Wolf Inns
Hotel Chain Management System

CSC 540
Database Management Concepts and Systems

Project Report # 1

Team C

Abhay Soni	(asoni3)
Aurora Tiffany-Davis	(attiffan)
Manjusha Trilochan Awasthi	(mawasth)
Samantha Scoggins	(smscoggi)

Table of Contents

Project Assumptions **4**
- Customers 4
- Staff 4
- System Interaction 4
- Rooms 5
- Services 5
- Check-In / Check-Out 6
- Reports 6
- Misc 6

Problem **7**
- Problem Description 7
- Using a database system to address the problem 7

User Classes **8**
- Corporate 8
- Manager 8
- Front Desk Representative 8
- Service Staff 8

Five Main Things **9**
- Hotels 9
- Rooms 9
- Customers 9
- Staff 9
- Stays 10

Realistic Situation **10**

Application Program Interfaces (APIs) **11**
- Information Processing 11
- Maintaining Service Records 13
- Maintaining Billing Accounts 13
- Reports 14
- (Support Methods) 15

Views **16**
- User Class “Corporate” 16
- User Class “Manager” 16
- User Class “Front Desk Representative” 16
- User Class “Service Staff” 16

Local E/R Diagrams **17**
- Legend for Local E/R Diagrams 17

Local E/R Diagram for Corporate View 18
Local E/R Diagram for Manager View 19
Local E/R Diagram for Front Desk Representative View 20
Local E/R Diagram for Service Staff View 21

Local E/R Documentation 22
User Classes 22
Entity Sets 22
Primary Key “ID” 23
Weak / Strong, Supporting / Non-Supporting 24
Relationships 24
Payment Information 26
Minimal Design 26

Local Relational Schema 27
Local Schema for Corporate View 27
Local Schema for Manager View 27
Local Schema for Front Desk Representative View 28
Local Schema for Service Staff View 28

Local Schema Documentation 29
Mechanical Approach 29
E/R Approach for Subclasses 29

Project Assumptions

1. Customers

    1.1. A customer may have at most one phone number (that the system tracks).
    1.2. A phone number may be shared by more than one customer.
    1.3. A customer may have at most one email address (that the system tracks).
    1.4. An email address may be shared by more than one customer.
    1.5. All Customers will have a United States Social Security Number.

2. Staff

    2.1. Each staff member has exactly one job title and department.
    2.2. A staff member's job title indicates what they are expected to do within the system. However, a staff member may need to fill in for a colleague, and so for example a service staff member may provide any type of service regardless of job title.
    2.3. A staff member's department is merely an attribute and is not expected to influence how the staff member interacts with the system.
    2.4. Each staff member is assigned to at most one hotel.
    2.5. A hotel has exactly one manager, and a manager manages exactly one hotel.
    2.6. Because "Front desk representatives… bill customers", "Billing Staff" is just another job title meaning "Front Desk Representative".
    2.7. Staff members know their own staff ID as well as the hotel ID of the hotel they are currently serving.
    2.8. All hotels are appropriately staffed to be able to support presidential suite customers as well as other customers.
    2.9. Staff dedicated to serve the presidential suite during a stay are not specially trained and have no specific job title or other inherent attribute.

3. System Interaction

    3.1. Any user of the system may read or write any information for any hotel.
    3.2. The system has no UI.

4. Rooms

    4.1. Room requests may involve more than room “category” (“Deluxe”, etc.). That is, assigning rooms to customers according to their request criteria may include some or all of: hotel location, room category, room (nightly) rate, maximum occupancy.
    4.2. Room prices vary by location and class of service, but are not strictly defined by location and class of service. That is, each room may be given a unique price when the room is added to the system.
    4.3. Each hotel may have any number of rooms of any given category. For example, “Presidential Suite” is not necessarily unique within a hotel.
    4.4. Reports of occupancy are expected to report occupancy by room, and not by guest (e.g. “4 rooms occupied” rather than “6 guests staying”).
    4.5. “Total occupancy” refers to the total number of rooms occupied.

5. Services

    5.1. Each instance of a service rendered for a customer is provided by one staff member. That is, staff members do not team up to bring room service, etc.
    5.2. The only standard services offered by the hotel are phone bills, dry cleaning, gyms, room service, and catering. All other services are considered “special requests”.
        5.2.1. Every Wolf Inns hotel charges the same rate/minute for phone use.
        5.2.2. Every Wolf Inns hotel charges the same rate/item for dry cleaning.
        5.2.3. Every Wolf Inns hotel charges the same rate/use for gyms.
        5.2.4. Every Wolf Inns hotel charges the same rate/meal for catering.
        5.2.5. Every Wolf Inns hotel charges the same rate for special requests.
        5.2.6. No Wolf Inns hotel charges customers for room service (ordering a meal to be delivered to your room is handled as a catering charge).
    5.3. Wolf Inns does not provide the service of “splitting your bill”.
        5.3.1. Each instance of a service rendered is associated with a single hotel room stay. For example, multiple occupied rooms cannot jointly request catering. That is, there is one bill per room.
        5.3.2. Each hotel bill has one customer responsible for payment.

6. Check-In / Check-Out

    6.1. A check-in cannot proceed without necessary billing information available:
        6.1.1. Customer SSN
        6.1.2. Customer name
        6.1.3. Customer billing address
        6.1.4. Payment method
        6.1.5. Card Type if payment is card:
            6.1.5.1. Example values: Mastercard, Visa, WolfInns
        6.1.6. Card # if payment method is card
    6.2. A check-in can proceed without unnecessary billing information available:
        6.2.1. Card type if payment method is not card
        6.2.2. Card # if payment method is not card
    6.3. The customer who checks in is responsible for paying for their stay. Information about guests physically staying in the rooms is not tracked, with the exception of tracking the number of such guests.
    6.4. The customer may not wish to use the same method of payment for future payments.
    6.5. A bill, and therefore a customer stay, is considered to take place in a single room. That is, if a customer books many rooms (family reunion, etc), separate bills will be generated for each room.
    6.6. Each customer stay will be identified by a generated ID which is unique across all stays regardless of which hotel the stay takes place in.
    6.7. Check-out time and End Date will be predetermined at check-in. The customer will specify how many days the stay will last, and the front desk representative will determine the latest mandatory check-out time. These can be updated upon checkout if necessary.

