Tugas IF4040 Pemodelan Data Lanjut

Temporal and Moving Objects Database

Oleh:
Kelompok 2

Ferdian Ifkarsyah	13517024
Hafidh Redyanto	13517061
Didik Supriadi	13517069
Muhammad Nurdin Husen	13517112
Louis Cahyadi	13517126

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TINGGI ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2020

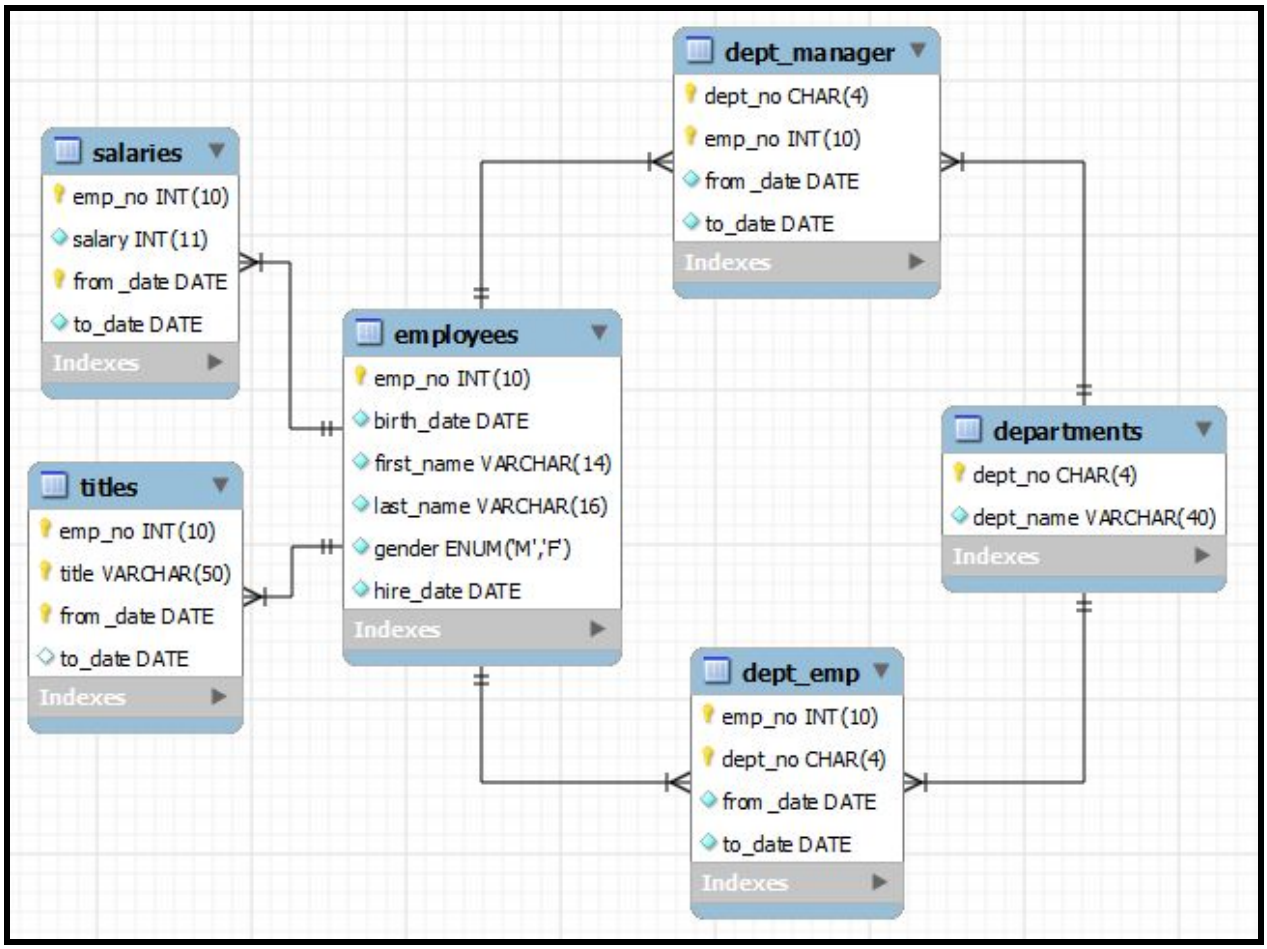
1. Deskripsi Studi Kasus

1.1. Deskripsi

1.2. Skema Basis Data dan Sumber

1.2.1. Skema Basis Data

Berikut ini merupakan skema basis data dari studi kasus yang dipilih pada pengerjaan tugas ini:



