



Data Engineering

Relational Database & Non-Relational Database

Taught by Pichaya Tandayya

Database vs File

- Why do we need a database?

DATABASE VS FILE

COMPARISON TABLE



vs

โครงสร้าง

มีโครงสร้างที่ชัดเจน



โครงสร้าง

ไม่มีโครงสร้างที่แน่นอน

ความปลอดภัย

มีระบบความปลอดภัยที่มั่นคง



ความปลอดภัย

ความปลอดภัยต่ำกว่าฐานข้อมูล

ประสิทธิภาพ

การค้นหาและจัดการข้อมูลเร็วและมีประสิทธิภาพ



ประสิทธิภาพ

การค้นหาและจัดการข้อมูลช้ากว่าฐานข้อมูล

ความซ้ำซ้อนของข้อมูล

มักมีการควบคุมและลดความซ้ำซ้อนในข้อมูล



ความซ้ำซ้อนของข้อมูล

ข้อมูลอาจมีการซ้ำซ้อนซึ่งอาจทำให้ข้อมูลไม่แน่นอน

Terminology

- Unstructured Data: Data lacking a predefined format, such as videos, emails, or social media posts
- Structured Data: Data organized in rows and columns, typically stored in relational databases
- Data Lake: Centralized storage for raw, unstructured, and semi-structured data at any scale
- Data Warehouse: A structured system for organizing, analyzing, and querying large volumes of data

Structured Data

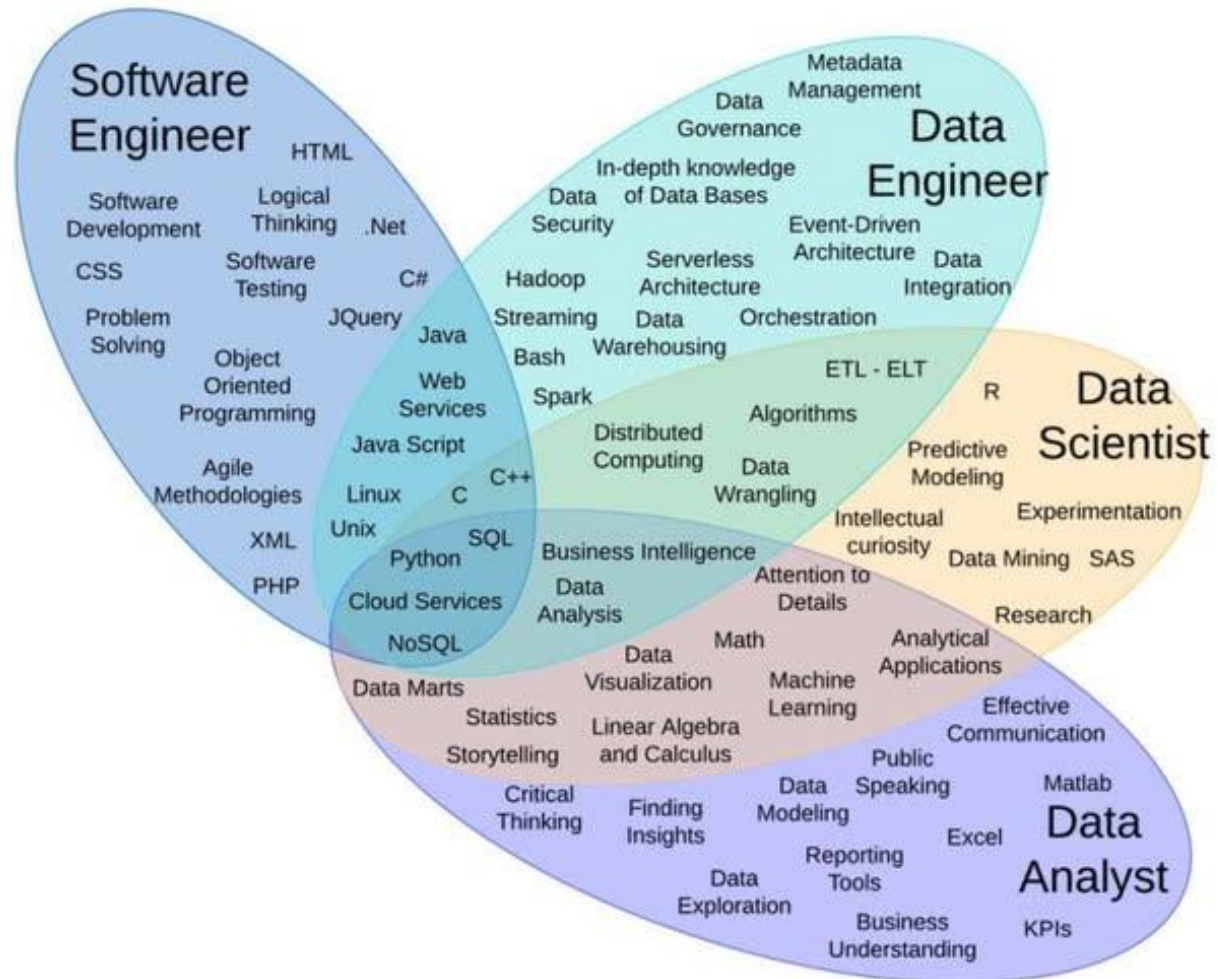
- มักเป็นข้อมูลเชิงตารางนิยมจัดเก็บไว้ใน database ประเภท Relational Database

id	name	department_id	created_at
1	Alice	25	2024-01-01 10:00
2	Bob	30	2024-01-02 11:15

Data Science & Database System

- ข้อมูลดิบที่พบบ่อยในงาน Data Science จะมีลักษณะเป็น Text File เช่น CSV, JSON หรือ Binary Format เฉพาะทางเช่น
- Excel XLS, HDFS, etc.
- การอ่านข้อมูลจำนวนมากจากไฟล์ทุกครั้งที่ต้องทำการประมวลผลข้อมูลนั้นจะทำให้ประสิทธิภาพของระบบ หรือ โมเดลทำงานได้ไม่มีประสิทธิภาพ
- การจัดเก็บข้อมูลในรูปแบบที่ สร้าง เพิ่มข้อมูล แก้ไข หรือ ลบข้อมูลได้อย่างรวดเร็ว มีประสิทธิภาพ เป็นเรื่องสำคัญ

SE vs DE vs DS vs DA

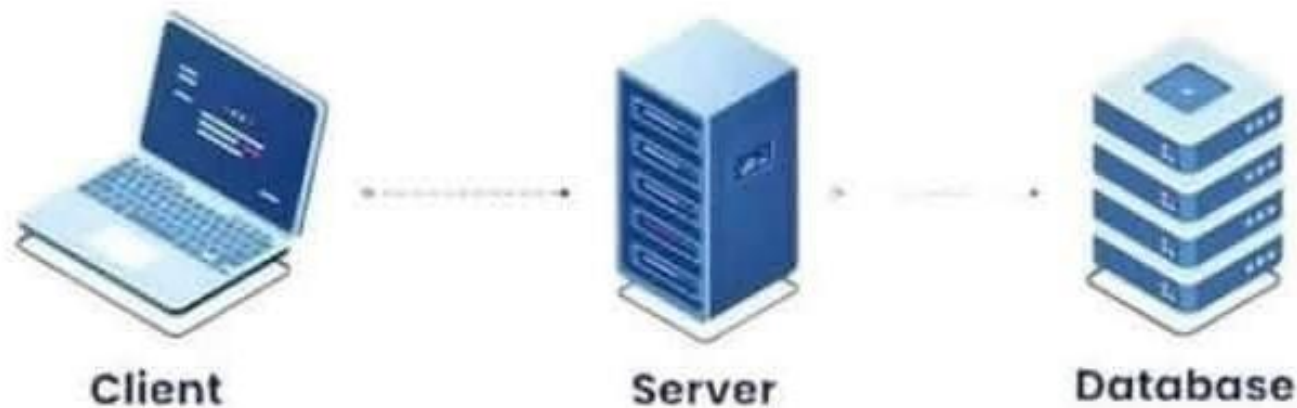


Workflow: Data Science & Data Engineering

- วิเคราะห์และประมวลผล Raw Data จาก CSV, JSON, TXT หรือ อื่นๆ
- ออกแบบวิธีการเก็บข้อมูล
 - Relational Database:
 - ออกแบบ Entity Relationship Diagram
 - Database Normalization
 - Non-Relational Database:
 - ออกแบบรูปแบบเอกสารสำหรับเก็บข้อมูล (นิยมใช้รูปแบบ JSON)
- นำเข้าข้อมูลเข้าสู่ฐานข้อมูล
- ประมวลผล พัฒนาแอปพลิเคชัน หรือ โมเดลการทำนายต่าง ๆ จากข้อมูลที่น่าเข้าแล้ว

Database Management System (DBMS)

- A system software used to manage the organization by storage, access, modification and integrity of data in a structured database



Why do we need DBMS?

