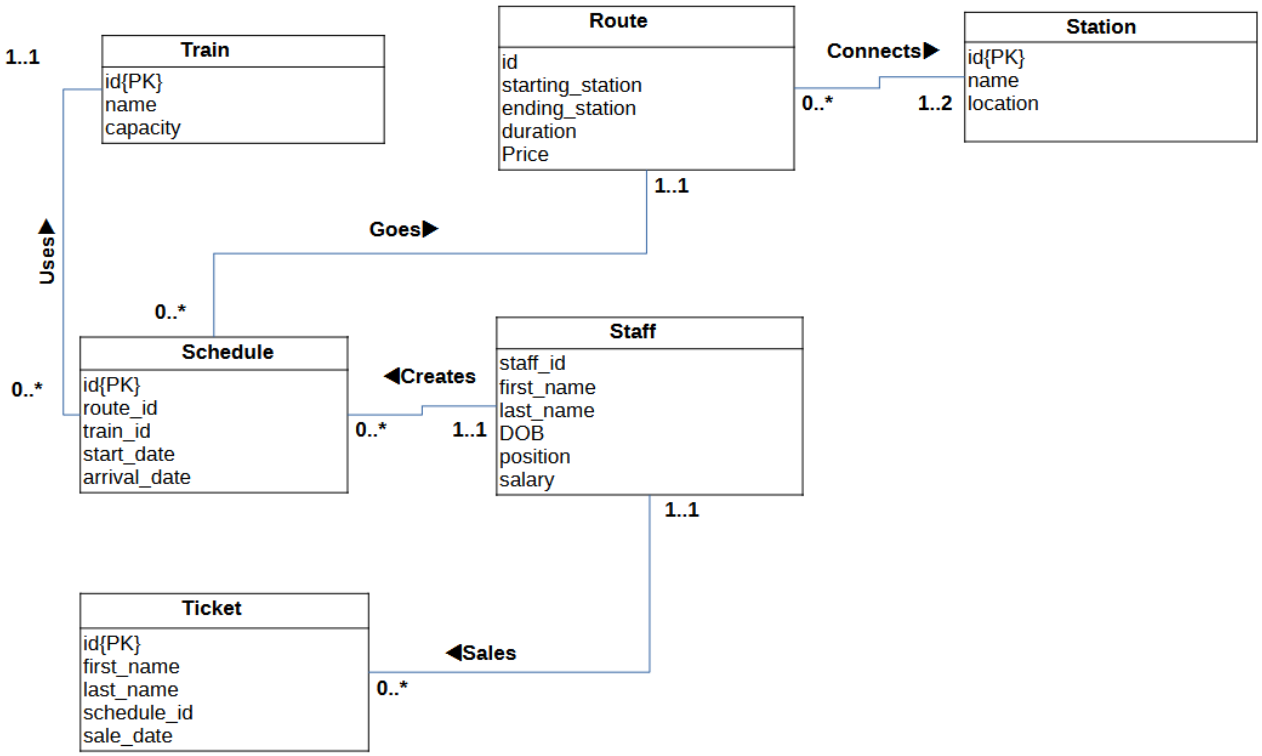
Table of Contents

Diagramatical View



Tables:

- **station**(id, name, location) : id primary key
- **train**(id, name, capacity): id primary key
- **route**(id starting_station, end_station, duration, price) : id primary key , starting_station and end_station reference station.id
- **schedule**(id, route_id, price, start_date, arrival_date, train_id) : id primary key , route_id reference route.id route.id)
- **ticket**(id, first_name, last_name, schedule_id) id primary key and schedule_id reference to schedule.id
- **staff**(id , first_name, last_name, sex , dob, position , salary, password)

