Sigma Systems, Inc.

Overview Data Model Kuali Student Accounts Receivables Management (KSA-RM) System

Sigma Systems
March 2012

Change Log

Author	Date	Changes
Paul	3/2/2012	Added a changelog.
Paul	3/2/2012	Per discussion with Michael, cleaned up the KSSA_CREDIT_PERMISSION table. ALLOWABLE_DEBIT_TYPE_ID. Cleaned up KSSA_ALLOCATION to create an autonumber primary key, and made ACCOUNT_ID -> ACCOUNT FLAG ->FLAG_TYPE, KSSA_INFORMATION.FLAG_ID became FLAG_TYPE_ID_FK
Paul	3/7/2012	Cleared formatting for data elements. Simplified layout.
Paul	3/7/2012	Did the major part of the account table.
Paul	3/12/2012	Cleaned up account tables.
Paul	3/12/2012	Standardized FRIENDLY_NAME->NAME. All types now have DESCRIPTION instead of INFORMATION. Added DESCRIPTION where missing to standardize _TYPE tables. For easier subclassing. Removed SIMPLE_TYPE (unused). CURRENCY.ID changed to an INT.
Paul	3/14/2012	Removed problematic _FK issue in ACNT table.

Contents

Data Models for the Transaction and Associated Classes

General Notes

- All tables are preceded with KSSA (Kuali Student – Student Accounts) to prevent table name clashes. This follows the rules established by Rice.
- Appropriate contractions are used as suggested on the Rice and KS wiki, including but not limited to TRN-Transaction and ID- Identifier, AMNT- amount. Foreign keys, where they apply to our tables, are tagged with _FK.
- These data models have been designed to support the permanence layer of the Transaction class, its children and its associated classes. Only the permanence layer will access this data structure directly.

There are a number of reused data structures that are not repeated in the document. It is assumed that they are understood.

CREATOR_ID and EDITOR_ID are the identifiers for the entity who creates, and if appropriate, subsequently edits a record. LAST_UPDATE is the date stamp for the last alteration to the structure. LEVEL refers to a general access level that is defined by the institution. It is stored as a plain INT, and a user must have a LEVEL equal to or greater than the LEVEL of the referenced information to be able to view it. Few levels are expected in reality (as roles exist to give much more granular controls) but an example might be 0 for Student (this is expected) 1 for external staff (departments, etc) 2 for internal bursar staff (and default for memos, etc.) and 3 for high-level employees in the bursar's office.

As all tables begin KSSA_, the prefix is not reused in the descriptions.

Debit Type

Generic

This model underlies the DebitType object, which is an associated class with the Debit class. Every Debit HAS_A DebitType. Note that DEBIT_TYPE and its relations are used to create a permanent store for the class DebitType. Although a similar subclass, CreditType exists, they are stored in different table structures.

Most often, transactions are categorized using a preset numbering system, (for example, TUT*** are tuition codes, etc.) However, there are times when the transaction codes do not permit flexible categorization for certain reporting purposes. Tags are an optional way to allow control over categories of transactions.

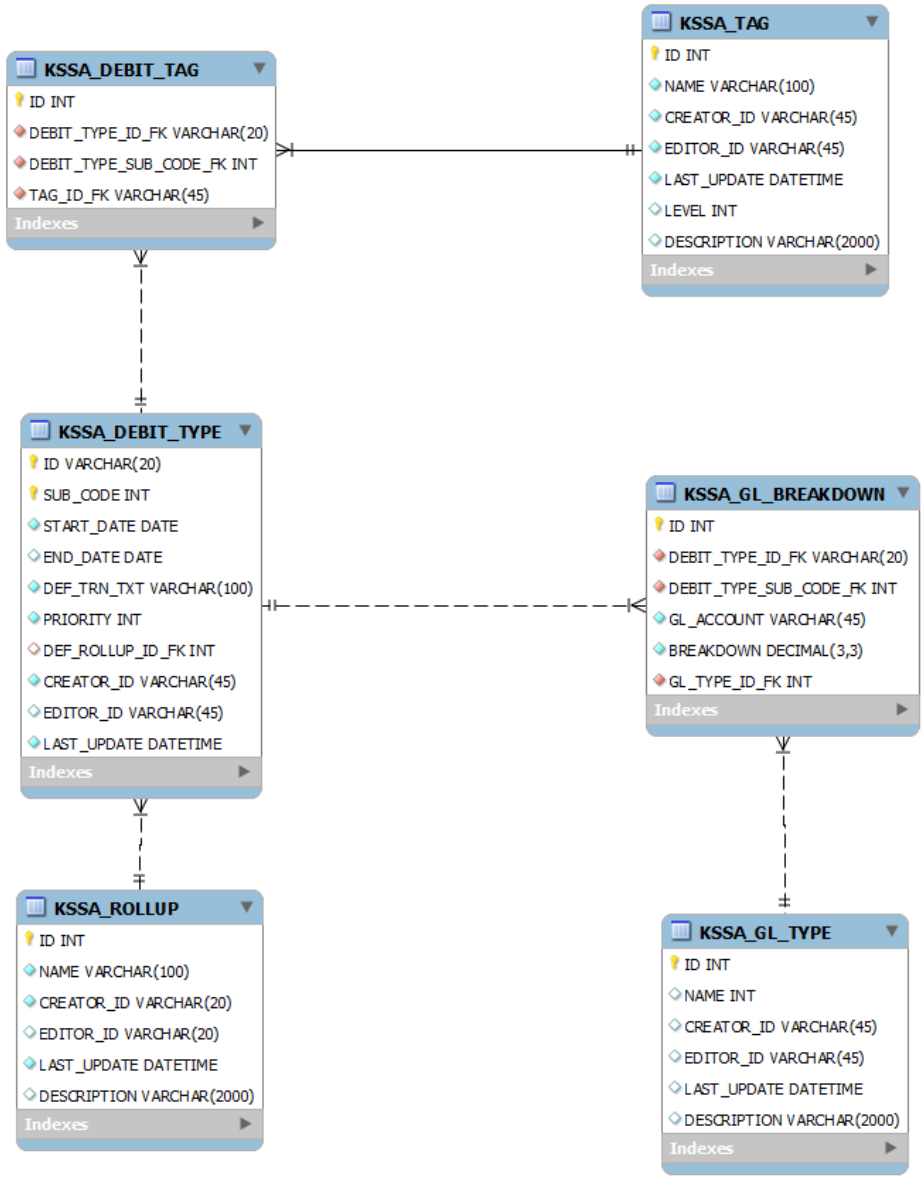

Table: DEBIT_TYPE
(Central definition of a debit type)

ID	Debit type identifier.
SUB_CODE	As debit types can change over time, the SUB_CODE field can be used to look up the specific details of a debit type at a certain period in time.
START_DATE	Mandatory field, defining when the debit type came into existence. By default, it is the date that the debit type was created.
END_DATE	Optional field. If null, it is the current debit type.
DEFAULT_TRN_TXT	Default text for this debit as shown on the statement. When a

	transaction is created, the default text is taken from this field, unless otherwise specified in the transaction.
PRIORITY	This field is used by payment application. As a general rule, higher priority debits will be paid before lower priority debits. Equal priority debits are paid off FIFO.
DEF_ROLLUP_ID_FK	Links the transaction to a default rollup. This can be overridden in the transaction. This foreign key is the primary key in the ROLLUP table.

Table: GL_BREAKDOWN

(Breakdown table to the general ledger.)

