

Presentation for the development phase.
Project: Build a data mart in SQL

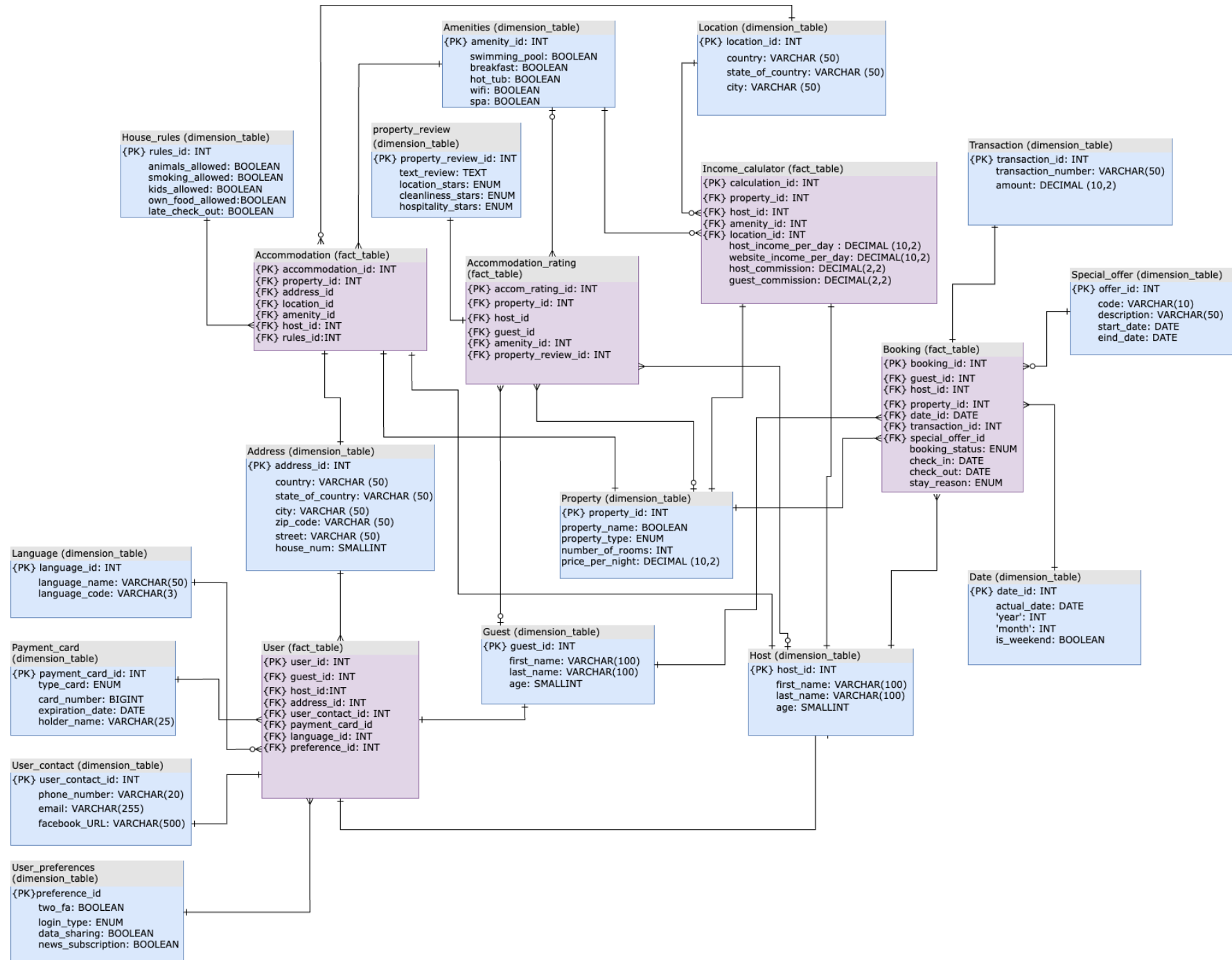
Author's name: Valeriia Lvova

Matriculation number: 92128860

Tutor's name: Pr. Musharaf Doger

Date: 17.02.2024

Revised ERM diagram.
Dimensional modeling.
Fact table is described by
dimension tables.