7. Reports

    7.1. Staff may need to “report occupancy by hotel, room type, date range, and city”, as well as “report total occupancy and percentage of rooms occupied” as noted in the project narrative. What this means is that these may be combined, e.g. “what was the percentage of Basic rooms occupied in Durham last week?” or “what is the current number rooms occupied in all Wolf Inns hotels?” etc.

8. Misc

    8.1. Wolf Inns does not accept reservations.
    8.2. Wolf Inns will never be exposed to a hacking attempt, such that any security related concerns, to include access control, is not a necessary concern in system design.
    8.3. Wolf Inns handles employee payroll, etc. through a separate system.
    8.4. If information that Wolf Inns should maintain is computable from other already explicitly stored information, then it is considered “maintained” and there is no need to explicitly store it.
    8.5. If a question about a customer, stay, hotel, etc. is not required by the project narrative, then there is no need to support it.

Problem

1. Problem Description

   We need a system to support many different tasks that the staff and management of a popular hotel chain need to perform. We need to store, update, and answer questions about information in many topics that relate to each other in complex ways. These topics include hotels, rooms, services, staff, customers, hotel stays, and bills. The amount of information is expected to be large and must be stored long-term. Staff are expected to ask wide-ranging questions about the stored information, and they will need to have accurate answers returned quickly. Likewise, staff will need to have updates to the information applied quickly. Multiple staff members are expected to attempt to make queries / updates at the same time.

2. Using a database system to address the problem

   Database management systems are specifically designed to:

   - Store large amounts of persistent data
     - Such as large amounts of hotel customer stay data
   - Quickly support (perhaps novel) queries about the data
     - Such as many queries about hotels, rooms, customers, bills, services, etc.
   - Quickly perform updates to the data
     - Such as updates to hotels, rooms, customers, bills, services, etc.
   - Safely handle multiple users accessing the data simultaneously.
     - Such as many staff across many hotels checking customers in and out.
   - Ensure consistency of the data
     - Such as single place (tables) to store information rather than multiple copies of information (like hand records). Also database system allows us to add constraints like unique key, primary key which are difficult and time consuming to maintain in traditional file system.

   This description of the capabilities of a DBMS is a wonderful fit for the Wolf Inns system problem described above.

   A simple set of files may be able to store large amounts of persistent data, but our other needs would not be supported nearly so well as with a DBMS. For example, user efficiency would be poor in performing complex queries across a simple set of file. Hardware constraints would limit the capabilities to upkeep a file system. Because it is more efficient to query a database than a simple set of files, we would need better hardware to support the same performance as a database. Additionally, a simple set of files would likely be less efficient in the use of space when compared to a database.

User Classes

1. Corporate

   Corporate users will maintain (enter / update / delete) basic information about hotels, rooms, services, and staff. Corporate users will also run reports on hotels, staff, revenue and occupancy.

2. Manager

   Managers will maintain (enter / update / delete) basic information about rooms within their respective hotel. Managers will also run reports on rooms, staff, revenue and occupancy related to their respective hotel. Finally, managers will run reports on all staff members serving a customer during a particular stay.

3. Front Desk Representative

   Also known as “Billing Staff”, front desk representatives will check room availability according to guest requirements, and will check guests in and out. Checking a guest in may include entering them as a customer into the system, and will include entering billing information (payment method, etc). Front desk representatives will also generate bills and itemized receipts for guests.

4. Service Staff

   Service staff fall under various job titles and will check room service assignments, and enter in services rendered by room number.

Five Main Things

1. Hotels

    1.1. Hotel ID
    1.2. Hotel name
    1.3. Hotel address
    1.4. Hotel city
    1.5. Hotel state
    1.6. Hotel phone number
    1.7. ID of the hotel manager

2. Rooms

    2.1. ID of the hotel
    2.2. Room number
    2.3. Room category
    2.4. Room Capacity (maximum occupancy)
    2.5. Nightly rate

3. Customers

    3.1. SSN
    3.2. Name
    3.3. Date of birth
    3.4. Phone number
    3.5. Email address

4. Staff

    4.1. Staff ID
    4.2. Name
    4.3. Date of Birth
    4.4. Job title
    4.5. Department
    4.6. Phone number
    4.7. Address
    4.8. Hotel currently serving

5. Stays

    5.1. Stay ID
    5.2. Number of guests
    5.3. Start date
    5.4. End date
    5.5. Check-in time
    5.6. Check-out time
    5.7. Payment method
    5.8. Card type
    5.9. Card number
    5.10. Billing Address
    5.11. Room number
    5.12. Hotel ID
    5.13. Customer SSN

Realistic Situation

Mr. Bean plans to visit New York with his family for a vacation. He approaches Wolf Inns Five Seasons Hotel in New York to book an Executive Suite for a week. The two operations (both from the “Information Processing” task) involved in this situation are:

- The front desk representative, Mr. Saggio, *checks if the room type requested is available during the requested week*.
- Mr. Saggio then *assigns the room* to Mr. Bean.

Application Program Interfaces (APIs)

Note: Unless a method argument is accompanied by a note “If NULL is passed for…”, then this argument is expected to always be passed when the method is called.

