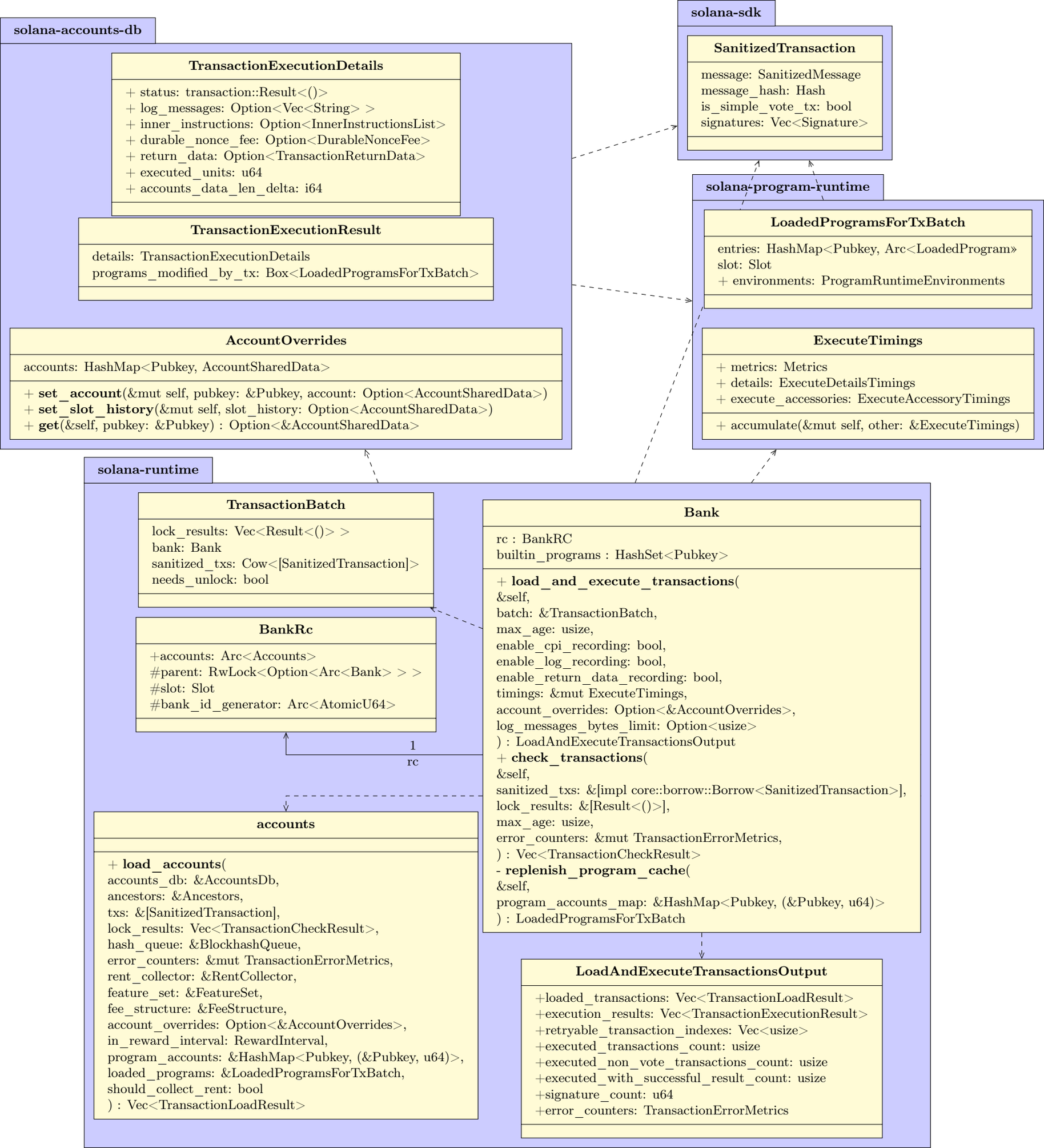
solana-accounts-db
solana-sdk
SanitizedTransaction
message: SanitizedMessage
message_hash: Hash
is_simple_vote_tx: bool
signatures: Vec<Signature>
TransactionExecutionDetails
+ status: transaction::Result<()>
+ log_messages: Option<Vec<String> >
+ inner_instructions: Option<InnerInstructionsList>
+ durable_nonce_fee: Option<DurableNonceFee>
+ return_data: Option<TransactionReturnData>
+ executed_units: u64
+ accounts_data_len_delta: i64
TransactionExecutionResult
details: TransactionExecutionDetails
programs_modified_by_tx: Box<LoadedProgramsForTxBatch>
solana-program-runtime
LoadedProgramsForTxBatch
entries: HashMap<Pubkey, Arc<LoadedProgram»
slot: Slot
+ environments: ProgramRuntimeEnvironments
AccountOverrides
accounts: HashMap<Pubkey, AccountSharedData>
+ set_account(&mut self, pubkey: &Pubkey, account: Option<AccountSharedData>)
+ set_slot_history(&mut self, slot_history: Option<AccountSharedData>)
+ get(&self, pubkey: &Pubkey) : Option<&AccountSharedData>
ExecuteTimings
+ metrics: Metrics
+ details: ExecuteDetailsTimings
+ execute_accessories: ExecuteAccessoryTimings
+ accumulate(&mut self, other: &ExecuteTimings)
solana-runtime
TransactionBatch
lock_results: Vec<Result<()> >
bank: Bank
sanitized_txs: Cow<[SanitizedTransaction]>
needs_unlock: bool
Bank
rc : BankRC
builtin_programs : HashSet<Pubkey>
+ load_and_execute_transactions(
&self,
batch: &TransactionBatch,
max_age: usize,
enable_cpi_recording: bool,
enable_log_recording: bool,
enable_return_data_recording: bool,
timings: &mut ExecuteTimings,
account_overrides: Option<&AccountOverrides>,
log_messages_bytes_limit: Option<usize>
) : LoadAndExecuteTransactionsOutput
+ check_transactions(
&self,
sanitized_txs: &[impl core::borrow::Borrow<SanitizedTransaction>],
lock_results: &[Result<()>],
max_age: usize,
error_counters: &mut TransactionErrorMetrics,
) : Vec<TransactionCheckResult>
- replenish_program_cache(
&self,
program_accounts_map: &HashMap<Pubkey, (&Pubkey, u64)>
) : LoadedProgramsForTxBatch
BankRc
+accounts: Arc<Accounts>
#parent: RwLock<Option<Arc<Bank> > >
#slot: Slot
#bank_id_generator: Arc<AtomicU64>
1
rc
accounts
+ load_accounts(
accounts_db: &AccountsDb,
ancestors: &Ancestors,
txs: &[SanitizedTransaction],
lock_results: Vec<TransactionCheckResult>,
hash_queue: &BlockhashQueue,
error_counters: &mut TransactionErrorMetrics,
rent_collector: &RentCollector,
feature_set: &FeatureSet,
fee_structure: &FeeStructure,
account_overrides: Option<&AccountOverrides>,
in_reward_interval: RewardInterval,
program_accounts: &HashMap<Pubkey, (&Pubkey, u64)>,
loaded_programs: &LoadedProgramsForTxBatch,
should_collect_rent: bool
) : Vec<TransactionLoadResult>
LoadAndExecuteTransactionsOutput
+loaded_transactions: Vec<TransactionLoadResult>
+execution_results: Vec<TransactionExecutionResult>
+retryable_transaction_indexes: Vec<usize>
+executed_transactions_count: usize
+executed_non_vote_transactions_count: usize
+executed_with_successful_result_count: usize
+signature_count: u64
+error_counters: TransactionErrorMetrics