"Loan Securitization Application: Credit Portfolio Analysis and Investor Reporting"

Bachelor's Thesis

In co-operation with **Raiffeisen Zentralbank Österreich (RZB) AG**,
Am Stadtpark 9, 1030 Vienna, Austria

Author: **Ivailo P. Sokolov**, Student ID 0351477
Under the supervision of:
Dipl.-Ing. Mag. Dr. **Albert Weichselbraun**
Institute for Information Business,
Vienna University of Economics and Business Administration,
Augasse 2-6, A-1090 Vienna, Austria

Abstract

This bachelor's thesis explores the most important aspects of understanding and preparing the data for a Collateralized Debt Obligation (CDO) transaction. Extensive theoretical background is given on the key concepts of credit portfolio management and key parameters such as Probability of Default (PD), Loss Given Default (LGD), Expected Loss (EL), Risk Weighted Assets (RWA), default correlation, etc. Moreover, a chapter is provided on the reasons behind the 2007 world financial crisis that started with and because of CDO instruments. The software application project that goes with this thesis is described with its most important functionalities in Chapters 3 and 4 and background is given on Microsoft SQL Server along with two exemplary Transact-SQL functions for generating stratification tables out of a pool of assets in Appendix 1 and generating amortization profiles in Appendix 2.

Acknowledgement

I would like to take the time to sincerely thank Dipl.-Ing., Mag., Dr. Albert Weichselbraun who was kind enough to accept this thesis as well as for his support and patience during the project's life cycle. My gratitude also goes again to him for his inspiring and very informative 'Commercial Databases' and 'SQL-2' classes that immensely helped clarify a great deal of the database and SQL world for me.

Keywords

collateralized debt obligation, stored procedure, T-SQL, stratification tables, financial crisis, Microsoft SQL Server

List of Abbreviations

Below is an alphabetically sorted list of the abbreviations used herein:

Abbreviation	Explanation
ABS	Asset Backed Security
ACID	Atomicity, Consistency, Isolation, Durability
ADP	Microsoft Access Desktop Project
CAD	Capital Adequacy Directive
CBO	Collateralized Bond Obligation
CCF	Currency Conversion Filter
CDO	Collateralized Debt Obligation
CDOROM	Moody's Investor Service CDOROM Application
CDS	Credit Default Swap
CLO	Collateralized Loan Obligation
CSV	Comma Separated Values file
DAO	Microsoft Data Access Objects
DBMS	Database Management System
DLL	Dynamically Linked Library
EAD	Exposure At Default
ECB	European Central Bank
EL	Expected Loss
ER	Entity-Relationship model
GCC	Group of Connected Customers
GUI	Graphical User Interface
HTML	HyperText Markup Language
IG	Investment Grade securities
JET	Microsoft Joint Engine Technology
LGD	Loss Given Default

Abbreviation	Explanation
MS	Microsoft Corporation
ODBC	Open Database Connectivity
OeNB	Österreichische Nationalbank
OLE	Object Linking and Embedding
PD	Probability of Default
PK	Primary Key
RA	Rating Agency
RDBMS	Relational Database Management System
REIT	Real Estate Investment Trust
RWA	Risk Weighted Assets
RZB	Raiffeisen Zentralbank Österreich AG
SPV	Special Purpose Vehicle
SQL	Structured Query Language
SSMSE	Microsoft SQL Server Management Studio Express
T-SQL	Transact SQL
UDF	User Defined Function
UML	Unified Modeling Language
VBA	Visual Basic for Applications
WAL	Weighted Average Life

Table of contents

A list of the main topics in this bachelor’s thesis:

1. Introduction and Motivation

With a balance sheet of around € 160 billon the RZB Group is the third largest banking group in Austria, a leader and pioneer in Central and Eastern Europe (CEE) and an important niche player in several other markets worldwide. Driven by rapid expansion in recent years, RZB is evaluating new ways for achieving Risk Weighted Assets (RWA) relief.

One of the possible ways to achieve RWA relief and free capital for further expansion is through securitization of assets. One of the means to transfer the risk to external investors is to structure Collateralized Debt Obligations (CDOs) – a combination of loans/bonds and/or other assets being packed into a single synthetic or true sale transaction that is later marketed and sold to the investors in different tranches.

The importance of adequate IT implementation of such transactions is paramount not only to the regulatory and legal aspects of the transaction, but also for successfully handling operational and strategic management and marketing tasks. The project at hand will try to follow a synthetic Collateralized Loan Obligation (CLO) transaction from assembling the data from different bank systems, through making a cutoff portfolio filter selection, then keeping track of the performance of the loans and making replenishment, and finally to generating investor reports with stratification tables.

Although during the past ten months the global market for CDO/CLO has become illiquid to the extent it is virtually non-existent (at the time of writing of this chapter – Nov 2008), as soon as the market recovers CDO deals will be in the focus again, albeit in a possible different regulatory situation and with possible structural changes. Standard software for securitization is almost not applicable as such and many banks are developing their own securitization software tools, even though they are perfectly aware of the current financial markets circumstances.

Along with a literature research and a brief section covering the theoretical background of credit portfolio management the main outcome of this RZB case study should be a model of an application for analyzing a portfolio of an example synthetic collateralized loan obligation deal. Functions for loan filtering and cutting will be included, as well as for replenishment and Reference Registry data management. However, the project and this thesis will not contain any RZB client or system data whatsoever, nor will it represent RZB's system or data structure in any particular form. The primary focus will fall on Microsoft Transact SQL (T-SQL) functions for data manipulation analysis of the defined data model, as well as on integration of those functions into a Microsoft Access Graphical User Interface (GUI) connected to a Microsoft SQL Server 2005 database.

2. Theoretical Background: Credit Portfolio Management

In this section the basics of securitization are discussed. Definitions are provided for some terms and figures that are especially relevant in this bachelor's thesis.

2.1 Collateralized Debt Obligations and Basics of Asset Securitization

The current project follows the gathering and analysis of data for the purposes of securitization. Securitization is a structured finance process in which assets – loans, bonds and other fixed-income receivables are put together and packaged into securities serving as collateral and sold to third party investors. Because those securities are always associated with the cash flow from the underlying pool of assets, the term Asset-Backed Securities (ABS) is often used as a generic term and many types of ABS are derived such as credit card receivables, auto loans, student loans, etc.

Collateralized Debt Obligations (CDOs) are a type of ABS and have been around since 1987. Collateralized Bond Obligations (CBOs) along with Collateralized Loan Obligations (CLOs) are actually CDOs with different types of underlying assets. CDOs are separate companies that own (or are connected with the cash flows of) those financial assets such as corporate loans or mortgage-backed securities [CDO_SA]. The CDO splits and sells the packages of cash flows to investors issuing notes for the different tranches – senior tranches (AAA), mezzanine tranches (AA to BB) and an unrated equity (also called "first loss piece") tranche. Those tranches are rated by recognized Rating Agencies (RAs) such as Moody's in this project. The tranches represent separate degrees of subordination, i.e. priorities of payment as losses are applied in reverse order starting from the equity tranche onwards. Coupons vary depending on the level of default risk whereby the AAA tranche pays the smallest coupon.

Depending on the type of funding and the desired outcome of the deal CDOs may roughly be classified as true sale (cash) and synthetic CDOs. In a true sale transaction not only the risk, but also the actual ownership of the underlying assets are fully transferred from the originator to the Special Purpose Vehicle (SPV) and are being refinanced through the capital market. The SPV is a corporate entity conceived specially for the purposes of the CDO deal to acquire the assets to be securitized and to also serve as a buffer by isolating some financial and regulatory risks. In case of a true sale CDO the complete pool of collateral to the underlying assets is also transferred.

Synthetic CDOs could be distinguished from true sale CDOs in the method of transfer of the receivables to the SPV. Unlike a true sale where ownership is legally transferred, in a synthetic CDO the credit risk to the underlying portfolio is transferred by the means of credit derivatives – Credit Default Swaps (CDS) [Br_Sch_05.] CDS are private contracts between two counterparties. The seller essentially gets periodic payments from the buyer and in return must cover the losses in case of a default (credit event) in regard of the reference entity [CFA_2008]. Essentially, although depending on the used accounting regulation, the securitized assets hence remain on the banks’ balance sheets [Br_05].

CDOs are created for three main purposes:

- Balance Sheet – whereby a holder of assets eligible for a CDO would like to shrink its balance sheet, reduce required regulatory and economic capital or just achieve cheaper funding costs

- Arbitrage – where an asset manager desires to gain assets under management and collect management fees. The assets are purchased from the sellers throughout the market and put into the CDO. Investors may wish to have the expertise of an asset manager and CDOs are one of the possible services for asset managers to offer their clients beside mutual and hedge funds

- Origination – Banks, insurance companies and REITs (real estate investment trusts) may wish to increase their equity capital [Dev_CDO]

The first CLO emissions in Europe began in 1995 as banks tried to lower the amounts on their balance sheets in order to increase their own capital [Raiff_Res]. Total global issuance peaked in 2006 at US $ 552 billion, reaching US $ 503 billion in 2007 and declined sharply as from Aug. 2007 due to the US sub-prime mortgage crisis [SIFMA_08].

2.2 Credit Portfolio Management Key Concepts

Banks have already abandoned the traditional transaction-by-transaction "originate-and-hold" approach and have already moved to the "portfolio approach" much like an investor. Capital plays a decisive role in this approach. Due to the broadness of the term "capital" it is usually split into three main categories [CPM_2003]:

- Equity capital, representing in a bank the sum of shareholder's equity and any retained earnings
- Regulatory capital, which refers to the risk-based capital requirements. Its purpose is to provide adequate resources
- Economic capital, which relays to the risk of the assets. The point of view is focused on determining how much capital is actually needed. It is a statistical measure of the resources required to meet unexpected losses over a given time period (e.g. one year) with a given level of certainty (e.g. 99,9%)

Modern portfolio theory, first defined by H. Markowitz in the 1950s, argues that by combining assets in a portfolio a higher expected return for a given level of risk is achieved. Alternatively, less risk could be achieved, for a given level of expected return. Diversification and correlation of assets play therefore a key role. Modern portfolio theory is also based on two critical assumptions – one that investors are "risk averse" (i.e. they prefer a lower risk where possible) and two that security returns are jointly normally distributed. Equity and credit asset portfolios differ substantially in the loss distribution. Due to the fact that losses of a traditional "originate-and-hold" credit portfolio are not normally distributed, large data sets have to be collected to simulate the loss distributions (to simulate the "long tails distributions") [CPM_2003].

Below is a list of abbreviations that are used in this project and are also present as fields in tables in the database:

- RWA (Risk Weighted Assets) is a term used to differentiate the bank’s assets according to their credit risk. Lower risk assets are discounted and some assets could even have 0% risk weight, e.g. some sovereign (government) debt. RWAs are relevant to the amount of required regulatory capital

- EAD (Exposure at Default) is the amount outstanding in the event of, and at the time of, counterparty’s default. Under Basel II banks need to provide EADs for each transaction in the banks’ internal IT systems. EAD for facilities (loan commitments) is according to Basel Guidelines the amount likely to be drawn in case of a default [BIS_1]

- LGD (Loss Given Default) is the amount of EAD that will not be recovered in case of a default. Basel II allows banks to assign fix LGDs to some claims not secured by collateral. For example senior claims on corporates, sovereigns and banks automatically get 45% LGD. Subordinated claims on those entities attract 75% LGD [BIS_2]

- PD (Probability of Default) is a function of the connection of company, industry and economy variables. It is essentially an internal risk rating [CPM_2003]. Under Basel II regulation the PD parameter is the likelihood that a loan will fall into default and is used in the calculation of economic and regulatory capital [BIS_3]

- EL (Expected Loss) is the mean of the loss distribution. It is not a risk but rather it is the cost of doing business, thus the price of any transaction must cover expected loss. Hence, when a bank determines the pricing of a particular loan type, the fact that some of the loans in that pool are expected ultimately default is taken into account

- WAL (Weighted Average Life) is a measure of credit risk and represents the weighted average of the times of principal repayment of an amortizing loan. The following formula illustrates the calculation of WAL, which is also implemented as a T-SQL function in this project:

$$\mathrm{WAL} = \sum_{i=1}^{n} \frac{P_i}{P} t_i,$$

WAL is thereby the sum of the fraction of the principal repaid in the coupon *i* multiplied by the time from the start of the coupon *i.*

2.3 Current Market Crisis and Collateralized Debt Obligations

The global financial crisis initially started in July 2007 when investors lost confidence in the value of CDOs consisting of securitized, mostly sub-prime mortgages. Essentially, when housing prices in the United States started to decline in 2006 coupled with an increase in the interest of floating-rate mortgages, delinquencies and foreclosures soon followed. The problem was that by that time ABS and CDOs against those mortgages were already massively issued and held by many major American and European financial institutions and the sudden decline in the quality of the collateral along with the already high leverage ratios of the biggest U.S. banks triggered extreme declines in own capital ratios that resulted in bank failures and global shortage of credit, equity markets crashes and even currency crisis in some countries. The crisis is still ongoing and the notional size of all losses is not yet fully clear [Calomiris_Nov_08].

