LAST RESORT HOTELS

Database Design & Implementation

Atlas Robinson, Avril Luo, Katie Carlson & Yanxi Chen

AGENDA

01
Entity Relationship Diagram
Overview of our ERD and assumptions made

02
Querying the Database
How the database is structured, how sample data added

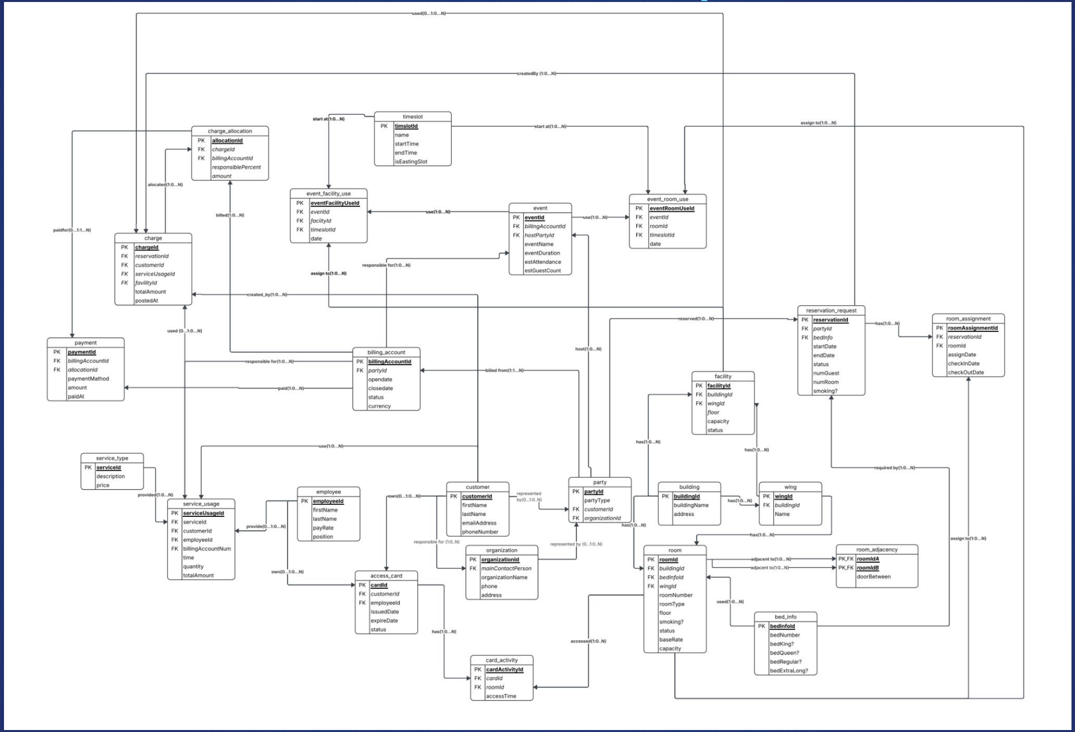
03
The Live Web App
Functions of the website and corresponding queries

04
Live Website Demo
Live demonstration

01

ENTITY RELATIONSHIP DIAGRAM & ASSUMPTIONS

OUR ERD and Assumptions

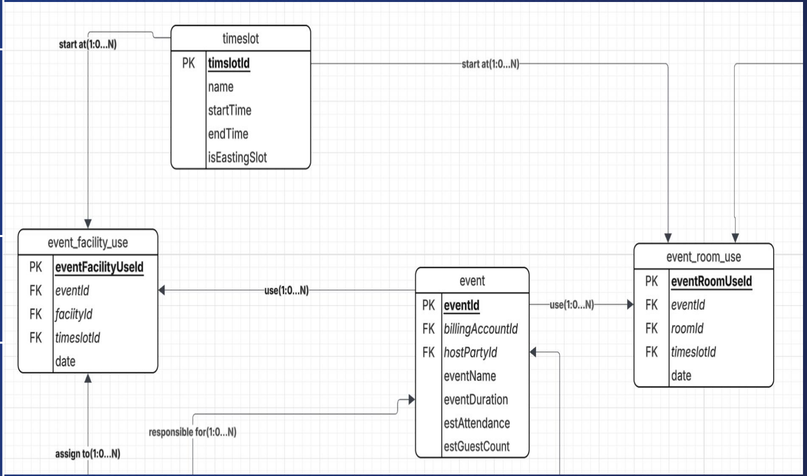


Link:
https://lucid.app/lucidchart/cbbf102e-99cb-4775-ab29-e671778adeb9/edit?invitationId=inv_abcd78d0-0b43-46ec-833b-d7597a16defe&page=krA0yKcaQt-9#

