Requirement Analysis of a Train Station

"Team Won"

We have chosen to model a railway station as a mini-world. The model will capture all of the operations which are required in a typical railway station.

The target users of the database are ticketing systems, employee management systems, train scheduling systems, etc. They will use the system to generate and cancel tickets, view train schedules, check late trains, manage employees and so on...

I Database Requirements

1 Entity Types

1. `train`:
   - `train_id`: integer (primary key)
   - `num_coach`: integer
   - `train_name`: string
   - `train_type`: `LOCAL | EXPRESS | SUPERFAST`
   - `last_maintenance`: date
2. `ticket`:
   - `ticket_number`: integer (primary key)
   - `date`: date
   - `train_number`: foreign key referencing `train`
   - `ticket_type`: `SLEEPER | AC | FIRST CLASS`
   - `seat`: composite attribute composed of `coach` (integer) and `berth` (integer)
   - `cost`: price of the ticket (derived attribute)
3. `building`:
   - `building_id`: integer (primary key)
   - `last_maintenance`: date
   - subclasses - food store, toilet, enquiry counter, ticket counter, etc.
4. `platform`:
   - `platform_number`: integer (primary key)
   - `usable_by`: multi-valued attribute with keys taken from `train.train_type`
   - `length`: integer
   - `last_cleaning`: date
5. `employee`:
   - `employee_id`: integer (primary key)
   - `name`: string
   - `dob`: date
   - `manager`: foreign key referencing `employee.employee_id`

- `salary`: integer

6. `passenger`:
   - `customer_id`: integer (primary key)
   - `ticket`: foreign key referencing `ticket`
   - `sex`: `M | F | O`
   - `dob`: date
7. `ticket_counter` (subclass of `building`):
   - `counter_id`: integer (primary key)
   - `employee_ids`: multivalued attribute referencing `employee.id` (foreign key)
   - `state`: `OPEN | CLOSED`
8. `engineer`, `station_master`, `cleaning_staff`, etc.
   - subclasses of `employee`

2 Weak Entities

1. Timetable: `train_number`: referencing `train.train_number` `arrival_time`: date `departure_time`: date `platform_number`: referencing `platform.platform_number`
2. Ticket (described above): is a weak entity dependent on `train.train_id`

3 Subclass

1. `Employee` = `ENGINEER | STATION MASTER | CLEANING STAFF | MANAGER` etc.
2. `Building` = `FOOD STORE | ENQUIRY | TOILET | OFFICE` etc.

4 Relationships

1. `ticket_counter` *sells* `ticket`s (one-to-many relationship)
2. `ticket_checker` *checks* `ticket`s (many-to-many relationship)
3. `manager` *supervises* `employee`s (one-to-many)
4. `cleaning_staff` `cleans` `platform`s (one-to-one)
5. `employee` *works* in a `building` (many-to-one)

5 Relationship of degree > 3

1. `passenger` *boards* `train` (other entities involved are `ticket`, `train` and `platform`)

6 Composite Attributes

1. `train_id`: `<train_number, train_name>`
2. `ticket.seat`: `<ticket.coach, ticket.berth>`

7 Multi-valued Attributes

1. `platform.usable_by`: one or more of `train.train_type`
2. `cleaning_staff.assigned_building`: one or more of `building.building_id`

8 Derived Attributes

1. `ticket.cost` derivable from `<train_id, train_type, seat, passenger.dob>` (senior-citizen discount)

II Functional Requirements

1. Generate new ticket. Parameters are `passenger` and `train` along with other ticket details.
2. Cancel ticket: cancels a given `ticket`. This requires us to refund the customer. To facilitate this, we add a new data requirement `payment_type: CASH | CARD | NETBANKING` to allow for refund processing. Thus, *the data requirements have been modified by a functional requirement.*
3. Add a new train to the system, or remove a seasonal train.
4. Hire, promote or fire employees; *change* employee type.
5. Update maintenance records of buildings and trains.
6. Generate the timetable (report).
7. Generate the number of tickets sold in a given month along with which trains had most sales (report).