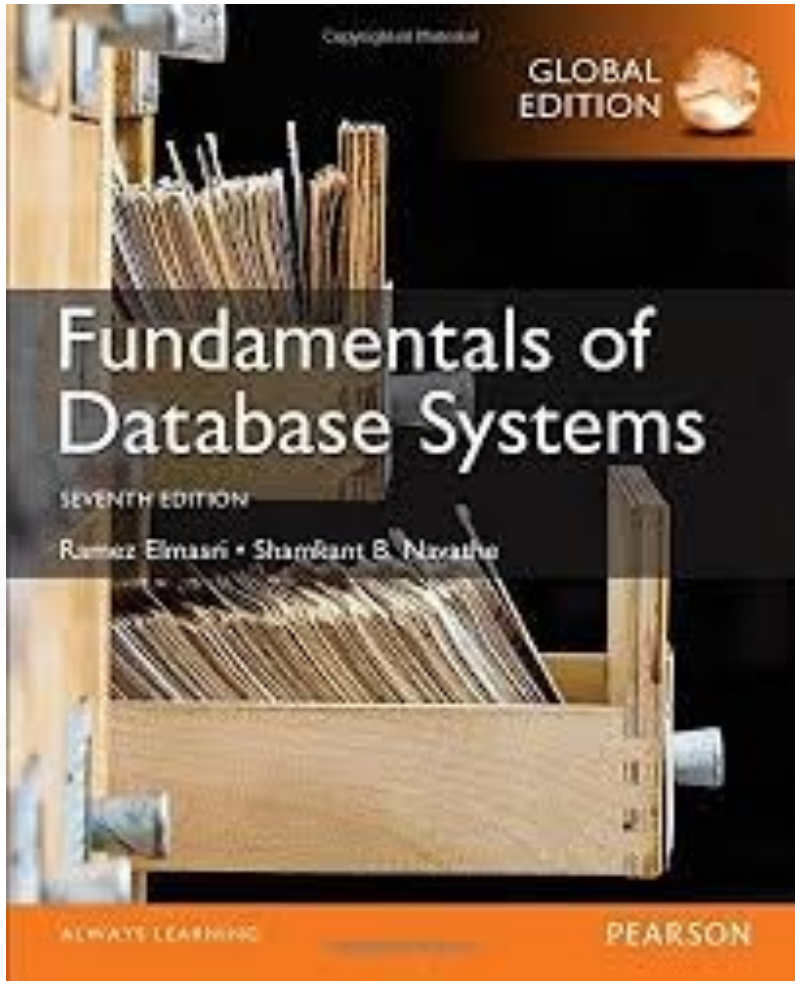
Database Systems

Program in Computer Engineering
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

Text

- Ramez Elmasri and Shamkant B. Navathe. "Fundamentals of Database Systems" 7th Edition., Pearson, 2017