1.2.2. Sumber

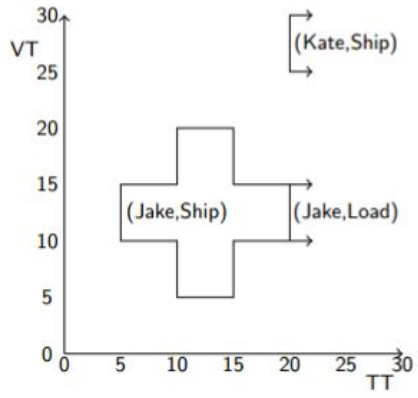
Sumber data diperoleh dari tautan berikut [MySQL Sample Databases](#) pada bagian MySQL's Sample Employee Database.

1.3. Model Representasi

Model representasi yang dipilih untuk studi kasus ini adalah Snodgrass tanpa menggunakan *transaction time.* Model representasi snodgrass dinilai sesuai dengan skema basis data yang digunakan dikarenakan data temporal yang digunakan memiliki *from_date* dan *to_date* yang merepresentasikan Vs dan Ve dari model snodgrass.

Snodgrass' Tuple-Timestamped Representation Scheme (2)

Example: the 1NF relation of *dept* relation



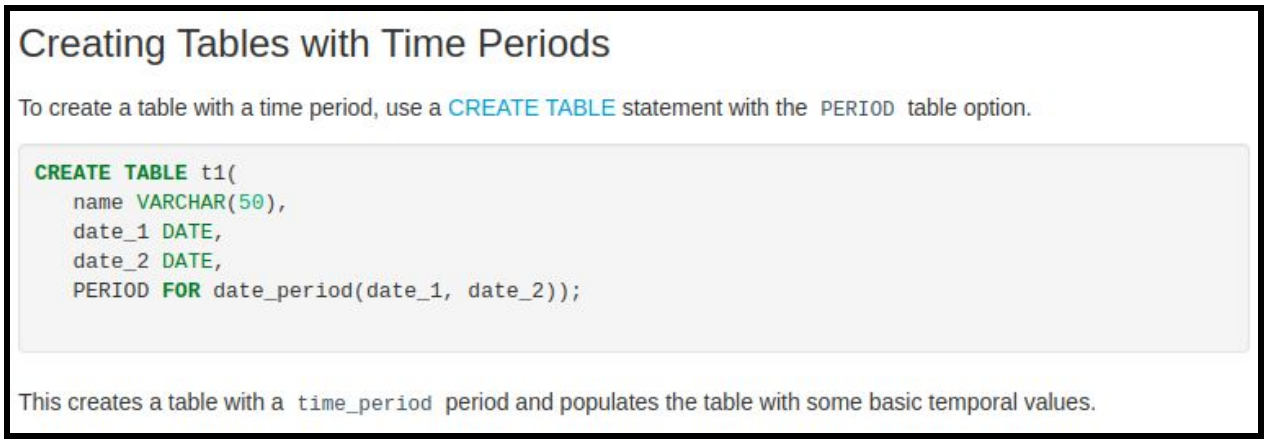
dept

Emp	Dept	T_s	T_e	V_s	V_e
Jake	Ship	5	9	10	15
Jake	Ship	10	14	5	20
Jake	Ship	15	19	10	15
Jake	Load	20	*UC*	10	15
Kate	Ship	20	*UC*	25	30

2. Alasan Pemilihan MariaDB

Beberapa alasan MariaDB dipilih sebagai DBMS adalah sebagai berikut.

