

## Final Report

### โครงการ Linked Out

#### เสนอ

รศ.ดร.ธาราทิพย์ สุวรรณศาสตร์

#### รายชื่อผู้จัดทำ

ชาญชัย	รัตน์ะศิวะกุล	6130115221
ฉั่มทิวต์ถ์	ติระนาทวิทยากุล	6130247421
ธีรภัทร	แก่งศิลาลัย	6130257721
บุญยวัชร	รุจิพิรานันท์	6131027921
พศวัฒน์	แดงใหญ่	6131029121
พศวัต	ไม้เหลื่อง	6131030721
ภูมิพัฒน์	จิรจรัส	6131037121
สรธันย์	กฤตวีรนนท์	6131044521

โครงการนี้เป็นส่วนหนึ่งของรายวิชา 2110322 Database Systems

ภาคการศึกษาต้น ปีการศึกษา 2563

จุฬาลงกรณ์มหาวิทยาลัย

## สารบัญ

1. ความเป็นมาของโครงการ.....	1
2. วัตถุประสงค์ของโครงการ.....	1
3. ฟังก์ชันของระบบ (System Functionalities).....	2
4. ER Diagram / Document Based Schema .....	3
4.2 Document – Based Schema.....	4
5. Schema Design (Referential Integrity): .....	8
6. Normalization .....	9
7. Database Dictionary .....	10
8. Indexing.....	15
9. Stored Procedures .....	15
PROCEDURE FindJobInRange (IN distance double, latitude double, longitude double).....	15
PROCEDURE FindJobInSalaryRange(IN start Integer, end Integer) .....	16
10. Stored Functions.....	19
FUNCTION GetDistance (lat1 double, long1 double, lat2 double, long2 double).....	19
11. Triggers .....	20
12. Referential Integrity Constrains .....	22
13. Execution Path .....	24
14. Complex Query .....	25
15. MongoDB .....	25
16. ภาคผนวก.....	28

## **1. ความเป็นมาของโครงการ**

เนื่องด้วยสถานการณ์การระบาดของไวรัส COVID-19 ในช่วงเวลาที่ผ่านมาที่ก่อให้เกิดผลกระทบในหลาย ๆ ด้าน โดยเฉพาะอย่างยิ่งในด้านเศรษฐกิจ ซึ่งส่งผลให้เกิดการเลิกจ้างพนักงานทั้งพาร์ทไทม์และพนักงานประจำเป็นจำนวนมาก โดยกลุ่มที่ได้รับผลกระทบโดยตรงคือกลุ่มที่ถูกบริษัทเชิญออก และบัณฑิตที่จบการศึกษาในปีการศึกษาที่ผ่านมาที่ไม่สามารถหางานทำได้

ทางคณะผู้จัดทำเล็งเห็นว่า การมีแอปพลิเคชันในการหางาน/หาพนักงาน ที่มีระบบแนะนำงาน/พนักงาน เป็นหนึ่งในทางเลือกที่ดีที่จะเข้ามาแก้ปัญหาการว่างงานในสถานการณ์การระบาดของไวรัส COVID-19 ได้ไม่มากนัก

ด้วยเหตุข้างต้น ทางคณะผู้จัดทำจึงจัดตั้งและพัฒนาเว็บแอปพลิเคชัน Linked-Out ขึ้น โดยเว็บแอปพลิเคชันนี้จะทำหน้าที่เป็นสื่อกลางในการหางาน/พนักงานที่ต้องการ โดยเว็บแอปพลิเคชัน Linked-Out จะมีระบบแนะนำที่พิจารณาจากข้อมูลด้านความสนใจของผู้ที่กำลังหางาน, ความต้องการของบริษัทที่กำลังรับพนักงาน และ สถานที่ที่ผู้ที่กำลังหางานสนใจที่จะไปทำงานในละแวกนั้น ๆ ฯลฯ เพื่อกระตุ้นให้ผู้ที่กำลังหางานได้เจอกับงานที่ตนเองต้องการ และ กระตุ้นให้บริษัทที่กำลังรับสมัครพนักงานได้เจอกับผู้ที่มีคุณสมบัติตามต้องการ ได้รวดเร็วยิ่งขึ้น

## **2. วัตถุประสงค์ของโครงการ**

เป้าหมายของโครงการนี้คือการพัฒนาเว็บแอปพลิเคชันเพื่อเป็นตัวกลางในการแก้ปัญหาการว่างงาน ด้วยการส่งเสริม และอำนวยความสะดวกให้การหางานใหม่หรือการหาพนักงานใหม่เป็นไปได้ง่ายยิ่งขึ้น โดยเฉพาะอย่างยิ่งในสถานการณ์การแพร่ระบาดของไวรัส COVID-19 ซึ่งส่งผลกระทบต่อระบบเศรษฐกิจจนทำให้เกิดการเพิ่มจำนวนการว่างงานในไทยเป็นจำนวนมาก

### 3. ฟังก์ชันของระบบ (System Functionalities)

#### 3.1. ส่วนการแสดงผล

- 1) แสดงผลประกาศจ้างงานตามคำค้นต่าง ๆ ได้ (Keywords / Tags / Location / Minimum Pay Out / งานรายเดือน / งานรายวัน) ได้
- 2) จัดเรียงข้อมูลเพื่อแสดงผลตามวันที่ลงประกาศ / ระยะทางจากสถานที่ที่ระบุ / คะแนนชื่อเสียงได้
- 3) ระบบอาจแนะนำ (Recommend) งานที่น่าสนใจแก่ผู้ใช้งานระบบได้

#### 3.2. ส่วนการจัดการผู้ใช้งาน

- 1) ระบบการสมัครสมาชิก และเปลี่ยนรหัสผ่าน
- 2) ส่วนการจัดการสมาชิก (ยกเลิกผู้ใช้ชั่วคราว / ถาวร / ดูข้อมูลและแก้ไขข้อมูลสมาชิกรายบุคคล / อนุมัติผู้ใช้งาน)
- 3) อาจมีการแนะนำ (Recommend) งานที่น่าสนใจแก่ผู้ใช้งานระบบได้ในอนาคต

#### 3.3. ส่วนการจัดการประกาศหางาน

- 1) เพิ่ม / ลบ / แก้ไข ประกาศหางาน
- 2) สร้างโฆษณาของประกาศหางานนั้น (Boost Post)

