Fiora Xu

Requirements Document for Delivery Driver Payroll Database

Selected Features for In-Memory Storage:

Current Driver Sessions

Purpose: To track currently logged-in drivers. This can be used to handle session management and provide quick access to the driver's current state without querying a relational database continuously.

Redis Data Structure: Hash. Each driver session can be stored in a hash with keys as driver_id and values containing session details like last active time, current route, and performance metrics.

Real-Time Route Tracking

Purpose: To track the real-time progress of drivers on different routes. This can include the start time, current location, and status of each route.

Redis Data Structure: Sorted Set. Each entry can have a route_id as the member and the timestamp or distance completed as the score for real-time tracking and easy retrieval of route progress.

Performance Metrics Cache

Purpose: To quickly access and update drivers' performance metrics for the current day, including stops completed, packages delivered, and time efficiency.

Redis Data Structure: Hash. Store each driver's daily performance metrics in a hash keyed by driver_id and date, allowing quick updates and retrievals.

Problem Domain Language Rules:

1. **Driver identification** : Every driver is uniquely identified by a driver id and has associated personal details such as name and contact information. A driver has a specific base pay id that links to their base pay rate.
2. **Stop Classification** : Each delivery stop is categorized by its type (residential, commercial) and is linked to a specific route.
3. **Time Tracking** : The time taken for each route by a driver on a specific date is recorded, including total time spent.
4. **Route Difficulty Assessment** : Each route is assessed for difficulty, which is a factor in determining pay rates.

5. **Package Volume Tracking** : The total number of packages delivered to each stop on a specific date is recorded.
6. **Base Pay Determination** : Base pay is determined by a base rate, a package per day threshold, and an extra rate for packages delivered beyond this threshold.
7. **Performance Evaluation** : Drivers' daily performance is evaluated based on the number of stops completed, packages delivered, and time efficiency relative to expectations.
8. **Bonus Allocation** : Bonuses are awarded based on specific criteria, including performance reasons, on a given date and are associated with a driver.

Identified Nouns and Actions:

Nouns:
- Driver (driver_id, name, contact, base Pay_id)
- Stops (Stop_id, Route_id, Stop Type)
- Time Taken (TimeTaken_id, Route_id, Driver_id, Date, Total Time)
- Route Difficulty (Route_id, Difficulty)
- Amount of Packages (Packages_id, Stop_id, Date, Total Packages)
- Base Pay (Base Pay_id, Base Rate, Package Threshold, Extra Rate Per Package)
- Performance (Performance_id, Driver_id, Date, Route_id, Stops Completed, Packages Delivered, Time Efficiency)
- Bonuses (Bonus_id, Driver_id, Date, Reason, Amount)

Verbs:
- Driver:
    - delivers packages
- Assigning base pay and package thresholds to drivers.
- Recording the total time taken per route by drivers.
- Categorizing stops by type (residential, commercial).
- Assessing route difficulty.
- Tracking the number of packages delivered per stop.
- Evaluating drivers' performance based on stops completed, packages delivered, and time efficiency.
- Awarding bonuses based on specific criteria related to performance.

Given the entities:

1. **Driver** (driver_id, name, contactInfo, basepay_id)
2. **Stops** (stop_id, route_id, stop_type)
3. **TimeTaken** (timeTaken_id, route_id, driver_id, date, totalTime)
4. **Route** (route_id, difficulty)

5. **AmountOfPackages** (stop_id, date, total_packages)
6. **BasePay** (basepay_id, base_rate, package_threshold, extra_rate_per_package)
7. **Performance**(performance_id, driver_id, date, stops_completed, packages_delivered, time_efficiency)
8. **Bonuses**(bonus_id, driver_id, date, reason, amount)

Functional Dependencies (FDs)

FDs for each entity:

1. Driver: driver_id → name, contactInfo, basepay_id
2. Stops: stop_id → route_id, stop_type
3. TimeTaken: timeTaken_id → route_id, driver_id, date, totalTime
4. Route: route_id → difficulty
5. AmountOfPackages: packages_id → stop_id, date, total_packages
6. BasePay: basepay_id → base_rate, package_threshold, extra_rate_per_package
7. Performance: performance_id → driver_id, date, route_id, stops_completed, packages_delivered, time_efficiency
8. Bonuses: bonus_id → driver_id, date, reason, amount

Relational Schema in BCNF

1. Driver(driver_id PK, name, contactInfo, basepay_id FK)
   - FD: driver_id → name, contactInfo, basepay_id
   - All attributes are functionally dependent on the primary key, so it's in BCNF.

2. Stops(stop_id PK, route_id FK, stop_type)
   - FD: stop_id → route_id, stop_type
   - In BCNF as stop_id is a superkey.

3. TimeTaken(timeTaken_id PK, route_id FK, driver_id FK, date, totalTime)
   - FD: timeTaken_id → route_id, driver_id, date, totalTime
   - timeTaken_id is a superkey, satisfying BCNF.

4. Route(route_id PK, difficulty)
   - FD: route_id → difficulty
   - route_id is a superkey, so it's in BCNF.

5. AmountOfPackages(packages_id PK, stop_id FK, date, total_packages)
   - FD: packages_id → stop_id, date, total_packages

- packages_id is a superkey, in BCNF.

6. BasePay(basepay_id PK, base_rate, package_threshold, extra_rate_per_package)
   - FD: basepay_id → base_rate, package_threshold, extra_rate_per_package
   - basepay_id is a superkey, so it's in BCNF.

7. Performance(performance_id PK, driver_id FK, date, route_id FK, stops_completed, packages_delivered, time_efficiency)
   - FD: performance_id → driver_id, date, route_id, stops_completed, packages_delivered, time_efficiency
   - performance_id is a superkey, in BCNF.

8. Bonuses(bonus_id PK, driver_id FK, date, reason, amount)
   - FD: bonus_id → driver_id, date, reason, amount
   - bonus_id is a superkey, in BCNF.

Relationships

Based on the relational schema and the functional dependencies provided, here are the relationships between the entities:

1. Driver to BasePay: One-to-One (1:1)
   - Each `Driver` is assigned exactly one `BasePay` through `basepay_id`. Since `basepay_id` is unique for each `Driver`, this forms a one-to-one relationship.

2. Driver to Performance: One-to-Many (1:N)
   - A `Driver` can have multiple `Performance` records over time, indicated by `performance_id`. Each `Performance` record is unique to a `Driver` on a specific date, route, etc.

3. Driver to Bonuses: One-to-Many (1:N)
   - A `Driver` can receive multiple `Bonuses`, as each `bonus_id` is unique and can be awarded to the `Driver` for various reasons on different dates.

4. Driver to TimeTaken: One-to-Many (1:N)
   - A `Driver` can have multiple `TimeTaken` records, each representing the time taken for different routes on different dates.

5. Route to Stops: One-to-Many (1:N)

- A `Route` can have multiple `Stops`, as stops are parts of a route. Each `Stop` is uniquely identified and associated with a `Route`.

6. Route to TimeTaken: One-to-Many (1:N)
- A single `Route` can be associated with multiple `TimeTaken` records, as different drivers might take the same route on different dates.

7. Stop to AmountOfPackages: One-to-Many (1:N)
- A `Stop` can have multiple `AmountOfPackages` records, as the number of packages delivered can vary by date.

8. Route to Performance: One-to-Many (1:N)
- A `Route` can be associated with multiple `Performance` records through different drivers and dates.