ID	Primary key autonumber identifier for the GL_BREAKDOWN table.
DEBIT_TYPE_ID_FK/ DEBIT_TYPE_SUB_CODE_FK	These foreign keys reference the DEBIT_TYPE table and identify a specific debit type at a specific period in time.
ACNT_TYPE_ID_FK	Different general ledger accounts can be implicated by different account types. The system can alter the general ledger account allocations based on the following account types: Direct Charge (ACD), Sponsor (ACS) and payment billing (PB). Friendly names and UI tips can be referenced from the ACNT_TYPE table.
GL_ACCOUNT	A single general ledger account.
GL_BREAKDOWN	If the debit type maps to a single general ledger account, then this field is 0. Where a debit type maps to a number of general ledger accounts, then the percentage breakdown for each is store in this field. Then final account in the list is given a breakdown 0, which means “allocate the remainder to this account”. An example is given below.
GL_TYPE_ID_FK	By default this is 0. This field allows for other ‘types’ of breakdowns to be defined, and can be referenced through the rules engine. For example, if a school writes off its transactions to a different set of GL accounts, then type 1 could be defined as the write-off type, and could be referenced here. Type 0 is reserved, and is always used for the standard transaction breakdown.

General Ledger Breakdown

Where schools divide single transactions over multiple general ledger accounts, these will be listed in this table, with BREAKDOWN amounts to spread the payment. In this case, percentages can be allocated, and there will be one "bucket" account, which receives the remainder of the funds. This prevents problems with fractional currency being unallocated or over allocated or under allocated. For example, a transaction that divides into two general ledger accounts as a 50/50 split would be defined as:
ACCOUNT 1 – 50%
ACCOUNT 2 – 0 (Bucket account)

Therefore a $100 transaction would divide as:
50% of $100 = $50. ACCOUNT 1 received $50
ACCOUNT 2 gets the remainder, therefore $100-$50 = $50 to account 2.

For a $99.99 transaction
50% of $99.99 = $49.995. With a rounding up, ACCOUNT 1 would be credited $50.00
ACCOUNT 2 would be credited with $99.99-$50 = $49.99

Table: GL_BREAKDOWN_TYPE

Defines rollups within the system.

ID	Primary autonumber key, autonumber. By default, 0 is the standard type.
NAME	Name of the GL breakdown type.
DESCRIPTION	Longer description available upon inspection.

Table: TAG

(General storage of tags within the system. Tags may be applied to both debit and credit types.)

ID	Autonumber primary key for the tag table.
NAME	Short, plain text name of the tag. For example "Tuition", "Books and Supplies".
LEVEL	As defined in the introduction, levels define who can or cannot see a certain item. The higher a user's level, the more the can see. The higher an item's level, the fewer users can see it. For example, tags such as "Tuition" may be visible to all users (level = 0) whereas certain tags might have a higher level to make them less visible to students and other staff.
DESCRIPTION	This is a general field that is used in UX presentment to give further information about the item. In this case, for example, if there were a tag of 1098T (a US tax form that reports tuition) and it were visible to the student, the student would be able to click on the tag and find out more about what it means that certain transactions were tagged as 1098T.

Table: DEBIT_TAG (|| CREDIT_TAG)

(Simple association table of DEBIT_TYPES to TAG.)

ID	Primary key, autonumber field for the association.
DEBIT_TYPE_ID_FK, DEBIT_TYPE_SUB_CODE_FK	Foreign keys of the DEBIT_TYPE table that associate the DEBIT_TYPE with the TAG
TAG_ID_FK	Foreign key for the TAG to associate the DEBIT_TYPE with the TAG.

Table: ROLLUP

Defines rollups within the system.

ID	Primary autonumber key, autonumber.
NAME	Name of the rollup as displayed on the UI
DESCRIPTION	Longer description available upon inspection.

Credit Type

Generic

This model underlies the CreditType class, which is an associated class with the Credit class. Every Credit object HAS_A CreditType.

Credit types are user configurable types that can be applied to an account. They are able to change over time as with DebitType however, it is expected that they will be more stable, and far less numerous than DebitType. Examples of credit types would be cash, credit card, check, financial aid, etc. There may be a need to handle different types of cash payments differently, so it is envisaged that many different credit types might be created.

Note that TAG, ROLLUP, and CREDIT_TAG are defined in the DEBIT_TYPE model. They are used identically here.

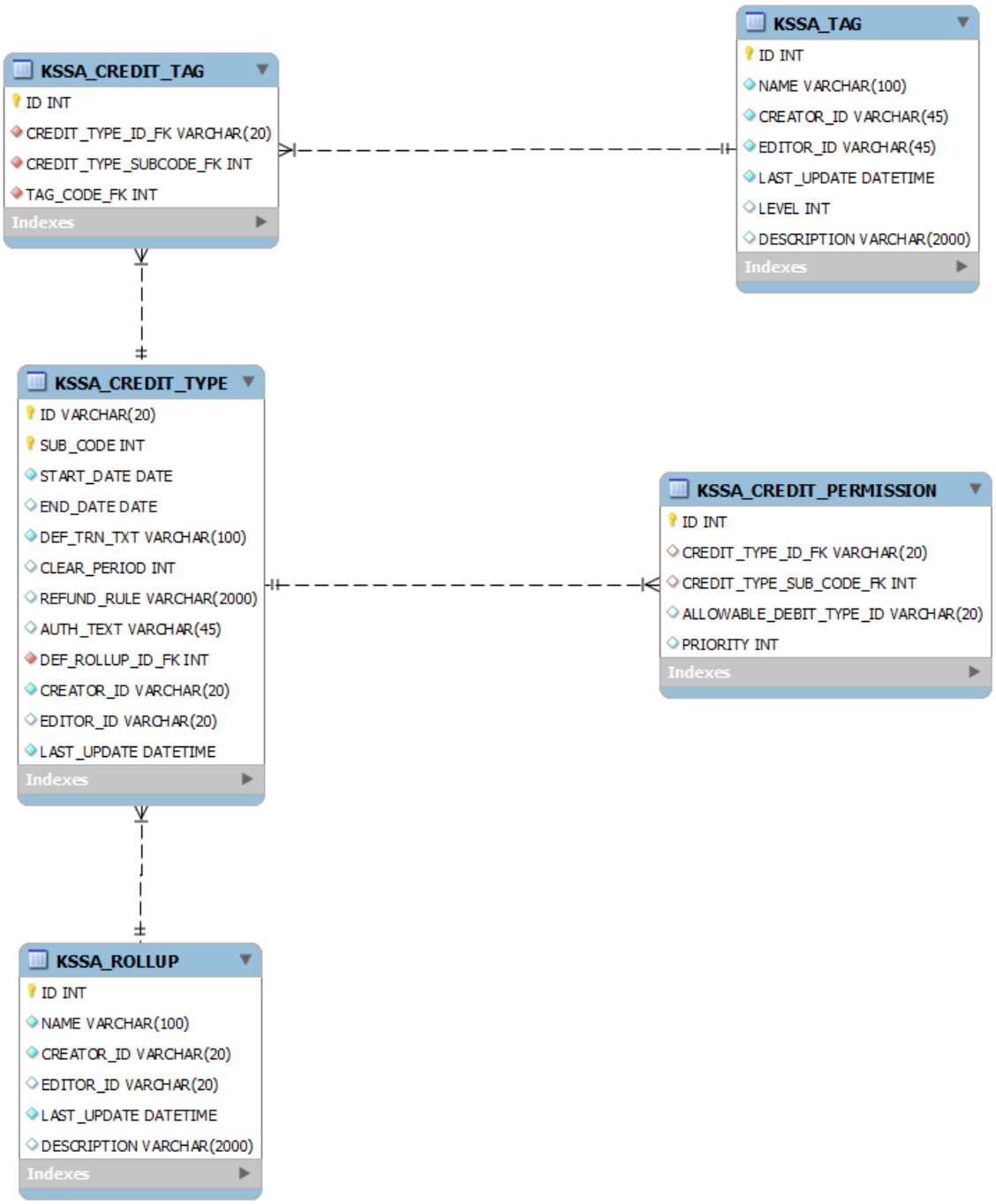

Table: CREDIT_TYPE

(General storage point for credit types.)

