University of Liège

Project 1

INFO0009 - Bases des données (Databases)

Yusuf Agirbas (s225049)
Ibrahim Touhami (s220915)
Duy Vu Dinh (s2401627)

Academic year 2024-2025

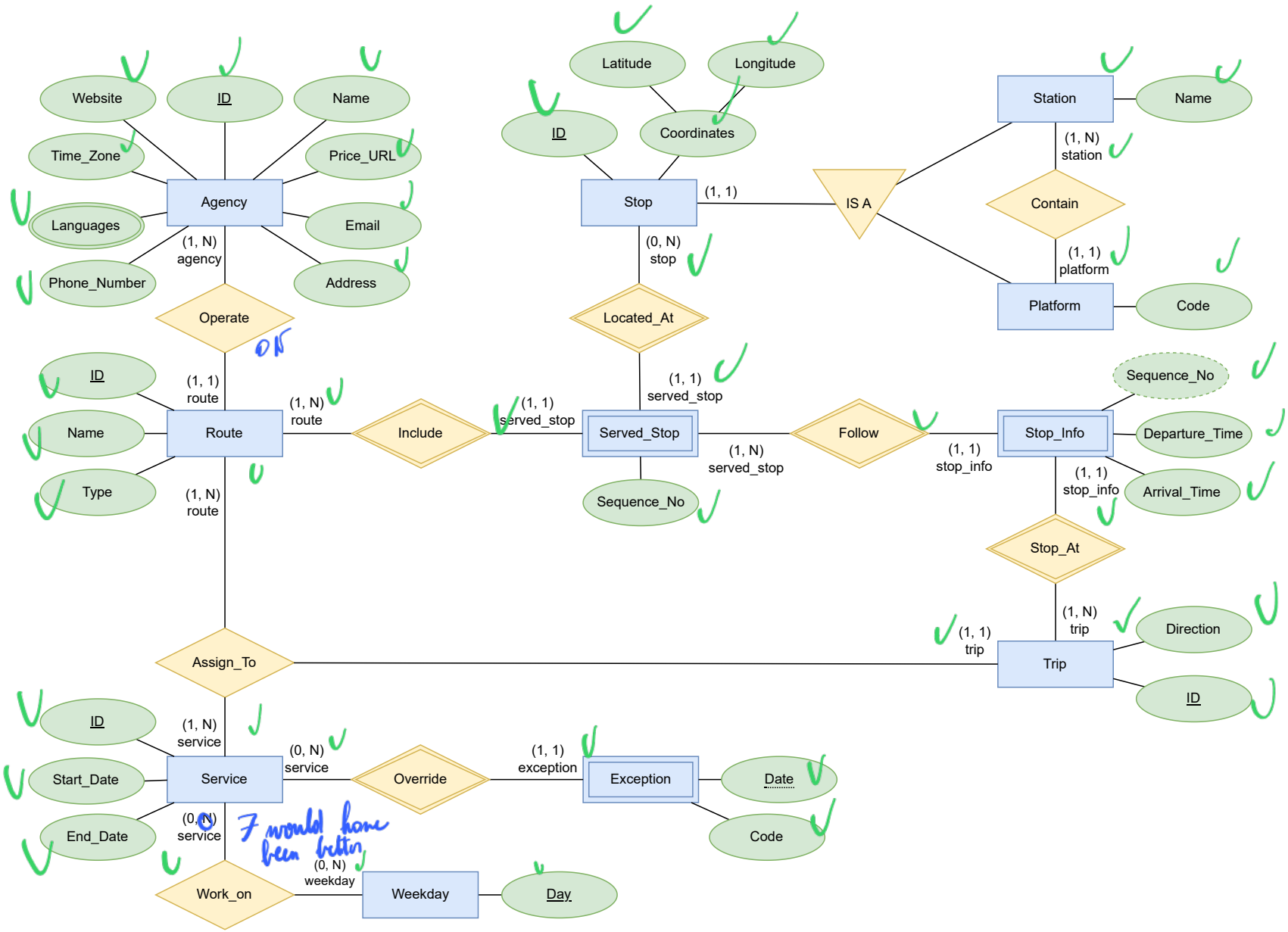

Figure 1: The proposed entity-relationship diagram.

Overview

The train schedule management system aims to provide a centralized, structured way to store, manage, and retrieve information about railway schedules. The system handles main components such as agencies, routes, stops, trips, and services while ensuring data integrity. This project provides the entity-relationship diagram, relational schema, integrity constraints, and normalization process to design a database for managing railway schedules.