OUR ERD and Assumptions

ERD Example:

Entity	Key Attributes	Relationships
event	**eventId**, *billingAccountId* (FK), *hostPartyId* (FK), eventName, eventDuration, estAttendance, estGuestCount	1→N **event_room_use**, 1→N **event_facility_use**
event_room_use	**eventRoomUseId**, *eventId* (FK), *roomId* (FK), *timeslotId* (FK), date	resolves many-to-many between **event** and **room**
event_facility_use	**eventFacilityUseId**, *eventId* (FK), *facilityId* (FK), *timeslotId* (FK), date	resolves many-to-many between **event** and **facility**
timeslot	**timeslotId**, name, startTime, endTime, isEastingSlot	referenced by event usage tables to manage scheduling



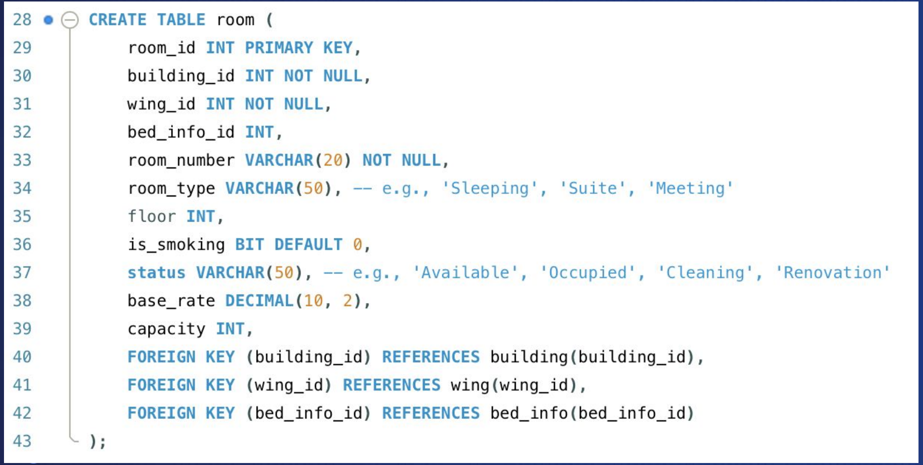
OUR ERD and Assumptions

1. The **hostPartyId** in **event** refers to a record in the party, which may represent a guest or a billing party.
2. **event_room_use** and **event_facility_use** are bridging tables handling M:N relationships with time-based usage.
3. **timeslot** defines fixed scheduling blocks for meetings and events.
4. **billing_account** centralizes payment responsibility for reservations, services, and events.
5. **card_activity** and **access_card** connect personnel to real-time room access logs.

