VEDO BOOKKEEPING SYSTEM

Comprehensive Transaction Logic Documentation

Generated: 11/10/2025

TABLE OF CONTENTS

1. INVOICE TRANSACTIONS

1.1 OVERVIEW

Invoices represent amounts owed by customers for goods or services provided. They follow double-entry accounting principles, affecting Accounts Receivable and Revenue accounts. Invoices can have credits applied, be paid through Receive Payment transactions, and support multi-currency operations.

1.2 DATABASE SCHEMA

Table: transactions (type: "invoice")

id - serial - Primary key, auto-incremented

reference - text - Invoice number (e.g., INV-1001)

type - transaction_type - Always "invoice" for invoices

date - timestamp - Invoice date

amount - double precision - Total amount in home currency

balance - double precision - Remaining unpaid balance

contactId - integer - FK to contacts (customer)

status - status - open, paid, partial, overdue

currency - varchar(3) - Foreign currency code

exchangeRate - decimal(18,6) - Exchange rate used

1.3 INVOICE CREATION FLOW

Step 1: Validate invoice data using invoiceSchema from shared/schema.ts

- Ensure contactId exists and is a valid customer
- Ensure reference number is unique
- Validate line items (at least 1 required)

Step 2: Calculate totals

- subTotal = sum of (quantity × unitPrice) for all line items
- taxAmount = sum of calculated tax for each line item
- totalAmount = subTotal + taxAmount

Step 3: Create transaction record

- Insert into transactions table with type = "invoice"
- Set balance = totalAmount (fully unpaid)
- Set status = "open"

Step 4: Generate ledger entries (double-entry bookkeeping)

a) Debit Accounts Receivable

- Debit: totalAmount
- Description: "Invoice #[reference]"

b) Credit Revenue Account(s)

- For each line item: Credit line item amount

c) Credit Sales Tax Payable (if tax applies)

- Credit: taxAmount

2. RECEIVE PAYMENT TRANSACTIONS

2.1 OVERVIEW

Receive Payment transactions record money received from customers to pay off invoices or create unapplied credits. They link to invoices via the payment_applications table, which is the source of truth for how payments are allocated.

2.2 PAYMENT CREATION FLOW

Step 1: Validate payment data

- Ensure contactId exists and is a customer
- Validate payment amount > 0
- Ensure paymentAccountId is a valid bank/cash account

Step 2: Calculate payment allocations

- User specifies which invoices to pay and amounts
- If payment > allocated: remainder becomes unapplied credit

Step 3: Create payment application records

- For each invoice being paid:
  - INSERT INTO payment_applications
  - paymentId: ID of this payment
  - invoiceId: ID of invoice being paid
  - amountApplied: Amount allocated to this invoice

Step 4: Generate ledger entries

a) Debit Bank/Cash Account

- Debit: total payment amount

b) Credit Accounts Receivable (for each invoice)

- Credit: amountApplied

Step 5: Calculate FX Gains/Losses (multi-currency)

• If invoice currency "` home currency:

- Compare invoice vs payment exchange rates

- Create fx_realizations record for audit trail

3. DEPOSIT TRANSACTIONS

3.1 OVERVIEW

Deposits record money deposited into bank accounts. They can represent direct income deposits, customer payments, or unapplied credits that can be applied to invoices later.

3.2 DEPOSIT BALANCE CONVENTION

Deposits representing unapplied credits store balance as NEGATIVE:

- $5,000 deposit !' balance = -$5,000
- This indicates $5,000 available to apply to future invoices
- When $2,000 applied !' balance becomes -$3,000
- When fully applied !' balance = $0, status = "completed"

4. SALES RECEIPT TRANSACTIONS

4.1 OVERVIEW

Sales Receipts represent immediate cash sales where payment is received at the time of sale. Unlike invoices (which create receivables), sales receipts record both the revenue and the cash receipt in a single transaction.

