NYU

HOTEL LAST RESORT

Group L

PRESENTED BY Laura Liu, Lucy Sha, Christina Chen, and Jack Zhong
12/11/2025

ERD OVERVIEW

01

NYU

ERD Overview

Represents full end-to-end hotel operations: **physical rooms → reservations → billing → access control**

Organized into four major groups:

- Physical Hotel Construction
- Reservation & Customer Management
- Billing & Transactions
- Access Control & Staff Management

Key ERD Assumptions

Hotel Structure Assumptions

- Rooms belong to a *hierarchical* layout **(building → wing → floor)**
- Room type uses **standardized classifications** defining the purpose (e.g., sleeping, meeting, working) and configuration of each room.
- Meeting_Space: Represents larger or **composite spaces** composed of individual meeting-type rooms. **"Type"** attribute in meeting space refers to available facilities, such as a patio, pool, or courtyard.
- Suites contain multiple rooms (often 1 bedroom + 1 meeting room)

NYU

Key ERD Assumptions Cont.

Customer & Reservation Assumptions

- Anyone interacting with the hotel is a **Customer** (occupant, payer, contact)
- Reservations may include **multiple rooms** and can be **extended**
- **Reservation_Party_Info** differentiates booker, payer, and occupant
- Customer requests can exist even **before** a room is assigned

Key ERD Assumptions Cont.

Billing & Access Assumptions

- A Billing record **aggregates multiple Transactions**, but is paid as one bill
- Every monetary action (room rate, service, fee) becomes a Transaction
- Each bill must correspond to the **payer** of the reservation, not necessarily the booker or occupant
- Customer and staff keycard swipes are fully logged for auditing