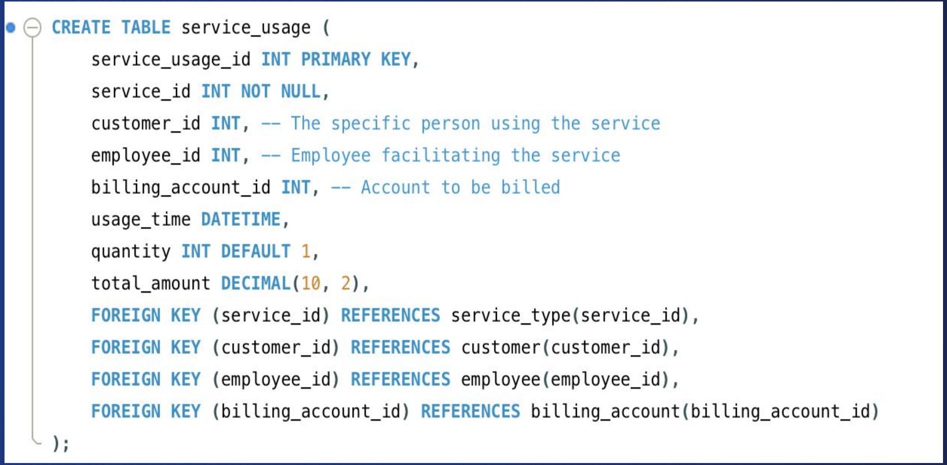
02

THE SQL DATABASE

Sample Table

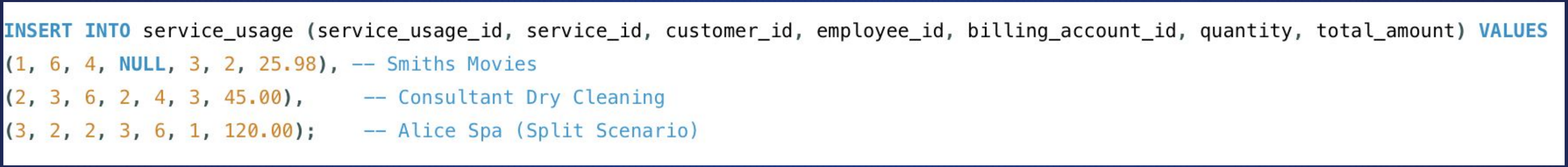


```sql
CREATE TABLE room (
    room_id INT PRIMARY KEY,
    building_id INT NOT NULL,
    wing_id INT NOT NULL,
    bed_info_id INT,
    room_number VARCHAR(20) NOT NULL,
    room_type VARCHAR(50), -- e.g., 'Sleeping', 'Suite', 'Meeting'
    floor INT,
    is_smoking BIT DEFAULT 0,
    status VARCHAR(50), -- e.g., 'Available', 'Occupied', 'Cleaning', 'Renovation'
    base_rate DECIMAL(10, 2),
    capacity INT,
    FOREIGN KEY (building_id) REFERENCES building(building_id),
    FOREIGN KEY (wing_id) REFERENCES wing(wing_id),
    FOREIGN KEY (bed_info_id) REFERENCES bed_info(bed_info_id)
);
```

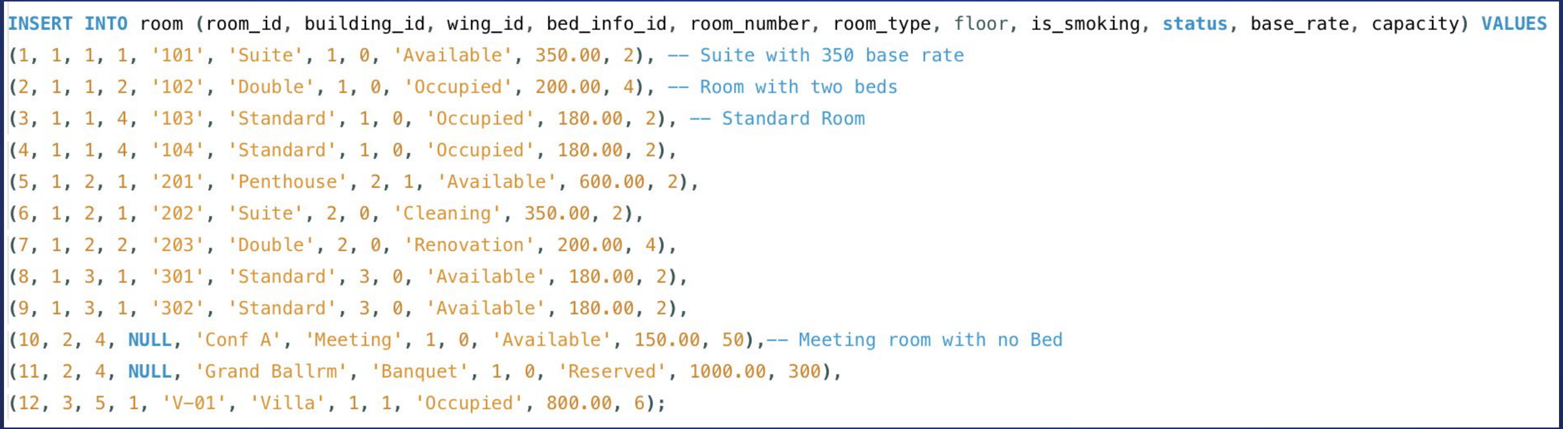


```sql
CREATE TABLE service_usage (
    service_usage_id INT PRIMARY KEY,
    service_id INT NOT NULL,
    customer_id INT, -- The specific person using the service
    employee_id INT, -- Employee facilitating the service
    billing_account_id INT, -- Account to be billed
    usage_time DATETIME,
    quantity INT DEFAULT 1,
    total_amount DECIMAL(10, 2),
    FOREIGN KEY (service_id) REFERENCES service_type(service_id),
    FOREIGN KEY (customer_id) REFERENCES customer(customer_id),
    FOREIGN KEY (employee_id) REFERENCES employee(employee_id),
    FOREIGN KEY (billing_account_id) REFERENCES billing_account(billing_account_id)
);
```