#### 3.4. ด้านการติดต่อสื่อสาร

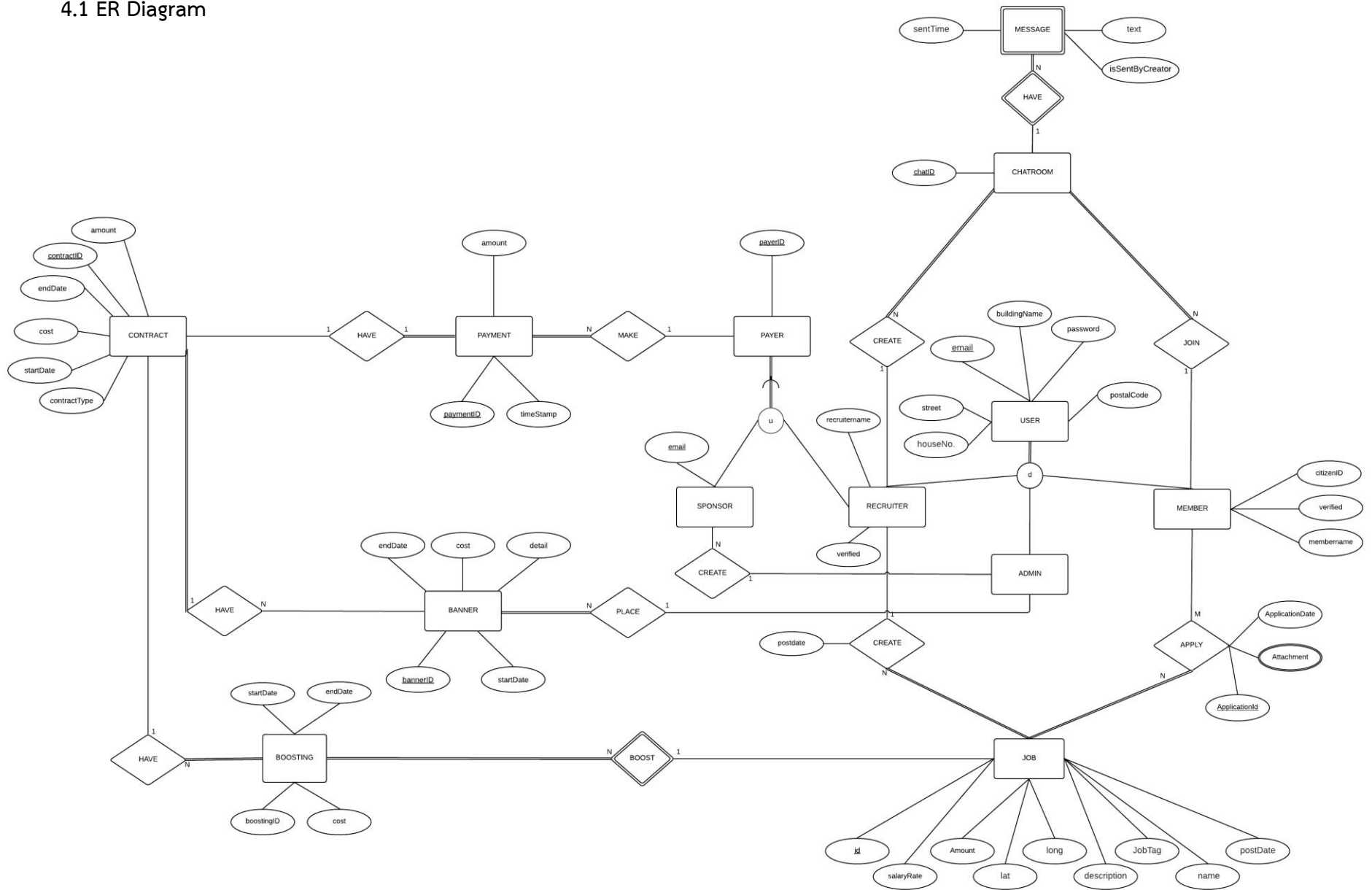
- 1) สร้าง / ตอบกลับ ข้อความส่วนตัวระหว่างผู้ใช้อื่น
- 2) รายงานข้อความที่ไม่พึงประสงค์ถึงผู้ดูแลระบบ

#### 3.5. การดูแลระบบ

- 1) อนุมัติโฆษณา และกำหนดช่วงเวลาของการโฆษณา (Boost Post)

## 4. ER Diagram / Document Based Schema

### 4.1 ER Diagram



## 4.2 Document – Based Schema

### Collection: User {

```
    email: string,  
    password: string,  
    buildingName: string,  
    houseNo: string,  
    street: string,  
    postalCode: string  
}
```

### Collection: Member {

```
    verified: boolean,  
    memberName: string,  
    email: string,  
    password: string,  
    CitizenID:string,  
    buildingName: string,  
    houseNo: string,  
    street: string,  
    district: string,  
    province: string,  
    postalCode: string  
}
```

### Collection: Admin {

```
    email: string,  
    password: string,  
    buildingName: string,  
    houseNo: string,  
    street: string,  
    district: string,  
    province: string,  
    postalCode: string }
```

**Collection: Recruiter {**

```
    verified: Date,  
    name: string,  
    email: string,  
    password: string,  
    buildingName: string,  
    houseNo: string,  
    street: string,  
    district: string,  
    province: string,  
    postalCode: string  
}
```

**Collection: Job {**

```
    id: int not null auto increment,  
    name: string,  
    description: string,  
    amount: integer,  
    lat: double,  
    long: double,  
    jobTag: [string],  
    salaryRate: unsigned long,  
}
```

**Collection: Sponsor {**

```
    email: string,  
}
```

**Collection: Payer {**

```
    payerID: int not null auto increment  
}
```

**Collection: Banner {**

```
    BannerID: int not null auto increment,  
    Detail: string,  
    Cost: float,  
    startDate: Date,  
    endDate: Date,  
}
```

**Collection: Contract {**

```
    contractID: int not null auto increment,  
    cost: float,  
    startDate: Date,  
    endDate: Date,  
    contractType: string,  
    Amount: int  
}
```

**Collection: Payment {**

```
    paymentID: int not null auto increment,  
    amount: float,  
    TimeStamp#: Date  
}
```

**Collection: Chatroom {**

```
    chatID: int not null auto increment  
}
```

**Collection: Message {**

```
    chatID: int not null auto increment,  
    sentTime: Date,  
    text: string  
    isSentByCreator: Boolean  
}
```



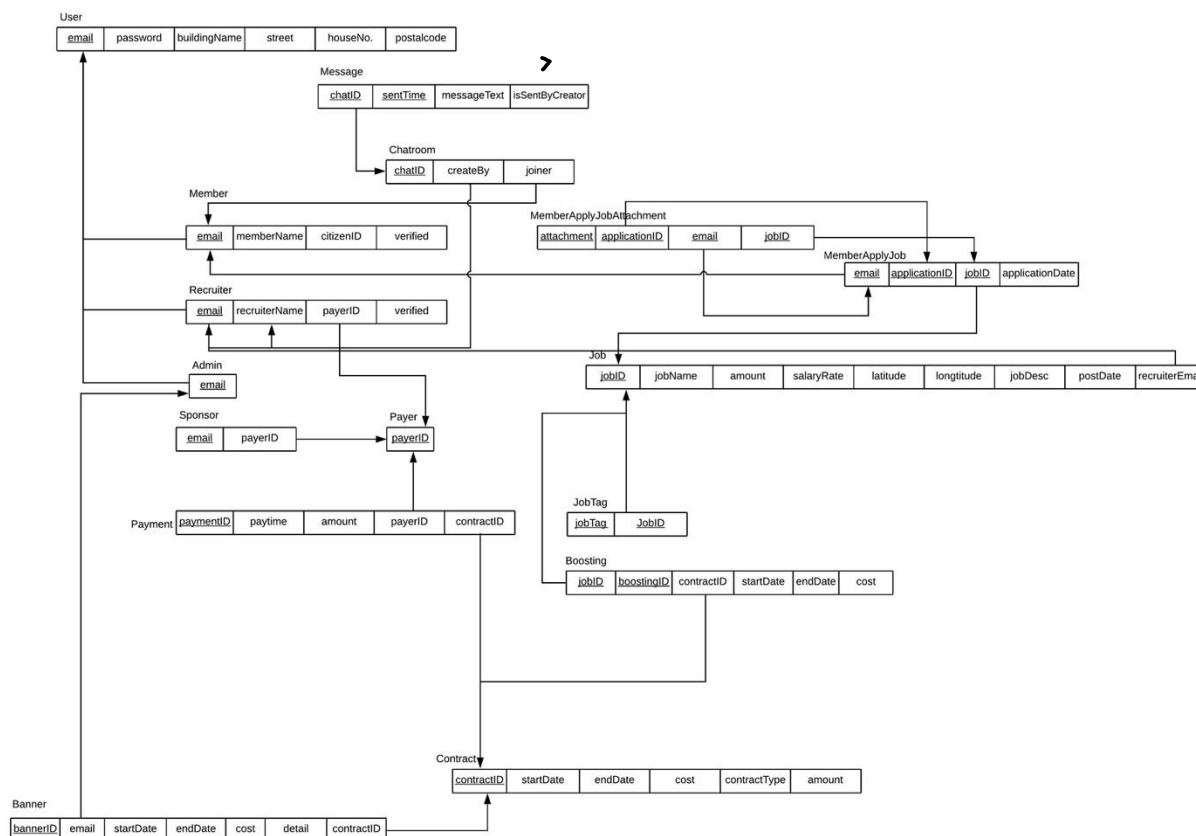
**Collection: Boosting {**

```
    JobID: int,  
    boostingID: int not null auto increment,  
    cost: float,  
    startDate: Date,  
    endDate: Date  
}
```