1 Entity-relationship diagram (ERD)

Our proposed entity-relationship diagram for this project is shown in Fig. 1. Below is the justification and some assumptions.

- Agency
  - Represents an organization that operates railway services.
- Route
  - Represents a sequence of stops that define a service's path.
  - Each Route belongs to exactly one Agency, but each Agency can operate multiple Routes. `(1, N) Agency` → `(1, 1) Route (Operate)`.
  - The combination of Name and Type must be unique.
- Stop
  - Represents a physical location where trains pick up or drop off passengers.
  - Specialization (IS A Hierarchy): A Stop must be either a Station or a Platform, but cannot be both simultaneously. `(1, 1) Stop` → `{Station, Platform}`.
- Station/Platform
  - A Station contains multiple Platforms, and each Platform belongs to exactly one Station (assumption). `(1, N) Station` → `(1, 1) Platform (Contain)`.
- Served_Stop
  - Represents a stop assigned to a route.
  - Sequence_No must start at 1 and increment without gaps.
  - A route must have at least two stops to be meaningful. In other words, a Route must have at least two Served_Stops to be valid. `(1, N) Route` → `(1, 1) Served_Stop (Include)`.
  - Each Served_Stop must refer to a Stop, but a Stop may exist without being part of a Route (assumption). `(0, N) Stop` → `(1, 1) Served_Stop (Located_At)`.
- Trip
  - Represents a scheduled instance of a route for a service.

  - A Route can have multiple Trips, but each Trip belongs to one Route. A Service can include multiple Trips, but each Trip belongs to exactly one Service. Therefore, a ternary relationship is formed `(Assign_To)`.

- Stop_Info
  - Represents actual stop timing for a specific trip.
  - Each Trip has detailed stop information stored in Stop_Info. `(1, N) Trip` → `(1, 1) Stop_Info (Stop_At)`.
  - Sequence_No is derived from Served_Stop.Sequence_No and Trip.Direction.
- Service
  - Represents recurring schedules for trips.
- Exception
  - Represents deviations from regular service schedules.
  - Not all Services will have exceptions, but each exception must reference a valid Service (assumption). `(0, N) Service → (1, 1) Exception (Override)`.
- Weekday
  - Represents days in a week.
  - A Service may apply to multiple weekdays, and a weekday may have multiple assigned services. In the ERD, we assume that there may be a day in a week that does not witness a service, and there may be a service that does not run on any day. `(0, N) Service` → `(0, N) Weekday (Work_On)`.

2 Attribute domains

Agency

- dom(Agency.ID) = alphanumeric
- dom(Agency.Name) = non-empty string
- dom(Agency.Website) = non-empty string (URL)
- dom(Agency.Time_Zone) = non-empty string (standard timezone format)
- dom(Agency.Languages) = non-empty 2-letter string follow the ISO 639-1 codes (e.g., "fr", "en")
- dom(Agency.Phone_Number) = alphanumeric
- dom(Agency.Price_URL) = non-empty string (URL)
- dom(Agency.Email) = non-empty string (email address) ∪ {null}
- dom(Agency.Address) = non-empty string

Route

- dom(Route.ID) = $\mathbb{N}^+$

- dom(Route.Name) = non-empty string
- dom(Route.Type) = non-empty string (e.g., "IC", "S", etc.)

Stop

- dom(Stop.ID) = alphanumeric
- dom(Stop.Latitude) = $\{x \in \mathbb{R} \mid -90 \leq x \leq 90\}$
- dom(Stop.Longitude) = $\{x \in \mathbb{R} \mid -180 \leq x \leq 180\}$

Station

- dom(Station.Name) = non-empty string

Platform

- dom(Platform.Code) = alphanumeric (e.g., "1", "2b", "C")

Served_Stop

- dom(Served_Stop.Sequence_No) = $\mathbb{N}^+$

Trip

- dom(Trip.ID) = alphanumeric
- dom(Trip.Direction) = $\{0, 1\}$

Stop_Info

- dom(Stop_Info.Sequence_No) = $\mathbb{N}^+$
- dom(Stop_Info.Departure_Time) = date/time $\cup$ {null}
- dom(Stop_Info.Arrival_Time) = date/time $\cup$ {null}