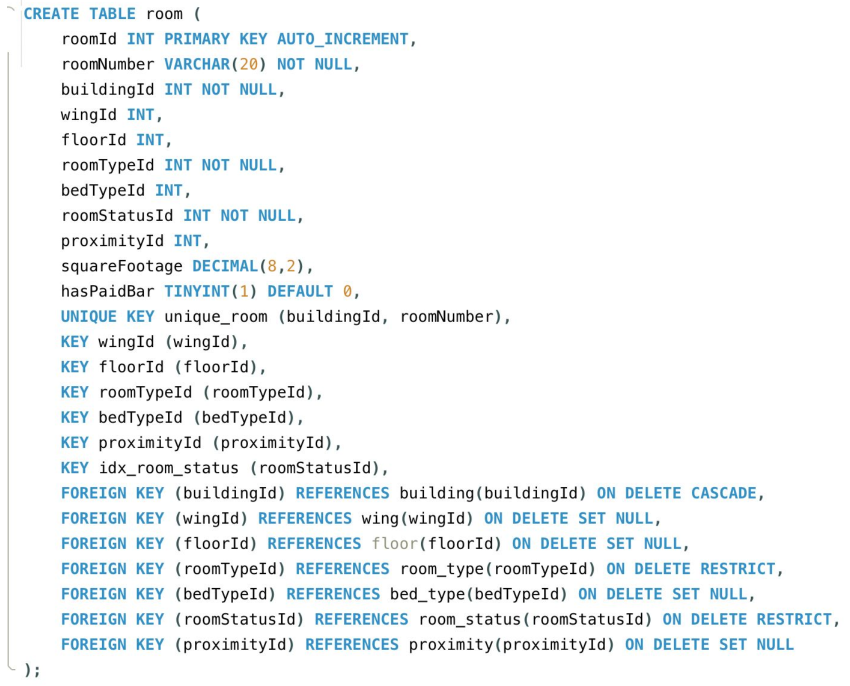
NYU

DATABASE & QUERIES

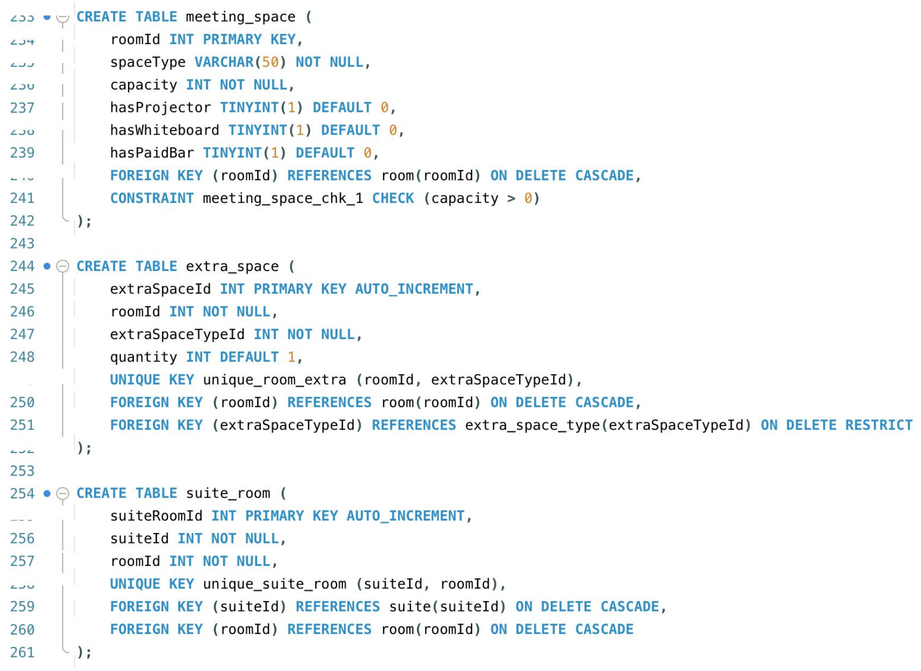
02

NYU

Database Setup



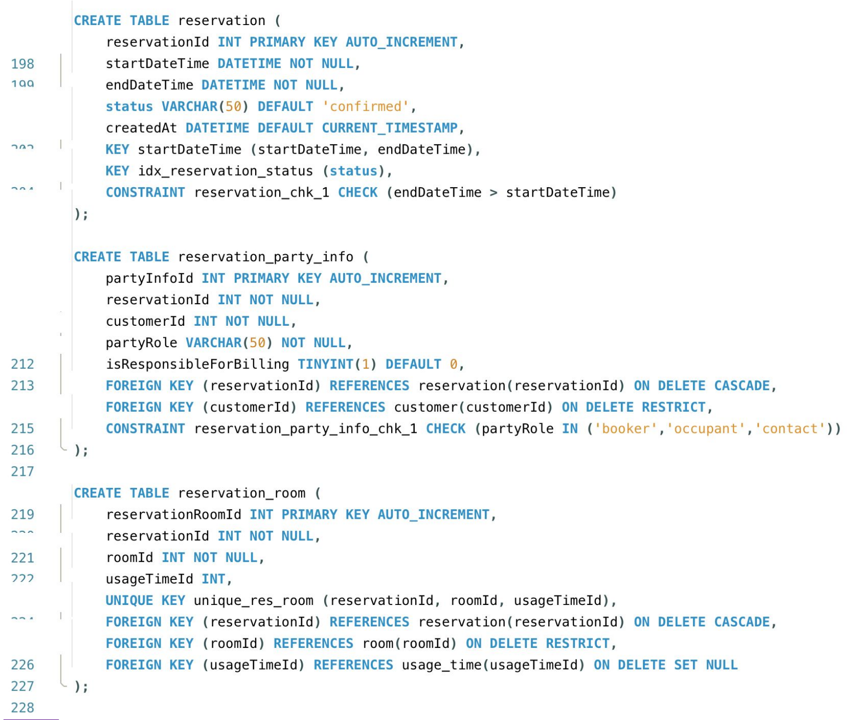
```sql
CREATE TABLE room (
    roomId INT PRIMARY KEY AUTO_INCREMENT,
    roomNumber VARCHAR(20) NOT NULL,
    buildingId INT NOT NULL,
    wingId INT,
    floorId INT,
    roomTypeId INT NOT NULL,
    bedTypeId INT,
    roomStatusId INT NOT NULL,
    proximityId INT,
    squareFootage DECIMAL(8,2),
    hasPaidBar TINYINT(1) DEFAULT 0,
    UNIQUE KEY unique_room (buildingId, roomNumber),
    KEY wingId (wingId),
    KEY floorId (floorId),
    KEY roomTypeId (roomTypeId),
    KEY bedTypeId (bedTypeId),
    KEY proximityId (proximityId),
    KEY idx_room_status (roomStatusId),
    FOREIGN KEY (buildingId) REFERENCES building(buildingId) ON DELETE CASCADE,
    FOREIGN KEY (wingId) REFERENCES wing(wingId) ON DELETE SET NULL,
    FOREIGN KEY (floorId) REFERENCES floor(floorId) ON DELETE SET NULL,
    FOREIGN KEY (roomTypeId) REFERENCES room_type(roomTypeId) ON DELETE RESTRICT,
    FOREIGN KEY (bedTypeId) REFERENCES bed_type(bedTypeId) ON DELETE SET NULL,
    FOREIGN KEY (roomStatusId) REFERENCES room_status(roomStatusId) ON DELETE RESTRICT,
    FOREIGN KEY (proximityId) REFERENCES proximity(proximityId) ON DELETE SET NULL
);
```



