Step 1: Identify Entities

Entity	Purpose	Attributes	Candidate Keys
Subscriber	Represents the customer/user.	subscriber_id, phone_number, full_name, registration_date, status	subscriber_id, phone_number
Call	The core record of the interaction.	call_id, caller_id, receiver_id, start_time, duration_seconds, call_type_id, tower_id	call_id
CallType	Categories for billing (Local, STD, ISD).	call_type_id, call_type_name, billing_rate_per_min, description	call_type_id, call_type_name
Tower	Infrastructure handling the signal.	tower_id, tower_name, region, latitude, longitude	tower_id

Step 2: Design Keys & Relationships

Relationship	Cardinality	Foreign Key	Required?
Subscriber → Call (Caller)	1:Many	Call.caller_id → Subscriber.subscriber_id	Yes
Subscriber → Call (Receiver)	1:Many	Call.receiver_id → Subscriber.subscriber_id	Yes
CallType → Call	1:Many	Call.call_type_id → CallType.call_type_id	Yes
Tower → Call	1:Many	Call.tower_id → Tower.tower_id	Yes

Primary Key Choice: I have chosen **Surrogate Keys** (integers) for all tables. While `phone_number` is a natural key, it can be reassigned or ported, whereas an internal ID remains immutable.

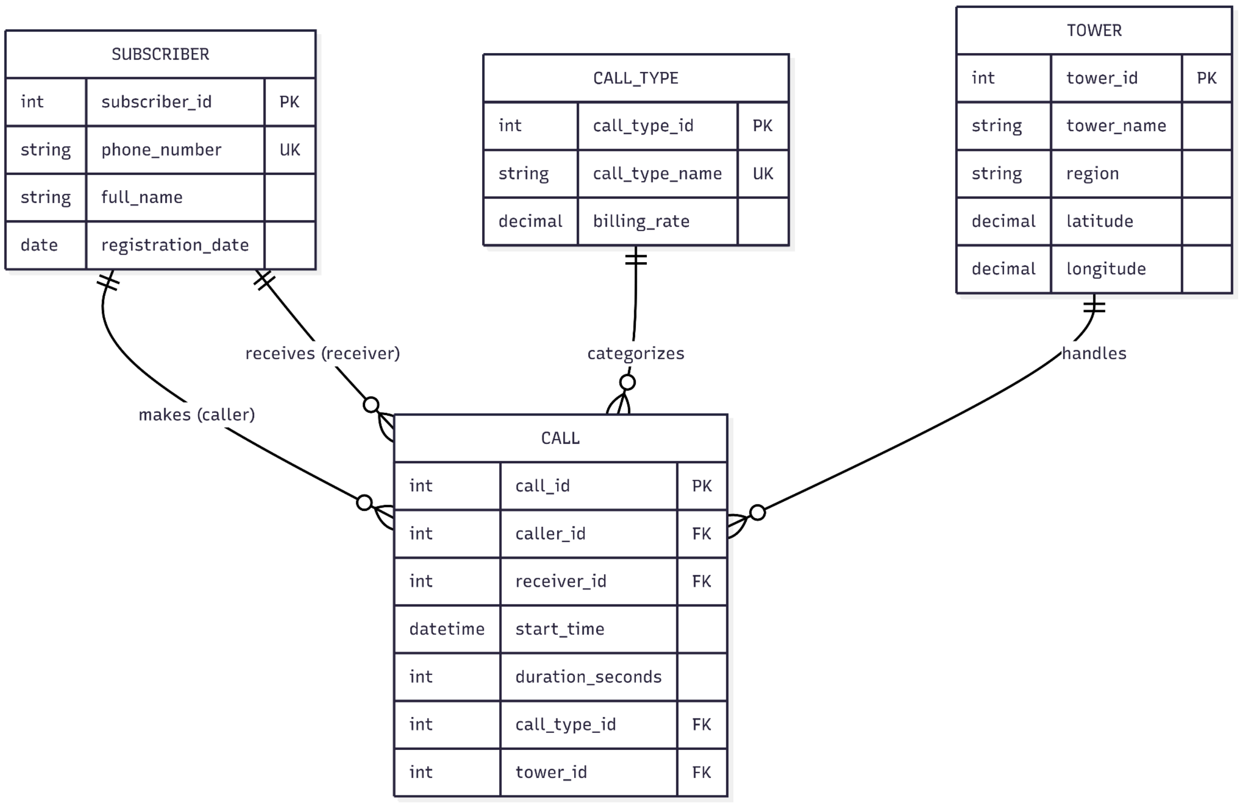
Step 3: Normalized Schema (3NF)

This schema ensures that subscriber details (like name) or tower details (like latitude) aren't repeated in every call record, saving millions of rows of redundant storage.

1. **Table: `subscribers`**
    - `subscriber_id` (PK), `phone_number` (Unique), `full_name`, `registration_date`
2. **Table: `call_types`**

    - `call_type_id` (PK), `call_type_name` (Unique), `billing_rate`
3. **Table: `towers`**
    - `tower_id` (PK), `tower_name`, `region`, `latitude`, `longitude`
4. **Table: `calls`**
    - `call_id` (PK), `caller_id` (FK), `receiver_id` (FK), `start_time`, `duration_seconds`, `call_type_id` (FK), `tower_id` (FK)

Step 4: ER Diagram Representation



SUBSCRIBER

int	subscriber_id	PK
string	phone_number	UK
string	full_name	
date	registration_date	

CALL_TYPE

int	call_type_id	PK
string	call_type_name	UK
decimal	billing_rate	

TOWER

int	tower_id	PK
string	tower_name	
string	region	
decimal	latitude	
decimal	longitude	

CALL

int	call_id	PK
int	caller_id	FK
int	receiver_id	FK
datetime	start_time	
int	duration_seconds	
int	call_type_id	FK
int	tower_id	FK

Step 5: Schema Justification

Why 3NF?

- Data Integrity: By separating `call_types` and `towers`, we ensure that if a tower's location is corrected, it updates for all historical calls immediately.
- Storage Efficiency: In a system generating millions of records daily, storing the string "International Standard Dialing" instead of an integer `3` would waste gigabytes of storage over time.

Denormalization Considerations

While the above is perfect for an OLTP (Transactional) system, for Analytics, we might denormalize:

- Materialized Views: We may create a view that flattens the `Subscriber` name and `CallType` into the `Call` record to speed up monthly billing reports.
- Partitioning: Because CDR data is time-heavy, we should partition the `calls` table by `start_time` (e.g., monthly partitions) to improve query performance.