Service

- dom(Service.ID) = alphanumeric
- dom(Service.Start_Date) = date (e.g. follows the form DD/MM/YYYY)
- dom(Service.End_Date) = date (e.g. follows the form DD/MM/YYYY)

Weekday

- dom(Weekday.Day) = non-empty string (e.g., "Monday", "Tuesday")

Exception

- dom(Exception.Date) = date
- dom(Exception.Code) = $\{1, 2\}$

Missing: The departure time of a stop is lower than the arrival time of a stop with a greater sequence number

3 Description of non-visible integrity constraints

- Agency.Languages has at least one value.
- Route links at least two Served_Stops.
- Combination of (Route.name, Route.type) is unique.
- ~~Stop cannot be both a Station and a Platform simultaneously.~~ IS A constraint
- Platform.Code is unique within each Station.
- Served_Stop.Sequence_No must be strictly increasing within each Route.
- Stop_Info.Departure_Time ≥ Stop_Info.Arrival_Time (except for the first stop in Trip)
- Stop_Info.Arrival_Time must be NULL for the first Served_Stop in a Trip.
- Stop_Info.Departure_Time must be NULL for the last Served_Stop in a Trip.
- Service.Start_Date ≤ Exception.Date ≤ Service.End_Date
- Service.Start_Date ≤ Service.End_Date
- If a Service has an Exception.Date with Exception.Code = 2 (removal), then there must exist another (exception) Service with the same Exception.Date where Exception.Code = 1 (addition).

↳ Impossible due to key

↳ Should be: An Exception with Code = 2 should happen on a day when the service is active.

4 Keys

4.1 Entity

- Agency: ID
- Route: ID
- Stop: ID
- Trip: ID
- Service: ID
- Station: ID (inherits from Stop)
- Platform: ID (inherits from Stop)
- Weekday: Day
- Route_Service: role "route" of "Operate_On" + role "service" of "Schedule"
- Served_Stop: role "route" of "Include" + role "stop" of "Located_At"
- Stop_Info: role "served_stop" of "Follow" + role "trip" of "Stop_At"
- Exception: Date + role "service" of "Override"

Don't exist in E-R diagram

4.2 Relationship

- Operate: role "route"
- Include: role "served_stop"
- Located_At: role "coordinates"
- Follow: role "stop_info"
- Stop_At: role "stop_info"
- Assign_To: role "trip"
- Operate_On: role "route_service"
- Schedule: role "route_service"
- Override: role "exception"
- Work_On: role "service" + role "weekday"
- Contain: role "platform"

5 Reduction to the relational model

- Agency(ID_agency, Name_agency, Website, Time_Zone, Phone_Number, Price_URL, Email, Address)
- Languages(ID_agency, Language)
- Route(ID_route, Name_route, Type_route, ID_agency)
- Stop(ID_stop, Latitude, Longitude)
- Station(ID_stop, Name_station)
- Platform(ID_stop, Code_platform, ID_stop_station)
- Served_Stop(ID_route, ID_stop, Sequence_No_served_stop)
- Trip(ID_trip, Direction, ID_route, ID_service)
- Stop_Info(ID_route, ID_stop, ID_trip, Arrival_Time, Departure_Time)
- Service(ID_service, Start_Date, End_Date)
- Exception(ID_service, Date, Code_exception)
- Weekday(Day)
- Work_On(ID_exception, ID_service)

6 Analysis of normal forms

6.1 Agency(ID_agency, Name_agency, Website, Time_Zone, Phone_Number, Price_URL, Email, Address)

Non-trivial functional dependency:

- {ID_agency} → {Name_agency, Website, Time_Zone, Phone_Number, Price_URL, Email, Address}

Candidate key:

- {ID_agency}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_agency} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_agency} is a candidate key, meaning all determinants are superkeys.

6.2 Languages(ID_agency, Language)

Non-trivial functional dependency:

- There are no non-trivial functional dependencies in this relation since all attributes are part of the primary key.

Candidate key:

- {ID_agency, Language}

Normal form analysis:

- **BCNF:** Satisfied - All attributes form part of the primary key, thus, there are no non-key attributes that could create problematic functional dependencies.

6.3 Route(ID_route, Name_route, Type_route, ID_agency)

Non-trivial functional dependencies:

- {ID_route} → {Name_route, Type_route, ID_agency}
- {Name_route, Type_route} → {ID_route, ID_agency}

Candidate keys:

- {ID_route}
- {Name_route, Type_route}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_route} and not on other non-key attributes.
- **BCNF:** Satisfied - {ID_route} is a candidate key and {Name_route, Type_route} is a superkey, meaning all determinants are superkeys.