**Relationship Collection: MemberApplyJob {**

```
    ApplicationId : int not null auto increment,  
    memberEmail: string,  
    jobId: int not null,  
    applicationDate: Date,  
    attachment: [Binary]  
}
```

## 5. Schema Design (Referential Integrity):



จากการวิเคราะห์ Schema ที่ทำ normalize แล้วพบว่า

- Member.email, Recruiter.email, Admin.email REFER ไปยัง User.email (FK)
- Boosting.contractID, Payment.contractID REFER ไปยัง Contract.contractID (FK)
- Recruiter.payerID, Sponsor.payerID Payment.payerID REFER ไปยัง Payer.payerID (FK)
- User.postalCode REFER ไปยัง Postalcode.postalCode (FK)
- Message.chatID REFER ไปยัง Chatroom.chatID (FK)
- Chatroom.createBy REFER ไปยัง Recruiter.email
- MemberApplyJobAttachment.applicationID REFER ไปยัง MemberApplyJob.applicationID
- MemberApplyJobAttachment.email REFER ไปยัง MemberApplyJob.email

## 6. Normalization

จาก Schema ในข้อ 5 นั้น เราได้ทำการ normalize เป็น 3NF แล้ว จากการวิเคราะห์ Schema ที่ทำ normalize แล้วพบว่า

- Member.email, Recruiter.email, Admin.email REFER ไปยัง User.email (FK)
- Boosting.contractID, Payment.contractID REFER ไปยัง Contract.contractID (FK)
- Recruiter.payerID, Sponsor.payerID Payment.payerID REFER ไปยัง Payer.payerID (FK)
- Message.chatID REFER ไปยัง Chatroom.chatID (FK)
- Chatroom.createBy REFER ไปยัง Recruiter.email
- MemberApplyJobAttachment.applicationID REFER ไปยัง MemberApplyJob.applicationID
- MemberApplyJobAttachment.email REFER ไปยัง MemberApplyJob.email

## 7. Database Dictionary

Entity Type Name: User

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>email</u>	string	Email of the User in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No
password	string	password of the User in the system stored in hash	hex or bcrypt	No
BuildingName	string	Building Name which is component of the full address (composite attribute).	UTF8	Yes
Street	string	Street Name which is component of the full address (composite attribute).	UTF8	No
HouseNo	string	House Number which is component of the full address (composite attribute).	number string with/without hyphen, slash.	No
PostalCode	string	Postal code which is component of the full address (composite attribute).	number string	No

Entity Type Name: Message

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
chatID	Int	ID of chat that made from recruiter	positive integer	No
sentTime	Date	Date which message is sent.	since linux epoch til now.	No
messageText	String	Message between recruiter and user.	UTF8	No
isSentByCreator	Boolean	Identify the if the creator of the chatroom sends the message, otherwise the joiner.	True / False	No

Entity Type Name: Chatroom

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>chatID</u>	Int	ID of chat that made from recruiter	positive integer	No
createBy	string	User that create chatroom which is recruiter	UTF8	No
joiner	string	Member who join the chatroom which is member	UTF8	No

## Entity Type Name: Member

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>email</u>	string	Email of the Member in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No
citizenid	string	Citizen ID of the member.	13 digit number string	No
verified	Date	Date which the verified status is approved.	since linux epoch til now.	Yes
memberName	string	Member's name. This must align to real name of the member.	UTF8	No

## Entity Type Name: Recruiter

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>email</u>	string	Email of the Recruiter in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No
verified	Date	Date which the verified status is approved.	since linux epoch til now.	Yes
recruiterName	string	Recruiter Organization name.	UTF8	No
payerID	string	Id for recruiter when make payment to boost their job.	number string	No

## Entity Type Name: Admin

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>email</u>	string	Email of the User in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No

## Entity Type Name: Job

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>jobID</u>	string	Unique identifier of job	number string	No
jobName	string	Job name	UTF8	No
jobDesc	string	Brief detail of the job	UTF8	Yes
amount	Int	Slot of recruiting available for application.	Positive Intgers	No
latitude	double	Coordinate of the workplace	GPS values	No
longitude	double	Coordinate of the workplace	GPS values	No

jobTag	{string}	Type of the job	list of UTF8 string	No
postDate	Date	Date when the job is posted.	since linux epoch til now.	No
recruiterEmail	string	Email of the Recruiter in the system that post the job.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\\.[A-Z]{2,4}\$")	No
salaryRate	unsigned long	Minimum payout	positive number	No

**Entity Type Name:** Sponsor

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
payerID	string	Id for sponsor when make payment to rent banner.	number string	No
<u>email</u>	string	Sponsor's email (used for contact)	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\\.[A-Z]{2,4}\$")	No

**Entity Type Name:** Banner

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>bannerID</u>	string	Unique identifier of contract document.	number string	No
email	string	Email of the Sponsor in the system.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\\.[A-Z]{2,4}\$")	No
cost	unsigned long	Cost for renting the banner.	positive integers	No
detail	string	Detail of the Banner	UTF8	Yes
contractID	string	Id of the banner's contract.	number string	No
StartDate	Date	Date which the banner start.	since linux epoch til now.	No
EndDate	Date	Date which the banner end.	since linux epoch til now.	No

**Entity Type Name:** Boosting

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>BoostingID</u>	string	Unique identifier of boosting.	number string	No
<u>jobID</u>	string	Unique identifier of job	number string	No
contractID	string	Id of the banner's contract.	number string	No
cost	unsigned long	Cost for boosting the job.	positive integers	No
StartDate	Date	Date which the boosting start.	since linux epoch til now.	No

EndDate	Date	Date which the boosting end.	since linux epoch til now.	No
---------	------	------------------------------	----------------------------	----

**Entity Type Name**      **Payment**

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
paymentID	string	Unique identifier of payment	number string	No
paytime	Date	Date which payment occurred	since linux epoch til now.	No
amount	float	Amount of payment	Positive Intgers	No
payerID	string	Id for sponsor or recruiter when make payment.	number string	No
contractID	string	Id of the banner's contract.	number string	No

**Entity Type Name:**      **JobTag**

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>jobTag</u>	{string}	Type of job	list of UTF8 string	No
<u>jobID</u>	Int	Unique identifier of job	number string	No