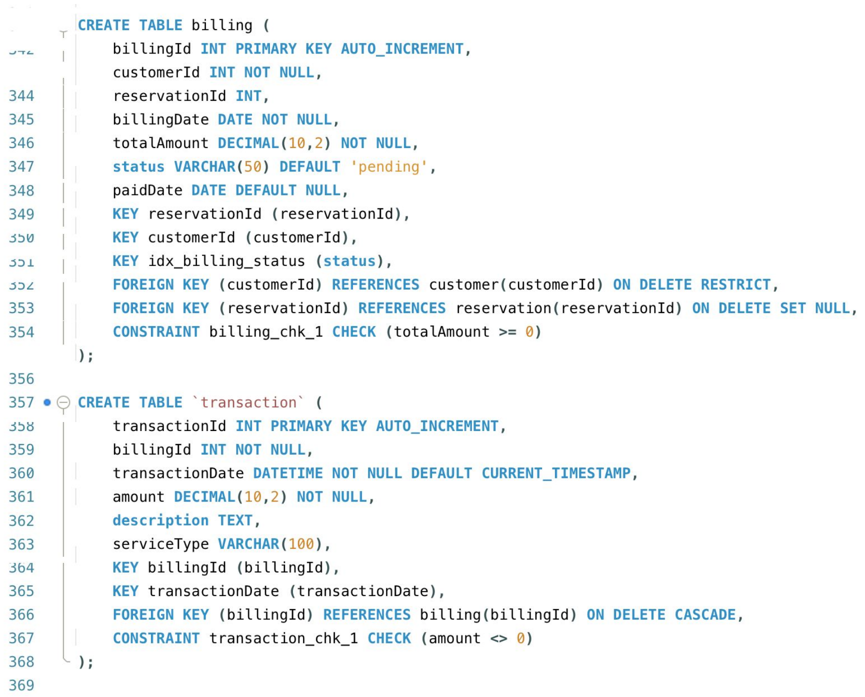
```sql
CREATE TABLE meeting_space (
    roomId INT PRIMARY KEY,
    spaceType VARCHAR(50) NOT NULL,
    capacity INT NOT NULL,
    hasProjector TINYINT(1) DEFAULT 0,
    hasWhiteboard TINYINT(1) DEFAULT 0,
    hasPaidBar TINYINT(1) DEFAULT 0,
    FOREIGN KEY (roomId) REFERENCES room(roomId) ON DELETE CASCADE,
    CONSTRAINT meeting_space_chk_1 CHECK (capacity > 0)
);

CREATE TABLE extra_space (
    extraSpaceId INT PRIMARY KEY AUTO_INCREMENT,
    roomId INT NOT NULL,
    extraSpaceTypeId INT NOT NULL,
    quantity INT DEFAULT 1,
    UNIQUE KEY unique_room_extra (roomId, extraSpaceTypeId),
    FOREIGN KEY (roomId) REFERENCES room(roomId) ON DELETE CASCADE,
    FOREIGN KEY (extraSpaceTypeId) REFERENCES extra_space_type(extraSpaceTypeId) ON DELETE RESTRICT
);

CREATE TABLE suite_room (
    suiteRoomId INT PRIMARY KEY AUTO_INCREMENT,
    suiteId INT NOT NULL,
    roomId INT NOT NULL,
    UNIQUE KEY unique_suite_room (suiteId, roomId),
    FOREIGN KEY (suiteId) REFERENCES suite(suiteId) ON DELETE CASCADE,
    FOREIGN KEY (roomId) REFERENCES room(roomId) ON DELETE CASCADE
);
```

