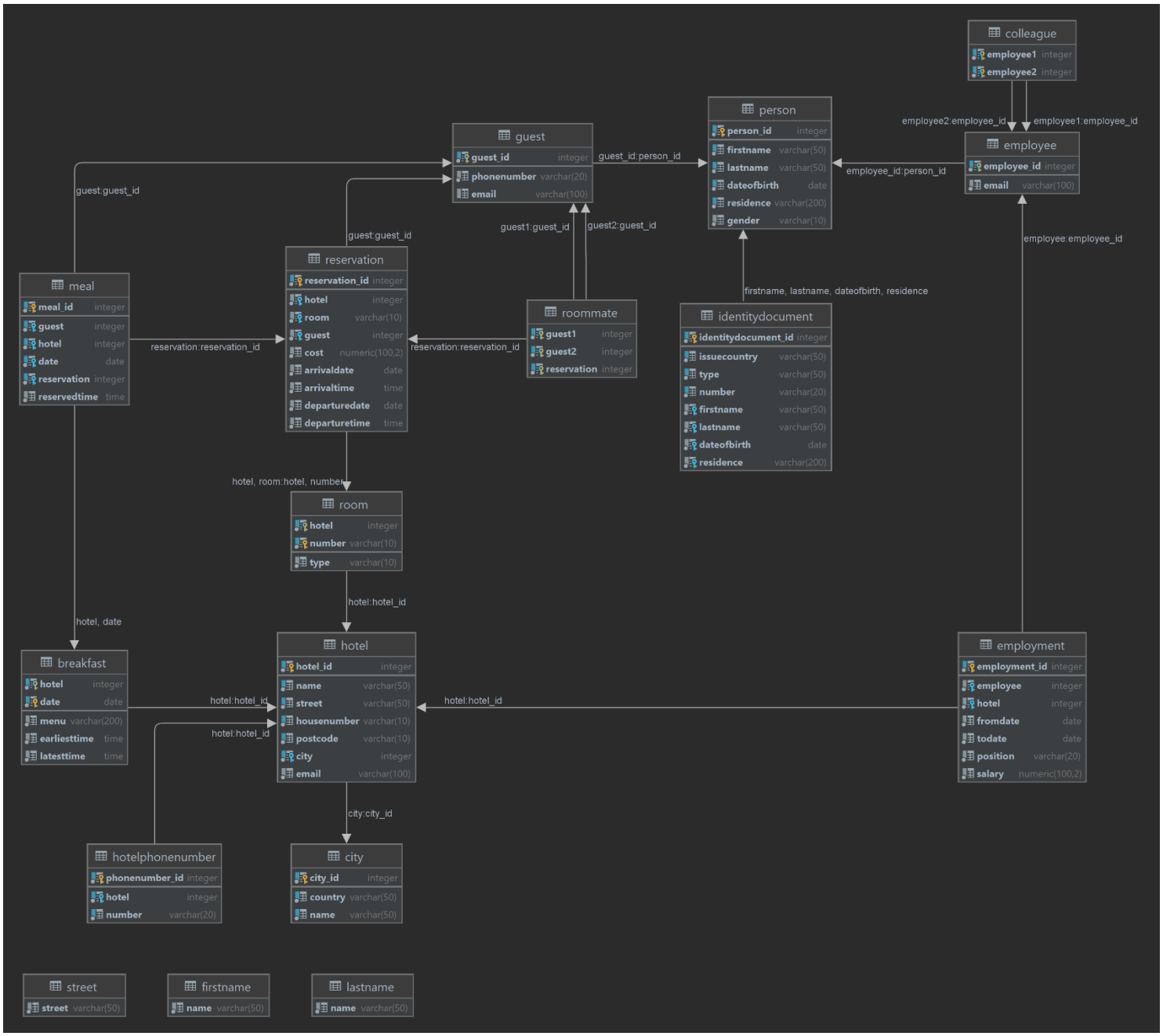
# HOTEL

## ER model

Výsledek transformace relačního modelu na ER model, ze kterého pak vyplývají dotazy vytvářející databázi:




Do obdélníků jsou umístěny názvy jednotlivých tabulek a jejich atributy s vhodnými typy. Šipky ukazují použití cizích klíčů tabulek jinými tabulkami.

