

DAMG 6210 - Database Design Document

Team Introduction – Group 4:

Name	Responsibility
Aravind Balaji	Schema Design, Tables, Constraints, Sequences, Indexes, Documentation
Akshay Dnyaneshwar Govind	Master Data, Seed Data, Test Cases, Permissions, Audit Log
Pranav Narendrabhai Patel	PL/SQL Packages, Stored Procedures, Functions, Triggers, Reports

Team combines strengths in data modeling, PL/SQL development, and testing to deliver a robust OLTP system.

2. Project Name

Project Title: Commuter Reservation System (CRS)

3. Problem Statement

The client's current train ticket booking process is manual and lacks proper data management capabilities, causing inconsistent data, overbooking issues, and operational inefficiencies. There is no real-time seat availability tracking, automated waitlist management, or validation of business rules. A centralized database system is required to:

- Manage trains, schedules, passengers, and reservations effectively.
- Enforce business rules at the database level (1-week advance booking, seat capacity limits, waitlist management).
- Ensure data accuracy through proper constraints and automated validation.
- Support efficient querying for seat availability and booking status in real-time.
- Provide transaction integrity for booking and cancellation operations.
- Enable waitlist promotion upon cancellations with clear audit trails.

4. Solution Approach (How the Problem Will Be Tackled & Resolved)

Approach:

- Requirement Gathering: Identify core entities (Train, Schedule, Passenger, Reservation) and business rules (40+5 seat capacity per class, 1-week advance booking, waitlist promotion logic).
- Conceptual Modeling: Create ERD depicting relationships between trains, schedules, passengers, and reservations.
- Logical Modeling: Translate into relational schema with primary/foreign keys, unique constraints, and check constraints.
- Normalization: Apply 1NF–3NF to eliminate redundancy and ensure data integrity.
- Implementation: Build schema with two user roles (Admin, Data User) and PL/SQL packages for booking/cancellation logic; add triggers for waitlist promotion.
- Testing & Reporting: Seed sample data, validate business rules, test exception scenarios, and create views for booking analytics.

Expected Outcome:

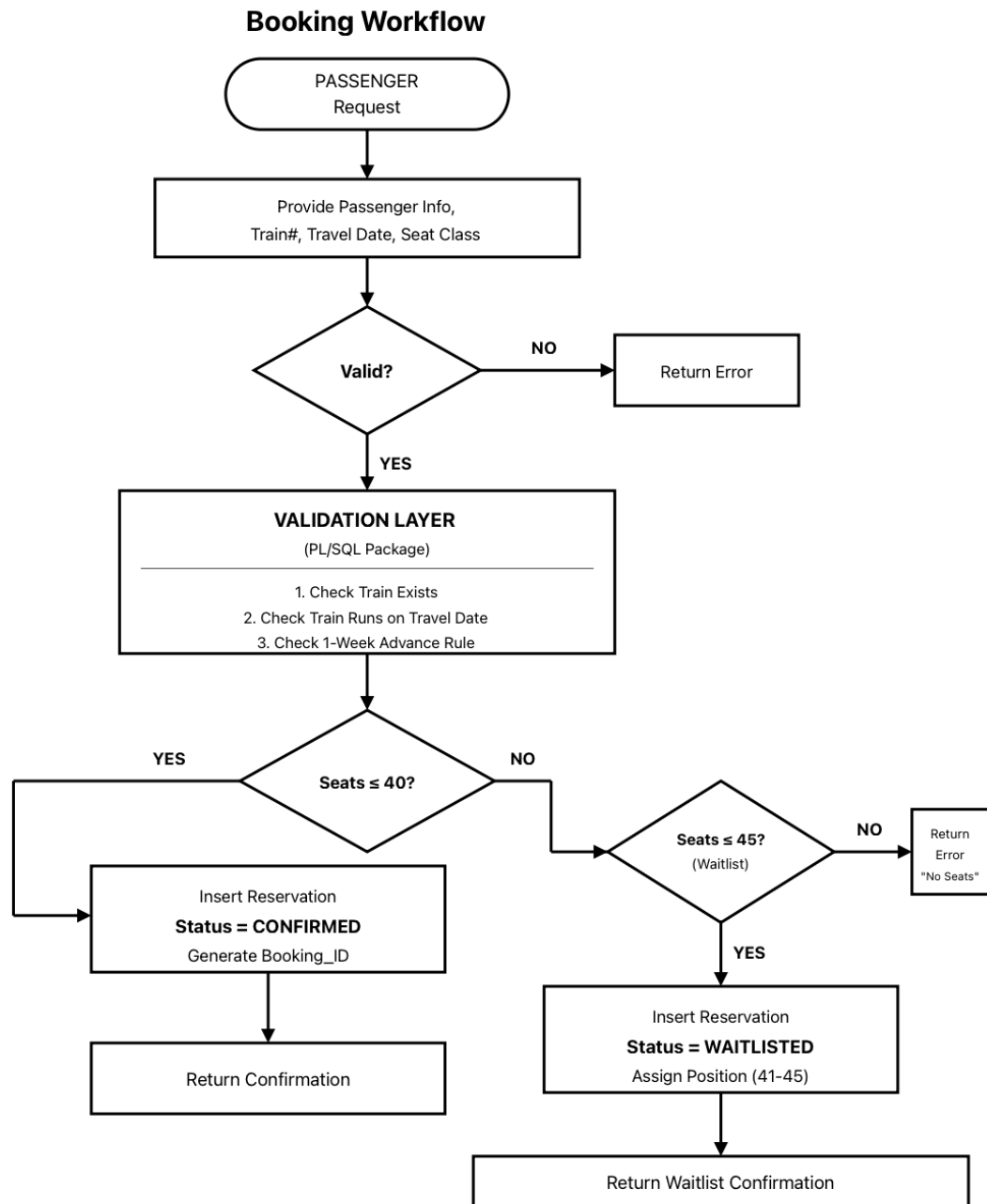
- Real-time seat availability tracking with automated capacity management.
- Zero redundancy with strong referential integrity.
- Automated waitlist promotion upon cancellations.
- Operational reports for booking trends, occupancy rates, and