Commands

```
-- Create station table
CREATE TABLE station (
    id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    location VARCHAR2(100)
);

-- Create train table
CREATE TABLE train (
    id NUMBER PRIMARY KEY,
    name VARCHAR2(100),
    capacity NUMBER
);

-- Create route table
CREATE TABLE route (
    id NUMBER PRIMARY KEY,
    starting_station NUMBER,
    end_station NUMBER,
    duration NUMBER,
    price NUMBER,
    FOREIGN KEY (starting_station) REFERENCES station(id),
    FOREIGN KEY (end_station) REFERENCES station(id)
);

-- Create schedule table
CREATE TABLE schedule (
    id NUMBER PRIMARY KEY,
    route_id NUMBER,
    start_date DATE,
    arrival_date DATE,
    train_id NUMBER,
    FOREIGN KEY (route_id) REFERENCES route(id),
    FOREIGN KEY (train_id) REFERENCES train(id)
);

-- Create ticket table
CREATE TABLE ticket (
    id NUMBER PRIMARY KEY,
    first_name VARCHAR2(100),
    last_name VARCHAR2(100),
    schedule_id NUMBER,
```

```
    FOREIGN KEY (schedule_id) REFERENCES schedule(id)
);

CREATE TABLE staff (
    id INT PRIMARY KEY,
    first_name VARCHAR2(50),
    last_name VARCHAR2(50),
    sex CHAR(1),
    dob DATE,
    position VARCHAR2(100),
    salary NUMBER(10, 2),
    password VARCHAR2(100)
);
```

Sequence:

```
CREATE SEQUENCE table_id_seq
    START WITH 1
    INCREMENT BY 1
    NOCACHE
    NOCYCLE;
```

Create a Sequence for Auto-Incrementing ID

```
CREATE SEQUENCE staff_id_seq START WITH 1 INCREMENT BY 1;
```

Trigger:

This trigger will calculate the arrival date.

```
CREATE OR REPLACE TRIGGER calculate_arrival_date
BEFORE INSERT ON schedule
FOR EACH ROW
DECLARE
    v_duration NUMBER;
BEGIN
    -- Retrieve the duration based on the inserted route_id
    SELECT duration INTO v_duration
    FROM route
    WHERE id = :NEW.route_id;

    -- Calculate the arrival date by adding the duration to the start date
    :NEW.arrival_date := :NEW.start_date + INTERVAL '1' HOUR * v_duration;
END;
```

/

Trigger to create ticket saledate

```
CREATE OR REPLACE TRIGGER ticket_sale_date_trigger
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
   :NEW.sale_date := SYSDATE;
END;
/
```

Auto_increament triggers

```
-- Trigger for station table
CREATE OR REPLACE TRIGGER station_trigger
BEFORE INSERT ON station
FOR EACH ROW
BEGIN
   SELECT table_id_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/

-- Trigger for train table
CREATE OR REPLACE TRIGGER train_trigger
BEFORE INSERT ON train
FOR EACH ROW
BEGIN
   SELECT table_id_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/

-- Trigger for route table
CREATE OR REPLACE TRIGGER route_trigger
BEFORE INSERT ON route
FOR EACH ROW
BEGIN
   SELECT table_id_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/

-- Trigger for schedule table
CREATE OR REPLACE TRIGGER schedule_trigger
```

```
BEFORE INSERT ON schedule
FOR EACH ROW
BEGIN
   SELECT table_id_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/

-- Trigger for ticket table
CREATE OR REPLACE TRIGGER ticket_trigger
BEFORE INSERT ON ticket
FOR EACH ROW
BEGIN
   SELECT table_id_seq.NEXTVAL INTO :NEW.id FROM dual;
END;
/

CREATE OR REPLACE TRIGGER staff_id_trigger
BEFORE INSERT ON staff
FOR EACH ROW
BEGIN
   SELECT staff_id_seq.NEXTVAL INTO :NEW.id FROM DUAL;
END;
/
```

View :

schedule_info_view created here provides a convenient way to access schedule information along with related data in a simplified manner.

```
CREATE or replace VIEW schedule_info_view AS
SELECT
   s.id AS schedule_id,
   start_station.name AS start_station,
   end_station.name AS end_station,
   TO_CHAR(s.start_date, 'YYYY-MM-DD') AS start_date,
   TO_CHAR(s.start_date, 'HH24:MI:SS') AS start_time,
   TO_CHAR(s.arrival_date, 'YYYY-MM-DD') AS arrival_date,
   TO_CHAR(s.arrival_date, 'HH24:MI:SS') AS arrival_time,
   t.name AS train_name,
   t.capacity AS train_capacity,
   r.price
FROM
   schedule s
JOIN
```

```
    route r ON s.route_id = r.id
JOIN
    station start_station ON r.starting_station = start_station.id
JOIN
    station end_station ON r.end_station = end_station.id
JOIN
    train t ON s.train_id = t.id;
```