1. Information Processing

1.1. Hotels

reportAllHotelInfo () the method to actually assign
return *String* allHotelInfo
addHotel (*String* name, *String* address, *String* city, *String* state, *String* phone, *int* managerID)
return *int* hotelID or NULL if error
updateHotelName (*int* hotelID, *String* name)
return *boolean* confirmation
updateHotelLocation (*int* hotelID, *String* address, *String* city, *String* state)
return *boolean* confirmation
updateHotelPhone (*int* hotelID, *String* phone)
return *boolean* confirmation
updateHotelManager(*int* hotelID, *String* managerID)
return *boolean* confirmation
deleteHotel (*int* hotelID)
return *boolean* confirmation

1.2. Rooms

reportRoomInfoForHotel (*int* hotelID)
return *String* roomInfo or NULL if error
addRoom (*int* hotelID, *int* roomNumber, *String* roomCategory, *int* roomCapacity, *float* nightlyRate)
return *boolean* confirmation
updateRoomCategory (*int* hotelID, *int* roomNumber, *String* roomCategory)
return *boolean* confirmation
updateRoomCapacity (*int* hotelID, *int* roomNumber, *int* roomCapacity)
return *boolean* confirmation
updateRoomRate (*int* hotelID, *int* roomNumber, *float* nightlyRate)
return *boolean* confirmation
deleteRoom (*int* hotelID, *int* roomNumber)
return *boolean* confirmation
reportStayIDbyRoom (*int* hotelID, *int* roomNumber)
return *int* StayID or *null* if error

1.3. Staff

* (also useful: “reportStaffInfo” method in “Reports” section)

addStaff (*String* name, *String* dateOfBirth, *String* jobTitle, *String* department, *String* phone, *String* address, *int* hotelID)
return *int* staffID or NULL if error
- NULL may be passed for hotelID, because a staff member may be added without yet being assigned to a particular hotel

updateStaffName (*int* staffID, *String* name)
return *boolean* confirmation

updateStaffDateOfBirth (*int* staffID, *String* dateOfBirth)
return *boolean* confirmation

updateStaffRole (*int* staffID, *String* jobTitle, *String* department)
return *boolean* confirmation

updateStaffPhoneNumber (*int* staffID, *String* phoneNumber)
return *boolean* confirmation

updateStaffAddress (*int* staffID, *String* address)
return *boolean* confirmation

updateStaffHotel (*int* staffID, *int* hotelID)
return *boolean* confirmation

deleteStaff (*int* staffID)
return *boolean* confirmation

1.4. Customers

reportCustomerInfo ()
return *String* customerInfo or NULL if error

addCustomer (*int* SSN, *String* name, *String* dateOfBirth, *String* phone, *String* email)
return *boolean* confirmation

updateCustomerName (*int* SSN, *String* name)
return *boolean* confirmation

updateCustomerDateOfBirth (*int* SSN, *String* dateOfBirth)
return *boolean* confirmation

updateCustomerPhoneNumber (*int* SSN, *String* phoneNumber)
return *boolean* confirmation

updateCustomerEmail (*int* SSN, *String* email)
return *boolean* confirmation

deleteCustomer (*int* SSN)
return *boolean* confirmation

1.5. Check In / Check out

checkRoomAvailability (*int* hotelID, *String* roomCategory, *int* roomCapacity, *float* roomRate, *int* roomNumber)
return *int[ ]* roomIDs or NULL if error
- If NULL is passed for roomCategory, then available rooms will not be filtered to include only rooms of a certain category
- If NULL is passed for roomCapacity, then available rooms will not be filtered to include only rooms of a certain capacity or greater
- If NULL is passed for roomRate, then available rooms will not be filtered to include only rooms of a certain nightly rate or lower
- if NULL is passed for roomNumber, then available rooms will not be filtered to include only rooms of a certain room number
- [this method makes use of getMatchingRooms method]

assignRoom (*int* hotelID, *int* roomNumber, *int* SSN, *String* paymentMethod, *String* cardType, *int* cardNumber, *String* billingAddress, *int* staffID)
return *int* stayID or NULL if error

releaseRoom (*int* hotelID, *int* roomNumber, *int* staffID)
return *boolean* confirmation

2. Maintaining Service Records

reportServiceRecordsForStay (*int* serviceRecordID)
return *String* serviceRecords or NULL if error

addServiceRecord (*int* hotelID, *int* roomNumber, *String* nameOfService, *int* staffID)
return *int* serviceRecordID or NULL if error

updateServiceRecordType (*int* serviceRecordID, *String* nameOfService)
return *boolean* confirmation

updateServiceRecordStaff (*int* serviceRecordID, *int* staffID)
return *boolean* confirmation

deleteServiceRecord (*int* serviceRecordID)
return *boolean* confirmation

reportServiceTypes ()
return *String* serviceTypes or NULL if error

addServiceType (*String* nameOfService, *int* costOfService)
return *boolean* confirmation

deleteServiceType (*String* nameOfService)
return *boolean* confirmation

3. Maintaining Billing Accounts

generateBill (*int* stayID)
return *int* amountOwed

generateReceipt (*int* stayID)
return *String* itemizedReceipt

4. Reports

reportOccupancyRooms (*int* hotelID, *String* roomCategory, *int* roomCapacity, *float* roomRate, *String startDate*, String endDate, *String* city)

return *int* numRoomsOccupied or NULL if error

- If NULL is passed for hotelID, then rooms will not be filtered to include only rooms within a certain hotel
- If NULL is passed for city, then rooms will not be filtered to include only rooms within a certain city
- If NULL is passed for roomCategory, then rooms will not be filtered to include only rooms of a certain category
- If NULL is passed for roomCapacity, then rooms will not be filtered to include only rooms of a certain capacity or greater
- If NULL is passed for roomRate, then rooms will not be filtered to include only rooms of a certain nightly rate or lower
- If NULL is passed for startDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range
- If NULL is passed for endDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range
- [this method makes use of getMatchingRooms method]

reportOccupancyPercentage (*int* hotelID, *String* roomCategory, *int* roomCapacity, *float* roomRate, *String startDate*, String endDate, *String* city)

return *float* percentageRoomsOccupied or NULL if error

- If NULL is passed for hotelID, then rooms will not be filtered to include only rooms within a certain hotel
- If NULL is passed for city, then rooms will not be filtered to include only rooms within a certain city
- If NULL is passed for roomCategory, then rooms will not be filtered to include only rooms of a certain category
- If NULL is passed for roomCapacity, then rooms will not be filtered to include only rooms of a certain capacity or greater
- If NULL is passed for roomRate, then rooms will not be filtered to include only rooms of a certain nightly rate or lower
- If NULL is passed for startDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range
- If NULL is passed for endDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range
- [this method makes use of getMatchingRooms method]

