance to source Account level arrears amounts
- Iss_LoanMaintenance and Iss_NewLoanSegmentRequest to bring back capitalised arrears data
- Iss_NotionalAccountBalanceLog to work out the number of months in arrears
- Iss_LoanAccountExt_LifeTime to source guaranteed cash reserve amounts
- Iss_AdditionalAuthorisedFacility to work out unused facility amounts
- Iss_LoanAccountStatus to work out the latest status on the Account
- Iss_LoanAccountCollectionDate and Iss_LoanAccountPaymentDetails to work out how many rollovers have taken place on a loan Account
- Iss_ProcessCustomNameValuePair to bring back data on the amount raised through Repossession Sales
- Iss_SharedOwnership to work out the shared ownership percentage currently with the customer



6.1.3. Star Schema



6.1.4. Linked Dimension Tables

- DIM_CO_Date
- DIM_CO_Account

6.1.5. Query Hints

The main thing to always bear in mind when querying a Snapshot Fact table is to always filter on a specific date in order to retrieve only one result per entity. To join the results of this Fact table to others, base the joins on the SSK as in the query below. No filtering of the dimension is required as the table will only hold one result per entity as enforced by the primary key.

```
SELECT
       dacc.DIM_CO_Account_SSK
       ,<Desired Fields>
FROM FACT_CO_Account facc
       JOIN DIM_CO_Account dacc
              ON dacc.DIM_CO_Account_DWSK = facc.DIM_CO_Account_DWSK
WHERE AsAt_DATE = @ReportDate
```

6.2. FACT_CO_AccountEventsTrx

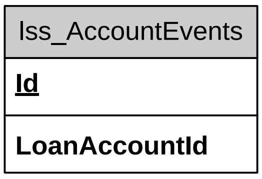
6.2.1. Purpose and How it Works

This table stores information on Episode and Actions. Part of the primary key of the table is the column FACT_CO_AccountEvents_SSK which is a copy of the primary key of the Iss_AccountEvents table within Servicing - Iss_AccountEvents.Id. This allows connections to be made out to the Iss_AccountEvents table via the Synonyms.

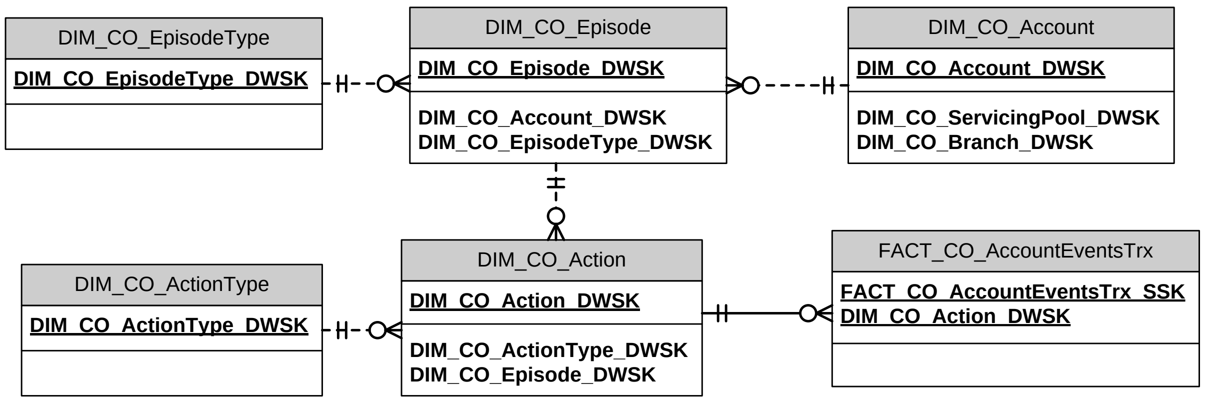
The table works as an adapted Transactional Fact table i.e. new records are added as new Actions are added and DWSKs are only stored once. Fields such as dates are updated as they change.

6.2.2. Place in Conceptual Model

This table sources its data exclusively from Account Events data in Servicing.



6.2.3. Star Schema



6.2.4. Linked Dimension Tables

- DIM_CO_Action

6.2.5. Query Hints

When retrieving data from this table, joining to a wider set of tables than the immediately connected DIM_CO_Action table will be necessary in order to retrieve the wider context of the information at hand. Whether filtering on one particular account or by dates the basic query will be similar. No particular filtering on the dimensions is required as the versions of the DWSK captured at the time will be the only ones present. Again, if this data is to be linked to a wider set of data, then the link should be made via the Account SSK, as in the query below:

```sql
SELECT
       dacc.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_AccountEventsTrx fae
       JOIN DIM_CO_Action dact
              ON dact.DIM_CO_Action_DWSK = fae.DIM_CO_Action_DWSK
       JOIN DIM_CO_ActionType dactt
              ON dactt.DIM_CO_ActionType_DWSK = dact.DIM_CO_ActionType_DWSK
       JOIN DIM_CO_Episode de
              ON de.DIM_CO_Episode_DWSK = dact.DIM_CO_Episode_DWSK
       JOIN DIM_CO_EpisodeType det
              ON de.DIM_CO_EpisodeType_DWSK = det.DIM_CO_EpisodeType_DWSK
       JOIN DIM_CO_Account dacc
              ON dacc.DIM_CO_Account_DWSK = de.DIM_CO_Account_DWSK
```

6.3. FACT_CO_AccountOnHoldAS

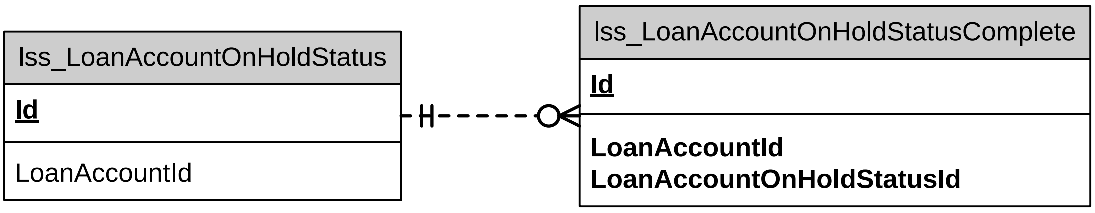
6.3.1. Purpose and How it Works

This table stores information on any Accounts marked as being on hold. Part of the primary key of the table is the column FACT_CO_AccountOnHoldAS_SSK which is a copy of the primary key of the lss_AccountOnHoldStatus table within Lss - lss_AccountOnHoldStatus.Id. This allows connections to be made out to the lss_AccountOnHoldStatus table via the Synonyms.

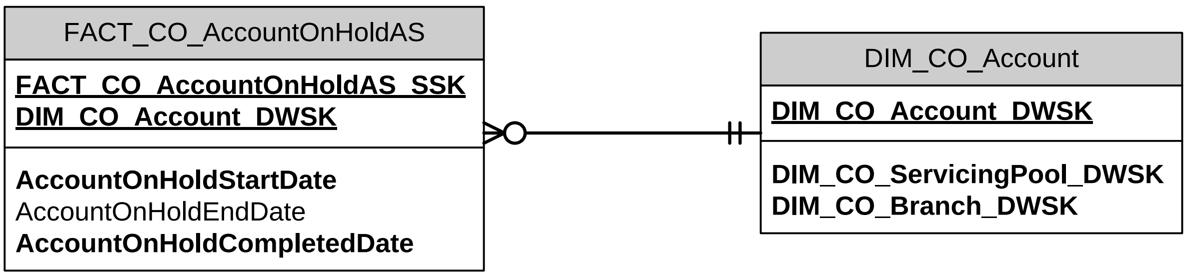
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.3.2. Place in Conceptual Model

This table sources its data from the Account On Hold data in Servicing.


Iss_LoanAccountOnHoldStatus
Id
LoanAccountId
Iss_LoanAccountOnHoldStatusComplete
Id
LoanAccountId
LoanAccountOnHoldStatusId

6.3.3. Star Schema



6.3.4. Linked Dimension Tables

- DIM_CO_Account

6.3.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries caused by SCD version. This can be done by either selecting the current version of everything:

```
SELECT
       da.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_AccountOnHoldAS faoh
       JOIN DIM_CO_Account da
              ON da.DIM_CO_Account_DWSK =faoh.DIM_CO_Account_DWSK
              AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       da.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_AccountOnHoldAS faoh
       JOIN DIM_CO_Account da
              ON da.DIM_CO_Account_DWSK = faoh.DIM_CO_Account_DWSK
              AND da.SCDStartDate <= @ReportDate
              AND da.SCDEndDate > @ReportDate
```

6.4. FACT_CO_AccountStatusActionsAS

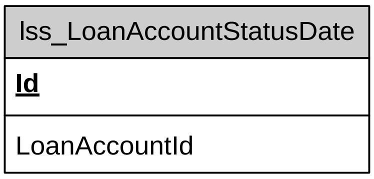
6.4.1. Purpose and How it Works

This table records the Actions (or dates of future events) that will affect the status of an Account. The primary key of the table is the column FACT_CO_AccountStatusActionsAS_SSK which is a copy of the primary key of the lss_LoanAccountStatusDate table within Lss - lss_LoanAccountStatusDate.Id. This allows connections to be made out to the lss_LoanAccountStatusDate table via the Synonyms.

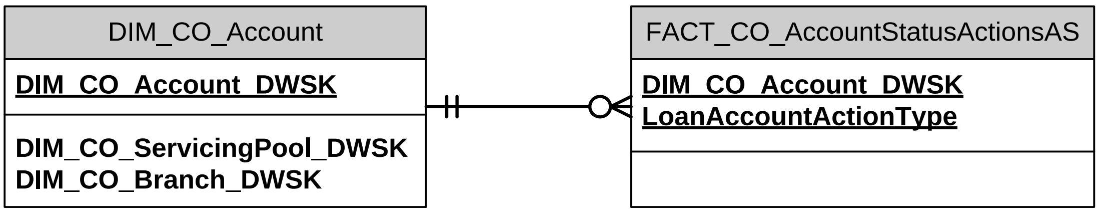
The table works as an Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and DWSKs are update to point to the latest version along with other values. This means that any previous values held in this table will be overwritten and no history kept.

6.4.2. Place in Conceptual Model

This table takes its data from the Account Actions information in Servicing



6.4.3. Star Schema



6.4.4. Linked Dimension Tables

- DIM_CO_Account

6.4.5. Query Hints

Because only one version of the DWSK is held, queries against this table are simple and require no filtering on the dimension table. When joining the results to other Fact table queries, ensure to join on the SSK however.

```
SELECT
       da.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_AccountStatusActionsAS fasa
       JOIN DIM_CO_Account da
              ON da.DIM_CO_Account_DWSK = fasa.DIM_CO_Account_DWSK
```

6.5. FACT_CO_AccountStatusTrx

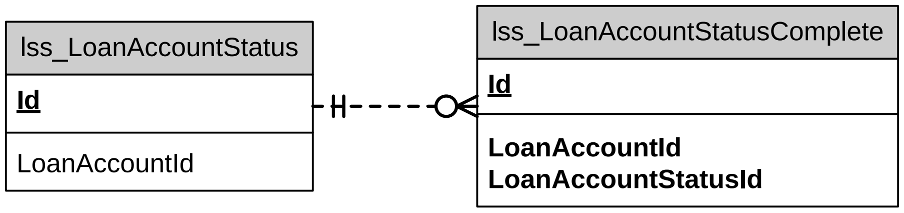
6.5.1. Purpose and How it Works

This table stores information on Account statuses. The primary key of the table is the column FACT_CO_AccountEvents_SSK which is a copy of the primary key of the lss_LoanAccountStatus table within Servicing - lss_LoanAccountStatus.Id. This allows connections to be made out to the lss_LoanAccountStatus table via the Synonyms.

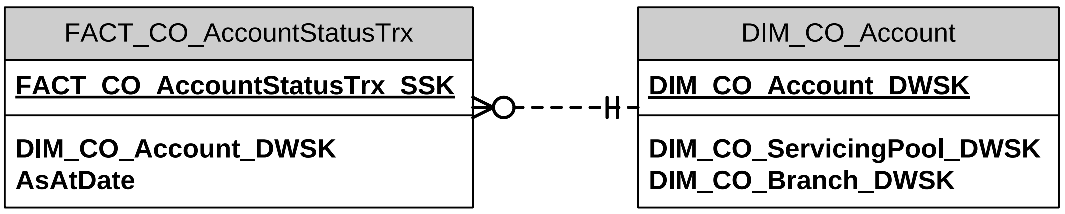
The table works as an adapted Transactional Fact table i.e. new records are added as new Actions are added and DWSKs are only stored once. Fields such as dates are updated as they change. There is a badly named field in this table – AsAtDate- which actually represents the StartDate of the the status.

6.5.2. Place in Conceptual Model

This table take its data from the Account Status information in Servicing



6.5.3. Star Schema



6.5.4. Linked Dimension Tables

- DIM_CO_Account

6.5.5. Query Hints

When querying this table, no particular filtering on the dimensions is required as the versions of the DWSK captured at the time will be the only ones present. Again, if this data is to be linked to a wider set of data, then the link should be made via the Account SSK, as in the query below:

```
SELECT
       da.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_AccountStatusTrx fas
       JOIN DIM_CO_Account da
              ON da.DIM_CO_Account_DWSK = fas.DIM_CO_Account_DWSK
```

6.6. FACT_CO_AddressHistoryAS

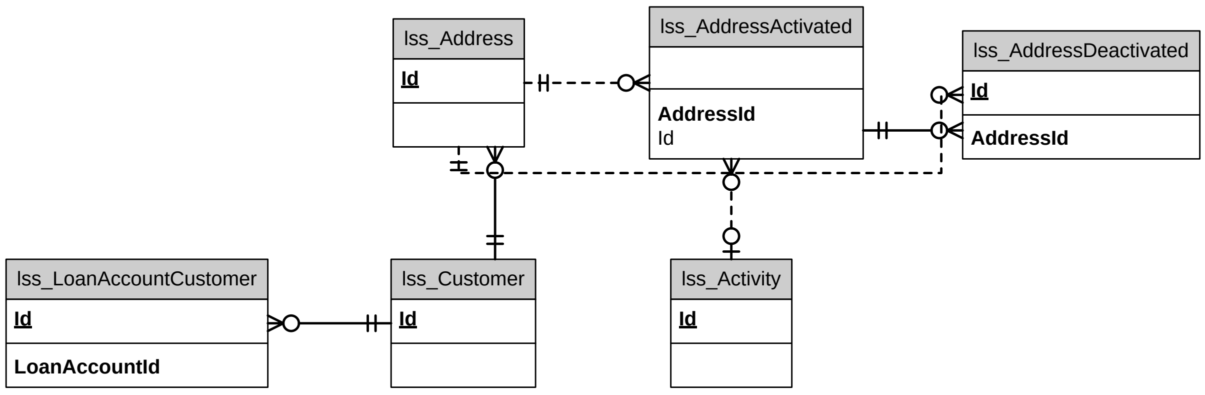
6.6.1. Purpose and How it Works

This table is used to track the history of addresses for Customers. Each address is given an “Activated” and “Deactivated” date, corresponding to the Start and End date for that address. It is an Accumulating Snapshot fact table, meaning that all data is updated to the latest version and in this case includes the foreign key DWSKs to be the latest version at the time the load or update is made, so there should only be one Customer DWSK per Customer SSK per Address.

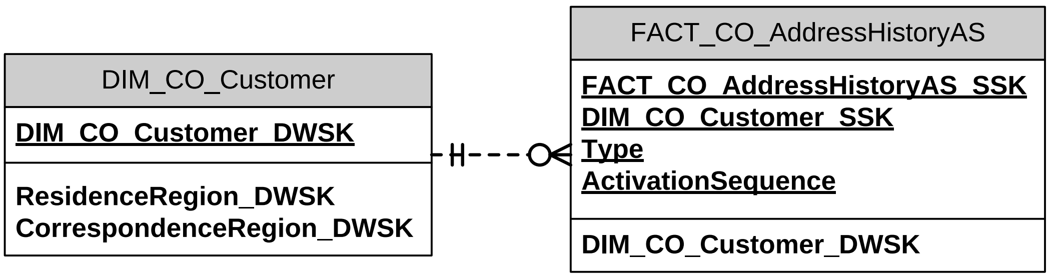
The column FACT_CO_AddressHistoryAS_SSK is a VARCHAR(50) representation of the Id field from the table Iss_Address and can be used to link out to the wider database using the synonyms.

6.6.2. Place in Conceptual Model

This table takes its data from the Address activation records in Servicing and links it to those Customers currently linked to Accounts.



6.6.3. Star Schema



6.6.4. Linked Dimension Tables

- DIM_CO_Customer

6.6.5. Query Hints

This table is straightforward to query as there should only be one result per Customer_SSK, per Address as the following shows:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,addr.<Desired Fields>
FROM FACT_CO_AddressHistoryAS addr
       JOIN DIM_CO_Customer cust
              ON addr.DIM_CO_Customer_DWSK = cust.DIM_CO_Customer_DWSK
```

If combining with other Fact table queries, the join should be made based on the DIM_CO_Customer_SSK and not the DWSK.

6.7. FACT_CO_ADVANCE

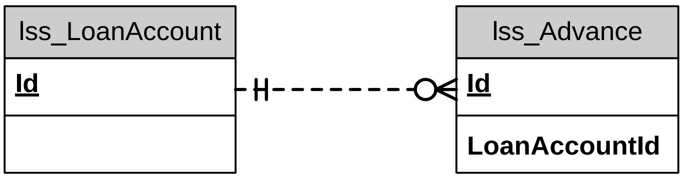
6.7.1. Purpose and How it Works

This table stores information at the Advance level. It works as a Snapshot Fact table i.e. new records are added on a daily basis and DWSKs are only stored once and not updated or multiplied.

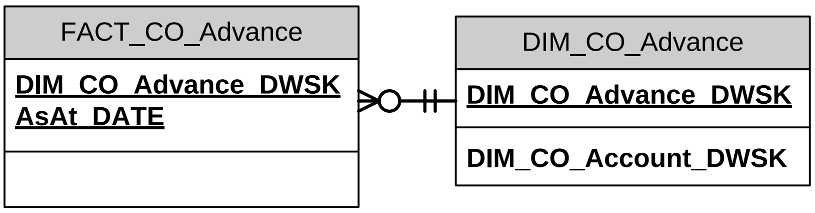
When Advances close, a configurable grace period (typically set to 100 days) is allowed before fact records stop being loaded

6.7.2. Place in Conceptual Model

This data sources its data from the Advance table in Servicing and also draws in the Security value used for LTV calculations in from the Account to which it is connected.



6.7.3. Star Schema



6.7.4. Linked Dimension Tables

- DIM_CO_Advance

6.7.5. Query Hints

The main thing to always bear in mind when querying a Snapshot Fact table is to always filter on a specific date in order to retrieve only one result per entity. To join the results of this Fact table to others, base the joins on the SSK as in the query below. No filtering of the dimension is required as the table will only hold one result per entity as enforced by the primary key.