**Entity Type Name:**      **Payer**

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>payerID</u>	string	Id for sponsor or recruiter when make payment.	number string	No

**Entity Type Name:**      **Contract**

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
contractID	string	Unique identifier of contract.	number string	No
startDate	Date	Date which the contract start.	since linux epoch til now.	No
endDate	Date	Date which the contract end.	since linux epoch til now.	No
cost	float	Amount of payment	Positive Intgers	No
contractType	string	Type of contract (banner or boosting)	UTF8	No
amount	Int	Number of banner or boosting that are in the contract.	Positive intgers	No

Relationship Name: MemberApplyJob

Attribute	Type	Descriptive Name	Valid Values	Allow Nulls
<u>jobID</u>	string	Unique identifier of job	number string	No
<u>applicationID</u>	string	Unqiue identifier of job's application.	number string	No
<u>email</u>	string	Email of the User in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No
applicationDate	Date	Date which applicant applies for the job	since linux epoch til now.	No

Relationship Name: MemberApplyJobAttachment

<u>Attribute</u>	Type	Descriptive Name	Valid Values	Allow Nulls
<u>attachment</u>	{Blob}	array of attachments; such as portfolio, CVs, certificate, resume, etc.	array of length 0 or more, each item is a valid file.	Yes
<u>email</u>	string	Email of the User in the system, uniquely identify the user.	REGEXP("^[A-Z0-9._%~]+@[A-Z0-9.-]+\.[A-Z]{2,4}\$")	No
<u>jobID</u>	string	Unique identifier of job	number string	No
<u>applicationID</u>	string	Unqiue identifier of job's application.	number string	No

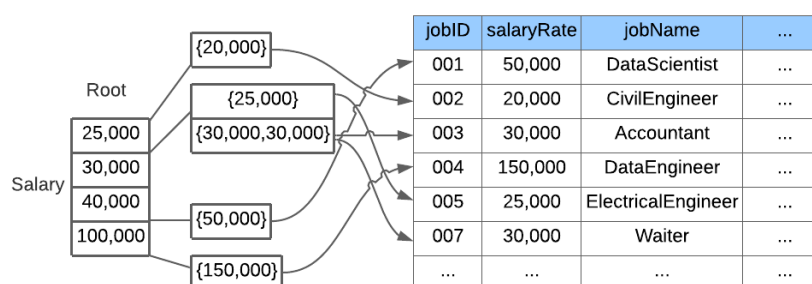


## 8. Indexing

เนื่องจากระบบฐานข้อมูลชุดนี้เป็นระบบฐานข้อมูลสำหรับการหางาน จึงมีความเป็นไปได้สูงที่จะมีจำนวนข้อมูลของงานต่าง ๆ เป็นจำนวนมาก ดังนั้น การจัดการข้อมูลของงานต่าง ๆ อย่างเป็นระบบจึงเป็นสิ่งจำเป็นในระบบฐานข้อมูลชุดนี้

จากที่กล่าวมาข้างต้น พวกเราจึงได้เลือก attribute ที่ชื่อว่า jobname และ salaryRate จาก Job มาเป็นตัวอย่างในการทำ Indexing เพราะว่า salaryRate และ jobName ความเป็นไปได้สูงที่จะถูกใช้เป็นตัวแทนในการค้นหางาน

โดยในการทำ Indexing กลุ่มเราได้เลือกใช้ เป็นแบบ Unclustered Tree Indexing เพราะว่า data structure แบบ Tree สามารถทำ range selection ได้ดี นอกจากนี้กลุ่มเรายังได้เลือกใช้ data entry เป็นแบบ Alternative 3 เพราะว่า Job อาจมี salary rate เท่ากันได้หลาย Job



## 9. Stored Procedures

**PROCEDURE FindJobInRange (IN distance double, latitude double, longitude double)**

ทำหน้าที่ในการแสดงผล Job ที่มีตำแหน่งอยู่ในระยะไม่เกิน distance วัดจาก latitude และ longitude ของ Input เทียบกับ latitude และ longitude ของ Job นั้น

รายละเอียดภายในคำสั่ง

```
CREATE PROCEDURE FindJobInRange(IN d Double, lat1 Double, long1 Double)
BEGIN
    SELECT *
    FROM Job J
    WHERE GetDistance(lat1,long1,J.latitude,J.longitude) <= d
    ORDER BY GetDistance(lat1,long1,J.latitude,J.longitude) DESC;
END$$
```

ตัวอย่างและผลลัพธ์การเรียกใช้งาน

```
mysql> CALL FindJobInRange(15,0,0);
```

jobID	jobName	amount	salaryRate	latitude	longitude	jobDesc	postDate	recruiterEmail
52	Research Associate	13	21448.6	6	9	Realigned radical customer loyalty	2020-04-04 12:46:00	lrenalsom@ed.gov
137	Web Designer II	4	28524.7	6	7	Re-contextualized client-server projection	2020-06-07 23:26:00	spoileqz@google.ru
118	Web Designer I	89	19193.3	5	6	Synergized heuristic superstructure	2020-01-06 08:12:00	aoleaghamq@yahoo.com
13	Research Nurse	6	11485.7	3	6	Sharable eco-centric interface	2020-09-27 01:24:00	hwanknj@seattletimes.com
119	Administrative Officer	62	22424.2	1	4	Phased hybrid installation	2020-08-23 03:32:00	tmowengh@booking.com

5 rows in set (0.08 sec)

Query OK, 0 rows affected (0.12 sec)

## PROCEDURE FindJobInSalaryRange(IN start Integer, end Integer)

ทำหน้าที่แสดง Job ที่ salaryRate มีค่าอยู่ระหว่าง start และ end

รายละเอียดภายในคำสั่ง

```
CREATE PROCEDURE FindJobInSalaryRange(IN start1 Double, end1 Double)
```

```
BEGIN
```

```
SELECT *
```

```
FROM Job
```

```
WHERE salaryRate between start1 and end1;
```

```
END$$
```

ตัวอย่างการเรียกใช้งาน

```
mysql> CALL FindJobInSalaryRange(5000,11000);$$
```

jobID	jobName	amount	salaryRate	latitude	longitude	jobDesc	postDate	recruiterEmail
20	Paralegal	68	10817.1	57	14	Stand-alone client-server help-desk	2020-06-26 01:00:00	tbanishevitznq@sun.com
36	Budget-Accounting Analyst I	94	10352.4	90	53	Function-based interactive challenge	2020-03-26 19:53:00	ndownhamo6@thetimes.co.uk
38	Food Chemist	33	10824.6	45	90	De-engineered user-facing capacity	2020-01-04 13:10:00	dthornewello8@gravatar.com
51	Biostatistician II	18	10590.9	10	51	Adaptive analyzing frame	2020-01-12 22:23:00	harnauducol@cocolog-nifty.com
85	Marketing Manager	93	10702.9	29	87	Public-key composite alliance	2020-04-04 09:11:00	fvarleypj@eventbrite.com
96	Engineer I	82	10875.4	94	90	Vision-oriented bandwidth-monitored time-frame	2020-03-03 09:50:00	dhardinpu@mysql.com
109	Human Resources Assistant III	86	10112.6	40	45	Focused regional knowledge base	2020-01-24 17:44:00	bcartmilly7@homestead.com
110	Librarian	46	10419.3	33	4	Diverse coherent task-force	2020-10-30 08:55:00	bkystonq8@121.jp
121	Engineer III	26	10878.7	31	91	Business-focused maximized capacity	2020-08-28 18:57:00	colomanqj@washington.edu
123	Civil Engineer	56	10702.1	19	69	Function-based background circuit	2020-09-18 04:36:00	bmumbersonq1@360.cn
133	Payment Adjustment Coordinator	31	10435.2	62	33	Extended bifurcated contingency	2020-08-21 13:49:00	pspybyqv@ustream.tv
139	Research Nurse	33	10346.2	96	38	Vision-oriented motivating array	2020-02-17 18:58:00	toldeyr1@prweb.com