NYU

Database Setup Cont.



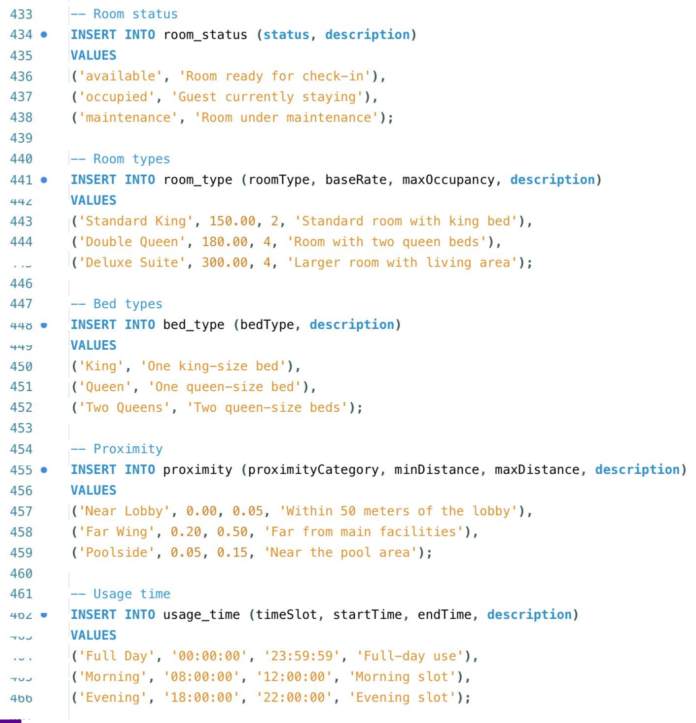
```sql
CREATE TABLE reservation (
    reservationId INT PRIMARY KEY AUTO_INCREMENT,
    startDateTime DATETIME NOT NULL,
    endDateTime DATETIME NOT NULL,
    status VARCHAR(50) DEFAULT 'confirmed',
    createdAt DATETIME DEFAULT CURRENT_TIMESTAMP,
    KEY startDateTime (startDateTime, endDateTime),
    KEY idx_reservation_status (status),
    CONSTRAINT reservation_chk_1 CHECK (endDateTime > startDateTime)
);

CREATE TABLE reservation_party_info (
    partyInfoId INT PRIMARY KEY AUTO_INCREMENT,
    reservationId INT NOT NULL,
    customerId INT NOT NULL,
    partyRole VARCHAR(50) NOT NULL,
    isResponsibleForBilling TINYINT(1) DEFAULT 0,
    FOREIGN KEY (reservationId) REFERENCES reservation(reservationId) ON DELETE CASCADE,
    FOREIGN KEY (customerId) REFERENCES customer(customerId) ON DELETE RESTRICT,
    CONSTRAINT reservation_party_info_chk_1 CHECK (partyRole IN ('booker','occupant','contact'))
);

CREATE TABLE reservation_room (
    reservationRoomId INT PRIMARY KEY AUTO_INCREMENT,
    reservationId INT NOT NULL,
    roomId INT NOT NULL,
    usageTimeId INT,
    UNIQUE KEY unique_res_room (reservationId, roomId, usageTimeId),
    FOREIGN KEY (reservationId) REFERENCES reservation(reservationId) ON DELETE CASCADE,
    FOREIGN KEY (roomId) REFERENCES room(roomId) ON DELETE RESTRICT,
    FOREIGN KEY (usageTimeId) REFERENCES usage_time(usageTimeId) ON DELETE SET NULL
);
```



