Keith James Baugh
TopTal screening process
Project Start: 28th September 2020
Project End Date: 12th October 2020

Online Bookstore Data Model

11th October 2020

Original Requirement Statement

A random client wants you to design a database for an online book store.

A client has specific requirements for a webshop. For that purpose, it is important to understand them so you can build a proper data model.

- A client wants to sell books in different genres. Each book should contain at least following information: ISBN code, title, genre/category, description
- A client would like to have a support to translate book titles, categories and descriptions.
- For each book, customers should be able to write reviews.
- A client would like that they can recommend similar books to a customer.
- A client would like to have an option of setting book prices in advance, e.g. for summer sales.
- Each customer should be able to register and provide personal information.
- A customer can buy books and should be able to see their purchasing history along with invoicing information.

1. Create an Entity-Relationship Model (ERM) diagram, including:
   a. A description of the entities and their attributes.
   b. Relationships
2. A database which implements the ERM.
3. SQL script that will create the whole database along with the constraints and relationships.
   a. Create a function which generates sample data.
4. Create a function/query that returns recommended/related books for a specific book.
5. Create a report query that will show sales stats to the customer by genre a month and a previous month.
6. Read - replica database on the same machine. Replication needs to be as real-time as possible.

7. Automatic backups of the database every two hours.
    - All requirements must be done under the same database manager (Eg: Oracle or SQL Server)

Deliverables

The GIT project directory contains the following items of note.

Deliverable: (1)

> *Create an Entity-Relationship Model (ERM) diagram, including:*
> *a. A description of the entities and their attributes.*
> *b. Relationships*

These are provided in this document, see 'Entity-Relationship-Diagram' and onwards for this item.

Deliverable: (2/3)

> *A database which implements the ERM.*
>
> *SQL script that will create the whole database along with the constraints and relationships.*
>
> *Create a function which generates sample data.*

(See Installation Instructions Below for these items, specifically scripts tt.ddl, dba.sql, and sample_data_load.sql)

Deliverable: (4)

> *Create a function/query that returns recommended/related books for a specific book.*

(See schema/func_get_books_recomm.sql for the code for this function)

Deliverable: (5)

> *Create a report query that will show sales stats to the customer by genre a month and a previous month.*

(See sql/sales_by_genre.sql in the delivery directory. This report requests the user to enter the MONTH and then produces are report for that month and its predecessor.)

Deliverable: (6)

> *Read - replica database on the same machine. Replication needs to be as real-time as possible.*

(See Installation Instructions, the Read Only Replica is created as part of the installation.)

Deliverable: (7)

> *Automatic backups of the database every two hours.*
>
> - *All requirements must be done under the same database manager (Eg: Oracle or SQL Server)*

(See Installation Instructions points (5) and (6)) Backup scripts are supplied and must be copied and modified to suit the target environment.

NOTES ON DELIVERY

Under normal circumstances it would be necessary to engage the client with results of issues and observations encountered during the development of the Data model. I understand that in the context of the screening-task, that this level of realism would be difficult to construct. Nevertheless, there are a number of important issues that need to be itemized, even for this 'simulated task'. These, in the context of the simulated client interaction would be for the protection of the client and his/her expectation regarding the delivery. All my concerns are listed below.

1. A high level statement of intent to engage in translation activities on Book-Titles, Categories, and Descriptions is expressed in the requirement, with little detail on what level of support should be provided in the data model. I have therefore delivered, a modest level of 'work-flow' within the tables holding this data. The translatable material in Books and Categories is held in a table separate from the 'singular' Book or Category tables. BOOK_INFO and CATEGORY_INFO are tables that extend the basic Book and Category object to include numerous translations. Within each, a TRANSLATION_STATUS field will hold the approval status for translation activities.

