Project TDS I - Hotel

Richard Ficek, FIC0024

June 9, 2025

Contents

1 DD S01 L02

`Task:` Explain the difference between the concept of data and information - an example in your project written in English

Data are raw, unprocessed facts without context or meaning. **Information** is data that has been processed, organized, or interpreted to provide meaning or value.

Example from the project:

- **Data:** In the `Guest` table, a single record such as:

  > firstname: "John", lastname: "Smith", email: "john.smith@email.com", birth_date: "1985-06-15"

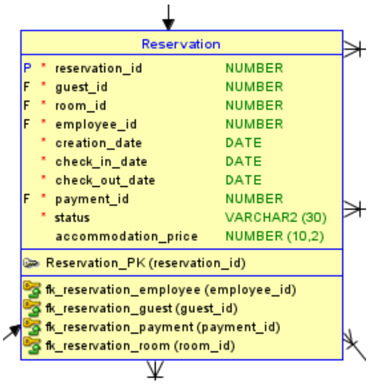
  is just a set of data points about a guest.

  Similarly, in the `Reservation` table, several records might look like:

reservation_id	1
guest_id	5
room_id	101
employee_id	3
creation_date	2024-04-01
check_in_date	2024-04-10
check_out_date	2024-04-15
payment_id	7
status	`"confirmed"`
accommodation_price	500.00

  These are just raw data entries about guests and reservations.

- **Information:** If we process the data, we can find that "Most guests who stayed in the hotel in 2024 were born after 1990," or, after analyzing the reservations, we can obtain information such as: *"In April 2024, the average length of stay for confirmed reservations was 3.5 nights, and the average accommodation price was 475.00. Most reservations were made for mid-April."*
  This information provides insight and helps in decision-making, such as targeting marketing campaigns to a younger audience or optimizing pricing and staffing for busy periods.



Reservation

P	*	reservation_id	NUMBER
F	*	guest_id	NUMBER
F	*	room_id	NUMBER
F	*	employee_id	NUMBER
	*	creation_date	DATE
	*	check_in_date	DATE
	*	check_out_date	DATE
F	*	payment_id	NUMBER
	*	status	VARCHAR2 (30)
		accommodation_price	NUMBER (10,2)

Reservation_PK (reservation_id)

fk_reservation_employee (employee_id)
fk_reservation_guest (guest_id)
fk_reservation_payment (payment_id)
fk_reservation_room (room_id)

2 DD S02 L02

`Task:` Entities, instances, attributes and identifiers - describe in examples on your project

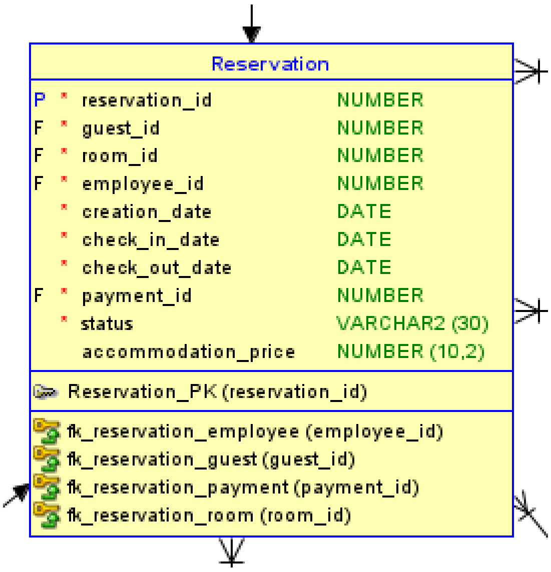
- **Entity:** An entity is an object or concept about which data is stored. In the hotel project, examples of entities are `Guest`, `Reservation`, `Room`, and `Employee`.
- **Instance:** An instance is a specific occurrence of an entity. For example, a single guest record: `guest_id: 1, firstname: "John", lastname: "Smith", email: "john.smith@email.com", birth_date: "1985-06-15"`
  is an instance of the `Guest` entity.
- **Attribute:** An attribute is a property or characteristic of an entity. For the `Reservation` entity, attributes include `reservation_id`, `guest_id`, `room_id`, `check_in_date`, `status`, and `accommodation_price`.
- **Identifier:** An identifier uniquely distinguishes each instance of an entity. For example, `guest_id` is the identifier for the `Guest` entity, and `reservation_id` is the identifier for the `Reservation` entity.

Example:

```
reservation_id: 5, guest_id: 2, room_id: 201, employee_id: 1, creation_date:
2024-05-01, check_in_date: 2024-05-10, check_out_date: 2024-05-15, payment_id:
4, status: "confirmed", accommodation_price: 600.00
```

In this example:

- `Reservation` is the entity.
- The quoted record is an instance of the `Reservation` entity.
- Each field (e.g., `check_in_date`, `status`) is an attribute.
- `reservation_id` is the identifier.