PROCEDURE UpdateUserAndMember (

IN i\_email varchar(255),

IN i\_buildingName varchar(50),

IN i\_houseNo varchar(10),

IN i\_street varchar(50),

IN i\_postalCode varchar(10),

IN i\_memberName varchar(255),

IN i\_citizenID varchar(13)

)

ทำหน้าที่ Update ข้อมูลของทั้งตาราง User และ Member (Member อยู่ใน domain ของ User)

รายละเอียดภายในคำสั่ง

CREATE PROCEDURE `UpdateUserAndMember` (

IN i\_email varchar(255),

IN i\_buildingName varchar(50),

IN i\_houseNo varchar(10),

IN i\_street varchar(50),

IN i\_postalCode varchar(10),

IN i\_memberName varchar(255),

IN i\_citizenID varchar(13)

)

BEGIN

DECLARE exit handler for sqlexception

begin

rollback;

resignal;

end;

start transaction;



## 10. Stored Functions

**FUNCTION GetDistance (lat1 double, long1 double, lat2 double, long2 double)**

ทำหน้าที่ Return ระยะทางระหว่างพิกัด 2 จุด

รายละเอียดคำสั่งภายใน

```
CREATE FUNCTION GetDistance(lat1 double, long1 double, lat2 double, long2
double)
    RETURNS double
    DETERMINISTIC
BEGIN
    declare dpi,dlamp double;
    declare x,y double;
    declare meter double;
    set dpi = (lat2-lat1) * pi()/180;
    set dlamp = (long2-long1) * pi()/180;
    set x = dlamp * cos((lat1+lat2)/2 * pi()/180);
    set y = dpi;
    set meter = 6371000*sqrt(x*x+y*y);
    RETURN meter;
END$$
```

ตัวอย่างการเรียกใช้

```
mysql> select Jobname, GetDistance(@chulalat,@chulalong,latitude,longtitude) as distance from Job
→ where GetDistance(@chulalat,@chulalong,latitude,longtitude) < 3000
→ order by GetDistance(@chulalat,@chulalong,latitude,longtitude);
```

Jobname	distance
Dental Hygienist	730.4310889132731
Assistant Media Planner	841.4478248843176
GIS Technical Architect	1268.1662564798373
Internal Auditor	1442.3431475574093
Programmer Analyst III	1565.1833250435543
Safety Technician II	1884.980918428296
VP Product Management	2215.2834977752486
Structural Engineer	2249.348407810449
Physical Therapy Assistant	2376.681611647702
Financial Advisor	2581.233807472943
Project Manager	2606.6263286491762
VP Sales	2637.2165856827414
Senior Financial Analyst	2640.3929338068356
Nurse	2695.44859401957
Nurse Practitioner	2803.125299207477
Teacher	2934.4051615326457

```
16 rows in set (0.08 sec)
```

## FUNCTION GetAnnualSalary (salary DOUBLE)

ทำหน้าที่คืนรายได้รายปีที่ได้จากเงินเดือน

รายละเอียดคำสั่งภายใน

```
FUNCTION GetAnnualSalary(salary DOUBLE)
```

```
    RETURNS DOUBLE
```

```
    DETERMINISTIC
```

```
BEGIN
```

```
    RETURN salary*12;
```

```
END$$
```

ตัวอย่างการใช้งาน

```
mysql> SELECT jobName, salaryRate, GetAnnualSalary(salaryRate) As annualSalary
-> FROM Job
-> WHERE GetAnnualSalary(salaryRate)>400000
-> ORDER BY GetAnnualSalary(salaryRate) DESC;
```

jobName	salaryRate	annualSalary
Selling Asses	100000	1200000

1 row in set (0.07 sec)

## 11. Triggers

### TRIGGER ActivateBoosting

เมื่อมีการเพิ่มข้อมูล Payment เข้ามาในระบบ แสดงว่าจะเกิดการ Boosting ขึ้น ถ้า Contract เป็นของ Boosting

รายละเอียดคำสั่งภายใน

```
CREATE TRIGGER ActivateBoosting
```

```
AFTER INSERT ON Payment FOR EACH ROW
```

```
BEGIN
```

```
    IF (SELECT C.contractType = 'Boosting'
```

```
        FROM Contract C WHERE new.contractID = C.contractID) THEN
```

```
        INSERT INTO
```

```
        Boosting(jobID,boostingID,cost,startDate,endDate,contractID)
```

```
        VALUES((SELECT J.jobID FROM Job J, Recruiter R
```

```

WHERE new.payerID = R.payerID AND R.email =
J.recruiterEmail),new.contractID,
new.amount,NOW(),NOW()+INTERVAL 30 DAY,new.contractID);
END IF;
END

```

ตัวอย่างการเรียกใช้งาน

```

mysql> INSERT INTO Payment(paytime, amount, payerID, contractID) VALUES(NOW(),10000,1,6);
Query OK, 1 row affected (0.06 sec)

mysql> select * from Boosting;

```

jobID	boostingID	cost	startDate	endDate	contractID
1	6	10000	2020-11-22 14:53:28	2020-12-22 14:53:28	6

```

1 row in set (0.09 sec)

mysql> _

```

### TRIGGER CreateChatroom

เมื่อ Recruiter ทำการ Approve ให้กับใบสมัครงาน ระบบจะสร้างห้องสนทนาให้ Recruiter และ Member ได้มาสนทนากัน โดย Trigger นี้จะทำหน้าที่ Insert ข้อมูลลงใน Chatroom เมื่อมีการ Update ค่า approved=1 ของข้อมูลในตาราง MemberApplyJob

รายละเอียดของคำสั่ง

```

CREATE TRIGGER CreateChatroom
AFTER UPDATE ON MemberApplyJob FOR EACH ROW
BEGIN
    IF new.approved = 1 THEN
        INSERT INTO Chatroom(createBy, joiner)
        VALUES((SELECT J.recruiterEmail FROM Job J WHERE J.jobID =
old.jobID) ,old.email);
    END IF;
END

```

## ตัวอย่างการเรียกใช้งาน

```
mysql> UPDATE MemberApplyJob SET approved = 1 WHERE applicationID=1;
-> $$
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> DELIMITER ;
mysql> select * from MemberApplyJob;
```

applicationID	email	jobID	applicationDate	approved
1	zkynvinly@nhs.uk	1	2020-02-22 22:11:00	1
2	zsavilljb@ebay.com	1	2020-02-22 22:11:00	0
3	zstanney17@jugem.jp	1	2020-02-22 22:11:00	0

```
3 rows in set (0.07 sec)

mysql> select * from Chatroom;
```

chatID	createBy	joiner
1	ceneasn7@paginegialle.it	zkynvinly@nhs.uk

```
1 row in set (0.07 sec)
```