2. My simple research has shown that, whilst books may be translated into different languages, each printed language edition would receive a different ISBN number. The model supports, as per the requirement, the creation of different language 'descriptive' texts, but, the book itself will, of course, still only be available in one language (related to the book's ISBN). A single ISBN field is available and this is designed to hold a 13 digit ISBN number. If the client wished additionally to hold 10-digit ISBN numbers, such a field would have to be added.
3. Generally, there is only modest mention of internationalization of the site data. As it is not specifically mentioned, the pricing data handled in this data model holds only a single price currency, and this currency has to be implied by the code that renders the data to the user. If specific nationalization is required in this area, it can be provided with some small model extensions.
4. Customer personal information is supported, with basic: title; first-name; middle-initial; last name; email-address. Additionally, the model supports the creation of multiple shipment addresses, and billing addresses. Whilst instruction is given in the detailed text that covers this entity type, on how the application should safeguard address information during order fulfilment, there is no audit of data and it would be suggested that consideration be given to extending the schema in the area of capturing address details at order-time to protect against data erosion.
5. In the interests of proper understanding, within requirement no.6.

> *"Read - replica database on the same machine. Replication needs to be as real-time as possible."*

..the term, 'database', is taken to mean all objects that are contained in the SCHEMA which is created to hold the bookstore data. This is to be seen as distinct from the 'complete' database that encompasses the bookstore data, together with undo tablespace, system tablespace, and other possible system auxiliary areas.

In other words, the requirement is interpreted to mean that the 'data itself' is the focus of the replication activity, and not necessarily the complete environment that envelops the same in an Oracle database context.

ENVIRONMENT CONSIDERATIONS

The code delivered in this release is intended for execution on an Oracle Database Version 11.2 or greater running on a UNIX flavour system. The Oracle database should use storage management facilities provided by Oracle's Advanced Storage Management (ASM) layer. Further notes on Oracle setup are given later.

Database Setup Considerations

This document does not cover the basic creation of the Oracle Databases used for this facility. The following are requirements that the database layers need to respect to function correctly. Variation would need to be checked with me for compliance.

```
NLS_NCHAR_CHARACTERSET          AL16UTF16
NLS_LANGUAGE                    AMERICAN
NLS_TERRITORY                   AMERICA
NLS_CURRENCY                    $
NLS_ISO_CURRENCY                AMERICA
NLS_NUMERIC_CHARACTERS          .,
NLS_CHARACTERSET                AL32UTF8
NLS_CALENDAR                    GREGORIAN
NLS_DATE_FORMAT                 DD-MON-RR
NLS_DATE_LANGUAGE               AMERICAN
NLS_SORT                        BINARY
NLS_TIME_FORMAT                 HH.MI.SSXFF AM
NLS_TIMESTAMP_FORMAT            DD-MON-RR HH.MI.SSXFF AM
NLS_TIME_TZ_FORMAT              HH.MI.SSXFF AM TZR
NLS_TIMESTAMP_TZ_FORMAT         DD-MON-RR HH.MI.SSXFF AM TZR
NLS_DUAL_CURRENCY               $
NLS_COMP                        BINARY
NLS_LENGTH_SEMANTICS            BYTE
NLS_NCHAR_CONV_EXCP             FALSE
NLS_RDBMS_VERSION               11.2.0.4.0
```

Notes on DB setup for Backups

I recommend that the databases (Master and Read-only) are BOTH configured to use Oracle's ASM storage layer. And that they BOTH are configured in ARCHIVELOG MODE and further that Block Change Tracking (BCT) is enabled for each. The requirement to backup the database every 2 hours is a very unusual request with an Oracle database, BCT will provide a very lightweight method for Oracle to determine blocks that have changed since the last backup was taken. I would actually recommend that the database is NOT backed up every two hours. This is

because, with Oracle's Online-Hot-backups, recoverability to the moment of failure is possible by application of archived redo logs to the previous full backup. Taking full backup every two hours will place significant load on the system at times when the customers will be using the system, and would provide no more safeguard of the database over the suggested backup and archive of logs. It would be normal for the DBA to recommend a backup strategy that would cover the customer's concerns and requirements, and Oracle is very capable of delivering data security regarding backups without compromising the performance of the system by repeating unnecessary backups.