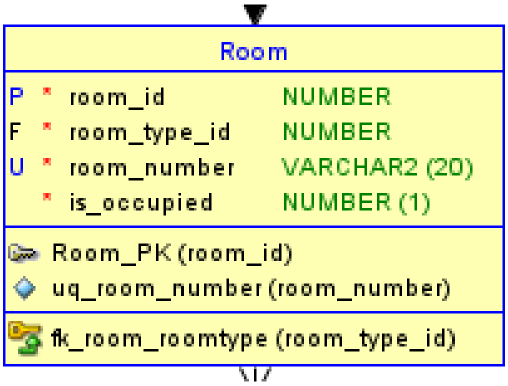
Reservation

P	*	reservation_id	NUMBER
F	*	guest_id	NUMBER
F	*	room_id	NUMBER
F	*	employee_id	NUMBER
	*	creation_date	DATE
	*	check_in_date	DATE
	*	check_out_date	DATE
F	*	payment_id	NUMBER
	*	status	VARCHAR2 (30)
		accommodation_price	NUMBER (10,2)

Reservation_PK (reservation_id)

fk_reservation_employee (employee_id)
fk_reservation_guest (guest_id)
fk_reservation_payment (payment_id)
fk_reservation_room (room_id)

Figure 1: Rezervace



Room

P	*	room_id	NUMBER
F	*	room_type_id	NUMBER
U	*	room_number	VARCHAR2 (20)
	*	is_occupied	NUMBER (1)

Room_PK (room_id)
uq_room_number (room_number)

fk_room_roomtype (room_type_id)

Figure 2: Pokoj

Database Tables and Attributes

Attributes: Guest

Attribute	Description
guest_id	Unique identifier for guest (auto-generated)
firstname	Guest's first name
lastname	Guest's last name
email	Guest's email address
phone	Guest's phone number
birth_date	Guest's date of birth
street	Guest's street address
city	Guest's city
postal_code	Guest's postal code
country	Guest's country
guest_type	Type of guest (standard, VIP, etc.)
registration_date	Date when guest registered (default: current date)
manager_id	Reference to managing employee
notes	Additional notes about the guest

Attributes: Employee

Attribute	Description
employee_id	Unique identifier for employee (auto-generated)
firstname	Employee's first name
lastname	Employee's last name
position	Employee's job position
street	Employee's street address
city	Employee's city
postal_code	Employee's postal code
country	Employee's country

Attributes: RoomType

Attribute	Description
room_type_id	Unique identifier for room type (auto-generated)
name	Name of the room type
bed_count	Number of beds in this room type

Attributes: Room

Attribute	Description
room_id	Unique identifier for room (auto-generated)
room_type_id	Reference to room type
room_number	Unique room number
is_occupied	Room occupation status (0=free, 1=occupied)

Attributes: Payment

Attribute	Description
payment_id	Unique identifier for payment (auto-generated)
total_accommodation	Total accommodation cost
total_expenses	Total additional expenses
payment_date	Date of payment
is_paid	Payment status (0=unpaid, 1=paid)

Attributes: Reservation

Attribute	Description
reservation_id	Unique identifier for reservation (auto-generated)
guest_id	Reference to guest making the reservation
room_id	Reference to reserved room
employee_id	Reference to employee handling the reservation
creation_date	Date when reservation was created
check_in_date	Guest check-in date
check_out_date	Guest check-out date
payment_id	Reference to payment for this reservation
status	Reservation status (confirmed, cancelled, etc.)
accommodation_price	Price for accommodation

Attributes: Service

Attribute	Description
service_id	Unique identifier for service (auto-generated)
name	Name of the service
description	Detailed description of the service

Attributes: ServiceUsage

Attribute	Description
usage_id	Unique identifier for service usage (auto-generated)
reservation_id	Reference to reservation using the service
service_id	Reference to service being used
quantity	Quantity of service used
total_price	Total price for service usage

Attributes: Feedback

Attribute	Description
feedback_id	Unique identifier for feedback (auto-generated)
guest_id	Reference to guest providing feedback
reservation_id	Reference to reservation being rated
rating	Numerical rating for the stay
note	Additional comments from guest
feedback_date	Date when feedback was submitted

Attributes: ServicePriceHistory

Attribute	Description
sph_id	Unique identifier for service price history (auto-generated)
service_id	Reference to service
price	Price of service during this period
valid_from	Start date of price validity
valid_to	End date of price validity

Attributes: RoomTypePriceHistory

Attribute	Description
rtph_id	Unique identifier for room type price history (auto-generated)
room_type_id	Reference to room type
price_per_night	Price per night during this period
valid_from	Start date of price validity
valid_to	End date of price validity