2.3.1 Collateralized Debt Obligations as a Reason for the Crisis

A very fundamental and basic assumption in the structured finance practice and literature has been just challenged as wrong: namely that through pooling of financial assets such as loans and bonds, the multitude of the produced tranches is much safer than their underlying pool. The main reason behind the current global financial market crisis is the empirical proof and unfortunate discovery that most, if not all of such securities are actually far riskier than many originally thought [HBS_Nov_08].

According to a recent Harvard Business School official publication [HBS_Nov_08], two key features of the CDO instruments in general fueled its spectacular issuance growth and inevitably led to the effective current shut down of this market:

- The first argument behind the fall of structured finance is that issuing a capital structure amplifies errors in evaluating the risk of the underlying assets. The core lies in the default correlation of assets – i.e. even slight errors in estimating the default correlation between the assets in the pool result in exponentially wrong assumptions about the risk of the single tranches in a CDO. The effect is all the more exaggerated if CDO^2 structures are used (CDOs with CDOs as underlying instruments). This essentially retranslates into completely wrong ratings being issued by the rating agencies

- The second important factor is the fact that the manufacture of CDOs and the securitization process in general substitutes risks that are largely diversifiable for risks that are highly systematic. In effect, during a severe economic downturn structured securities suffer much more damage than traditional corporate securities of equal rating. Key word here is also the rating itself, provided by the rating agencies who have been in the single-name rating business for decades but have only recently started to include structured finance ratings giving them same-scale ratings. The widely popular AAA, AA, A, BBB, etc have been used as independent creditworthiness assessments for long enough time so that not only individual private investors, but also big banks and government agencies have been led to believe that an AAA-rated CDO is as secure, if not more secure than a traditional AAA-rated corporate bond

To illustrate the first argument above regarding the amplified problem of default correlation, the process of pooling and tranching will be illustrated by an example. Let us assume a portfolio pool consisting of two identical securities A and B, both of which exhibit the same probability of default P_d. Upon default the assets would pay 0 €, otherwise if not they would return 100 € each. Thereby the total notional value of the underlying portfolio would be 200 € and two 100 € tranches – a senior and junior could be issued against that capital structure:

- The junior tranche would be structured as such as to bear the first 100 € losses on the portfolio and would pay 100 € if both assets do not default
- The senior tranche would only default if in this case both assets default

Critical here in order to be able to calculate the expected cash flows for both tranches is the necessity to know the probability of both assets defaulting at the same time. In the example above it is described by a single parameter – the joint probability of default that represents the default correlation of the assets [HBS_Nov_08].

It is exactly the gross miscalculation of this parameter that played the critical role in the wrong estimations of the Rating Agencies, resulting in the incorrect ratings given, the subsequent exhaustion of AA and AAA-tranches that seemed to be untouchable before, the following downgrades of the CDOs, the downfall of their prices, and the immediate losses to anyone that held them on mark-to-market valuation in their books.

Therefore, in the aforementioned example if the assets have **no correlation** whatsoever, the senior tranche would pay either 0 € or 100 €, just like the individual assets, *except* that it would be *less* likely to default than any of the assets alone (e.g. if A and B have $P_d = 10\%$, the senior tranche has $P_d = 1\%$). However, if the two bonds are **perfectly correlated,** i.e. they would default at the same time, no credit enhancement (the ability to possess lower risk of default due to diversification) is achieved for the senior tranche. Naturally, the higher the default correlation between the assets in the underlying pool, the greater risk that the senior claim inherits from the underlying assets.

To illustrate how extremely hard to determine through historical data and thus in effect misleading the estimation of the joint default correlation parameter is (at least before the full blast of the sub-prime and then the global financial crisis), one needs to take a look at how close the annualized default rate values are within the first 10 rating categories of the big Rating Agencies. Fitch's Portfolio Credit Model showed variations between 0.02 and 0.75 percent within that range, until the model was eventually updated in December 2008 [FITCH_1]. Fitch Inc. also concluded in 2007 that almost 60 percent of all global structured products were AAA-rated, in contrast to less than 1 percent of the single-name corporate issues. According to Charles Calomiris, Columbia University Professor, "idiotic Loss Given Default and Probability of Default assumptions" were made by the Rating Agencies due to "plausible deniability equilibrium" circumstances that allowed for such low loss projections that in effect fueled the high leverage and huge growth [Calomiris_Nov_08].

This has proved true to the shock and dismay of all, as the sub-prime crisis spread more and uncorrelated assets (such as mortgages in different regions in a country) suddenly became highly correlated with sharply rising default and delinquency rates once the floating-rate part of their payments increased sharply in accordance with the general interest rate. A, AA and even AAA tranches CDO transactions once deemed safe were suddenly hit and exposed as “toxic”, due to the high default correlation of their underlying pool of assets. Thus, apart from the idiosyncratic risk associated with each asset’s creditworthiness and risk, the systemic risk, i.e. the systemic risk of the macro environment in which the assets or assets’ holders are active, represent a significant threat [Chan-Lau_Lu_2006].

2.3.2 Crisis’ Early Symptoms, Some Indicators

A number of scholars, analysts and prominent investors, have expressed enormous concern that the credit boom that followed in the mid-2000s after the world had recovered from the dot-com crash brought inevitable high systemic risks associated with the abovementioned fundamental flaws attribute to nearly all types of ABS-like derivatives as well as the rise of Credit Default Swaps. Notwithstanding the ruling majority of theories that explained and justified the positive side of the financial engineering revolution taking place with securitization and credit derivatives swiftly gaining pace along with the positive effects of globalization and global interconnectivity, some were still concerned about the side effects, risks and eventual worst-case scenarios that could arise from the unregulated spreading of the securitization practices in general [WSJ_Feb_08].

Some experts such as Morgan Stanley’s Chief Economist Stephen Roach warned on asset bubbles being potentially blown by the central banks [WSJ_Feb_08] keeping the interest rates low for a prolonged time after the dot-com crash recovery while others researched the links between low interest rates and excessive risk taking and predatory lending. [Ioannidou_et_al_07] The International Monetary Fund’s former Chief Economist Raghuram Rajan on the other hand explained the reasons for higher risk tolerance by the skewed incentive structure for financial market participants. This forced fund and bank managers to originate and securitize exponentially more loans (see figure 1 below) thereby causing the high leverage ratios and eventually the sub-prime crisis [IMF_Rajan_2006].

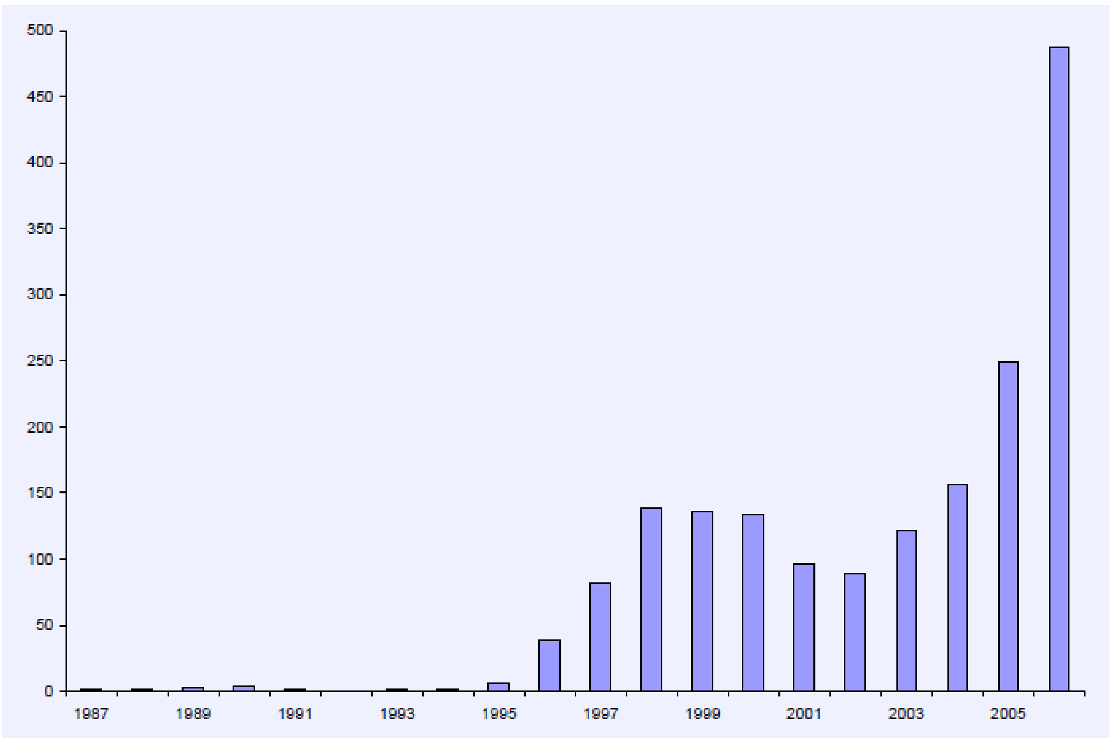


Figure 1: Annual Cash CDO Issuance, Source [Calomiris_Nov_08]

In addition to Cash and Synthetic CDOs, Single Tranche CDOs had also gained wide popularity by the time the sub-prime problems began to be visible. Single Tranche CDOs consist of a single tranche, whereby the investor is exposed to any losses that might occur in a specified range within the predefined so-called attachment points. E.g. an investor in a 6-9 % mezzanine tranche is liable only for the losses within that range. This is possible due to the flexibility of the Credit Default Swaps. In Europe, the iTraxx index became very popular, consisting of the top 125 Investment Grade (IG) companies in Europe, having 0-3%, 3-6%, 6-9%, 9-12% and 12-22% standard attachment points.

