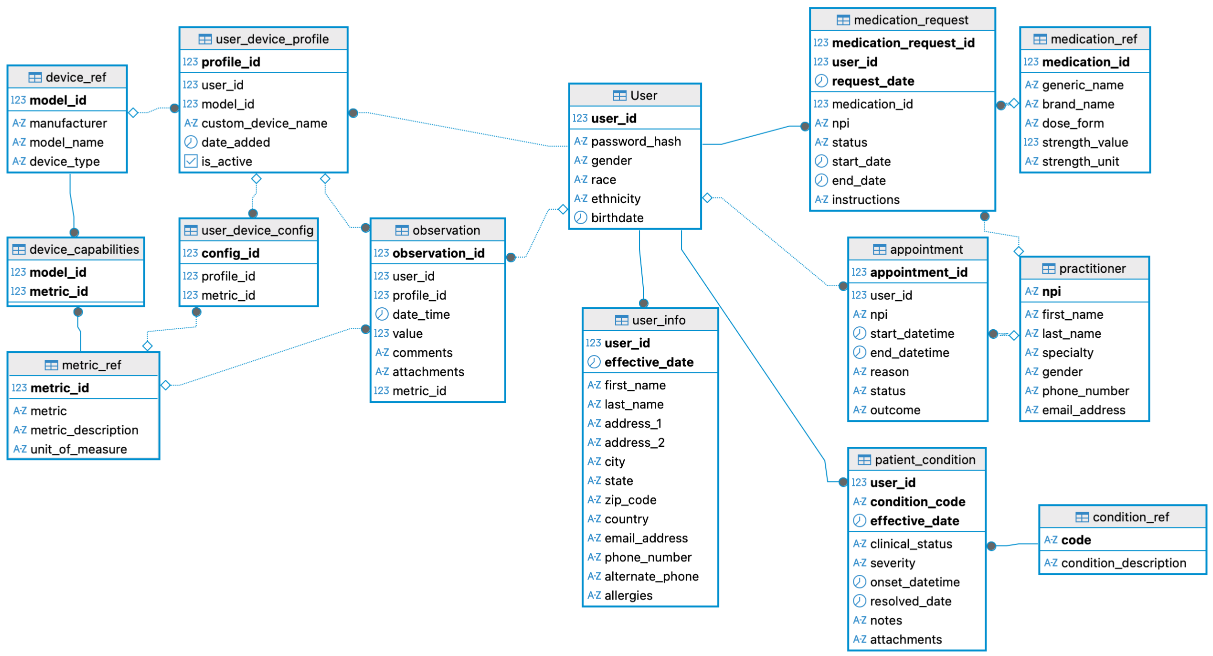
Alec Torres

Business problem

The business problem is that the current data model is missing the infrastructure to accommodate the type of that the devices output. As a result, I created a whole new structure that is able to accommodate both the device data as well as EHR style records and opted for a vertical style capturing of the device data to prioritize analysis. This will ensure scalability and enable real-time analysis, both of which are business requirements of HealthTrack.

Logical Data Structure



The data model can be split into two distinct groups that capture their own respective forms of data. The clinical group captures typical data you would from EHR records that would be imported from other companies. This is allow for integrating with health care providers. The second structure is the transactional capture of health metrics associated with wearables.

The reference tables create a standardized catalogs to enable data integrity of domain knowledge and thus systemic stability. It allows for convenient modifications to the reference tables in a constantly changing field in terms of market pressure and medical innovations or paradigm shifts.

The flow of how the health metrics vertical structure works was envisioned for analytics as follows:

From device_ref and metric_ref you get device capabilities, a mapping of what each device is capable of. Then, user_device_profile captures what specifically devices each user has and from there, user_device_config becomes populated.

User_device_config is the solution I came up with to account for the custom_device_name of user_device_profile. When entering a custom device, it would allow for the end user to select what metrics the device captures. And then for a model that has been standardized in the system, I imagine that this table would be auto populated when it is selected by the user. This essentially allows for the system to account for unknowns while still maintaining referential integrity for the known devices.

Furthermore, to clarify, within user_device_profile there is constraint added that the user must either add a model_id OR a custom_device_name. This is to account for unknown or custom devices as well as gain the ability to streamline introduction of new devices from the market. Periodically a data admin could task an AI agent to analyze the custom_device_name column for patterns in names, cross reference them on the web, and then standardize the naming and metric combination into device_ref and device_capabilities.

Essentially, device_ref, metric_ref and device_capabilities are managed by an admin/system, user_device_profile by the user, and user_device_config a combination of system/user

Finally, all of this is culminated in the observation table. Device_ref, metric_ref and user_device_profile define hard rules through foreign keys while device_capabilities and user_device_config define the rules that govern what should be flowing into the observation table. I envisioned that what data actually gets captured into observation is enforced in the application layer, not the schema. This allows for observation to be loose at write time so that discrepancies at capture, due to device failure or if a device captures a metric not known to be possible by the system, a crash doesn't happen.

At this time it is important to note that user_device_config exists essentially for the application layer. Observation doesn't rely on it in that it can accept any metric that exists in metric_ref. I imagined that this table would aid in the building of a UI that filters and displays either what the user created (custom device) or what the device is known by the system to capture.

Privacy and security outline

Private and security concerns that should be implemented are:

- Access control
    - Distinct roles with minimum necessary privileges
    - Row level security to protect password/username, PHI
- Encryption & Authentication
- Logging audits
- HIPAA specific requirements