- Ease of access
- Hassle-free file management
- Storage and management of databases
- Avoiding redundancy (by data normalization)

Different Types of DBMS

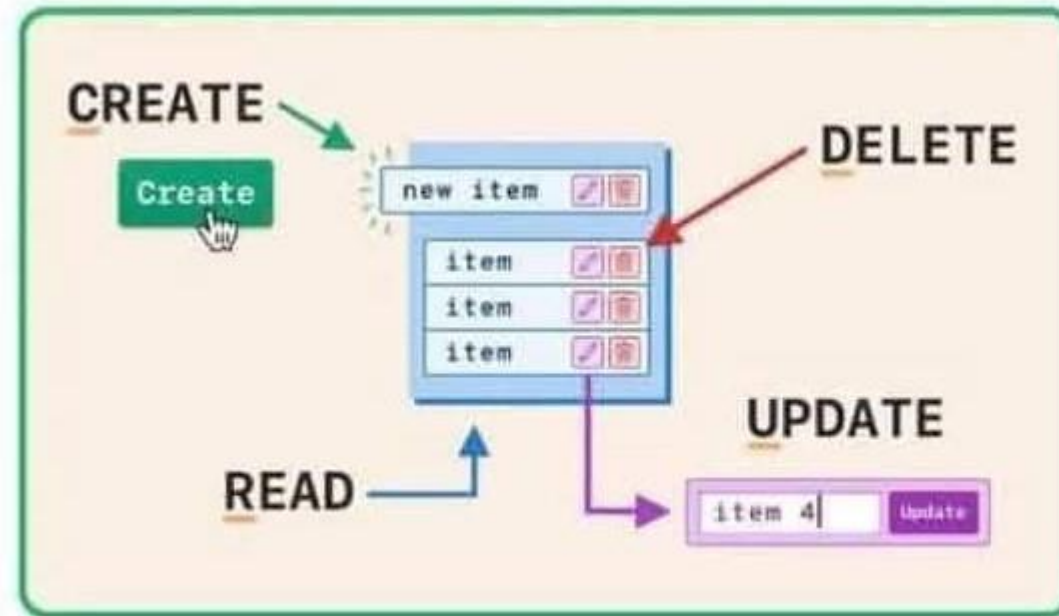
- Centralized database
- Distributed database
- Relational database
- NoSQL database
- Cloud database
- Object-oriented database
- Hierarchical database
- Personal database

Examples of DBMS Applications

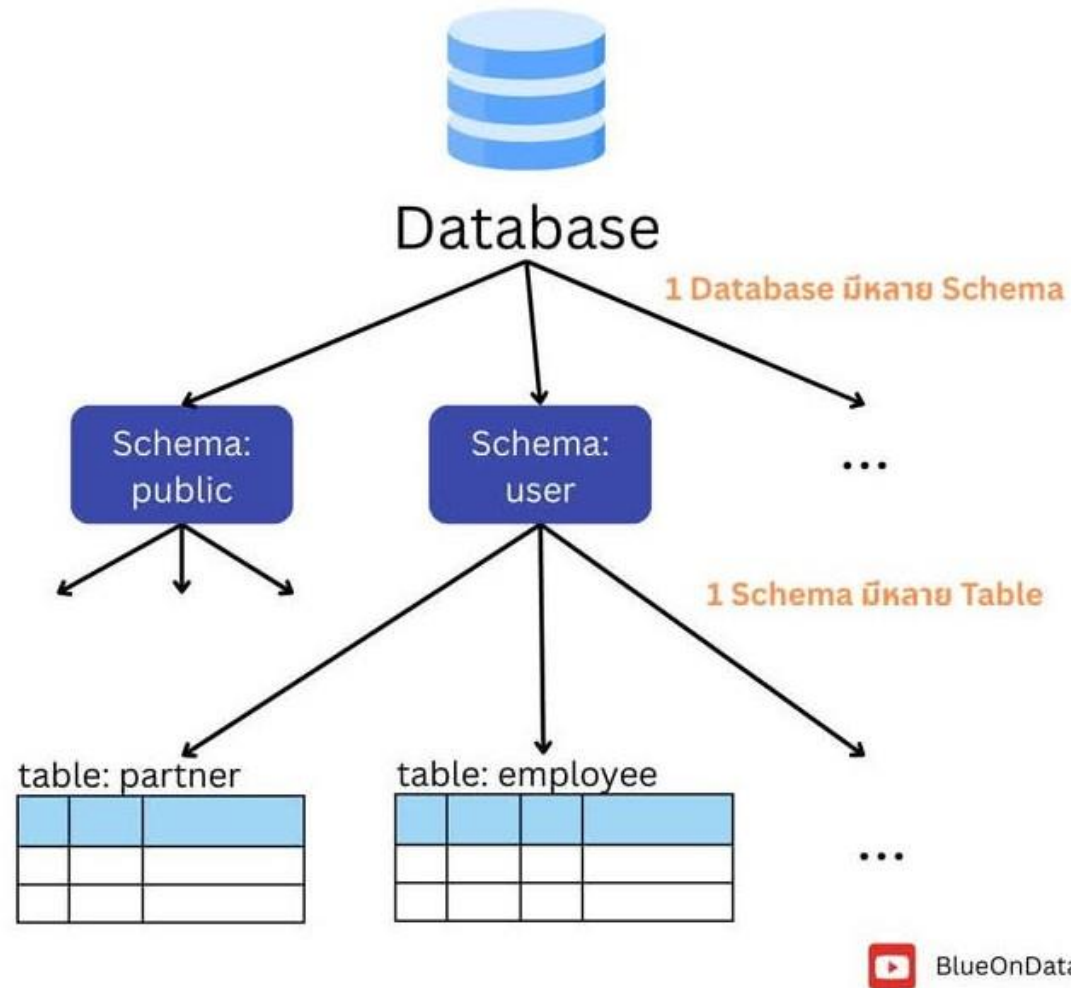
- Scientific database
- Library management system
- Banking
- Social media sites
- Online shopping
- Human resource management
- Manufacturing
- Healthcare system
- Education Sector
- Telecommunication sector
- Financial sector
- Military purpose
- Agricultural field
- Railway and airline reservation system
- Ride-sharing concept

CRUD Operations in DBMS

- Create
- Read
- Update
- Delete



Relational Database Structure



การออกแบบ Database



ข้อมูลดิบ

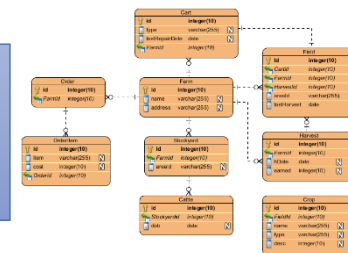
เมื่อได้รับข้อมูลมา สิ่งแรกที่ต้องทำ คือการวิเคราะห์ข้อมูลว่าข้อมูลที่ได้รับมามีลักษณะอย่างไร เพื่อจะได้นำไปสู่การออกแบบ ERD ที่สะดวก และง่ายยิ่งขึ้น

วิเคราะห์
ข้อมูล

นำข้อมูลใส่ลง
ในตาราง

ออกแบบ ER diagram ตาม
รูปแบบของข้อมูลที่ได้
วิเคราะห์ไว้ในเบื้องต้น

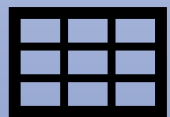
ออกแบบ ERD



ทำ
Normalization

หลังจากออกแบบ ER Diagram แล้ว จะได้ตารางสำหรับใส่ข้อมูลเบื้องต้น แต่ต้องทำ Normalization เพื่อลดความซ้ำซ้อนของข้อมูลที่จะบันทึกลงไป