## Vytváření databázi

Pro vytváření jednotlivých tabulek v databázi je použit příkaz CREATE. Jsou zavedeny vhodné typy u jednotlivých atributů, atributové a tabulkové integritní omezení, cizí klíče, primární klíče a umělé klíče.

Tabulka City:

```
CREATE TABLE IF NOT EXISTS City (
    city_ID SERIAL PRIMARY KEY,
    country VARCHAR(50) NOT NULL,
    name VARCHAR(50) NOT NULL,
    UNIQUE (country, name)
);
```

Tabulka Hotel:

```
CREATE TABLE IF NOT EXISTS Hotel (
    hotel_ID SERIAL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    street VARCHAR(50) NOT NULL,
    houseNumber VARCHAR(10) NOT NULL,
    postcode VARCHAR(10) NOT NULL CHECK (postcode NOT LIKE '%[^0-9]%'),
    city INTEGER NOT NULL REFERENCES City (city_ID) ON UPDATE CASCADE ON DELETE CASCADE,
    email VARCHAR(100) NOT NULL CHECK (email LIKE '_%@_%.__%'),
    UNIQUE (name, street, houseNumber, postcode, city)
);
```

Tabulka HotelPhoneNumber:

```
CREATE TABLE IF NOT EXISTS HotelPhoneNumber (
    phoneNumber_ID SERIAL PRIMARY KEY,
    hotel INTEGER NOT NULL REFERENCES Hotel (hotel_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    number VARCHAR(20) NOT NULL CHECK (number NOT LIKE '%[^0-9+]%'),
    UNIQUE (hotel, number)
);
```

Tabulka Room:

```
CREATE TABLE IF NOT EXISTS Room (
    hotel INTEGER,
    number VARCHAR(10),
    type VARCHAR (10) NOT NULL,
    PRIMARY KEY (hotel, number),
    CONSTRAINT Room_fk_Hotel FOREIGN KEY (hotel) REFERENCES Hotel (hotel_ID)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

Tabulka Person:

```
CREATE TABLE IF NOT EXISTS Person (
    person_ID SERIAL PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    dateOfBirth DATE NOT NULL,
    residence VARCHAR(200) NOT NULL,
    gender VARCHAR(10) NOT NULL,
    UNIQUE (firstName, lastName, dateOfBirth, residence)
);
```

Tabulka IdentityDocument:

```
CREATE TABLE IF NOT EXISTS IdentityDocument (
    identityDocument_ID SERIAL PRIMARY KEY,
    issueCountry VARCHAR(50) NOT NULL,
    type VARCHAR(50) NOT NULL,
    number VARCHAR(20) NOT NULL,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    dateOfBirth DATE NOT NULL,
    residence VARCHAR(200) NOT NULL,
    UNIQUE (issueCountry, type, number),
    CONSTRAINT IdentityDocument_fk_Person FOREIGN KEY (firstName, lastName, dateOfBirth,
residence)
        REFERENCES Person (firstName, lastName, dateOfBirth, residence)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

Tabulka Employee:

```
CREATE TABLE IF NOT EXISTS Employee (
    employee_ID INTEGER PRIMARY KEY,
    email VARCHAR(100) NOT NULL,
    CONSTRAINT Employee_fk_Person FOREIGN KEY (employee_ID) REFERENCES Person (person_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Employee_ch_email CHECK (email LIKE '_%@_%.__%')
);
```

Tabulka Employment:

```
CREATE TABLE IF NOT EXISTS Employment (
    employment_ID SERIAL PRIMARY KEY,
    employee INTEGER NOT NULL,
    hotel INTEGER NOT NULL,
    fromDate DATE NOT NULL,
    toDate DATE NOT NULL,
    position VARCHAR(20) NOT NULL,
    salary DECIMAL(100, 2) NOT NULL,
    CONSTRAINT Employment_fk_employee FOREIGN KEY (employee) REFERENCES Employee
(employee_ID)
      ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Employment_fk_hotel FOREIGN KEY (hotel) REFERENCES Hotel (hotel_ID)
      ON UPDATE CASCADE ON DELETE CASCADE
);
```

Tabulka Colleague:

```
CREATE TABLE IF NOT EXISTS Colleague (
    employee1 INTEGER NOT NULL,
    employee2 INTEGER NOT NULL,
    PRIMARY KEY (employee1, employee2),
    CONSTRAINT Colleague_fk_Employee1 FOREIGN KEY (employee1) REFERENCES Employee
(employee_ID)
     ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Colleague_fk_Employee2 FOREIGN KEY (employee2) REFERENCES Employee
(employee_ID)
     ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Colleague_ch_employee_ID CHECK (employee1 != Colleague.employee2)
);
```