ID	Primary key for the credit type.
SUB_CODE	As debit types can change over time so too can credit types. The SUB_CODE field can be used to look up the specific details of a debit type at a certain period in time.

START_DATE	Mandatory field, defining when the credit type came into existence. By default, it is the date that the credit type was created.
END_DATE	Optional field. If null, it is the current credit type.
DEFAULT_TRN_TXT	Default text for this credit as shown on the statement. When a transaction is created, the default text is taken from this field, unless otherwise specified in the transaction.
CLEAR_PERIOD	A time period, specified in days, after which the payment is considered to be 'cleared'. For example, an institution may implement a 10-day hold on check payments.
REFUND_RULE	Currently just a text placeholder for a way of encoding this rule. It is only applicable if the isRefundable flag is true in the credit object. Some use cases are listed below. Note that the general REFUND_RULE for a CREDIT_TYPE can be overridden in the TRANSACTION table. If TRANSACTION.REFUND_RULE = null, then CREDIT_TYPE.REFUND_RULE is used. Otherwise TRANSACTION.REFUND_RULE Is followed. TRANSACTION.REFUND_RULE permits the same rules as CREDIT_TYPE.REFUND_RULE, as well as also permitting a refund to another KSA account. This field is still in development.
AUTH_TXT	This is a friendly text field to assist the user when processing payments. It stipulates the expected reference that the payment will be. In the case of a credit card, for example, the authorization code from the credit card company might be the reference stored for the transaction. In the case of a check, the bank information and the check number might be stored. This is institution and payment specific.
DEF_ROLLUP_ID_FK	Links the transaction to a default rollup. This can be overridden in the transaction. This foreign key is the primary key in the ROLLUP table.

Refund Rule Use Cases

The refund rule allows for the following scenarios:

- A refund in cash or equivalent may be issued on this amount after the clearing period (in the case of cash, the clearing period would be 0) (example, cash, checks)
- A refund to the original source may be made after the clearing period. (example, credit cards)
- A refund to the original source may be made for a specified period of time, after which a refund in cash or equivalent may be made. (for example, credit cards.)

There may be other uncaptured refund rules, therefore at this point, this field is not yet finalized.

Table: CREDIT_PERMISSION

(Stores the relationship between credit types (generally

	payments) to debit types (generally charges). Used by payment application to decide what charges can be paid by what payments. A credit type may have many credit permissions. This table is primarily used by the payment application algorithm.)
ID	Autonumbered primary key.
CREDIT_TYPE_ID_FK, CREDIT_TYPE_SUB_CODE_FK	Identifies the exact credit type that is being referenced.
ALLOWABLE_DEBIT_TYPE_ID	This can either be a foreign key for a debit type, or a masked debit type. If a payment can be applied to any charge, then a wildcard would be in this field. Masking follows the basic SQL wildcard options.
PRIORITY	Priority states the priority of a credit to pay off a certain group of transactions. This is only used if the system needs to break a tie between Debits of the same priority. If two debits have the same priority and the credit is allowed to pay them both, it will pay off the higher priority codes first, before applying the remainder to the remaining codes.

Transaction

Generic

The transaction data model is used to store the discrete transactions within the KSA system. Every charge and payment (and deferment) is stored as a transaction.

TRANSACTION is identified by ID and lists all the headline information about any transaction. Every Transaction can be either a Credit or a Debit. A Credit can be a Deferment or a Payment, a Debit is a Charge. A debit HAS_A DEBIT_TYPE, a Credit HAS_A CREDIT_TYPE. For clarity, CREDIT_TYPE and DEBIT_TYPE are defined earlier.

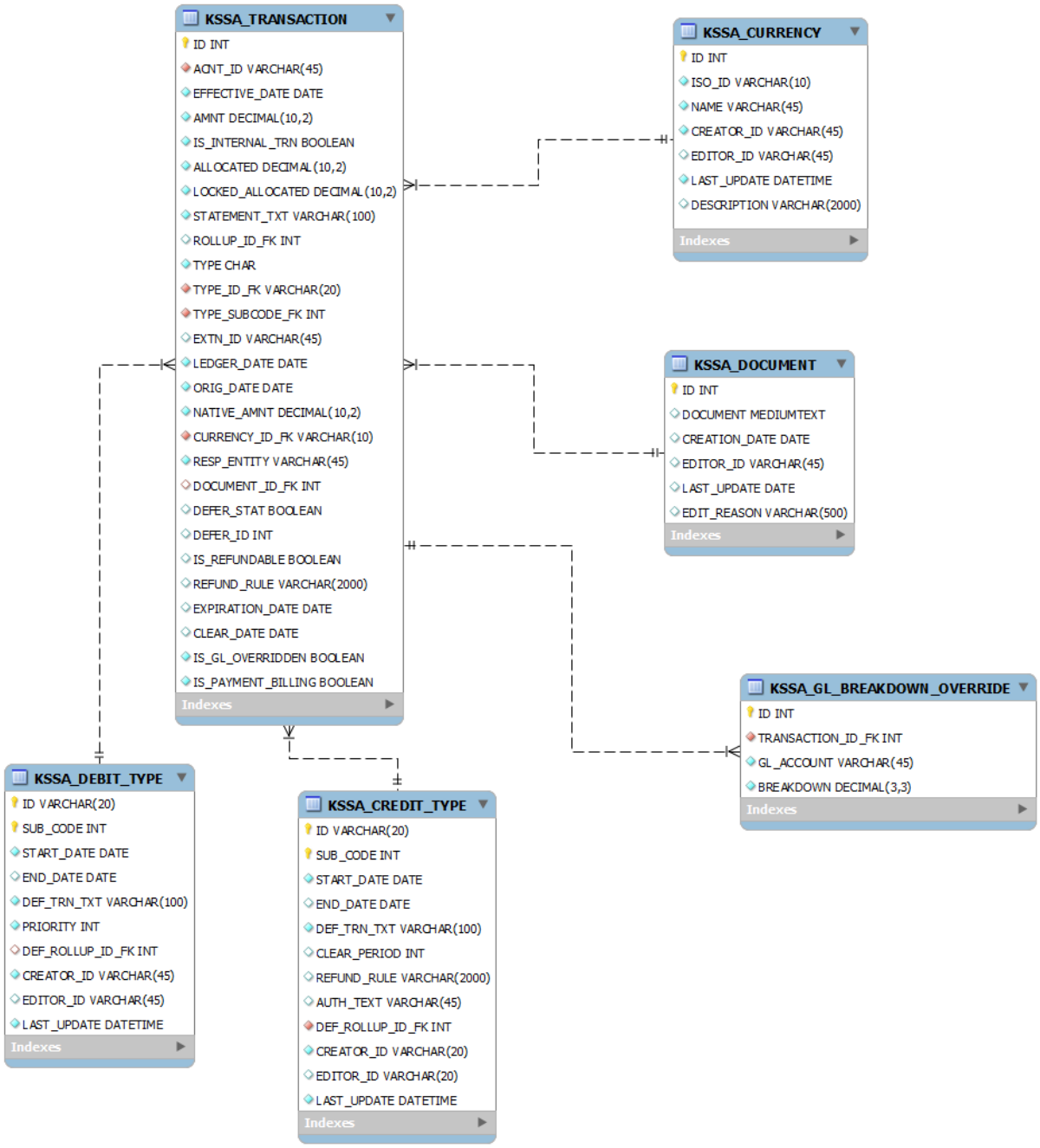

Table: TRANSACTION

(Stores transactional financial data.)