Sample Data



```sql
INSERT INTO service_usage (service_usage_id, service_id, customer_id, employee_id, billing_account_id, quantity, total_amount) VALUES
(1, 6, 4, NULL, 3, 2, 25.98), -- Smiths Movies
(2, 3, 6, 2, 4, 3, 45.00),      -- Consultant Dry Cleaning
(3, 2, 2, 3, 6, 1, 120.00);     -- Alice Spa (Split Scenario)
```



```sql
INSERT INTO room (room_id, building_id, wing_id, bed_info_id, room_number, room_type, floor, is_smoking, status, base_rate, capacity) VALUES
(1, 1, 1, 1, '101', 'Suite', 1, 0, 'Available', 350.00, 2), -- Suite with 350 base rate
(2, 1, 1, 2, '102', 'Double', 1, 0, 'Occupied', 200.00, 4), -- Room with two beds
(3, 1, 1, 4, '103', 'Standard', 1, 0, 'Occupied', 180.00, 2), -- Standard Room
(4, 1, 1, 4, '104', 'Standard', 1, 0, 'Occupied', 180.00, 2),
(5, 1, 2, 1, '201', 'Penthouse', 2, 1, 'Available', 600.00, 2),
(6, 1, 2, 1, '202', 'Suite', 2, 0, 'Cleaning', 350.00, 2),
(7, 1, 2, 2, '203', 'Double', 2, 0, 'Renovation', 200.00, 4),
(8, 1, 3, 1, '301', 'Standard', 3, 0, 'Available', 180.00, 2),
(9, 1, 3, 1, '302', 'Standard', 3, 0, 'Available', 180.00, 2),
(10, 2, 4, NULL, 'Conf A', 'Meeting', 1, 0, 'Available', 150.00, 50),-- Meeting room with no Bed
(11, 2, 4, NULL, 'Grand Ballrm', 'Banquet', 1, 0, 'Reserved', 1000.00, 300),
(12, 3, 5, 1, 'V-01', 'Villa', 1, 1, 'Occupied', 800.00, 6);
```

03

THE LIVE APP FUNCTIONS & QUERIES

Guest Access and Function

- Look up all *available* rooms (with building & wing info)
- See own confirmed reservations
- see own event
- See event details (limited information)
- Look up own billing_account
- see own charges

SQL Example:

```
SELECT r.room_id, r.room_number,
r.room_type, r.base_rate, r.capacity, r.floor,
r.is_smoking, w.wing_name, b.building_name

FROM room r
JOIN wing w ON r.wing_id = w.wing_id
JOIN building b ON r.building_id = b.building_id

WHERE r.status = 'Available';
```

Employee Access and Function

- Look up ALL rooms (with building & wing info)
- Sort rooms (price / status / capacity / room_type)
- Look up all pending requests
- see the room assigned
- Full event info（host, billing, counts）
- occupied rooms/facilities
- available rooms
- See all accounts + total charges
- List all charges

SQL Example:

```
SELECT c.*, ca.responsible_percent, ca.amount

FROM charge c JOIN charge_allocation ca ON
c.charge_id = ca.charge_id

WHERE ca.billing_account_id = %s;
```

04

LIVE WEBSITE DEMONSTRATION

Website Demo

THANKS!