```
SELECT
       dacc.DIM_CO_Advance_SSK
       ,<Desired Fields>
FROM FACT_CO_Advance fadv
       JOIN DIM_CO_Advance dadv
              ON dadv.DIM_CO_Advance_DWSK = fadv.DIM_CO_Advance_DWSK

WHERE AsAt_DATE = @ReportDate
```

6.8. FACT_CO_AdvanceSourceAS

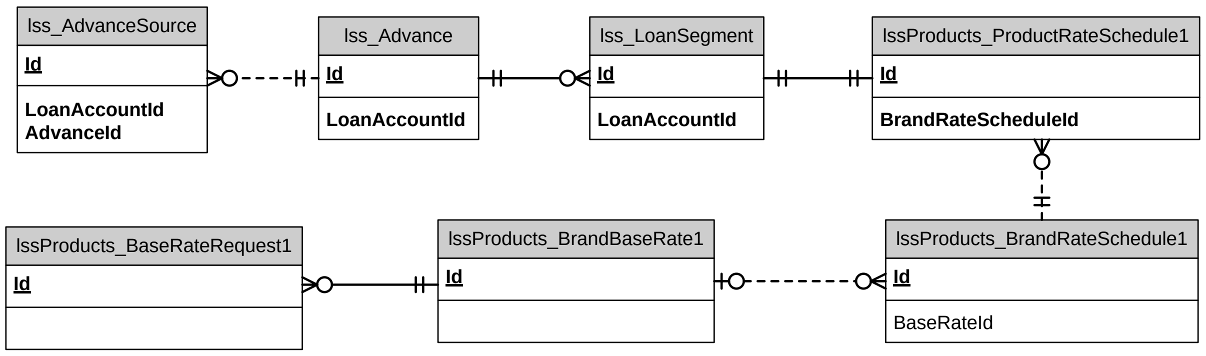
6.8.1. Purpose and How it Works

This table stores information on the source of Advance. Part of the primary key of the table is the column FACT_CO_AdvanceSourceAS_SSK which is a copy of the primary key of the Iss_AdvanceSource table within Lss - Iss_AdvanceSource.Id. This allows connections to be made out to the Iss_AdvanceSource table via the Synonyms.

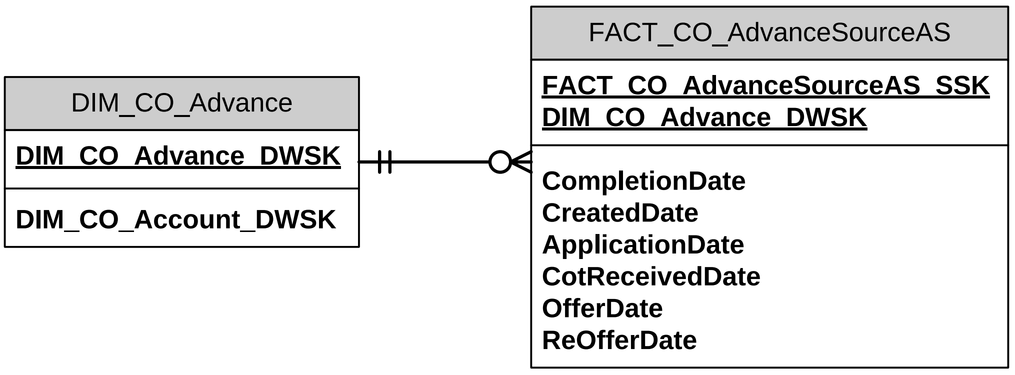
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new Advances are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.8.2. Place in Conceptual Model

This table retrieves most of its data from the Advance Source table in Servicing. It also links out to the Servicing Product information via the Segments in order to retrieve information regarding the long term interest rates.



6.8.3. Star Schema



6.8.4. Linked Dimension Tables

- DIM_CO_Advance

6.8.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       da.DIM_CO_Advance_SSK
       --,<Desired Fields>
FROM FACT_CO_AdvanceSourceAS fas
       JOIN DIM_CO_Advance da
              ON da.DIM_CO_Advance_DWSK = fas.DIM_CO_Advance_DWSK
              AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       da.DIM_CO_Advance_SSK
       --,<Desired Fields>
FROM FACT_CO_AdvanceSourceAS fas
       JOIN DIM_CO_Advance da
              ON da.DIM_CO_Advance_DWSK = fas.DIM_CO_Advance_DWSK
              AND da.SCDStartDate <= @ReportDate
              AND da.SCDEndDate > @ReportDate
```

6.9. FACT_CO_AffordabilityCheckNVP

6.9.1. Purpose and How it Works

This table has been deprecated.

6.10. FACT_CO_AliasHistoryAS

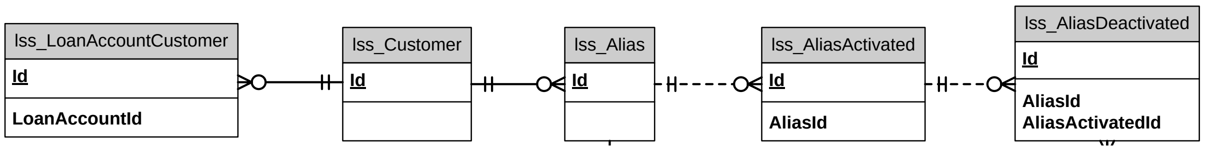
6.10.1. Purpose and How it Works

This table stores the history of aliases for Customers. Each alias is given an Activation and Deactivation date corresponding to the Start and End dates for when the alias was applicable.

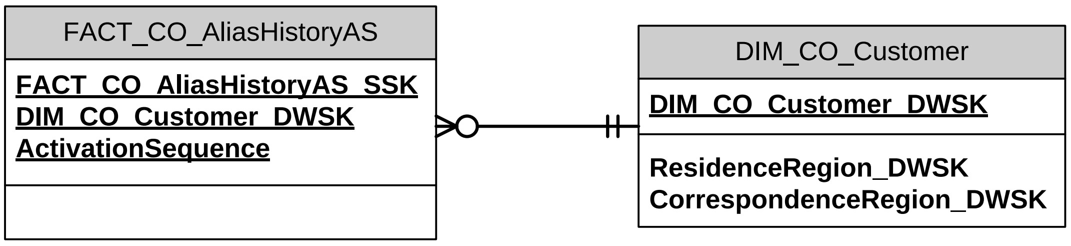
It is an Accumulating Snapshot table and as such only holds the latest version of the data with the exception of the DWSK version of the Customer, so it is possible to return "duplicate" entries in the result set unless care is taken to only return a single record per customer and address.

6.10.2. Place in Conceptual Model

This table sources its data from the Customer Alias tables in Servicing.



6.10.3. Star Schema



6.10.4. Linked Dimension Tables

- DIM_CO_Customer

6.10.5. Query Hints

To ensure that only one record per Customer and Alias is returned, use the either of the following approaches:

1) To return the current Customer data only, then use a query like the following:

```
SELECT
       cust.DIM_CO_Customer_SSK
       ,cust.<Desired Fields>
       ,ah.<Desired Fields>
FROM FACT_CO_AliasHistoryAS ah
       JOIN DIM_CO_Customer cust
              ON cust.DIM_CO_Customer_DWSK = ah.DIM_CO_Customer_DWSK
WHERE SCDStatus = 'C'
```

2) To return a historic customer version use a query like the following:

```
SELECT
        cust.DIM_CO_Customer_SSK
        ,cust.<Desired Fields>
        ,ah.<Desired Fields>
FROM FACT_CO_AliasHistoryAS ah
        JOIN DIM_CO_Customer cust
                ON cust.DIM_CO_Customer_DWSK = ah.DIM_CO_Customer_DWSK
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

6.11. FACT_CO_ArrearsEpisode

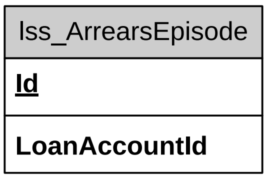
6.11.1. Purpose and How it Works

This table stores information on any Arrears Episodes that have been created against an Account. The primary key of the table is the column FACT_CO_ArrearsEpisode_SSK which is a copy of the primary key of the lss_ArrearsEpisode table within Lss - lss_ArrearsEpisode.Id. This allows connections to be made out to the lss_ArrearsEpisode table via the Synonyms.

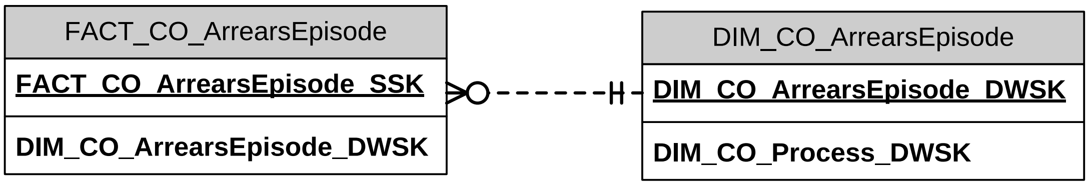
The table works as an Accumulating Snapshot Fact table i.e. new records are added as new statuses are added. Values are updated to the latest value, including DWSKs. This means that any previous values held in this table will be overwritten and no history kept.

6.11.2. Place in Conceptual Model

This table takes its data exclusively from the Arrears Episode table in Servicing.



6.11.3. Star Schema


FACT_CO_ArrearsEpisode
FACT_CO_ArrearsEpisode_SSK
DIM_CO_ArrearsEpisode_DWSK
DIM_CO_ArrearsEpisode
DIM_CO_ArrearsEpisode_DWSK
DIM_CO_Process_DWSK

6.11.4. Linked Dimension Tables

- DIM_CO_ArrearsEpisode

6.11.5. Query Hints

Querying this table is straightforward as everything is updated to the latest DWSK. Again, when joining to other Fact table results, make sure to join on the SSK (in this case the Account SSK) as in the query below:

```
SELECT
        da.DIM_CO_Account_SSK
        --,<Desired Fields>
FROM FACT_CO_ArrearsEpisode fae
        JOIN DIM_CO_ArrearsEpisode dae
                ON dae.DIM_CO_ArrearsEpisode_DWSK = fae.DIM_CO_ArrearsEpisode_DWSK
        JOIN DIM_CO_Process dp
```

```
                ON dp.DIM_CO_Process_DWSK = dae.DIM_CO_Process_DWSK
        JOIN DIM_CO_Account da
                ON da.DIM_CO_Account_DWSK = dp.DIM_CO_Account_DWSK
```

6.12. FACT_CO_AssetSecurityChargesAS

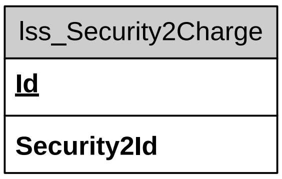
6.12.1. Purpose and How it Works

This table stores information on any charges held on Securities. Part of the primary key of the table is the column FACT_CO_AssetSecurityChargesAS_SSK which is a copy of the primary key of the Iss_Security2Charge table within Lss - Iss_Security2Charge.Id. This allows connections to be made out to the Iss_Security2Charge table via the Synonyms.

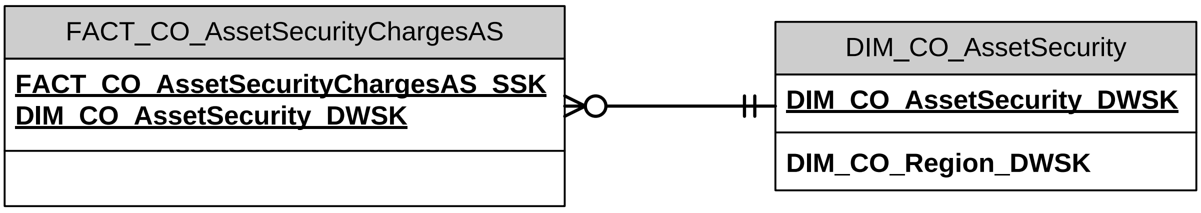
The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.12.2. Place in Conceptual Model

This table takes its data solely from the Security Charges information in Servicing.



6.12.3. Star Schema


FACT_CO_AssetSecurityChargesAS
FACT_CO_AssetSecurityChargesAS_SSK
DIM_CO_AssetSecurity_DWSK
DIM_CO_AssetSecurity
DIM_CO_AssetSecurity_DWSK
DIM_CO_Region_DWSK

6.12.4. Linked Dimension Tables

- DIM_CO_AssetSecurity

6.12.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
        da.DIM_CO_AssetSecurity_SSK
        --,<Desired Fields>
FROM FACT_CO_AssetSecurityChargesAS fasc
        JOIN DIM_CO_AssetSecurity da
                ON da.DIM_CO_AssetSecurity_DWSK = fasc.DIM_CO_AssetSecurity_DWSK
                AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
        da.DIM_CO_AssetSecurity_SSK
        --,<Desired Fields>
FROM FACT_CO_AssetSecurityChargesAS fasc
```

```
       JOIN DIM_CO_AssetSecurity da
              ON da.DIM_CO_AssetSecurity_DWSK = fasc.DIM_CO_AssetSecurity_DWSK
              AND da.SCDStartDate <= @ReportDate
              AND da.SCDEndDate > @ReportDate
```

6.13. FACT_CO_AuditItemTrx

6.13.1. Purpose and How it Works

This table has now been deprecated.

6.14. FACT_CO_BrandFeatureValueNVP

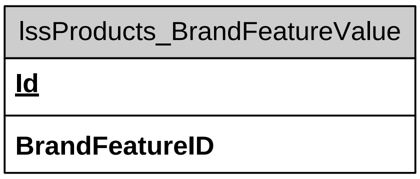
6.14.1. Purpose and How it Works

This table stores information on Brand Feature values associated with Products. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get "duplicate" records in queries due to the multiple SCD versions present in the results.

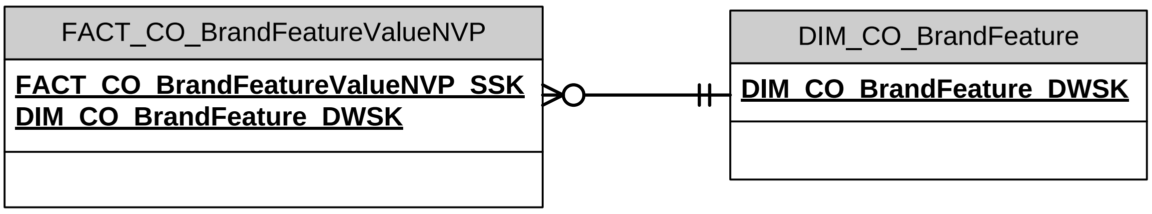
The table stores information generically as name value pairs and is normally used in a wider scope to identify the particular attributes of Products. For this reason queries will normally involve the combination of this table and with IF_COProduct_COBFeature to establish which are relevant to which Products.

6.14.2. Place in Conceptual Model

This table takes its data exclusively from the Brand Feature Value table in Servicing.



6.14.3. Star Schema



6.14.4. Linked Dimension Tables

- DIM_CO_BrandFeature

6.14.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dbf.DIM_CO_BrandFeature_SSK
```

```
	--,<Desired Fields>
FROM FACT_CO_BrandFeatureValueNVP fbfv
	JOIN DIM_CO_BrandFeature dbf
		ON fbfv.DIM_CO_BrandFeature_DWSK = dbf.DIM_CO_BrandFeature_DWSK
		AND dbf.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
	dbf.DIM_CO_BrandFeature_SSK
	--,<Desired Fields>
FROM FACT_CO_BrandFeatureValueNVP fbfv
	JOIN DIM_CO_BrandFeature dbf
		ON fbfv.DIM_CO_BrandFeature_DWSK = dbf.DIM_CO_BrandFeature_DWSK
		AND dbf.SCDStartDate <= @ReportDate
		AND dbf.SCDEndDate > @ReportDate
```

One other technique, which may be of benefit is the use of PIVOT to create single records per DIM_CO_BrandFeature. Unfortunely, due to the limitations of the PIVOT syntax, the name of the values you are looking for must be known ahead of time. However, if this is the case then you can use a query like the below to achieve this:

```
WITH cte_Source AS
(
	SELECT
		dbf.DIM_CO_BrandFeature_SSK
		,fbfv.KeyName
		,fbfv.Value
	FROM FACT_CO_BrandFeatureValueNVP fbfv
		JOIN DIM_CO_BrandFeature dbf
			ON fbfv.DIM_CO_BrandFeature_DWSK = dbf.DIM_CO_BrandFeature_DWSK
			AND dbf.SCDStatus = 'C'
)
SELECT *
FROM cte_Source
PIVOT
(
	MAX(Value)
	FOR KeyName IN ([KeyName1],[KeyName2], /*...*/ [KeyNameN])
) pvt
```

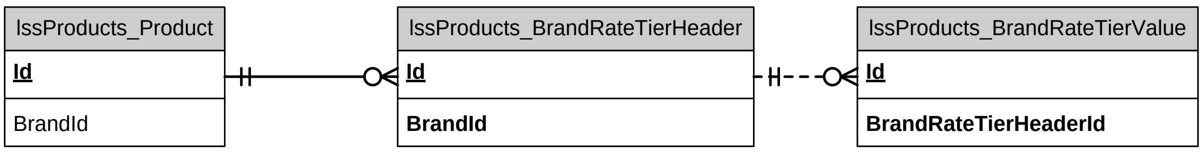
6.15. FACT_CO_BRANDRATETIERVALUEAS

6.15.1. Purpose and How it Works

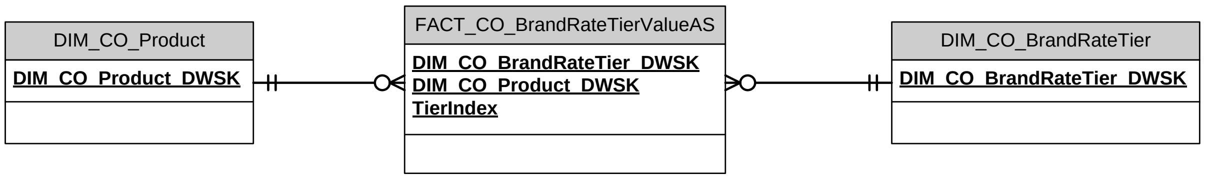
This table stores information on tier ranges associated with interest rates. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get "duplicate" records in queries due to the multiple SCD versions present in the results.

6.15.2. Place in Conceptual Model

This table takes its data from the Rate Tier information in Servicing Products and links it back to the Servicing Product.


IssProducts_Product
Id
BrandId
IssProducts_BrandRateTierHeader
Id
BrandId
IssProducts_BrandRateTierValue
Id
BrandRateTierHeaderId

6.15.3. Star Schema



6.15.4. Linked Dimension Tables

- DIM_CO_BrandRateTier
- DIM_CO_Product

6.15.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       --<Desired Fields>
FROM FACT_CO_BrandRateTierValueAS fbrt
       JOIN DIM_CO_BrandRateTier dbrt
              ON dbrt.DIM_CO_BrandRateTier_DWSK = fbrt.DIM_CO_BrandRateTier_DWSK
              AND dbrt.SCDStatus = 'C'
       JOIN DIM_CO_Product dp
              ON dp.DIM_CO_Product_DWSK = fbrt.DIM_CO_Product_DWSK
              AND dp.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       --<Desired Fields>
FROM FACT_CO_BrandRateTierValueAS fbrt
       JOIN DIM_CO_BrandRateTier dbrt
              ON dbrt.DIM_CO_BrandRateTier_DWSK = fbrt.DIM_CO_BrandRateTier_DWSK
              AND dbrt.SCDStartDate <= @ReportDate
              AND dbrt.SCDEndDate > @ReportDate
       JOIN DIM_CO_Product dp
              ON dp.DIM_CO_Product_DWSK = fbrt.DIM_CO_Product_DWSK
              AND dp.SCDStartDate <= @ReportDate
              AND dp.SCDEndDate > @ReportDate
```

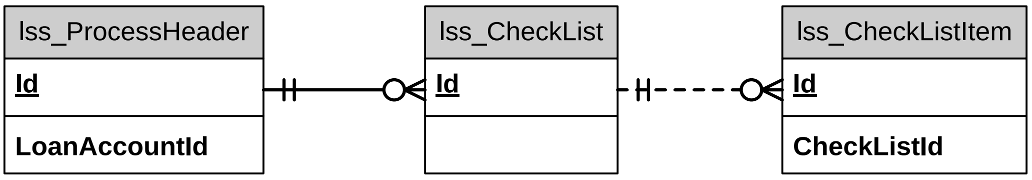
6.16. FACT_CO_CheckListItemNVP

6.16.1. Purpose and How it Works

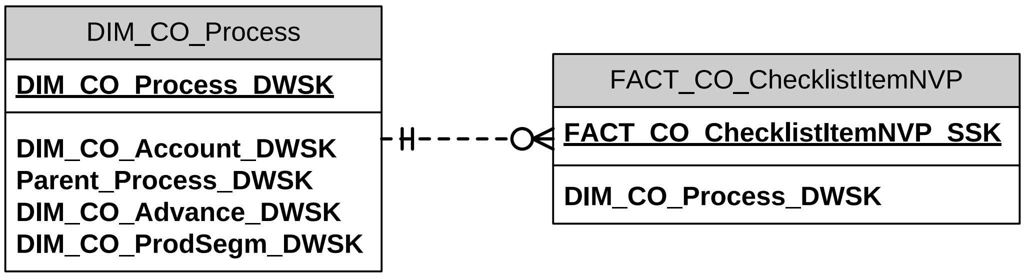
This table stores information on any checklists assocated with Products. The table works as an Accumulating Snapshot Fact table i.e. new records are added as new Checklists are added. Other values are updated, including DWSKs. The table is a Name Value Pair and a so some further logic may need to be implemented (e.g. PIVOT syntax) in order to manipulate the data into a usable format.