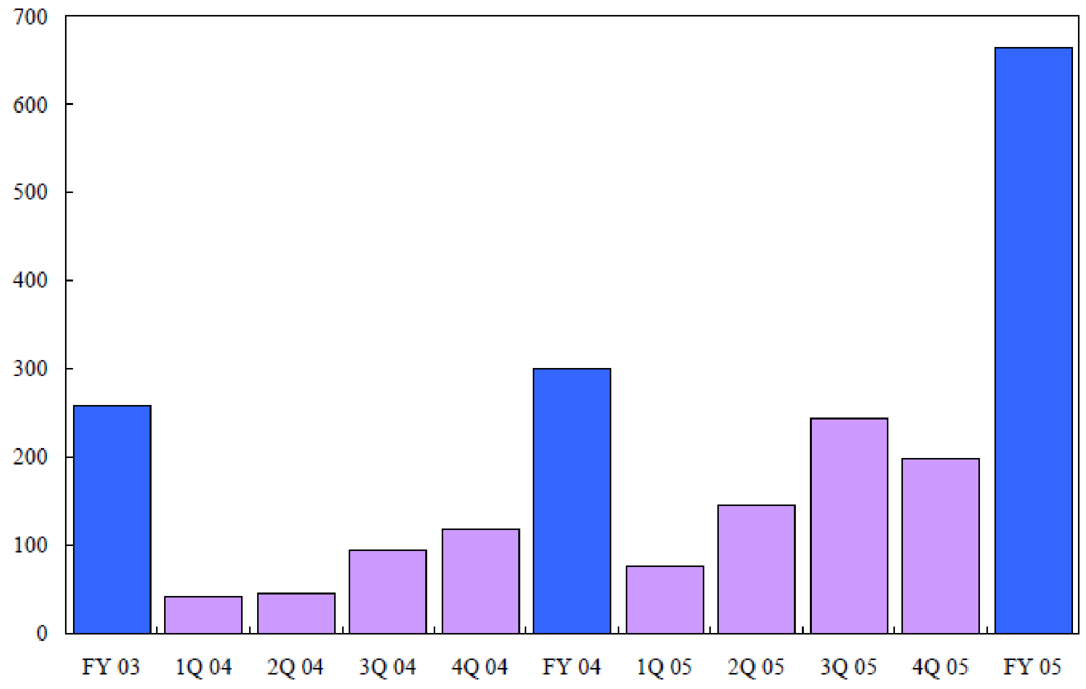


Figure 2: Single Tranche CDOs total notional amount outstanding 2003-2005 in the United States alone, Source Credit*flux*

As early as April 2006, IMF's economists Chan-Lau and Lu, using the market prices on STCDO tranches, created a model for tracking the idiosyncratic and systematic risks in a portfolio and the market where the portfolio is selected. By observing price co-movements of the equity and super-senior tranches allowed them to identify the driving force in the market: idiosyncratic risk or systemic risk. They concluded that systemic risk is a major concern for those in charge of ensuring financial stability and presented charts and data that pinpointed future problems concerning systemic risk [Chan-Lau_Lu_2006].

2.3.3 New CDO Issuance and Developments in 2009

Due to the general inability to place new CDO deals on the market especially after the Lehman Brothers failure, European banks have begun to concentrate on structuring new ABS/CDO exclusively for the purpose of accessing liquidity from the European Central Bank (ECB). The newly structured

Asset Backed Securities are used as collateral for a repo (repurchase agreement) with the ECB and are subject to special "haircuts", i.e. essentially accepted by the ECB at discounted prices.

However, those CDOs should adhere to special eligibility criteria and a plethora of additional rules [ECB_Nov_08] in order to be accepted due to what seems to be the general intent of the ECB to act as 'lender of last resort' against repos with quality assets only, rather than to become 'purchaser of last resort' of troubled or toxic assets (such as the U.S. TARP program). This intent was strengthened by the Sep 4, 2008 tightening of the criteria (increase in haircuts in valuation, wider prohibition on collateral which has "close-links" with the pledgor and amendments to the requirements for rating agency reports [Orrick_08]). Jan 19, 2009, the ECB extended further the restrictions, requiring AAA rating at issuance and excluding CDO^2 completely from eligibility. Feb 3, 2009, Moody's Investors Service announced the new version of its CDOROM modeling tool that has significantly updated assumptions regarding asset default probability, correlation and recovery rate [Moody's]. An export to Moody's CDOROM is handled by the software application in this thesis (see chapter 4.6).

2.3.4 Examples in Austria, Application in Practice

A preliminary version of the software application outlined in this thesis was used for filtering RZB's portfolio and helped cutting out two CDO transactions unique to the Austrian market – Cash Collateralized Bond Obligations (CBOs) with total principal amount of the notes of ca. 1,5 Bn € in Dec 2008. The transactions were used as collateral for the European Central Bank for provision of liquidity in order to secure additional funding and bolster further the bank's relatively good own capital ratio.

3. Project Description

The initial definition of the project as well as technical details and the way the project was run are presented below.

3.1 General Project Description

The original idea of this project was to construct a software application to support the process of securitization in order to achieve RWA relief. Initially a synthetic Collateralized Loan Obligation (CLO) was the intended type of transaction to be modeled, but eventually two Collateralized Bond Obligations (CBOs) used as collateral for repoing with the European Central Bank were successfully cut out using the software.

Technically, the primary goals of the software application are the gathering and organization of bank-wide department data within one single database with a layer of functionality on top of that database. Forms for filtering and cutting, queries, reports and export files had to be generated, all securitization and investor reporting related. Although not the primary concern at first, as the current financial crisis spread out and global markets froze, the orientation of the project inevitably turned also towards more filtering functionalities in order to increase the options and ways to find appropriate assets for structuring – maximizing the potential portfolio volume.

The software project concept was created with the intent of keeping the data structure as bank-neutral as possible, although being realized as part of the IT-Practice course. It does not represent or cite RZB's core bank systems or data structure in any particular form, rather it is a generalized approach at researching and handling the technical difficulties of organizing data needed to originate relatively complex financial instruments such as CLOs or CBOs.

3.2 Choice of Software Development Tools

In order to realize such a software project in a non-IT department of a major bank with standardized end-user PCs, the potential choice of development tools available was very narrow. Microsoft Office tools, essentially Microsoft Excel with Visual Basic for Applications (VBA) scripts and/or Microsoft Access as a database and database front-end application were the only tools available, because of the inability (security and process concerns) to install a web server for a web interface. Nevertheless, after a research on the potential data load that had to be channeled through the application the need for a full-scale Database Management System (DBMS) was clearly identified. Microsoft SQL Server 2005 was chosen as the DBMS for this project along with a Microsoft Access ADP (Access "Desktop" Project) Graphical User Interface (GUI) to for client PCs.

Microsoft Access is a relational DBMS based on the Joint Engine Technology (JET) as the underlying relational database engine and a number of GUI and software development tools including VBA as object-based scripting language. The JET engine supports a variety of basic relational database functionality – entity (i.e. the ability to keep identifiable records with primary keys that are not null) and referential integrity (the ability to enforce consistency between coupled tables via a primary and foreign key), locking mechanisms for concurrent access (pessimistic and optimistic locking on the `UPDATE` statement), constraints definition on the fields, ANSI-92 SQL queries, Unicode characters as well as views [MSDN_JET]. The JET/Access database is kept within a single file and could be accessed using Microsoft's Data Access Objects (DAO) in Visual Basic (or in Microsoft's Active Server Pages). From a purely technical data access point of view, JET is already considered a deprecated technology by Microsoft and as of 2007 is no longer included in Microsoft Data Access Components (MDAC). [MDAC_Roadmap]

Specific to SQL Server 2005 is the proprietary Transact-SQL (T-SQL) extension to SQL, a dialect that Microsoft adapted since Sybase SQL Server. [Wikipedia, T-SQL] T-SQL enhances the standard ANSI-SQL with [Factsheet_TSQL]:

- Control-of-flow features - `BEGIN` and `END`, `BREAK`, `CONTINUE`, `GOTO`, `IF` and `ELSE`, `RETURN` and `WHILE`. `TRY` and `CATCH` are also used for capturing errors when executing a sequence of queries in a `BEGIN` and `END` statement blocks. All of the mentioned control-of-flow statements find usage in stored procedures and functions throughout this project
- Definition of variables local to the script running them (global variables are not supported though), whereby the variables are typed
- Various support functions, e.g. for handling date and time and strings processing
- Improvements to the `DELETE` and `UPDATE` queries (a `FROM`-clause could be added whereby other tables could also be joined). The ability to join multiple tables in `UPDATE` queries simplified significantly some of the queries in this project

The Express Edition, which is the scaled down, free-of-charge edition of SQL Server also runs the application in this project without any problems, due to the lack of any features specific to the more powerful versions such as database clustering, full-text searching or the integrated reporting and analysis services.

All tables and views in this project that were initially created in the original Microsoft Access JET-Engine format were upsized to SQL Server 2005 with the integrated Microsoft SQL Server tools. Although the majority of MS Access applications of such size do not need upsizing, some specific factors such as the number of concurrent network access users (over 5 users), the need to use programmability features such as stored procedures and table valued inline functions as well as the some security issues contributed to the decision to upsize to MS SQL Server [Haught_Upsize_SQL].

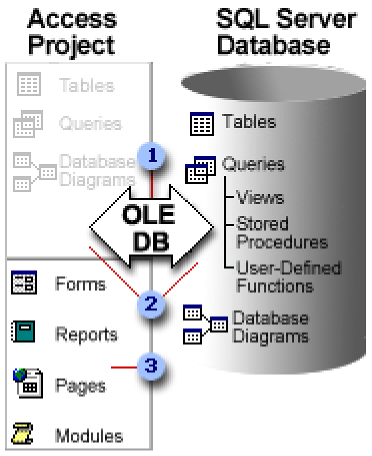


Figure 3: A connection to MS SQL Server via OLE DB [MSDN_02]

The Microsoft Access project being used represents a Microsoft Access data file that provides a relatively fast native access mode to a Microsoft SQL Server database through the OLE DB component database architecture (that also provides access to many other types of data sources, including relational data, flat files, and spreadsheets). Unlike a typical Microsoft Access database an Access project does not contain any data or data definition based objects such as tables, views, database diagrams, stored procedures or scalar or inline functions. Instead, these database objects are stored in the SQL Server database. In essence, an .ADP file contains only code-based or HTML-based database objects: forms, reports, the name and location of data access pages, VBA macros, and modules [MSDN_02].

Instead of using Microsoft Access for editing the database entries or writing the SQL queries, SQL Server Management Studio Express (SSMSE) is used in this project for configuring, managing, and administering all components within Microsoft SQL Server. SSMSE centers on the Object Explorer for navigating through the database objects and integrates visual and script tools for authoring them. SSMSE is provided free of charge for download at Microsoft's website, but the Express version lacks certain features such as Analysis Services, Integration Services, Notification Services and Reporting Services [http://www.microsoft.com/downloads].

3.3 Project Cycle and Project Phases Results

The project research and development process should normally be sequential, moving closely along the definitions of the waterfall model. The following phases were considered:

1. Requirement and field analysis
2. Project definition and system design
3. Data model design
4. Functional and technical implementation
5. Test case presentation

Each new phase may only begin after the previous one was completely and thoroughly researched, documented and implemented and all of the resulting documents are prepared. However, minor changes in already completed phases were done due to serious grounds such as changes in the structure of the export data from the bank's core systems. The only exception is the data model design phase, where the Entity-relationship (ER) model and database diagrams could be updated, should more database fields become necessary later on during the realization of the project. Although not a best-practice, the sheer complexity of this project implies minor changes of the data model, in order to fit the time frame set for completion. Each project phase should deliver tangible and clearly identifiable results. Here is a list of the results from each phase:

Phase	Result
1. Requirement and field analysis	• Literature research • An extensive section on the theoretical background of credit portfolio analysis • Definition of terms – CDO, CLO, Reference Obligation, Reference Registry, Pool Cut etc. • Analysis of the relevant aspects for the IT implementation
2. Project definition and system design	• Definition of the exact project scope and the results expected; definitions of assumptions and conditions • Description of the used software; choice of Database Management System (DBMS) • Project test case definition • UML Component diagram
3. Data model design	• Physical data model – ER model, database diagram
4. Functional and technical implementation	• SQL tables, views, functions and stored procedures implementation • User interface implementation
5. Test case presentation	• Filling the database with dummy example data • A presentation and go-though along the lifetime of a transaction

