Story UFC-195: Transactions Type Filtering

The following document will go over the design of transactions type filtering, as outlined in Story ticket UFC-195.

Acceptance Criteria

For reference, the following are the acceptance criteria for this ticket:

- I can see a set of buttons on the Transactions page, one for each type.
- In each type button, I can see the total amount of money for the type, for the currently selected date range.
- In each type button, I can see a 'From' amount which is the total amount for these types from the date range interval before the current date range.
- In each button, I can see a percentage representing the difference between the current amount and the 'From' amount.
- I can see that all of these buttons are selected (by default).
- I can click a button to deselect it and remove the filter for that type of transaction from the Transactions list/table.

Design Brainstorming

Account Balances Date Index

So this'll be the ticket where we finally implement the `Account Balances Date Index`, that was described in detail as part of Story UFC-199: Transactions Date Filtering.

In terms of implementation, I was thinking that we could leverage the `DateIndex` structure we built back for the transactions date index, but now that I think about it, I think this new balances date index is ever so slightly different enough that we can't use it 1-1. This is because the new index stores a *balance* as the bottom level data type, whereas the old one stored *ids*.

This might not seem like a big deal (the structure itself doesn't change much; just need to store a number instead of a list of IDs), but it completely changes the patterns used for modifying and accessing the structure.

This is mostly problematic because we don't have a list at the bottom of the data structure any more. Instead of operating on a list of IDs (e.g. for removable), we actually now care more for operating at the day level.

So... are we gonna have to create a more generic 'DateIndex' structure, and then create specialized versions for the TransactionsDateIndex and AccountBalancesDateIndex?

You know **what**? **Screw this**. I'm just gonna calculate the balances on the fly, every time. Cause fuck it. Nobody's gonna care, nobody's gonna notice, it's probably gonna be fast enough. So whatever. Linear time it is.

Maybe we can just add a bigger cache to the selectors, so that we at least get speedups for exact calculations that we've already done. Looks like the `re-reselect` library might be good here.

Wait a minute, we didn't even *need* the account balances index for this story!!! The Transaction Type cards don't show asset or liability balances. So **screwing this** was the correct play regardless!!!

Type Filter Hook

Other than that, we'll also need a new Context for holding the type filters. This hook will be of the 'pagination' type, whereby it'll take in the transactions and filter them as opposed to generating transactions itself; that is, it'll be a part of the `useFilteredTransactions` pipeline.

It should be called `useTransactionTypes`.

Update: Welp, everything's wrong. And by everything, I mean the notion of the `useFilteredTransactions` pipeline. Let me explain.

So before this story, the pipeline consisted of just paginating (and ID mapping, but that's irrelevant). But now we want to add in type filtering. Only problem is that the Pagination provider requires the transactions as a prop to get the length.

So yeah, there's *literally* no way that the 'all in one' pipeline could work. We now have to split each hook up into its own so that we can do the type filtering at the Transaction scene level so that the Pagination provider receives the right number of transactions.

Date Range Amounts Hook

We'll also need a new hook that leverages the `useDateRange` Context hook to take the date range and select the account balances (From, Current) for the Type cards.

Since we need the balances by transaction type, this hook should be called `useDateRangeTransactionAmountsByType`.

New Type Card

Speaking of which, we won't need many components for this story. Really, it's *just* the new Type cards.

- Hmm, we have the existing `TransactionTypeOption`, which is technically what these new components are, so are these 'new' Type cards just a different variant?

- And if they are a different variant, how do we represent that? As a `variant` prop? As a flag prop like `withAmounts`? Or just implicitly by passing `fromAmount` and `currentAmount`?
- I think it shouldn't be a `variant` prop, since I don't expect there to be more than these 2 'variants'. So I think we just go with the implicit `fromAmount` and `currentAmount`.
- And we don't need to pass in the percentage; it can be calculated at the component level.

Also, we can break down the new layout into smaller atoms (AmountChange, CurrentAmount, FromAmount) so that we can reuse them in the Account Summaries later.

Component Breakdown

Atoms

- AmountChange (calculates the percentage from two values and color codes them accordingly; needs an option to display the difference as well)
- CurrentAmount (displays an amount with the differently colored dollar sign)
- FromAmount (displays an amount with the 'from' prefix)

Molecules

- [modification] TransactionTypeOption (add new balance props with new layout; use AmountChange, CurrentAmount, and FromAmount)

Organisms

- TransactionTypeFilters (TransactionTypeOption x4, grab types from `useTransactionTypes`, set types for `useTransactionTypes`, grab from/to balances from `useDateRangeTransactionAmountsByType`)

Scenes

- [modification] Transactions (add TransactionTypeFilters)

Tasks

- Build the `useTransactionTypes` Context hook.
- Build the `useDateRangeTransactionAmountsByType`.
- Build the `TransactionTypeFilters` and subcomponents.