ID	Autonumbered primary key to make each transaction unique.
ACNT_ID	KSA account number with which this transaction is associated.
EFFECTIVE_DATE	This is the date that the transaction is considered "current" on the account. This is the date around which all processing is based. A transaction does not begin to 'age' until the this

	date is current. This date is also used in relation to the credit and debit types to define how the transaction behaves.
AMNT	This is the value of the transaction in the system specified currency. Note that although the decimalization is overkill for US dollars, the wider field is used to accommodate other currencies.
IS_INTERNAL_TRN	Boolean that dictates whether a transaction is considered "internal" or not. Internal transactions are generally not presented to customers.
ALLOCATED	The amount of currency) that is allocated. For a payment, it is the amount of the payment that is allocated to a charge. For a charge, it is the amount of the charge that has been paid.
LOCKED_ALLOCATED	This works the same as ALLOCATED, except this payment allocation cannot be de-allocated by the automatic payment allocation module. A CSR might choose to lock certain payments and charges together, or a school might choose to freeze all allocations from a previous period to stop the system from de-allocating earlier allocations.
STATEMENT_TXT	'Friendly text' for a transaction that is displayed to explain the purpose of the transaction. This is derived from the credit or debit type in the first instance, but might be overridden in an individual transaction.
ROLLUP_ID_FK	A field used to group similar transactions in to a rollup. For example, if a number of add/drops occur within a period, all of the charges and refunds might be given the ROLLUP_ID that refers to "Tuition". Then, on the initial view of the statement, the word "Tuition" would appear with a net of the values. Further inspection would allow the full list of transactions that make up the rollup appear. The ROLLUP table is defined above.
TYPE	References the type of transaction; acceptable values are TCP for TRANSACTION->CREDIT->PAYMENT, and TCD for TRANSACTION->CREDIT->DEFERMENT and TDC for TRANSACTION->DEBIT->CHARGE.
TYPE_ID_FK	Links to the DEBIT_TYPE or CREDIT_TYPE definitions, depending on the value of TYPE.
TYPE_SUBCODE_FK	This links to the sub code of the DEBIT_TYPE or CREDIT_TYPE definitions, depending on the value of TYPE. The subcode is derived from the type id and the effective date at the point of instantiation.
EXTN_ID	External transaction identifier. For external systems that generate a transaction identifier will populate this with the foreign identifier. If the transaction comes via batch, then the batch id will be here. For payments, the expected authorization code will be inserted here. For example, a credit card payment might use the authorization code as the EXTN_ID.
LEDGER_DATE	The date that the transaction was entered in to the ledger within the KSA-RM system. The "ledger" referred to is the list

	of transactions relating to an account that is held within the KSA system. It does not refer to the general ledger.
ORIG_DATE	Records when the transaction was generated. This is most useful for transactions occurring in legacy systems that upload their transactions in batches. In many cases, this will be the same as the LEDGER_DATE as transactions will be generated and posted on the same day.
NATIVE_AMNT	Amount of the transaction in the native currency of the transaction. Where the native currency and the system currency are the same, NATIVE_AMNT and AMNT will be the same.
CURRENCY_ID_FK	Currency identifier for the transaction. In most cases, this will equal the system currency. It is stored as ISO4217 currency codes. CURRENCY_ID links to the CURRENCY table if a friendly version of the currency is needed (E.X. AUD can be referenced to "Australian Dollar", or appropriate string as required by the language of the country the system is established in.)
RESP_ENTITY	This is the identifier for the entity who created the transaction. As this can be a non-KSA entity, the KIM entity identifier is stored.
DOCUMENT_ID_FK	Identifier for an XML document that holds information regarding the transaction. This could be a number of different elements. For example, a bookstore transaction could send the names of the books and their prices in the document. This document is for information presentment and is not intended to be used as part of the accounting process.
DEFER_STAT	Boolean that is only applicable to debit transactions. It will be null for credits. If True, then this debit has been deferred. The identifier for the deferment will be stored in DEFER_ID.
DEFER_ID	Reciprocating transaction reference if a transaction is deferred. In the case of a deferment transaction, this will point to the transaction which it defers. In the case of a deferred transaction, this will point at the deferment transaction.
IS_REFUNDABLE	A Boolean that is only applicable to credits. It answers the question "is this credit refundable?" If false, the transaction cannot be refunded if it causes a credit balance. This would be the case for types of credits.
REFUND_RULE	Defines the refund processing rules. If this is blank, then it defaults to the refund rule defined in CREDIT_TYPE. However, if this value is set, then it overrides the rules in CREDIT_TYPE and takes priority. This rule follows the same format as CREDIT_TYPE refund rule, except it also permits the option of refunding to another KSA account. This override refund rule is required to permit overpayment refunds to other sources. Use cases would be ParentPLUS loans, as well as sponsorship overpayments.

	As with REFUND_RULE in the CREDIT_TYPE data model, the exact meaning of this field is not yet fully defined, as research is ongoing into use cases.
EXPIRATION_DATE	Only applicable to DEFERMENT transactions, otherwise it is set to null. On this date, the deferment will be expired.
CLEAR_DATE	Only applicable to PAYMENT transactions, otherwise it is set to null. The clear date is used in refund processing as the date as which a transaction is considered to be 'like cash'. For example, when processing a check payment, an institution may choose not to refund against a check until 10 days have passed. By default, this is derived from the system date, as well as the CREDIT_TYPE.CLEAR_PERIOD.
IS_GL_OVERRIDDEN	Answers the question "is the default GL breakdown for this transaction overridden?" If this is TRUE, then the default breakdown to the GL as referenced through the debit type on the transaction is ignored, and the values derived from KSSA_GL_BREAKDOWN_OVERRIDE are used instead.

Table: GL_BREAKDOWN_OVERRIDE

(If the general ledger breakdown for a transaction type is not the appropriate breakdown for the transaction, then the IS_GL_OVERRIDEN is set to true, and this table is referenced to give the desired GL breakdown for the transaction.)

ID	Autonumbered primary key.
TRANSACTION_ID_FK	Foreign key to link the transaction to row(s) in this breakdown table.
GL_ACCOUNT	The GL account number to be credited.
BREAKDOWN	The percentage breakdown of the amount. See KSSA_GL_BREAKDOWN table for more information relating to these fields.

Table: CURRENCY

(Stores information about currencies used within the system. A transaction may not be denominated in a currency that does not exist within this table. The system currency will also be present in this table.)

ID	System generated primary key.
ISO_ID	Identifier for the currency in ISO 4217 format. (Examples, EUR, GBP, AUD)
NAME	Name of the currency in the appropriate language for the system. For example Euro, British Pound, Australian Dollar.
DESCRIPTION	If an extended description is warranted in the UI, it would be

	placed here.

Table: DOCUMENT

(Stores documents as they relate to transactions. It contains details about the transaction that are useful to the user. An example might be if a student purchases books in the bookstore, the document could contain which books were purchased. This information is not intended to be 'interpreted' by the system, rather provide customer service information to the student. The exact format of the document and permissible entries are still under review.

ID	Autonumbered PK for the document.
DOCUMENT	The actual XML formatted document.
CREATION_DATE	The date the document was entered into the system. Often, documents will be transmitted with transactions; therefore they will have the same date as those transactions.
EDIT_REASON	Documents are not generally editable. When a user has the appropriate permissions to edit a document, they should indicate a reason for the document being edited. This will be stored in this field.

Allocation

Generic

The allocation table links together transactions so that the system can track which payments were allocated to which charges. This also means that payments can be de-allocated from charges, and reallocated dynamically, should situations change on the account.

The KSSA_ALLOCATION table is the table of record for allocation information. The TRANSACTION data structure contains grouped information on allocations, but only the ALLOCATION table contains the actual reference of payments and charges. If for whatever reason, the two were to become unsynchronized, the ALLOCATION table would be the table to verify.