View to show ticket information

```
CREATE VIEW ticket_info_view AS
SELECT
    t.id AS ticket_id,
    t.first_name || ' ' || t.last_name AS passenger_name,
    start_station.name AS start_station,
    end_station.name AS end_station,
    TO_CHAR(s.start_date, 'YYYY-MM-DD') AS start_date,
    TO_CHAR(s.start_date, 'HH24:MI:SS') AS start_time,
    TO_CHAR(s.arrival_date, 'YYYY-MM-DD') AS arrival_date,
    TO_CHAR(s.arrival_date, 'HH24:MI:SS') AS arrival_time,
    r.price,
    tr.name AS train_name
FROM
    ticket t
JOIN
    schedule s ON t.schedule_id = s.id
JOIN
    route r ON s.route_id = r.id
JOIN
    station start_station ON r.starting_station = start_station.id
JOIN
    station end_station ON r.end_station = end_station.id
JOIN
    train tr ON s.train_id = tr.id;
```

View to show yearly revenue

```
CREATE OR REPLACE VIEW daily_revenue_report_view AS
SELECT
    TO_CHAR(t.sale_date, 'YYYY-MM-DD') AS day,
    COUNT(t.id) AS tickets_sold,
    SUM(r.price) AS revenue,
    calculate_tax(0.1, SUM(r.price)) AS tax,
    SUM(r.price) - calculate_tax(10.0, SUM(r.price)) AS revenue_after_tax
```

```
FROM
   ticket t
JOIN
   schedule s ON t.schedule_id = s.id
JOIN
   route r ON s.route_id = r.id
GROUP BY
   TO_CHAR(t.sale_date, 'YYYY-MM-DD');
```

View to show weekly revenue

```
CREATE OR REPLACE VIEW weekly_revenue_report_view AS
SELECT
   TO_CHAR(t.sale_date, 'IYYY-IW') AS year_week,
   COUNT(t.id) AS tickets_sold,
   SUM(r.price) AS revenue,
   calculate_tax(0.1, SUM(r.price)) AS tax,
   SUM(r.price) - calculate_tax(10.0, SUM(r.price)) AS revenue_after_tax
FROM
   ticket t
JOIN
   schedule s ON t.schedule_id = s.id
JOIN
   route r ON s.route_id = r.id
GROUP BY
   TO_CHAR(t.sale_date, 'IYYY-IW');
```

View to show monthly revenue

```
CREATE OR REPLACE VIEW monthly_revenue_report_view AS
SELECT
   TO_CHAR(t.sale_date, 'YYYY-MM') AS month,
   COUNT(t.id) AS tickets_sold,
   SUM(r.price) AS revenue,
   calculate_tax(0.1, SUM(r.price)) AS tax,
   SUM(r.price) - calculate_tax(10.0, SUM(r.price)) AS revenue_after_tax
FROM
   ticket t
JOIN
   schedule s ON t.schedule_id = s.id
JOIN
   route r ON s.route_id = r.id
GROUP BY
```

```
    TO_CHAR(t.sale_date, 'YYYY-MM');
```

View to show revenue by route

```
CREATE OR REPLACE VIEW route_performance_report_view AS
SELECT
    r.id AS route_id,
    start_station.name || ' to ' || end_station.name AS route_name,
    COUNT(t.id) AS tickets_sold,
    SUM(r.price) AS revenue,
    calculate_tax(0.1, SUM(r.price)) AS tax, -- Assuming a tax rate of 10%
    SUM(r.price) - calculate_tax(0.1, SUM(r.price)) AS revenue_after_tax
FROM
    ticket t
JOIN
    schedule s ON t.schedule_id = s.id
JOIN
    route r ON s.route_id = r.id
JOIN
    station start_station ON r.starting_station = start_station.id
JOIN
    station end_station ON r.end_station = end_station.id
GROUP BY
    r.id, start_station.name, end_station.name
ORDER BY
    revenue DESC;
```