6.16.2. Place in Conceptual Model

This table takes its data from the Checklist information in Servicing and relates it back to the linked Process.



6.16.3. Star Schema



6.16.4. Linked Dimension Tables

- DIM_CO_Process

6.16.5. Query Hints

Querying this table is straightforward as only the latest DWSK is held. Further benefit may be added by using a technique like PIVOT to only return a single row per DIM_CO_Process_SSK, such as in the query below:

```
WITH cte_Source AS
(
	SELECT
		dp.DIM_CO_Process_SSK
		,fcl.ChecklistItemName
		,fcl.Value
	FROM FACT_CO_ChecklistItemNVP fcl
		JOIN DIM_CO_Process dp
			ON dp.DIM_CO_Process_DWSK = fcl.DIM_CO_Process_DWSK
)
SELECT *
FROM cte_Source
PIVOT
(
	MAX(Value)
	FOR CheckListItemName IN
([CheckListItemName1],[CheckListItemName2],/*...*/[CheckListItemNameN])
) pvt
```

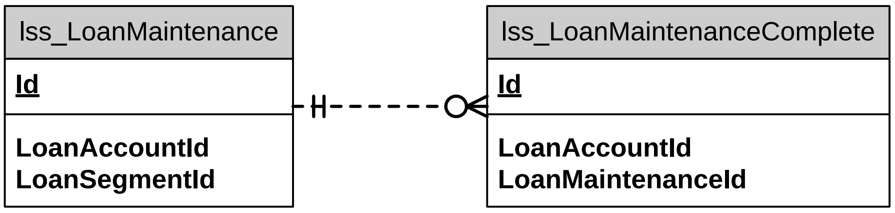
6.17. FACT_CO_ContractChangeAS

6.17.1. Purpose and How it Works

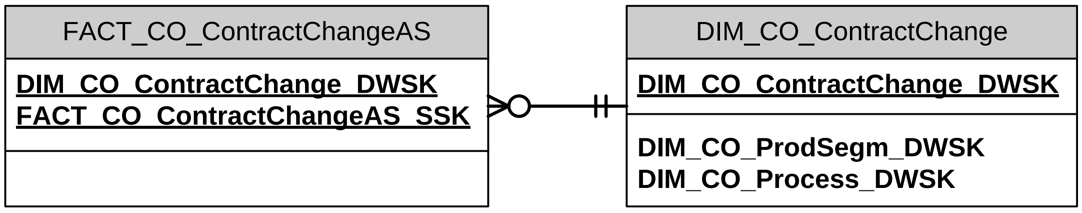
This table stores information on any contract changes on a Segment. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.17.2. Place in Conceptual Model

This table sources its data from the loan maintenance tables in Servicing to report any Contract Changes.



6.17.3. Star Schema



6.17.4. Linked Dimension Tables

- DIM_CO_ContractChange

6.17.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dp.DIM_CO_ProdSegm_SSK
       --,<Desired Fields>
FROM FACT_CO_ContractChangeAS fcc
       JOIN DIM_CO_ContractChange dcc
              ON dcc.DIM_CO_ContractChange_DWSK = fcc.DIM_CO_ContractChange_DWSK
              AND dcc.SCDStatus = 'C'
       JOIN DIM_CO_ProdSegm dp
              ON dp.DIM_CO_ProdSegm_DWSK = dcc.DIM_CO_ProdSegm_DWSK
              AND dp.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       dp.DIM_CO_ProdSegm_SSK
       --,<Desired Fields>
FROM FACT_CO_ContractChangeAS fcc
       JOIN DIM_CO_ContractChange dcc
              ON dcc.DIM_CO_ContractChange_DWSK = fcc.DIM_CO_ContractChange_DWSK
              AND dcc.SCDStartDate <= @ReportDate
              AND dcc.SCDEndDate > @ReportDate
       JOIN DIM_CO_ProdSegm dp
              ON dp.DIM_CO_ProdSegm_DWSK = dcc.DIM_CO_ProdSegm_DWSK
              AND dp.SCDStartDate <= @ReportDate
              AND dp.SCDEndDate > @ReportDate
```

6.18. FACT_CO_CreditReference

6.18.1. Purpose and How it Works

See Origination entry

6.19. FACT_CO_CreditReferenceCustomer

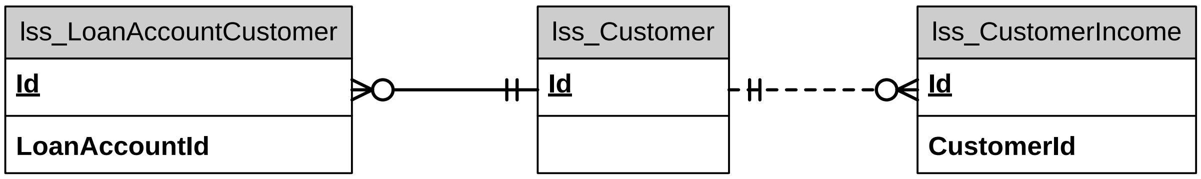
6.19.1. Purpose and How it Works

See Origination entry.

6.20. FACT_CO_CustomerIncomeAS

6.20.1. Place in Conceptual Model

This table sources its data from the Customer Income records in Servicing.



6.20.2. Star Schema

See Origination entry

6.20.3. Linked Dimension Tables

See Origination entry

6.20.4. Query Hints

See Origination entry

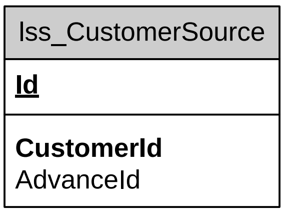
6.21. FACT_CO_CustomerSourceAS

6.21.1. Purpose and How it Works

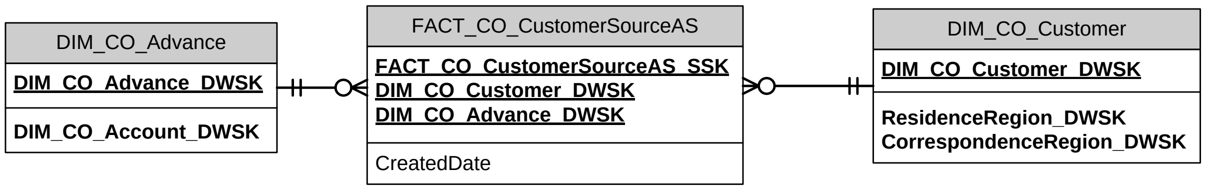
This table stores information on the source of Customer information. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new records are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.21.2. Place in Conceptual Model

This table sources its data from the Customer Source records in Servicing



6.21.3. Star Schema



6.21.4. Linked Dimension Tables

- DIM_CO_Advance
- DIM_CO_Customer

6.21.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dc.DIM_CO_Customer_SSK
       ,da.DIM_CO_Advance_SSK
       --,<Desired Fields>
FROM FACT_CO_CustomerSourceAS fcs
       JOIN DIM_CO_Customer dc
              ON dc.DIM_CO_Customer_DWSK = fcs.DIM_CO_Customer_DWSK
              AND dc.SCDStatus = 'C'
       JOIN DIM_CO_Advance da
              ON da.DIM_CO_Advance_DWSK = fcs.DIM_CO_Advance_DWSK
              AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       dc.DIM_CO_Customer_SSK
       ,da.DIM_CO_Advance_SSK
       --,<Desired Fields>
FROM FACT_CO_CustomerSourceAS fcs
       JOIN DIM_CO_Customer dc
              ON dc.DIM_CO_Customer_DWSK = fcs.DIM_CO_Customer_DWSK
              AND dc.SCDStartDate <= @ReportDate
              AND dc.SCDEndDate > @ReportDate
       JOIN DIM_CO_Advance da
              ON da.DIM_CO_Advance_DWSK = fcs.DIM_CO_Advance_DWSK
              AND da.SCDStartDate <= @ReportDate
              AND da.SCDEndDate > @ReportDate
```

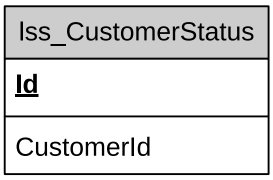
6.22. FACT_CO_CUSTOMERSTATUSAS

6.22.1. Purpose and How it Works

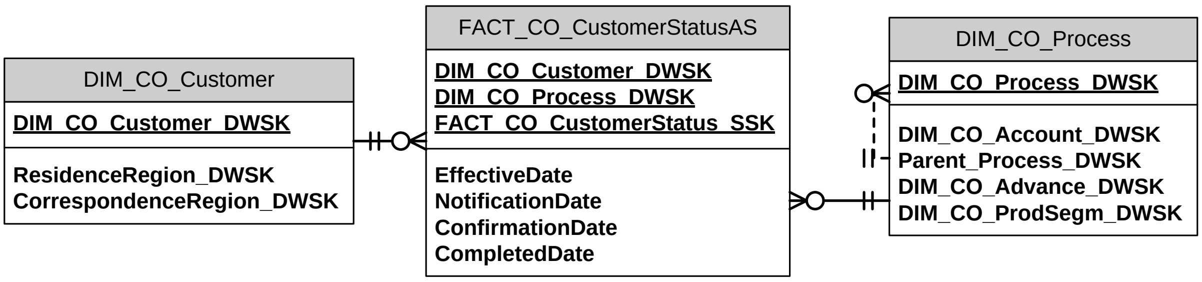
This table stores information on any Customer Statuses. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.22.2. Place in Conceptual Model

This table takes its data from the Customer Status information in Servicing



6.22.3. Star Schema



6.22.4. Linked Dimension Tables

- DIM_CO_Customer
- DIM_CO_Process

6.22.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dc.DIM_CO_Customer_SSK
       ,dp.DIM_CO_Process_SSK
       --,<Desired Fields>
FROM FACT_CO_CustomerStatusAS fcs
       JOIN DIM_CO_Customer dc
              ON dc.DIM_CO_Customer_DWSK = fcs.DIM_CO_Customer_DWSK
              AND dc.SCDStatus = 'C'
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = fcs.DIM_CO_Process_DWSK
              AND dp.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       dc.DIM_CO_Customer_SSK
       ,dp.DIM_CO_Process_SSK
       --,<Desired Fields>
FROM FACT_CO_CustomerStatusAS fcs
       JOIN DIM_CO_Customer dc
              ON dc.DIM_CO_Customer_DWSK = fcs.DIM_CO_Customer_DWSK
              AND dc.SCDStartDate <= @ReportDate
              AND dc.SCDEndDate > @ReportDate
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = fcs.DIM_CO_Process_DWSK
```

```
                AND dp.SCDStartDate <= @ReportDate
                AND dp.SCDEndDate > @ReportDate
```

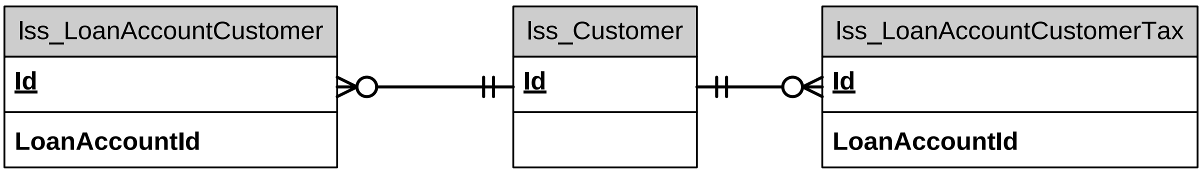
6.23. FACT_CO_CustomerTaxStatusHistoryAS

6.23.1. Purpose and How it Works

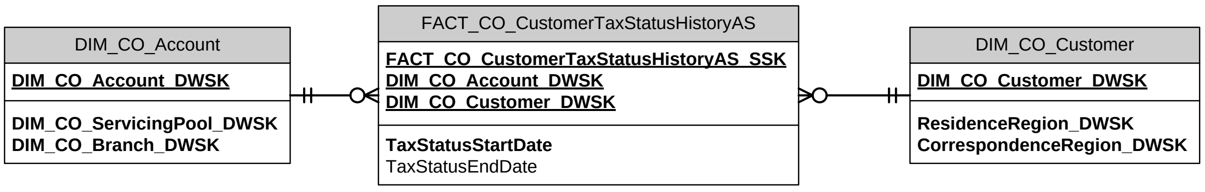
This table stores information on the tax status history of the Customer. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.23.2. Place in Conceptual Model

This table sources its data from the Customer Tax Status records in Servicing



6.23.3. Star Schema



6.23.4. Linked Dimension Tables

- DIM_CO_Customer
- DIM_CO_Account

6.23.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       dc.DIM_CO_Customer_SSK
       ,da.DIM_CO_Account_SSK
       --,<Desired Fields>
FROM FACT_CO_CustomerTaxStatusHistoryAS fcts
       JOIN DIM_CO_Customer dc
              ON dc.DIM_CO_Customer_DWSK = fcts.DIM_CO_Customer_DWSK
              AND dc.SCDStatus = 'C'
       JOIN DIM_CO_Account da
              ON da.DIM_CO_Account_DWSK = fcts.DIM_CO_Account_DWSK
              AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       dc.DIM_CO_Customer_SSK
```

```
        ,da.DIM_CO_Account_SSK
        --,<Desired Fields>
FROM FACT_CO_CustomerTaxStatusHistoryAS fcts
        JOIN DIM_CO_Customer dc
                ON dc.DIM_CO_Customer_DWSK = fcts.DIM_CO_Customer_DWSK
                AND dc.SCDStartDate <= @ReportDate
                AND dc.SCDEndDate > @ReportDate
        JOIN DIM_CO_Account da
                ON da.DIM_CO_Account_DWSK = fcts.DIM_CO_Account_DWSK
                AND da.SCDStartDate <= @ReportDate
                AND da.SCDEndDate > @ReportDate
```

6.24. FACT_CO_DocumentItem

6.24.1. Purpose and How it Works

This table has now been deprecated.

6.25. FACT_CO_DocumentXML

6.25.1. Purpose and How it Works

See the Origination entry.

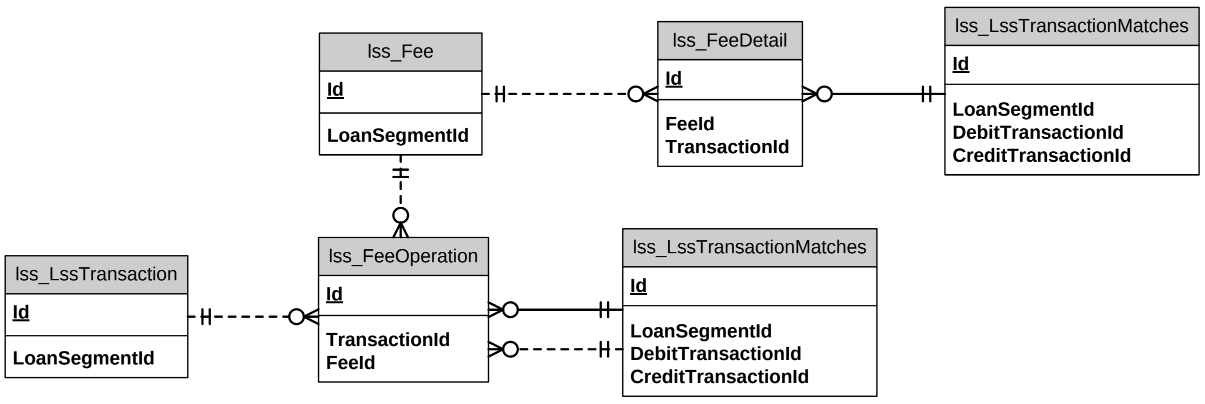
6.26. FACT_CO_FeeAS

6.26.1. Purpose and How it Works

See Origination entry

6.26.2. Place in Conceptual Model

This table takes its data from the Fee information in Servicing, along with their associated transactions.



6.26.3. Star Schema

See Origination entry

6.26.4. Linked Dimension Tables

See Origination entry

6.26.5. Query Hints

See Origination entry

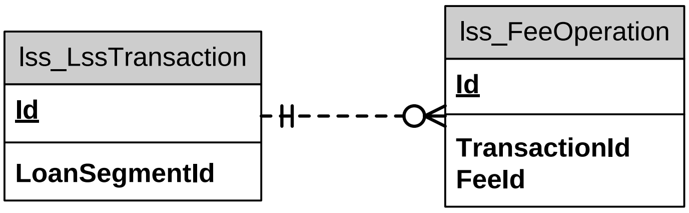
6.27. FACT_CO_FeeOperationsAS

6.27.1. Purpose and How it Works

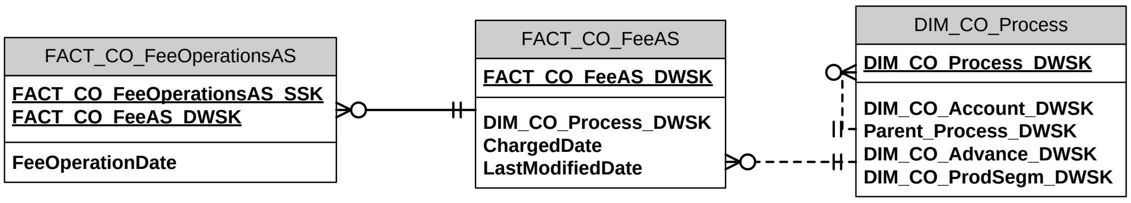
This table stores information on any operations on fees that table place.The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new operations are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get "duplicate" records in queries due to the multiple SCD versions present in the results.

6.27.2. Place in Conceptual Model

This table takes its data from the Fee information in the Servicing database.



6.27.3. Star Schema



6.27.4. Linked Dimension Tables

- See FACT_CO_FeesAS

6.27.5. Query Hints

- See FACT_CO_FeeAS

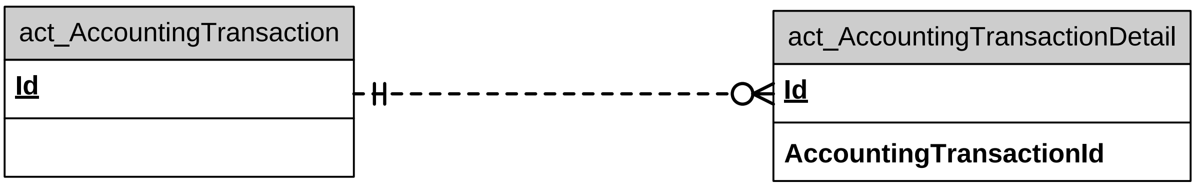
6.28. FACT_CO_GLEntryTrx

6.28.1. Purpose and How it Works

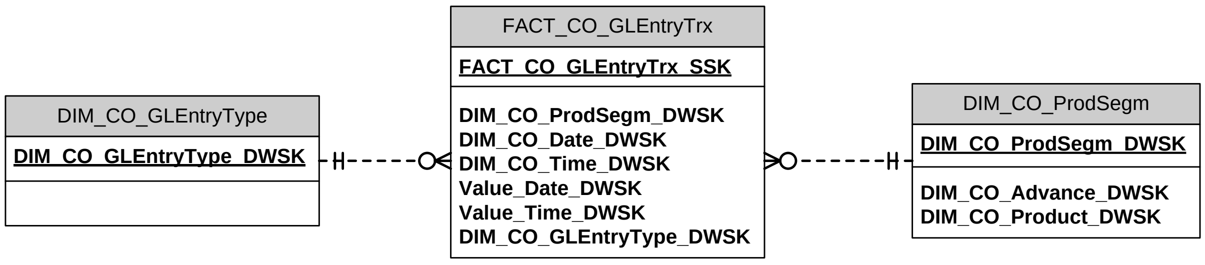
This table stores information on General Ledger transactions. The table works as a Transactional Fact table i.e. new records are added as new transactions are added and DWSKs are only stored once.