Tabulka Guest:

```
CREATE TABLE IF NOT EXISTS Guest (
    guest_ID INTEGER,
    phoneNumber VARCHAR(20) NOT NULL,
    email VARCHAR(100),
    PRIMARY KEY (guest_ID),
    CONSTRAINT Guest_fk_Person FOREIGN KEY (guest_ID) REFERENCES Person (person_ID)
     ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Guest_ch_phoneNumber CHECK (phoneNumber NOT LIKE '%[^0-9+]%'),
    CONSTRAINT Guest_ch_email CHECK (email LIKE '_%@_%.__%')
);
```

Tabulka Reservation:

```
CREATE TABLE IF NOT EXISTS Reservation (
    reservation_ID SERIAL PRIMARY KEY,
    hotel INTEGER NOT NULL,
    room VARCHAR(10) NOT NULL,
    guest INTEGER NOT NULL,
    cost DECIMAL(100, 2) NOT NULL,
    arrivalDate DATE NOT NULL,
```

```
    arrivalTime TIME NOT NULL,
    departureDate DATE NOT NULL,
    departureTime TIME NOT NULL,
    CONSTRAINT Reservation_fk_Room FOREIGN KEY (hotel, room) REFERENCES Room (hotel, number)
        ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Reservation_fk_Guest FOREIGN KEY (guest) REFERENCES Guest (guest_ID)
        ON UPDATE CASCADE ON DELETE SET NULL,
    CONSTRAINT Reservation_ch_cost CHECK (cost > 0),
    CONSTRAINT Reservation_ch_arrivalTime CHECK (arrivalTime > '14:00:00'),
    CONSTRAINT Reservation_ch_departureDate CHECK (departureDate > Reservation.arrivalDate),
    CONSTRAINT Reservation_ch_departureTime CHECK (departureTime < '11:00:00')
);
```

Tabulka Roommate:

```
CREATE TABLE IF NOT EXISTS Roommate (
    guest1 INTEGER NOT NULL,
    guest2 INTEGER NOT NULL,
    reservation INTEGER NOT NULL,
    PRIMARY KEY (guest1, guest2, reservation),
    CONSTRAINT Roommate_fk_Guest1 FOREIGN KEY (guest1) REFERENCES Guest (guest_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Roommate_fk_Guest2 FOREIGN KEY (guest2) REFERENCES Guest (guest_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Roommate_fk_Reservation FOREIGN KEY (reservation) REFERENCES Reservation
(reservation_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Roommate_ch_guest CHECK (guest1 != guest2)
);
```

Tabulka Breakfast:

```
CREATE TABLE IF NOT EXISTS Breakfast (
    hotel INTEGER REFERENCES Hotel (hotel_ID) ON UPDATE CASCADE ON DELETE SET NULL,
    date DATE,
    menu VARCHAR(200) NOT NULL,
    earliestTime TIME NOT NULL,
    latestTime TIME NOT NULL,
    PRIMARY KEY (hotel, date),
    CONSTRAINT Breakfast_ch_latestTime CHECK (latestTime > Breakfast.earliestTime)
);
```

Tabulka Meal:

```
CREATE TABLE IF NOT EXISTS Meal (
    meal_ID SERIAL PRIMARY KEY,
    guest INTEGER NOT NULL,
    hotel INTEGER NOT NULL,
    date DATE NOT NULL,
    reservation INTEGER NOT NULL,
    reservedTime TIME NOT NULL,
    UNIQUE (guest, hotel, date, reservation),
    CONSTRAINT Meal_fk_Guest FOREIGN KEY (guest) REFERENCES Guest (guest_ID)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Meal_fk_Breakfast FOREIGN KEY (hotel, date) REFERENCES Breakfast (hotel,
date)
        ON UPDATE CASCADE ON DELETE CASCADE,
    CONSTRAINT Meal_fk_Reservation FOREIGN KEY (reservation) REFERENCES Reservation
(reservation_ID)
        ON UPDATE CASCADE ON DELETE CASCADE
);
```

**Založení tabulek v databázi**

Většina atributů je generovaná pomoci tzn. kartézského součinu pro vyplnění tabulek větším počtem testovacích dat.