1. Data kasus studi didapatkan dalam skema *relational*. Sehingga, tidak perlu dilakukan perubahan basis data.
2. MariaDB mendukung table *application-time period* untuk melakukan *query* pada tabel secara efisien.



Creating Tables with Time Periods

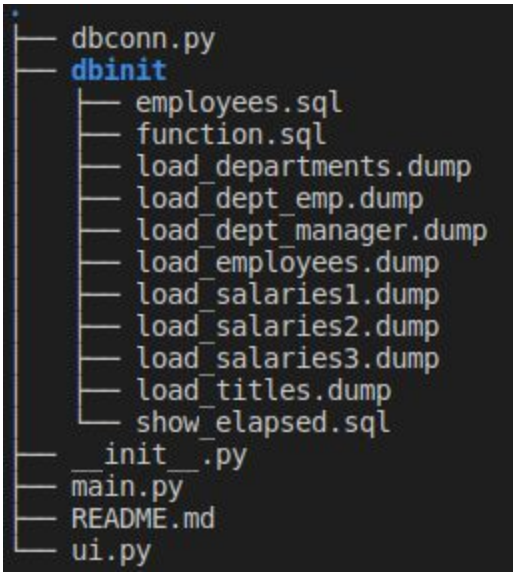
To create a table with a time period, use a CREATE TABLE statement with the `PERIOD` table option.

```
CREATE TABLE t1(
   name VARCHAR(50),
   date_1 DATE,
   date_2 DATE,
   PERIOD FOR date_period(date_1, date_2));
```

This creates a table with a `time_period` period and populates the table with some basic temporal values.

3. Arsitektur Program

Program yang dibuat untuk mengetes temporal query berbentuk CLI(Command Line Interface) sederhana.



Root folder berisi **main.py, dbconn.py** dan **ui.py**. File **main.py** adalah file utama program tempat dimulainya flow eksekusi. Kemudian, file **dbconn.py**, sesuai namanya berperan untuk menginisiasi koneksi database. File **ui.py** berisi fungsi **print_menu** yang memberi panduan cara memakai program.

Folder **dbinit** berisi file-file yang sudah harus dieksekusi sebelum database MariaDB siap dipakai oleh program. Fungsi-fungsi kustom yang didefinisikan untuk *temporal query* diletakkan pada file **function.sql**.

4. Implementasi

Pada bagian ini akan dijelaskan implementasi dari semua operasi yang dideskripsikan pada materi *temporal algebra*, termasuk operasi update, insert, dan delete.

4.1. Tipe Data

Untuk menyatakan valid time pada tabel - tabel yang ada digunakan kolom from_date dan to_date dengan tipe data DATE. Selain itu, diimplementasikan fitur *application-time period* dengan cara

```
ALTER TABLE <table_name>
ADD PERIOD FOR valid_period(date_start, date_end);
```

```
ALTER TABLE salaries
ADD PERIOD FOR valid_period(from_date, to_date);
```

Dengan syarat nilai dari from_date < to_date.

4.2. Operasi

4.2.1. Insert

Operasi insert dapat dilakukan dengan query sederhana dikarenakan nilai *from_date* dan *to_date* merupakan atribut umum dari sebuah tabel dengan tipe datetime.

```
INSERT INTO <table_name>
VALUES <value>
```

```
INSERT INTO salaries
VALUES (id01, 10000, '2020-01-01', '2020-10-01');
```

4.2.2. Delete

Operasi delete dengan kondisi temporal dapat dilakukan dengan memanfaatkan fitur PORTION dan PERIOD yang disediakan oleh MariaDB. Dengan menggunakan PORTION dan PERIOD maka, MariaDB secara otomatis melakukan penanganan from_date dan to_date yang baru. Sebagai contoh, jika terdapat record

emp_no	salary	from_date	to_date
10001	100000	2020-01-01	2020-12-31

Dan dilakukan penghapusan pada period ‘2020-03-15’ hingga ‘2020-07-15’ maka hasilnya adalah

emp_no	salary	from_date	to_date
10001	100000	2020-01-01	2020-03-15
10001	100000	2020-07-15	2020-12-31