6.4 Stop(ID_stop, Latitude, Longitude)

Non-trivial functional dependency:

- {ID_stop} → {Latitude, Longitude}

Candidate key:

- {ID_stop}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_stop} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_stop} is a candidate key, meaning all determinants are superkeys.

6.5 Station(ID_stop, Name_station)

Non-trivial functional dependency:

- {ID_stop} → {Name_station}

Candidate key:

- {ID_stop}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_stop} and not on other non-key attributes.

- **BCNF:** Satisfied - The only determinant {ID_stop} is a candidate key, meaning all determinants are superkeys.

6.6 Platform(ID_stop, Code_platform, ID_stop_station)

Non-trivial functional dependency:

- {ID_stop}→ {Code_platform, ID_stop_station}

Candidate key:

- {ID_stop}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_stop} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_stop} is a candidate key, meaning all determinants are superkeys.

6.7 Served_Stop (ID_route, ID_stop, Sequence_No_served_stop)

Non-trivial functional dependency:

- {ID_route, ID_stop} → {Sequence_No_served_stop}

Candidate key:

- {ID_route, ID_stop}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - Sequence_No_served_stop is fully dependent on the key {ID_route, ID_stop}.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_route, ID_stop} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_route, ID_stop} is a candidate key, meaning all determinants are superkeys.

6.8 Trip(ID_trip, Direction, ID_route, ID_service)

Non-trivial functional dependency:

- {ID_trip} → {Direction, ID_route, ID_service}

Candidate key:

- {ID_trip}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_trip} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_trip} is a candidate key, meaning all determinants are superkeys.

6.9 Stop_Info(ID_route, ID_stop, ID_trip, Arrival_Time, Departure_Time)

Non-trivial functional dependency:

- {ID_route, ID_stop, ID_trip} → {Arrival_Time, Departure_Time}

Candidate key:

- {ID_route, ID_stop, ID_trip}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - Arrival_Time and Departure_Time are fully dependent on the key {ID_route, ID_stop, ID_trip}.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_route, ID_stop, ID_trip} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_route, ID_stop, ID_trip} is a candidate key, meaning all determinants are superkeys.

6.10 Service(ID_service, Start_Date, End_Date)

Non-trivial functional dependency:

- {ID_service} → {Start_Date, End_Date}

Candidate key:

- {ID_service}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.

- **2NF:** Satisfied - The primary key is a single attribute, so there cannot be any partial dependencies.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_service} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_service} is a candidate key, meaning all determinants are superkeys.

6.11 Route_Service(ID_route, ID_service)

Non-trivial functional dependency:

- There are no non-trivial functional dependencies in this relation since all attributes are part of the primary key.

Candidate key:

- {ID_route, ID_service}

Normal form analysis:

- **BCNF:** Satisfied - All attributes form part of the primary key, thus, there are no non-key attributes that could create problematic functional dependencies.

6.12 Exception(ID_service, Date, Code_exception)

Non-trivial functional dependency:

- {ID_service, Date} → {Code_exception}

Candidate key:

- {ID_service, Date}

Normal form analysis:

- **1NF:** Satisfied - All attributes contain only atomic, non-repetitive values that are constant over time.
- **2NF:** Satisfied - Date is fully dependent on the key {ID_service, Date}.
- **3NF:** Satisfied - All non-key attributes depend only on the candidate key {ID_service, Date} and not on other non-key attributes.
- **BCNF:** Satisfied - The only determinant {ID_service, Date} is a candidate key, meaning all determinants are superkeys.

6.13 Weekday(Day)

Non-trivial functional dependency:

- There are no non-trivial functional dependencies in this relation since all attributes are part of the primary key.

Candidate key:

- {Day}

Normal form analysis:

- **BCNF:** Satisfied - All attributes form part of the primary key, thus, there are no non-key attributes that could create problematic functional dependencies.

6.14 Work_On(ID_exception, ID_service)

Non-trivial functional dependency:

- There are no non-trivial functional dependencies in this relation since all attributes are part of the primary key.

Candidate key:

- {ID_exception, ID_service}

Normal form analysis:

- **BCNF:** Satisfied - All attributes form part of the primary key, thus, there are no non-key attributes that could create problematic functional dependencies.