Examples of re-allocation would include the posting of charges to the account that are of higher priority than earlier transactions, or the refund of a transaction that had previously been paid.

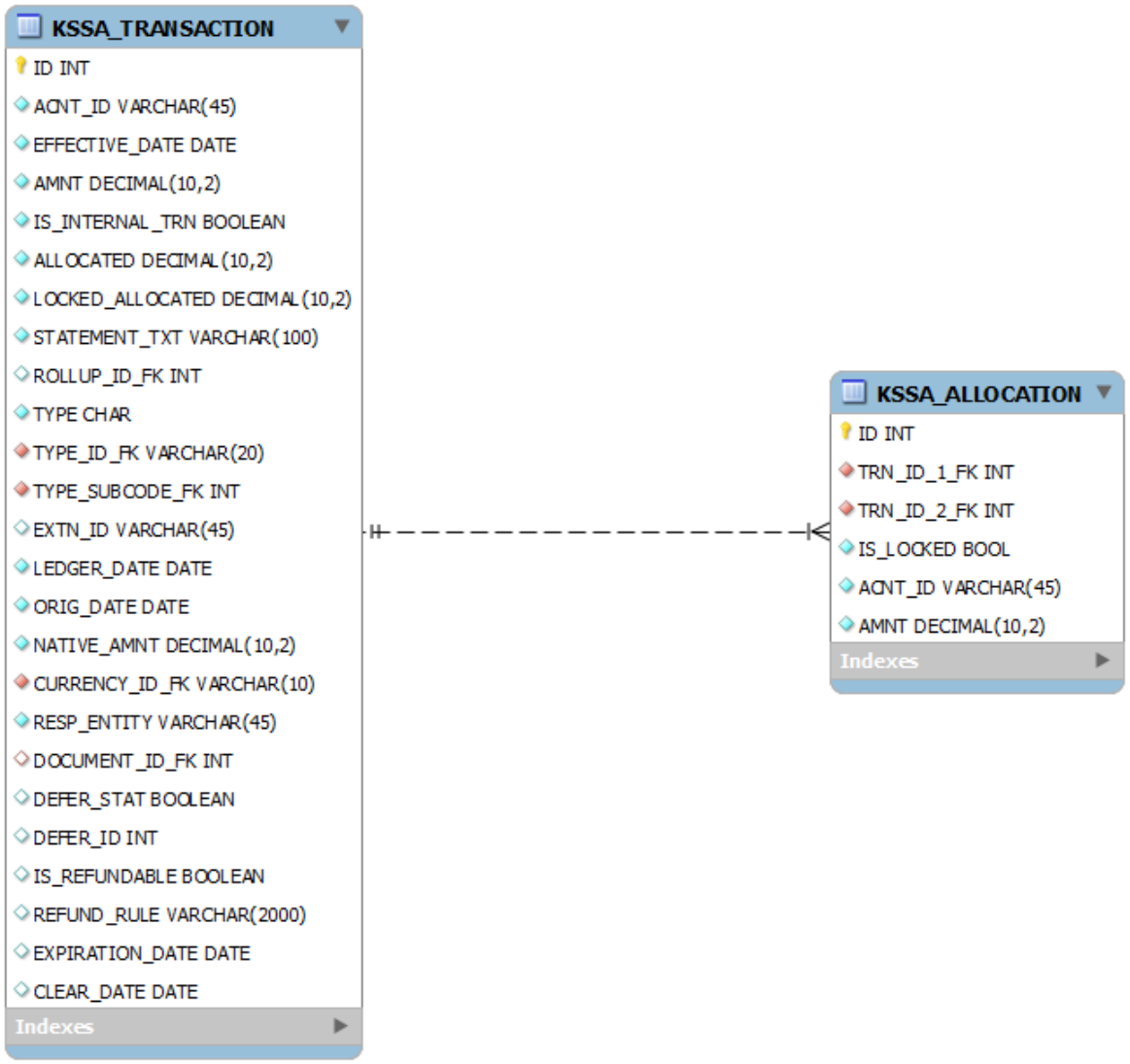

Table: ALLOCATION

(Tracks allocations between different transactions.)

ID	Autonumbered PK for the allocation.
TRN_ID_1_FK	Transaction identifier for the first transaction. This transaction is the "payer" – that is to say that the transaction identified in this key is the credit balance that will pay off a debit balance.
TRN_ID_2_FK	Transaction identifier for the second transaction. This transaction is the "paid" transaction. That is to say that the transaction identified by this key will have its debit balance a paid with funds from transaction 1.
IS_LOCKED	If this is a locked allocation (that is to say that the payment allocation routine is not allowed to deallocate the payment) then this value will be true.
ACNT_ID	This value can be derived from either of the transactions, as

	allocations can only occur on a single account. However it is included to permit easier tracking of allocations to accounts. It points to the account identifier as referenced in the ACNT table.
AMNT	The amount of the allocation. This is recorded in the default system currency. There may be many allocations from a payment that are used to pay off a single charge, and many charges may be paid off by a single payment.

Account Information

Generic

Account information is the general data format for types of information that can be attached to an account. It supports the Information Class and its children, Memo, FollowUpMemo, Flag and Alert. All information types are associated with an account, and may be optionally associated with a single transaction.

In a practical sense, a Memo is a small piece of text that can be viewed by a CSR to give more information as to why something is happening on an account. It may be generated by the system or by a user, and exists to give a human readable log of actions on the account.

A flag is a predefined, computer readable piece of information on an account. Flags can be read by humans, but can also be used as part of an automated decision making process. For example, there might be an insufficient funds flag. Based on configuration, the system may issue certain types of holds or bars, depending on such a flag.

An alert is a message placed on an account that is displayed when an account is accessed by a CSR. A practical example might be that the address on the account is found to be incorrect. In addition to putting a flag on the account to show an incorrect address, the system might also have an alert that simply informs the CSR that there is a problem with the address on the account. The CSR would then be able to make a decision as to how to proceed with a student, if presented with that information.

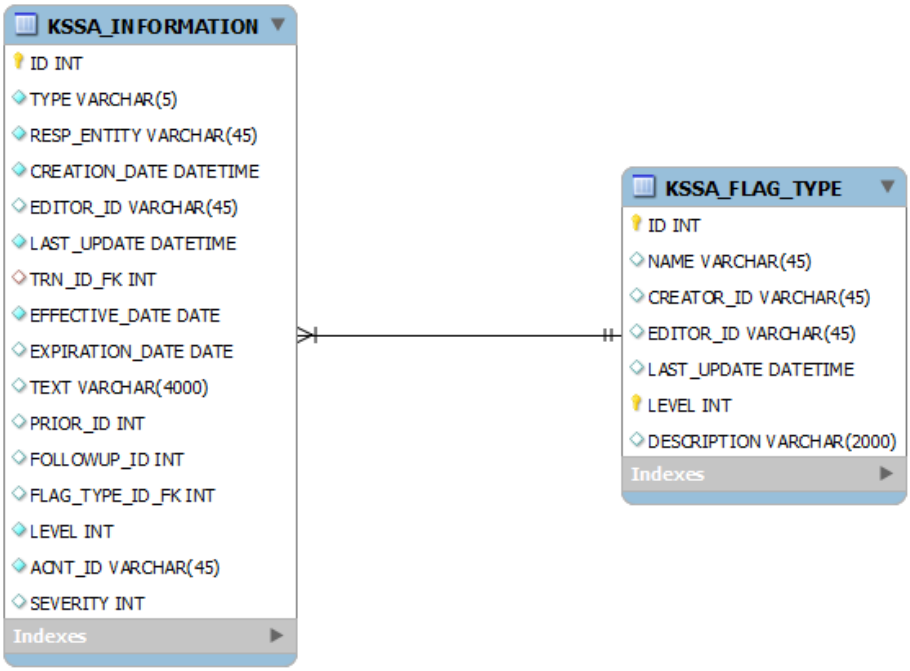

Table: INFORMATION

(Stores multiple types of information about an account or a transaction.)