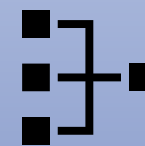
การออกแบบ Database (มุมมอง Data Science)



CSV/RAW Data



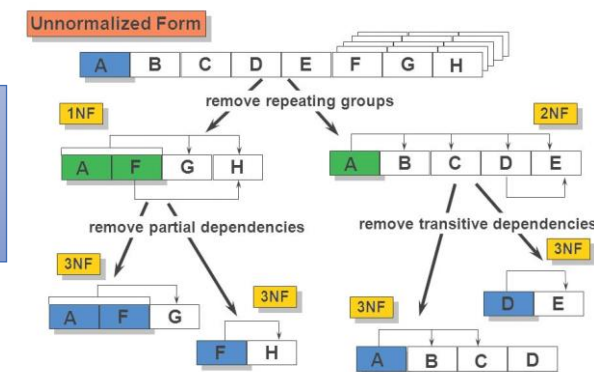
วิเคราะห์
ข้อมูล



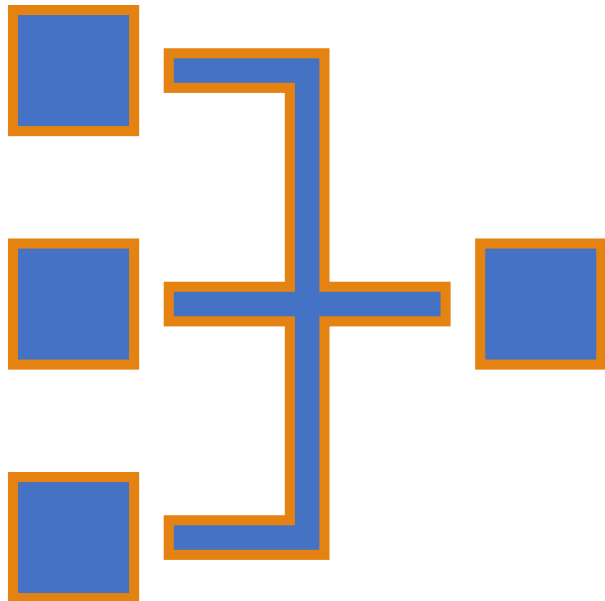
ออกแบบ ERD

นำข้อมูลใส่ลงใน
ตาราง

ทำการ
Normalization

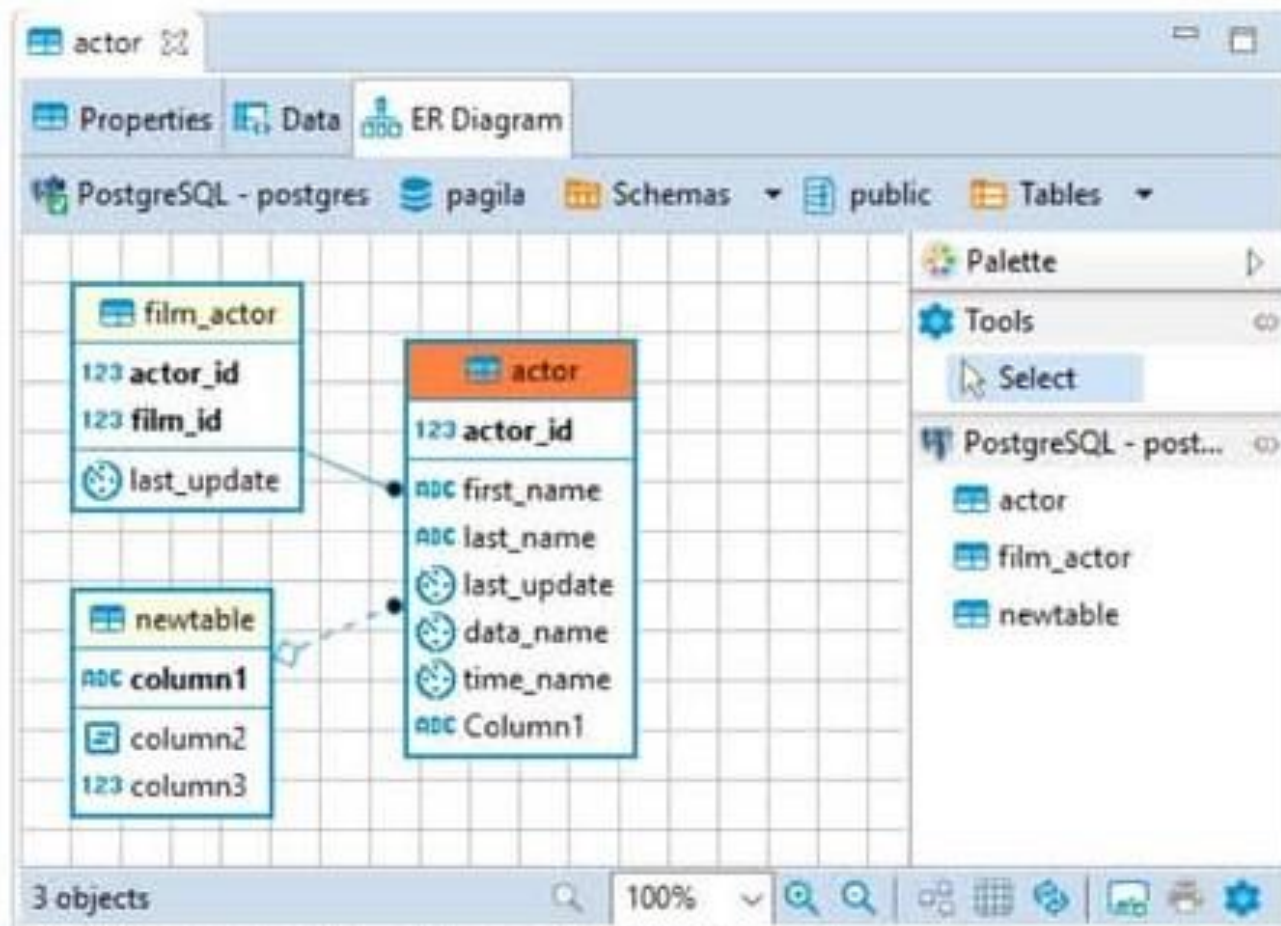


การออกแบบ Relational Database



- Entity Relationship (ER) Diagram
 - Diagram แสดงความสัมพันธ์ระหว่างข้อมูลภายในระบบใดๆ ใช้ออกแบบระบบงานใดๆก็ได้แต่นิยมมากในการใช้ออกแบบฐานข้อมูลแบบ Relational
 - ขั้นตอนการออกแบบ:
 1. ระบุว่าในระบบมีวัตถุ หรือกลุ่มข้อมูล (Entity) กี่สิ่ง อะไรบ้าง
 2. ระบุความสัมพันธ์ระหว่าง Entity ภายในระบบ
 3. กำหนด Property ให้แต่ละ Entity