revenue analysis. OLTP Justification:

This system follows an Online Transaction Processing (OLTP) architecture. Each user action—booking a ticket, checking seat availability, canceling a reservation, or promoting waitlist passengers—is a short, atomic transaction on a normalized schema with ACID properties. Concurrency is supported via row-level locking and indexes on primary/foreign keys.

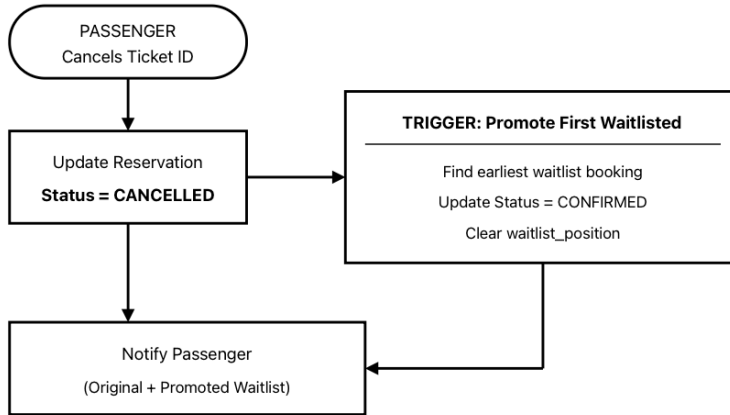
5. Workflow Diagram:

Booking workflow:



Cancellation workflow

Cancellation Workflow



6. Database Conceptual Model (ER Diagram Description)

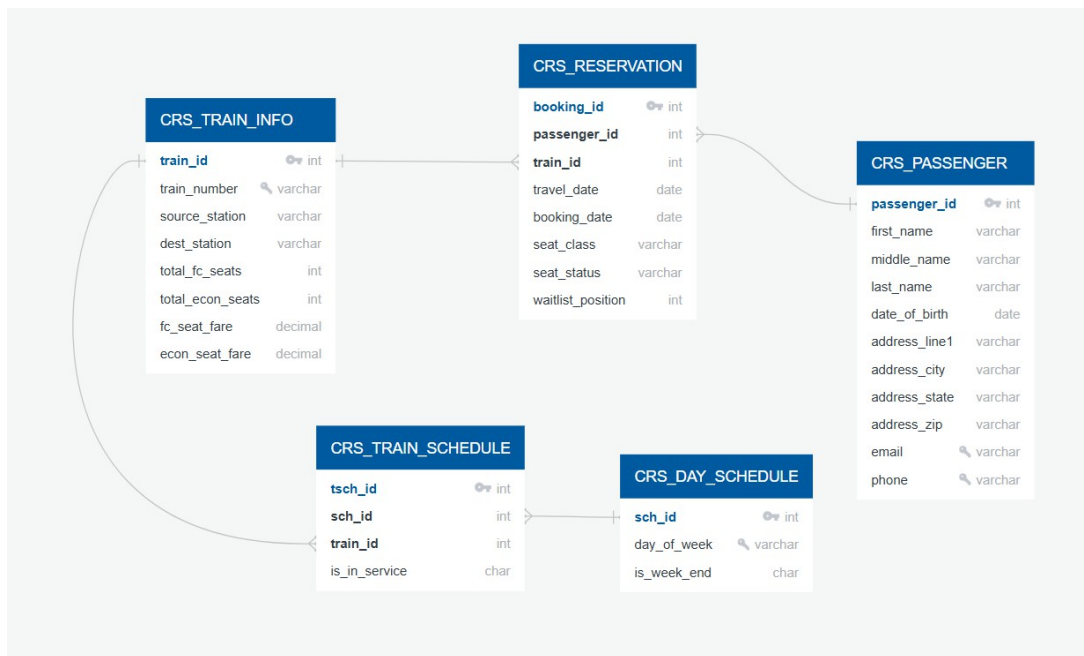
Entities & Attributes:

Entity	Attributes
CRS_TRAIN_INFO	Train_ID (PK), Train_Number (UNIQUE), Source_Station, Dest_Station, Total_FC_Seats, Total_Econ_Seats, FC_Seat_Fare, Econ_Seat_Fare
CRS_DAY_SCHEDULE	Sch_ID (PK), Day_of_Week (UNIQUE), Is_Week_End → removes day/weekend redundancy
CRS_TRAIN_SCHEDULE	TSch_ID (PK), Sch_ID (FK → CRS_DAY_SCHEDULE), Train_ID (FK → CRS_TRAIN_INFO), Is_In_Service → Bridge table for M:N relationship
CRS_PASSENGER	Passenger_ID (PK), First_Name, Middle_Name, Last_Name, Date_of_Birth, Address_Line1, Address_City, Address_State, Address_Zip, Email (UNIQUE), Phone (UNIQUE)
CRS_RESERVATION	Booking_ID (PK), Passenger_ID (FK → CRS_PASSENGER), Train_ID (FK → CRS_TRAIN_INFO), Travel_Date, Booking_Date, Seat_Class, Seat_Status, Waitlist_Position

Key Design Highlights:

- Introduced CRS_DAY_SCHEDULE master table to eliminate day/weekend redundancy → All tables are now in BCNF.
- CRS_TRAIN_SCHEDULE bridge table resolves Many-to-Many relationship between trains and days.
- Single-column primary keys eliminate partial dependencies in all tables.
- Email and Phone are UNIQUE to prevent duplicate passenger records.
- Seat_Status CHECK constraint ensures only valid values: CONFIRMED, WAITLISTED, CANCELLED.
- Seat_Class CHECK constraint allows only: FC (First Class), ECON (Economy).

Entity Relationship (ER) Diagram:



Relationships:

- CRS_TRAIN_INFO ↔ CRS_DAY_SCHEDULE (M:N via CRS_TRAIN_SCHEDULE)
- CRS_TRAIN_INFO → CRS_RESERVATION (1:M)
- CRS_PASSENGER → CRS_RESERVATION (1:M)

7. Relational Model (Logical Design)

Table Name	Primary Key	Foreign Keys	Notes / Important Constraints
CRS_TRAIN_INFO	Train_ID	—	Train_Number UNIQUE
CRS_DAY_SCHEDULE	Sch_ID	—	Day_of_Week UNIQUE, Is_Week_End CHECK ('Y', 'N')
CRS_TRAIN_SCHEDULE	TSch_ID	Sch_ID → CRS_DAY_SCHEDULE, Train_ID → CRS_TRAIN_INFO	Is_In_Service CHECK ('Y', 'N')
CRS_PASSENGER	Passenger_ID	—	Email UNIQUE, Phone UNIQUE
CRS_RESERVATION	Booking_ID	Passenger_ID → CRS_PASSENGER, Train_ID → CRS_TRAIN_INFO	Seat_Class CHECK ('FC', 'ECON'), Seat_Status CHECK ('CONFIRMED', 'WAITLISTED', 'CANCELLED')

Detailed Table Specifications:

CRS_TRAIN_INFO

Primary Key: Train_ID

Unique Constraints:

Train_Number Attributes:

Train_ID: System-generated unique identifier

Train_Number: User-facing train identifier (e.g.,

"TR-101") Source_Station: Departure station

name

Dest_Station: Arrival station name

Total_FC_Seats: Total first class

capacity (40) Total_Econ_Seats: Total

economy capacity (40) FC_Seat_Fare:

First class ticket price Econ_Seat_Fare:

Economy ticket price

CRS_DAY_SCHEDULE

Primary Key: Sch_ID

Unique Constraints: Day_of_Week

Check Constraints: Is_Week_End IN

('Y', 'N') Attributes:

Sch_ID: System-generated unique identifier (1-7)

Day_of_Week: Day name (Monday through

Sunday)

Is_Week_End: Weekend flag: 'Y' for Sat/Sun, 'N' for Mon-Fri

Purpose: Provides day reference to eliminate redundancy; contains 7 static rows

CRS_TRAIN_SCHEDULE

Primary Key:

TSch_ID Foreign

Keys:

Sch_ID → CRS_DAY_SCHEDULE(Sch_ID)

Train_ID →

CRS_TRAIN_INFO(Train_ID) Check

Constraints: Is_In_Service IN ('Y', 'N')

Purpose: Defines which days each train operates (resolves M:N relationship between trains and schedules)

CRS_PASSENGER

Primary Key: Passenger_ID

Unique Constraints: Email,

Phone Attributes:

Passenger_ID: System-generated unique identifier

First_Name, Middle_Name, Last_Name: Name split for atomic values

(1NF) Date_of_Birth: Used for age-based categorization

(minor/major/senior)

Address_Line1, Address_City, Address_State, Address_Zip: Address

components Email: Email address (unique across all passengers)

Phone: Phone number (unique across all passengers)

CRS_RESERVATION

Primary Key: Booking_ID

Foreign Keys:

Passenger_ID →

CRS_PASSENGER(Passenger_ID) Train_ID →

CRS_TRAIN_INFO(Train_ID)

Check Constraints:

Seat_Class IN ('FC', 'ECON')

Seat_Status IN ('CONFIRMED', 'WAITLISTED', 'CANCELLED')

Business Logic Attributes:

Booking_ID: System-generated unique ticket identifier

Travel_Date: Date of journey (must be \leq booking_date +

7 days) Booking_Date: Transaction timestamp

Seat_Class: 'FC' (First Class) or 'ECON' (Economy)

Seat_Status: Booking state

(CONFIRMED/WAITLISTED/CANCELLED)

Waitlist_Position: NULL if confirmed, 1-5 if waitlisted

Business Rules:

40 confirmed seats per class → positions 1-40

5 waitlist seats per class → positions 41-45

On cancellation: First waitlisted ticket promoted to CONFIRMED

8. Normalization and Redundancy Analysis

First Normal Form (1NF):

Rule: All attributes are atomic; no multi-valued columns or repeating groups. Application:

Passenger name split into First_Name, Middle_Name, Last_Name (atomic values). Address components separated into Address_Line1, City, State, Zip.

Each booking is a separate row in

CRS_RESERVATION. No arrays or comma-separated values in any column.

Before (Violation):

PASSENGER_BOOKING

Name	Trains_Booked	Phone_Numbers
John Doe	TR-101, TR-102, TR-105	555-1234, 555-56

After (1NF Compliant):

CRS_PASSENGER

Passenger_ID	First_Name	Last_Name	Phone
1	John	Doe	555-1234

CRS_RESERVATION (separate rows)

Booking_ID	Passenger_ID	Train_ID
B001	1	101
B002	1	102
B003	1	105

Second Normal Form (2NF):

Rule: Every non-key attribute is fully dependent on the entire composite key. No partial dependencies. Application:

All tables use single-column primary keys (Train_ID, Passenger_ID, Booking_ID, TSch_ID).

CRS_RESERVATION stores only Passenger_ID (FK) and Train_ID (FK), not passenger names or train details.

Passenger details stored once in CRS_PASSENGER; train details stored once in CRS_TRAIN_INFO. Before (Violation - Partial Dependency):

BOOKING_DETAILS (Composite PK: Passenger_ID + Train_ID)

Passenger_ID	Train_ID	Passenger_Name	Train_Route	Seat_Class
1	101	John Doe	NYC-BOS	FC

Problem: Passenger_Name depends only on Passenger_ID (partial dependency) Problem: Train_Route depends only on Train_ID (partial dependency)

After (2NF Compliant):

CRS_PASSENGER (separate table)

Passenger_ID	First_Name	Last_Name
1	John	Doe

CRS_TRAIN_INFO (separate table)

Train_ID	Source_Station	Dest_Station
101	NEW York	Boston

CRS_RESERVATION (no partial dependencies)

Booking_ID	Passenger_ID	Train_ID	Seat_Class
B001	1	101	FC

Third Normal Form (3NF):

Rule: All transitive dependencies removed. Every non-key attribute depends directly on the primary key. Application:

Day/weekend information moved to CRS_DAY_SCHEDULE reference table (eliminates day_name → is_weekend transitive dependency).