First step: Create the 'datamart_Lvova' database if it doesn't already exist. This query should be run separately from the rest of the code. The remaining code can be executed in one step.

```
CREATE DATABASE datamart_Lvova;
```

Messages Notifications Data Output

CREATE DATABASE

Query returned successfully in 101 msec.

The code snippet below shows how the guest table is created.

```
-- 6 Guest table
CREATE TABLE guest (
    guest_id INT PRIMARY KEY,
    first_name VARCHAR(100),
    last_name VARCHAR(100),
    age SMALLINT
);
```

The code snippet below shows how to insert data into the guest table. The same principal is applied for other tables. Each table has 20 records.

```
--| Values for guest table
INSERT INTO guest (guest_id, first_name, last_name, age) VALUES
(1, 'Daan', 'Jansen', 28),
(2, 'Emma', 'De Vries', 26),
(3, 'Levi', 'Bakker', 32),
(4, 'Sophie', 'Van Dijk', 30);
```

Additionally, fact tables have been created. This fact table contains foreign keys that correspond to the primary keys of dimensional tables. These dimensional tables provide detailed descriptions and context for the data in the fact tables.

It is necessary first to create the referenced tables, including 'guest', 'host', 'address', 'user_contact', 'payment_card', 'language', and 'user_preference'. These tables provide the foundational data, linking all the information together comprehensively.

```
-- User fact table
CREATE TABLE user_fact(
  user_id INT PRIMARY KEY,
  guest_id INT,
  host_id INT,
  address_id INT,
  user_contact_id INT,
  payment_card_id INT,
  language_id INT,
  preference_id INT,
  FOREIGN KEY (guest_id) REFERENCES guest(guest_id),
  FOREIGN KEY (host_id) REFERENCES host(host_id),
  FOREIGN KEY (address_id) REFERENCES address(address_id),
  FOREIGN KEY (user_contact_id) REFERENCES user_contact(user_contact_id),
  FOREIGN KEY (payment_card_id) REFERENCES payment_card(payment_card_id),
  FOREIGN KEY (language_id) REFERENCES language(language_id),
  FOREIGN KEY (preference_id) REFERENCES user_preferences(preference_id)
);
```

The implementation design assigns a single ID to each user, which is utilized both for hosting and guest roles. To accommodate dimensional modeling principles, these roles have been distinctly partitioned into two separate tables.

```
select *
from guest
natural join host;
```

Messages Notifications Data Output					
	first_name character varying (100) 🔒	last_name character varying (100) 🔒	age smallint 🔒	guest_id integer 🔒	host_id integer 🔒
1	Daan	Jansen	28	1	1
2	Emma	De Vries	26	2	2
3	Levi	Bakker	32	3	3
4	Sophie	Van Dijk	30	4	4
5	Bram	Visser	35	5	5
6	Tess	Smit	29	6	6
7	Sem	Meijer	31	7	7
8	Anna	De Boer	27	8	8
9	Finn	Mulder	33	9	9
10	Eva	Bos	25	10	10
11	Gabriel	García	34	11	11
12	Lucas	Rodriguez	29	12	12
13	Sofia	Silva	28	13	13
14	Mateo	Martinez	35	14	14
15	Camila	Lopez	26	15	15
16	Oliver	Smith	30	16	16
17	Charlotte	Jones	31	17	17
18	Harry	Taylor	32	18	18
19	Amelia	Brown	27	19	19
20	Jack	Wilson	29	20	20









The output of this query provides a clear indication of the usage frequency of each language among the users, which is crucial for understanding user preferences and potentially guiding language-related decisions or features within the application or service. It provides us as well that not each language is met in user_fact table that proves One-to-Zero-or-Many relationship.

```
SELECT
    l.language_name,
    COUNT(u.language_id) AS language_count
FROM
    user_fact u
LEFT JOIN
    language l ON u.language_id = l.language_id
GROUP BY
    l.language_name;
```