Allan Turing

A. M. Turing (1950) Computing Machinery and Intelligence. *Mind 49:* 433-460.

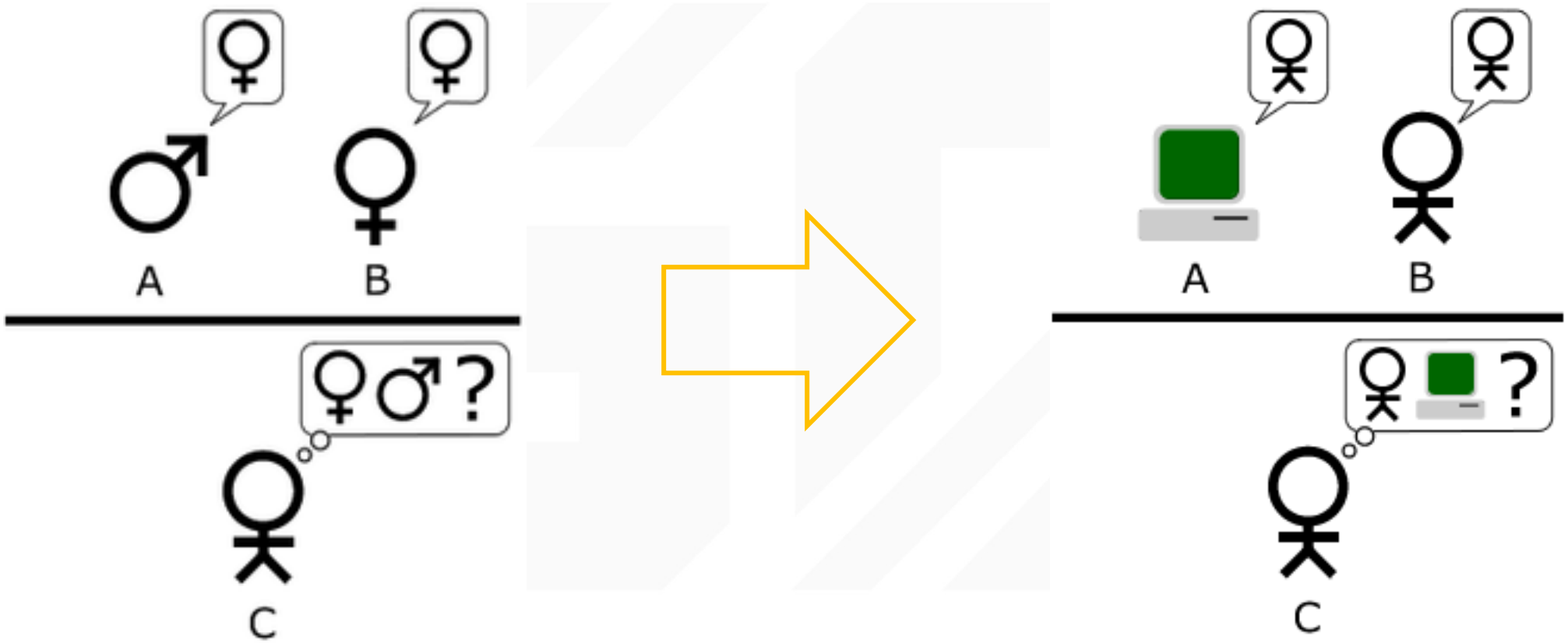
COMPUTING MACHINERY AND INTELLIGENCE

By A. M. Turing

1. The Imitation Game

- Can machine think?
  - What is a "machine"?
  - What is "think"?

A.M. Turing (1950) Computing Machinery and Intelligence. *Mind 49*: 433 - 460


A
B
C
?
A
B
C
?

Common ground of how people communicate

- ???

Common ground of how people communicate

- Mutual knowledge
- Mutual beliefs
- Mutual assumptions

Clark, Herbert H.; Brennan, Susan E. (1991), Resnick, L. B.; Levine, J. M. (eds.), *Perspectives on socially shared cognition*, American Psychological Association, ISBN 1-55798-376-3

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Databases and Database Users

Types of Databases and Database Applications

- Traditional Applications:
  - Numeric and Textual Databases
- More Recent Applications:
  - Multimedia Databases
  - Geographic Information Systems (GIS)
  - Biological and Genome Databases
  - Data Warehouses
  - Mobile databases
  - Real-time and Active Databases