## 12. Referential Integrity Constrains

1. Member มี primary key เป็น email เมื่อใส่ email ที่มีอยู่ในระบบอยู่แล้ว จึงเกิด Error ขึ้น

```
mysql> select * from Member
-> WHERE email = 'zstanney17@jugem.jp';
```

email	memberName	citizenID	verified
zstanney17@jugem.jp	Osgood Menier	1073101747899	2020-02-22 22:11:00

```
1 row in set (0.06 sec)

mysql> INSERT INTO Member VALUES ('zstanney17@jugem.jp', 'Sorathan Boom', '1073101747898', '2020-02-22 22:11:00');
ERROR 1062 (23000): Duplicate entry 'zstanney17@jugem.jp' for key 'PRIMARY'
```



2. เนื่องจาก Member มีการใช้ email ซึ่งเป็น Foreign Key ที่นำมาจากตาราง User เมื่อลบข้อมูล ใน User ออก ข้อมูลใน member ที่มี email เดียวกันกับข้อมูลนั้น จะถูกลบออกด้วย

```
mysql> select * from User Where email= 'tkangsilalai@gmail.com';
```

email	userPassword	buildingName	houseNo	street	postalCode
tkangsilalai@gmail.com	12345678	BaanFYO	129	Rama I	20140

```
1 row in set (0.07 sec)
```

```
mysql> select * from Member Where email= 'tkangsilalai@gmail.com';
```

email	memberName	citizenID	verified
tkangsilalai@gmail.com	Theerapat Kangsilalai	1200101850555	2020-02-22 22:11:00

```
1 row in set (0.06 sec)
```

```
mysql> DELETE FROM User WHERE email='tkangsilalai@gmail.com';
```

Query OK, 1 row affected (0.06 sec)

```
mysql> select * from User Where email= 'tkangsilalai@gmail.com';
```

Empty set (0.06 sec)

```
mysql> select * from Member Where email= 'tkangsilalai@gmail.com';
```

Empty set (0.08 sec)

3. เนื่องจากตาราง Chatroom มีการใช้ Email จากตาราง Recruiter ซึ่งเมื่อมีการลบ Recruiter ค่าในตาราง Chatroom จะถูกลบออกไปด้วย

```
mysql> select * from Chatroom;
```

chatID	createBy	joiner
1	ceneasn7@paginegialle.it	zkynvinly@nhs.uk

```
1 row in set (0.10 sec)
```

```
mysql> DELETE FROM Recruiter WHERE email='ceneasn7@paginegialle.it';
```

Query OK, 1 row affected (0.08 sec)

```
mysql> select * from Chatriim;
```

ERROR 1146 (42S02): Table 'LinkedOut.Chatriim' doesn't exist

```
mysql> select * from Chatroom;
```

Empty set (0.11 sec)

### 13. Execution Path

#### Execution path 1

```

SELECT R.recruiterName,SUM(B.cost) AS "Money Wasted"
FROM Boosting B, Job J, Recruiter R
WHERE J.jobID=B.jobID AND J.recruiterEmail=R.email
GROUP BY R.recruiterName
ORDER BY SUM(B.cost) DESC
LIMIT 10;

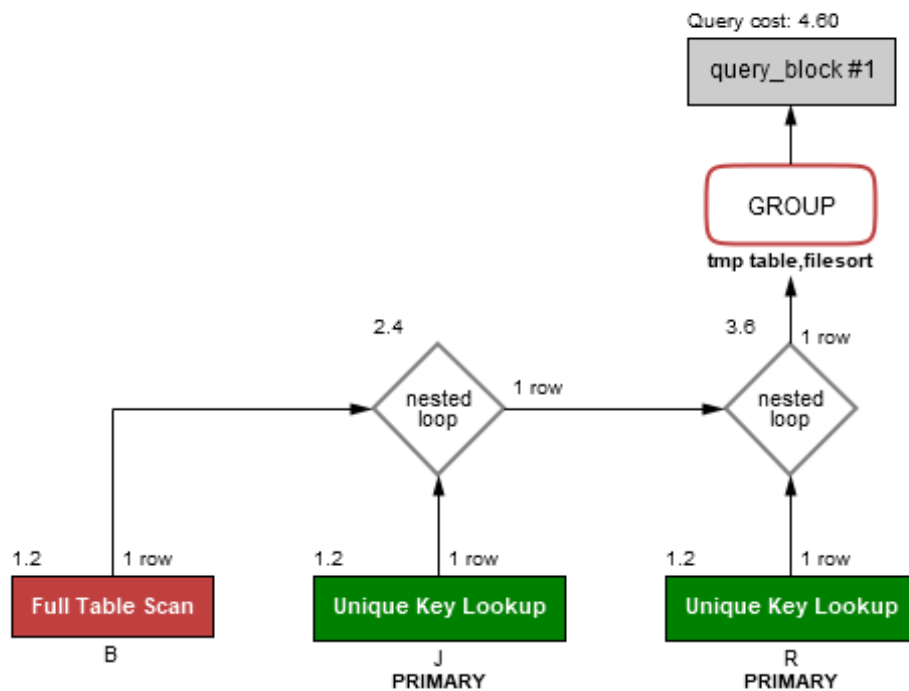
```

#### Execution path 2

```

SELECT recruiterName,SUM(cost)
FROM Boosting B join Job J ON J.jobID=B.jobID
join Recruiter R ON J.recruiterEmail=R.email
group by R.recruiterName
ORDER BY SUM(B.cost) DESC
LIMIT 10;

```



## 14. Complex Queries

- Query สำหรับหางานที่มีผู้สมัครมากที่สุด 10 อันดับแรก ที่มีระยะอยู่ไกลจากระยะของผู้ใช้ (ในที่นี้คือ 13.717,100.4493) และมีรายได้รายปีมากกว่า 100,000 บาทต่อปี

```

SELECT
    jobName AS "Name of the job",
    GetAnnualSalary(salaryRate) AS "Annual Salary",
    count(applicationID) AS "Total number of applications",
    GetDistance(13.717,100.4493,latitude,longitude)/1000 AS "Distance from your
location(km)",
    U.email AS "Contract of the recruiter"
FROM MemberApplyJob M
    NATURAL JOIN Job J
    INNER JOIN Recruiter R ON R.email = J.recruiterEmail
    INNER JOIN User U ON U.email = R.email
WHERE
    GetDistance(13.717,100.4493,latitude,longitude) <= 20000
    AND GetAnnualSalary(salaryRate) >= 100000
GROUP BY jobID
ORDER BY count(applicationID) desc
LIMIT 10;

```

ตัวอย่างการรัน

```
mysql> SELECT
-> jobName AS "Name of the job",
-> GetAnnualSalary(salaryRate) AS "Annual Salary",
-> count(applicationID) AS "Total number of applications",
-> GetDistance(13.717,100.4493,latitude,longitude)/1000 AS "Distance from your location(km)",
-> U.email AS "Contract of the recruiter"
-> FROM MemberApplyJob M
-> NATURAL JOIN Job J
-> INNER JOIN Recruiter R ON R.email = J.recruiterEmail
-> INNER JOIN User U ON U.email = R.email
-> WHERE
-> GetDistance(13.717,100.4493,latitude,longitude) <= 20000
-> AND GetAnnualSalary(salaryRate) >= 100000
-> GROUP BY jobID
-> ORDER BY count(applicationID) desc
-> LIMIT 10;
```