reportTotalOccupancyRooms (*int* hotelID)

return *int* numRoomsOccupied or NULL if error

- If NULL is passed for hotelID, then rooms will not be filtered to include only rooms within a certain hotel
- [this method makes use of reportOccupancyRooms method]

reportTotalOccupancyPercentage (*int* hotelID)
return *float* percentageRoomsOccupied or NULL if error
- If NULL is passed for hotelID, then rooms will not be filtered to include only rooms within a certain hotel
- [this method makes use of reportOccupancyPercentage method]

reportStaffInfo (*int* hotelID)
return *String* staffInfo or NULL if error
- If NULL is passed for hotelID, then staff members will not be filtered to include only members which currently serve a certain hotel

reportStaffServingDuringStay (*int* stayID)
return *String* staffServing or NULL if error

reportHotelRevenue (*int* hotelID, *String* startDate, *String* endDate)
return *float* hotelRevenue or NULL if error

5. (Support Methods)

getMatchingRooms (*int* hotelID, *String* city, *String* roomCategory, *int* roomCapacity, *float* roomRate, *boolean* available, *String* startDate, *String* endDate)
return *int[ ]* roomIDs or NULL if error
- If NULL is passed for hotelID, then rooms will not be filtered to include only rooms within a certain hotel
- If NULL is passed for city, then rooms will not be filtered to include only rooms within a certain city
- If NULL is passed for roomCategory, then rooms will not be filtered to include only rooms of a certain category
- If NULL is passed for roomCapacity, then rooms will not be filtered to include only rooms of a certain capacity or greater
- If NULL is passed for roomRate, then rooms will not be filtered to include only rooms of a certain nightly rate or lower
- If NULL is passed for available, then rooms will not be filtered to include only rooms based on room availability
- If NULL is passed for startDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range
- If NULL is passed for endDate, then rooms will not be filtered to include only rooms which match the given criteria during a certain date range

reportStayID (*int* SSN, *int* hotelID, *int* roomNumber, *String* date)
return *int* stayID

Views

1. User Class “Corporate”

    1.1. All hotels
    1.2. All rooms within each hotel
    1.3. Occupancy by hotel, room type, date range, or city
    1.4. Total occupancy and percentage of rooms occupied across all hotels
    1.5. All staff
    1.6. Revenue earned by a hotel during a date range
    1.7. WolfInns services and costs

2. User Class “Manager”

    2.1. All rooms within a hotel
    2.2. Occupancy by room type or date range, hotel
    2.3. Total occupancy and percentage of rooms occupied, by hotel
    2.4. Hotel staff, grouped by role (job title), hotel
    2.5. Revenue earned by a hotel during a date range
    2.6. All staff members serving the customer during a stay

3. User Class “Front Desk Representative”

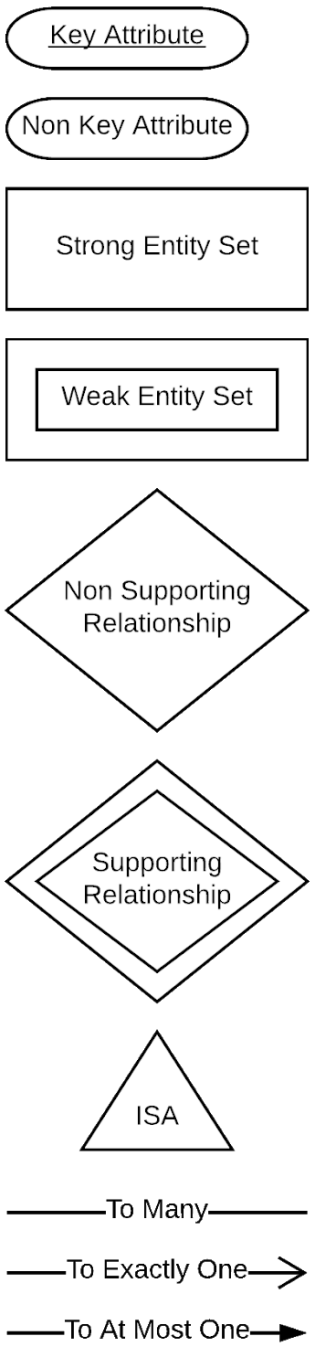
    3.1. All available rooms in a hotel which match customer preferences
    3.2. All customer information
    3.3. Customer stay information in a hotel
    3.4. A bill for a stay, including total amount owed, as well as an itemized receipt