View to show yearly revenue

```
CREATE OR REPLACE VIEW yearly_revenue_report_view AS
SELECT
    EXTRACT(YEAR FROM t.sale_date) AS year,
    COUNT(t.id) AS tickets_sold,
    SUM(r.price) AS revenue,
    calculate_tax(10.0, SUM(r.price)) AS tax, -- Assuming a tax rate of 10%
    SUM(r.price) - calculate_tax(10.0, SUM(r.price)) AS revenue_after_tax
FROM
    ticket t
JOIN
    schedule s ON t.schedule_id = s.id
JOIN
    route r ON s.route_id = r.id
GROUP BY
    EXTRACT(YEAR FROM t.sale_date);
```

Stored procedure :

Procedure to add schedule

```
CREATE OR REPLACE PROCEDURE add_schedule (
    p_route_id IN NUMBER,
    p_start_date IN TIMESTAMP,
    p_train_id IN NUMBER
)
IS
BEGIN
    INSERT INTO schedule (id, route_id, start_date, train_id)
    VALUES (table_id_seq.NEXTVAL, p_route_id, p_start_date, p_train_id);
END;
/
```

Procedure to add ticket

```
CREATE OR REPLACE PROCEDURE add_ticket (
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_schedule_id IN NUMBER
)
IS
BEGIN
    INSERT INTO ticket (id, first_name, last_name, schedule_id)
    VALUES (table_id_seq.NEXTVAL, p_first_name, p_last_name, p_schedule_id);
END;
/
```

Procedure to remove tickets

```
CREATE OR REPLACE PROCEDURE remove_ticket (
    p_ticket_id IN NUMBER
)
IS
BEGIN
    DELETE FROM ticket
    WHERE id = p_ticket_id;
```

```
END;
/
```

Procedure to remove schedule and it’s tickets

```
CREATE OR REPLACE PROCEDURE remove_schedule_and_tickets (
    p_schedule_id IN NUMBER
)
IS
BEGIN
    -- Remove tickets associated with the given schedule ID
    DELETE FROM ticket
    WHERE schedule_id = p_schedule_id;

    -- Remove the schedule itself
    DELETE FROM schedule
    WHERE id = p_schedule_id;

    -- Commit the transaction
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Schedule and associated tickets removed successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error: Unable to remove schedule and associated tickets.');
END;
/
```

Procedure to update staff info

```
CREATE OR REPLACE PROCEDURE update_staff_info (
    p_id IN INT,
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2,
    p_sex IN CHAR,
    p_dob IN DATE,
    p_position IN VARCHAR2,
    p_salary IN NUMBER,
    p_password IN VARCHAR2
)
IS
BEGIN
    UPDATE staff
```

```
    SET
        first_name = p_first_name,
        last_name = p_last_name,
        sex = p_sex,
        dob = p_dob,
        position = p_position,
        salary = p_salary,
        password = p_password
    WHERE id = p_id;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Staff information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error: Unable to update staff information.');
END;
/
```

Procedure to update ticket holder information

```
CREATE OR REPLACE PROCEDURE update_ticket_holder_info (
    p_ticket_id IN INT,
    p_first_name IN VARCHAR2,
    p_last_name IN VARCHAR2
)
IS
BEGIN
    UPDATE ticket
    SET
        first_name = p_first_name,
        last_name = p_last_name
    WHERE id = p_ticket_id;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Ticket holder information updated successfully.');
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Error: Unable to update ticket holder information.');
END;
/
```

Function :

Functrion to Calculate Tax

```
CREATE OR REPLACE FUNCTION calculate_tax (
    p_tax_rate IN NUMBER,
    p_taxable_amount IN NUMBER
)
RETURN NUMBER
IS
    v_tax NUMBER;
BEGIN
    v_tax := p_taxable_amount * (p_tax_rate / 100);
    RETURN v_tax;
END;
/
```