### Tabulka City:

Jsou ručně přidány města ze tří zemí pomocí standardního příkazu INSERT INTO.

```
INSERT INTO City (country, name)
VALUES
    ('Czechia', 'Prague'),('Czechia', 'Brno'),('Czechia', 'Ostrava'),
    ('Czechia', 'Liberec'),('Germany', 'Berlin'),('Germany', 'Dresden'),
    ('Germany', 'Munich'),('Germany', 'Hamburg'),('Poland', 'Warsaw'),
    ('Poland', 'Krakow'),('Poland', 'Lublin'),('Poland', 'Gdansk');
```

### Tabulka Hotel:

Je založena a vyplněna pomocná tabulka Street, použitá pro vytváření atributu street. Čísla domů a PSČ jsou generovaný náhodně pomocí funkcí random() a ceil(), která zaokrouhluje výsledek.

```
DROP TABLE IF EXISTS Street CASCADE;
CREATE TABLE Street (
    street VARCHAR(50) NOT NULL
);

INSERT INTO Street
VALUES ('First street'),('Second street'),('Third street'),
       ('Fourth street'),('Fifth street'),('Main street'),
       ('World street'),('Hill street'),('Lake street');

INSERT INTO Hotel (name, street, houseNumber, postcode, city, email)
SELECT c.name || 'Hotel', s.street,
       ceil(random() * 100), ceil(random() * 80000) + 10000, c.city_ID,
       c.name || 'Hotel' || '@' || (case ceil(random() * 3)
           when 1 then 'gmail'
           when 2 then 'mail'
           when 3 then 'yahoo'end) || '.com'
FROM City AS c, Street AS s
WHERE random() > 0.85;
```

### Tabulka HotelPhoneNumber:

Telefonní čísla hotelů jsou generovány náhodně.

```
INSERT INTO HotelPhoneNumber (hotel, number)
SELECT h.hotel_ID, '+' || ceil(random() * 8999999999) + 1000000000
FROM Hotel AS h, generate_series(1,4)
WHERE random() > 0.60;
```

### Tabulka Room:

Každý hotel má náhodné číslo a typ. Pokoje jsou očíslovány v pořadí od prvního až do nějakého maxima.

```
INSERT INTO Room (hotel, number, type)
SELECT h.hotel_ID, n, (array['single', 'double', 'triple'])[ceil(random() * 3)]
FROM Hotel AS h, generate_series(1, (ceil(random() * 90)+10)::INTEGER) AS n;
```

### Tabulka Person:

Jsou založeny a vyplněny pomocné tabulky FirstName a LastName pro vytváření odpovídajících atributů. Každá osoba má náhodné datum narození a gender.

```
CREATE TABLE FirstName (
    name VARCHAR(50) NOT NULL
);

INSERT INTO FirstName
VALUES ('Alex'), ('Kris'), ('Tracy'), ('Dakota'), ('Skyler'),
       ('Charlie'), ('River'), ('Justice'), ('Frankie');
```

```
CREATE TABLE LastName (
    name VARCHAR(50) NOT NULL
);

INSERT INTO LastName
VALUES ('White'), ('Black'), ('Purple'), ('Blue'), ('Green'),
       ('Brown'), ('Yellow'), ('Pink'), ('Gray');

INSERT INTO Person (firstName, lastName, dateOfBirth, residence, gender)
SELECT f.name, l.name,
       timestamp '1930-01-01' + random()
                              * (timestamp '2000-01-01' - timestamp '1930-01-01'),
       c.country || ', ' || c.name || ', ' || s.street || ' ' || ceil(random() * 100),
       (array['male', 'female', 'other'])[ceil(random() * 3)]
FROM FirstName AS f, LastName AS l, City AS c, Street AS s
WHERE random() > 0.50;
```

Tabulka IdentityDocument:

Každý doklad odpovídá jedné osobě. Země jsou převzaty z trvalých adres osob. Typy a čísla dokladů jsou generovány náhodně.

```
INSERT INTO IdentityDocument (issueCountry, type, number,
                              firstName, lastName, dateOfBirth, residence)
SELECT split_part(p.residence, ', ', 1),
       (array['passport', 'driver licence'])[ceil(random() * 2)],
       ceil(random() * 1000000000) + 10000000000,
       p.firstName, p.lastName, p.dateOfBirth, p.residence
FROM Person AS p;
```