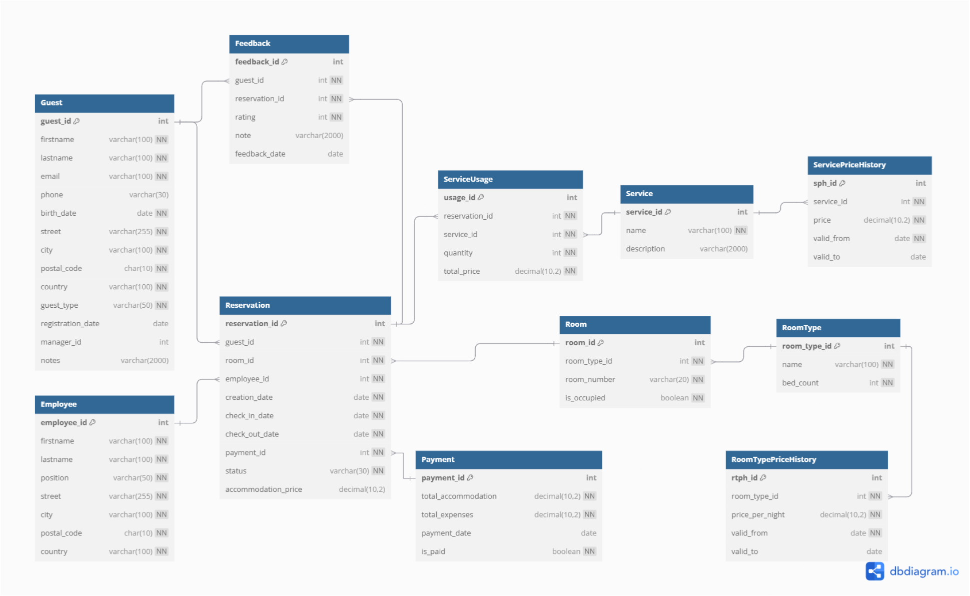
3 DD S03 L01

`Task:` Describe all relations in your database in English, including cardinality and membership obligation - each relation in two sentences (page 10)

- **Guest – Reservation:** Each guest can have zero or more reservations (1:N). Every reservation must be linked to exactly one guest (mandatory).
- **Room – Reservation:** Each room can be associated with zero or more reservations over time (1:N). Every reservation must be assigned to exactly one room (mandatory).
- **Employee – Reservation:** Each employee can create or manage zero or more reservations (1:N). Every reservation must be linked to exactly one employee (mandatory).
- **Payment – Reservation:** Each payment can be linked to one or more reservations (1:N). Every reservation must have exactly one payment (mandatory).
- **RoomType – Room:** Each room type can be assigned to zero or more rooms (1:N). Every room must have exactly one room type (mandatory).
- **Service – ServiceUsage:** Each service can be used in zero or more service usages (1:N). Every service usage must refer to exactly one service (mandatory).
- **Reservation – ServiceUsage:** Each reservation can have zero or more service usages (1:N). Every service usage must be linked to exactly one reservation (mandatory).
- **Guest – Feedback:** Each guest can provide zero or more feedback entries (1:N). Every feedback must be linked to exactly one guest (mandatory).
- **Reservation – Feedback:** Each reservation can have zero or more feedback entries (1:N). Every feedback must be linked to exactly one reservation (mandatory).
- **Service – ServicePriceHistory:** Each service can have zero or more price history records (1:N). Every price history record must be linked to exactly one service (mandatory).
- **RoomType – RoomTypePriceHistory:** Each room type can have zero or more price history records (1:N). Every price history record must be linked to exactly one room type (mandatory).

4 DD S03 L02

`Task:` Draw an ER diagram according to conventions



Guest		
guest_id	int	
firstname	varchar(100)	NN
lastname	varchar(100)	NN
email	varchar(100)	NN
phone	varchar(30)	
birth_date	date	NN
street	varchar(255)	NN
city	varchar(100)	NN
postal_code	char(10)	NN
country	varchar(100)	NN
guest_type	varchar(50)	NN
registration_date	date	
manager_id	int	
notes	varchar(2000)	

Employee		
employee_id	int	
firstname	varchar(100)	NN
lastname	varchar(100)	NN
position	varchar(50)	NN
street	varchar(255)	NN
city	varchar(100)	NN
postal_code	char(10)	NN
country	varchar(100)	NN

Feedback		
feedback_id	int	
guest_id	int	NN
reservation_id	int	NN
rating	int	NN
note	varchar(2000)	
feedback_date	date	

Reservation		
reservation_id	int	
guest_id	int	NN
room_id	int	NN
employee_id	int	NN
creation_date	date	NN
check_in_date	date	NN
check_out_date	date	NN
payment_id	int	NN
status	varchar(30)	NN
accommodation_price	decimal(10,2)	

ServiceUsage		
usage_id	int	
reservation_id	int	NN
service_id	int	NN
quantity	int	NN
total_price	decimal(10,2)	NN

Service		
service_id	int	
name	varchar(100)	NN
description	varchar(2000)	

ServicePriceHistory		
sph_id	int	
service_id	int	NN
price	decimal(10,2)	NN
valid_from	date	NN
valid_to	date	

Room		
room_id	int	
room_type_id	int	NN
room_number	varchar(20)	NN
is_occupied	boolean	NN

RoomType		
room_type_id	int	
name	varchar(100)	NN
bed_count	int	NN

Payment		
payment_id	int	
total_accommodation	decimal(10,2)	NN
total_expenses	decimal(10,2)	NN
payment_date	date	
is_paid	boolean	NN

RoomTypePriceHistory		
rtph_id	int	
room_type_id	int	NN
price_per_night	decimal(10,2)	NN
valid_from	date	NN
valid_to	date	

Figure 3: ER Diagram for Hotel Database

5 DD S30 L04

`Task:` Matrix diagram with relationships, draw for your solution

	Guest	**Employee**	**Room**	**RoomType**	**Reservation**	**Payment**	**Service**	**ServiceUsage**	**Feedback**	**ServicePriceHistory**
Guest					1:N				1:N	
Employee					1:N					
Room				1:N	1:N					
RoomType			N:1							1:N
Reservation	N:1	N:1	N:1			1:1		1:N	1:N	
Payment					1:1					
Service								1:N		1:N
ServiceUsage					N:1		N:1			
Feedback	N:1				N:1					
ServicePriceHistory				N:1			N:1			

6 DD S04 L01

`Task:` Supertypes and subtypes – define at least one instance of a supertype and a subtype in your project

In the hotel project, an example of a supertype and subtype can be found in the `Guest` entity. The `Guest` table has an attribute `guest_type`, which can distinguish between different subtypes such as "Regular" and "VIP".