Berikut ini merupakan query pembuatan prosedur untuk melakukan penghapusan data pada periode waktu tertentu.

```
DROP PROCEDURE IF EXISTS employees.DeleteTemporal;

DELIMITER $$
$$
CREATE DEFINER=`root`@`%` PROCEDURE
`employees`.`DeleteTemporal`(
      IN table_name VARCHAR(15),
      IN start_date VARCHAR(10),
      IN end_date VARCHAR(10)
)
BEGIN
      SET @sql = CONCAT('DELETE FROM ', table_name, " FOR
PORTION OF valid_period FROM '", start_date, "' TO '",
end_date, "'");
      PREPARE stmt FROM @sql;
      EXECUTE stmt;
      DEALLOCATE PREPARE stmt;
END$$
DELIMITER ;
```

```
CALL employees.DeleteTemporal(‘salaries’,
‘1986-06-26’,’1987-06-27);
```

Selain itu operasi ini dapat juga dilakukan berdasarkan kondisi tertentu menggunakan klausa WHERE seperti contoh di bawah ini

```
DELETE salaries
FOR PORTION OF valid_period
FROM DATE '2012-03-15'
TO DATE '2012-07-15'
WHERE emp_no = 10001;
```

4.2.3. Update (non temporal)

Operasi update pada row berdasarkan kondisi non-temporal dapat dilakukan dengan query sederhana mengingat nilai from_date dan to_date merupakan atribut umum dari sebuah tabel dengan tipe datetime.

```
UPDATE <table_name>
SET <something>
WHERE <condition>
```

```
UPDATE salaries
SET salary = 123456
WHERE emp_no = 10001 AND from_date = ‘1987-06-26’ AND
to_date = ‘1988-06-25’
```

4.2.4. Update (temporal)

Operasi update dengan kondisi temporal dapat memanfaatkan fitur PORTION dan PERIOD dari MariaDB. Dengan menggunakan PORTION dan PERIOD maka, MariaDB secara otomatis melakukan penanganan from_date dan to_date yang baru seperti pada operasi delete. Bentuk dan contoh query yang dapat digunakan adalah sebagai berikut:

```
UPDATE <table_name> FOR PORTION OF valid_period
FROM <start_date>
TO <end_date>
SET <something>
WHERE <condition>;
```

```
UPDATE salaries FOR PORTION OF valid_period
FROM ‘1986-06-26'
TO '1987-06-26'
SET salary = 100000
WHERE emp_no = 10001
```

4.2.5. Temporal Projection

Operasi temporal projection dapat dilakukan dengan menggunakan operasi SELECT atribut tertentu dari sebuah tabel ditambah dengan valid time dari data tersebut. Sehingga implementasinya dilakukan dengan membuat sebuah prosedur yang menjalankan SELECT pada atribut terpilih ditambah dengan atribut from_date dan to_date. Berikut adalah query untuk membuat prosedur projection

```
DROP PROCEDURE IF EXISTS employees.projection;

DELIMITER $$
```

```
$$
CREATE PROCEDURE `employees`.`projection`(
IN table_name VARCHAR(15),
IN column_name VARCHAR(15))
BEGIN
      SET @sql = CONCAT('SELECT ',column_name, ',
from_date, to_date FROM ', table_name);
      PREPARE stmt FROM @sql;
      EXECUTE stmt;
      DEALLOCATE PREPARE stmt;
END$$
DELIMITER ;
```

```
CALL employees.projection('salaries','emp_no');
```

4.2.6. Temporal Selection

Dikarenakan informasi temporal dari sebuah data disimpan pada tabel sebagai salah satu atribut maka operasi selection dapat menggunakan operasi SELECT dari SQL pada umumnya.

```
SELECT <column_name>
FROM <table_name>
WHERE <condition>
```

```
SELECT *
FROM salaries
WHERE emp_no = 10001
```