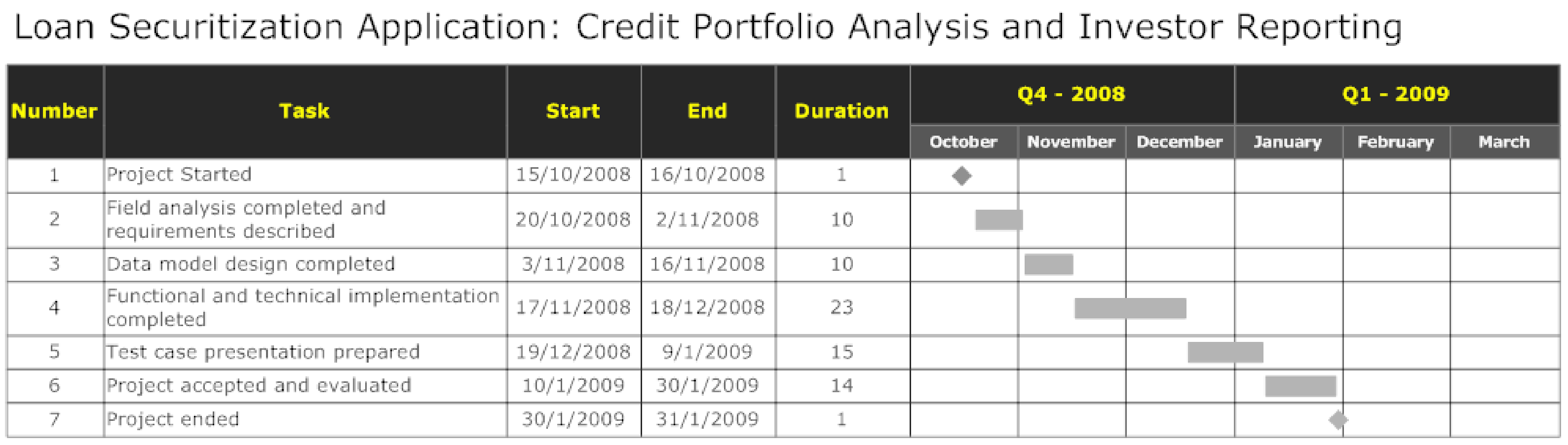
3.4 Project Milestones

The project milestones in this bachelor's thesis follow closely the different project phases defined in 3.3. Below is the list of milestones.

Milestone	Deadline
1. Project started	20th October, 2008
2. Field analysis completed and requirements described	3rd November, 2008
3. Data model design completed	17th November, 2008
4. Functional and technical implementation completed	19th December, 2008
5. Test case presentation prepared	10th January, 2009
6. Project accepted and evaluated	30th January, 2009
7. Project ended	12th February, 2009

3.3 Project Gantt chart

Figure 4 below shows a Gantt chart of the milestones of this project described in Section 3.4 with the deadlines relevant to each milestone.



Loan Securitization Application: Credit Portfolio Analysis and Investor Reporting

Number	Task	Start	End	Duration	Q4 - 2008			Q1 - 2009		
					October	November	December	January	February	March
1	Project Started	15/10/2008	16/10/2008	1	◆					
2	Field analysis completed and requirements described	20/10/2008	2/11/2008	10	■	■				
3	Data model design completed	3/11/2008	16/11/2008	10		■				
4	Functional and technical implementation completed	17/11/2008	18/12/2008	23		■	■			
5	Test case presentation prepared	19/12/2008	9/1/2009	15			■	■		
6	Project accepted and evaluated	10/1/2009	30/1/2009	14				■		
7	Project ended	30/1/2009	31/1/2009	1				◆		

Figure 4: Project Gantt-chart

3.5 Unified Modeling Language and Entity-Relationship Diagrams

The system modeling of this project was done with the open source software tool Dia [www.gnome.org/projects/dia]

The UML Component Diagram below outlines the objects relevant to the IT-aspect of the securitization process. Notwithstanding the fact that not all components were feasible to be realized, it serves as a guideline and an overview of all the needed functionality to make it an all-round application for securitization.

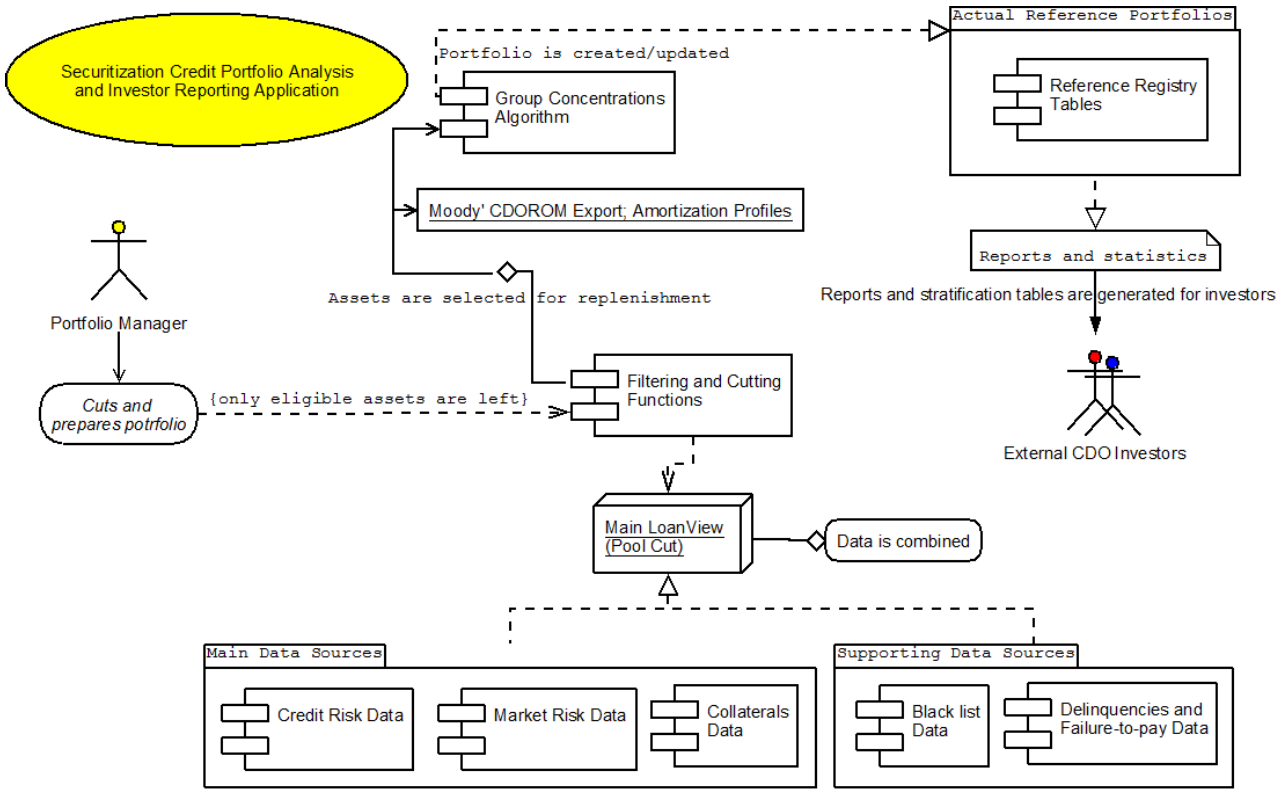

Securitization Credit Portfolio Analysis and Investor Reporting Application
Portfolio is created/updated
Group Concentrations Algorithm
Actual Reference Portfolios
Reference Registry Tables
Moody' CDOROM Export; Amortization Profiles
Reports and statistics
Reports and stratification tables are generated for investors
Portfolio Manager
Assets are selected for replenishment
External CDO Investors
Cuts and prepares potrfolio
{only eligible assets are left}
Filtering and Cutting Functions
Main LoanView (Pool Cut)
Data is combined
Main Data Sources
Credit Risk Data
Market Risk Data
Collaterals Data
Supporting Data Sources
Black list Data
Delinquencies and Failure-to-pay Data

Figure 5: UML Component Diagram of a fully-fledged investor reporting and portfolio analysis application

The Entity-Relationship (ER) diagram below tries to show the important table entities in the project. Marked grey are data source import tables (see Section 4.1), whereas `tblLoanView` is shown bold due to the fact that the application centers around it and almost all other tables compute data to compliment it. Multi-value attributes are used instead of writing out all attributes for the purpose of simplicity. Special markings also have the output tables, namely stratification tables reports (see Section 4.9) and amortization profiles (section 4.8). For additional clarity of the diagram, all attributes stem from the corners of the entities, whereas all relations stem from the middle.

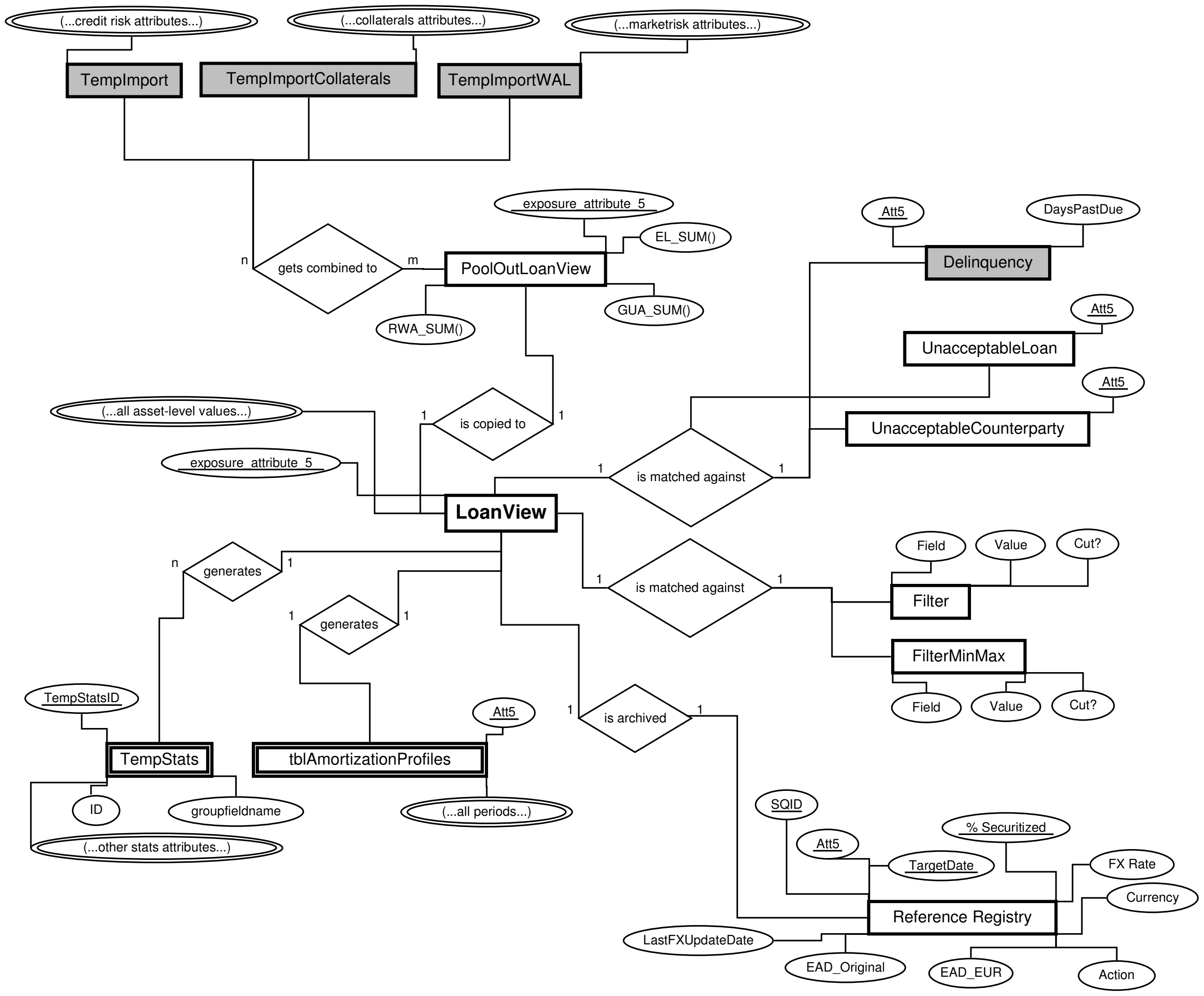


Figure 6: ER diagram with the most important entities in this project

4. Implementation

Chapter 4 goes into the technical details of the software application. References to the MS SQL Server database entities – tables, views, stored procedures and functions are made. The structure of the application as well as its most important functionality is described below. Noteworthy is that Chapter 4 follows the sequence of operation of the application, i.e. the workflow path of the procedures/functions to be run at any target date, *inter alios* importing the data, combining the main loan table (`tblLoanView`) from multiple views with aggregated values (RWA sum, etc.), calculation of all asset collaterals from applicable categories, calculation of WAL for each asset, pre-filtering (matching blacklists, removing those assets that are already 100% securitized, etc.), filtering (cutting based on custom defined filters), establishing the maximum group concentrations limits, exporting the remaining assets in `tblLoanView` to Moody's CDOROM Monte-Carlo based simulation and if it shows satisfactory results printing stratification tables (used e.g. for investor reporting).