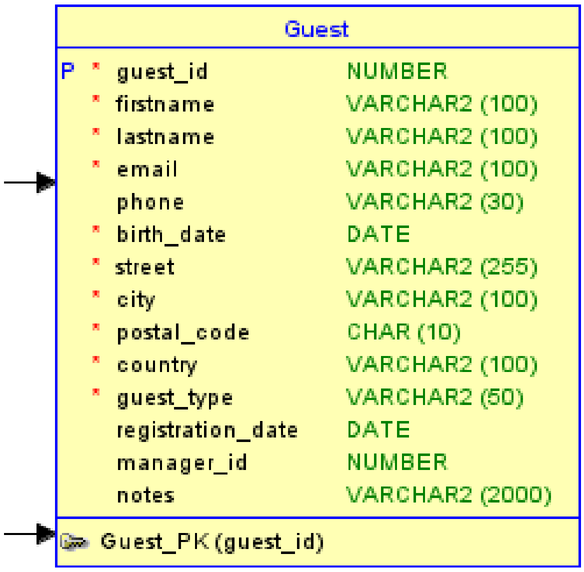
Supertype: `Guest` (stores general information about all guests, regardless of type).
Subtypes:

- `Regular Guest` (a standard guest, e.g., `guest_type = "Regular"`)
- `VIP Guest` (a guest with VIP status, e.g., `guest_type = "VIP"`)

Example instance:

`guest_id: 10, firstname: "Alice", lastname: "Brown", email: "alice.brown@email.com", guest_type: "VIP"`

This record is an instance of the supertype `Guest` and the subtype `VIP Guest`.



Guest			
P	*	guest_id	NUMBER
	*	firstname	VARCHAR2 (100)
	*	lastname	VARCHAR2 (100)
	*	email	VARCHAR2 (100)
		phone	VARCHAR2 (30)
	*	birth_date	DATE
	*	street	VARCHAR2 (255)
	*	city	VARCHAR2 (100)
	*	postal_code	CHAR (10)
	*	country	VARCHAR2 (100)
	*	guest_type	VARCHAR2 (50)
		registration_date	DATE
		manager_id	NUMBER
		notes	VARCHAR2 (2000)

Guest_PK (guest_id)

Figure 4: Guest table

7 DD S04 L02

`Task:` Description of business rules for your project

- Each guest must provide a unique email address and basic personal information to register.
- A reservation can only be created if the selected room is available for the entire requested period.
- Every reservation must be linked to exactly one guest, one room, one employee, and one payment.
- Each payment must cover the total accommodation and any additional expenses, and can be marked as paid or unpaid.
- Services can be used only by guests with an active reservation, and each service usage must be linked to a reservation.
- Feedback can only be submitted by guests who have completed a reservation.
- Room prices and service prices can change over time, but each price change must be recorded in the corresponding price history table.
- Each room must belong to exactly one room type, and each room type can have multiple rooms.
- Only one guest type (Individual or Company) can be assigned to each guest.
- Employees are responsible for managing reservations and must be assigned to each reservation.

Table 13: Guest Table Attributes

Name	**Data Type**	**Length**	**Key**	**Null**	**Index**	**IO**	**Meaning**
guest_id	int	-	P	N	A	-	Guest ID
firstname	varchar	100	-	N	-	-	Guest first name
lastname	varchar	100	-	N	-	-	Guest last name
email	varchar	100	-	N	-	-	Guest email
phone	varchar	30	-	A	-	-	Guest phone number
birth_date	date	-	-	N	-	-	Guest birth date
street	varchar	255	-	N	-	-	Street
city	varchar	100	-	N	-	-	City
postal_code	char	10	-	N	-	-	Postal code
country	varchar	100	-	N	-	-	Country
guest_type	varchar	50	-	N	-	-	Guest type
registration_date	date	-	-	A	-	-	Registration date
notes	clob	-	-	A	-	-	Notes about guest

Table 14: Employee Table Attributes

Name	**Data Type**	**Length**	**Key**	**Null**	**Index**	**IO**	**Meaning**
employee_id	int	-	P	N	A	-	Employee ID
firstname	varchar	100	-	N	-	-	Employee first name
lastname	varchar	100	-	N	-	-	Employee last name
position	varchar	50	-	N	-	-	Employee position
street	varchar	255	-	N	-	-	Street
city	varchar	100	-	N	-	-	City
postal_code	char	10	-	N	-	-	Postal code
country	varchar	100	-	N	-	-	Country

Table 15: RoomType Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
room_type_id	int	-	P	N	A	-	Room type ID
name	varchar	100	-	N	-	-	Room type name
bed_count	int	-	-	N	-	-	Bed count