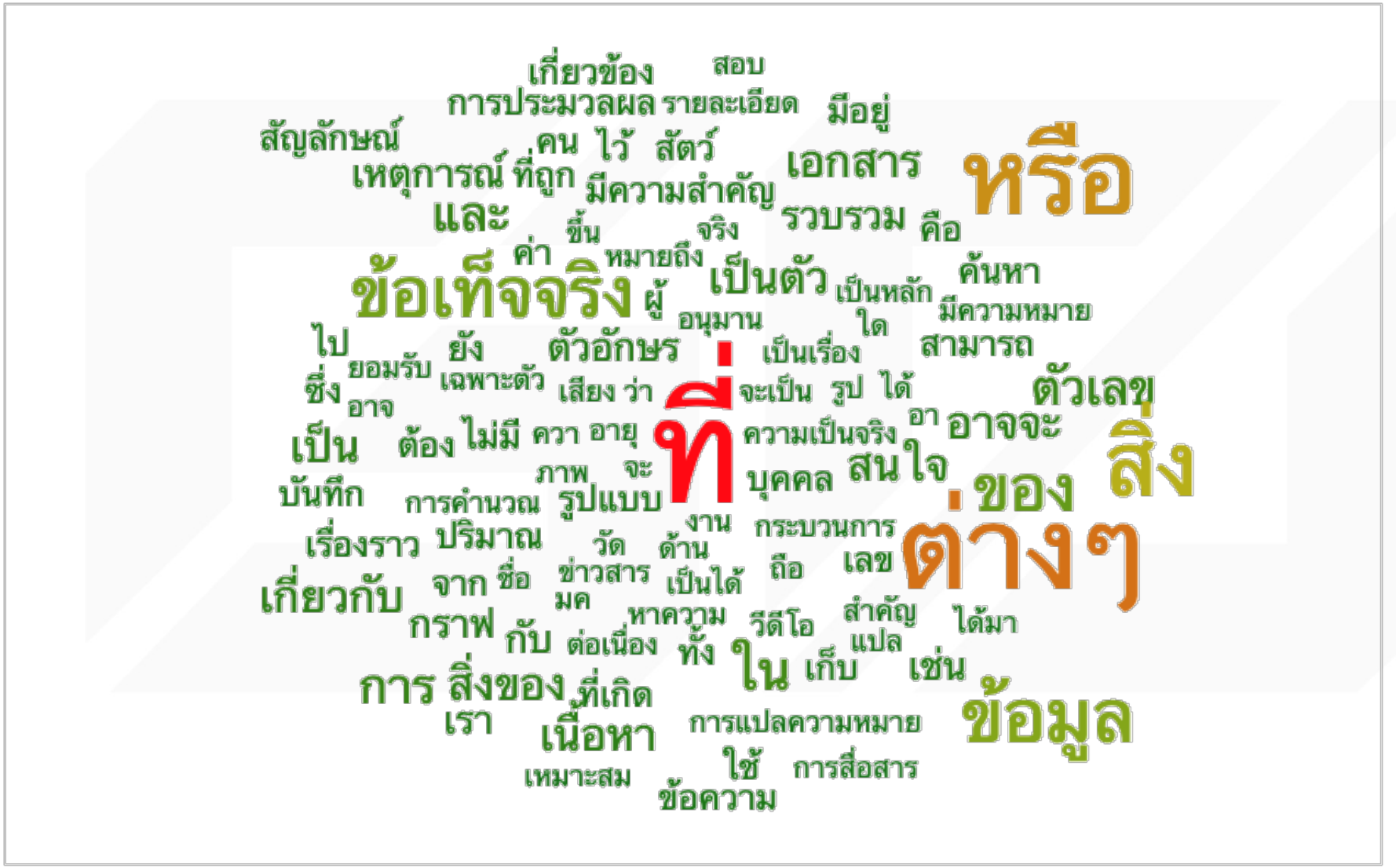
Recent Developments (1)

- Social Networks started capturing a lot of information about people and about communications among people-posts, tweets, photos, videos in systems such as:
  - Facebook
  - Twitter
  - Linked-In
- All of the above constitutes data
- Search Engines- Google, Bing, Yahoo : collect their own repository of web pages for searching purposes