Backup Restrictions

A backup process is defined which relies on the database saving its data to an ASM diskgroup +DG_FLASH. Whilst the data is assumed to reside on another disk group named: +DG_DATA. Externalization of the data (to a UNIX mount-point) is performed by the backup routines to provide a secondary copy of the backup data. If the client has an alternative setup, any changes must be updated in the backup scripts supplied to match with the actual set-up.

Anatomy of the GIT delivery

The following describe the contents of the GIT repository sub-directories and files.

sql - supporting SQL scripts and reports

bin - This directory holds the backup ksh scripts that invoke rman to perform the backups. There are two rman driver scripts also in the same directory.

utils - maintenance korn shell and perl scripts used in the construction of the SQL scripts (not a deliverable item)

tests - function test scripts used in unit testing the delivered functions.

schema - object creation scripts used by the main installation script (tt.ddl).

sample_data - holds SQL scripts that are used by the main script used to load sample data (sample_data_load.sql)

config_data - holds SQL scripts that are used to create main configuration data. This is sample data, but, it is of the configurational form (languages, payment_methods etc), rather than customer or book related sample data.

Installation Instructions

Installation of the database objects in this delivery can be achieved by running three files.

1. Prior to execution, the file (env.sql) must be edited and the values set for the customer's particular environment. The following environment specific parameters should be set. As there are two databases, the MASTER and the READONLY database, and each one requires a schema to owner the objects, the following parameters are necessary.

   ### env.sql - Setting parameters

   - DEFTABLESPACE - the name of the default tablespace to be set for the schema that will be made to hold the bookstore objects. This is the same for both MASTER and READONLY systems.
   - MASTERDB - the TNS alias for the database used as the master for the bookstore.
   - READONLYDB - the TNS alias for the database used as the read-only copy of the bookstore.
   - MASTERUSERNAME - the name of the schema to be created in the master database to own the bookstore objects.
   - READONLYUSERNAME - the name of the schema to be created in the read-only database to own the bookstore objects.
   - MASTERPASSWORD/READONLYPASSWORD - the passwords to be set for the master and read-only schemas when these are made.
   - SYSPASSMASTER/SYSPADDREADONLY - the passwords for the SYS accounts in the master and read-only databases. These are required to create the owning schemas and to grant the required system privileges to allow the bookstore objects to be created.
2. Having set the parameters that define the environment in the client's systems, start SQL*plus in nolog form and run the following command.

   ```
   bash% sqlplus /nolog @dba.sql
   ```

3. When this completes, the schemas to hold the bookstore objects will have been created and you can now run the main object creation script, while still logged in to SQL*Plus...

   ```
   SQL> @tt.ddl
   ```

4. Now you can install the sample data.

   ```
   SQL> @sample_data_load.sql
   ```

This completes the installation of the bookstore objects and has set up the READ-ONLY copy of the objects in the database nominated for this purpose.

