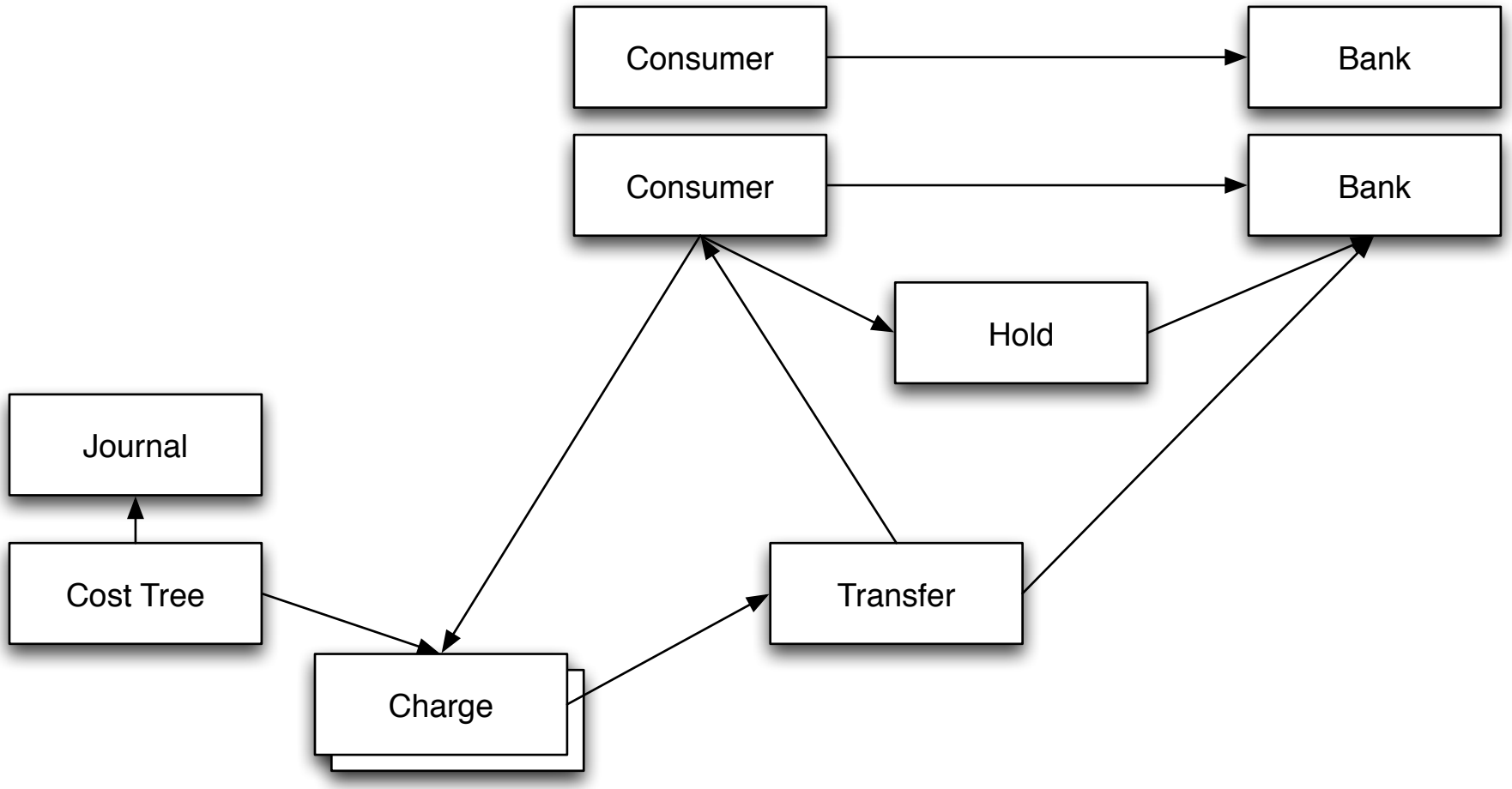
Consumers regularly need to charge money to their associated banks as part of their every day operations.  Charges result in two immediate changes: a transfer is added to the ledger's transfer table, and a charge is added to the current journal's cost tree.  Both the charge and the transfer refer to one another.  Every charge has one transaction, and every transaction has one charge.

The journal may be closed and replaced by a "fresh" journal at times, to optimize search speed. When this happens, it should not have any effect.

Holds, similar to transfers, can be placed against a bank by a consumer.  When and if they are executed, the hold is deleted in the same transaction in which a transfer is made.

A bank's effective "available balance" is its original balance minus all transfers and holds.



Consumers sometimes need to charge money so that they can set up banks. When they do this, they add charges to the currently open invoice, opening one if needed. The invoice is later (presumably very soon thereafter) closed and sent to the customer who pays for the invoice. The payment is added to the ledger as a refundable credit, and this payment triggers payment processing. During payment processing, credit applications are gathered from existing credits and used to pay invoices in full.

There is some algorithm for ordering credits to apply. First we apply nonrefundable credit, then older payments, and so on. This should rarely be of interest.

It's okay for credits to sit around with unapplied balances. They'll get used later, or turned into refunds, if applicable. Refunds are just applications of another sort.

When an invoice gets paid, it doesn't have to actually create a bank and associate it with anything. For example, there is not *necessarily* any reason for a payment for a domain registration to create a bank.

There may be more than one open invoice at a time. When the customer is notified of his invoice, the newest invoice is displayed in detail, and others as summaries or virtual line items.