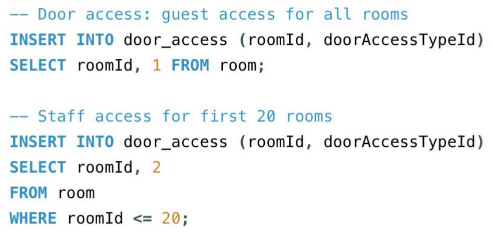
```sql
CREATE TABLE billing (
    billingId INT PRIMARY KEY AUTO_INCREMENT,
    customerId INT NOT NULL,
    reservationId INT,
    billingDate DATE NOT NULL,
    totalAmount DECIMAL(10,2) NOT NULL,
    status VARCHAR(50) DEFAULT 'pending',
    paidDate DATE DEFAULT NULL,
    KEY reservationId (reservationId),
    KEY customerId (customerId),
    KEY idx_billing_status (status),
    FOREIGN KEY (customerId) REFERENCES customer(customerId) ON DELETE RESTRICT,
    FOREIGN KEY (reservationId) REFERENCES reservation(reservationId) ON DELETE SET NULL,
    CONSTRAINT billing_chk_1 CHECK (totalAmount >= 0)
);

CREATE TABLE `transaction` (
    transactionId INT PRIMARY KEY AUTO_INCREMENT,
    billingId INT NOT NULL,
    transactionDate DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
    amount DECIMAL(10,2) NOT NULL,
    description TEXT,
    serviceType VARCHAR(100),
    KEY billingId (billingId),
    KEY transactionDate (transactionDate),
    FOREIGN KEY (billingId) REFERENCES billing(billingId) ON DELETE CASCADE,
    CONSTRAINT transaction_chk_1 CHECK (amount <> 0)
);
```

NYU

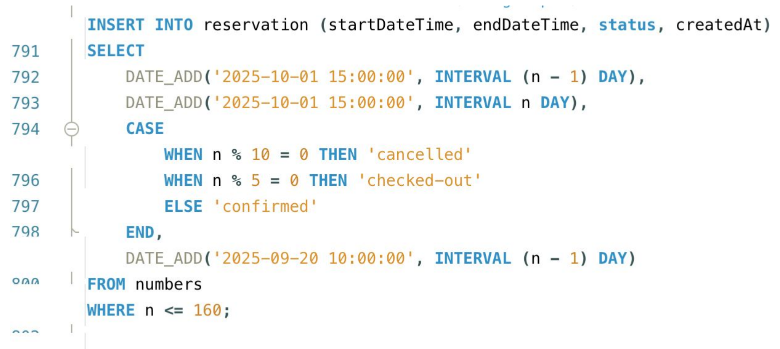
Database Setup Cont.



```sql
-- Room status
INSERT INTO room_status (status, description)
VALUES
('available', 'Room ready for check-in'),
('occupied', 'Guest currently staying'),
('maintenance', 'Room under maintenance');

-- Room types
INSERT INTO room_type (roomType, baseRate, maxOccupancy, description)
VALUES
('Standard King', 150.00, 2, 'Standard room with king bed'),
('Double Queen', 180.00, 4, 'Room with two queen beds'),
('Deluxe Suite', 300.00, 4, 'Larger room with living area');

-- Bed types
INSERT INTO bed_type (bedType, description)
VALUES
('King', 'One king-size bed'),
('Queen', 'One queen-size bed'),
('Two Queens', 'Two queen-size beds');

-- Proximity
INSERT INTO proximity (proximityCategory, minDistance, maxDistance, description)
VALUES
('Near Lobby', 0.00, 0.05, 'Within 50 meters of the lobby'),
('Far Wing', 0.20, 0.50, 'Far from main facilities'),
('Poolside', 0.05, 0.15, 'Near the pool area');

-- Usage time
INSERT INTO usage_time (timeSlot, startTime, endTime, description)
VALUES
('Full Day', '00:00:00', '23:59:59', 'Full-day use'),
('Morning', '08:00:00', '12:00:00', 'Morning slot'),
('Evening', '18:00:00', '22:00:00', 'Evening slot');
```



```sql
-- Door access: guest access for all rooms
INSERT INTO door_access (roomId, doorAccessTypeId)
SELECT roomId, 1 FROM room;

-- Staff access for first 20 rooms
INSERT INTO door_access (roomId, doorAccessTypeId)
SELECT roomId, 2
FROM room
WHERE roomId <= 20;
```



```sql
INSERT INTO reservation (startDateTime, endDateTime, status, createdAt)
SELECT
    DATE_ADD('2025-10-01 15:00:00', INTERVAL (n - 1) DAY),
    DATE_ADD('2025-10-01 15:00:00', INTERVAL n DAY),
    CASE
        WHEN n % 10 = 0 THEN 'cancelled'
        WHEN n % 5 = 0 THEN 'checked-out'
        ELSE 'confirmed'
    END,
    DATE_ADD('2025-09-20 10:00:00', INTERVAL (n - 1) DAY)
FROM numbers
WHERE n <= 160;
```