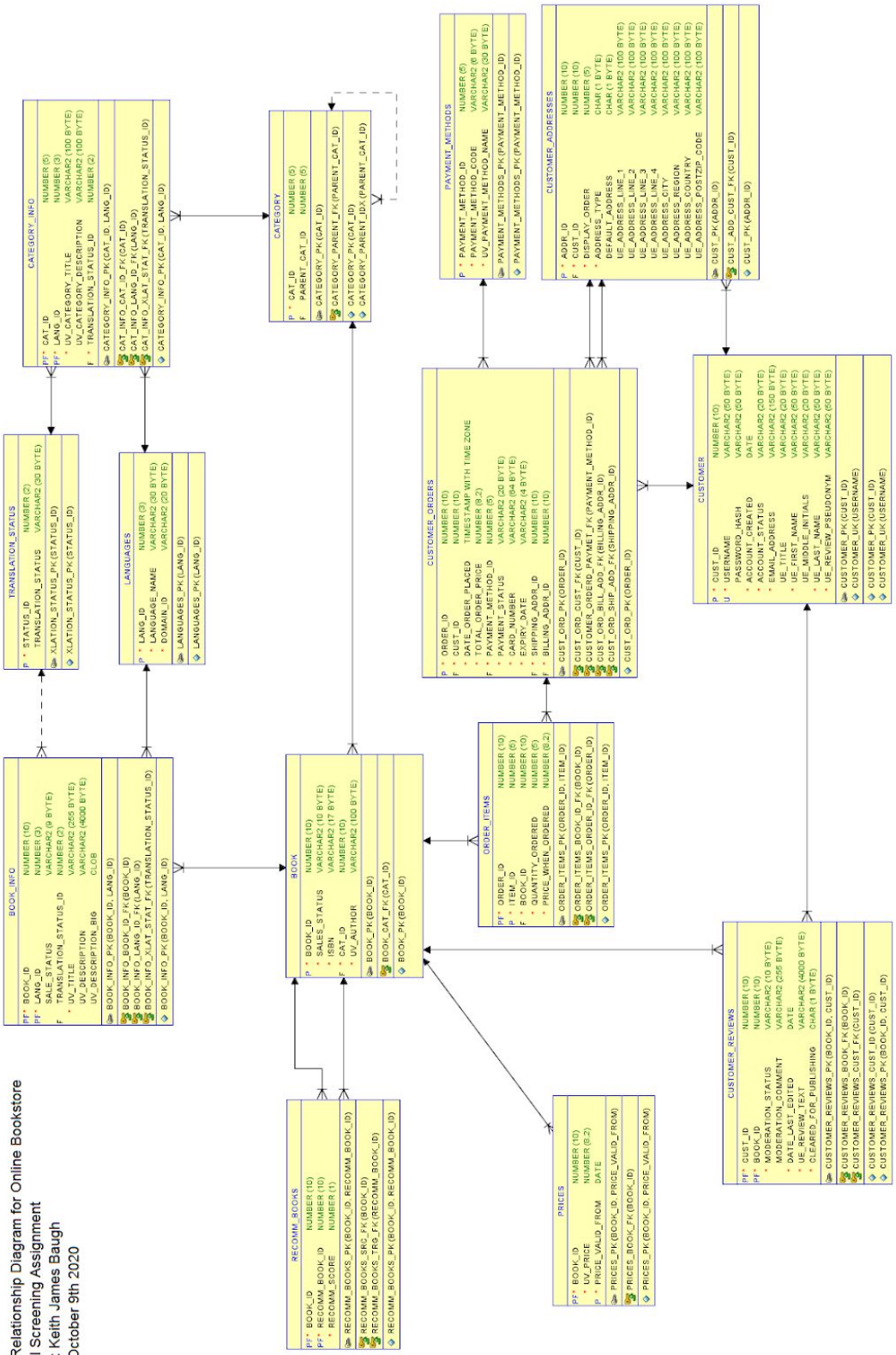
5. BACKUP scripts are saved under the 'bin' directory. Copy the contents of the directory onto the database server that supports the Master and Read-Only databases. These scripts should be amended to set the environment parameters relevant to the client-system: ORACLE_HOME; HOME; ORACLE_SID; PATH will need consideration and setting as per the target system.
6. Once configured, set up the following crontab job to perform the database backups as requested in the requirements:

```
0 2,4,6,8,10,12,14,16,18,20,22 * * * /path_to/backup_incr.ksh

0 0 * * * /path_to/backup_full.ksh
```

Where /path_to/backup_full.ksh and /path_to/backup_incr.ksh are the locations on the database server where the backup files were copied in (5) above. These jobs will perform a backup every 2 hours (incremental) and a (full) backup at midnight every night.

Entity Relationship Diagram



SCHEMA OBJECTS

NOTES ON NOMENCLATURE

The following note and conventions are relevant to the naming of objects in the data model.