4.1 Data Sources Description

In this project a general data quality situation as at any major bank is presumed. In order to have enough information to prepare a CDO for securitization the below listed data sources represent the minimum range of information needed, whereby that information is usually obtained from several different bank core or supporting systems. It is important to note that any of those data sources could be imported in the application via the buttons in the Import data tab in the main form from either a .CSV (comma separated values text file), a .XLS (MS Excel file with a single sheet) or a .MDB (MS Access file with a single table). The data sources are:

- **Credit risk data**, which is the main data source around which the application is structured. Credit risk data is normally used for example for regulatory reporting purposes e.g. to the OeNB (Austrian National Bank). The data from this data source is oriented towards the Capital Adequacy Directive (CAD) of Basel I & Basel II to facilitate the regulatory as well as economic capital calculation. It encompasses some basic indicators specific to each asset and asset class such as PD, LGD, CCF, exchange rates and collateral values as well as RWA calculation values

- **Market risk data**, used for calculation of the actual Weighted Average Life of loans (see Section 4.4). The market risk data refers to the amount of money actually outstanding as well as the exact target dates when the money on the principal of the loans is to be repaid. E.g. whereas the credit risk data shows the credit risk amount (such as a loan that has been granted to a customer), market risk data shows exactly how much money the customer has drawn out. Amortization tables are also produced from this data (see Section 4.7)

- **Collaterals data**, extracted for determining the extent to which a certain asset is secured. Extensive collateral information – type of collateral, collateral category and mode, is provided on account and currency level. This data has to be mapped by building a multi-variable attribute ("attribute5") to match the data throughout the application (such as the credit risk data and data from other sources). The combinations of collateral type, category and mode are mapped to 7 different bucket categories, and a T-SQL function is used to aggregate the amount of the sum of the collaterals for each category

- **Delinquency and default/failure to pay data**, used to filter out already troubled and potentially troublesome assets. The data is usually delivered in MS Excel or MS Access format automatically generated by the bank's core systems and gets imported in the application and then mapped/deleted on asset level. Depending on the MaxDelinquencyDays_Filter parameter in the Settings tab / `tblSettings`, the delinquent assets are cut accordingly. The max days past due of an asset for securitization could differ among transactions due to legal or risk requirements

- **Black list data** for assets non-eligible for securitization, used to filter out wrongly assigned assets and those where securitization is not applicable due to other, for example customer related issues. Black lists on asset as well as entity group level are provided in the application. Stored procedures with `DELETE` queries joined with the main `tblLoanView` are used to immediately filter out any entities matching the black lists. A special form, *frmEditUnacceptableCounterparty*, invoked from the Blacklists tab in *frmMain*, facilitates the edition and removal of counterparty-level blacklists and provides search capabilities as well (e.g. looking up a counterparty's id by company name)

4.2 Combination of Data Sources – "Loan View"

The **credit risk data source** is the central piece of information towards which all other data sources throughout the application are matched. It contains the main asset-level data (relevant for filtering in the securitization process), such as asset contract type and ID, Exposure at Default (EAD), Loss Given Default (LGD), Probability of Default (PD), and Expected Loss (EL). Supplementary information later on used for filtering and determination of the group concentration limits is also contained herein. Counterparty ID and Group ID represent the asset's company structure within the bank, i.e. a company may have a loan and its parent company may have another one, but they are both treated as belonging to the same group. One of the key aspects used for filtering is the asset's internal rating, determined by the bank's corporate or financial institutions analysis departments. The internal rating is then mapped to Moody's Investors Service (or any other) rating agency's ratings via a predefined table, which also contains the notching points for middle-ratings.

The table `tblLoanView` is comprised of the credit risk data and contains additional fields for matching data from the other data sources as well. Moody's rating, Percentage Secured and Weighted Average Life of each asset are mapped and written into `tblLoanView`. Other key values such as RWA sum, EAD sum (total amount outstanding) as well as sum of the unsecured parts of a given asset/account are also calculated and saved in `tblLoanView`.

It is noteworthy to mention that all functionality in the Match/Combine Data tab in *frmMain* tab, used to calculate values is a combination of T-SQL stored procedures, functions and views. A stored procedure is a precompiled collection of SQL and control-of-flow statements that are processed as a unit; a T-SQL function on the other hand takes parameters, has a body of statements but also returns a value or a table. [MSDN_02]

The initial import of data into `tblLoanView` is done via the button "Create Basic Loan View" in the same tab, which runs the T-SQL stored procedure `sp_CopyToLoanView`. This stored procedure tries to re-import the contents of the raw `tblTempImport` table into `tblLoanView`. Instead of using `tblTempImport` directly, rather the view `vwPoolOut_LoanView` is used that contains sums and other aggregation key data from `tblTempImport`. Using a `WHILE`-loop along with `BEGIN TRY/BEGIN CATCH` T-SQL statements, the stored procedure copies all valid records from table `TempImport` into `tblLoanView`. The incorrect records, i.e. those that contain stings in integer fields or the like are moved in the `CATCH`-clause to the table `tblTempImportError` accordingly.

4.3 Calculation of Collaterals

Every asset/account could possess multiple collaterals, which would be sold to recover any losses that occur in case of a credit event. Assuming the collaterals are stored in a database outside of the application, a MS Access file exported from the external database has to be imported in the application. The table `tblTempImportCollaterals` is used for this purpose and it delivers the relevant fields for the current target date: Contract Nr, Collateral Category (for example guarantee, security paper, mortgage, cash etc.), Type of Collateral (pledge, escrow), Other Characteristics (mode – with/without), Account Nr, Currency and the actual Amount.

After the aforementioned data is imported via the *funImportFromAccess()* VBA function into the SQL Server database to table `tblTempImportCollaterals`, a number of processing steps are being consecutively executed against the data:

- The sum of the collateral per account is determined

- Collateral type, category and mode are differentiated; an assembled attribute 'Category_Type_Mode' is built

- Guarantees from the credit risk data source (`tblLoanView`) are taken whereas the guarantees from the collaterals data source are ignored (an assumption is made for this project that the guarantees from the regulatory credit risk data source are stress-tested and those listed in the original collaterals data source are not and are thereby not accepted as collateral by the RAs)

- The multi-value attribute 'Category_Type_Mode' is split into eight basic categories which could later on be used for filtering of assets/accounts: n/a, cash, securities, real estate, machinery, receivables, other, guarantees

The T-SQL stored procedure `sp_CalculateCollaterals` which is called via the Calculate Collaterals button from the GUI takes care of those steps. It utilizes the MS SQL Server 2005-specific `PIVOT`-command, which transforms multiple rows into multiple columns in a single row. `PIVOT` rotates a table-valued expression by turning the unique values from one column in the expression into multiple columns in the output and performs aggregations on the remaining column values in the final output [MSDN_01].

4.4 Calculation of Weighted Average Life

The `CalculateWAL` T-SQL stored procedure (executed via a button in the Match/Combine data tab) is used to calculate the estimated Weighted Average Life for each asset. It takes the following values as input (from german – account number, loan number, currency, due date and expected cash flow):

tblTempImportWAL

kontonummer	kredit	währung	fälligkeit	cashflow
xxxxxxxxxxx	010	USD	29.09.2009	100000
xxxxxxxxxxx	010	USD	17.11.2009	200000

After this data is imported to the temporary table, it is moved to `tblWAL`, where calculations are being performed on it as follows: the volume of principal payments is being multiplied by the remaining payment periods and is being summed; afterwards this sum divided by the outstanding volume and that gives the asset's WAL.

4.5 Filtering and Cutting

Filtering the assets based on diversified static and dynamically calculated criteria is the central feature of the application. For this purpose a special form (*frmFilterMain*) is prepared that is invoked from the "Cut loans" button in the Filter/Group data tab of the main application form (*frmMain*). If the flag/checkbox "test purposes" is selected then pre-filtering is skipped and hence blacklists, unacceptable counterparties and delinquent assets, as well as already securitized assets in the Reference Registry history table (`tblReferenceRegistry`) are left in `tblLoanView` for cutting/analysis. After pre-filtering the form *frmFilterMain* pops up and provides a neat GUI for definition, save/load and execution of a filter against the assets in tblLoanView.

Standard fields for selection and fields for comparison have to be defined first in `tblSettings` in the Settings tab of *frmMain*. Then, the filter in *frmFilterMain* works twofold – there are two sub-forms for **key/value filtering** and **number range** filtering. The key/value filtering is selection based – delete all assets `WHERE field = value`, whereas the number range filtering is comparison based, i.e. delete all values `NOT BETWEEN` two numerical values). Exemplary default filter values are:

tblSettings	
Key	**Value**
FieldsForSelection	country_code;internal_rating_original;incorporation_country_original
FieldsForComparison	ead_pre_ccf|-1:1500000;RWA%|0,3:0,9
StandardValues_NotSelected	AT|DE|CH|CORP1|CORP2|CORP3

Thus, the filter in the table above denotes the following default filter – all assets from Austria, Germany and Switzerland (having country ISO-codes AT, DE and CH, matched against the country_code and incorporation_country_original `tblLoanView` fields) whose EAD is up to 15 Mio EUR, whose RWA range is 30-90% and that have the internal rating (creditworthiness) of CORP1 through CORP3.

The value field contains "|" and ";" delimited values that represent `tblLoanView` field names and range values. The StandardValues_NotSelected key represents all field values that will be omitted by cutting, i.e. those that should represent the portfolio.

Upon startup (the On_Open event), *frmFilterMain* takes those values and builds the tables `tblFilter` and `tblFilterMinMax` with the field/values combinations. Using the custom checkboxes the user then selects/removes additional values (sets the cut parameter to *true*) from the filter and/or modifies the standard range values directly. After the filter has been defined the *funFilterLoanViewTable()* VBA function is invoked via the Cut button. This function opens both filter tables (`tblFilter` and `tblFilterMinMax`) iteratively and executes the abovementioned `DELETE` key/value and range statements. Additionally it saves the filter history in `tblFilterHistoryTemp` with the intention left to save this table along with the other history values when adding the approved assets later on to the Reference Registry (thus enabling traceability of all the past filters applied).

Additionally, the Advanced Cut form is available, which shows an editable grid with `WHERE`-clauses of the `DELETE FROM tblLoanView` that enable custom delete queries to be defined and if the *AutoRun* flag is set, they are also deleted at the pre-filtering stage. This is necessary for some complexer filter queries on the portfolio that go beyond the key/value and min/max ranges, e.g. logical combinations of `AND` and `OR` statements such as `DELETE FROM tblLoanView WHERE (country = 'DE' OR country = 'CH') AND internal_rating_original NOT IN ('CORP2', 'CORP3', 'CORP4');`

4.6 Group Concentrations Algorithm

As was pointed out in Chapter 2, default correlation plays a major role in defining any portfolio for securitization. Hence, larger concentrations of assets within a single Group of Connected Customers (GCC) or loans exceeding certain limits have to be capped to a predefined value deemed appropriate for the transaction (the CapLimit value defined in `tblSettings` is set and retrieved on application startup in the *funGetGlobalVariables()* VBA function).