Messages Notifications <u>Data Output</u>		
<div><div><div>≡+</div><div></div><div>▼</div><div></div><div>▼</div><div></div></div><div><div></div><div></div><div></div></div></div>		
	language_name character varying (50)	language_count bigint
1	German	1
2	Portuguese	1
3	Dutch	10
4	Spanish	6
5	French	1
6	Italian	1










To determine which country has the most users, we can use this specific query. The query counts the number of users from each country and then identifies the country with the highest count.

```
SELECT country, COUNT(*) as the_most_popular_country
FROM address
GROUP BY country
ORDER BY the_most_popular_country DESC
LIMIT 1;
```

Messages Notifications Data Output		
<div><div>≡+</div><div>▼</div><div>▼</div><div></div><div></div><div></div><div></div></div>		
	country character varying (50) 	the_most_popular_country bigint 
1	Netherlands	10

Given the significant popularity of the Netherlands on our website, we have the opportunity to ascertain whether users maintain local contacts by examining phone numbers prefixed with the Dutch country code "+31".

```
-- +31 is the code of the Netherlands country
SELECT *
FROM user_contact
WHERE phone_number LIKE '+31%';
```

Messages Notifications Data Output				
        				
	user_contact_id [PK] integer	phone_number character varying (20)	email character varying (255)	facebook_url character varying (500)
1	1	+31201234567	daan.jansen@email.com	https://facebook.com/daanjansen
2	2	+31201234568	emma.devries@email.com	https://facebook.com/emmadevries
3	3	+31201234569	levi.bakker@email.com	https://facebook.com/levibakker
4	4	+31201234570	sophie.vandijk@email.com	https://facebook.com/sophievandijk
5	5	+31201234571	bram.visser@email.com	https://facebook.com/bramvisser
6	6	+31201234572	tess.smit@email.com	https://facebook.com/tesssmit
7	7	+31201234573	sem.meijer@email.com	https://facebook.com/semmeijer
8	8	+31201234574	anna.deboer@email.com	https://facebook.com/annadeboer
9	9	+31201234575	finn.mulder@email.com	https://facebook.com/finnmulder
10	10	+31201234576	eva.bos@email.com	https://facebook.com/evabos

```
select first_name, last_name, age
from guest
where age > 30;
```

This query shows all guest who older than 30 years.

Messages Notifications <u>Data Output</u>			
<div><div><div>≡+</div><div><div></div><div>▼</div></div><div><div></div><div>▼</div></div><div><div></div></div></div><div><div></div><div><div></div><div></div></div></div></div>			
	first_name character varying (100)	last_name character varying (100)	age smallint
1	Levi	Bakker	32
2	Bram	Visser	35
3	Sem	Meijer	31
4	Finn	Mulder	33
5	Gabriel	García	34
6	Mateo	Martinez	35
7	Charlotte	Jones	31
8	Harry	Taylor	32

In this analysis, we assess whether all users possess valid payment cards. The results indicate that 13 users are equipped with current payment cards, whereas 7 users possess cards that have expired. To address this, it's necessary to either reach out to these users for card updates or implement restrictions within our software to ensure transactions are only processed with valid cards.

```
SELECT
    COUNT(*) AS total_users,
    SUM(CASE WHEN expiration_date > CURRENT_DATE THEN 1 ELSE 0 END) AS users_with_active_cards
FROM
    user_fact uf
JOIN
    payment_card pc ON uf.payment_card_id = pc.payment_card_id;
```

Messages Notifications Data Output		
	total_users bigint	users_with_active_cards bigint
1	20	13

Based on my ERM (Entity-Relationship Model) diagram, a single set of house rules is met multiple times in the accommodation fact table, indicating a 1:M (one-to-many) relationship.

```
SELECT ac.property_id, hr.rules_id
FROM house_rules hr
INNER JOIN accommodation ac
ON hr.rules_id = ac.rules_id
WHERE hr.smoking_allowed IS TRUE AND hr.animals_allowed IS TRUE AND
hr.kids_allowed IS TRUE AND hr.own_food_allowed IS TRUE AND hr.late_check_out IS TRUE;
```