Train schedule details live in CRS_TRAIN_SCHEDULE

bridge table. Model details referenced via FK in

CRS_RESERVATION.

Before (Violation - Transitive Dependency):

RESERVATION_WITH_DAY (Not in 3NF)

Booking_ID	Train_ID	Travel Date	Day Name	Is Weekend
B001	101	2025-12-05	Friday	N
B002	102	2025-12-06	Saturday	Y
B003	103	2025-12-06	Saturday	Y

Problem: Is_Weekend depends on Day_Name, not on Booking_ID (transitive dependency) After (3NF Compliant):

CRS_DAY_SCHEDULE (reference table)

Sch_ID	Day_of_Week	Is_Week_End
1	Monday	N
2	Tuesday	N
3	Wednesday	N
4	Thursday	N
5	Friday	N
6	Saturday	Y
7	Sunday	Y

CRS_RESERVATION (no transitive dependencies)

Booking_ID	Train_ID	Travel_Date
B001	101	2025-12-05
B002	102	2025-12-06

9. Security Design

To ensure data confidentiality, integrity, and controlled access across departments, role-based privileges are implemented at both schema and transaction levels. The Commuter Reservation System (CRS) uses the principle of least privilege — each user role has only the access required to perform its function.

a) User Roles

Role	Description	Example Privileges
DB_ADMIN	Manage database objects, users, and privileges.	CREATE, ALTER, DROP, GRANT, REVOKE
CRS_ADMIN_USER	Schema owner; creates tables, procedures, and grants permissions.	CREATE TABLE, CREATE PROCEDURE, GRANT EXECUTE
CRS_DATA_USER	Execute stored procedures for booking/cancellation operations.	EXECUTE on CRS_BOOKING_PKG, CRS_PASSENGER_PKG
CRS_REPORT_USER	Generates read-only reports and analytics.	SELECT only on BOOKING_REPORT_VIEW, REVENUE_VIEW, OCCUPANCY_VIEW
CRS_SERVICE_USER	Backend application user; calls procedures via API.	EXECUTE on booking/cancellation procedures

b) Data Access Controls

- Role-based access control (RBAC): Each department (Admin, Operations, Reporting) is assigned a unique role to isolate data access.
- Authentication & Authorization: Database users authenticate via role assignment; no direct INSERT/UPDATE/DELETE access to base tables for application users.

- Procedure-level security: CRS_DATA_USER can only execute stored procedures, not directly modify tables. All business logic and validation encapsulated in packages.
- Data encryption: Sensitive information such as passenger email and phone can be protected using Oracle Transparent Data Encryption (TDE).
- Audit triggers: Every booking, cancellation, and waitlist promotion is logged automatically to ensure traceability.

c) Audit and Compliance

- To maintain accountability and data lineage, all transactional operations are recorded in an AUDIT_LOG table.
- Each log entry captures:
(USER_ID, TABLE_NAME, OPERATION, TIMESTAMP, RECORD_ID)

This supports:

- Data recovery and rollback of unintended changes.

- Compliance with internal security standards and traceability policies.
- Monitoring of unusual or unauthorized activity (e.g., bulk cancellations).

The combination of role-based privileges, procedure-level security, audit trails, and encryption ensures that the CRS database remains secure, compliant, and reliable. These measures also align with OLTP best practices, where real-time transactions require strict access and integrity control.

10. Conclusion

The Commuter Reservation System (CRS) is a fully normalized OLTP database supporting end-to-end train ticket booking operations from reservation to cancellation to waitlist management. It ensures:

- Accurate capacity management with 40 confirmed + 5 waitlist seats per class per train.
- Real-time seat availability tracking through automated counting queries.
- Automated waitlist promotion upon cancellations via database triggers.
- Comprehensive booking history with audit trails for every transaction.

The design enforces referential integrity, reduces redundancy to zero, and provides a scalable foundation for operational reporting. All business rules (1-week advance booking, unique email/phone, seat capacity limits) are enforced at the database level through constraints, procedures, and triggers, ensuring data consistency and reliability for a production-grade train reservation platform.