6.28.2. Place in Conceptual Model

This table takes its data from the GL Transactions in the LssAccounting database.



6.28.3. Star Schema



6.28.4. Linked Dimension Tables

- DIM_CO_ProdSegm
- DIM_CO_GLEntryType

6.28.5. Query Hints

The main thing to bear in mind when querying this table is the spread of DWSKs that will be present over time. Thus filtering on a particular DWSK should never be done. Queries spanning an extended period should link to other queries via the SSKs:

```
SELECT
       dp.DIM_CO_ProdSegm_SSK
       --,<Desired Fields>
FROM FACT_CO_GLEntryTrx fgl
       JOIN DIM_CO_ProdSegm dp
              ON dp.DIM_CO_ProdSegm_DWSK = fgl.DIM_CO_ProdSegm_DWSK
       JOIN DIM_CO_GLEntryType dgl
              ON dgl.DIM_CO_GLEntryType_DWSK = fgl.DIM_CO_GLEntryType_DWSK
```

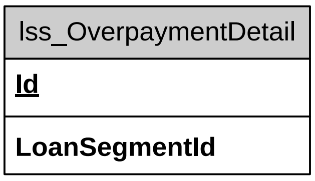
6.29. FACT_CO_OverpaymentDetailTrx

6.29.1. Purpose and How it Works

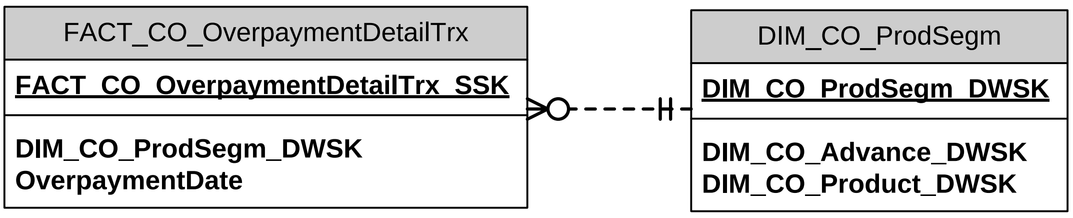
This table stores information on Overpayment transactions. The table works as a Transactional Fact table i.e. new records are added as new transactions are added and DWSKs are only stored once.

6.29.2. Place in Conceptual Model

This table takes its data from the Overpayments information in Servicing.



6.29.3. Star Schema



6.29.4. Linked Dimension Tables

- DIM_CO_ProdSegm

6.29.5. Query Hints

The main thing to bear in mind when querying this table is the spread of DWSKs that will be present over time. Thus filtering on a particular DWSK should never be done. Queries spanning an extended period should link to other queries via the SSKs:

```
SELECT
       dp.DIM_CO_ProdSegm_SSK
       --,<Desired Fields>
FROM FACT_CO_OverpaymentDetailTrx fop
       JOIN DIM_CO_ProdSegm dp
            ON dp.DIM_CO_ProdSegm_DWSK = fop.DIM_CO_ProdSegm_DWSK
```

6.30. FACT_CO_PaymentSchedule

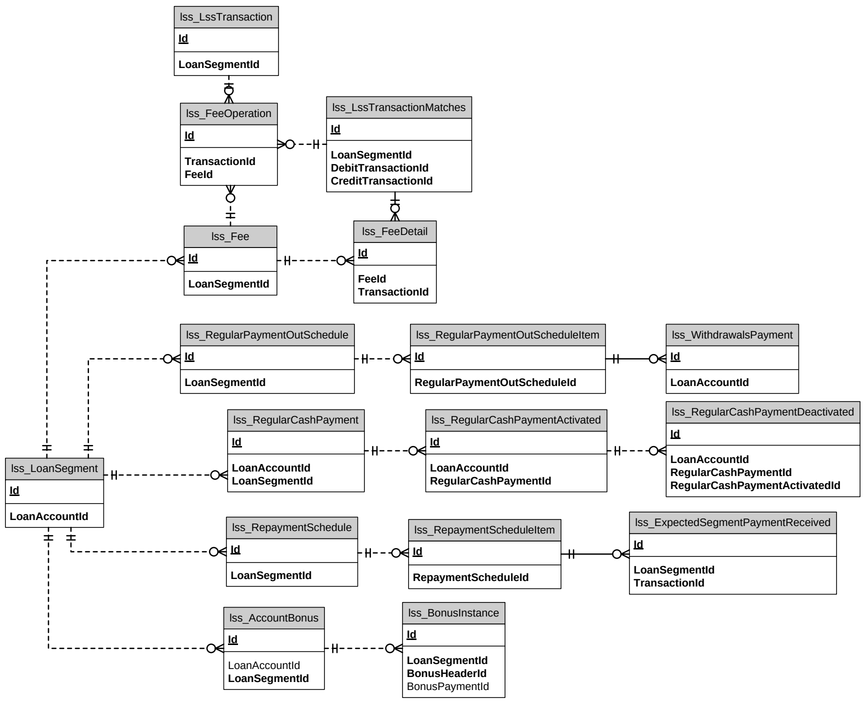
6.30.1. Purpose and How it Works

This table stores information on the current position of a schedule. The table works as a Periodic Snapshot Fact table i.e. new records are added every day and DWSKs are only stored once.

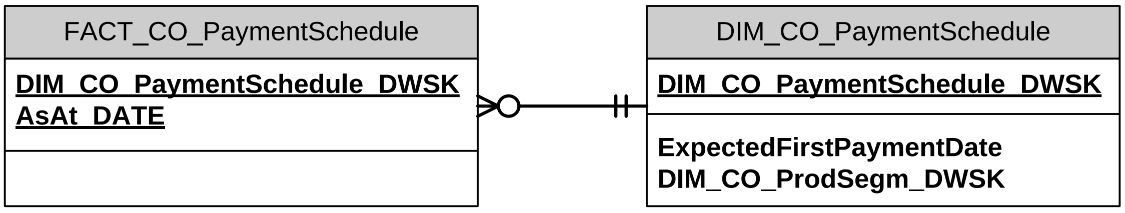
6.30.2. Place in Conceptual Model

This table combines information from a number of different Schedule sources:

- It retrieves information on scheduled repayments (i.e. loan payments) from the Repayment Schedule tables in Servicing
- It retrieves information on scheduled payments out (e.g. cash reserve drawdowns, withdrawals) from the Regular Payment Out and Regular Cash Reserve tables in Servicing
- It retrieves scheduled bonus payments (for Savings Accounts) from the Bonus tables in Servicing
- It retrieves scheduled fee payments from the Fee tables in Servicing.



6.30.3. Star Schema



6.30.4. Linked Dimension Tables

- DIM_CO_PaymentSchedule

6.30.5. Query Hints

Two things need to be taken into account when querying this table:

1) More than one schedule can be present of the same type in the Data Warehouse is one has supersceded another, therefore the one that was current at the time should be filtered on.
2) The particular date when reporting is required in order to only return one result per schedule.

This can be done using a query like:

```
SELECT
       --,<Desired Fields>
FROM FACT_CO_PaymentSchedule fps
       JOIN DIM_CO_PaymentSchedule dps
              ON dps.DIM_CO_PaymentSchedule_DWSK = fps.DIM_CO_PaymentSchedule_DWSK
WHERE fps.AsAt_DATE = @ReportDate
AND dp.IsCurrentStartDate <= @ReportDate
AND dp.IsCurrentEndDate > @ReportDate
```

6.31. FACT_CO_PAYMENTSCHEDULEAS

6.31.1. Purpose and How it Works

6.31.2. Place in Conceptual Model

See FACT_CO_PaymentSchedule entry

6.31.3. Star Schema

6.31.4. Linked Dimension Tables

6.31.5. Query Hints

6.32. FACT_CO_PAYMENTSCHEDULEITEMTRX

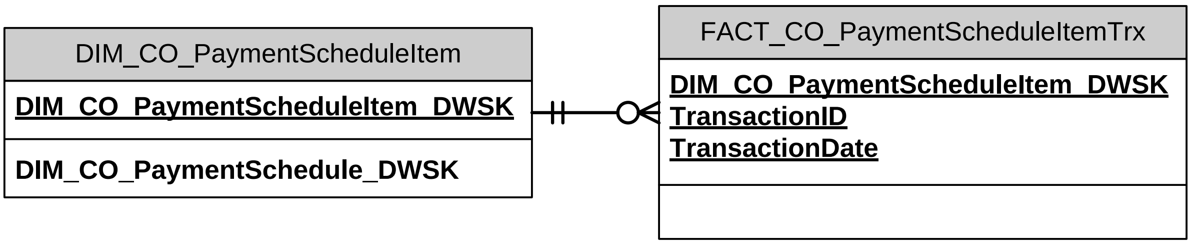
6.32.1. Purpose and How it Works

This table stores information on schedule expectations being met. The table works as an adapted Transactional Fact table i.e. new records are added as new transactions are added and DWSKs are only stored once. Fields such as dates are updated as they change.

6.32.2. Place in Conceptual Model

See FACT_CO_PaymentSchedule entry

6.32.3. Star Schema


DIM_CO_PaymentScheduleItem
DIM_CO_PaymentScheduleItem_DWSK
DIM_CO_PaymentSchedule_DWSK
FACT_CO_PaymentScheduleItemTrx
DIM_CO_PaymentScheduleItem_DWSK
TransactionID
TransactionDate

6.32.4. Linked Dimension Tables

- DIM_CO_PaymentScheduleItem

6.32.5. Query Hints

This table is straightforward to query since the DWSKs are only stored once e.g.

```
SELECT
       --,<Desired Fields>
FROM FACT_CO_PaymentScheduleItemTrx fpsi
       JOIN DIM_CO_PaymentScheduleItem dpsi
              ON dpsi.DIM_CO_PaymentScheduleItem_DWSK =
fpsi.DIM_CO_PaymentScheduleItem_DWSK
       JOIN DIM_CO_PaymentSchedule dps
              ON dps.DIM_CO_PaymentSchedule_DWSK =
dpsi.DIM_CO_PaymentScheduleItem_DWSK
```

6.33. FACT_CO_ProdSegm

6.33.1. Purpose and How it Works

This table stores information at the Segment level. It works as a Snapshot Fact table i.e. new records are added on a daily basis and DWSKs are only stored once and not updated or multiplied.

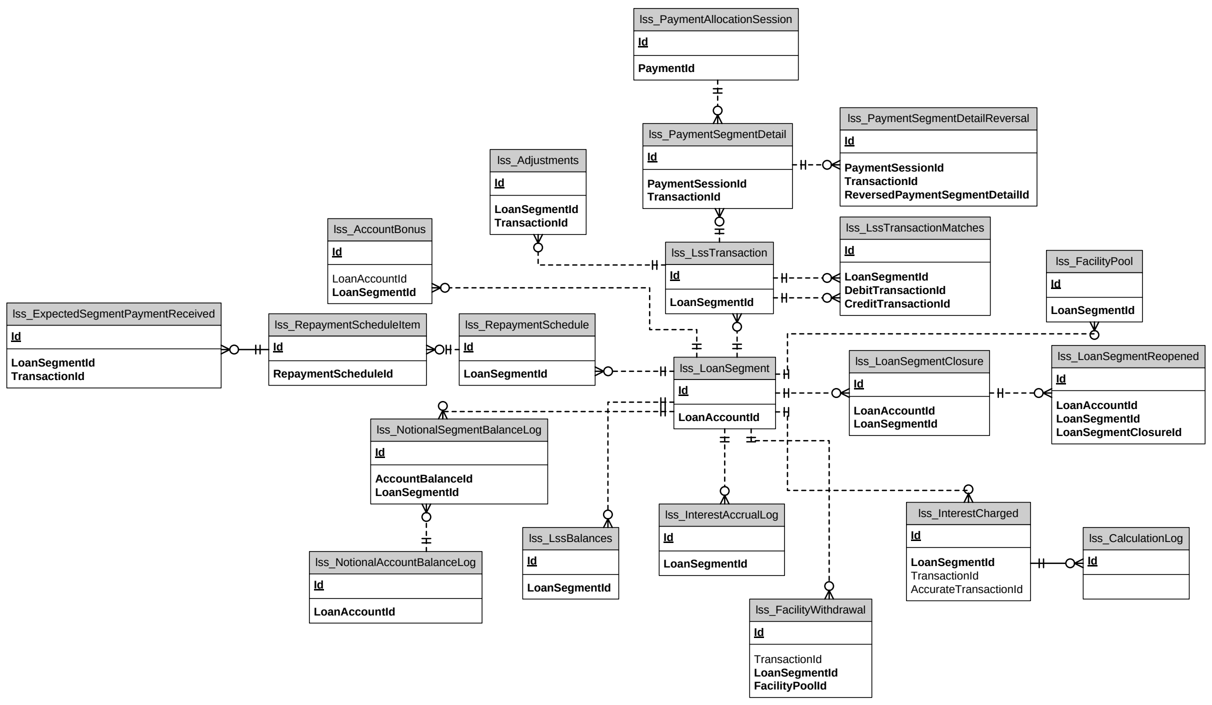
When Segments close, a configurable grace period (typically set to 100 days) is allowed before fact records stop being loaded

6.33.2. Place in Conceptual Model

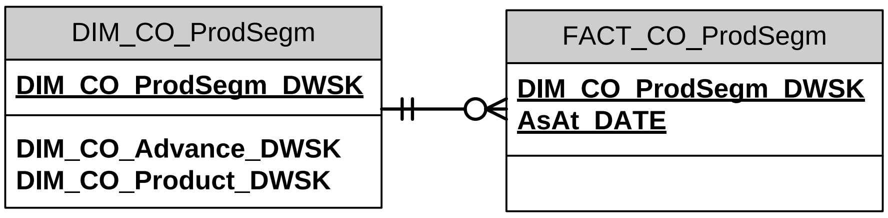
This table sources data from several different places within the Servicing schema. Specifically it retrieves:

- The majority of the Segment specific information from the Segment table
- Interest information from the Interest Accrual Log
- Several things are calculated from Transactions including
    - Waterfall Payments
    - Overpayments
    - Redemption Payments
    - Withdrawals
    - Deposits
    - Balance
    - Write Offs
    - Uncleared Amounts
- Facility, Retention and Drawdown information from the Facility Pool tables
- Several “positions” (e.g. Principal Charged and Satisfied, Fees Charged and Satisfied) from the “Balances” table
- Payments expected and Days Overdue from the repayments schedules
- Arrears from the Notional Segment Balance Log

- Bonuses from the Account Bonus table
- Reversion Rates from the Calculation Log



6.33.3. Star Schema



6.33.4. Linked Dimension Tables

- DIM_CO_ProdSegm

6.33.5. Query Hints

The main thing to always bear in mind when querying a Snapshot Fact table is to always filter on a specific date in order to retrieve only one result per entity. To join the results of this Fact table to others, base the joins on the SSK as in the query below. No filtering of the dimension is required as the table will only hold one result per entity as enforced by the primary key.

```
SELECT
       dps.DIM_CO_ProdSegm_SSK
       ,<Desired Fields>
FROM FACT_CO_ProdSegm fps
       JOIN DIM_CO_ProdSegm dps
              ON dps.DIM_CO_ProdSegm_DWSK = fps.DIM_CO_ProdSegm_DWSK
WHERE AsAt_DATE = @ReportDate
```

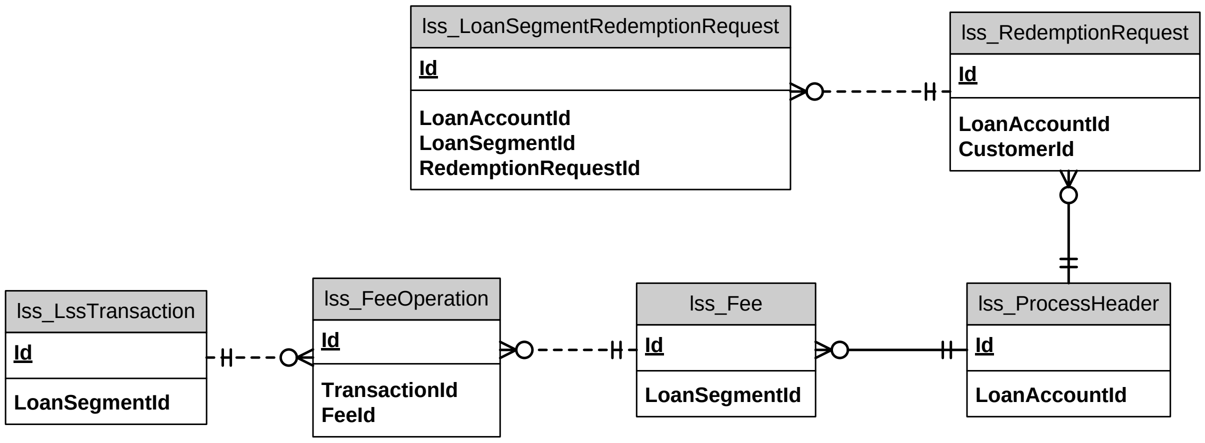
6.34. FACT_CO_REDEMPTIONAS

6.34.1. Purpose and How it Works

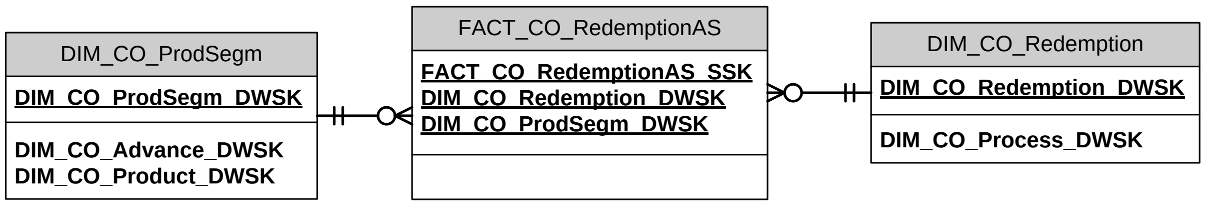
This table stores information on any Redemption requests that take place. The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get “duplicate” records in queries due to the multiple SCD versions present in the results.

6.34.2. Place in Conceptual Model

This table sources its data from the Redemption request tables and the Fee tables.