Table 16: Room Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
room_id	int	-	P	N	A	-	Room ID
room_type_id	int	-	F(RoomType)	N	-	-	Room type ID
room_number	varchar	20	-	N	U	-	Room number
is_occupied	boolean	-	-	N	-	-	Room occupancy

Table 17: Payment Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
payment_id	int	-	P	N	A	-	Payment ID
total_accommodation	decimal	-	-	N	-	-	Total accommodation cost
total_expenses	decimal	-	-	N	-	-	Total expenses
payment_date	date	-	-	A	-	-	Payment date
is_paid	boolean	-	-	N	-	-	Payment status

Table 18: Reservation Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
reservation_id	int	-	P	N	A	-	Reservation ID
guest_id	int	-	F(Guest)	N	-	-	Guest ID
room_id	int	-	F(Room)	N	-	-	Room ID
employee_id	int	-	F(Employee)	N	-	-	Employee ID
creation_date	date	-	-	N	-	-	Creation date
check_in_date	date	-	-	N	-	-	Check-in date
check_out_date	date	-	-	N	-	-	Check-out date
payment_id	int	-	F(Payment)	N	-	-	Payment ID
status	varchar	30	-	N	-	-	Reservation status

Table 19: Service Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
service_id	int	-	P	N	A	-	Service ID
name	varchar	100	-	N	-	-	Service name
description	clob	-	-	A	-	-	Service description

Table 20: ServiceUsage Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
usage_id	int	-	P	N	A	-	Service usage ID
reservation_id	int	-	F(Reservation)	N	-	-	Reservation ID
service_id	int	-	F(Service)	N	-	-	Service ID
quantity	int	-	-	N	-	-	Quantity
total_price	decimal	-	-	N	-	-	Total price

Table 21: Feedback Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
feedback_id	int	-	P	N	A	-	Feedback ID
guest_id	int	-	F(Guest)	N	-	-	Guest ID
reservation_id	int	-	F(Reservation)	N	-	-	Reservation ID
rating	int	-	-	N	-	-	Rating (1-5)
comment	clob	-	-	A	-	-	Comment
feedback_date	date	-	-	A	-	-	Feedback date

Table 22: ServicePriceHistory Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
sph_id	int	-	P	N	A	-	Service price history ID
service_id	int	-	F(Service)	N	-	-	Service ID
price	decimal	-	-	N	-	-	Service price
valid_from	date	-	-	N	-	-	Valid from
valid_to	date	-	-	A	-	-	Valid to

Table 23: RoomTypePriceHistory Table Attributes

Name	Data Type	Length	Key	Null	Index	IO	Meaning
rtph_id	int	-	P	N	A	-	Room type price history ID
room_type_id	int	-	F(RoomType)	N	-	-	Room type ID
price_per_night	decimal	-	-	N	-	-	Price per night
valid_from	date	-	-	N	-	-	Valid from
valid_to	date	-	-	A	-	-	Valid to

8 DD S05 L01

`Task:` Include at least one portable and one non-portable binding in your project

In the project, the following types of bindings are present:

Table 24: Portable and Non-portable Bindings in the Project

Table	**Portable Binding (Example)**	**Non-portable Binding (Example)**
Guest	Foreign key: `manager_id` referencing `Employee` (supported in all major RDBMS)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific auto-increment syntax)
Employee	Primary key: `employee_id` as PK (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
RoomType	Primary key: `room_type_id` as PK (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
Room	Foreign key: `room_type_id` referencing `RoomType` (portable)	`uq_room_number UNIQUE (room_number)` (constraint name syntax is not portable)
Payment	Primary key: `payment_id` as PK (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
Reservation	Foreign keys: `guest_id`, `room_id`, `employee_id`, `payment_id` (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
Service	Primary key: `service_id` as PK (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
ServiceUsage	Foreign keys: `reservation_id`, `service_id` (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
Feedback	Foreign keys: `guest_id`, `reservation_id` (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
ServicePriceHistory	Foreign key: `service_id` referencing `Service` (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)
RoomTypePriceHistory	Foreign key: `room_type_id` referencing `RoomType` (portable)	`NUMBER GENERATED ALWAYS AS IDENTITY` (Oracle-specific)

9 DD S05 L03

`Task:` Have at least one M:N relationship without information and one M:N relationship with information in your project

Table 25: Examples of M:N Relationships in the Project

Relationship	**Join Table**	**With Information**	**Description / Additional Attributes**
Guest – Room	GuestRoom	No	Only tracks which guests have stayed in which rooms; join table contains only `guest_id`, `room_id`
Reservation – Service	ServiceUsage	Yes	Tracks which services were used in which reservations; join table contains `reservation_id`, `service_id`, plus additional attributes like `quantity`, `total_price`

10 DD S06 L01

`Task:` Incorporate at least one 1:N identifying relationship into your project, with the fact that the transferred foreign key will also be the key in the new table

Table 26: Example of 1:N Identifying Relationship

Parent Table	**Child Table**	**Identifying Foreign Key**	**Primary Key in Child**
RoomType	RoomTypePriceHistory	room_type_id	(1) `rtph_id` (simple PK) (2) `room_type_id`, `valid_from` (composite PK, foreign key is part of PK)

In the current design, each `RoomTypePriceHistory` record has its own primary key (`rtph_id`) and a mandatory foreign key (`room_type_id`) referencing `RoomType`. In an alternative identifying relationship, the primary key of `RoomTypePriceHistory` could be a composite key (`room_type_id`, `valid_from`), making the foreign key also part of the primary key, which is typical for identifying relationships.