Tabulka Employee:

Některé osoby jsou zaměstnaní v jednom nebo více hotelů a mají pracovní e-mail.

```
INSERT INTO Employee (employee_ID, email)
SELECT p.person_ID, p.lastName || p.person_ID || '@' || (case ceil(random() * 3)
    when 1 then 'gmail'
    when 2 then 'mail'
    when 3 then 'yahoo'end) || '.com'
FROM Person AS p
WHERE random() > 0.70;
```

Tabulka Employment:

Náhodně se generují datum zahájení platnosti pracovní smlouvy a pozice zaměstnance. Následně se pomocí příkazu UPDATE přidávají datum konce platnosti pracovní smlouvy, závislé na datu zahájeni, a mzda v eurech, závislá na pozici.

```
INSERT INTO Employment (employee, hotel, fromDate, toDate, position, salary)
SELECT e.employee_ID, ceil(random() * (SELECT count(*) FROM Hotel)),
       timestamp '1980-01-01' + random()
                 * (timestamp '2022-01-01' - timestamp '1980-01-01'), '9999-12-31',
       (array['director', 'manager', 'cleaner', 'receptionist'])[ceil(random() * 4)],
       ceil(random() * 200) * 5 + 500
FROM Employee AS e, generate_series(1, 20)
WHERE random() > 0.70;

UPDATE Employment
SET toDate = fromDate + random() * (INTERVAL '2 years') + INTERVAL '6 months',
    salary = (case position
        when 'director' then ceil(random() * 200) * 5 + 1000
        when 'manager' then ceil(random() * 100) * 5 + 500
        when 'cleaner' then ceil(random() * 50) * 5 + 200
        when 'receptionist' then ceil(random() * 75) * 5 + 300 end);
```

### Tabulka Colleague:

Tabulka obsahuje všechny páry zaměstnanců, které pracovali v jednom hotelu současně.

```
INSERT INTO Colleague (employee1, employee2)
SELECT DISTINCT e1.employee_ID, e2.employee_ID
FROM Employee AS e1 JOIN Employment Emp1 on e1.employee_ID = Emp1.employee,
     Employee AS e2 JOIN Employment Emp2 on e2.employee_ID = Emp2.employee
WHERE (e1.employee_ID != e2.employee_ID) AND (Emp1.hotel = Emp2.hotel)
        AND (Emp1.fromDate BETWEEN Emp2.fromDate AND Emp2.toDate);
```

### Tabulka Guest:

Některé osoby jsou hosty hotelů. Mají telefonní číslo a nepovinně e-mail.

```
INSERT INTO Guest (guest_ID, phoneNumber, email)
SELECT p.person_ID, '+' || ceil(random() * 8999999999) + 1000000000,
       p.firstName || p.lastName || p.person_ID || '@' ||
       (case ceil(random() * 3)
           when 1 then 'gmail'
           when 2 then 'mail'
           when 3 then 'yahoo'end) || '.com'
FROM Person AS p
WHERE random() > 0.50;
```

### Tabulka Reservation:

Hosty mohou mít rezervací v uvedených hotelech. Všechna data se generují náhodně. Tabulka má více než 32k řádků.

```
INSERT INTO Reservation (hotel, room, guest, cost,
                         arrivalDate, arrivalTime, departureDate, departureTime)
SELECT r.hotel, r.number, g.guest_ID, ceil(random() * 20) * 5 + 50,
       timestamp '2022-01-01' + random()
                 * (timestamp '2024-01-01' - timestamp '2022-01-01'),
       date_trunc('minute', time '15:00:00' + random()
                 * (time '23:59:00' - time '15:00:00')),
       '9999-12-31', date_trunc('minute', time '05:00:00'
                 + random() * (time '11:00:00' - time '05:00:00'))
FROM Room AS r, Guest AS g
WHERE random() > 0.95;

UPDATE Reservation
SET departureDate = arrivalDate + random() * (INTERVAL '7 days') + INTERVAL '1 day';
```

### Tabulka Roommate:

Každý host, který má rezervaci, se může ubytovat s dalšími hosty.

```
INSERT INTO Roommate (guest1, guest2, reservation)
SELECT DISTINCT g1.guest_ID, g2.guest_ID, R.reservation_ID
FROM Guest AS g1 JOIN Reservation R on g1.guest_ID = R.guest,
     Guest AS g2
WHERE (random() > 0.9999) AND (g1.guest_ID != g2.guest_ID);
```