4. User Class “Service Staff”

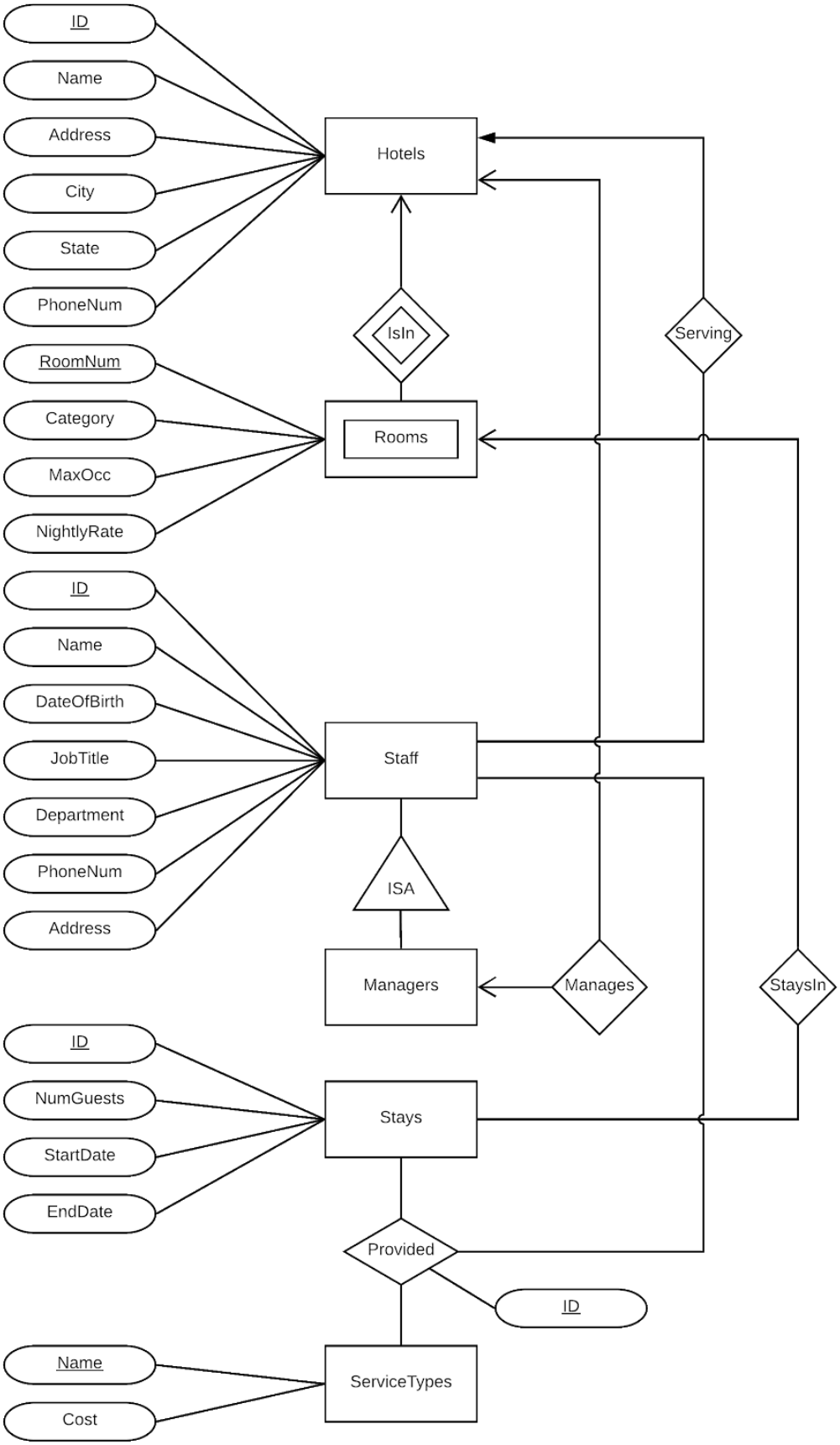
    4.1. Services rendered to a particular room during a stay, as well as which staff members provided these services