Example of ER Diagram



Tables in the system
can be visualized
using ER Diagrams

การออกแบบ Relational Database

- ตัวอย่าง: ระบบฐานข้อมูลการสมัครงานเดิน-วิ่ง มินิมาราธอน

1. กำหนด Entity

Runner

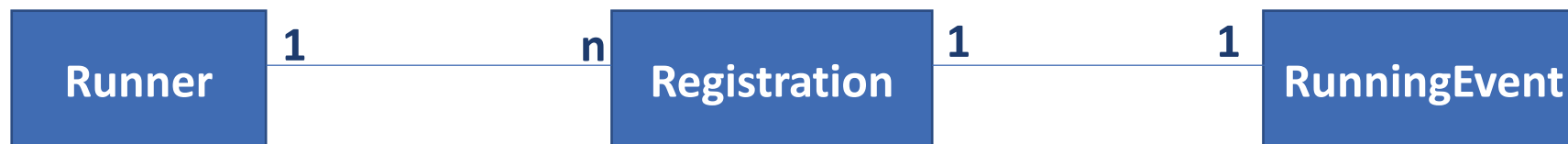
Registration

RunningEvent

การออกแบบ Relational Database

- ตัวอย่าง: ระบบฐานข้อมูลการสมัครงานเดิน-วิ่ง มินิมาราธอน

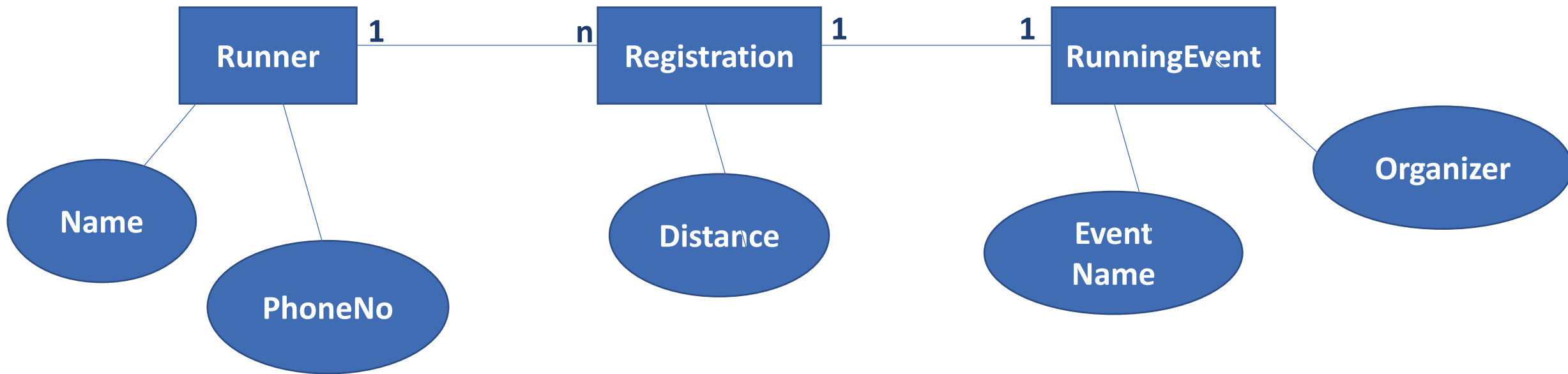
2. ระบุความสัมพันธ์ระหว่าง Entity ว่าเป็นแบบ 1..1, 1..n, หรือ n.. n



การออกแบบ Relational Database

- ตัวอย่าง: ระบบฐานข้อมูลการสมัครงานเดิน-วิ่ง มินิมาราธอน

3. กำหนด Property



Database Normalization

- หลังจากการออกแบบ ER Diagram ขั้นตอนสำคัญในการออกแบบฐานข้อมูลแบบ Relational คือการ Normalize
- เป้าหมายของการ Normalize คือการป้องกันไม่ให้ฐานข้อมูลมีการบันทึกข้อมูลซ้ำซ้อนกันจนใช้พื้นที่ในการบันทึกข้อมูลอย่างไม่มีประสิทธิภาพ
- โครงสร้าง Relational Database ที่ Normalize แล้วจะเรียกว่า Normal Form (NF)
 - ระดับของ NF
 - 1NF
 - 2NF
 - 3NF
 - BCNF

Database Normalization

- ตัวอย่างตารางของข้อมูลที่มีความซ้ำซ้อนกัน ไม่อยู่ในรูปแบบ Normal Form
 - ตารางเก็บข้อมูลผู้สมัครงานวิ่ง

รหัสลูกค้า	ชื่อ	นามสกุล	งานวิ่งที่สมัคร	หมายเลข Bib	อำเภอที่จัด	จังหวัดที่จัดงาน
1	ชินพงศ์	อังสุโชติเมธี	วิ่งวิทยา	20111	หาดใหญ่	สงขลา
				20118	หาดใหญ่	สงขลา
			วิ่งวิสา	20112	สทิงพระ	สงขลา
2	ณัฐวุฒิ	ตัน	วิ่งวิสา	20115	สทิงพระ	สงขลา

Database Normalization – 1NF

- 1NF : ตารางที่เก็บจะต้องไม่มี Record ใดเก็บข้อมูลแบบ Multi-Value
 - การแก้ไข : ปรับ Multi Value เป็นแถวใหม่เพิ่มเติมทั้งหมด

รหัสลูกค้า	ชื่อ	นามสกุล	งานวิที่สมัคร	หมายเลข Bib	อำเภอที่จัด	จังหวัดที่จัดงาน
1	ชินพงศ์	อังสุโชติเมธี	วิงวิทยา	20111	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วิงวิทยา	20118	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วิงวิสาระ	20112	สทิงพระ	สงขลา
2	ณัฐวุฒิ	ตัน	วิงวิสาระ	20115	สทิงพระ	สงขลา

Database Normalization – Primary Key

- ตารางใดๆใน Relational Database จำเป็นต้องมี Primary Key
- Primary Key คือ Field หรือ กลุ่มของ Field ที่มีคุณสมบัติดังนี้
 - ไม่มีค่าใดๆซ้ำซ้อนกัน
 - ไม่มีค่าว่าง (null)

ตัวอย่าง : รหัสลูกค้า, งานวิจัยที่สมัคร, หมายเลข Bib เป็น Primary Key

รหัสลูกค้า	ชื่อ	นามสกุล	งานวิจัยที่สมัคร	หมายเลข Bib	อำเภอที่จัด	จังหวัดที่จัดงาน
1	ชินพงศ์	อังสุโชติเมธี	วังวิทยา	20111	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วังวิทยา	20118	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วังวิสวะ	20112	สทิงพระ	สงขลา
2	ณัฐวุฒิ	ตัน	วังวิสวะ	20115	สทิงพระ	สงขลา

Database Normalization - Functional Dependency

- หมายถึงการที่ Field ใดๆในตารางมีความสัมพันธ์เชื่อมโยงถึงกันเมื่อจำเป็นต้องใช้งานข้อมูล เช่น **ชื่อ นามสกุล** และ **รหัสลูกค้า** เป็น Functional Dependency กัน
 - Partial Functional Dependency: การที่ Field นั้นๆเกี่ยวข้องกับ Primary Key เพียงบาง Field

รหัสลูกค้า	ชื่อ	นามสกุล	งานวิจัยที่สมัคร	หมายเลข Bib	อำเภอที่จัด	จังหวัดที่จัดงาน
1	ชินพงศ์	อังสุโชติเมธี	วังวิทยา	20111	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วังวิทยา	20118	หาดใหญ่	สงขลา
1	ชินพงศ์	อังสุโชติเมธี	วังวิสวะ	20112	สติงพระ	สงขลา
2	ณัฐวุฒิ	ตัน	วังวิสวะ	20115	สติงพระ	สงขลา

Database Normalization – 2NF

- 2NF : ตารางที่เป็น 2NF นั้นทุกๆ Field จะต้องมีความสัมพันธ์กับ Primary Key แบบ Full Functional Dependency เท่านั้น
 - การปรับตารางให้เป็น 2NF: ให้แยกกลุ่มของ Partial Functional Dependency ออกมาเป็นตารางใหม่
 - เพิ่ม Field ในตารางที่ถูกแยกออกมาโดยระบุข้อมูลที่สื่อถึงตารางที่แยกออกมาโดยอาศัย Primary Key
 - ข้อมูลที่ใช้สื่อถึง Primary Key ของอีกตารางเรียกว่า Foreign Key

รหัสลูกค้า	ชื่อ	นามสกุล
1	ชินพงศ์	อังสุโชติเมธี
2	ณัฐวุฒิ	ตัน

งานวิ่ง	อำเภอที่จัด	จังหวัดที่จัดงาน
วิ่งวิทยา	หาดใหญ่	สงขลา
วิ่งวิสวะ	สติงพระ	สงขลา

รหัสลูกค้า (FK)	งานวิ่ง (FK)	หมายเลข Bib
1	วิ่งวิทยา	20111
1	วิ่งวิทยา	20118
1	วิ่งวิสวะ	20112
2	วิ่งวิสวะ	20115

รหัสลูกค้า, งานวิ่งที่สมัคร, หมายเลข Bib เป็น Primary Key

Foreign Key (FK)

แต่ละตาราง จะมีการกำหนด PK (Primary Key) และ FK (Foreign key)