Name of the job	Annual Salary	Total number of applications	Distance from your location(km)	Contract of the recruiter
Business Systems Development Analyst	254223.59999999998	18	1.015699192464075	tnequestnd@nps.gov
Environmental Tech	135283.2	15	16.81086661449863	rendersonnf@e-recht24.de
Analog Circuit Design manager	320856	14	15.590323932757036	aitzkovicini@drupal.org
Structural Engineer	338059.19999999995	10	17.5603583451517	lbiddulphn9@studiopress.com
Nurse	340933.19999999995	9	9.930506603258756	sramelna@abc.net.au
Business Systems Development Analyst	160170	9	18.41801952491581	atrousdalene@360.cn
Director of Sales	295303.19999999995	7	5.373832729244575	bmcilriachn8@creativecommons.org
Assistant Media Planner	136279.2	4	7.250811896909274	cravenscraftng@ehow.com

8 rows in set (0.09 sec)

- แสดงผู้ที่จ่ายเงินซื้อ Boosting มากที่สุดในระบบ 5 อันดับแรก

```

SELECT recruiterName
FROM Recruiter natural join Payment natural join Contract
WHERE contractType = 'Boosting'
GROUP by recruiterName
ORDER BY sum(cost*amount) DESC
limit 5;

```

- แสดง Job ที่มีผู้สมัครมากที่สุด 10 อันดับแรก ในระยะเวลา 90 วัน

```

select jobName,jobDesc,count(jobID) as Applied
from MemberApplyJob natural join Job
where applicationDate >= date_add(curdate(),interval -90 DAY)
group by jobID
order by Applied desc;

```

ตัวอย่างการรัน

```

mysql> select jobName,jobDesc,count(jobID) as Applied
  -> from MemberApplyJob natural join Job
  -> where applicationDate ≥ date_add(curdate(),interval -90 DAY)
  -> group by jobID
  -> order by Applied desc
  -> limit 10;

```

jobName	jobDesc	Applied
Analog Circuit Design manager	Cross-platform background complexity	5
Structural Engineer	Versatile empowering monitoring	4
Business Systems Development Analyst	Face to face actuating concept	2
Business Systems Development Analyst	Synergized intangible neural-net	2
Assistant Media Planner	Switchable full-range application	2
Desktop Support Technician	Automated regional local area network	2
Structural Engineer	Extended upward-trending success	2
Nurse	Fully-configurable client-driven product	1
Paralegal	Cloned zero tolerance standardization	1
Environmental Tech	Re-engineered leading edge toolset	1

10 rows in set (0.03 sec)

## 15. MongoDB

ใช้ในการค้นหาคำค้นของงานต่าง ๆ ทำ Tag ที่ Recruiter ติดไว้ในระบบ

```

db.JobTag.aggregate([
  {$match:{$text: { $search: "service"}}},
  {$lookup:{
    from: "Job",
    localField: "jobID",

```

```

        foreignField: "jobID",
        as: "jobbo" }},
    { $unwind: "$jobbo" },
    { $project: { "_id": 0, "jobName": "$jobbo.jobName" } }
  ] ).pretty()

```

```

{ "jobName" : "Environmental Tech" }
{ "jobName" : "Mechanical Systems Engineer" }
{ "jobName" : "Product Engineer" }
{ "jobName" : "Cost Accountant" }
{ "jobName" : "Legal Assistant" }
{ "jobName" : "Research Nurse" }
{ "jobName" : "Analog Circuit Design manager" }
{ "jobName" : "Mechanical Systems Engineer" }
{ "jobName" : "Safety Technician IV" }
{ "jobName" : "Safety Technician III" }
{ "jobName" : "Developer III" }
{ "jobName" : "Nuclear Power Engineer" }
{ "jobName" : "Food Chemist" }
{ "jobName" : "Cost Accountant" }
{ "jobName" : "Executive Secretary" }
{ "jobName" : "Paralegal" }
{ "jobName" : "Mechanical Systems Engineer" }
{ "jobName" : "Executive Secretary" }
{ "jobName" : "Nurse Practitioner" }
{ "jobName" : "Information Systems Manager" }

```

ภาคผนวก

## LinkedOut Demo Application

The screenshot shows the 'Find Job' section of the LinkedOut Demo Application. At the top, there is a dark blue navigation bar with the following links: 'Find Job' (highlighted in blue), 'Find job by tag (Mongo)', 'Member Registration', and 'Chatroom'. Below the navigation bar, the 'Find Job' section is titled. It features two tabs: 'by salary range' (selected) and 'by geological range'. Under the 'by salary range' tab, there is a 'Salary Range' section with two input fields: the first contains '0' and the second contains '10000', separated by a minus sign. Below these fields is a 'Submit' button. Under the 'Result' section, there is a table with the following headers: 'jobID', 'jobName', 'jobDesc', 'amount', 'latitude', 'longitude', 'postDate', 'recruiterEmail', and 'salaryRate'. The table is currently empty, and a 'No Data' message with a folder icon is displayed below the table headers.

เป็น Application บางส่วนซึ่งมีไว้เพื่อการนำเสนอโครงการงาน มีการใช้ INSERT, DELETE, UPDATE, และ QUERY ซึ่งมีการเรียกใช้ referential integrity constraints, triggers, stored-procedure, และ transaction รวมทั้งมีการใช้ MongoDB ปัจจุบันโฮสต์ไว้ที่ <http://35.247.180.41:3000/> (อาจมีการนำออกหลังจบภาคการศึกษา)

## Demo Application Hierarchy

\* /

|

| - Find Job

    | - by salary range

        | | - PROCEDURE FindJobInSalaryRange

        | | - Select 1 Table

        |

    | - by geological range

        | - PROCEDURE FindJobInRange

            | - Select 1 Table

            | - FUNCTION GetDistance

\* /mongo

|

| - Find Job (Mongo)

    | - by job tag

        | - Select 1 Collection w/ multi-value

            attribute: `jobTags`

\* /register

|

| - Member Registration

    | | - TRANSACTION

        | | - Insert 2 Tables

        |

| - Get User by email

    | | - Select 1 Table

    |

| - Get Member by email



- | |- Select 1 Table
- |
- | - Update Member
  - | - PROCEDURE updateUserAndMember
  - | - TRANSACTION
  - | - Update 2 Tables

\* /chatroom

- |
- | - Show all Chatrooms
  - | |- Select 1 Table
  - |
- | - Show Messages
  - | |- Select 2 Tables JOINED
  - |
- | - Create Chatroom
  - | |- Insert 1 Table
  - |
- | - Create Message
  - | |- Insert 1 Table
  - |
- | - Delete Chatroom
  - | |- Delete 1 Table CASCADE 1 Table
  - |
- | - RecruiterApproveApplication
  - | - Member Apply Job
    - | |- Insert 1 Table
    - |
  - | - Get JobApplication
    - | |- Select 1 Table

|

| - Recruiter Approve

| - Update 1 Table

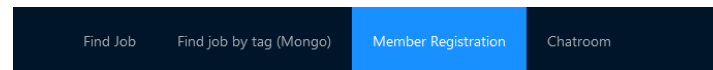
| - TRIGGER/AFTER: CreateChatroom

| - Select 1 Table

| - Insert 1 Table

## INSERT 2 Tables with Transaction

### Member Registration



#### Member Registration

Please fill in the form.