Local E/R Diagrams

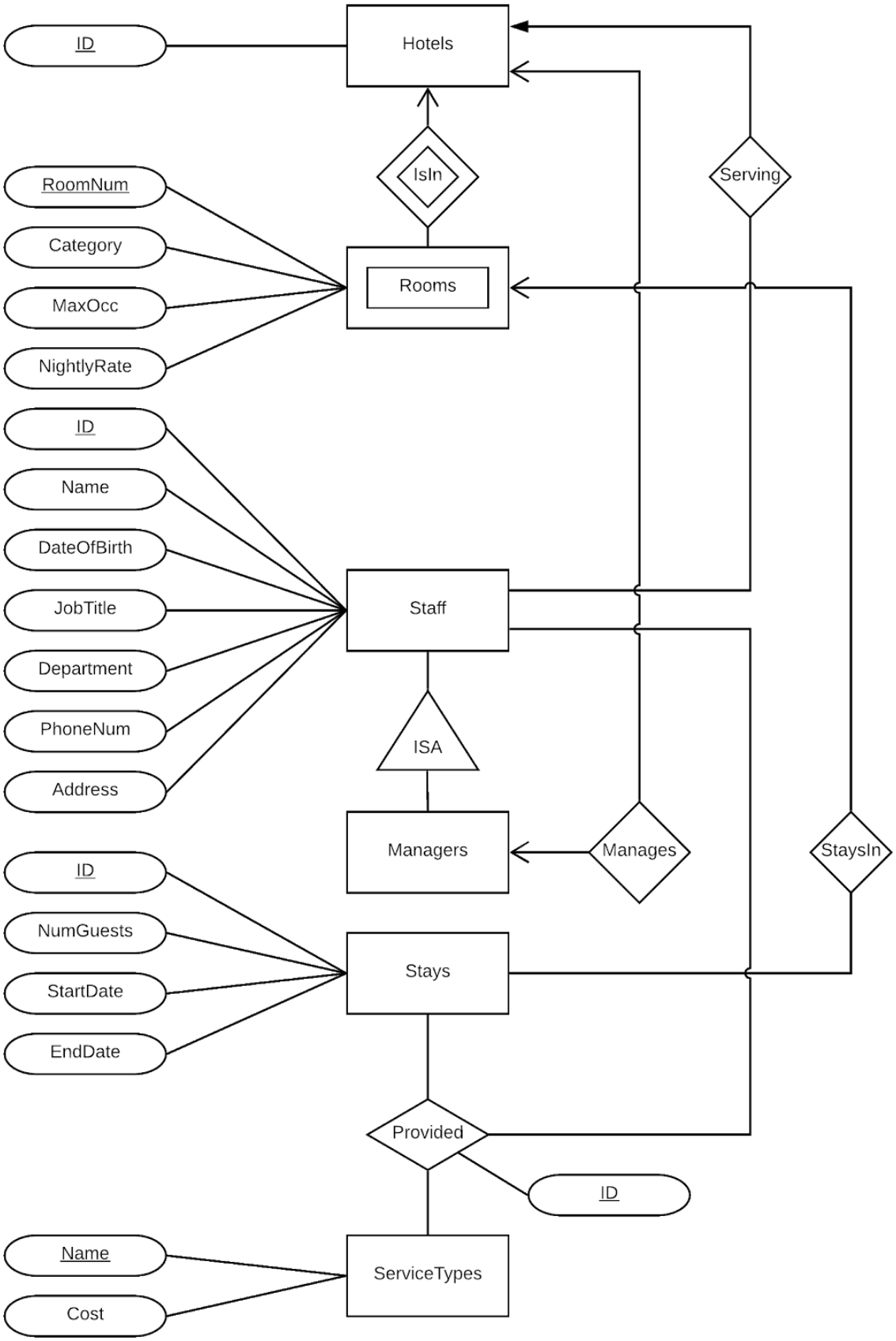
1. Legend for Local E/R Diagrams



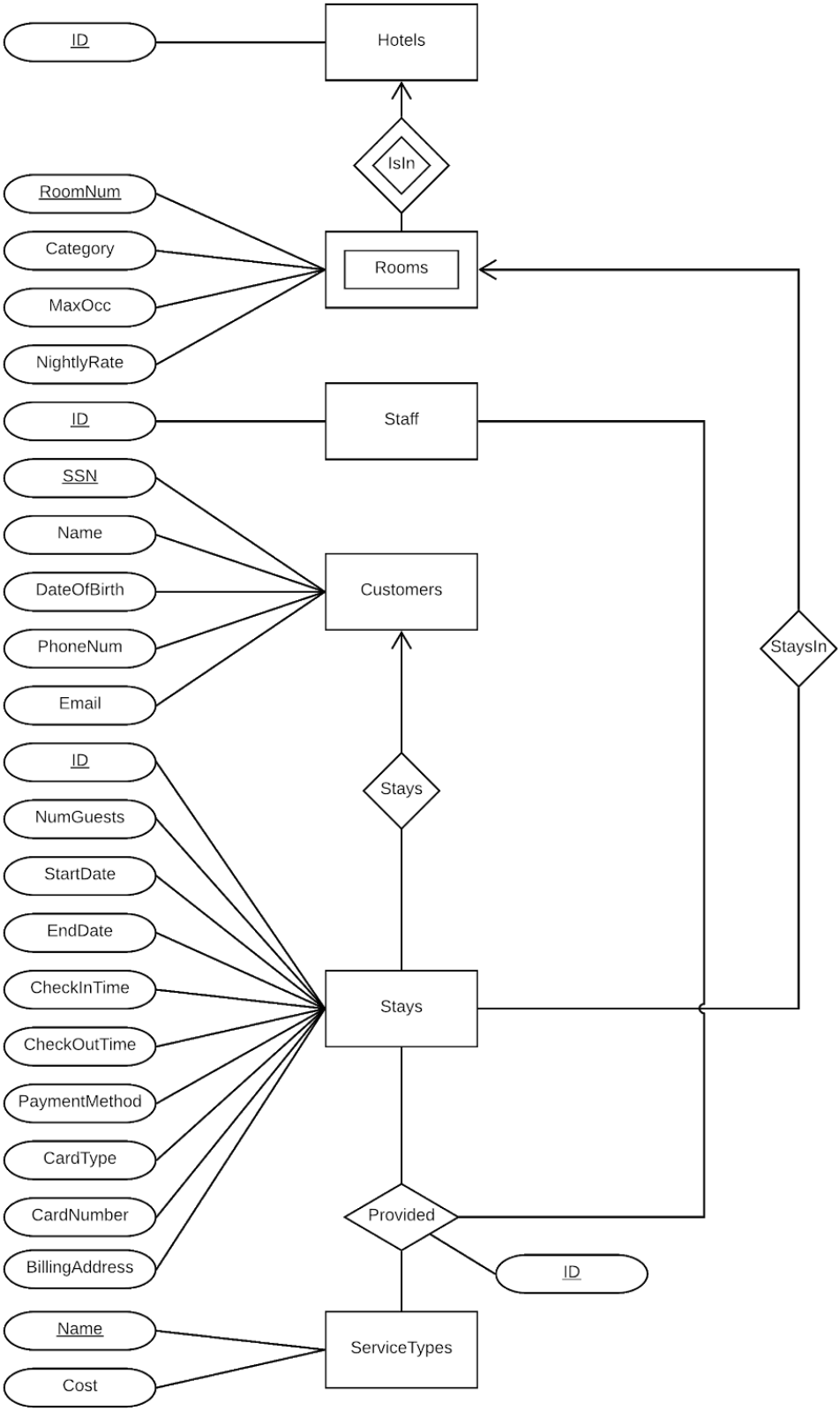
2. Local E/R Diagram for Corporate View



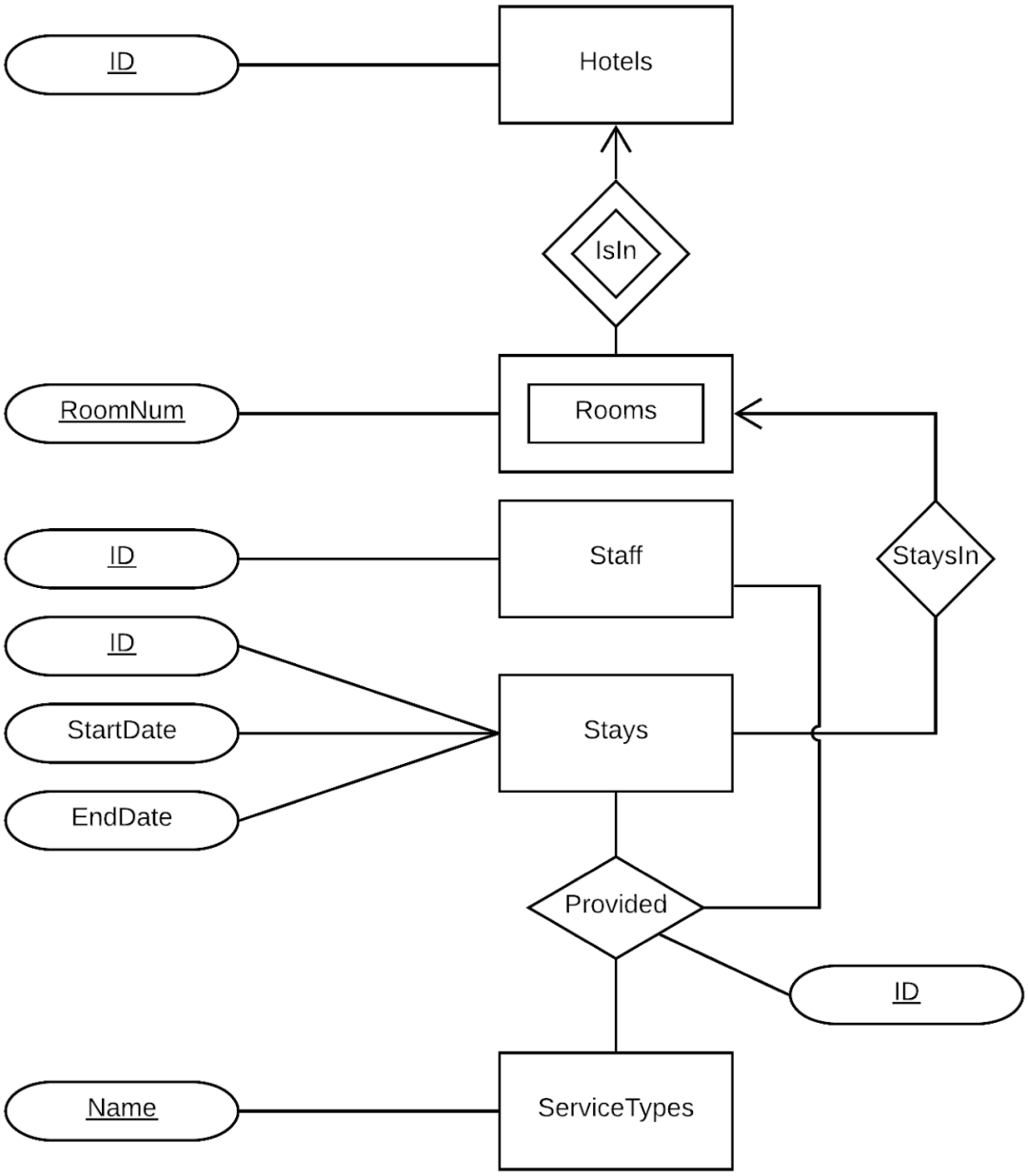
3. Local E/R Diagram for Manager View



4. Local E/R Diagram for Front Desk Representative View



5. Local E/R Diagram for Service Staff View



Local E/R Documentation

1. User Classes

   As noted earlier in the report, our user classes are “Corporate”, “Manager”, “Front Desk Representative”, and “Service Staff”. Each of these has a local E/R diagram.

2. Entity Sets

   Each entity set was established for a specific reason.

   2.1. Hotels

   If hotels lacked an entity set and were tracked as attributes within another entity set, we could expect massive redundancy, with all of the update and deletion anomaly possibilities that come with that. Many rooms may exist in the same hotel.

   2.2. Rooms

   This entity set was established for similar reasons as with Hotels. Many stays may take place in the same room.

   2.3. Staff

   This entity set was established for similar reasons as with Hotels. Many services rendered may be provided by the same staff member.

   2.4. Managers

   This entity set was established in order to indicate the unique constraint on managers that does not exist for other staff types. A hotel must have exactly one manager, and a manager must manage exactly one hotel.

   2.5. Customers

   This entity set was established for similar reasons as with Hotels. Many stays may be booked by the same customer.

   2.6. Stays

   This entity set was established for similar reasons as with Hotels. Many services may be rendered during the same stay.

2.7. ServiceTypes

This entity set was established for similar reasons as with Hotels. Many services of the same type may be rendered.

3. Primary Key “ID”

The entity sets “Rooms”, “Customers”, and “ServiceTypes” do not have an “ID” primary key

3.1. Rooms

There is a easy way to uniquely identify a Wolf Inns room (by room number and hotel) and this is why no extra “ID” was added.

3.2. Customers

There is an easy way to uniquely identify a customer (by social security number) and this is why no extra “ID” was added.