6.34.3. Star Schema



6.34.4. Linked Dimension Tables

- DIM_CO_ProdSegm
- DIM_CO_Redemption

6.34.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```sql
SELECT
        --,<Desired Fields>
FROM FACT_CO_RedemptionAS fr
        JOIN DIM_CO_Redemption dr
                ON dr.DIM_CO_Redemption_DWSK = fr.DIM_CO_Redemption_DWSK
                AND dr.SCDStatus = 'C'
        JOIN DIM_CO_ProdSegm dp
                ON dp.DIM_CO_ProdSegm_DWSK = fr.DIM_CO_ProdSegm_DWSK
                AND dp.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       --,<Desired Fields>
FROM FACT_CO_RedemptionAS fr
       JOIN DIM_CO_Redemption dr
              ON dr.DIM_CO_Redemption_DWSK = fr.DIM_CO_Redemption_DWSK
              AND dr.SCDStartDate <= @ReportDate
              AND dr.SCDEndDate > @ReportDate
       JOIN DIM_CO_ProdSegm dp
              ON dp.DIM_CO_ProdSegm_DWSK = fr.DIM_CO_ProdSegm_DWSK
              AND dp.SCDStartDate <= @ReportDate
              AND dp.SCDEndDate > @ReportDate
```

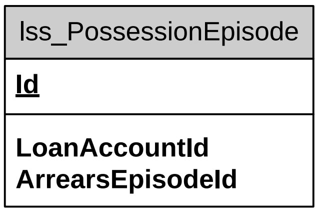
6.35. FACT_CO_REPOSSESSION

6.35.1. Purpose and How it Works

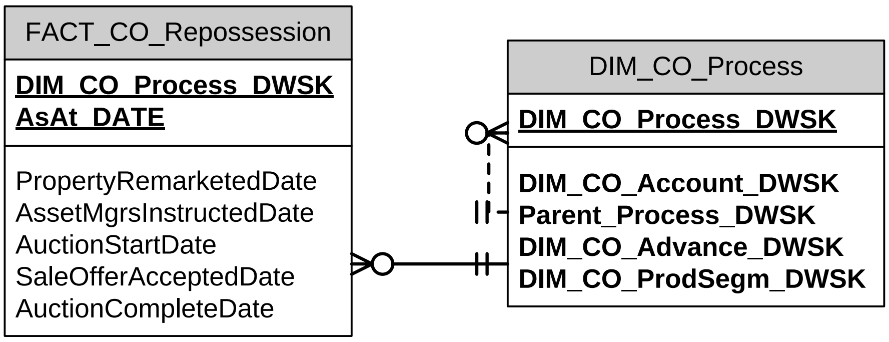
This table stores information on Reposession that take place. It works as a Snapshot Fact table i.e. new records are added on a daily basis and DWSKs are only stored once and not updated or multiplied.

6.35.2. Place in Conceptual Model

This table gets its data from the Possession Episode table in Servicing.



6.35.3. Star Schema



6.35.4. Linked Dimension Tables

- DIM_CO_Process

6.35.5. Query Hints

The main thing to always bear in mind when querying a Snapshot Fact table is to always filter on a specific date in order to retrieve only one result per entity. To join the results of this Fact table to others, base the joins on the SSK as in the query below. No filtering of the dimension is required as the table will only hold one result per entity as enforced by the primary key.

```
SELECT
       --,<Desired Fields>
FROM FACT_CO_Repossession fr
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = fr.DIM_CO_Process_DWSK
WHERE fr.AsAt_DATE = @ReportDate
```

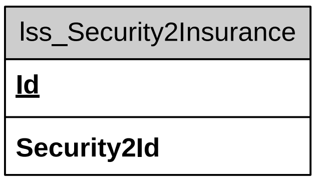
6.36. FACT_CO_SecurityInsuranceAS

6.36.1. Purpose and How it Works

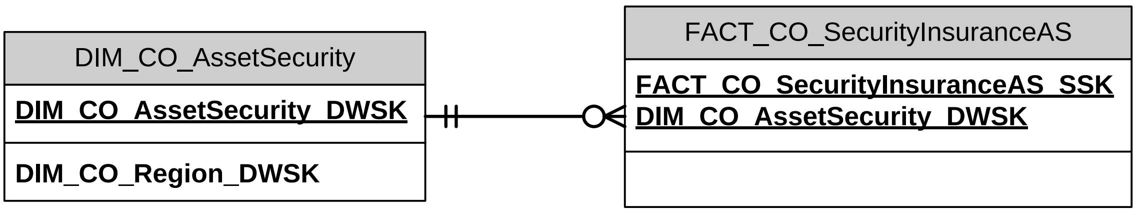
This table stores information on any insurance held against Securities (including MIGs).The table works as an adapted Accumulating Snapshot Fact table i.e. new records are added as new statuses are added and also when new DWSKs are generated. Other values are updated. This means that any previous values held in this table will be overwritten and no history kept. It is also possible to get "duplicate" records in queries due to the multiple SCD versions present in the results.

6.36.2. Place in Conceptual Model

This tables sources its data from the Security Insurance table in Servicing



6.36.3. Star Schema


DIM_CO_AssetSecurity
DIM_CO_AssetSecurity_DWSK
DIM_CO_Region_DWSK
FACT_CO_SecurityInsuranceAS
FACT_CO_SecurityInsuranceAS_SSK
DIM_CO_AssetSecurity_DWSK

6.36.4. Linked Dimension Tables

- DIM_CO_AssetSecurity

6.36.5. Query Hints

The main thing to bear in mind with this table is to take care to filter out duplicate entries. This can be done by either selecting the current version of everything:

```
SELECT
       da.DIM_CO_AssetSecurity_SSK
       --,<Desired Fields>
FROM FACT_CO_SecurityInsuranceAS fsi
       JOIN DIM_CO_AssetSecurity da
              ON da.DIM_CO_AssetSecurity_DWSK = fsi.DIM_CO_AssetSecurity_DWSK
              AND da.SCDStatus = 'C'
```

Or by reporting at a given moment in time:

```
SELECT
       da.DIM_CO_AssetSecurity_SSK
       --,<Desired Fields>
FROM FACT_CO_SecurityInsuranceAS fsi
       JOIN DIM_CO_AssetSecurity da
```

```
ON da.DIM_CO_AssetSecurity_DWSK = fsi.DIM_CO_AssetSecurity_DWSK
AND da.SCDStartDate <= @ReportDate
AND da.SCDEndDate > @ReportDate
```

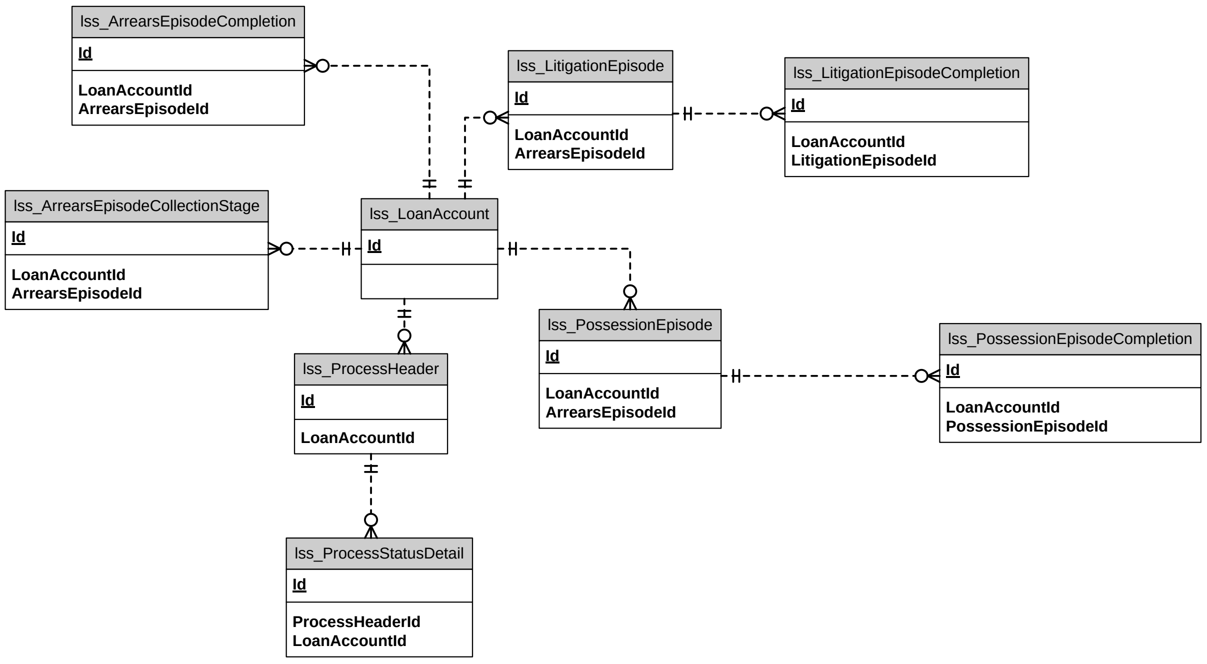
6.37. FACT_CO_StatusTrx

6.37.1. Purpose and How it Works

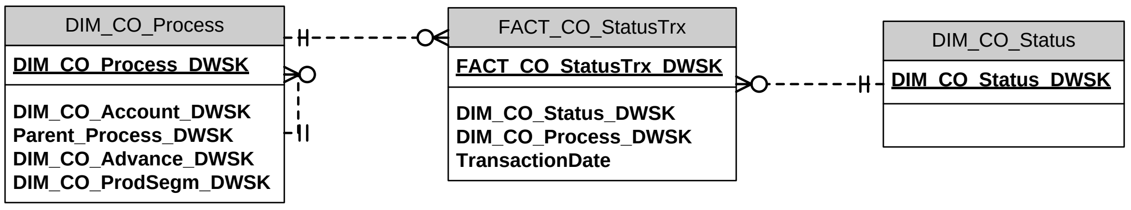
This table stores information on Status transactions. The table works as a Transactional Fact table i.e. new records are added as new transactions are added and DWSKs are only stored once.

6.37.2. Place in Conceptual Model

This table sources its data from the Process status tables and also from other tables which are translated as Processes within the Warehouse



6.37.3. Star Schema



6.37.4. Linked Dimension Tables

- DIM_CO_Process
- DIM_CO_Status

6.37.5. Query Hints

With Transactional fact tables the main thing to remember is the spread of DWSK versions that may be presents, so that when grouping data over a period of time the SSK should be used i.e.

```
SELECT
```

```
       dp.DIM_CO_Process_SSK
       --,<Desired Fields>
FROM FACT_CO_StatusTrx fs
       JOIN DIM_CO_Status ds
              ON ds.DIM_CO_Status_DWSK = fs.DIM_CO_Status_DWSK
       JOIN DIM_CO_Process dp
              ON dp.DIM_CO_Process_DWSK = fs.DIM_CO_Process_DWSK
```

6.38. FACT_CO_TASK

6.38.1. Purpose and How it Works

See Origination entry

6.39. FACT_CO_TASKSTATUSTRX

6.39.1. Purpose and How it Works

See the Origination entry.

6.40. IF_CO_CUSTOMGROUP_GLENTRYTYPE

6.40.1. Purpose and How it Works

This table allows GL transactions to be grouped into a “chart of accounts” structure that dictates how they should aggregate. This is then used by the Servicing cube to present transactions in this structure and aggregate them correctly. This table should not be used in Warehouse queries

6.40.2. Place in Conceptual Model

This table takes its data from an Excel spreadsheet.

6.41. IF_COACCOUNT_COASSETSECURITY

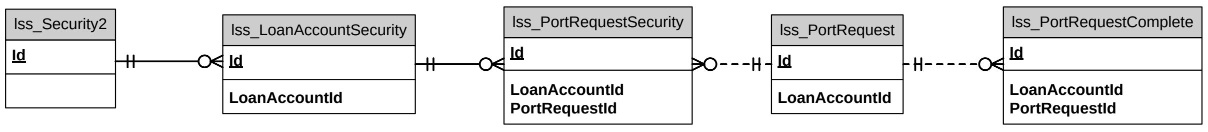
6.41.1. Purpose and How it Works

This table is used to track the relationship between Accounts and Securities. It acts like an adapted Accumulating Snapshot Fact table in that items are updated (the dates associated with the relationship) but new DWSK versions are included in the table as they arise without updating existing entries.

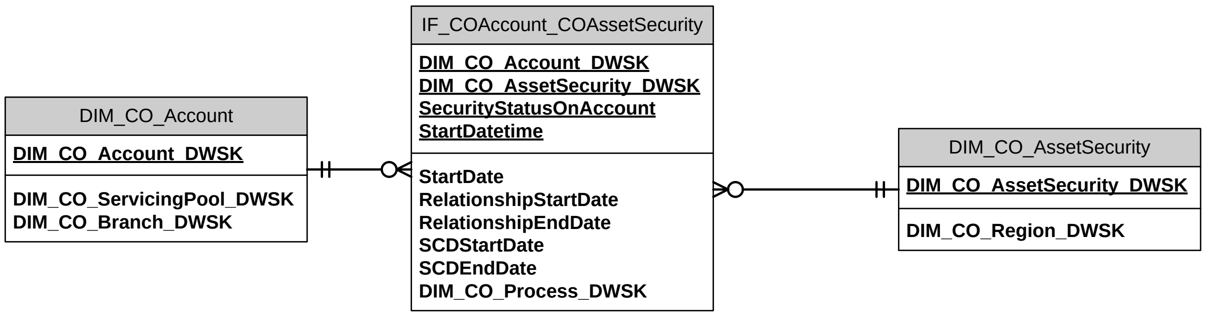
To handle this there are two sets of dates on the table. Start Date and End Date represent the period of the relationship between two SSKs i.e. the start and end of the relationship between the two entities. The SCDStartDate and SCDEndDate represent the length of time between two DWSKs i.e. the relationship between two different versions of the entities.

6.41.2. Place in Conceptual Model

This table takes its data from the Account/Security relationship table and the Port tables.



6.41.3. Star Schema



6.41.4. Linked Dimension Tables

- DIM_CO_Account
- DIM_CO_AssetSecurity

6.41.5. Query Hints

Usually the reason to use this table will be to link Security details to Accounts. To do this care must be taken to take into account the multiple DWSKs in place.

A query like the one below can be used for this purpose, which uses the current versions of the dimension records and links them to a single instance of each relationship, anchored on the Start Date of the relationship.

```
WITH cte_Account AS
(
        SELECT
                DIM_CO_Account_SSK
                ,AccountNo
                --,<Desired Fields>
        FROM DIM_CO_Account
        WHERE SCDStatus = 'C'
),
cte_Relationship AS
(
        SELECT
                da.DIM_CO_Account_SSK
                ,das.DIM_CO_AssetSecurity_SSK
                --,<Desired Fields>
        FROM IF_COAccount_COAssetSecurity ifas
                JOIN DIM_CO_Account da
                        ON da.DIM_CO_Account_DWSK = ifas.DIM_CO_Account_DWSK
                        AND da.SCDStartDate <= ifas.StartDate
                        AND da.SCDEndDate > ifas.StartDate
                JOIN DIM_CO_AssetSecurity das
                        ON das.DIM_CO_AssetSecurity_DWSK = ifas.DIM_CO_AssetSecurity_DWSK
                        AND das.SCDStartDate <= ifas.StartDate
                        AND das.SCDEndDate > ifas.StartDate
),
cte_AssetSecurity AS
(
        SELECT
                da.DIM_CO_AssetSecurity_SSK
                --,<Desired Fields>
        FROM DIM_CO_AssetSecurity da
        WHERE da.SCDStatus = 'C'
)
SELECT
```

```
       --,<Desired Fields>
FROM cte_Account da
JOIN cte_Relationship r
       ON da.DIM_CO_Account_SSK = r.DIM_CO_Account_SSK
JOIN cte_AssetSecurity das
       ON das.DIM_CO_AssetSecurity_SSK = r.DIM_CO_AssetSecurity_SSK
```

6.42. IF_COAccount_CoCust

6.42.1. Purpose and How it Works

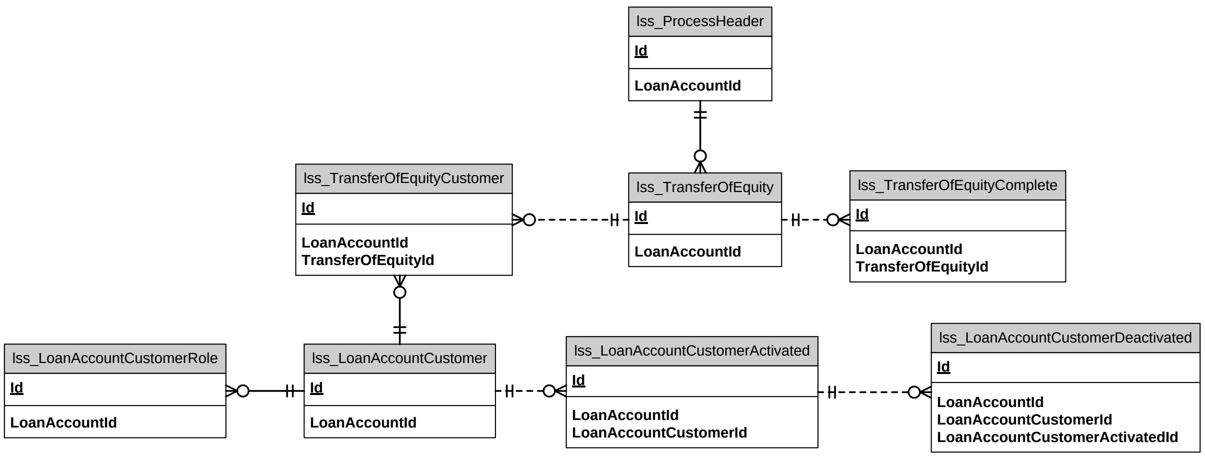
This table is used to track the relationship between Accounts and Customers. It acts like an adapted Accumulating Snapshot Fact table in that items are updated (the dates associated with the relationship) but new DWSK versions are included in the table as they arise without updating existing entries.

To handle this there are two sets of dates on the table. Start Date and End Date represent the period of the relationship between two SSKs i.e. the start and end of the relationship between the two entities. The SCDStartDate and SCDEndDate represent the length of time between two DWSKs i.e. the relationship between two different versions of the entities.

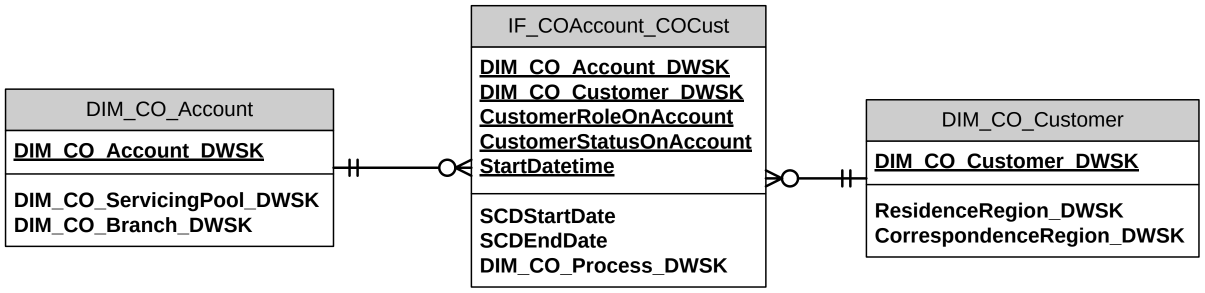
It is also important to note the “CustomerRoleOnAccount” field which differentiates what type of Customer is being considered. An Account Holder will always be denoted by a role of “Borrower”.

6.42.2. Place in Conceptual Model

This table sources its data from the Account/Customer relationship tables and the Transfer of Equity tables.



6.42.3. Star Schema



6.42.4. Linked Dimension Tables

- DIM_CO_Account
- DIM_CO_Customer

6.42.5. Query Hints

Usually the reason to use this table will be to link Customer details to Accounts. To do this care must be taken to take into account the multiple DWSKs in place.

A query like the one below can be used for this purpose, which uses the current versions of the dimension records and links them to a single instance of each relationship, anchored on the Start Date of the relationship.