11 DD S06 L02-04

`Task:` Demonstrate that your schema is in first, second, and third normal form

Table 27: Normalization Forms and Tables in the Project

Normal Form	**Tables in the Project**
First Normal Form (1NF)	All tables: `Guest`, `Employee`, `RoomType`, `Room`, `Payment`, `Reservation`, `Service`, `ServiceUsage`, `Feedback`, `ServicePriceHistory`, `RoomTypePriceHistory`. *All attributes are atomic, there are no repeating groups or arrays.*
Second Normal Form (2NF)	All tables: `Guest`, `Employee`, `RoomType`, `Room`, `Payment`, `Reservation`, `Service`, `ServiceUsage`, `Feedback`, `ServicePriceHistory`, `RoomTypePriceHistory`. *All non-key attributes are fully functionally dependent on the whole primary key. No partial dependencies exist.*
Third Normal Form (3NF)	All tables: `Guest`, `Employee`, `RoomType`, `Room`, `Payment`, `Reservation`, `Service`, `ServiceUsage`, `Feedback`, `ServicePriceHistory`, `RoomTypePriceHistory`. *No transitive dependencies between non-key attributes. All non-key attributes depend only on the primary key.*

All tables in the project schema meet the requirements for 1NF, 2NF, and 3NF.

12 DD S07 L01

`Task:` Try to define ARC in your project (can be defined in ORACLE SQL Developer Data Modeler)

An example of an ARC (Alternative Relationship Constraint) in the hotel project could be in the `Payment` entity. For instance, a payment could be associated either with a reservation or with a service usage, but not both at the same time. This can be modeled by having two nullable foreign keys in the `Payment` table (`reservation_id` and `usage_id`) and enforcing that exactly one of them is not null for each payment.

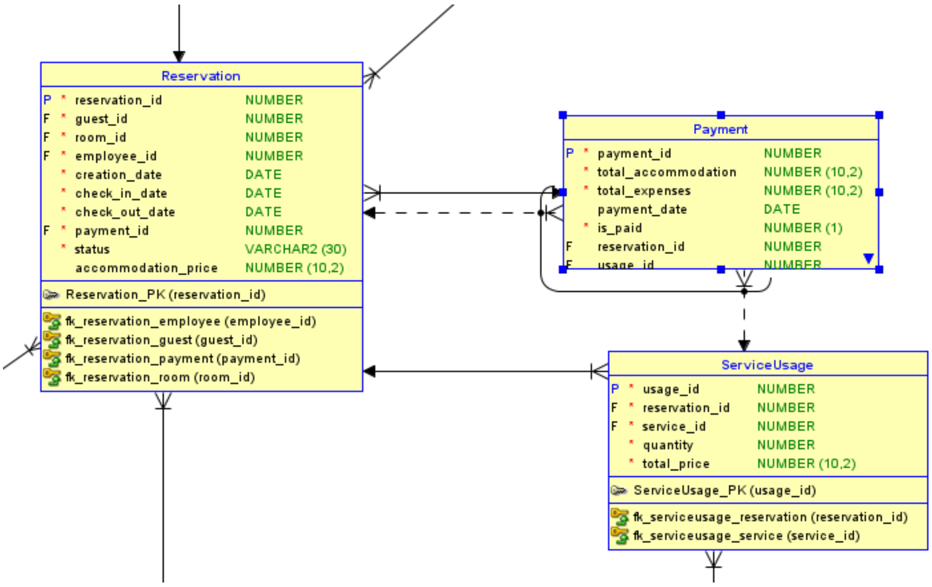

Reservation
P * reservation_id NUMBER
F * guest_id NUMBER
F * room_id NUMBER
F * employee_id NUMBER
* creation_date DATE
* check_in_date DATE
* check_out_date DATE
F * payment_id NUMBER
* status VARCHAR2 (30)
accommodation_price NUMBER (10,2)
Reservation_PK (reservation_id)
fk_reservation_employee (employee_id)
fk_reservation_guest (guest_id)
fk_reservation_payment (payment_id)
fk_reservation_room (room_id)
Payment
P * payment_id NUMBER
* total_accommodation NUMBER (10,2)
* total_expenses NUMBER (10,2)
payment_date DATE
* is_paid NUMBER (1)
F reservation_id NUMBER
F usage_id NUMBER
ServiceUsage
P * usage_id NUMBER
F * reservation_id NUMBER
F * service_id NUMBER
* quantity NUMBER
* total_price NUMBER (10,2)
ServiceUsage_PK (usage_id)
fk_serviceusage_reservation (reservation_id)
fk_serviceusage_service (service_id)

Figure 5: Alternative Relationship Constraint

13 DD S07 L02

`Task:` Try to define hierarchical and recursive relations in your project

A hierarchical or recursive relationship can be represented in the `Employee` table, where an employee can be a manager of other employees. This is modeled by adding a nullable foreign key `manager_id` referencing `employee_id` in the same table, allowing you to build an organizational hierarchy.