4.2 KEY DIFFERENCES

Invoice:

- Creates Accounts Receivable
- Records revenue, then later payment reduces AR
- Has balance to track

Sales Receipt:

- Records revenue AND cash simultaneously
- No AR involved
- Status always "completed"
- Cannot have payments applied to it

5. CUSTOMER CREDIT TRANSACTIONS

5.1 OVERVIEW

Customer Credits (Credit Memos) are issued to customers to reduce their account balance. Common scenarios include product returns, service adjustments, billing corrections, and goodwill gestures.

5.2 CREATION FLOW

Step 1: Create customer credit transaction

- amount = total credit amount
- balance = amount (fully unapplied initially)
- status = "unapplied_credit"

Step 2: Generate ledger entries

a) Debit Sales Revenue (reverses revenue)

- Debit: credit amount

b) Credit Accounts Receivable (reduces customer balance)

- Credit: credit amount

Step 3: Make credit available for application

- Credit appears in customer's available credits list
- Can be applied to existing or future invoices

6. JOURNAL ENTRY TRANSACTIONS

6.1 OVERVIEW

Journal Entries are manual accounting entries that allow direct manipulation of the general ledger. They are used for adjusting entries, depreciation, equity transactions, and any transaction not covered by standard transaction types.

6.2 CRITICAL VALIDATION

Total Debits MUST EQUAL Total Credits (within 0.001 tolerance)

Valid Entry:

- Entry 1: Debit Rent Expense $1,000
- Entry 2: Credit Cash $1,000
  - Result: ' Accepted

Invalid Entry:

- Entry 1: Debit Rent Expense $1,000
- Entry 2: Credit Cash $900
  - Result: ' Rejected with error

7. PAYMENT APPLICATIONS TABLE

7.1 OVERVIEW

The payment_applications table is the SOURCE OF TRUTH for tracking which payments have been applied to which invoices. This creates a many-to-many relationship allowing one payment to be split across multiple invoices and one invoice to receive payments from multiple sources.

7.2 TABLE SCHEMA

id - serial - Primary key

paymentId - integer - FK to transactions (payment type)

invoiceId - integer - FK to transactions (invoice type)

amountApplied - double precision - Amount allocated

createdAt - timestamp - When application was created

7.3 USAGE EXAMPLE

Single Payment to Multiple Invoices:

Payment: PAY-002 for $5,000

Invoices:

- INV-1002: $2,000

- INV-1003: $1,500

- INV-1004: $1,000

Unapplied: $500

Payment Applications Created:

• Record 1: paymentId=6, invoiceId=11, amountApplied=$2,000

• Record 2: paymentId=6, invoiceId=12, amountApplied=$1,500

- Record 3: paymentId=6, invoiceId=13, amountApplied=$1,000

Result:

- Payment has $500 unapplied (balance $500)
- All 3 invoices fully paid (balance $0)

8. DELETION CASCADE LOGIC

8.1 GENERAL PRINCIPLES

1. Atomic Operations

- All deletions wrapped in database transactions
- If any step fails, entire operation rolls back

2. Cascade Restoration

- Restore all linked transactions to pre-deletion state
- Update balances and statuses accurately

3. Complete Cleanup

- Delete from payment_applications first
- Then delete ledger_entries
- Then delete line_items
- Finally delete the transaction itself

4. Account Balance Reversal

- Reverse all ledger entry effects on account balances
- Follow account type rules (debit/credit nature)

8.2 DELETION ORDER OF OPERATIONS

1. Validation - Verify transaction exists
2. Find Linked Transactions - Query payment_applications

3. Restore Linked Balances - Add back applied amounts

4. Delete Link Records - Remove payment_applications

5. Reverse Account Balances - Update all affected accounts

6. Delete Ledger Entries - Remove accounting entries

7. Delete Line Items - Remove transaction details

8. Delete Transaction - Remove main record

9. Recalculate Balances - Verify data integrity

10. Return Result - Provide feedback on restoration

END OF DOCUMENTATION

For questions or clarifications about this documentation,
please refer to the Vedo codebase or contact the development team.