### Tabulka Breakfast:

Každý hotel předem organizuje snídaňové menu pro následující 2 týdny. Data se generují náhodně. Funkce data_trunc(…) zaokrouhluje čas na minuty.

```
INSERT INTO Breakfast (hotel, date, menu, earliestTime, latestTime)
SELECT h.hotel_ID, curDate, (array['scrambled egg', 'bavarian sausages', 'club sandwich',
        'oatmeal porridge', 'yogurt with muesli', 'pancakes with jam'])[ceil(random() * 4)],
       date_trunc('minute', time '05:00:00' + random()
                        * (time '07:30:00' - time '05:00:00')),
       date_trunc('minute', time '10:00:00' + random()
```

```
                        * (time '11:00:00' - time '10:00:00'))
FROM Hotel AS h, generate_series('2022-01-01', '2022-01-14', INTERVAL '1 day') as curDate;
```

Tabulka Meal:

Hosté mající rezervací v hotelech mají tam možnost snídaně po celou dobu pobytu.

```
INSERT INTO Meal (guest, hotel, date, reservation, reservedTime)
SELECT g.guest_ID, R.hotel, curDate, R.reservation_ID,
       date_trunc('minute', b.earliestTime + random() * ((b.latestTime) - (b.earliestTime)))
FROM Guest AS g JOIN Reservation R on (g.guest_ID = R.guest),
     generate_series(R.arrivalDate, R.departureDate, INTERVAL '1 day') as curDate,
     Breakfast AS b
WHERE (random() > 0.50) AND (curDate BETWEEN '2022-01-01' AND '2022-01-14')
    AND (b.date = curDate) AND (R.hotel = b.hotel);
```

**SQL dotazy pro získaní údajů z databáze**

1) Výsledek dotazu ukazuje počet rezervací v hotelech německých měst, pokud je větší než 5000. Data jsou seřazeny podle počtu rezervací sestupně.

Dotaz pokrývá:

- agregaci a podmínku na hodnotu agregační funkce
- řazení
- vnořený SELECT

3 rows

| | hotelname | profit |
|---|---|---|
| 1 | WarsawHotel12 | 298750 |
| 2 | WarsawHotel15 | 300320 |
| 3 | WarsawHotel17 | 289250 |

```
SELECT T.city AS City, count(T.rs) AS Reservations
FROM (
    SELECT c.country AS country, c.name AS city, R.reservation_ID AS rs
    FROM City AS c, Hotel AS H, Reservation AS R
    WHERE (c.city_ID = H.city) AND (H.hotel_ID = R.hotel)
    ) AS T
WHERE T.country IN ('Germany')
GROUP BY City HAVING (count(T.rs) > 5000)
ORDER BY Reservations DESC;
```

2) Výsledek dotazu ukazuje všech hostů, které mají příjmení „Blue“ a jméno „River“, s jejich e-maily.

Dotaz pokrývá:

- vnitřní spojení tabulek
- podmínku na data
- množinové operace

```
SELECT p.firstName, p.lastName, G.email
FROM Person AS p INNER JOIN Guest G
                on p.person_ID = G.guest_ID
WHERE p.lastName = 'Blue'
 INTERSECT
SELECT p.firstName, p.lastName, G.email
FROM Person AS p
     INNER JOIN Guest G on p.person_ID = G.guest_ID
WHERE p.firstName = 'River';
```