3.3. ServiceTypes

There is an easy way to uniquely identify a service type (by name) and this is why no extra “ID” was added.

All other entity sets have an “ID” primary key which is only meaningful in that it uniquely identifies entities in the set.

3.4. Hotels

The use of a hotel ID is specified in the project narrative. Otherwise, we likely would have uniquely identified hotels by address.

3.5. Staff

The use of a staff ID is specified in the project narrative. The use of a staff ID is preferred because without an ID, staff members would require a compound key with many attributes. Multiple staff members may share the same name, age, job title, contact information, etc.

3.6. Stays

The use of a stay ID is preferred because without an ID, stays would require a compound key with many attributes. Multiple stays may share the same number of guests, start date, end date, hotel, etc.

4. Weak / Strong, Supporting / Non-Supporting

4.1. The entity set “Rooms” is a weak entity set, supported by the relationship “IsIn”.

As noted above, there is a easy way to uniquely identify a Wolf Inns room (room number and hotel) and this is why no extra “ID” was added. This makes a weak entity set and supporting relationship.

We are comfortable with this because we believe it is faithful to the domain of a hotel chain.

4.2. The entity set “Managers” has an “ISA” relationship to “Staff”.

The only interesting thing about a manager from an E/R diagram perspective is that there is a constraint on managers that does not exist for other staff types. A hotel must have exactly one manager, and a manager must manage exactly one hotel.

4.3. All other entity sets are strong, and all other relationships non-supporting.

In the absence of a specific reason to make an entity set weak, it is strong by default to avoid extra references when performing queries.

5. Relationships

Relationships, and their associated constraints, were each established for a specific reason.

5.1. IsIn

- This relationship is necessary to to link rooms to hotels, for example to answer queries about occupancy rates within hotels.
- A given room must exist in ***exactly one*** hotel.
- A hotel may, and certainly is expected to, have ***many*** rooms.