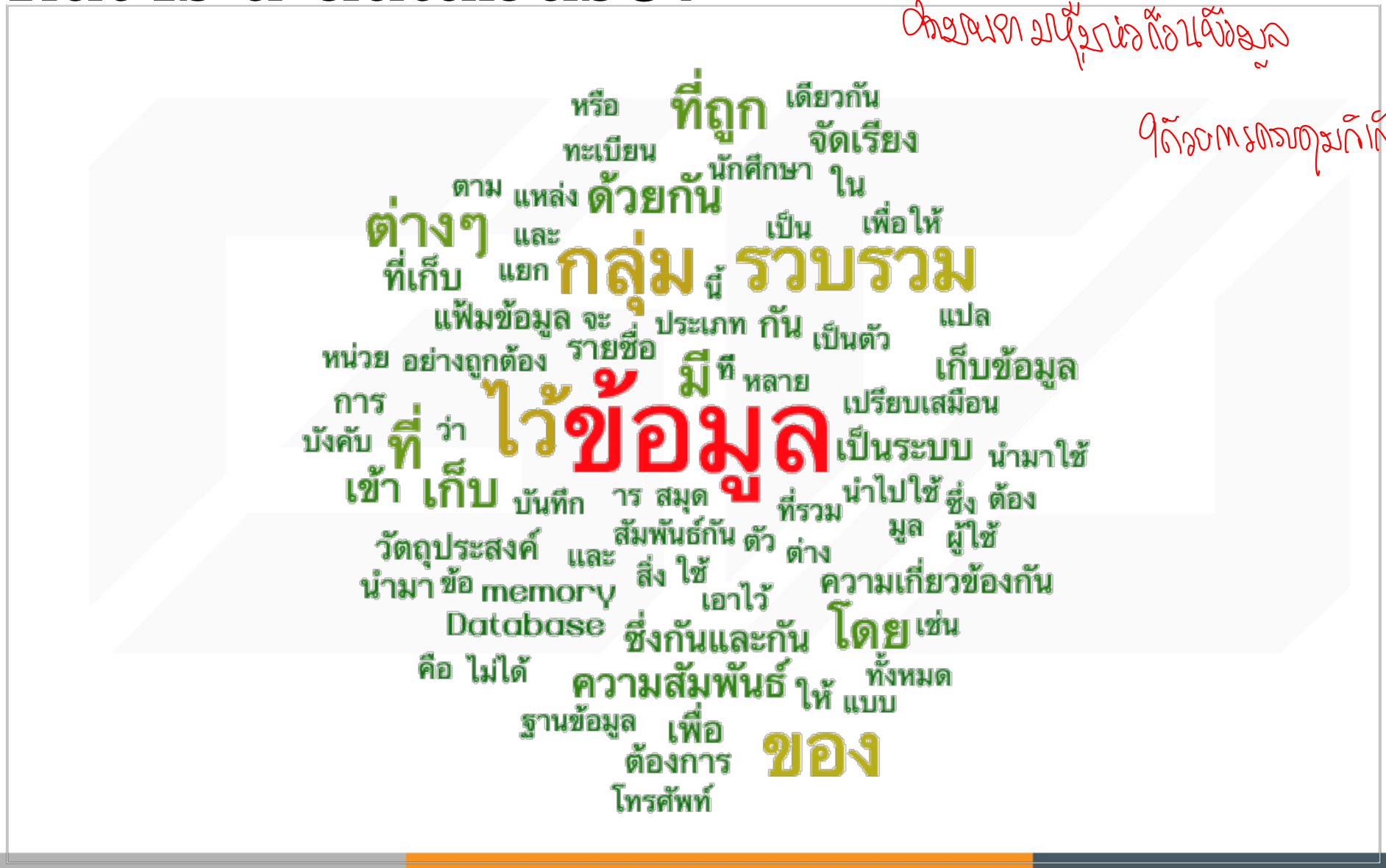
Recent Developments (2)

- New Technologies are emerging from the so-called non-database software vendors to manage vast amounts of data generated on the web:
- Big Data storage systems involving large clusters of distributed computers
- NOSQL (Not Only SQL) systems
- A large amount of data now resides on the "cloud" which means it is in huge data centers using thousands of machines.