4.2.7. Temporal Union

Operasi temporal union dapat dilakukan dengan melakukan operasi UNION pada kedua tabel dan dilakukan *grouping* berdasarkan *primary key* kedua tabel. Setelah itu untuk setiap *grouping* tuple, nilai *from_date* dikalkulasi dengan nilai min(*from_date*) dan nilai *to_date* dikalkulasi dengan nilai max(*to_date*) untuk operasi temporal unionnya. Operasi ini hanya berlaku untuk dua buah tabel yang memiliki atribut yang identik.

```
SELECT <primary_key>, MIN(from_date) as from_date,
MAX(to_date) as to_date
FROM
(SELECT * from <table_name>
UNION
SELECT * FROM <table_name>) AS u
GROUP BY <primary_key>
```

```
SELECT emp_no, dept_no, min(from_date) as from_date,
max(to_date) as to_date
FROM
(SELECT * FROM dept_manager
UNION
SELECT * FROM dept_emp) AS u
GROUP BY emp_no, dept_no;
```

4.2.8. Temporal Difference

Operasi temporal difference dapat dilakukan dengan melakukan operasi DIFFERENCE pada kedua tabel dan dan dilakukan *grouping* berdasarkan *primary key* kedua tabel. Setelah itu untuk setiap *grouping* tuple, nilai *from_date* dikalkulasi dengan nilai min(*from_date*) dan nilai *to_date* dikalkulasi dengan nilai max(*to_date*) untuk operasi temporal difference. Operasi ini hanya berlaku untuk dua buah tabel yang memiliki atribut yang identik.

```
SELECT <primary_key>, MIN(from_date) as from_date,
MAX(to_date) as to_date
FROM
(SELECT * from <table_name>
EXCEPT
SELECT * FROM <table_name>) AS d
GROUP BY <primary_key>
```

```
SELECT emp_no, dept_no, min(from_date) as from_date,
max(to_date) as to_date
FROM
(SELECT * FROM dept_emp
EXCEPT
SELECT * FROM dept_manager) AS u
GROUP BY emp_no, dept_no LIMIT 20;
```

4.2.9. Temporal Join

Operasi temporal join dapat dilakukan dengan melakukan operasi INNER JOIN pada kedua tabel berdasarkan kolom yang sama pada kedua tabel. Setelah itu, nilai `from_date` dikalkulasi dengan nilai, jika `<t1.from_date> >= <t2.from_date>` maka `from_date` sama dengan `<t1.from_date>` dan jika sebaliknya maka `from_date` sama dengan `<t2.from_date>`. Hal tersebut berlaku juga untuk nilai `to_date` yang

dikalkulasi dengan nilai `<t1.to_date> <= <t2.to_date>` untuk operasi temporal join.

```
SELECT <column-name>
      IF(<t1.from_date> >= <t2.from_date>,
            <t1.from_date>, <t2.from_date>)
                  AS from_date,
      IF(<t1.to_date> <= <t2.to_date>,
            <t1.to_date>, <t2.to_date>)
                  AS to_date
FROM <t1> INNER JOIN <t2>
ON <t1.column_name> = <t2.column_name>
```

```
SELECT dept_manager.emp_no,
      dept_manager.dept_no,
      salaries.salary,
      IF(dept_manager.from_date >= salaries.from_date,
            dept_manager.from_date, salaries.from_date)
                  AS from_date,
      IF(dept_manager.to_date <= salaries.to_date,
            dept_manager.to_date, salaries.to_date)
                  AS to_date
FROM dept_manager INNER JOIN salaries
ON dept_manager.emp_no = salaries.emp_no;
```

4.2.10. Timeslice Operator

Pada model representasi yang digunakan, hanya terdapat valid time untuk informasi temporal dari sebuah data. Sehingga timeslice operator menjadi tidak diperlukan pada model ini.