Scope of the queries

What are we presenting:

- **Inventory & Availability:** Browse/search filters, room types, statuses
- **Occupancy & Scheduling:** Daily snapshot, rooms ↔ reservations
- **Sales & Revenue:** Top customers, billing totals, cash-flow by date
- **Events & Venues:** Capacity/equipment, bookings, sales summaries
- **Service Operations:** Customer requests queue, call logs
- **Access & Security:** Reader swipes, movement, incident signals
- **Structure & Lookups:** Buildings/wings/floors/types for clean grouping

NYU

Queries Overview

```sql
-- Revenue by customers (Billing total)
SELECT c.customerId,
   c.lastName AS last_name,
   c.firstName  AS first_name,
   COUNT(*) AS bills,
   SUM(b.totalAmount) AS total_billed
   FROM billing AS b
   JOIN customer AS c ON c.customerId = b.customerId
   GROUP BY c.customerId, c.lastName, c.firstName
   ORDER BY total_billed DESC;

-- Available sleeping rooms
SELECT rm.roomId, rm.roomNumber, b.buildingName, rt.roomType, bt.bedType,
               rm.squareFootage, rm.hasPaidBar, rs.status AS room_status
    FROM room AS rm
    JOIN building AS b ON b.buildingId = rm.buildingId
    JOIN room_type AS rt ON rt.roomTypeId = rm.roomTypeId
    LEFT JOIN bed_type AS bt ON bt.bedTypeId = rm.bedTypeId
    JOIN room_status AS rs ON rs.roomStatusId = rm.roomStatusId
    LEFT JOIN meeting_space AS ms ON ms.roomId = rm.roomId
    WHERE ms.roomId IS NULL AND rs.status = 'available'
        ORDER BY b.buildingName, rm.roomNumber;
```

```sql
-- Staff list with their info (For management)
SELECT s.staffId, s.firstName, s.lastName, s.email, s.phone,
        s.role, s.department, s.hireDate, s.isActive
    FROM staff AS s
    ORDER BY s.department, s.lastName, s.firstName;

-- Staff door-reader swipes by department and reader location
SELECT s.department, rd.location, COUNT(*) AS staff_swipes
    FROM reading_info AS ri
    JOIN staff_card_assignment AS sca ON sca.staffcardId = ri.staffcardId
    JOIN staff AS s ON s.staffId = sca.staffId
    JOIN readers AS rd ON rd.readersId = ri.readerID
    GROUP BY s.department, rd.location
    ORDER BY staff_swipes DESC;
```

```sql
-- Open customer requests (For front desk employees)
SELECT cr.depositStatus, COUNT(*) AS open_requests
    FROM customer_requests AS cr
    WHERE cr.resolved = 'N'
    GROUP BY cr.depositStatus
    ORDER BY open_requests DESC;
```

NYU

WEB APP DEMO & APPLICATIONS

03

NYU

LINK TO Ed WORKSPACE

https://edstem.org/us/courses/87001/workspaces/pEj6onRTJbWkNtSXTsM9YbkAs9DONcPJ

THANK YOU!