The group concentrations algorithm is intended to be run after all pre-cutting steps are run and the assets are filtered/cut. The key is to determine the maximum percentage securitized that any asset may have and save that value because it is important not only for determining the total volume of the portfolio (legally binding), but also for future reference when monitoring the portfolio. This refers especially to **synthetic CDO transactions**, where an asset is not actually fully transferred but only the potential gains and losses on it are assumed by investors (see Section 2.1). It is important to mention here that every month updated values for all data sources are imported and the new state/performance of the portfolio is updated and history is kept in the `tblReferenceRegistry` master table and its details tables that contain an exact copy of `tblLoanView` for each Target Date. Thus once determined the field "%Securitized" is being followed up until the asset has been repaid. As an example for different situations consider e.g. a large holding customer having 3 loans (6 Mio, 5 Mio and 2 Mio) to its 3 different sub-companies. Let our cap limit be 10 Mio EUR per group entity. That means that initially the 6 Mio asset could be fully securitized to 100%, the 5 Mio asset would only be securitized to 80% and the third asset will not be securitized at all at the time. Only after several time periods, after some of the assets from that group are repaid and limit is freed for the group could the third 2 Mio asset from the example be securitized.

The VBA function *funCreateGroupConcentrations()* is called via the button in the "Filter/Group data" tab. This function essentially does 3 things:

- Determines the sum outstanding for each group and writes it to each asset belonging to that group (field sum_out in `tblLoanView`)

- Iteratively goes though all assets to set the %Securitized for each asset. However, it first checks via the table-valued T-SQL function `funGetOutstandingPerGroup(caplimit, entity_group)` if there are already any securitized assets in `tblReferenceRegistry` belonging to the group of the current asset (aggregated by the entity_group parameter) and in such case it returns the lowered cap limit and sets a lowered %Securitized accordingly. This facilitates the central process of **replenishment**, i.e. the subsequent ramp up of the portfolio with additional assets, once the existing assets are repaid and limits have been freed up

- Eventually after having determined the %Securitized, another stored procedure is called to updates all records with the equivalent proportional percent collateral coverage (i.e. 50% of a loan is securitized thus 50% of the collateral could be allocated, should any credit event arise).

4.7 Moody's CDOROM Export

Moody's CDOROM is a Microsoft Excel based Monte Carlo simulation model used for calculating the expected loss on tranches of synthetic CDOs. It is the same model Moody's own analysts use to rate and monitor synthetic CDOs [Moodys]. A Monte Carlo Simulation is generally a class of algorithms that computes hundreds of thousands of scenario outcomes (the number of which is definable in Moody's CDOROM) by repeated random sampling, instead of a small number of discrete scenarios, whereby the ability to predict the probability of different outcomes is one of its key features. The actual Monte Carlo Simulation is a C++ Dynamically-Linked Library (.DLL) file that is run externally from Moody's CDOROM application and takes Microsoft Excel data as input. Our software application produces the data by the means of three different views that aggregate all values and compute and pass to CDOROM some dynamic parameters such as different haircuts for the different countries (DE, AT and CH only though) as well as country-based LGDs per Moody's definition. The three T-SQL-views (`vwLoanViewMoodysExportCDO1{_LGD{_Haircut}}`) utilize some additional constructs such as `CASE WHEN ... THEN END`, `REPLACE`, `CAST` and `CONVERT` to be able to extract the data from `tblLoanView` directly by the means of the `sp_CreateMoodysExport` stored procedure without any need to write additional code, e.g. VBA code.

4.8 Amortization Profiles

The Amortization Profiles represent the principal payment schedule of the assets in the current pool (`tblLoanView`). This is a very central piece of information, relevant not only for a variety of tasks during the CDO structuring process such as estimation of diverse parameters on the notes and investor reporting, but also required by the Rating Agencies as well. Figure 7 below shows an example of an amortization profile for a 250.000,00 loan with maturity in 62 months:

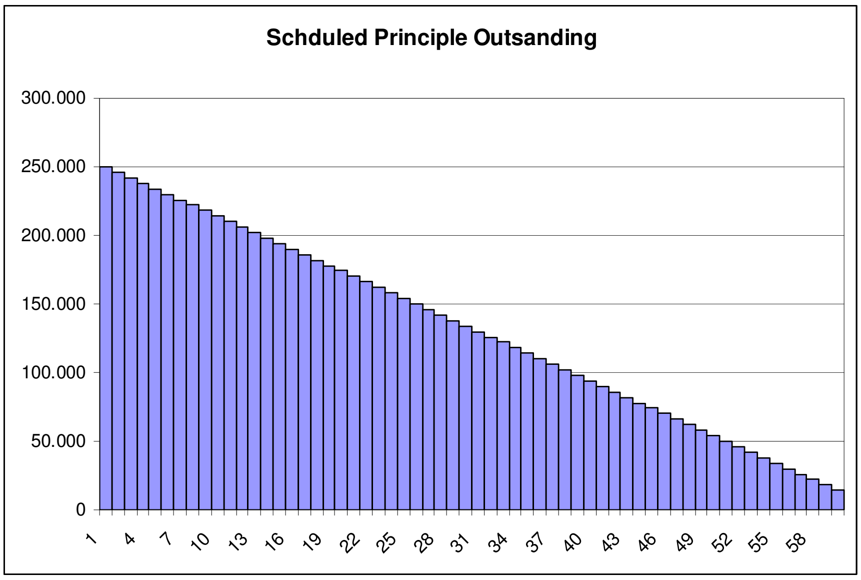


Figure 7: Typical loan repayment schedule with even principal payments

In this current project, in order to calculate and output such profiles the market risk data has to be evaluated, but instead of computing WAL (a single value) of an asset, its whole life repayment schedule has to be put on a time line axis. Let us consider the following example below – a 2,5 Mio CAD short term loan, whereby during the last 3 months there is no payment on the principal:

Market risk data - `tblTempImportWAL`					
`stichtag`	`kontonummer`	`kredit`	`währung`	`fälligkeit`	`cashflow`
30.06.2008	11111111	5	CAD	05.12.08	278.788,14
30.06.2008	11111111	5	CAD	04.01.09	278.788,14
30.06.2008	11111111	5	CAD	03.02.09	278.788,14
30.06.2008	11111111	5	CAD	05.03.09	278.788,14
30.06.2008	11111111	5	CAD	04.04.09	278.788,14
30.06.2008	11111111	5	CAD	04.05.09	278.788,14
30.06.2008	11111111	5	CAD	03.06.09	278.788,14
30.06.2008	11111111	5	CAD	04.09.09	548.483,02
…	*another asset, etc.*	…	…	…	*(sth. else)*

The task now is to create from all the lines above a dynamically generated output table. The table has to have as many columns (periods) as the number of periods of the latest maturing asset, whereby months when no payment is due are filled with 0. Note that the number of columns is not fixed, because it depends on the latest maturity of an asset in the current pool. The table should be such as:

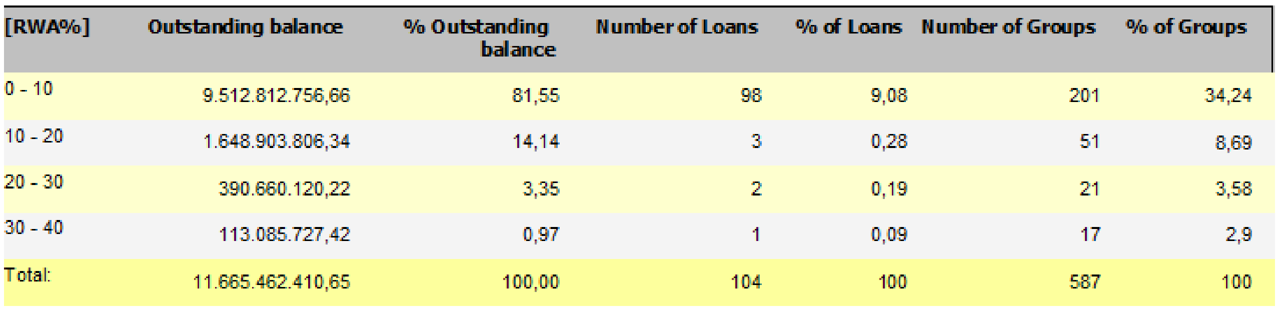
Output table - `tblAmortizationProfiles`

Att5	05.12.2008	04.01.2009	03.02.2009	05.03.2009	04.04.2009	04.05.2009	03.06.2009	03.07.2009	03.08.2009	03.09.2009
systemid* 11111111	278.788,14	278.788,14	278.788,14	278.788,14	278.788,14	278.788,14	278.788,14	0	0	548483,02
another asset etc	…	…	…	*(e.g. last payment here)*						
another asset etc	…	…	…	…	…	…	…	*(e.g. last payment here)*		

A single stored procedure – `funCreateAmortizationProfiles` takes care of the whole process. This function utilizes a number of more advanced SQL-features such as cursors (a control structure for successive iteration and processing of records in a result set), derived tables, control-of-flow structures, advanced type-casting and strings processing, pivoting, dynamic table regeneration and temporary tables as well as an additional table valued T-SQL function – `funGetAmortizationProfilesWALONLY()`. The above mentioned stored procedure essentially fill records in the table `tblAmortizationPerPeriod` `{Att5;varchar,PK|SteppedPeriod;int,PK|Amount;float|Period;int}` – the no-payment periods that have to be generated as well as the periods where money is due (that are calculated and returned from the aforementioned function). Then, at the end the whole table contents (i.e. all tuples) are pivoted and the desired outcome table is produced. A known drawback to this method is the performance in case eventually a great number of rows *and* columns have to be present in the output, but this function is practically only run after the portfolio pool is already cut (proper assets have already been selected) and therefore the performance does not suffer. To view the stored procedure's source code, please refer to Appendix 2.

4.9 Reports and Stratification Tables

This chapter presents in detail one of the more complicated functionalities in this project – the generation of stratification table reports. The two T-SQL stored procedures allow for parameterized generation of stratification tables on any given field of the `tblLoanView` table, i.e. any asset-specific field which has to be queried for over-stratification. One of the purposes, except for statistical overview is to facilitate the recognition and extent of default correlation among the assets in a number of criteria).



[RWA%]	Outstanding balance	% Outstanding balance	Number of Loans	% of Loans	Number of Groups	% of Groups
0 - 10	9.512.812.756,66	81,55	98	9,08	201	34,24
10 - 20	1.648.903.806,34	14,14	3	0,28	51	8,69
20 - 30	390.660.120,22	3,35	2	0,19	21	3,58
30 - 40	113.085.727,42	0,97	1	0,09	17	2,9
Total:	11.665.462.410,65	100,00	104	100	587	100

Figure 8: A stratification report on RWA percentage in a portfolio, generated by the application

First, a differentiation is made among stratification criteria – we have stratification **buckets** and stratification **min/max/step ranges**.