22 rows

| | firstname | lastname | email |
|---|---|---|---|
| 1 | River | Blue | RiverBlue3178@mail.com |
| 2 | River | Blue | RiverBlue3049@gmail.com |
| 3 | River | Blue | RiverBlue2894@mail.com |
| 4 | River | Blue | RiverBlue3017@mail.com |
| 5 | River | Blue | RiverBlue3214@mail.com |
| 6 | River | Blue | RiverBlue2897@mail.com |
| 7 | River | Blue | RiverBlue2936@mail.com |
| 8 | River | Blue | RiverBlue3136@gmail.com |
| 9 | River | Blue | RiverBlue3180@mail.com |
| 10 | River | Blue | RiverBlue3179@yahoo.com |
| 11 | River | Blue | RiverBlue2940@mail.com |
| 12 | River | Blue | RiverBlue3182@yahoo.com |
| 13 | River | Blue | RiverBlue2935@mail.com |
| 14 | River | Blue | RiverBlue3288@mail.com |
| 15 | River | Blue | RiverBlue3050@gmail.com |
| 16 | River | Blue | RiverBlue3181@gmail.com |
| 17 | River | Blue | RiverBlue3286@mail.com |
| 18 | River | Blue | RiverBlue3052@gmail.com |
| 19 | River | Blue | RiverBlue2941@gmail.com |
| 20 | River | Blue | RiverBlue3016@mail.com |
| 21 | River | Blue | RiverBlue3134@mail.com |
| 22 | River | Blue | RiverBlue3094@yahoo.com |

3) Výsledek dotazu ukazuje hotely, které podávají bavorské klobásy nebo club sandwich na snídani 1 ledna 2022.

Dotaz pokrývá:

- vnitřní spojení tabulek
- podmínku na data
- vnořený SELECT

```sql
SELECT h.name || h.hotel_ID AS Hotel, B.date, B.earliestTime, B.earliestTime, B.menu
FROM Hotel AS h INNER JOIN Breakfast B on h.hotel_ID = B.hotel
WHERE exists (
    SELECT * FROM Breakfast
    WHERE (b.date = '2022-01-01') AND
          (b.menu IN ('bavarian sausages', 'club sandwich')));
```

8 rows

| | hotel | date | earliesttime | earliesttime | menu |
|---|---|---|---|---|---|
| 1 | HamburgHotel2 | 2022-01-01 | 06:59:00 | 06:59:00 | club sandwich |
| 2 | LiberecHotel3 | 2022-01-01 | 06:47:00 | 06:47:00 | bavarian sausages |
| 3 | GdanskHotel6 | 2022-01-01 | 07:14:00 | 07:14:00 | bavarian sausages |
| 4 | LiberecHotel7 | 2022-01-01 | 07:07:00 | 07:07:00 | club sandwich |
| 5 | HamburgHotel9 | 2022-01-01 | 07:10:00 | 07:10:00 | club sandwich |
| 6 | BrnoHotel10 | 2022-01-01 | 06:12:00 | 06:12:00 | club sandwich |
| 7 | MunichHotel11 | 2022-01-01 | 06:46:00 | 06:46:00 | club sandwich |
| 8 | GdanskHotel18 | 2022-01-01 | 05:39:00 | 05:39:00 | bavarian sausages |

4) Výsledek dotazu ukazuje hotely, které mají větší než 1 kontaktních telefonních čísel. Data jsou seřazeny podle abecedy vzestupně.

Dotaz pokrývá:

- vnější spojení tabulek
- agregaci a podmínku na hodnotu agregační funkce
- řazení

```sql
SELECT H.name || H.hotel_ID AS HotelName, count(n) AS Numbers
FROM HotelPhoneNumber AS n
     LEFT OUTER JOIN Hotel H on n.hotel = H.hotel_ID
GROUP BY HotelName HAVING (count(n) > 1)
ORDER BY HotelName ASC;
```

9 rows

| | hotelname | numbers |
|---|---|---|
| 1 | BrnoHotel16 | 2 |
| 2 | DresdenHotel14 | 3 |
| 3 | DresdenHotel4 | 3 |
| 4 | GdanskHotel18 | 2 |
| 5 | GdanskHotel6 | 2 |
| 6 | HamburgHotel2 | 4 |
| 7 | HamburgHotel8 | 2 |
| 8 | HamburgHotel9 | 2 |
| 9 | LublinHotel13 | 2 |

5) Výsledek dotazu ukazuje příjmy hotelů podle rezervací ve městě s kódem ‘9‘.

Dotaz pokrývá:

- vnější spojení tabulek
- podmínku na data
- agregaci

```sql
SELECT H.name || H.hotel_ID AS HotelName, sum(R.cost) AS Profit
FROM Reservation AS R LEFT OUTER JOIN Hotel H on R.hotel = H.hotel_ID
WHERE H.city = '9' GROUP BY HotelName
```

3 rows

| | hotelname | profit |
|---|---|---|
| 1 | WarsawHotel12 | 298750 |
| 2 | WarsawHotel15 | 300320 |
| 3 | WarsawHotel17 | 289250 |