```
WITH cte_Account AS
(
       SELECT
              DIM_CO_Account_SSK
              ,AccountNo
              --,<Desired Fields>
       FROM DIM_CO_Account
       WHERE SCDStatus = 'C'
),
cte_Relationship AS
(
       SELECT
              da.DIM_CO_Account_SSK
              ,dc.DIM_CO_Customer_SSK
              --,<Desired Fields>
       FROM IF_COAccount_COCust ifac
              JOIN DIM_CO_Account da
                     ON da.DIM_CO_Account_DWSK = ifac.DIM_CO_Account_DWSK
                     AND da.SCDStartDate <= ifac.StartDate
                     AND da.SCDEndDate > ifac.StartDate
              JOIN DIM_CO_Customer dc
                     ON dc.DIM_CO_Customer_DWSK = ifac.DIM_CO_Customer_DWSK
                     AND dc.SCDStartDate <= ifac.StartDate
                     AND dc.SCDEndDate > ifac.StartDate
),
cte_Customer AS
(
       SELECT
              dc.DIM_CO_Customer_SSK
              --,<Desired Fields>
       FROM DIM_CO_Customer dc
       WHERE dc.SCDStatus = 'C'
)
SELECT
       --,<Desired Fields>
FROM cte_Account da
JOIN cte_Relationship r
       ON da.DIM_CO_Account_SSK = r.DIM_CO_Account_SSK
JOIN cte_Customer dc
       ON dc.DIM_CO_Customer_SSK = r.DIM_CO_Customer_SSK
```

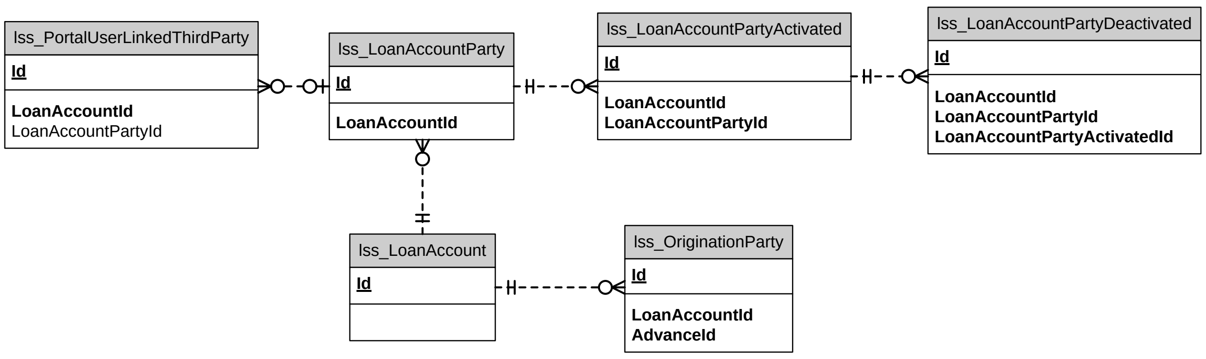
6.43. IF_COPROCESS_COINTERMEDIARY

6.43.1. Purpose and How it Works

See Origination entry

6.43.2. Place in Conceptual Model

This table retrieves its data from two locations within Servicing. One from the Portal User Linked Third Parties of type “Servicing Agents” and also from Origination Party information for parties of type “Introducer”, “Packager” or “SatelliteNetwork”.



6.43.3. Star Schema

See Origination entry

6.43.4. Linked Dimension Tables

See Origination entry

6.43.5. Query Hints

See Origination entry

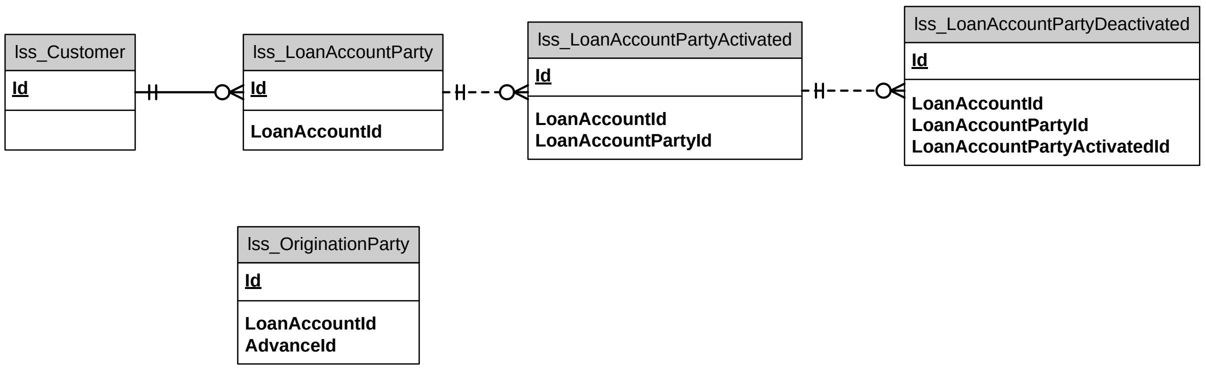
6.44. IF_COPROCESS_COTHIRDPARTY

6.44.1. Purpose and How it Works

See Origination entry

6.44.2. Place in Conceptual Model

This table sources its data from the Loan Account Party and Origination Party tables. It filters out those parties captured in the IF_COProcess_COIntermediary table.



6.44.3. Star Schema

See Origination entry

6.44.4. Linked Dimension Tables

See Origination entry

6.44.5. Query Hints

See Origination entry

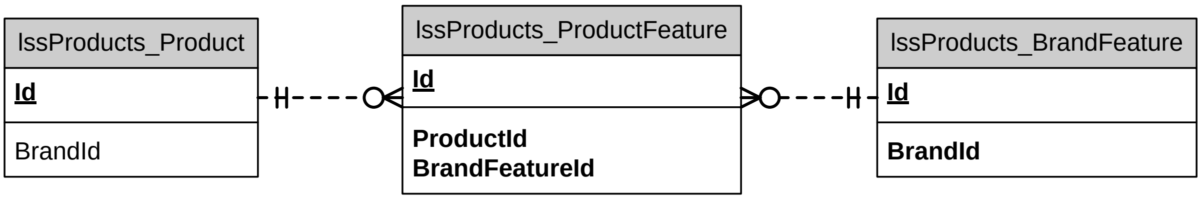
6.45. IF_COPRODUCT_COBFEATURE

6.45.1. Purpose and How it Works

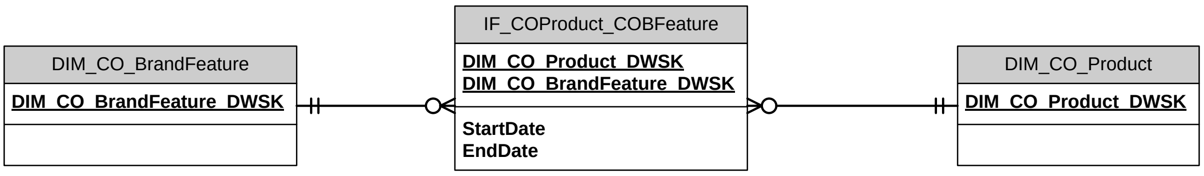
This table records the relationship between Products and Features. It loads new values every time there is a new combination of DWSKs which means their could be multiple versions of the same relationship in place. An IsCurrent flag is used to store the latest set of relationships.

6.45.2. Place in Conceptual Model

This table takes its data from the Product Feature relationship tables in the Servicing Products database.



6.45.3. Star Schema



6.45.4. Linked Dimension Tables

- DIM_CO_Product
- DIM_CO_BrandFeature

6.45.5. Query Hints

This information will not usually be queried in isolation but in combination with data stored in tables like FACT_CO_BrandFeatureValueNVP. A query like the one below can be used to identify the current state of the relationship and to link the data together:

```
WITH cte_Product AS
(
	SELECT
		DIM_CO_Product_SSK
		--,<Desired Fields>
	FROM DIM_CO_Product
	WHERE SCDStatus = 'C'
),
cte_Relationship AS
(
	SELECT
		dp.DIM_CO_Product_SSK
		,db.DIM_CO_BrandFeature_SSK
		--,<Desired Fields>
	FROM IF_COProduct_COBFeature ifpb
		JOIN DIM_CO_Product dp
			ON dp.DIM_CO_Product_DWSK = ifpb.DIM_CO_Product_DWSK
		JOIN DIM_CO_BrandFeature db
			ON db.DIM_CO_BrandFeature_DWSK = ifpb.DIM_CO_BrandFeature_DWSK
WHERE ifpb.IsCurrent = 1
),
cte_BrandFeature AS
(
        SELECT
                DIM_CO_BrandFeature_SSK
                --,<Desired Fields>
        FROM DIM_CO_BrandFeature
        WHERE SCDStatus = 'C'
)
SELECT
        --,<Desired Fields>
FROM cte_Product dp
JOIN cte_Relationship r
        ON dp.DIM_CO_Product_SSK = r.DIM_CO_Product_SSK
JOIN cte_BrandFeature db
        ON db.DIM_CO_BrandFeature_SSK = r.DIM_CO_BrandFeature_SSK
```

6.46. DIM_CO_Account

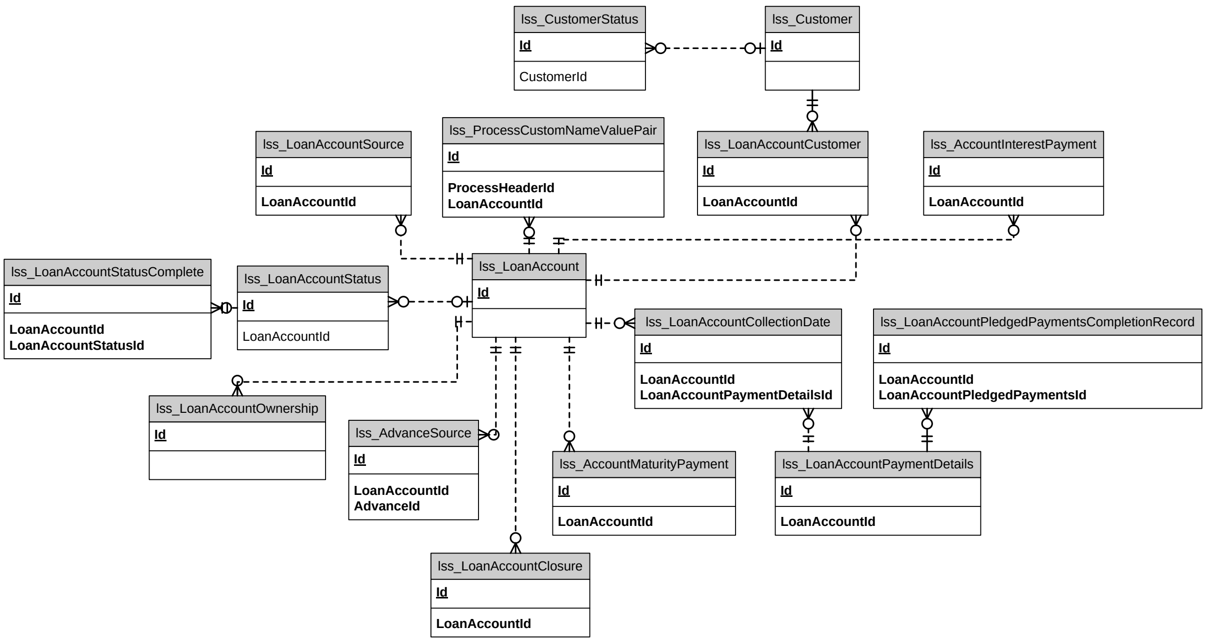
6.46.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.46.2. Place in Conceptual Model

This table sources most of its data from the Account level information in Servicing. In addition it brings in other areas of information to supplement this, namely:

- Account Status information
- Collection dates to identify rollovers
- The Customer tables to work out if the Account has UK residents
- The Account Maturity Payment table to work out the maturity date of the Account
- The Account Interest Payment table to work out the payment type of Interest on the Account
- The Advance Source table to work out if an Account is fully, partially or not regulated.
- A combination of Account and Customer information to work out the SCV status on the Account.



6.46.3. Associated Hierarchies

- DIM_CO_ServicingPoolCompany
    - DIM_CO_ServicingPool
        - DIM_CO_Account
            - DIM_CO_Advance
                - DIM_CO_ProdSegm

6.46.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Account
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Account
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.47. DIM_CO_ACTION

6.47.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.47.2. Place in Conceptual Model

This data is taken from the Account Events table in Servicing.

Iss_AccountEvents
Id
LoanAccountId

6.47.3. Associated Hierarchies

- DIM_CO_Account
    - DIM_CO_Episode
        - DIM_CO_Action

6.47.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
```

```
FROM DIM_CO_Action
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Action
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.48. DIM_CO_ACTIONTYPE

6.48.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.48.2. Place in Conceptual Model

This data is taken from the Account Events table in Servicing.

Iss_AccountEvents
Id
LoanAccountId

6.48.3. Associated Hierarchies

- DIM_CO_ActionType
    - DIM_CO_Action

6.48.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ActionType
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ActionType
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

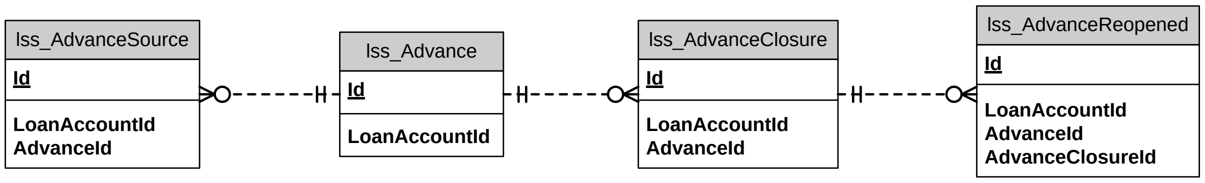
6.49. DIM_CO_ADVANCE

6.49.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.49.2. Place in Conceptual Model

This table sources its data from the Advance level information in Servicing.



6.49.3. Associated Hierarchies

- DIM_CO_ServicingPoolCompany
    - DIM_CO_ServicingPool
        - DIM_CO_Account
            - DIM_CO_Advance
                - DIM_CO_ProdSegm

6.49.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Advance
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Advance
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

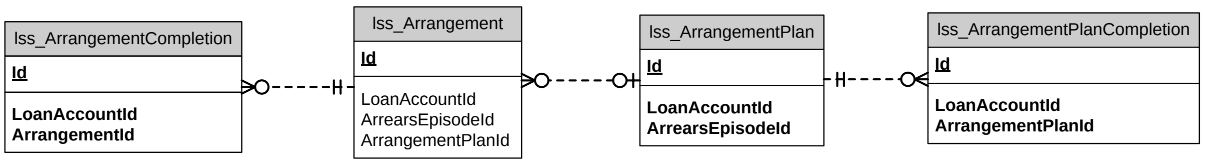
6.50. DIM_CO_ARRANGEMENT

6.50.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.50.2. Place in Conceptual Model

This table sources its data from the Arrangement tables in Servicing.



6.50.3. Associated Hierarchies

- DIM_CO_Process
    - DIM_CO_ArrearsEpisode
        - DIM_CO_Arrangement

6.50.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Arrangement
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Arrangement
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

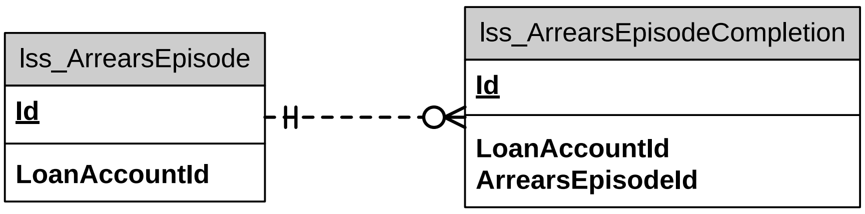
6.51. DIM_CO_ArrearsEpisode

6.51.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.51.2. Place in Conceptual Model

This table sources its data from the Arrears Episode tables in Servicing



6.51.3. Associated Hierarchies

- DIM_CO_Process
    - DIM_CO_ArrearsEpisode

6.51.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
```

```
       <Desired Fields>
FROM DIM_CO_ArrearsEpisode
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ArrearsEpisode
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

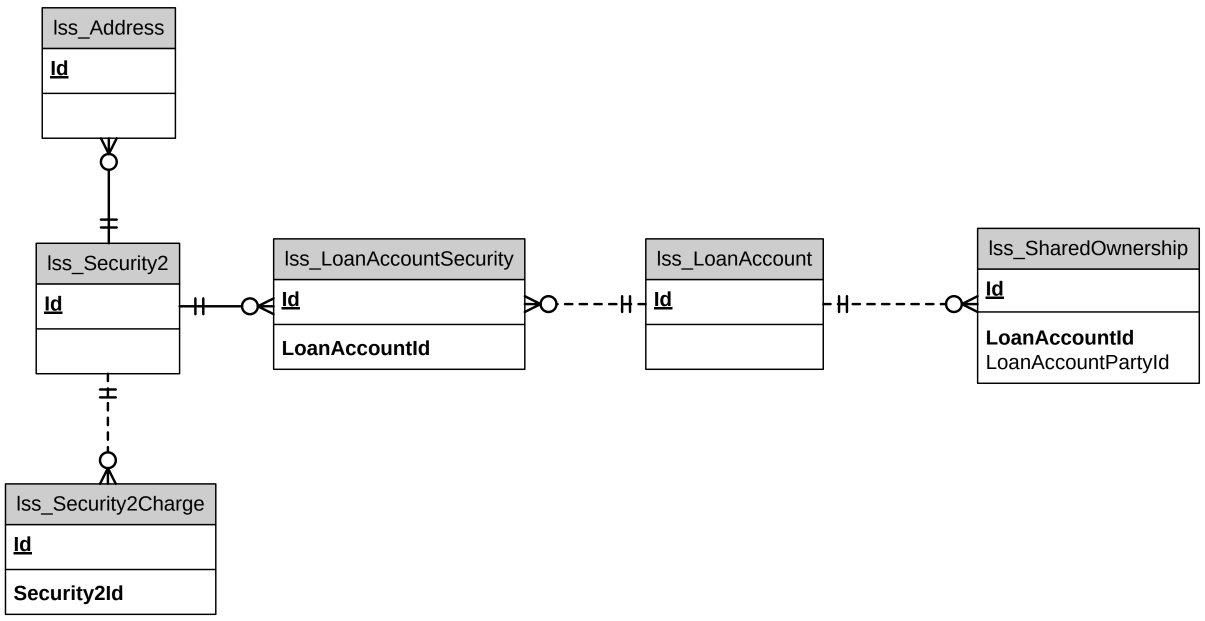
6.52. DIM_CO_AssetSecurity

6.52.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.52.2. Place in Conceptual Model

This table sources its data from the Security table in Servicing and enriches it with Shared Ownership, Address and Security Charge data.



6.52.3. Associated Hierarchies

- DIM_CO_AssetSecurity
    - DIM_CO_Valuation
        - DIM_CO_ValComparable

6.52.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
```

```
       <Desired Fields>
FROM DIM_CO_AssetSecurity
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_AssetSecurity
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

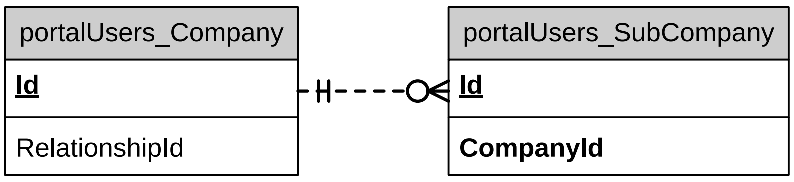
6.53. DIM_CO_BRANCH

6.53.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.53.2. Place in Conceptual Model

This table sources its data from the Portal Users company tables.



6.53.3. Associated Hierarchies

- DIM_CO_BranchParentCompany
    - DIM_CO_Branch
        - DIM_CO_Account

6.53.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Branch
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Branch
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

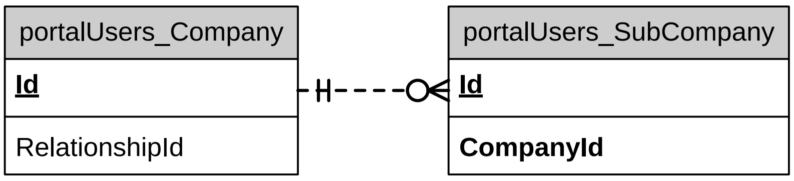
6.54. DIM_CO_BranchParentCompany

6.54.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.54.2. Place in Conceptual Model

This table sources its data from the Portal Users company tables.



6.54.3. Associated Hierarchies

- DIM_CO_BranchParentCompany
  - DIM_CO_Branch
    - DIM_CO_Account

6.54.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BranchParentCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BranchParentCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.55. DIM_CO_BrandFeature

6.55.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.55.2. Place in Conceptual Model

This table sources its data from the Feature information in Servicing products.

IssProducts_BrandFeature
Id
BrandId

6.55.3. Associated Hierarchies

- DIM_CO_BrandFeature
    - DIM_CO_BrandFeatureCheck

6.55.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandFeature
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandFeature
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.56. DIM_CO_BrandFeatureCheck

6.56.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.56.2. Place in Conceptual Model

This table sources its data from the Feature Check information in Servicing Products

IssProducts_BrandFeatureCheck
Id
BrandFeatureID

6.56.3. Associated Hierarchies

- DIM_CO_BrandFeature
    - DIM_CO_BrandFeatureCheck

6.56.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandFeatureCheck
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandFeatureCheck
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.57. DIM_CO_BrandRateTier

6.57.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.57.2. Place in Conceptual Model

This table sources its data from the Rate Tier tables in the Servicing Products database.

IssProducts_BrandRateTierHeader
Id
BrandId

6.57.3. Associated Hierarchies

None

6.57.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandRateTier
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_BrandRateTier
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

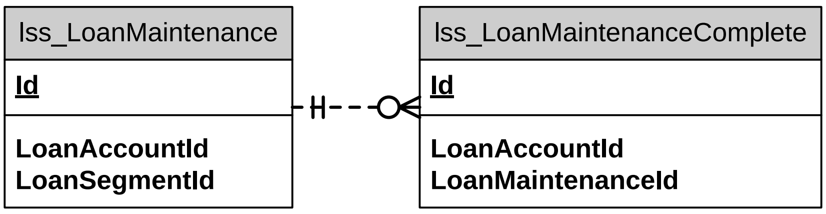
6.58. DIM_CO_ContractChange

6.58.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.58.2. Place in Conceptual Model

This table takes its data from the loan maintenance tables in Servicing



6.58.3. Associated Hierarchies

- DIM_CO_Process
  - DIM_CO_ContractChange

- DIM_CO_ProdSegm
  - DIM_CO_ContractChange

6.58.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ContractChange
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ContractChange
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

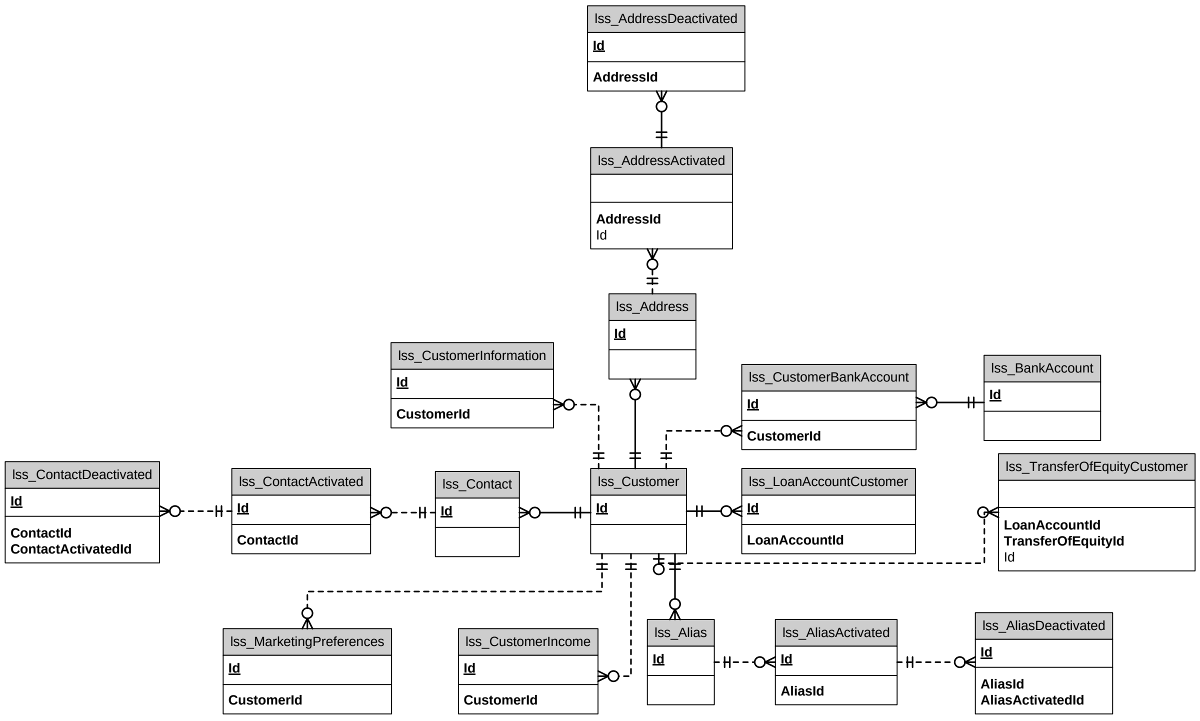
6.59. DIM_CO_Customer

6.59.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.59.2. Place in Conceptual Model

This table sources its data from the Customer information in Servicing.



6.59.3. Associated Hierarchies

None

6.59.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Customer
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Customer
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.60. DIM_CO_CustomGroup

6.60.1. Purpose and How it Works

This table represents a custom group of GL Transactions that can be used to create a spoke "chart of accounts" within the Servicing cube.

6.60.2. Place in Conceptual Model

This data is populated from a spreadsheet.

6.60.3. Associated Hierarchies

None

6.60.4. Query Hints

This table should not be used in Warehouse queries.

6.61. DIM_CO_Drawdown

6.61.1. Purpose and How it Works

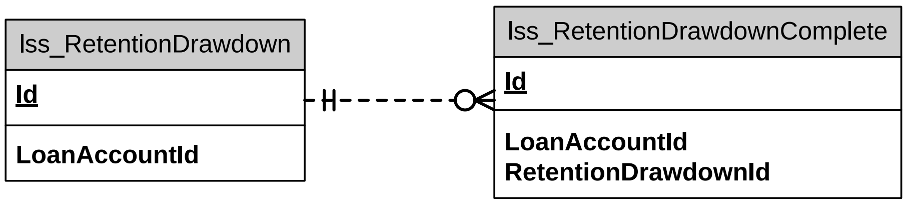
This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

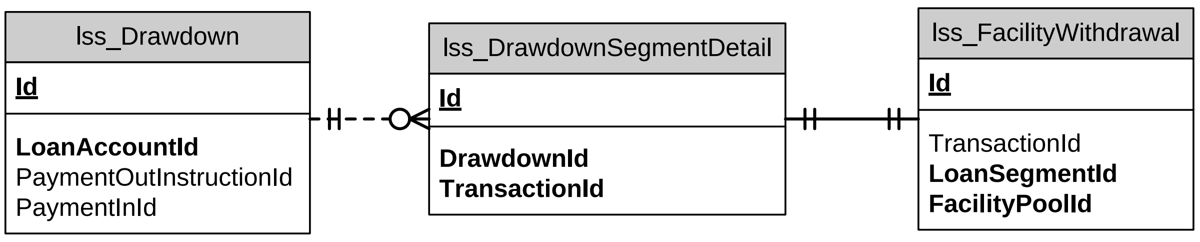
6.61.2. Place in Conceptual Model

This table captures two forms of Drawdown information:

- Retention Drawdowns
- Facility Drawdowns

Both are captured from the appropriate set of tables





6.61.3. Associated Hierarchies

- DIM_CO_Process
  - DIM_CO_Drawdown

6.61.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Drawdown
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Drawdown
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.62. DIM_CO_Episode

6.62.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.62.2. Place in Conceptual Model

This data is taken from the Account Events table in Servicing.

Iss_AccountEvents
Id
LoanAccountId

6.62.3. Associated Hierarchies

- DIM_CO_Account
    - DIM_CO_Episode
        - DIM_CO_Action

6.62.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Episode
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Episode
WHERE SCDStartDate <= @ReportDate
```

```
AND SCDEndDate > @ReportDate
```