- **Buckets** are essentially stratification grouped by the values of a certain field, e.g. currency – information about number of loans, current principal outstanding and their concentration (percentage to total) is shown for each currency separately along with the number of groups and percent of groups

- **Min/max/step ranges** display the same information as bucket stratifications but instead of grouping by values the data is rather displayed grouped by stepped ranges of all the values of a numeric field

Two tables (for buckets and min/max stratifications) are provided that contain the field definitions for the report generation. The purpose is that these could be extended with new fields at any time and thus the application is more dynamic. Groupfield refers to the field in tblLoanView and step refers to the step in the min/max range within the field values (see figure 8 above, e.g. 0-10, 10-20, etc). Below are some examples from these tables:

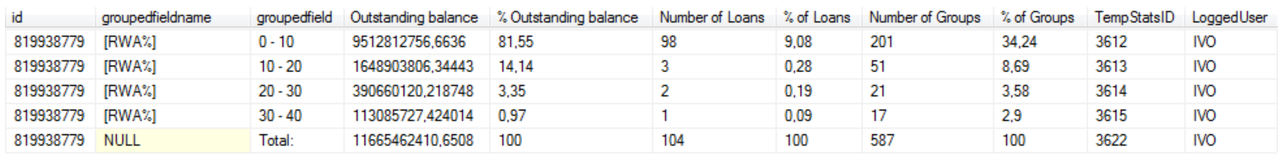
tblBucketStatsMinMax				
id	**min**	**step**	**max**	**groupfield**
2	0	10	100	[RWA%]
3	0	100000	17000000	RWA
4	0	1	11	WAL_Total

tblBucketStats	
id	**groupfield**
4	description
5	coutnry_code
6	incorporation_country_original
7	internal_rating_original
8	currency

Two list-boxes in the Reports/Statistics tab in *frmMain* display those tables respectively and provide the user with the ability for multiple selections on only those fields that the user would like to see in the generated report. After selecting the desired fields the VBA function *funFill_tblTempStatsAllReports*() is invoked. In order for this function to generate the actual reports it executes the following steps

- First it deletes all records in `tblTempStats` for the current user (field LoggedUser). This is done because many users are supposed to generate reports at the same time and due to the fact that the reports are initially prepared in a temporary table whose whole contents is then passed to the Microsoft Access Reports functionality (filtered again on LoggedUser), one needs to extract the currently logged Windows user and take it into account in the filter. The global VBA variable *strCurrentUserName* is used for passing as the LoggedUser parameter and it is initialized on application startup using a call on the Windows API function *GetUserNameA()* in advapi32.dll

- It iterates the selected fields in the list-boxes and calls (via the `exec` SQL-command) one of the two T-SQL stored procedures, `sp_GetStats` or `sp_GetStatsBuckets`. These procedures produce the table below and also add a "Total" footer. They take a number of parameters along with a random id that is later used to group the report. Below is the `tblTempStats` that was generated before the example report above was generated:



id	groupedfieldname	groupedfield	Outstanding balance	% Outstanding balance	Number of Loans	% of Loans	Number of Groups	% of Groups	TempStatsID	LoggedUser
819938779	[RWA%]	0 - 10	9512812756,6636	81,55	98	9,08	201	34,24	3612	IVO
819938779	[RWA%]	10 - 20	1648903806,34443	14,14	3	0,28	51	8,69	3613	IVO
819938779	[RWA%]	20 - 30	390660120,218748	3,35	2	0,19	21	3,58	3614	IVO
819938779	[RWA%]	30 - 40	113085727,424014	0,97	1	0,09	17	2,9	3615	IVO
819938779	NULL	Total:	11665462410,6508	100	104	100	587	100	3622	IVO

`sp_GetStatsBuckets` uses control-of-flow statements (`IF`, `ELSE`) together with a `WHILE`-loop to iterate through all steps (the step value is also passed as a parameter) and build an `INSERT`-statement for each stratification category using a **derived query** (MS SQL Server has the ability to create derived tables on the fly and then use these derived tables within a query). The `nvarchar` variable `nsql` is used to dynamically build the SQL string which is then executed by calling the integrated `EXEC sp_executesql @nsql` integrated stored procedure. Ultimately, the footer row is added by using the same technique.

5. Implementation Differences in Other SQL-Standards

As stated in section 3.2, the main reason behind choosing Microsoft Access in combination with Microsoft SQL Server was the ability to easily combine both by the means of an ADP-Project and the restrictions of additional software installations in the RZB (this thesis is part of the IT-practical training course – ITP). The most apparent choice for a different RDBMS would be either PostgreSQL, or Oracle – both very advanced and commonly used RDBMS. A migration of the current project to either systems would be theoretically possible whereby the data model (table structure) could virtually be transferred seamlessly by means of a proper migration tool (due to the lack of any special "rules" on the tables such as domain values, master/detail relationship definitions between tables, i.e. all tables are currently 'independent' and consist of the basic MS SQL-Server data types). The T-SQL functions on the other hand, as well as some more complicated views would need rewriting in order to work in either Oracle or PostgreSQL.

A very extensive study by Troels Arvin [http://troels.arvin.dk/db/rdbms/] compares these popular RDBMS, whereby the most relevant differences that relate to this project are:

- Different implementation of the `TOP`-n results `SELECT`
- MS SQL Server and Oracle do not support boolean variables, PostgreSQL does
- Difference in strings handling: the `SUBSTRING`-function differs in MS SQL Server and PostgreSQL (parameters), Oracle uses `SUBSTR` instead; MS SQL Server needs `LTRIM` and `RTRIM` instead of the standard `TRIM`

used by the others; MS SQL Server concatenates with a ‘+’ instead of the standard ‘||’

- IDENTITY is not supported in Oracle and PostgreSQL as column property, thus triggers have to be generated for auto id generation

A viable option for migration of the project to a lesser known RDBMS would be Firebird – the open source spin-off in 2000 of the popular at the time InterBase 6.0 by Borland. It is a fully-fledged relational DBMS server with support for a number of features such as triggers, stored procedures, User Defined Functions (UDFs – external functions), referential integrity support and ACID-compliancy (Atomicity, Consistency, Isolation, Durability – prerequisites for reliable transactions defined by Jim Gray end of the 1970s). Additionally, the latest 2.1 Firebird version supports derived tables, procedural triggers recursive queries as well as `MERGE`-statements (simultaneous `INSERT` and `UPDATE` depending on conditions) [Firebird_Borrie]. Firebird is a relatively lightweight as it produces a single database file (incremental backups are also possible) and the server module is under 12 MB and runs on both Linux and Windows systems.

6. Outlook and Conclusions

The conclusion from this bachelor's thesis and the software project that supplements it is twofold. First of all, collateralized debt obligations are very complex structures as outlined in Chapters 1 and 2. Secondly, in order to prepare a real-world securitization deal a lot of effort and proper data design is neccessary which also demands in-depth knowledge of database design. Furthermore, in order to make effective use of the data model, theoretical and practical knowledge of SQL is required as two complex stored procedures are demonstrated in Appendix 1 and 2. Lastly, the current outlook is closesly related to the whole Section 2.3 which gives an overview of the possible causes of the current financial crisis in regard to CDOs and ABS. The latest information as of Feb 2009 suggests that further regulation and restructuring of the structured finance industry is expected. Hence it is very probable that future deals would substantially differ than those up until today, thus the ability and expertise to quickly adapt the data models that represent the IT-'mirror' of those deals is of huge importance.

7. Bibliography

Below is a list of the literature and online sources referenced herein:

[HBS_Nov_08] Coval, Joshua D./Jurek, Jakub/Stafford, Erik. (2008) The Economics of Structured Finance, Harvard Business School Working Paper 09-060. http://www.hbs.edu/research/pdf/09-060.pdf

[CDO_SA] Lucas, Douglas J./Goodman, Laurie S./Fabozzi, Frank J. (2006): Collateralized Debt Obligations, Structures and Analysis, 2nd Edition, John Wiley &Sons, Inc., Hoboken, Chapter 1. ISBN 0-471-71887-4

[Dev_CDO] Manning, Rebecca J./ Lucas, Douglas J./Goodman, Laurie S./Fabozzi, Frank J. (2007): Developments in Collateralized Debt Obligations. New Products and Insights, John Wiley &Sons Inc., Page 6. ISBN 0-470-13554-9

[Br_Sch_05] Braun, Hendryk/Schmidt, Daniel (2005): Qualitative und quantitative Faktoren bei der Analyse von Asset-Backed-Securities. In: Gruber, Josef/Gruber, Walter/Braun, Hendryk (Hrsg.): Praktiker-Handbuch Asset-Backed-Securities und Kreditderivate. Strukturen, Preisbildung, Anwendungsmöglichkeiten, aufsichtliche Behandlung. Schäffer-Poeschel, Stuttgart, Page 191-211. ISBN: 3-791-02300-4

[CFA_2008] CFA Institute (2008). Derivatives and Alternative Investments, Page G-11, Boston: Pearson Custom Publishing, ISBN 0-536-34228-8.

[Raiff_Res] Raiffeisen Research (2007): Collateralized Debt Obligations, Synthetic CDOs – Formula One of Structured Credit, Page 2 http://www.rzb.at/eBusiness/services/resources/media/1023296711504-1023296711595_1025308884300_1027623427364_1027623627472-369678458155597420-1-NA-DE-123.pdf

[Br_05] Braun, Hendryk (2005): Klassifizierung von Asset-Backed-Securites. In: Gruber, Josef/Gruber, Walter/Braun, Hendryk (Hrsg.): Praktiker-Handbuch Asset-Backed-Securities und Kreditderivate. Strukturen, Preisbildung, Anwendungsmöglichkeiten, aufsichtliche Behandlung. Schäffer-Poeschel, Stuttgart, Page 61-77. ISBN: 3-791-02300-4

[SIFMA_08] Securities Industry and Financial Markets Association- http://www.sifma.org/research/pdf/SIFMA_CDOIssuanceData2008.pdf

[CPM_2003] Smithson, Charles W., Credit Portfolio Management (2003). , John Wiley &Sons, Inc., Hoboken, Chapters 1, 2 and 3. ISBN 0-471-32415-9

[BIS_1] Bank for International Settlements, Overview of the New Basel Capital Accord BIS Consultative Document, April 2003 - http://www.bis.org/bcbs/cp3ov.pdf

[BIS_2] Bank for International Settlements, Basel Committee on Banking Supervision, http://www.bis.org/publ/bcbs107a.pdf

[BIS_3] Bank for International Settlements, Basel Committee on Banking Supervision, Basel II: Revised international capital framework (BCBS), http://www.bis.org/publ/bcbsca.htm

[MSDN_01] SQL Server 2008 Books Online (October 2008), Using PIVOT and UNPIVOT, http://msdn.microsoft.com/en-us/library/ms177410.aspx

[FITCH_1] The Fitch Portfolio Credit Model, Fitch Inc. http://www.fitchratings.com/jsp/corporate/ToolsAndModels.faces?context=2&detail=117

[Chan-Lau_Lu_2006] Jorge A. Chan-Lau, Yinqiu Lu. Idiosyncratic and Systemic Risk in the European Corporate Sector: A CDO Perspective, April 2006. International Monetary Fund ,WP/06/107 Working Paper www.imf.org/external/pubs/ft/wp/2006/wp06107.pdf

[Calomiris_Nov_08] Charles Calomiris. The Subprime Turmoil: What's Old, What's New, and What's Next, 9th Jacques Polak Annual Research Conference November13 - 14, 2008. Washington, DC

[IMF_Rajan_2006] Raghuram G. Rajan. Monetary Policy and Incentives, Address at the Bank of Spain Conference on Central Banks in the 21st Century, June 8, 2006

[Ioannidou_et_al_07] Vasso P. Ioannidou, Steven Ongena and Jose Luis Peydró. The impact of short-term interest rates on risk-taking: hard evidence, 12 Oct 2007 http://www.voxeu.org/index.php?q=node/641

[WSJ_Feb_08] Manas Chakravarty, The Wall Street Journal, Remarkable prescience of Raghuram Rajan Feb 2, 2008

[Orrick_08] Orrick, Herrington & Sutcliffe LLP. ECB Tightens Collateral Criteria for Repo Facilities, Sep 4, 2008 http://www.orrick.com/fileupload/1459.pdf

[ECB_Nov_08] European Central Bank. The Implementation of Monetary Policy in the Euro Area, November 2008 http://www.ecb.eu/pub/pdf/other/gendoc2008en.pdf?310f9ae8a4ca80b4b97c43f9cb5414d8

[Haught_Upsize_SQL] Haught Dan, Strategic Initiatives: Evolving Microsoft Access Applications to Microsoft SQL Server. FMS Professional Solutions http://www.fmsinc.com/FMSUpsize/docs/EvolvingMicrosoftAccessApplications.pdf