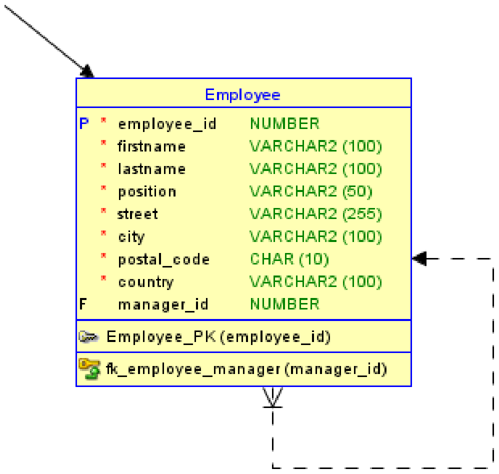


Figure 6: Rekurzivv

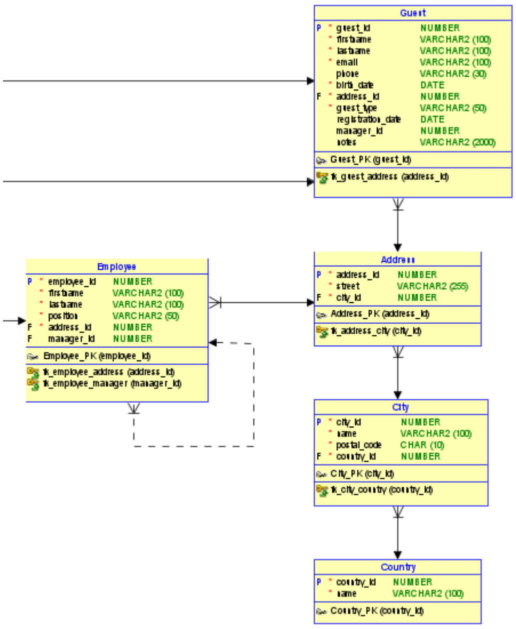


Figure 7: Pokoj

14 DD S07 L03

`Task:` Describe how you record historical data in your system

Historical data in the system is recorded using dedicated history tables, such as `RoomTypePriceHistory` and `ServicePriceHistory`. These tables store records of price changes over time, including the start and end dates for each price, allowing the system to track and retrieve historical pricing information for rooms and services.

15 DD S09 L02

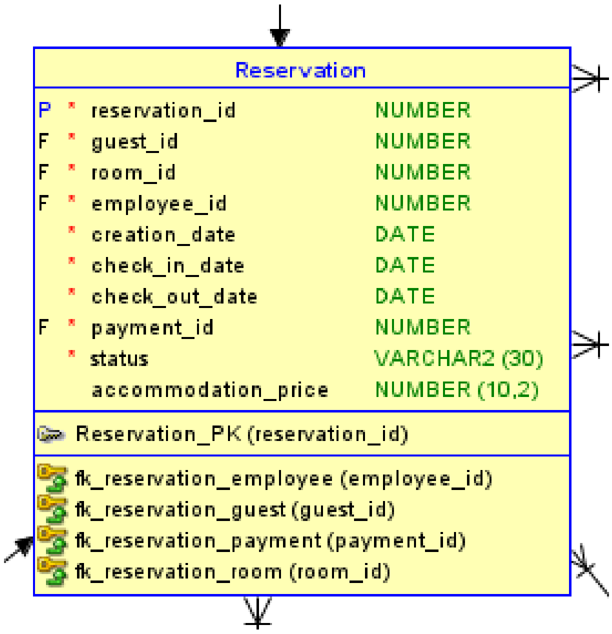
`Task:` Try journaling in your project, i.e. saving past historical data (for example salary changes, workplace changes, etc.)

Journaling in the hotel project is implemented by using dedicated history tables that store changes over time. For example, the `RoomTypePriceHistory` and `ServicePriceHistory` tables record every change in room type prices and service prices, including the period of validity for each price. This allows the system to keep a complete record of all past prices and retrieve historical data as needed.

16 DD S10 L01

`Task:` Revise your design according to conventions for the readability of your schema

Main ERD – Reservation-Centric View



Reservation

P	*	reservation_id	NUMBER
F	*	guest_id	NUMBER
F	*	room_id	NUMBER
F	*	employee_id	NUMBER
	*	creation_date	DATE
	*	check_in_date	DATE
	*	check_out_date	DATE
F	*	payment_id	NUMBER
	*	status	VARCHAR2 (30)
		accommodation_price	NUMBER (10,2)

Reservation_PK (reservation_id)

fk_reservation_employee (employee_id)
fk_reservation_guest (guest_id)
fk_reservation_payment (payment_id)
fk_reservation_room (room_id)

Figure 8: Main ER Diagram centered around Reservation

Diagram Description

This diagram describes the relationships and structure of the hotel reservation system.

- **Guest** stores customer personal data.
- **Employee** represents staff, including those managing reservations.
- **Reservation** links a guest, room, employee, and payment.
- **Room** is typed using `RoomType`, which in turn has a price history.
- **Service** and **ServiceUsage** are linked to a reservation.
- **Payment** connects to reservations.
- **Feedback** captures guest opinions tied to reservations.
- Tables like `ServicePriceHistory` and `RoomTypePriceHistory` track historical pricing.