1. Fields which carry information that may be directly displayed to the user are prefixed with 'UV_' to signify User-Visible data.
2. Fields which hold data directly entered by the end-user carry the prefix 'UE_' to signify User-Entered data. This can be used to identify fields that should be regarded to contain unformatted data.
3. All numerically keyed fields have a suffix of _ID.

BOOK

This is the table that declares a book and attributes it with its ISBN number, and author(s). Very little other information is available for the book at this level. Possible future enhancements would include a BLOB field to contain a book image. As language variants of a book's title and description are needed, all language specific data is off-loaded into the BOOK_INFO table.

book_id

A numeric value to uniquely a book in the bookstore. A sequence generator is available to create the next value for this field: seq_book_id. New book_ids should be generated with a call to the pseudo-column seq_book_id.nextval when inserting into this field.

sales_status

This field is used to describe the current status of the book in the bookstore. Not all books created within the table are considered to be available for sale. This field will allow for the status of the book to be one of the following checked values: 'ONSALE'; 'BACKORDER'; and 'WITHDRAWN'.

Field Values and their meanings

ONSALE - means that the book is cleared to be offered for sale on the bookstore.

BACKORDER - means that the book is offered for sale on the store, but that it is not immediately available.

WITHDRAWN - means that the book is not currently to be offered for sale on the bookstore.

isbn

This field holds the 13 digit ISBN number. Alternative ISBN numbers are available, (e.g. ISBN 10-digit). If used, these other ISBN values should be saved into a different field, (changes required). There is sufficient capacity in the field sizing to allow for the conventional '-' separation of ISBN sub-fields (e.g. 978-4-23-142320-x).

It is advised, that the ISBN field is subjected to a validation check, to validate the check-digit. This is not covered in this delivery.

uv_author

The name of the author or authors for the book. This field is displayed to the books-store customer.

cat_id

The numeric category id for the book's classification. This is a foreign key into the CATEGORY table. A book can ONLY have one category, but, this can be changed with a small schema change on request.

PRICES

Prices are related to books, each book may have multiple prices defined. The primary key for this table is the book_id and the price_valid_from field. This allows multiple prices to be set for each book with post-dated 'validity' dates. The application should render the price, from this table, that has the greatest 'price_valid_from' date that pre-dates the current system time. There is no provision for multiple currencies or multiple prices in those currencies. If this is desired, changes will be required in the model to facilitate.

As price determination can, in some instances, be dynamic, application developers should be mindful of including price information in any 'cached' renderings of book data.

book_id

The numeric id of the book, foreign keyed against the BOOK table.

uv_price

The 'price' of the book expressed in the general currency adopted by the site for sales. The format of this numeric data is: NUMBER(8,2) allowing max price to be set at 999999.99

price_valid_from

The date from which the price for the book is to be used. The application should select the price that has the greatest ‘price_valid_from’ that pre-dates the current date and time. The time component for this date will be that of the main database server time. This should be understood when setting time and date information for future prices changes.

BOOK_INFO

This is the table that holds the bulk of the user-visible data relating to the book. It's primary key combines the book_id with the lang_id to allow for each book to have multiple language variants of the user-visible data. This is where translations of the book are stored and controlled. Main book data comprising: title; and description are stored in this table in the language defined by 'lang_id'. The lang_id links to the languages table where each language supported is defined. The table also contains translation status data to provide some level of support and workflow around translations.

book_id

This numeric Id of the book with which this language-specific information record is associated. This is a foreign key into the BOOK table.

lang_id

This numeric Id of the language in which this BOOK_INFO data is written. It is a foreign key in the set of supported languages defined in LANGUAGES.

translation_status_id

This is the status id of the language translation held within this BOOK_INFO record. To ease the introduction of support for translation activities, the set of translation status values made be adjusted by making entries in the TRANSLATION_STATUS table.

uv_title

The book title expressed in the language implied by the value of the lang_id value.

uv_description

The book description expressed in the language implied by the value of the lang_id value. The value is allowed to be 4000 bytes in total (bytes not characters). If more than 4000 bytes is required, then the description of the book should be stored in a CLOB field in the uv_description_big field and this field should be set to NULL

uv_description_big

The book description expressed in the language implied by the value of the lang_id value. Only to be used where the description of the book exceeds 4000 bytes. Otherwise always set this value to NULL.