[MSDN_02] Understanding a Microsoft Access ADP Project, http://office.microsoft.com/en-us/access/HP052731031033.aspx

[MDAC_Roadmap] Shirolkar, Henry et al, Data Access Technologies Road Map, http://msdn.microsoft.com/en-us/library/ms810810.aspx, Dec 2008

[Zelders _TSQL] Zelders Xander, SQL Server Fact sheet, April 2008 http://www.dotnet4all.com/snippets/factsheet%20SQL%20Server.pdf

[MSDN_JET] Description of the new features that are included in Microsoft Jet 4.0, MSDN, http://support.microsoft.com/kb/275561/en

8. Appendix 1: Example of a T-SQL Stored Procedure

One of the targets of this project is to leverage as much as possible on Microsoft SQL Server's programmability. Below is an example of one of the more complicated T-SQL stored procedures in the project – `sp_GetStatsBuckets`:

```
ALTER PROCEDURE [dbo].[sp_GetStatsBuckets](@usedfield varchar(255), @randomid varchar(255), @min
bigint, @max bigint, @step bigint, @loggeduser varchar(6))
AS
BEGIN
        -- SET NOCOUNT ON added to prevent extra result sets from interfering with SELECTs
        SET NOCOUNT ON;
        DECLARE @sql varchar(2000)
        DECLARE @nsql nvarchar(2000)
        DECLARE @stepismax bit
        DECLARE @i bigint
        SET @i = @min
        SET @stepismax = 0
        WHILE (@i <= @max) -- Loop through the steps
        BEGIN
                -- I am using a derived query to produce a SELECT statement
                SET @sql = 'INSERT INTO tblTempStats '
                SET @sql = @sql + '(id, loggeduser, groupedfieldname, groupedfield, [Outstanding
                balance], [% Outstanding balance], [Number of Loans], [% of Loans], [Number of
                Groups], [% of Groups])'
                IF (@stepismax = 1)
                        SET @sql = @sql + ' SELECT ' + @randomid + ' as id, ''' + @loggeduser + '''
                        as loggeduser, ''' + @usedfield + ''' as groupedfieldname, ''>'+ CAST((@i +
                        @step) as varchar) + ''''
                ELSE
                        SET @sql = @sql + ' SELECT ' + @randomid + ' as id, ''' + @loggeduser + '''
                        as loggeduser, ''' + @usedfield + ''' as groupedfieldname, ''' + CAST(@i as
                        varchar) + ' - '+ CAST((@i + @step) as varchar) + ''''
                SET @sql = @sql + ' as groupfield, sum(ent.sumtotal) as [Outstanding Balance], '
                SET @sql = @sql + ' round(sum(ent.sumtotal) / cast((select sum(outstanding) as sumo
                from tblloanview) as float) * 100,2) as [% Outstanding balance], count(*) as [Number
                of Loans],  round(count(*) / cast((select count(*) as cnt from tblloanview) as
                float) * 100,2) as [% of Loans], max([Number of Groups]) as [Number of Groups],
                max([% of Groups] ) as [% of Groups]'
                SET @sql = @sql + ' from ( '
                SET @sql = @sql + ' SELECT  sum(outstanding) as sumtotal,  count(distinct
                entity_group) as [Number of Groups], round(count(distinct entity_group) /
                cast((select count(distinct entity_group) as distgroups from tblloanview) as float)
                * 100,2) as [% of Groups] '
                SET @sql = @sql + ' from tblloanview group by ' + @usedfield
                if (@stepismax = 1)
                        SET @sql = @sql + ' having sum(' + @usedfield + ') > ' + cast((@i + @step) as
                        varchar) + ' ) ent'
                ELSE
                        SET @sql = @sql + ' having sum(' + @usedfield + ') between ' + cast(@i as
                        varchar) + ' AND ' + cast((@i + @step)  as varchar) + ' ) ent'
                SET @nsql = cast(@sql as nvarchar(2555))
                EXEC sp_executesql @nsql
                SET @i = @i + @step
                IF (@i + @step = @max)
                        SET @stepismax = 1
                IF (@i + @step > @max)
                BREAK
          ELSE
                CONTINUE
       END
       -- Remove rows where values are NULL
       SET @sql = 'DELETE FROM tblTempStats WHERE ID = ' + @randomid + ' AND [Outstanding balance]
       IS NULL'
       SET @nsql = cast(@sql as nvarchar(2555))
       EXEC sp_executesql @nsql
       -- Insert footer line
       SET @sql = 'INSERT INTO tblTempStats'
       SET @sql = @sql + ' (loggeduser, groupedfield, '
       SET @sql = @sql + ' id, [Outstanding balance], [Number of Loans], [Number of Groups], [%
       Outstanding balance], [% of Loans], [% of Groups])'
       SET @sql = @sql + ' SELECT ''' + @loggeduser + ''' as loggeduser, ''Total:'' as groupfield,
       *, (select count(distinct entity_group) as distgroups from tblloanview) as [Number of
       Groups], 100 as [% Outstanding balance], 100 as [% of Loans], 100 as [% of Groups] '
       SET @sql = @sql + 'FROM funGetStatsTotals (' + @randomid + ');'
       SET @nsql = cast(@sql as nvarchar(1000))
       EXEC sp_executesql @nsql
END
```

9. Appendix 2: Amortization Profiles T-SQL Functions

Belows is the source code for the stored procedure and table valued function used in Section 4.8

9.1 *funCreateAmortizationProfiles*

This is the main procedure for generation of amortization profiles:

```
ALTER PROCEDURE [dbo].[funCreateAmortizationProfiles](@iStep int)

AS
BEGIN
       SET NOCOUNT ON;
       DELETE FROM tblAmortizationPerPeriod;
       DECLARE @ymax int
       DECLARE  @iRowId  bigint
       DECLARE @counter int
       DECLARE @faellig float
       DECLARE @periode int
       DECLARE @jahr int
       DECLARE @monat int
       DECLARE @att nvarchar (2000)
       DECLARE @firstperiod int,
                      @firstyear int,
                      @firstmonth int,
                      @curperiod int,
                      @curmonth int,
                      @lastmonthoffset int
       DECLARE @strcurmonth varchar(2)
       -- declare the cursor
       DECLARE LoanDetail CURSOR FOR
```

```
SELECT DISTINCT    kontowhg
FROM      vwWALAmortizationPerPeriodFull
OPEN LoanDetail
FETCH LoanDetail INTO @iRowId
-- start the main processing loop.
WHILE @@Fetch_Status = 0
   BEGIN
        -- detailed row-by-row processing
        SELECT DISTINCT @att = att5 from vwWALAmortizationPerPeriodFull WHERE kontowhg =
                @iRowID;
        SET @att = (REPLICATE('0', 14-LEN(cast(@iRowID as varchar))) + cast(@iRowID as
                varchar) + '*SOME_SYSTEMID')
        SELECT @ymax = MAX(jahr) FROM tblWAL WHERE Att5 =   @att  ;
        SELECT @lastmonthoffset = MAX(periode) FROM tblWAL WHERE Att5 =   @att  ;
        SET @lastmonthoffset = cast(right(cast(@lastmonthoffset as varchar), 2) as int)
        SELECT @firstperiod = min(periode) from vwWALAmortizationPerPeriodFull WHERE Att5 =
                @att  ;
        SET @firstyear = cast(left(cast(@firstperiod as varchar), 4) as int);
        SET @firstmonth = cast(right(cast(@firstperiod as varchar), 2) as int);
        SET @lastmonthoffset =        @lastmonthoffset -  @firstmonth
        SET @curmonth = @firstmonth;
        SET @counter = 0;
        WHILE @counter < (@iStep * @ymax) +  @lastmonthoffset + 1
        BEGIN
                IF @curmonth < 10
                        SET @strcurmonth = '0' + cast(@curmonth as varchar)
                ELSE
                        SET @strcurmonth = cast(@curmonth as varchar);
                        SET @curperiod = cast(cast(@firstyear + round((@counter + @firstmonth
                                - 1) / 12, 0) as varchar) + @strcurmonth as int);
                INSERT INTO tblAmortizationPerPeriod VALUES (@att, @counter + 1, NULL,
                        @curperiod);
                SET @counter = @counter + 1;
                SET @curmonth = @curmonth + 1;
                IF @curmonth = 13
                SET @curmonth = 1;
   END
   FETCH LoanDetail INTO @iRowId
   END
CLOSE LoanDetail
DEALLOCATE LoanDetail
UPDATE tblAmortizationPerPeriod
        SET Amount = faellig
FROM tblAmortizationPerPeriod
INNER JOIN vwWALAmortizationPerPeriodFull ON tblAmortizationPerPeriod.Att5 =
        vwWALAmortizationPerPeriodFull.Att5
        AND tblAmortizationPerPeriod.Period = vwWALAmortizationPerPeriodFull.periode;
INSERT INTO tblAmortizationPerPeriod
SELECT * FROM funGetAmortizationProfilesWALONLY();
--pivoting
DROP TABLE tblAmortizationProfiles;
CREATE TABLE #tempPIVOT (
        Att5 varchar(50),
        Variable int,
        VaribleValue float
)
INSERT INTO #tempPIVOT
SELECT Att5, Period,  Amount FROM tblAmortizationPerPeriod
ORDER BY period
DECLARE @columns VARCHAR(8000)
SELECT @columns = COALESCE(@columns + ',[' + cast(Variable as varchar) + ']',
                                      '[' + cast(Variable as varchar)+ ']')
FROM    #tempPIVOT
GROUP BY Variable
ORDER BY Variable
DECLARE @query VARCHAR(8000)
SET @query = 'SELECT *
INTO tblAmortizationProfiles
FROM #tempPIVOT
PIVOT (
 MAX(VaribleValue)
 FOR [Variable]
```

```
         IN (' + @columns + ')
         )
         AS p'
        EXECUTE(@query)
        DROP TABLE #tempPIVOT;
        RETURN
END
```

9.2 *funGetAmortizationProfilesWALONLY()*

The stored procedure in the previous section refers to this table valued T-SQL function that returns a table as its result:

```
ALTER FUNCTION [dbo].[funGetAmortizationProfilesWALONLY] ()
RETURNS @resulttable TABLE (
        Att5 varchar(50) primary key,
        SteppedPeriod int,
        Amount float,
        Period int
)
AS
BEGIN
        DECLARE @partkey_year int,
                @partkey_month int;
        SELECT @partkey_year = partkey_year, @partkey_month = partkey_month FROM
                vwLoanViewPartitionKeyDate;
        INSERT INTO @resulttable (Att5, SteppedPeriod, Amount, Period)
        SELECT nestedtbl.exposure_attribute_5 as Att5, nestedtbl.SteppedPeriod, nestedtbl.Amount,
                cast(nestedtbl.endyear_final as varchar) + cast(nestedtbl.endmonth_final as varchar)
        as Period FROM
        (
                SELECT DISTINCT exposure_attribute_5, 1 as SteppedPeriod, Outstanding as Amount,
                ((cast(wal * 12 as int) % 12) + @partkey_month) as endmonth_original,
                CASE WHEN ((cast(wal * 12 as int) % 12) + @partkey_month) > 12 then
                (cast(wal * 12 as int) / 12) + 1 + @partkey_year
                WHEN((cast(wal * 12 as int) % 12) + @partkey_month) <= 12 then
                (cast(wal * 12 as int) / 12) + @partkey_year
                END as endyear_final,
                CASE WHEN((cast(wal * 12 as int) % 12) + @partkey_month) > 12 then
                ((cast(wal * 12 as int) % 12) + @partkey_month) % 12
                WHEN((cast(wal * 12 as int) % 12) + @partkey_month) <= 12 then
                (cast(wal * 12 as int) % 12) + @partkey_month
                END as endmonth_final
                FROM tblLoanView WHERE not tblLoanView.wal is null
        and exposure_attribute_5 not in (select att5 from tblAmortizationPerPeriod)
        ) nestedtbl

RETURN
END
```