Email:

Password:

Building name:

House number:

Street:

Postal code:

Member name:

Citizen ID:

```
export const registerMember = async (ctx) => {
  console.log(ctx.request.body)
  const { email, password, name,
    citizenID, houseNo, street,
    postalCode, buildingName } = ctx.request.body

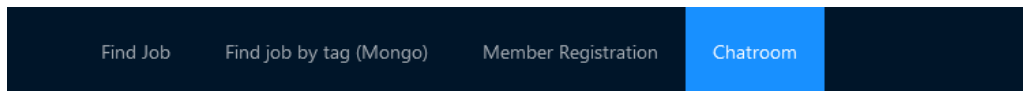
  const conn = await ctx.state.dbPool.getConnection()
  await conn.query('START TRANSACTION')
  try {
    await conn.query(
      `INSERT INTO User(email,userPassword,buildingName,houseNo,street,postalCode)
      VALUES (?, ?, ?, ?, ?, ?)`,
      [email, password, buildingName, houseNo, street, postalCode]
    )
  } catch (error) {
    await conn.query('ROLLBACK')
    console.log(error)
    ctx.status = HttpStatus.BAD_REQUEST
    return
  }

  try {
    await conn.query(
      `INSERT INTO \Member\ (email, memberName, citizenID, verified)
      VALUES (?, ?, ?, ?)`,
      [email, name, citizenID, null]
    )
  } catch (error) {
    await conn.query('ROLLBACK')
    console.log(error)
    ctx.status = HttpStatus.BAD_REQUEST
    return
  }

  await conn.query('COMMIT')
  ctx.status = HttpStatus.OK
}
```

## DELETE 1 Table Cascade 1 Table

### Delete Chatroom



#### Chatroom

Show all Chatrooms / Show Messages / Create Chatroom / Create Message /  
Delete Chatroom / RecruiterApproveApplication

##### Delete Chatroom

chatID

Fire

```
export const deleteChatroom = async (ctx) => {  
  const { chatID } = ctx.request.body  
  const [rows, fields] = await ctx.state.dbPool.query("DELETE FROM Chatroom WHERE chatID=?", [chatID])  
  ctx.status = HttpStatus.OK  
  ctx.body = rows  
}
```

Table Name: <input type="text" value="Message"/>		Schema: <b>LinkedOut</b>	
Charset/Collation: <input type="text" value="utf8"/>	<input type="text" value="utf8_bin"/>	Engine: <input type="text" value="InnoDB"/>	
Comments: <input type="text"/>			
Foreign Key Name	Referenced Table	Column	Referenced Column
fk_message_chatID	'LinkedOut'. 'Chatroom'	<input checked="" type="checkbox"/> chatID	chatID
		<input type="checkbox"/> sentTime	
		<input type="checkbox"/> messageText	
		<input type="checkbox"/> isSentByCreator	
		Foreign Key Options	
		On Update: <input type="text" value="RESTRICT"/>	
		On Delete: <input type="text" value="CASCADE"/>	
<input type="checkbox"/> Skip in SQL generation			

## UPDATE 1 Table Trigger Select 1 Table and Insert 1 Table

### Recruiter Approve JobApplication

#### Recruiter Approve

applicationID:

Fire

```
export const approveApplication = async (ctx) => {
  console.log(ctx.request.body)
  const { applicationID, isApprove } = ctx.request.body

  const result = await ctx.state.dbPool.query(
    `UPDATE MemberApplyJob SET approved = ? WHERE applicationID = ?`,
    [isApprove, applicationID])

  ctx.status = HttpStatus.OK
  ctx.body = result
}
```



Table Name: MemberApplyJob

Schema: LinkedOut

Charset/Collation: utf8 utf8\_bin

Engine: InnoDB

Comments:

BEFORE INSERT  
AFTER INSERT  
BEFORE UPDATE  
▼ AFTER UPDATE  
CreateChatroom  
BEFORE DELETE  
AFTER DELETE

```
1 CREATE DEFINER='admin'@'%' TRIGGER `CreateChatroom` AFTER UPDATE ON `MemberApplyJob` FOR EACH ROW BEGIN
2 IF new.approved = 1 THEN
3 INSERT INTO Chatroom(createBy, joiner)
4 VALUES((SELECT J.recruiterEmail FROM Job J WHERE J.jobID = old.jobID),old.email);
5 END IF;
6 END
```

## UPDATE 2 Tables in PROCEDURE with TRANSACTION

### Update Member

Update Member

Email:

Building name:

House number:

Street:

Postal code:

Member name:

Citizen ID:

Submit

```
export const updateMember = async (ctx) => {
  console.log(ctx.request.body)
  const { email, password, name,
    citizenID, houseNo, street,
    postalCode, buildingName} = ctx.request.body

  const [rows, fields] = await ctx.state.dbPool.query(
    'CALL UpdateUserAndMember(?,?,?,?,?,?,?)'
    , [email, buildingName, houseNo, street, postalCode, name, citizenID]);
  ctx.status = HttpStatus.OK
  ctx.body = rows
}
```

```
CREATE PROCEDURE `UpdateUserAndMember`(
  IN i_email varchar(255),
  IN i_buildingName varchar(50),
  IN i_houseNo varchar(10),
  IN i_street varchar(50),
  IN i_postalCode varchar(10),
  IN i_memberName varchar(255),
  IN i_citizenID varchar(13)
)
BEGIN
  DECLARE exit handler for sqlexception
  begin
    rollback;
    resignal;
  end;

  start transaction;

  update `User` set buildingName=i_buildingName, houseNo=i_houseNo, street=i_street, postalCode=i_postalCode
  where email=i_email;

  update `Member` set memberName=i_memberName,citizenID=i_citizenID where email=i_email;

  commit;

END
```

## QUERY 2 Tables JOINED Together

### Show All Messages

#### Chatroom

Show all Chatrooms / Show Messages / Create Chatroom / Create Message

Delete Chatroom / RecruiterApproveApplication

#### Show Messages

chatID

Fetch

chatID	sentTime	messageText	sentBy
6	2020-12-11T17:06:33.000Z	A1	abenbownn@smh.com.au
6	2020-12-11T17:06:38.000Z	A2	abenbownn@smh.com.au
6	2020-12-11T17:06:43.000Z	B1	aalberti4e@census.gov
6	2020-12-11T17:06:48.000Z	A3	abenbownn@smh.com.au

< 1 >

```
export const getMessage = async (ctx) => {
  const { chatID } = ctx.request.query
  const [rows, fields] = await ctx.state.dbPool.query(
    `SELECT
      M.chatID,
      M.sentTime,
      M.messageText,
      IF(M.isSentByCreator, C.createBy, C.joiner) as sentBy
    FROM Message M
    LEFT JOIN Chatroom C
    ON M.chatID = C.chatID
    WHERE M.chatID = ?`,
    [chatID]
  )


  ctx.status = HttpStatus.OK
  ctx.body = rows
}
```

## SELECT 1 Collection with multi-valued attribute

### Find Job by Tags

**Find Job By Tag (Mongo)**

jobTag:

jobID	jobName
 No Data	

```
export const findJobMongo = async (ctx) => {
  console.log(ctx.request.body)

  const {jobTag} = ctx.request.body

  const res = await ctx.state.jobsWithTags.aggregate([
    { $unwind: '$jobTags' },
    {
      $replaceRoot:{
        newRoot:{
          $mergeObjects:
            [
              { _id: "$_id", jobID: '$jobID', jobName: '$jobName' },
              "$jobTags"
            ]
        }
      },
    },
    { $match: { 'jobTag': jobTag } },
    {
      $group: {
        _id: "$_id",
        "jobID": { $first: "$jobID" },
        "jobName": { $first: "$jobName" }
      },
    },
    { $project: { _id:0, jobID:1, jobName:1 } },
    { $sort: { jobID:1 } }
  ])

  ctx.status = HttpStatus.OK;
  ctx.body = res
}
```