RECOMM_BOOKS

This table is used to link one book, to potentially several other books, for the purpose of creating recommendations against books. In each case, the recommendation is keyed from the original book, and recommended books can be selected from the 'recomm_book_id' field. As it is possible to recommend several books for each book, this table also offers a score value, which may be used by the application to order the recommendations based on relevancy.

book_id

The book_id of the book for which this recommendation is being made. It is a foreign key into the BOOK table.

recomm_book_id

The book_id of the book which is being recommended against the 'book_id'.. It is also a foreign key into the BOOK table.

recomm_score

A single digit numeric score value giving this particular recommendation a weighting of relevance. The application should order recommendations in accordance with this value. It is suggested that the convention be higher scores rank higher than lower scores.

Categories and Category Information

Overview

The design of the data-model section that covers Categories, provides for the translation (and translation workflow) of category titles, and their descriptions, into multiple languages. Allocation of a book to a specific category, (or multiple categories with schema change) is possible, this is done by setting the cat_id for the book record. However, book classifications are not simply single-level data structures, and so any model must be able to provide for a hierarchical structuring of different categories. For example: using very basic research against wiki-pedia, (See Ref 1) - The category of: Historical Non-fiction can have sub-categories of: Academic history; Genealogy; Narrative; People's history etc.

The CATEGORY table holds the main category hierarchy. This data can be regarded as 'independent' from any particular book classification as it just provides the 'standard' genres to describe books in general. The table has a primary key of the cat_id, and an optional additional

element, the parent_cat_id to denote a category's parent.. Data can be saved in this table to define a hierarchy of categories.

Whilst the CATEGORY table holds the structure of the category hierarchy, the CATEGORY_INFO table table contains the titles and descriptions rendered in multiple languages. Again, this table, as for BOOK_INFO contains a translation_status_id to facilitate basic translation workflow.

A book can be given one category, by setting the 'cat_id' field in the BOOK table.

CATEGORY

This table holds the hierarchical structure of the book categories in numerically coded form. No text, or user visible data is held in this table. This table defines which category_id may be used to describe and classify books.

cat_id

The numeric category id for the category.

parent_cat_id

The 'optional' identity of the parent category for a subcategory. If this field is null, the category defined in 'cat_id' will be at the top level of the hierarchy.

CATEGORY_INFO

This table is used to convey meaning to the codes declared in the CATEGORY table (cat_ids). Against each distinct 'cat_id' a set of language titles and descriptions can be provided. Rendering this data for each 'level' in a category hierarchy can be used to provide a 'bread-crumb' listing of a given classification from the top level.

cat_id

The numeric category id for the category. This is foreign keyed off the CATEGORY table.

lang_id

The id of the language for in which the textual title and description of the category is written.

uv_category_title

The title of the category (given by cat_id) expressed in the language whose code is given ny (lang_id). This is visible to the user.

uv_category_description

The description of the category (given by cat_id) expressed in the language whose code is given ny (lang_id). This is visible to the user.

CUSTOMER

The customer table holds the data considered to be personal and 'singular' for a particular person. This includes: name, email_address and username. Additionally, a pseudonym is held to represent the person in book reviews. This is the first of the customer-related tables that must be populated when the application sets up a customer account.

cust_id

The numerical id that represents the customer in the database. There is no foreign key, this field is the source of customer ids in the database. It is the primary key of the CUSTOMER table.

username

The log-in username of the customer. The site should decide - in what form this username should be created. (e.g. will it be an email address or a user-created name). The application will be responsible for validating new usernames and setting passwords when allowing a new user to register with the site. As this is likely to be a field that the application needs to search on, this field has been given a separate index to support indexed lookup during log-in, and perhaps new-account username uniqueness checks.