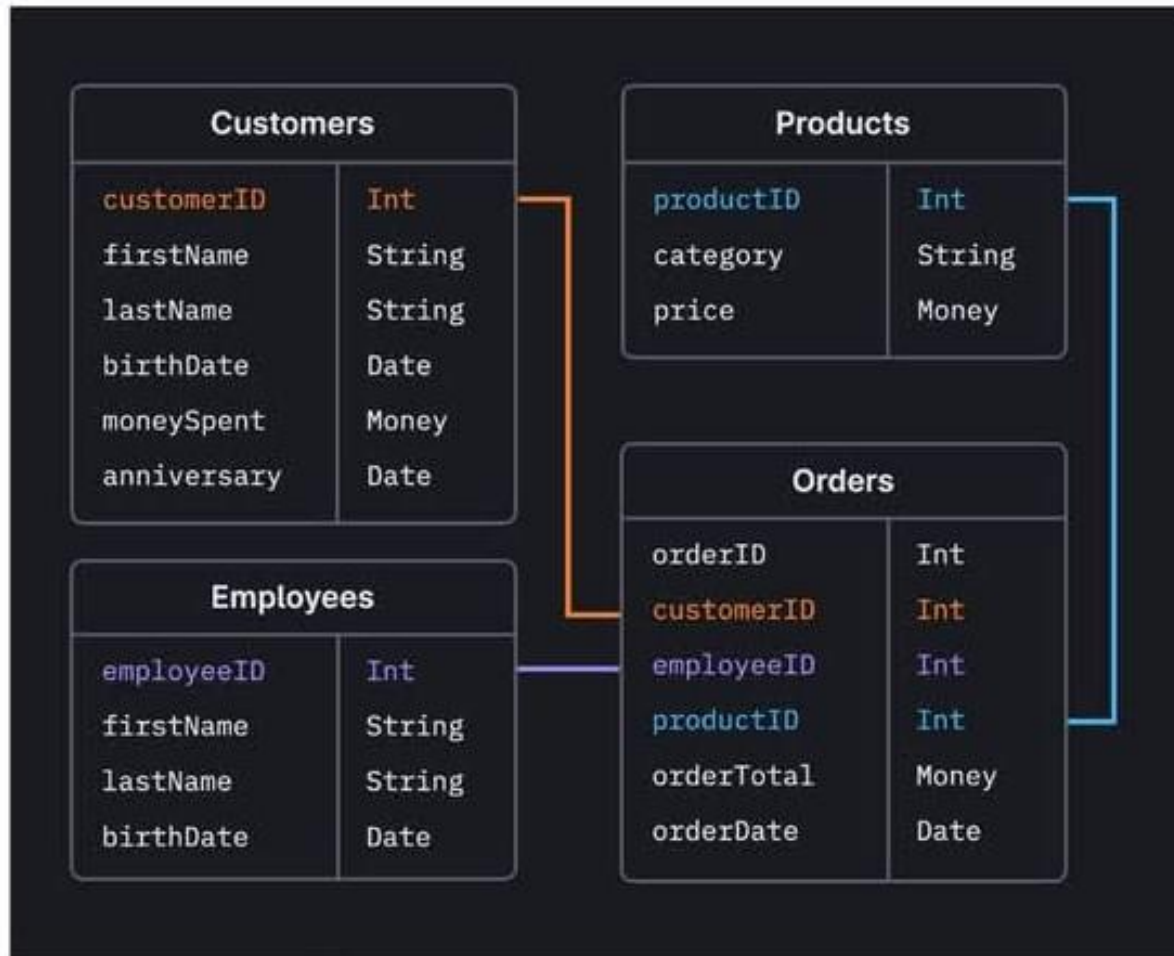
Primary key: ค่าของแต่ละแถวที่จะไม่มีการซ้ำกันเลย (เหมือนค่าประจำตัว)

id	name	department_id	created_at
1	Alice	3	2024-01-01 10:00
2	Bob	40	2024-01-02 11:15

Foreign key: ค่าที่จะอ้างอิงไปยัง PK ของอีกตาราง

- Field หรือกลุ่มของ Field ในตารางใด ๆ ที่อ้างอิงถึง PRIMARY KEY ในตารางอื่น
- ตารางที่มี Foreign Key จะเรียกว่า child table และตารางที่มี primary key เรียกว่า referenced table หรือ parent table.
- FK สร้างความสัมพันธ์ระหว่าง 2 ตาราง

PK and FK Example



<https://planetScale.com/blog/schema-design-101-relational-databases/>

- PK and FK enable us to see the relationships amongst tables

Database Normalization – 3NF

- Transitive Dependency: การที่ Field ที่ไม่ได้เป็น Primary Key มี Functional Dependency ระหว่างกัน
 - เช่น **อำเภอ** เป็น Transitive Dependency กับ **จังหวัด**
- 3NF: กำจัด Transitive Dependency

รหัสลูกค้า, งานวิ่งที่สมัคร, หมายเลข Bib เป็น Primary Key

รหัสลูกค้า	ชื่อ	นามสกุล
1	ชินพงศ์	อังสุโชติเมธี
2	ณัฐวุฒิ	ตัน

งานวิ่ง	อำเภอที่จัด (FK)
วิ่งวิทยา	หาดใหญ่
วิ่งวิสาระ	สตึงพระ

งานวิ่ง	อำเภอที่จัด	จังหวัดที่จัดงาน
วิ่งวิทยา	หาดใหญ่	สงขลา
วิ่งวิสาระ	สตึงพระ	สงขลา

อำเภอ	จังหวัด
หาดใหญ่	สงขลา
สตึงพระ	สงขลา

รหัสลูกค้า (FK)	งานวิ่ง (FK)	หมายเลข Bib
1	วิ่งวิทยา	20111
1	วิ่งวิทยา	20118
1	วิ่งวิสาระ	20112
2	วิ่งวิสาระ	20115

Database Normalization – BCNF or 3.5NF

- Candidate Key: ตารางที่มีฟิลด์ใดๆที่มีค่า Unique เช่นเดียวกัน สามารถเป็น Primary Key ได้ เพียงแต่อาจจะไม่ถูกเลือกนำมาใช้เป็น Primary Key
- Boyce-Codd Normal Form (BCNF): ทุกๆ Field ในตารางจะต้องมี Functional Dependency กับทั้ง Primary และ Candidate Key เสมอ
- 3NF ส่วนใหญ่จะเป็น BCNF ด้วย

<u>รหัสลูกค้า</u>	<u>ชื่อ</u>	<u>นามสกุล</u>
1	ชินพงศ์	อังสุโชติเมธี
2	ณัฐภูมิ	ตัน

<u>อำเภอ</u>	<u>จังหวัด</u>
หาดใหญ่	สงขลา
สทิงพระ	สงขลา

<u>งานวิ่ง</u>	<u>อำเภอที่จัด (FK)</u>
วิ่งวิทยา	หาดใหญ่
วิ่งวิสวะ	สทิงพระ

<u>รหัสลูกค้า (FK)</u>	<u>งานวิ่ง (FK)</u>	<u>หมายเลข Bib</u>
1	วิ่งวิทยา	20111
1	วิ่งวิทยา	20118
1	วิ่งวิสวะ	20112
2	วิ่งวิสวะ	20115

Database Normalization – Candidate Keys

- **Candidate Key** คือ แอททริบิวต์ที่มีความสามารถที่จะเป็น Primary key ได้
- จากตารางนี้ FirstName และ LastName ถือว่าเป็น **Candidate Key** เพราะ ใช้
ระบุด้านอื่นๆ ในตารางได้ครบถ้วนเหมือน Primary key

Student ID	FirstName	LastName	CourseID
S2345	Harry	Potter	C002
S1254	James	Price	A042
S2349	Hermione	Granger	P302
S1853	John	Murray	P302
S1198	Elon	Musk	C002
S4589	Simon	Holland	A042
S8514	Anne	Black	H602

ตารางนี้
ชื่อไม่ซ้ำกัน

Database Normalization - Student/Professor Table


ข้อกำหนด

อาจารย์ 1 สอน 1 วิชา

วิเคราะห์

- นักเรียน 1 คนสามารถเรียนได้มากกว่า 1 วิชา
- บางวิชาอาจมีอาจารย์ผู้สอนหลายคน
- มี dependency ระหว่าง Subject และ Professor

รหัสนักศึกษา



ID	Subject	Professor
101	Java	Mayank
101	C++	Kartik
102	Java	Sarthak
103	C#	Lakshay
104	Java	Mayank

Database Normalization – BCNF Satisfied

Professor is now the primary key.