5.2. Serving

- This relationship is necessary to link staff members to the hotels they serve, as this link is specified in the project narrative.
- A Wolf Inns employee is assigned to ***at most one*** hotel. This is one of our project assumptions.
- A hotel may, and certainly is expected to, have ***many*** staff members assigned to it.

5.3. ISA

- This relationship is necessary to show that manager are a type of staff.

5.4. Manages

- This relationship is necessary to link managers to the hotels that they manage.
- A manager manages ***exactly one*** hotel. This is one of our project assumptions.
- A hotel is managed by ***exactly one*** manager. This is one of our project assumptions.

5.5. StaysIn

- This relationship is necessary to link hotel stays to specific rooms, for example to answer queries about occupancy rates.
- A hotel stay is associated with ***exactly one*** hotel room. This is one of our project assumptions.
- A hotel room may, and certainly is hoped to, have ***many*** stays associated with it.

5.6. Stays

- This relationship is necessary to link hotel stays to customers, for example to generate bills for customers at the end of their stay.
- A stay is associated with ***exactly one*** customer. Although several guests may stay in a single room, there is one customer who checks in and out.
- One customer may, and certainly is hoped to (repeat business) be associated with ***many*** hotel stays.

5.7. Provided

- This relationship is necessary to link services types provided, to the staff members who provided them, and to the hotel stays for which they were provided.
- One service type, provided during one hotel stay, may be provided (over many instances) by ***many*** staff members. For example, Joe may bring room service to Raleigh room 305, and the following day Mary may also bring room service to Raleigh room 305.
- One service type, provided by one staff member, may be provided for ***many*** hotel stays. For example, Jow may bring room service to Raleigh room 305, and also to Raleigh room 136.
- One staff member may, for one hotel stay, provide ***many*** service types. This is one of our project assumptions (staff may fill in for colleagues).

6. Payment Information

   Payment method, card type, card number, and billing address are all attributes of a stay, rather than attributes of a customer. This is to facilitate the project assumption we have made that a customer may use different payment methods on different stays.

7. Minimal Design

   7.1. No additional entity sets or relationships are included because without a compelling reason to include such relations, they would introduce inefficiencies to our system.

   7.2. Local E/R diagrams include only those entity sets, relationships, and attributes necessary to support the user class view in question. Thus, for example:

   - The Service Staff local E/R diagram includes only the primary key attribute “Name” for the “ServiceType” entity set. Service staff do not need to know, or see, how much a particular service type costs.

   - The Front Desk Representative local E/R diagram includes both “Name” and “Cost” for the “ServiceType” entity set. Front desk representatives need access to cost information in order to generate a bill and itemized receipt for the customer.

Local Relational Schema

1. Local Schema for Corporate View

   Hotels (ID, Name, Address, City, State, PhoneNum, ManagerID)

   Rooms (RoomNum, HotelID, Category, MaxOcc, NightlyRate)

   Staff (ID, Name, DateOfBirth, JobTitle, Department, PhoneNum, Address, HotelID)

   Managers (ID, HotelID)

   Stays (ID, NumGuests, StartDate, EndDate, RoomNum, HotelID)

   ServiceTypes (Name, Cost)

   Provided (ID, StayID, StaffID, ServiceName)

2. Local Schema for Manager View

   Hotels (ID)

   Rooms (RoomNum, HotelID, Category, MaxOcc, NightlyRate)

   Staff (ID, Name, DateOfBirth, JobTitle, Department, PhoneNum, Address, HotelID)

   Managers (ID, HotelID)

   Stays (ID, NumGuests, StartDate, EndDate, RoomNum, HotelID)

   ServiceTypes (Name, Cost)

   Provided (ID, StayID, StaffID, ServiceName)

3. Local Schema for Front Desk Representative View

   Hotels (ID)

   Rooms (RoomNum, HotelID, Category, MaxOcc, NightlyRate)

   Staff (ID)

   Customers (SSN, Name, DateOfBirth, PhoneNum, Email)

   Stays (ID, NumGuests, StartDate, EndDate, CheckInTime, CheckOutTime, PaymentMethod, CardType, CardNumber, BillingAddress, RoomNum, HotelID, CustomerSSN)

   ServiceTypes (Name, Cost)

   Provided (ID, StayID, StaffID, ServiceName)

4. Local Schema for Service Staff View

   Hotels (ID)

   Rooms (RoomNum, HotelID)

   Staff (ID)

   Stays (ID, StartDate, EndDate, RoomNum, HotelID)

   ServiceTypes (Name)

   Provided (ID, StayID, StaffID, ServiceName)

Local Schema Documentation

1. Mechanical Approach

   Translations from local E/R diagrams to local relation schemas were done mechanically.

   1.1. Each entity set was translated into a relation schema with the same set of attributes as shown in the entity set.
   1.2. Each many to one relationship was translated into additional attributes from the “one” side of the relationship, shown in the schema representing the “many” side of the relationship.
   1.3. Each many to many relationship was translated into a relation schema with attributes representing the keys for the participating entity sets.

2. E/R Approach for Subclasses

   The E/R style approach was used to translate the “Manager” subclass entity set.

   2.1. One extra “Managers” relation schema was created for the subclass, with the key attribute of the “Staff” schema serving as the unique identifier for entities in the “Managers” set.

   2.2. The E/R approach was used for several reasons.

      2.2.1. Because this approach mimics the structure of the E/R diagram, it is intuitive to the system designers.

      2.2.2. This approach allows for compact storage with an eye to system flexibility.

         - If managers, in the future, required additional attributes, this approach would not result in the proliferation of NULL values as would be seen in the Nulls approach.

         - If, in the future, additional subclasses of Staff were required, this approach would not result in the proliferation of relation schemas as would be seen in the object-oriented approach.

         - We do not expect that the lookup speed from the “Managers” schema to the “Staff” schema in order to query, for example, the manager’s name, will be overly burdensome.

            - There is no project task or operation indicating that this type of lookup will be routine.
            - Therefore we do not think that this inherent “con” in using the E/R approach will impact us to an unacceptable degree.