password_hash

In the absence of any other nominated authentication method, the password_hash is provided to save hashed-password data for the purpose of validating user login attempts. The application will be responsible for the use made of this field. If external authentication is used, then this field can be left NULL.

account_created

This field is set with the date and time of the account record creation. It's value defaults to the current SYSDATE of the system on which the database is running. It does not need to be referenced in the initial record-creation (INSERT) statement as a default value is defined at the table leve. It is a DATE field, holding time resolution to the nearest second.

account_status

Account status is used to set the 'status' of the customer account with checked values of: 'REGISTERED','CONFIRMED','VERIFIED','QUARANTINED'. The application should verify the status of a customer before accepting orders. Though a set of account status values are given here, the

customer is best placed to advise on their own requirements, at which time the values can be adjusted.

Field Values and their meanings

> REGISTERED: A customer has just registered with the site by entering their name and email address. The email address had not yet been confirmed.
>
> CONFIRMED: A customer has confirmed, via a confirmation process, that the email address entered is in their control.
>
> VERIFIED: A customer has made a successful purchase though a payment card. The address given during purchase for the Billing address was validated by the payment card interaction.
>
> QUARANTINED: The customer has been placed into a quarantine state to prevent further purchase attempts. This would be done by the support team if suspicious activity was detected.

email_address

The email address of the customer.

ue_title

The customer's title. (e.g. Mr, Mrs, Miss, Dr etc.)

ue_first_name, ue_middle_initials, ue_last_name

The customer's first, middle and last names.

ue_review_pseudonym

A Pseudo-name that the customer can choose to represent them when writing reviews for books. As this should be a unique identifier for individuals, there is a UNIQUE index created on this column to give a fast method of checking for intended pseudonym usage.

CUSTOMER_ORDERS

Each customer order is unique, and represents a purchase of one or more items from the bookstore, these are detailed in ORDER_ITEMS.

order_id

A unique identifier to represent an order. An order being a complete set of purchased items made in one payment transaction. One order may comprise multiple items. A sequence generator is provided to support the generation of this value SEQ_ORDER_ID.

cust_id

The numeric id of the customer making this order, this is a foreign key into the CUSTOMER table and defines the customer who made the order.

date_order_placed

This is the date and time that the user placed an order. For the avoidance of ambiguity, this field is of type TIMESTAMP WITH TIMEZONE and is set on record creation (using a default column value of SYSDATE), it will not require to be set in the row's creation 'INSERT' statement The locale of this data and time information will be that of the current system time of the database server. It may be necessary to convert this into time data more relevant to the locale from which the customer has placed this order, this is easier to do if TIMEZONE is a part of the data type.

total_order_price

The total price of all items in the ORDER_ITEMS table for this order. (Each multiplied by their respective Quantities, of course). This data must be maintained and calculated by the application.

payment_method_id

The numeric identifier of the payment method, chosen by the customer as the means of payment for the order. (See PAYMENT_METHODS.)

payment_status

The status of the payment as defined by the payment system. This is left as free-text in the absence of any known set of values from these systems at this time. It is suggested that these are validated in some form, (check-constraint or lookup table), before commissioning the taking of payments. This would require additional work to configure this level of validation.

card_number

The (suggested) encrypted version of the payment card used to make the payment for this order. This may additionally be used in the event of a refund. The size of this field is 64 bytes which should be enough for a reasonable encryption system to render a reversibly encrypted version of the payment card. It is advised to perform encryption outside the database for this field. It is not advised to save unencrypted card details.

expiry_date

The expiry date for the payment card used for payment of this order. The format will be 'MMYY' in text form.

shipping_addr_id

The numeric id of the address (customer_addresses) for the shipping address to be used to fulfil this order.

billing_addr_id

The numeric id of the address (customer_addresses) for the billing address to be used to fulfil this order.

CUSTOMER_ADDRESSES

Overview

A customer (or user) will be able to have multiple addresses saved on their account. Each of which is nominated as a Billing address or a Shipping address. Within each type, the application should ensure that there is only one such address defined as the default, as this can then be displayed to the users when making an order. Other addresses can be offered for selection if the default is unsuitable for a particular order.