Professor Table

Professor	Subject
Mayank	Java
Kartik	C++
Sarthak	Java
Lakshay	C#
Mayank	Java

Student Table

PID	SID	Subject	Professor
1	101	Java	Mayank
2	101	C++	Kartik
3	102	Java	Sarthak
4	103	C#	Lakshay
5	104	Java	Mayank

Database Normalization - BCNF Decomposition

EmployeeID	ProjectID	ProjectLeader
101	P03	Grey
101	P01	Chris
102	P04	Hudson
103	P02	Pedro

EmployeeID	ProjectID	ProjectLeader	ProjectID
101	P03	Grey	P03
101	P01	Chris	P01
102	P04	Hudson	P04
103	P02	Pedro	P02

Source:
<https://medium.com/@yg17381/normalization-in-dbms-297269fe7e9f>

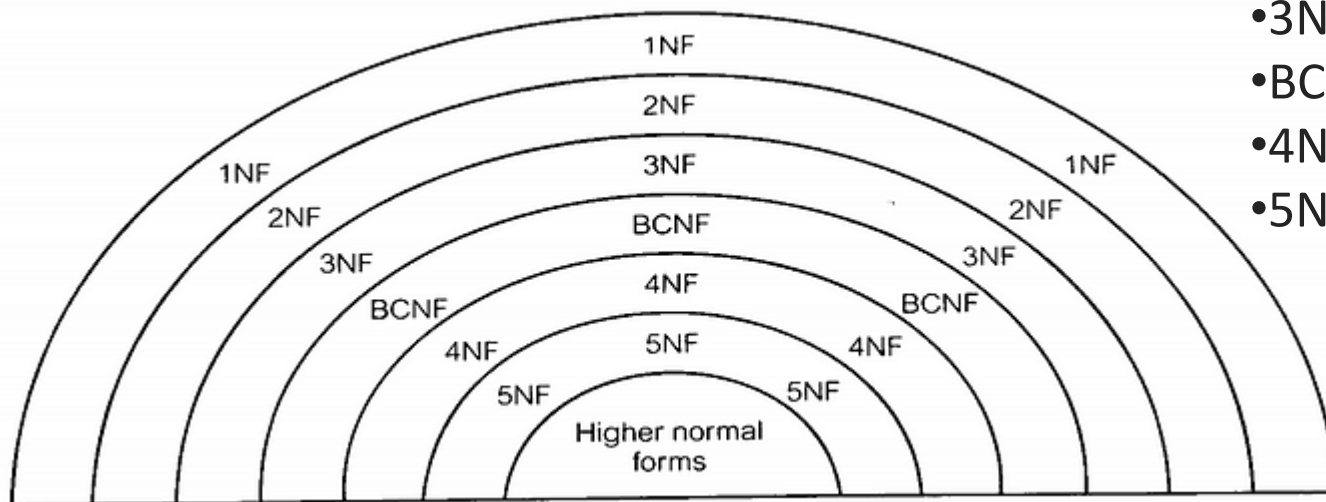
Database Normalization – 3NF but not BCNF

- การกู้ยืมเงินหนึ่ง ๆ อาจมี customer หลายคน ดังนั้น Primary Key จะเป็น LoanID + CustomerID
- มี LoanID สามารถหา LoanAmount แต่ไม่ใช่ Primary Key ดังนั้นตารางนี้ไม่ใช่ BCNF

BranchNo	CustomerID (PK)	LoanID (PK)	LoanAmount
1	100512	3141	150000
1	122099	3444	25000
2	099344	6501	30000

Database Normalization

- การทำ Normalization จะต้องทำเป็นขั้นตอน โดยเริ่มจาก **Unnormalized Form** หรือ ข้อมูลดั้งเดิม > **1NF** > **2NF** > **3NF** > **BCNF** > **4NF** > **5NF** จนได้รูปแบบของ Higher normal forms โดยที่ห้ามข้ามขั้นตอน ถ้าไม่อยู่ในรูปแบบก่อนหน้านี้ เช่น ถ้าไม่อยู่ในรูปของ 1NF จะข้ามไปทำ 3NF เลยไม่ได้



- 1NF ปรับปรุง **Multi-Value Records**
- 2NF กำจัด **Partial Dependency**
- 3NF กำจัด **Transitive Dependency**
- BCNF กำจัด **Candidate Key**
- 4NF กำจัด **Mutivalued Dependency**
- 5NF ทำ **Join Dependency** แล้ว ได้ตารางที่เหมือนเดิม

Exercise

- จง Normalize ตารางดังต่อไปนี้ให้อยู่ขั้นต่ำในรูป 3NF

รหัส นักศึกษา	ชื่อ-นามสกุล	รหัสวิชา	ชื่อวิชา	ชื่อ ผู้สอน	คะแนน ที่ได้	เกรดที่ ได้
48270315	Chinna Etche	344-243	Web Programming	Chin	85	A
		344-346	Game Programming	Ang	80	B
52432093	Natta Keng	344-511	Computer Organization	Ang	75	B+
		345-103	Com Creative	Chin	90	A

Structure Query Language (SQL)

- เป็นภาษาที่ใช้ในการจัดการ Relational Database
- ได้รับการยอมรับว่าเป็นภาษามาตรฐานในการจัดการ Relational Database โปรแกรมฐานข้อมูล Relational Database ทุกโปรแกรมในปัจจุบันจึงใช้ภาษา SQL ในการจัดการ Database
- โปรแกรม Relational Database ที่เป็นที่ยอมรับและใช้ภาษา SQL ในการจัดการ
 - MySQL
 - MariaDB
 - SQLite
 - PostgreSQL
 - Microsoft SQL
 - Microsoft Access
 - Oracle Database

MySQL

- โปรแกรมฐานข้อมูล Relational Database ที่โด่งดังที่สุดคือ MySQL (<https://www.mysql.com>) เนื่องจากเป็นระบบ Opensource ที่ใช้งานได้ฟรีและมีประสิทธิภาพ
- MySQL ในปัจจุบันนี้ลิขสิทธิ์ได้ตกเป็นของบริษัท Oracle ตั้งแต่วันที่ 27 มกราคม 2553 โดย Oracle ได้เผยแพร่ MySQL เวอร์ชันเบสิคแบบไม่คิดค่าใช้จ่าย แต่ฟังก์ชันขั้นสูงระดับ Enterprise Grade หรือโปรแกรมช่วยเหลืออื่นๆนั้นจะมีค่าใช้จ่าย

MariaDB

- Programmer ในทีมที่พัฒนา MySQL มาตั้งแต่ต้นได้เลือกที่จะสร้างโปรเจกต์ MariaDB ขึ้นมาใหม่โดยใช้โค้ดพื้นฐานเดียวกับ MySQL เพื่อให้โครงการ Opensource ยังคงมีโปรแกรม Relational Database ที่ยังใช้งานได้ฟรี และไม่ผู้ใดเป็นเจ้าของสิทธิขาดโดยแท้จริงต่อไป
- ปัจจุบัน: MariaDB ได้รับความนิยมสูงมากทั้งในระดับการพัฒนาเว็บไซต์เล็กๆ ไปจนถึงโครงการใหญ่ โดย Programmer ที่เดิมใช้ MySQL ได้ทยอยเปลี่ยนมาใช้ MariaDB แทนเรื่อยๆ เพื่อป้องกันปัญหาลิขสิทธิ์กับบริษัท Oracle และการได้รับการ Update และ Security Patch ต่างๆจาก Opensource Community ที่มักจะไวกว่าของ Oracle เองเสมอ
- Website: <https://www.mariadb.org>

Structure Query Language (SQL)

- คำสั่งพื้นฐานของภาษา SQL
 - CREATE TABLE
 - SELECT
 - INSERT
 - UPDATE
 - DELETE

Structure Query Language (SQL)