Messages Notifications Data Output



	property_id integer	rules_id integer
1	4	1
2	17	1

This query proves 1:M relationship. The output has 1 the same amenity that belongs to 2 different properties.

```
SELECT a.amenity_id, ac.property_id
from amenities a
INNER JOIN accommodation ac
on a.amenity_id = ac.amenity_id
where a.swimming_pool is true and a.breakfast is false and
a.wi-fi is false and a.hot_tub is true and a.spa is false ;
```

Messages Notifications <u>Data Output</u>		
<div><div><div>☰+</div><div>📄</div><div>▼</div><div>📋</div><div>▼</div><div>🗑</div><div>🗄</div><div>⬇</div><div>📈</div></div></div>		
	amenity_id integer 🔒	property_id integer 🔒
1	6	9
2	6	10

This code snippet identifies accounts with potentially low protection settings, specifically those without two-factor authentication enabled. Based on the output data, an email should be sent to these users to inform them and prevent potential data leaks.

```
CREATE TYPE login_type AS ENUM ('google', 'email', 'phone_number');
-- 2 User preferences table
CREATE TABLE user_preferences (
    preference_id INT PRIMARY KEY,
    two_fa BOOLEAN,
    login_type login_type NOT NULL,
    data_sharing BOOLEAN,
    news_subscription BOOLEAN
);
```

```
SELECT pr.preference_id, uf.user_id
FROM user_preferences pr
LEFT JOIN user_fact uf
ON pr.preference_id = uf.preference_id
WHERE pr.two_fa IS FALSE;
```

Messages Notifications Data Output



	preference_id integer	user_id integer
1	2	2
2	4	4
3	6	6
4	8	8
5	10	10
6	12	12
7	14	14
8	16	16
9	18	18
10	20	20









By executing the specified query, we can identify the property with the highest rental price in our database. This information allows us to then calculate the total earnings generated from this particular property through our website.

```
SELECT *
FROM property
WHERE price_per_night = (
    SELECT MAX(price_per_night)
    FROM property
);
```

Messages Notifications Data Output					
<div><div><div>≡+</div><div></div><div>▼</div><div></div><div>▼</div><div></div><div></div><div></div><div></div></div></div>					
	property_id [PK] integer	property_name character varying (100)	property_type property_type	number_of_rooms integer	price_per_night numeric (10,2)
1	11	Luxury Villa	Villa	4	300.00
2	20	Royal Villa	Villa	6	300.00

This query efficiently calculates the total revenue generated from confirmed bookings for each specified property by multiplying the duration of each stay (in days) by the property's nightly rate.

```
SELECT
    b.property_id,
    SUM((b.check_out_date - b.check_in_date) * p.price_per_night) AS total_earnings
FROM
    booking b
JOIN
    property p ON b.property_id = p.property_id
WHERE
    b.property_id IN (11, 20) AND
    b.booking_status = 'Confirmed'
GROUP BY
    b.property_id;
```

Messages Notifications Data Output		
<div><div>≡+</div><div> </div><div> </div><div></div><div></div><div></div><div></div></div>		
	property_id integer	total_earnings numeric
1	20	2700.00

This query efficiently retrieves all confirmed and paid bookings, incorporating a 5% discount on the total paid amounts.

```
SELECT b.property_id, b.transaction_id, b.booking_status, t.amount, s.offer_id, s.description
FROM booking b
INNER join special_offer s
ON b.offer_id = s.offer_id
INNER join transaction t
ON b.transaction_id = t.transaction_id
WHERE b.booking_status = 'Confirmed'
```