5. Contoh Query dan Hasil

- Buatlah contoh untuk *queries* yang melibatkan *temporal relationships* pada *Allen's 13 interval relations*.

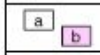
precedes	meets	overlaps	finished by	contains	starts	equals	started by	during	finishes	overlap-ped by	met by	preceded by
												
p	m	o	F	D	s	e	S	d	f	O	M	P

Table 1. Allen's thirteen basic relations

5.1. Query

ID	Query	SQL Syntax	Interval Relations
Q1	Temukan daftar department dan jumlah employee-nya pada tahun 1999.	SELECT de.dept_no, dn.dept_name, COUNT(de.emp_no) as count_employee FROM employees e JOIN dept_emp de ON e.emp_no = de.emp_no JOIN departments dn ON de.dept_no = dn.dept_no WHERE de.from_date >= '1999-01-01' AND de.from_date <= '1999-01-01' GROUP BY dn.dept_no	contains
Q2	Temukan jumlah employee untuk setiap departemen pada saat ini.	SELECT dn.dept_no, dn.dept_name, COUNT(de.emp_no) as count_employee FROM employees e JOIN dept_emp de ON e.emp_no = de.emp_no JOIN departments dn ON de.dept_no = dn.dept_no WHERE de.to_date >= NOW() GROUP BY dn.dept_no ORDER BY dn.dept_no	finished by
Q3	Temukan jumlah employee yang sudah keluar pada setiap departemen sebelum manajer saat ini menjadi manajer di departemen itu.	SELECT dn.dept_no, dn.dept_name, COUNT(de.emp_no) as count_employee FROM employees e JOIN dept_emp de ON e.emp_no = de.emp_no JOIN departments dn ON de.dept_no = dn.dept_no JOIN dept_manager dm ON dm.dept_no = dn.dept_no WHERE de.to_date <= dm.from_date AND dm.to_date >= NOW() GROUP BY dn.dept_no ORDER BY dn.dept_no	precedes by
Q4	Temukan employee yang keluar dari department “Production” pada saat manajer	SELECT e.first_name, e.last_name, e.gender FROM employees e JOIN dept_emp de ON e.emp_no = de.emp_no JOIN departments dn ON de.dept_no =	meets

	departemen tersebut menjabat	dn.dept_no JOIN dept_manager dm ON dm.dept_no = dn.dept_no WHERE de.to_date = dm.from_date AND dm.to_date >= NOW() AND dn.dept_name = "Production"	
Q5	Temukan jumlah employee pada setiap departemen yang sedang bekerja pada departemen tersebut namun pernah bekerja dengan manajer sebelumnya	SELECT dn.dept_no, dn.dept_name, COUNT(de.emp_no) as count_employee FROM departments dn JOIN dept_manager dm ON dn.dept_no = dm.dept_no JOIN dept_emp de ON dn.dept_no = de.dept_no JOIN employees e ON de.emp_no = e.emp_no WHERE dm.to_date <= NOW() AND de.from_date <= dm.to_date AND de.to_date >= NOW() GROUP BY dn.dept_no	overlaps
Q6	Temukan semua employee dan jabatannya pada departemen Finance yang	SELECT e.emp_no, e.first_name, e.last_name, t.title FROM departments dn JOIN dept_manager dm ON dn.dept_no =	starts

	masuk bersamaan dengan manajer pada departemen tersebut	dm.dept_no JOIN dept_emp de ON dn.dept_no = de.dept_no JOIN employees e ON de.emp_no = e.emp_no JOIN titles t ON t.emp_no = e.emp_no WHERE de.from_date = dm.from_date AND dm.to_date >= NOW() AND de.to_date >= NOW() AND t.to_date >= NOW() AND dn.dept_name = "Finance"	
Q7	Temukan semua employee lain yang memiliki waktu kontrak kerja sama dengan ‘Georgi Facello’	SELECT e2.emp_no, e2.first_name, e2.last_name, e2.gender FROM dept_emp de1 JOIN employees e1 ON e1.emp_no = de1.emp_no JOIN dept_emp de2 JOIN employees e2 ON e2.emp_no = de2.emp_no WHERE e1.first_name = 'Georgi' AND e1.last_name = 'Facello' AND de1.from_date = de2.from_date AND de1.to_date = de2.to_date AND !(e2.first_name = 'Georgi' AND e2.last_name = 'Facello')	equal