- CREATE TABLE: เป็นคำสั่งที่มีไว้สร้างตารางใหม่
 - Syntax:

```
CREATE TABLE IF NOT EXISTS [ชื่อตาราง] (  
    [ชื่อ field] [ชนิดข้อมูล],  
    [ชื่อ field] [ชนิดข้อมูล],  
    [ชื่อ field] [ชนิดข้อมูล],  
    .  
    .  
);
```

Structure Query Language (SQL)

- CREATE TABLE: เป็นคำสั่งที่มีไว้สร้างตารางใหม่
 - Syntax:

```
CREATE TABLE IF NOT EXISTS [ชื่อตาราง] (  
    [ชื่อ field] [ชนิดข้อมูล],  
    [ชื่อ field] [ชนิดข้อมูล],  
    [ชื่อ field] [ชนิดข้อมูล],  
    .  
    .  
);
```

Structure Query Language (SQL)

- CREATE TABLE: เป็นคำสั่งที่มีไว้สร้างตารางใหม่
 - ตัวอย่าง

```
CREATE TABLE IF NOT EXISTS students (  
    id varchar(8),  
    name varchar(50),  
    surname varchar(50),  
    faculty varchar(20)  
);
```



students			
ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschoetmetee	IT
48270315	Chinna	Etchegarray	CS

Structure Query Language (SQL)

- INSERT: ใช้เพิ่ม Record ใหม่เข้าไปในตาราง

- Syntax:

- INSERT INTO [ชื่อตาราง] ([field1], [field2],)
 - VALUES ([value1], [value2],);

- ตัวอย่าง

INSERT INTO students (ID, Name, Surname, Faculty) VALUES ('48222331', 'Chinsan', 'Damkern', 'CPE')



students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmetee	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE

Structure Query Language (SQL)

- UPDATE: ใช้แก้ไข Record ที่มีอยู่แล้วในตาราง

- Syntax:

- UPDATE [ชื่อตาราง] SET [ชื่อ Field] = [ค่าที่ต้องการแก้ไข], [ชื่อ Field] = [ค่าที่ต้องการแก้ไข],

.....

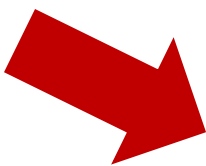
WHERE [เงื่อนไข]

- ตัวอย่าง

UPDATE students SET Name='PingPong', Surname='Sushi', Faculty='Med'

WHERE ID='48222331'

students



ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmetee	IT
48270315	Chinna	Etchegarray	CS
48222331	PingPong	Sushi	Med

Structure Query Language (SQL)

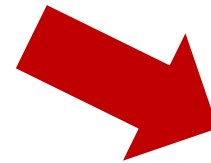
- DELETE: ใช้ลบ Record

- Syntax:

- DELETE FROM [ชื่อตาราง] WHERE [เงื่อนไข]

- ตัวอย่าง

DELETE FROM students WHERE Name="Chinnapong"



students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothete	IT
48270315	Chinna	Etchegarray	CS
48222331	PingPong	Sushi	Med

Structure Query Language (SQL)

- SELECT: ใช้ทำการเรียกข้อมูลจากออกมาแสดง
 - Syntax:
 - SELECT [ชื่อ Field], [ชื่อ Field],
FROM [ชื่อตาราง], [ชื่อตาราง]
WHERE [เงื่อนไข];

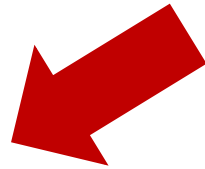
Structure Query Language (SQL)

- ตัวอย่าง

SELECT * FROM students;

students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmete	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE



ผลลัพธ์

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmete	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE

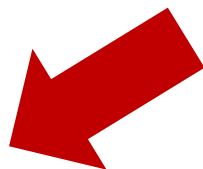
Structure Query Language (SQL)

- ตัวอย่าง

SELECT ID, Name FROM students;

students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmete	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE



ผลลัพธ์

ID	Name
52432093	Chinnapong
48270315	Chinna
48222331	Chinsan

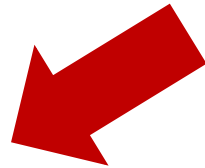
Structure Query Language (SQL)

- ตัวอย่าง

```
SELECT * FROM students
WHERE ID = '52432093';
```

students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmetee	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE



ผลลัพธ์

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmetee	IT

Structure Query Language (SQL)

students


ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmetee	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE

s344_243

ID	Midterm	Final	Grade
52432093	30	50	A
48270315	20	50	B

```
SELECT students.ID, students.Name, students.Surname, s344_243.Midterm
s344_243.Final, s344_243.Grade FROM students, s344_243
WHERE students.ID = s344_243.ID;
```

ผลลัพธ์



ID	Name	Surname	Faculty	Midterm	Final	Grade
52432093	Chinnapong	Anguschothmetee	IT	30	50	A
48270315	Chinna	Etchegarray	CS	20	50	B

Structure Query Language (SQL)

students

ID	Name	Surname	Faculty
52432093	Chinnapong	Anguschothmete	IT
48270315	Chinna	Etchegarray	CS
48222331	Chinsan	Damkern	CPE

s344_243

ID	Midterm	Final	Grade
52432093	30	50	A
48270315	20	50	B

SELECT students.ID, students.Name, students.Surname, s344_243.Midterm
s344_243.Final, s344_243.Grade FROM students, s344_243

WHERE (students.ID = s344_243.ID) AND (s344_243.Midterm > 20);



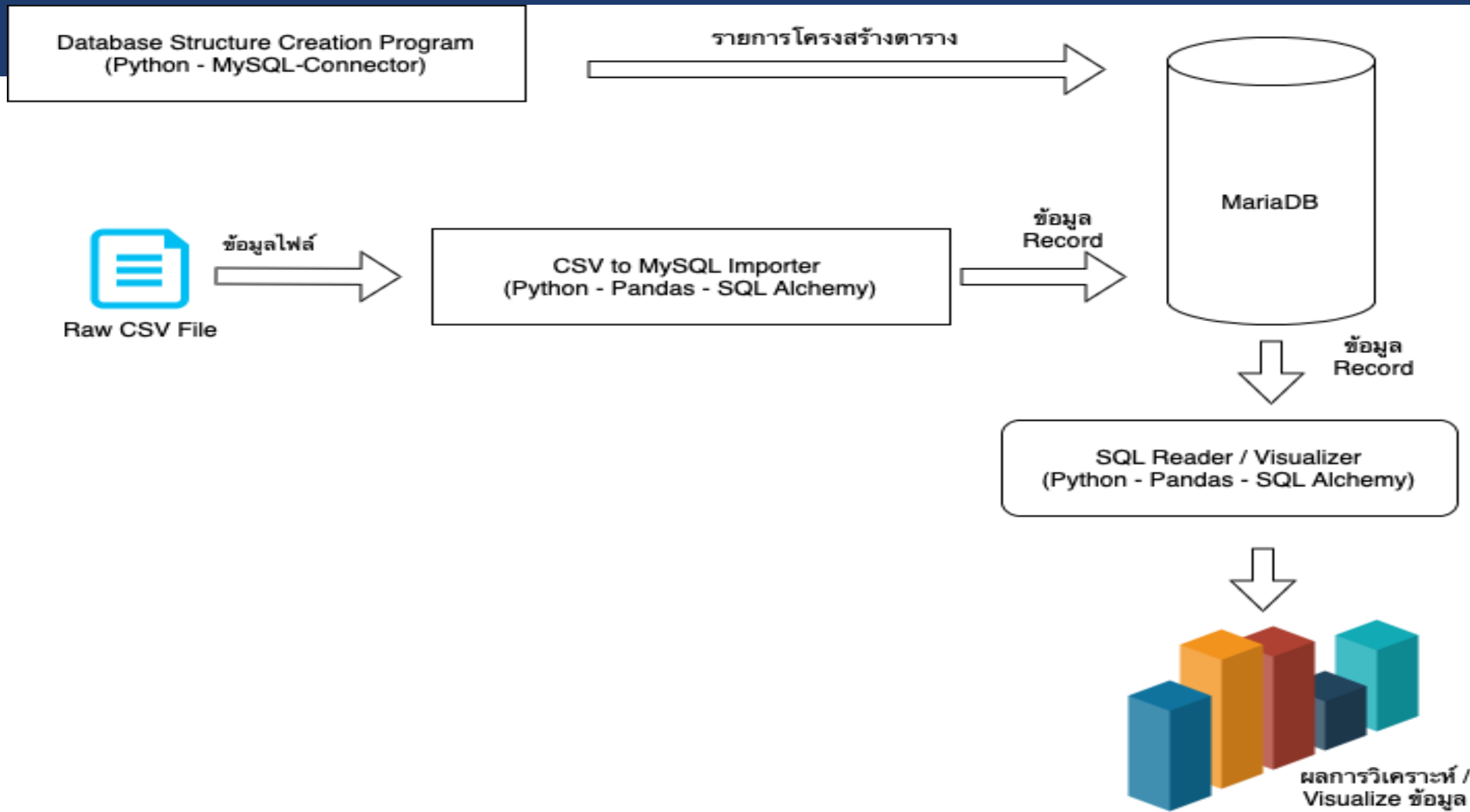
ผลลัพธ์

ID	Name	Surname	Faculty	Midterm	Final	Grade
52432093	Chinnapong	Anguschothmete	IT	30	50	A