Messages Notifications Data Output

	property_id integer	transaction_id integer	booking_status booking_status	amount numeric (10,2)	offer_id integer	description character varying (100)
1	2	1	Confirmed	1187.50	1	Spring Discount 2023 (5% off)
2	4	2	Confirmed	712.50	2	Summer Promotion 2023 (5% off)
3	10	5	Confirmed	1064.00	5	Early Bird Booking 2024 (5% off)
4	12	6	Confirmed	266.00	6	Valentine's Day Special 2024 (5% off)
5	14	7	Confirmed	1396.50	1	Spring Discount 2023 (5% off)
6	18	9	Confirmed	403.75	3	Fall Getaway 2023 (5% off)
7	20	10	Confirmed	2565.00	4	Winter Retreat 2023 (5% off)
8	5	13	Confirmed	1330.00	1	Spring Discount 2023 (5% off)
9	7	14	Confirmed	684.00	2	Summer Promotion 2023 (5% off)
10	9	15	Confirmed	598.50	3	Fall Getaway 2023 (5% off)
11	13	17	Confirmed	1187.50	5	Early Bird Booking 2024 (5% off)
12	15	18	Confirmed	555.75	6	Valentine's Day Special 2024 (5% off)
13	17	19	Confirmed	1092.50	1	Spring Discount 2023 (5% off)
14	19	20	Confirmed	902.50	2	Summer Promotion 2023 (5% off)

This written query effectively retrieves and lists all confirmed bookings, showing their check-in dates and associated date IDs, ordered by the check-in date.

```
select d.date_id, b.check_in_date
from date d
left join booking b
on d.date_id=b.date_id
where b.booking_status = 'Confirmed'
order by b.check_in_date;
```

Messages Notifications Data Output		
	date_id integer	check_in_date date
1	1	2023-03-15
2	2	2023-06-10
3	5	2024-02-14
4	6	2024-04-01
5	7	2024-06-18
6	9	2025-03-10
7	10	2025-06-01
8	13	2026-02-14
9	14	2026-04-01
10	15	2026-06-18
11	17	2027-03-10
12	18	2027-06-01
13	19	2027-08-15
14	20	2027-09-05

This query efficiently identifies the most frequently utilized promotional offers, revealing their popularity among bookings. By further analyzing the distribution channels of these offer codes to users, we can discern the most effective marketing strategies.

```
SELECT
    COUNT(b.offer_id) AS offer_usage_count,
    s.description, s.start_date
FROM
    special_offer s
INNER JOIN
    booking b ON s.offer_id = b.offer_id
GROUP BY
    s.offer_id, s.description
ORDER BY
    offer_usage_count DESC;
```

Messages Notifications Data Output			
	offer_usage_count bigint	description character varying (100)	start_date date
1	4	Summer Promotion 2023 (5% off)	2023-06-01
2	4	Spring Discount 2023 (5% off)	2023-03-01
3	3	Early Bird Booking 2024 (5% off)	2024-01-15
4	3	Winter Retreat 2023 (5% off)	2023-12-01
5	3	Valentine's Day Special 2024 (5% off)	2024-02-01
6	3	Fall Getaway 2023 (5% off)	2023-09-01

This query retrieves all properties that have received the best reviews. The output includes the property with an ID of 9 multiple times, indicating it is well-maintained. This property can be considered for further analysis as needed.

```
select a.property_id, p.property_review_id, p.text_review
from property_review p
left join accommodation_rating a
on a.property_review_id = p.property_review_id
where p.location_stars= '5' and p.cleanliness_stars='5' and p.hospitality_stars='5';
```

Messages Notifications Data Output			
	property_id integer	property_review_id integer	text_review text
1	2	3	Outstanding property with excellent amenities.
2	9	10	Superb experience. Would definitely come back.
3	3	14	Great property with friendly hosts. Highly recommende...
4	9	19	Excellent property with top-notch amenities.

This query calculates the total income the website would earn in a day from all properties, under the assumption that every property listed in the income_calculator table is booked for that day.

```
SELECT SUM(website_income_per_day) AS total_website_income  
FROM income_calculator;
```

Messages Notifications Data Output



	total_website_income numeric	
1	504.75	