For security reasons, the application developers should consider preventing a user from changing address-records which have been validated against payment options. Shipping is not part of this schema design, but consideration should be given to this aspect to prevent card-fraud.

addr_id

The numerical id for the address record. This is the primary key for this table.

cust_id

The numerical id for a customer. This value is used to associate the customer with other related data internally.

display_order

When a user requires several addresses, as is permitted in the data model, this field is used to define the preferential display order of these addresses when the user is editing their profile data.

address_type

A single character to depict a Billing Address ‘B’ or a Shipping Address ‘S’. These values are value-checked in the database.

default_address

A Y/N field value to signify that this is the default address to be used for shipments. It is up to the application (or a trigger can be written) to enforce the singularity of a ‘Y’ setting amongst multiple address variations.

ue_address_line_1

The first line of the users address.

ue_address_line_2

The second line of the users address.

ue_address_line_3

The third line of the users address.

ue_address_line_4

The fourth line of the users address.

ue_address_city

The user’s City

ue_address_region

The user’s region (county or state)

ue_address_country

The user’s country of residence.

ue_address_postzip_code

The user's postcode or Zip code for their address.

ORDER_ITEMS

Individual books which form part of an order. Each distinct book requires a row in this table, but multiple books of the same ISBN number can be defined using quantity_ordered.

order_id

The unique numeric id that identifies the order of which this item is a part.

Item_id

The item id. This is a sequential number starting at 1 for the first item and incrementing upwards. It signifies the unique item (in this order). Its range is 1..99999

book_id

The id of the book represented by this order item.

quantity_ordered

The number of books of this (book_id) that is being ordered at this time.

price_when_ordered

The price that was relevant at the time this order was placed. As the price can vary over time, its value at the time of the order is made, is captured and saved in this field.

CUSTOMER_REVIEWS

This table holds the set of book reviews which have been written by customers themselves. There is basic moderation workflow provisioned, which may require consultation with the client to extend, or adapt to their intended way of working.

cust_id

The customer id of the person authoring the book review.

book_id

The id of the book being reviewed.

Moderation_status

The status of this book review as defined by the moderator. It's values are:

PENDING - awaiting moderator approval.

CLEARED - cleared for display by the moderator.

REJECTED - rejected by the moderator.

Only 'CLEARED' reviews should be displayed. REJECTED should be passed back to the customer for amendment.

Moderation_comment

The comment that the moderator may choose to make to explain why a review has been rejected.

Date_last_edited

The date of the last review edit. Once 'CLEARED" a review should not be editable by the customer.

Ue_review_text

The actual review text, written by the customer on the book. There is only one language that a review can be expressed in, or rather, there is only one review text that the customer can create for a book.

LANGUAGES

This table is used to define the set of languages that the translation services are able to process.

lang_id

The numeric id of the language. This is used internally to express language choice in other tables.

language_name

The name of the language.

domain_id

An optional domain_id that may be used by the web-services to associate a language with a particular web-domain.

PAYMENT_METHODS

This table is used to define the set of payment methods that the bookstore is able to accept from the customer.

payment_method_id

The numeric id for this payment method.

payment_method_code

A ‘back-office’ human readable code for the payment method, limited to 6 characters.

uv_payment_method_name

The name of the payment method, as may be made visible to the user. No specific requirement for translation of this has been requested, but, it is advised to extend the data model to include this facility.

TRANSLATION_STATUS

This table is used to define a set of translation status values that will drive the translation process workflow. It is configurational in function.

Status_id

The coded id for the translation status. This is used in the *_INFO tables to identity a language for translation activities.

translation_status

The status of the translation work-flow activity. This table is being left for the client to define the steps in the workflow of the translation and verification and approval workflow.

References:

1. List of writing genres. (https://en.wikipedia.org/wiki/List_of_writing_genres)

   Used as a source of hierarchical writing genres, in the supplied sample data.