ID	Autonumbered primary key for the piece of information.
TYPE	Set to IM (Information->Memo) IMF (Information->Memo->FollowUpMemo), IF (Information->Flag) or IA (Information->Alert) to define the type of class that is derived from this entry.
RESP_ENTITY	The identifier for the entity that created the account information.
CREATION_DATE	This is the date that is set when the information is saved to the system.
TRN_ID_FK	This may be populated if the memo relates specifically to a transaction. As an example, if a deferment were issued on an account, the system may prompt the user to enter a reason for the deferment, which would be then linked as a memo to the specific transaction.
EFFECTIVE_DATE	A date for the information is the date on which the memo is considered "in force", allowing information to be placed on the account that does not show until the future.
EXPIRATION_DATE	The date after which the information is considered not effective. None of these dates affect the actual existence of the account information (that is to say that an expired piece of information is not deleted) but they allow a counselor a faster view of currently applicable information, rather than having to trawl through comments that no longer make sense. A comment can be set to expire on its own (by setting

	the EXPIRATION_DATE when the memo is created) or by calling the expire() method, which will populate the EXPIRATION_DATE with the previous day's date.
TEXT	TEXT is the actual text of the memo or the alert. This is a VARCHAR2 field, limiting the size of the memo to 4000 characters. Should more space be needed, a follow-on memo could be created.
PRIOR_ID	A non-null value here indicates that the memo is part of a chain, and links to the memo that precedes it. The existence of this field indicates that the memo is a follow-on memo.
FOLLOWUP_ID	Indicates that a memo has had information added to it. This identifier points the system to the next memo in the chain.
FLAG_TYPE_ID_FK	This is a foreign key, linking to the identifier for a flag that is stored in the table KSSA_FLAG. Flags are entirely user defined. As well as their flag code, flags also have a human-readable friendly name.
LEVEL	Indicates the level of user who can view this information. All users of the KSA system have a LEVEL indicator, and anything that is their level or below is visible (for memos, flags, tags, etc.) It is envisaged that students would have a level of 0, so they could view certain flags, tags, and alerts. It is not expected that any memos would be visible to them. In the case of a flag, the flag has a level identified with it, which is loaded as the default level of the class.
ACNT_ID	Identifies the account identifier against which the information is lodged.
SEVERITY	Identifies the severity of the flag. The higher this number, the more severe. The actual meaning of the severity is decided by any rule that acts upon this value.

Table: FLAG_TYPE

(Stores the attributes for the types of flags that exist in the system.)

ID	Autonumbered primary key.
NAME	A friendly name for the flag. This would be displayed to the user.
LEVEL	Minimum level of the user who could see this type of flag.
DESCRIPTION	Readable description of the flag with more detail than the high-level NAME field. For example "This user has bounced a check on their account. The bursar's office will not accept checks from those users who have in the past bounced a check. They should use another payment method."

Activity Data Model

Generic

KSSA_ACTIVITY persists the objects that make up the activity log, supporting the activity service. This provides a security log of activity that happens within the system.

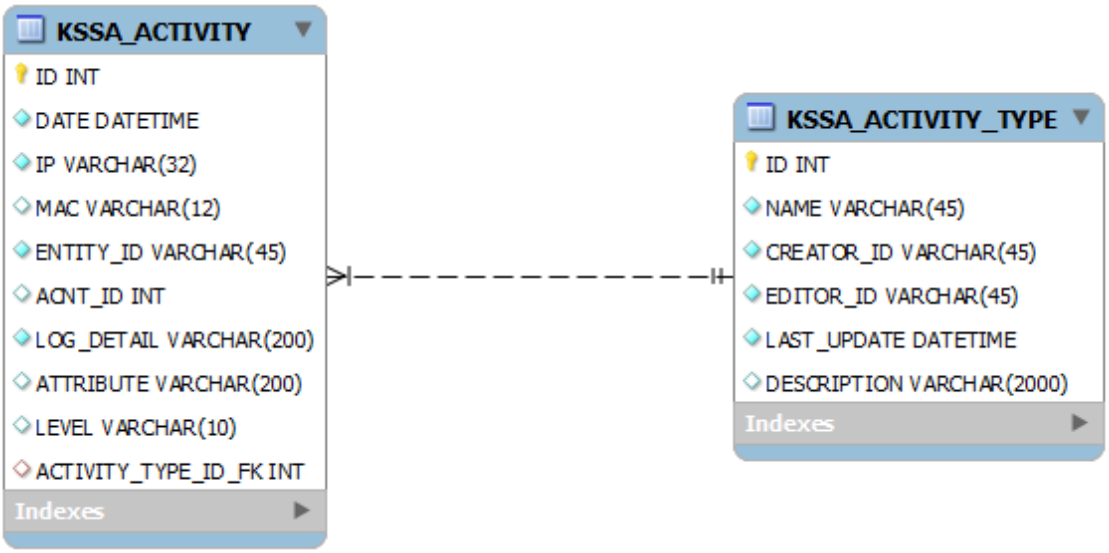

Table: ACTIVITY

(Tracks activity within the KSA system.)

ID	Autonumbered primary key for the ACTIVITY table.
DATE	Date and time of the logged activity.
IP	IP address of the originating system that caused the activity.
MAC	Optional MAC address of the system that caused the activity.
ENTITY_ID	The entity identifier for the user/system that caused the activity.
ACNT_ID	The account against which the activity was logged, if applicable.
LOG_DETAIL	Readable explanation of the activity that has occurred. For example "A new credit type was created within the system."
ATTRIBUTE	An optional attribute that describes the activity I more detail. For example, if a new debit type, called "Bookstore Charge" was created, then the attribute would be set as "Bookstore Charge".
LEVEL	The level is the level of the activity, which is a numerical value, The lower the value, the more transactional the detail, the higher, the more serious the detail. We will adopt the Apache LogLevel names for this purpose. These are emerg, alert, crit, error, warn, notice, info, debug. More information can be found at http://httpd.apache.org/docs/2.0/mod/core.html#loglevel
ACTIVITY_TYPE_ID_FK	A classification of the problem, for example EXCEPTION,

	SECURITY, etc. These types are defined in the KSSA_ACTIVITY_TYPE table. It is envisaged that the basic types of error will be predefined in the software, but their names can be configured, and other types can be added in future revisions.

Table: ACTIVITY_TYPE
(Stores the types of activities that exist in the system. While these types can be configured, it is envisioned that the KSA system will come with predefined types that can be customized (by name) but it is not, at this time, envisioned that the administrator would add new types of activities.)

ID	Autonumbered primary key.
NAME	A friendly name for the activity. This would be displayed to the user.
DESCRIPTION	UI Description of the activity type.

Account

Generic

The KSSA_ACNT table is a central reference point that ties together a great number of tables.

KSSA_ACNT is the central account table. ID is the account identifier as is referenced throughout the rest of the system as ACNT_ID. This is the identifier of a KSA account.

KSSA_ACNT and many of its related tables are used to populate the Account objects. Two key tables store the information that relates to an account. In addition to the core KSSA_ACNT table, there is also KSSA_ACNT_PROTECTED_INFO; a table of details that are considered more sensitive than some other types of data. For the sake of security, a regular user that has reporting access to the data in KSA would NOT have access to this table. This populates the AccountProtectedInformation class, which, in addition to not being loaded as default, triggers memo entries when data within that class are accessed. This table has a one-to-one relationship with ACNT, and therefore KSSA_ACNT_PROTECTED_INFO.ID = KSSA_ACNT.ID

KSSA_TRANSACTION is defined earlier in this document.