6.63. DIM_CO_EpisodeType

6.63.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.63.2. Place in Conceptual Model

This data is taken from the Account Events table in Servicing.

lss_AccountEvents
<u>Id</u>
LoanAccountId

6.63.3. Associated Hierarchies

- DIM_CO_EpisodeType
    - DIM_CO_Episode

6.63.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_EpisodeType
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_EpisodeType
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

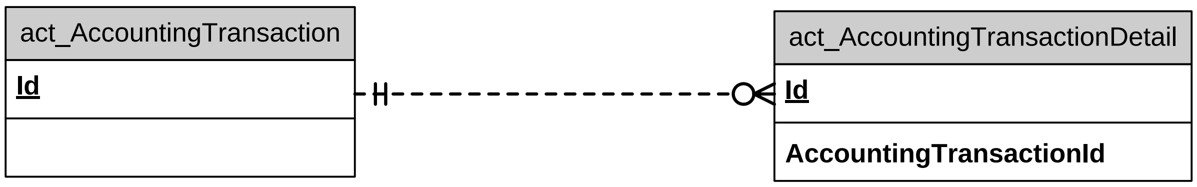
6.64. DIM_CO_GLEntryType

6.64.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.64.2. Place in Conceptual Model

This table takes its data from the GL Transactions in the LssAccounting database.



6.64.3. Associated Hierarchies

None

6.64.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_GLEntryType
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_GLEntryType
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

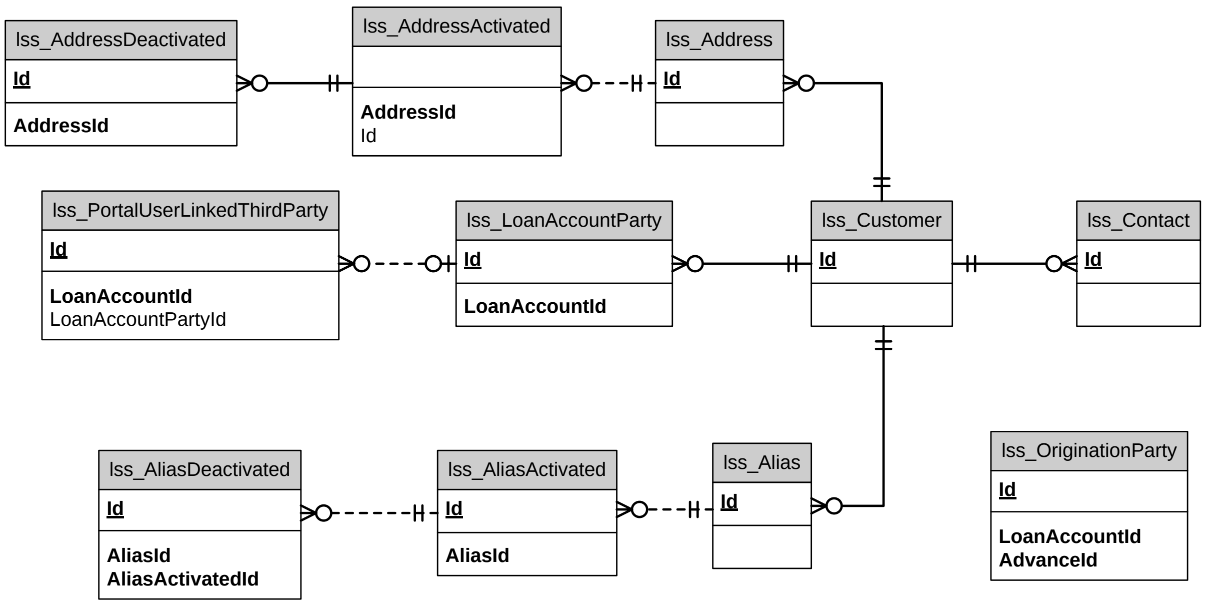
6.65. DIM_CO_INTERMEDIARY

6.65.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.65.2. Place in Conceptual Model

This table retrieves its data from two locations within Servicing. One from the Portal User Linked Third Parties of type “Servicing Agents” and also from Origination Party information for parties of type “Introducer”, “Packager” or “SatelliteNetwork”.



6.65.3. Associated Hierarchies

- DIM_CO_IntermediaryCompany
    - DIM_CO_Intermediary

6.65.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Intermediary
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Intermediary
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.66. DIM_CO_IntermediaryCompany

6.66.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.66.2. Place in Conceptual Model

This table sources its data from two sources. First it takes all of the known information from the company records in the Portal Users database.

portalUsers_Company
Id
RelationshipId

It also takes a feed of information from the Origination Third Parties table in Servicing.

lss_OriginationParty
Id
LoanAccountId AdvanceId

6.66.3. Associated Hierarchies

- DIM_CO_IntermediaryCompany
  - DIM_CO_Intermediary

6.66.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.67. DIM_CO_IntermediaryCompanyLocation

6.67.1. Purpose and How it Works

See Origination entry.

6.68. DIM_CO_Investor

6.68.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.68.2. Place in Conceptual Model

This table takes its data from the Asset Owner Investor Splits table in Portal Users.

portalUsers_AssetOwnerInvestorSplits
Id

6.68.3. Associated Hierarchies

None

6.68.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Investor
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Investor
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.69. DIM_CO_PAYMENTSCHEDULE

6.69.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.69.2. Place in Conceptual Model

See the FACT_CO_PaymentSchedule entry

6.69.3. Associated Hierarchies

- DIM_CO_PaymentSchedule
    - DIM_CO_PaymentScheduleItem

6.69.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_PaymentSchedule
```

```
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_PaymentSchedule
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.70. DIM_CO_PaymentScheduleItem

6.70.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.70.2. Place in Conceptual Model

See the FACT_CO_PaymentSchedule entry

6.70.3. Associated Hierarchies

- DIM_CO_PaymentSchedule
    - DIM_CO_PaymentScheduleItem

6.70.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

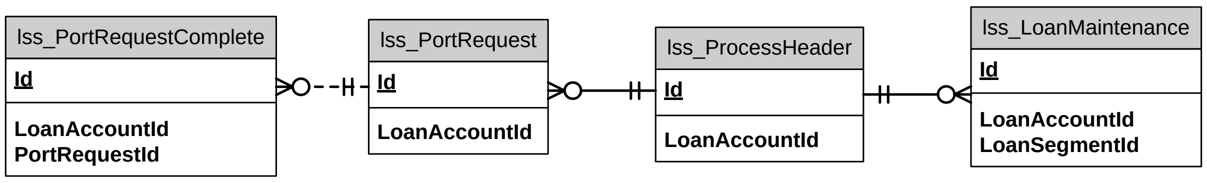
6.71. DIM_CO_Port

6.71.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.71.2. Place in Conceptual Model

This table takes its data from the Port request tables and add data on the New Guaranteed Facility from the Loan Maintenance record.



6.71.3. Associated Hierarchies

- DIM_CO_Process
    - DIM_CO_Port

6.71.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_IntermediaryCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.72. DIM_CO_Process

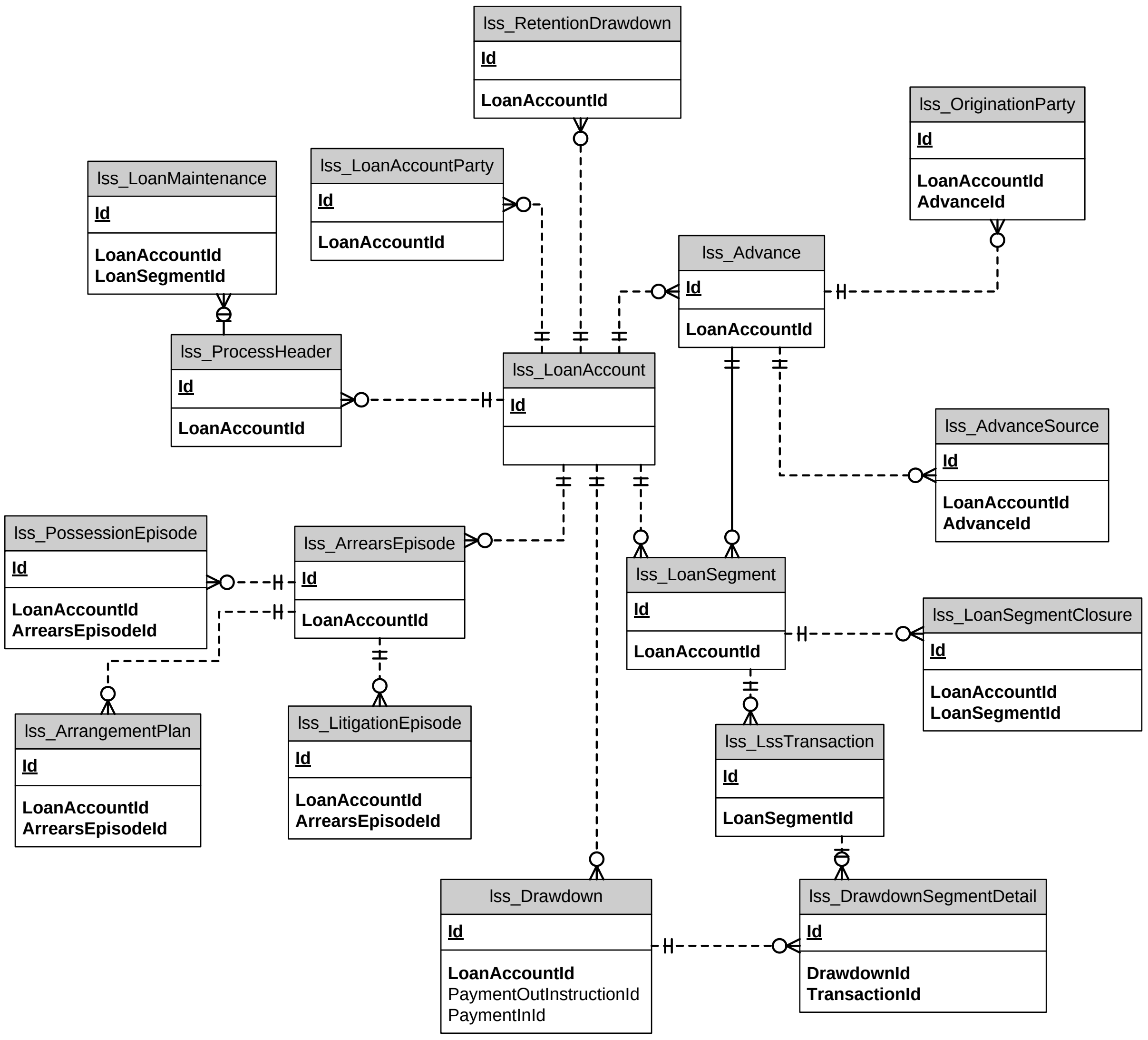
6.72.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.72.2. Place in Conceptual Model

This table sources its data from the Process table in Servicing and combines it with other tables where the Warehouse “creates” Processes for:

- Arrears, Litigation, Possession, Arrangements
- Third Parties
- Intermediaries
- Drawdowns