17 DD S10 L02

`Task:` Generic modeling – consider, possibly describe or use a generic model of data structures in your solution, how this approach is more advantageous compared to traditional data structure design methods

todo

18 DD S11 L01

`Task:` Describe examples of integrity constraints on your project for entities, bindings, attributes, and user-defined integrity

Table 28: Examples of Integrity Constraints in the Project

Constraint Type	**Example**	**Description**
Entity Integrity	Primary Key (`guest_id`, `reservation_id`, etc.)	Ensures each record is uniquely identified and primary keys cannot be NULL or duplicated.
Entity Integrity	IDENTITY Property	`GENERATED ALWAYS AS IDENTITY` automatically generates unique sequential values for primary keys.
Referential Integrity	Foreign Key (`fk_reservation_guest`, `fk_reservation_room`, `fk_serviceusage_reservation`)	Ensures referenced records exist and prevents orphaned records in related tables.
Domain Integrity	NOT NULL (`firstname`, `lastname`, `email`)	Ensures essential attributes must always have a value.
Domain Integrity	Data Type (`VARCHAR2(100)`, `NUMBER(10,2)`, `DATE`)	Restricts attribute values to specific types and formats.
Domain Integrity	DEFAULT Value (`registration_date`, `creation_date`)	Automatically assigns a value (e.g., `SYSDATE`) if none is provided.
User-Defined Integrity	UNIQUE (`uq_room_number`)	Ensures each room has a unique room number.
User-Defined Integrity	CHECK (e.g., `rating` 1–5, `is_occupied` 0/1, `is_paid` 0/1)	Enforces business rules and valid value ranges for attributes.
User-Defined Integrity	Custom Business Rules	E.g., check-out date must be after check-in date (enforced by triggers or application logic).

These integrity constraints work together to ensure data consistency, prevent invalid data entry, and maintain the overall reliability of the hotel management database system.

19 DD S11 L02-04

`Task:` Generate a relational schema from your conceptual model and note the changes that have occurred in the schema and why

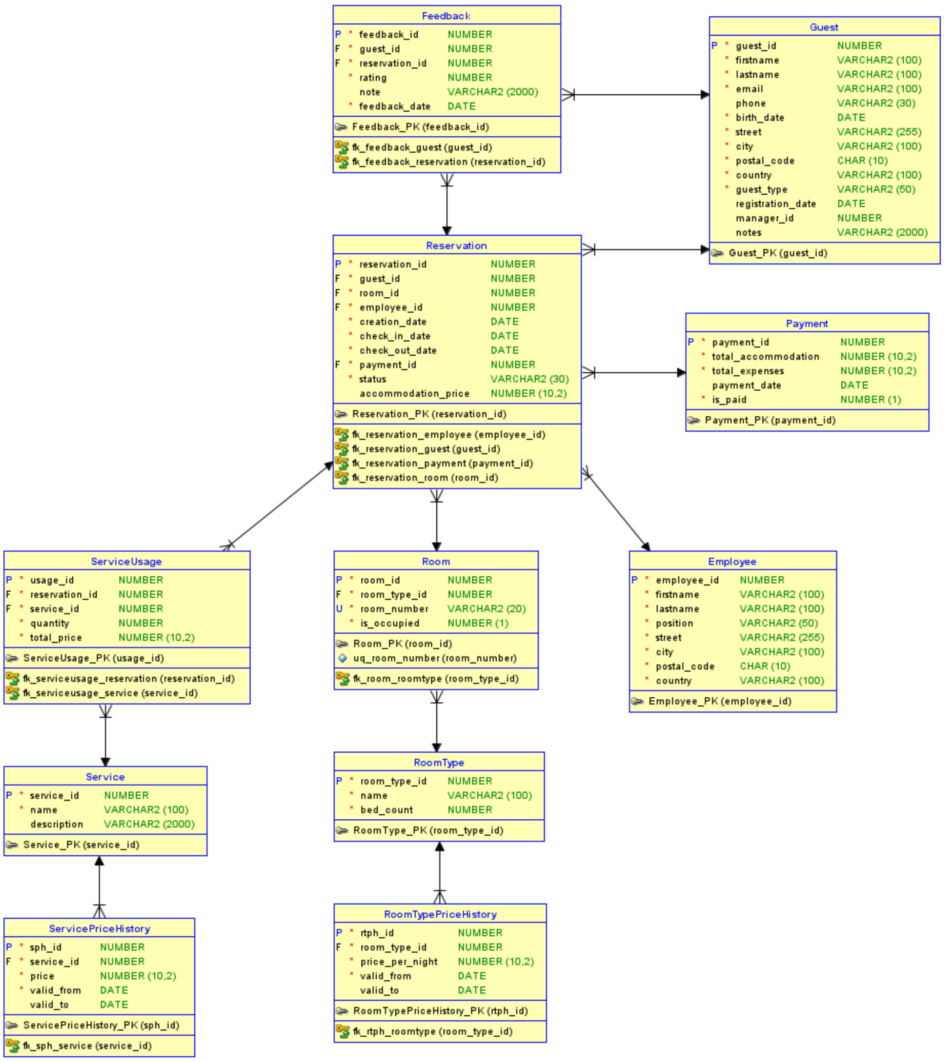


Figure 9: ER Diagram for Hotel Database