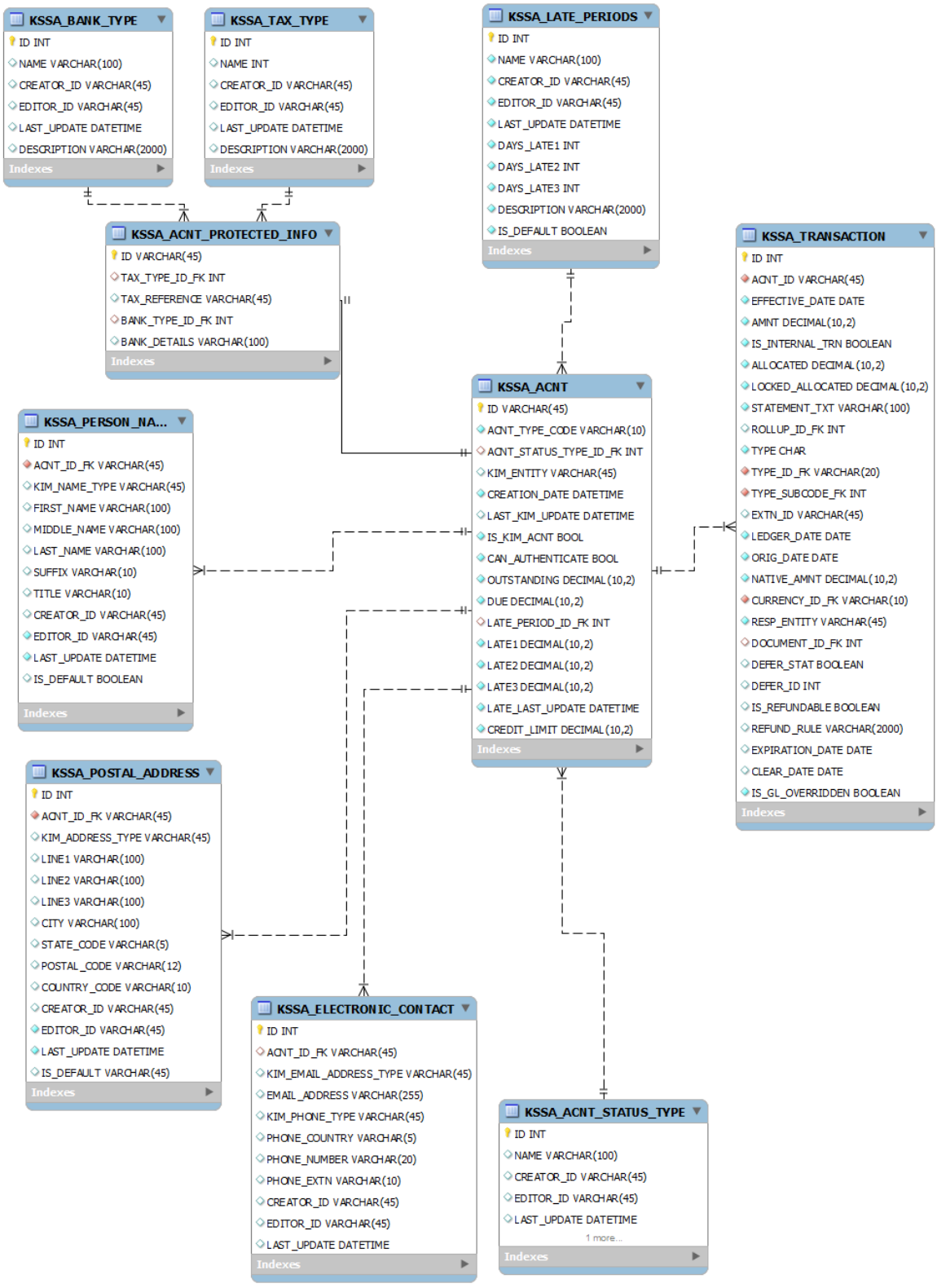
KSSA_BANK_TYPE
ID INT
NAME VARCHAR(100)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
DESCRIPTION VARCHAR(2000)
Indexes
KSSA_TAX_TYPE
ID INT
NAME INT
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
DESCRIPTION VARCHAR(2000)
Indexes
KSSA_LATE_PERIODS
ID INT
NAME VARCHAR(100)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
DAYS_LATE1 INT
DAYS_LATE2 INT
DAYS_LATE3 INT
DESCRIPTION VARCHAR(2000)
IS_DEFAULT BOOLEAN
Indexes
KSSA_ACNT_PROTECTED_INFO
ID VARCHAR(45)
TAX_TYPE_ID_FK INT
TAX_REFERENCE VARCHAR(45)
BANK_TYPE_ID_FK INT
BANK_DETAILS VARCHAR(100)
Indexes
KSSA_TRANSACTION
ID INT
ACNT_ID VARCHAR(45)
EFFECTIVE_DATE DATE
AMNT DECIMAL(10,2)
IS_INTERNAL_TRN BOOLEAN
ALLOCATED DECIMAL(10,2)
LOCKED_ALLOCATED DECIMAL(10,2)
STATEMENT_TXT VARCHAR(100)
ROLLUP_ID_FK INT
TYPE CHAR
TYPE_ID_FK VARCHAR(20)
TYPE_SUBCODE_FK INT
EXTN_ID VARCHAR(45)
LEDGER_DATE DATE
ORIG_DATE DATE
NATIVE_AMNT DECIMAL(10,2)
CURRENCY_ID_FK VARCHAR(10)
RESP_ENTITY VARCHAR(45)
DOCUMENT_ID_FK INT
DEFER_STAT BOOLEAN
DEFER_ID INT
IS_REFUNDABLE BOOLEAN
REFUND_RULE VARCHAR(2000)
EXPIRATION_DATE DATE
CLEAR_DATE DATE
IS_GL_OVERRIDDEN BOOLEAN
Indexes
KSSA_ACNT
ID VARCHAR(45)
ACNT_TYPE_CODE VARCHAR(10)
ACNT_STATUS_TYPE_ID_FK INT
KIM_ENTITY VARCHAR(45)
CREATION_DATE DATETIME
LAST_KIM_UPDATE DATETIME
IS_KIM_ACNT BOOL
CAN_AUTHENTICATE BOOL
OUTSTANDING DECIMAL(10,2)
DUE DECIMAL(10,2)
LATE_PERIOD_ID_FK INT
LATE1 DECIMAL(10,2)
LATE2 DECIMAL(10,2)
LATE3 DECIMAL(10,2)
LATE_LAST_UPDATE DATETIME
CREDIT_LIMIT DECIMAL(10,2)
Indexes
KSSA_PERSON_NA...
ID INT
ACNT_ID_FK VARCHAR(45)
KIM_NAME_TYPE VARCHAR(45)
FIRST_NAME VARCHAR(100)
MIDDLE_NAME VARCHAR(100)
LAST_NAME VARCHAR(100)
SUFFIX VARCHAR(10)
TITLE VARCHAR(10)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
IS_DEFAULT BOOLEAN
Indexes
KSSA_POSTAL_ADDRESS
ID INT
ACNT_ID_FK VARCHAR(45)
KIM_ADDRESS_TYPE VARCHAR(45)
LINE1 VARCHAR(100)
LINE2 VARCHAR(100)
LINE3 VARCHAR(100)
CITY VARCHAR(100)
STATE_CODE VARCHAR(5)
POSTAL_CODE VARCHAR(12)
COUNTRY_CODE VARCHAR(10)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
IS_DEFAULT VARCHAR(45)
Indexes
KSSA_ELECTRONIC_CONTACT
ID INT
ACNT_ID_FK VARCHAR(45)
KIM_EMAIL_ADDRESS_TYPE VARCHAR(45)
EMAIL_ADDRESS VARCHAR(255)
KIM_PHONE_TYPE VARCHAR(45)
PHONE_COUNTRY VARCHAR(5)
PHONE_NUMBER VARCHAR(20)
PHONE_EXTN VARCHAR(10)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
Indexes
KSSA_ACNT_STATUS_TYPE
ID INT
NAME VARCHAR(100)
CREATOR_ID VARCHAR(45)
EDITOR_ID VARCHAR(45)
LAST_UPDATE DATETIME
1 more...
Indexes

Table: ACNT

(Tracks the fundamentals of an account within the KSA system.)