6.72.3. Associated Hierarchies

Multiple

6.72.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Process
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Process
WHERE SCDStartDate <= @ReportDate
AND SCDEndDate > @ReportDate
```

6.73. DIM_CO_ProdSegm

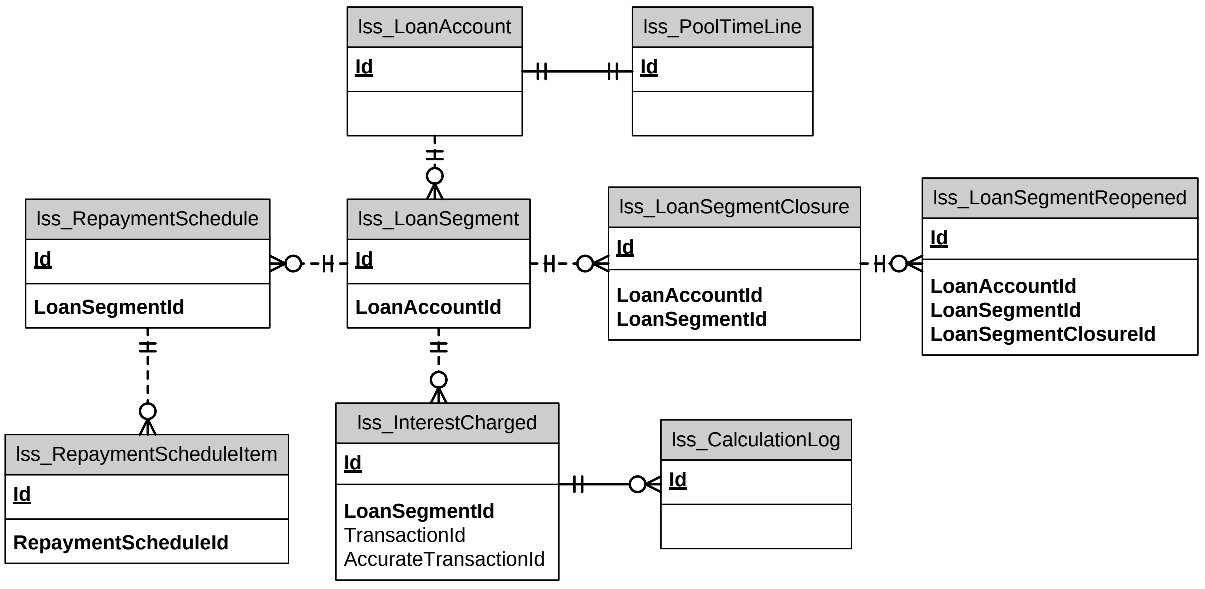
6.73.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.73.2. Place in Conceptual Model

This table takes its data from the Segment table in Servicing and enriches it with:

- Reversion Rate Change Dates from the Calculation Log table
- Whether the Segment is fixed or not based on the Rate Schedule



6.73.3. Associated Hierarchies

- DIM_CO_ServicingPoolCompany
  - DIM_CO_ServicingPool
    - DIM_CO_Account
      - DIM_CO_Advance
        - DIM_CO_ProdSegm
- DIM_CO_Product
  - DIM_CO_ProdSegm

6.73.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ProdSegm
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ProdSegm
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

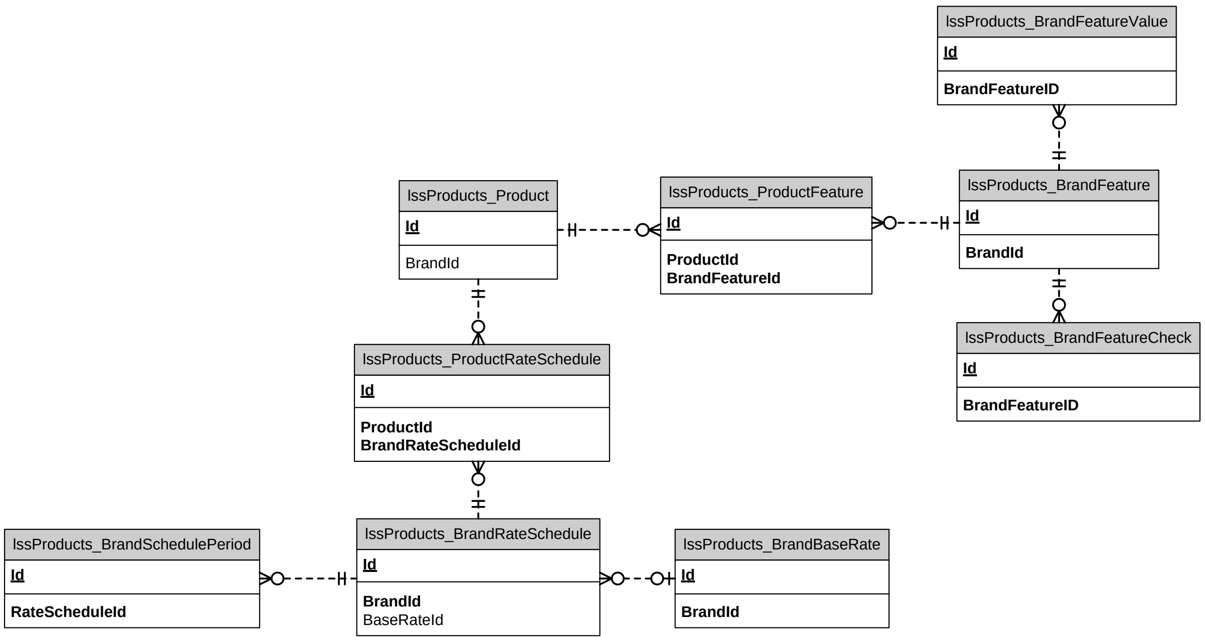
6.74. DIM_CO_Product

6.74.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.74.2. Place in Conceptual Model

This table sources its data from the Servicing Products database


IssProducts_BrandFeatureValue
Id
BrandFeatureID
IssProducts_Product
Id
BrandId
IssProducts_ProductFeature
Id
ProductId
BrandFeatureId
IssProducts_BrandFeature
Id
BrandId
IssProducts_BrandFeatureCheck
Id
BrandFeatureID
IssProducts_ProductRateSchedule
Id
ProductId
BrandRateScheduleId
IssProducts_BrandSchedulePeriod
Id
RateScheduleId
IssProducts_BrandRateSchedule
Id
BrandId
BaseRateId
IssProducts_BrandBaseRate
Id
BrandId

6.74.3. Associated Hierarchies

- DIM_CO_Product
    - DIM_CO_ProdSegm

6.74.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Product
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Product
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

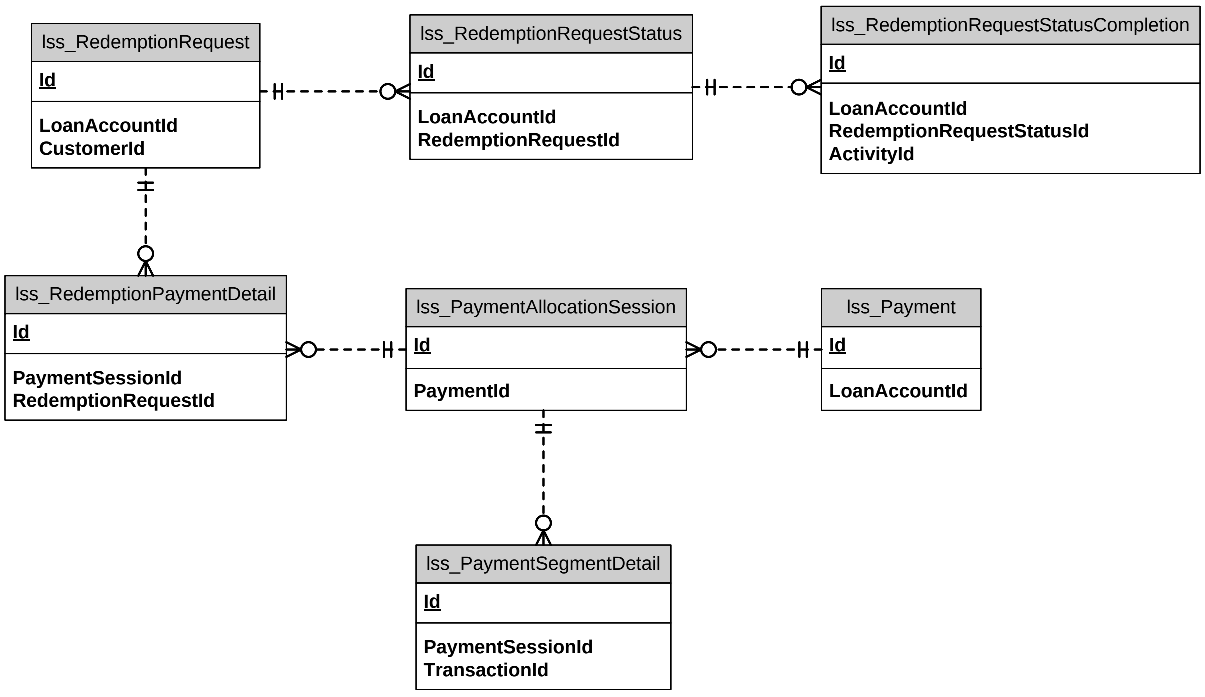
6.75. DIM_CO_Redemption

6.75.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.75.2. Place in Conceptual Model

This table takes its data from the Redemption and associated Payment tables in Servicing.



6.75.3. Associated Hierarchies

- DIM_CO_Process
  - DIM_CO_Redemption

6.75.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Redemption
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Redemption
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.76. DIM_CO_Role

6.76.1. Purpose and How it Works

See Origination entry.

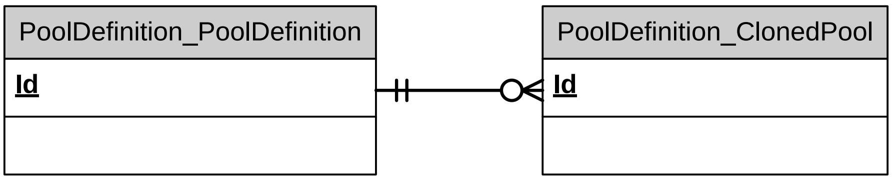
6.77. DIM_CO_ServicingPool

6.77.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.77.2. Place in Conceptual Model

This table sources its data from the Pool tables in the Pool Definition database



6.77.3. Associated Hierarchies

- DIM_CO_ServicingPoolCompany
    - DIM_CO_ServicingPool
        - DIM_CO_Account
            - DIM_CO_Advance
                - DIM_CO_ProdSegm

6.77.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ServicingPool
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ServicingPool
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

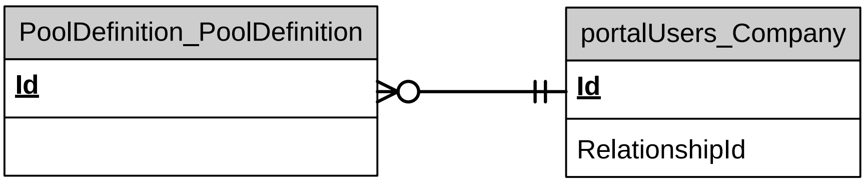
6.78. DIM_CO_ServicingPoolCompany

6.78.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.78.2. Place in Conceptual Model

This table sources its data from a combination of Portal Users and Pool Definition data



6.78.3. Associated Hierarchies

- DIM_CO_ServicingPoolCompany
  - DIM_CO_ServicingPool
    - DIM_CO_Account
      - DIM_CO_Advance
        - DIM_CO_ProdSegm

6.78.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ServicingPoolCompany
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ServicingPoolCompany
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.79. DIM_CO_StaffMember

6.79.1. Purpose and How it Works

See Origination entry.

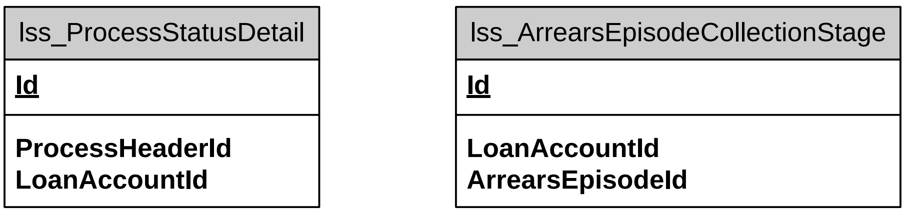
6.80. DIM_CO_STATUS

6.80.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.80.2. Place in Conceptual Model

This table sources some of its data from the Process and Arrears tables in Servicing



The rest is a set of statuses generated by the Warehouse itself to describe the Processes it creates.

6.80.3. Associated Hierarchies

None

6.80.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Status
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_Status
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

6.81. DIM_CO_TASK

6.81.1. Purpose and How it Works

See Origination entry

6.82. DIM_CO_TEAM

6.82.1. Purpose and How it Works

See Origination entry.

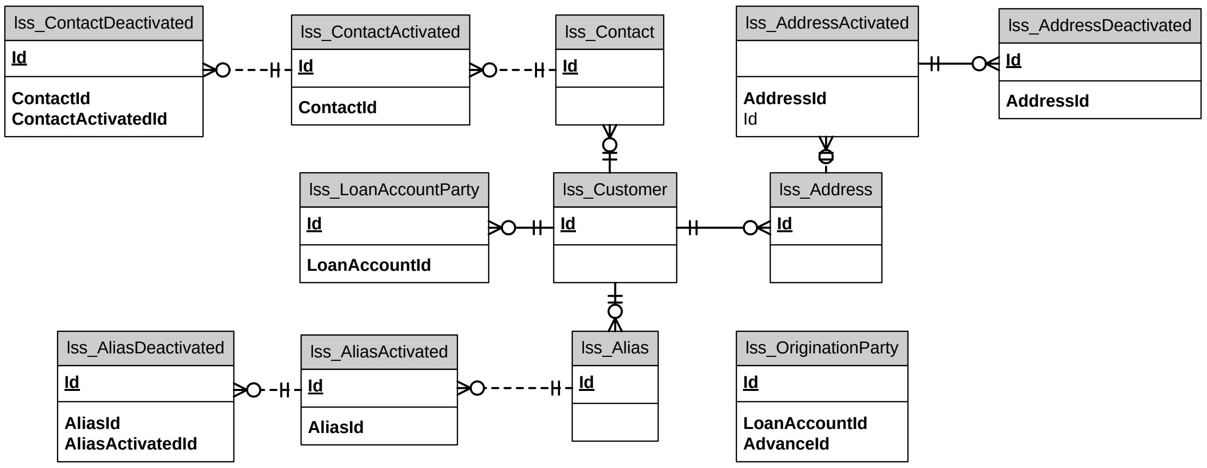
6.83. DIM_CO_THIRDPARTY

6.83.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.83.2. Place in Conceptual Model

This table sources its data from the Party and Origination Party information in Servicing.



6.83.3. Associated Hierarchies

None

6.83.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ThirdParty
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ThirdParty
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

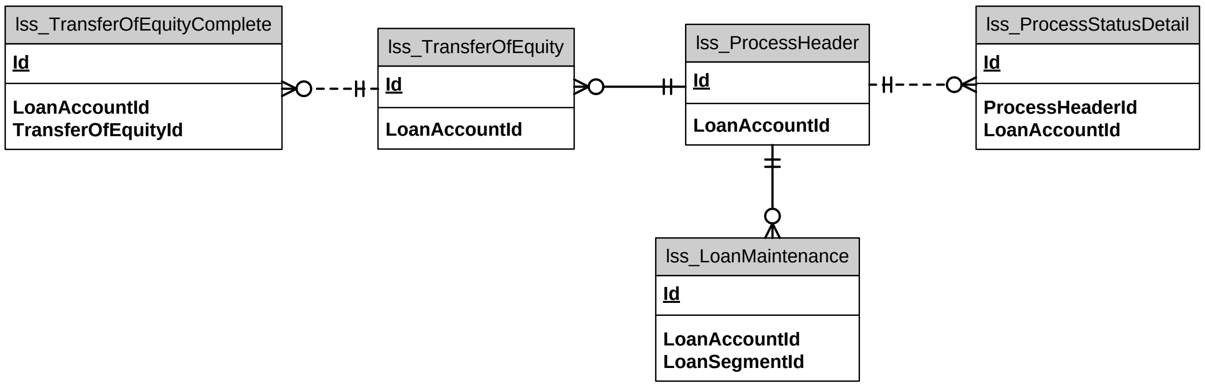
6.84. DIM_CO_TRANSFEROFEQUITY

6.84.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.84.2. Place in Conceptual Model

This table sources its data from the Transfer of Equity tables in Servicing and the associated Process tables.



6.84.3. Associated Hierarchies

- DIM_CO_Process
    - DIM_CO_TransferOfEquity

6.84.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_TransferOfEquity
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_TransferOfEquity
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

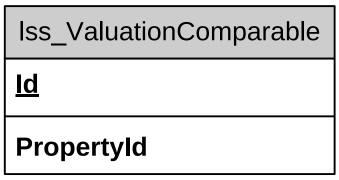
6.85. DIM_CO_VALCOMPARABLE

6.85.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.85.2. Place in Conceptual Model

This table sources its data from the Valuation Comparable table in Servicing.



6.85.3. Associated Hierarchies

- DIM_CO_AssetSecurity
    - DIM_CO_Valuation
        - DIM_CO_ValComparable

6.85.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ValComparable
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
       <Desired Fields>
FROM DIM_CO_ValComparable
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```

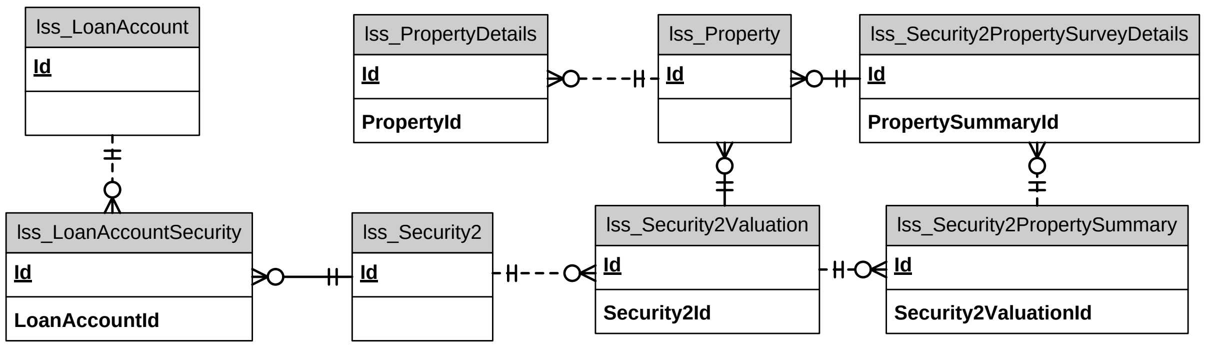
6.86. DIM_CO_VALUATION

6.86.1. Purpose and How it Works

This table stores the contextual information regarding the entities. It behaves as a Type 2 SCD table (i.e. every change is tracked).

6.86.2. Place in Conceptual Model

This table sources its data from the Security and associated Valuation tables in Servicing



6.86.3. Associated Hierarchies

- DIM_CO_AssetSecurity
    - DIM_CO_Valuation
        - DIM_CO_ValComparable

6.86.4. Query Hints

Usually this table will not be queried in isolation, but in combination with the linked tables. For queries involving those tables, please refer to their specific sections. Should a query on just this table be required, then the main thing to take into account is to only return a single record per entity. This can either be done by viewing the current record:

```
SELECT
        <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStatus = 'C'
```

Or by querying a point in time (although runs the risk of eliminating entities that were created after the specified date:

```
SELECT
        <Desired Fields>
FROM DIM_CO_Valuation
WHERE SCDStartDate <= @ReportDate

AND SCDEndDate > @ReportDate
```