5.2. Hasil

ID Query	**Hasil**

Q1

dept_no	dept_name	count_employee
d001	Marketing	4
d002	Finance	6
d003	Human Resources	3
d004	Production	19
d005	Development	17
d006	Quality Management	6
d007	Sales	8
d008	Research	6
d009	Customer Service	6

Q2

dept_no	dept_name	count_employee
d001	Marketing	14842
d002	Finance	12437
d003	Human Resources	12898
d004	Production	53304
d005	Development	61386
d006	Quality Management	14546
d007	Sales	37701
d008	Research	15441
d009	Customer Service	17569

Q3

dept_no	dept_name	count_employee
d001	Marketing	620
d002	Finance	303
d003	Human Resources	639
d004	Production	7469
d005	Development	3298
d006	Quality Management	1259
d007	Sales	1356
d008	Research	556
d009	Customer Service	1913

Q4

first_name	last_name	gender
Geoff	Linnainmaa	M
Hugo	Ernst	M
Domenico	Majewski	M
Kagan	Majewski	F
Navid	Aumann	M
Susanne	Schaft	M
Rimli	Merli	F

Q5

dept_no	dept_name	count_employee
d001	Marketing	5549
d002	Finance	3940
d003	Human Resources	5896
d004	Production	75048
d005	Development	28735
d006	Quality Management	17077
d007	Sales	14994
d008	Research	5262
d009	Customer Service	20384

Q6

emp_no	first_name	last_name	title
299982	Juichirou	Hiraishi	Senior Staff
426497	Sanjai	Streit	Senior Staff
441136	Kenroku	Negoita	Senior Staff

Q7

emp_no	first_name	last_name	gender
29250	Radoslaw	Besancenot	M
35565	Chiradeep	Yeung	F
52742	Sashi	Farris	F
64316	Bartek	Pesch	F
84305	Minghong	Kalloufi	F
92983	Masoud	Rosca	M
208735	Niranjan	Narlikar	M
220884	Eberhardt	Schaft	M
223987	Wayne	Lorch	F
228917	Nechama	Bennet	F
233046	Gregory	Cummings	M
250198	Mingdong	Seiwald	M
277581	Siddarth	Anick	M
282495	Ronghao	Glowinski	M
291039	Xumin	Zirintsis	M
417583	Jamaludin	Gischer	M
426700	Naqui	Restivo	F

6. Pembagian Kerja Anggota Kelompok

NIM	**Nama**	**Tugas**
13517024	Ferdian Ifkarsyah	Struktur laporan & program, data studi kasus
13517061	Hafidh Redyanto	Allen 13's query dan hasil, laporan
13517069	Didik Supriadi	Implementasi operasi 4.2.7 & 4.2.9, laporan
13517112	Muhammad Nurdin Husen	Implementasi operasi 4.2.7 & 4.2.8, laporan
13517126	Louis Cahyadi	Implementasi operasi 4.2.1 - 4.2.6, laporan

7. Diskusi dan Kesimpulan

Dengan selesainya tugas ini, kami mendapat beberapa kesimpulan sebagai berikut:

- Model representasi Snodgrass dipilih karena secara sangat natural dan secara langsung dapat diterapkan ke dalam banyak skema relasional yang sudah dipelajari sebelumnya.
- Pemilihan database yang tepat dapat mengurangi *development cost* karena proses *develop* fungsi tertentu menjadi trivial. Dalam kasus kami, ini adalah fungsi FOR PORTION dari MariaDB.
- Porsi waktu desain perlu ditingkatkan agar temporal query yang sudah diimplementasikan dapat masuk menjadi suatu cerita dalam program.