ID	The account identifier. It is referenced throughout the rest of the system as ACNT_ID. This is the identifier of a KSA (not a KIM) account.
ACNT_TYPE_CODE	Used to disambiguate the classes that can be produced from these tables. Possible values are AND for Account->NonChargeable->Delegate, ACD for Account->Chargeable->DirectChargeAccount and ACS for Account->Chargeable->SponsorAccount. Friendly versions of these names for UI purposes can be queried in the ACNT_TYPE table, searching on the CODE field.
ACNT_STATUS_TYPE_ID_FK	References the ACNT_STATUS_TYPE table, referencing the status of the account. The account status is a single value that, can be used to impose certain limits on the account. This is a high-level status of the account, and is used in combination with other account indicators, such as flags.
KIM_ENTITY	If the account is derived from a KIM account, then the KIM identifier is stored here. This is often referred to as the NetID of the user, and will often follow the format of first initial/last name.
CREATION_DATE	The date that the account was established within the KSA system.
LAST_KIM_UPDATE	If the account is a KIM account, LAST_KIM_UPDATE is the date and time that the details were validated with the KIM datastore.
IS_KIM_ACNT	Boolean that answers the question “did this account originate from the KIM system?” If true, then the KIM fields are implied. If false, then the KIM fields will be null.
CAN_AUTHENTICATE	Boolean that answers the question “can this user authenticate into the KSA system?”
OUTSTANDING	The balance of the account, in the system currency that is outstanding on the account. This includes any amounts that are not yet due.
DUE	The balance of the account that is due at this moment in time. That is, all the transactions which have an effectiveDate of today or before.
LATE_PERIOD_ID_FK	LATE_PERIOD_ID_FK references the late payment table. This permits the system to have configurable groups of late payment “buckets”. For example, a normal student might be considered “late” once their balance has been due for 30 days, and then they are progressively later at 60 and 90 days. Whereas for some customers (maybe a sponsoring employer) there may be an agreement that the account is not past due until 60/90/120 days.
LATE1..3	The aged balance in “buckets” according to the LATE_PERIOD table.

LAST_LATE_UPDATE	Date and time when the account last went through the ageing process.
CREDIT_LIMIT	Credit limit for the account. Enforcement of this limit is during the transaction creation process.

Table: ACNT_PROTECTED_INFO
(Tracks types of data that are required for system functioning that are considered more sensitive than other types of data.)

ID	The account identifier. It is referenced throughout the rest of the system as ACNT_ID. This is the identifier of a KSA (not a KIM) account.
TAX_TYPE_ID_FK	A foreign key to the TAX_TYPE table. This is a configurable list of types of tax identifiers that the system might accept. If the system is deployed in the US, a TAX_TYPE might be "Social Security Number".
TAX_REFERENCE	The actual tax identifier as referenced in TAX_TYPE.
BANK_TYPE_ID_FK	A foreign key to the bank type table, permitting the storage of different types of bank information. For example, an ACH type used in the US would require a routing number, and account number, and an account type (checking, savings, etc.) An IBAN type would store the information differently.
BANK_DETAILS	The actual detail as references in BANK_TYPE, for example, the actual IBAN of the account holder.

Table: BANK_TYPE
(A simple type table that defines the different types of bank information that might be stored in the KSA system.)

ID	Autonumbered primary key.
NAME	Friendly name for the type of bank information. For example "ACH" or "IBAN".
DESCRIPTION	A longer description of the expected value in the field.

Table: TAX_TYPE
(A simple type table that defines the different types of tax information that might be stored in the KSA system.)

ID	Autonumbered primary key.
NAME	Friendly name for the tax type identifier. For example "U.S. Social Security Number" or "British National Insurance Number".
DESCRIPTION	A longer description of the tax number type to assist

	operators in understanding where this information may come from, expected format, etc.

Table: PERSON_NAME

(Tracks a person's name. This structure closely mimics the KIM name standard, and is stored in its own table to permit changes that may occur in the future. This format might be problematic for internationalization.)

ID	Name identifier.
ACNT_ID_FK	Foreign key link to the account in question.
KIM_NAME_TYPE	If the name is derived from KIM, the name type is stored here. This permits us to update the name from KIM by retrieving the correct name record if a student has more than one registered name. If the field is null, then the name is only stored within KSA.
FIRST_NAME	First name of the person.
MIDDLE_NAME	Middle name of the person.
LAST_NAME	Last name of the person.
SUFFIX	Freeform suffix, allowing for appended titles or generational information.
TITLE	Freeform prefix, allowing for titles that come before the name.
IS_DEFAULT	Answers the question "is this the default name used on the account"?

Table: POSTAL_ADDRESS

(Tracks a postal address. This structure closely mimics the KIM name standard, and is stored in its own table to permit changes that may occur in the future. In particular, this format may be a problem for internationalization.)

ID	Autonumbered primary key
ACNT_ID_FK	Foreign key link to the account in question.
KIM_ADDRESS_TYPE	If the address is derived from KIM, then this is set to the address type. If it is null, then the address is stored locally only, in KSA.
LINE1..3	Lines 1 through 3 of the address.
CITY	City part of the address. This should be interpreted openly as the locality name.
STATE_CODE	If the country has states or other major localities as part of its addresses, then they are stored here.
POSTAL_CODE	If the address has a coded address, it is stored here.
COUNTRY_CODE	Code for the country of the address.
IS_DEFAULT	Answers the question "is this the default address on the account?"

Table: ELECTRONIC_CONTACT

(Tracks the electronic contact information for an account. Closely mimics KIM information.)

ID	Autonumbered primary key
ACNT_ID_FK	Foreign key link to the account in question.
KIM_EMAIL_ADDRESS_TYPE	If the email address is derived from KIM, then this is set to the address type. If it is null, then the email address is stored locally only, in KSA.
EMAIL_ADDRESS	The actual email address.
KIM_PHONE_TYPE	If the phone number is derived from KIM, then this is set to the phone type. If it is null, then the number only exists within KSA.
PHONE_COUNTRY	The country code for the number. In the absence of a precedent on the use of this field, and the constraints of the field, we will store an ISO3166 country code.
PHONE_NUMBER	The major numerical part of the number.
PHONE_EXTN	The extension part of the number, if applicable.

Table: ACNT_STATUS_TYPE

(Storage for the different account statuses that can be applied to an account.)

ID	Autonumbered primary key.
NAME	Friendly name for the account status. For example “In good standing”.
DESCRIPTION	A longer description of what it means to be in that status.

Table: LATE_PERIODS

(Stores the different period definitions against which an account might be aged.)

ID	Autonumbered primary key.
NAME	The name of the late period definition as displayed to the user.
DAYS_LATE1..3	The number of days after the effective date of a transaction that the account is considered to be in the appropriate late bucket. The standard reference model would be LATE1=30, LATE2=60, LATE3=90.
DESCRIPTION	A more verbose description of the late model, giving a representative better information as to what types of accounts this might be the appropriate late period definition for.
IS_DEFAULT	Boolean that answers the question “is this the default late period definition?”