ปฏิบัติการ: Relational Database

- MariaDB + Python + Pandas –

Lab: Program Flow



Database API Specification 2.0

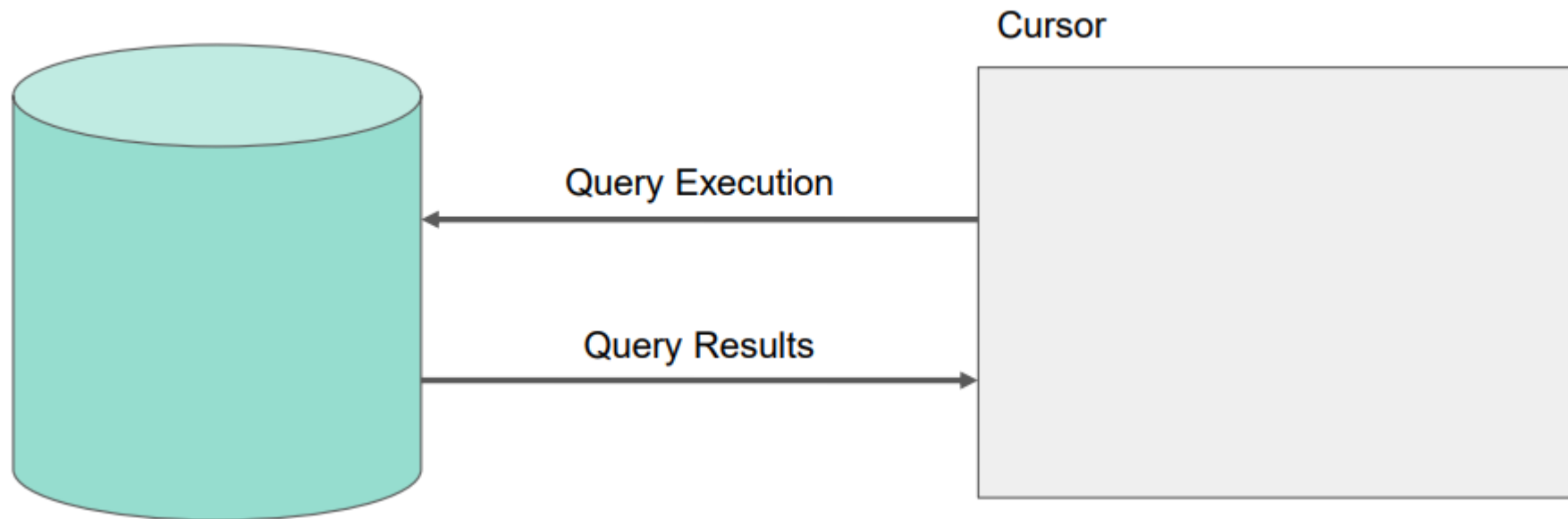
Module Interface

- Constructors
 - Access to the database via Connection
 - object connect(parameters...)
- Globals (apilevel, threadsafety, paramstyle)
- Exceptions

Connections

- `.commit()` explicitly commit any pending transactions to the database.
- `.rollback()` optional method causes a transaction to be rolled back to the starting point.
- `.close()` closes the connection to the database permanently
- `.cursor()` returns a Cursor object which uses current Connection

Cursors



Cursors

Methods

- `execute(operations [, parameters])`
- `executemany(operation, seq_of_parameters)`
- `fetchone()`
- `fetchmany([size=cursor.arraysize])`
- `fetchall()`
- `callproc(procname [, parameters])`
- `nextset()`
- `arraysize`
- `setinputsizes(sizes)`
- `setoutputsizes(size [, column]) M`

Attributes

- `description`
- `rowcount`

Type Constructors and Objects

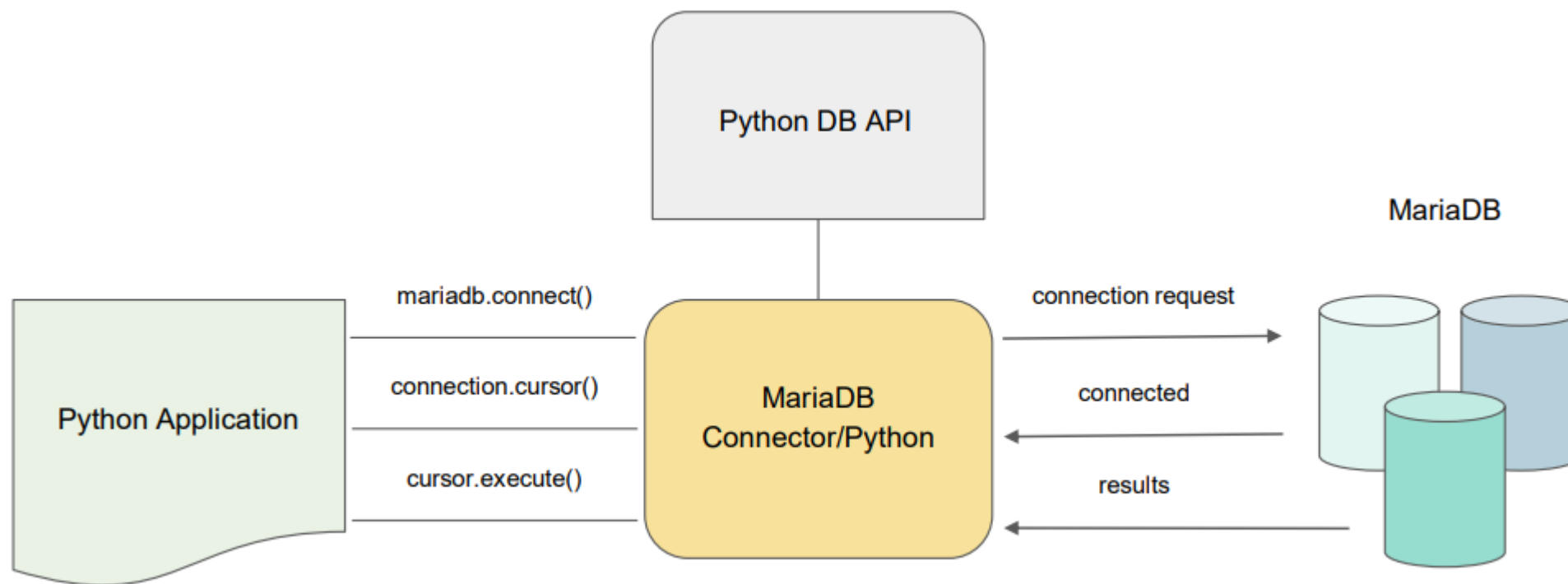
Constructors

- Date(year, month, day)
- Time(hour, minute, second)
- Timestamp(year, month, date, hour, minute, second)
- DateFromTicks(ticks)
- TimeFromTicks(ticks)
- TimestampFromTicks(ticks)

Objects

- STRING
- BINARY
- NUMBER
- DATETIME
- ROWID

MariaDB Connector/Python



MariaDB Connector/Python

- General availability (GA) June 2020
- Python DB API 2.0 (PEP-249) compliant
- Written in C and wraps MariaDB Connector/C client Installation • Python 3 (v 3.6+)
- MariaDB Connector/C (v 3.1.5+)