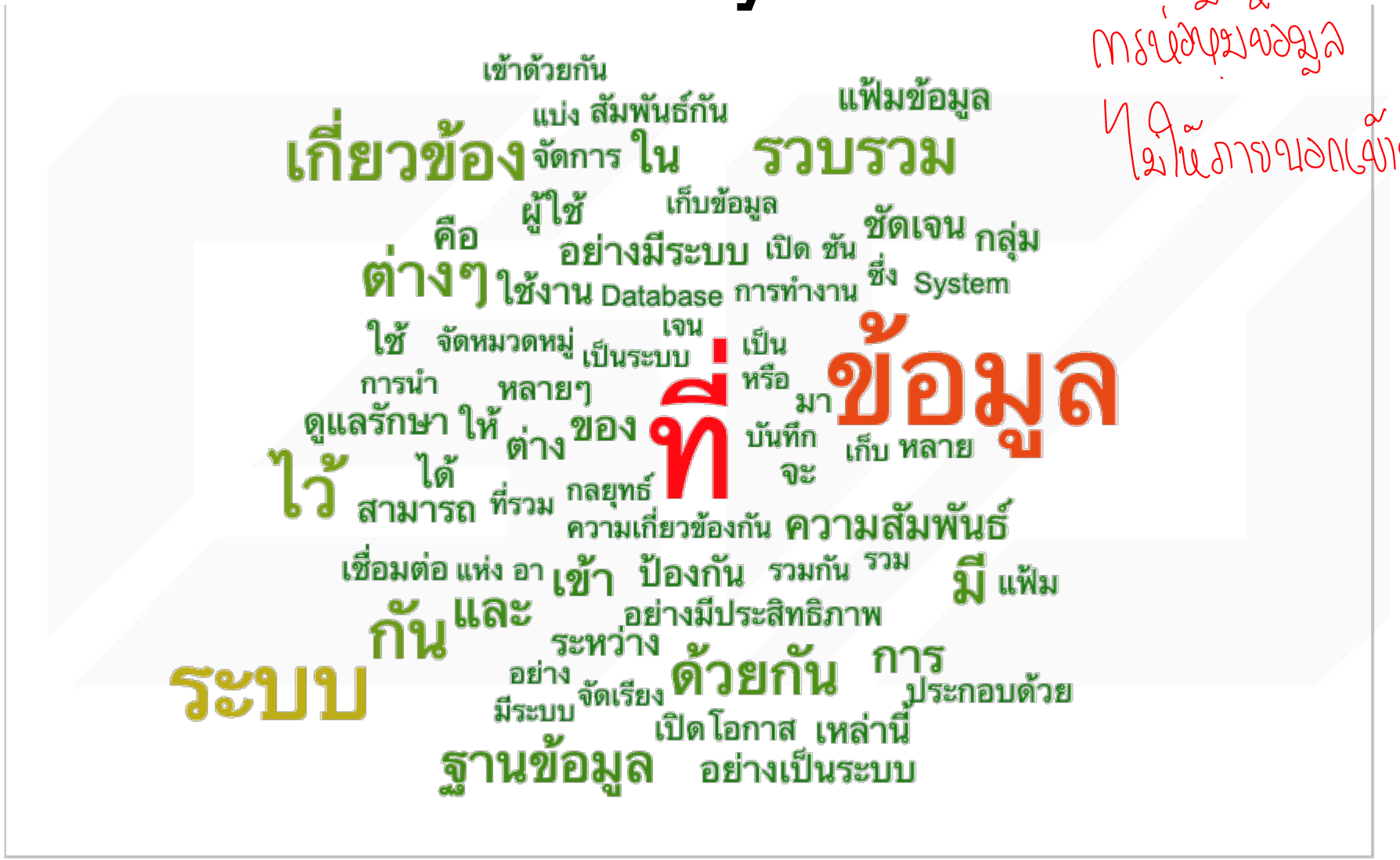
What are data?



What is a database?



What is a database system?



การแชร์ข้อมูล
ไม่ให้การขอเข้าถึงข้อมูลที่ไม่ได้รับอนุญาต

Basic Definitions

- **Data**:
  - Known facts that can be recorded and have an implicit meaning.
- **Database**:
  - A collection of related data.
- **Mini-world**:
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.
- **Database Management System (DBMS):**
  - A software package/ system to facilitate the creation and maintenance of a computerized database.
- **Database System**:
  - The DBMS software together with the data itself. Sometimes, the applications are also included.

Simplified database system environment

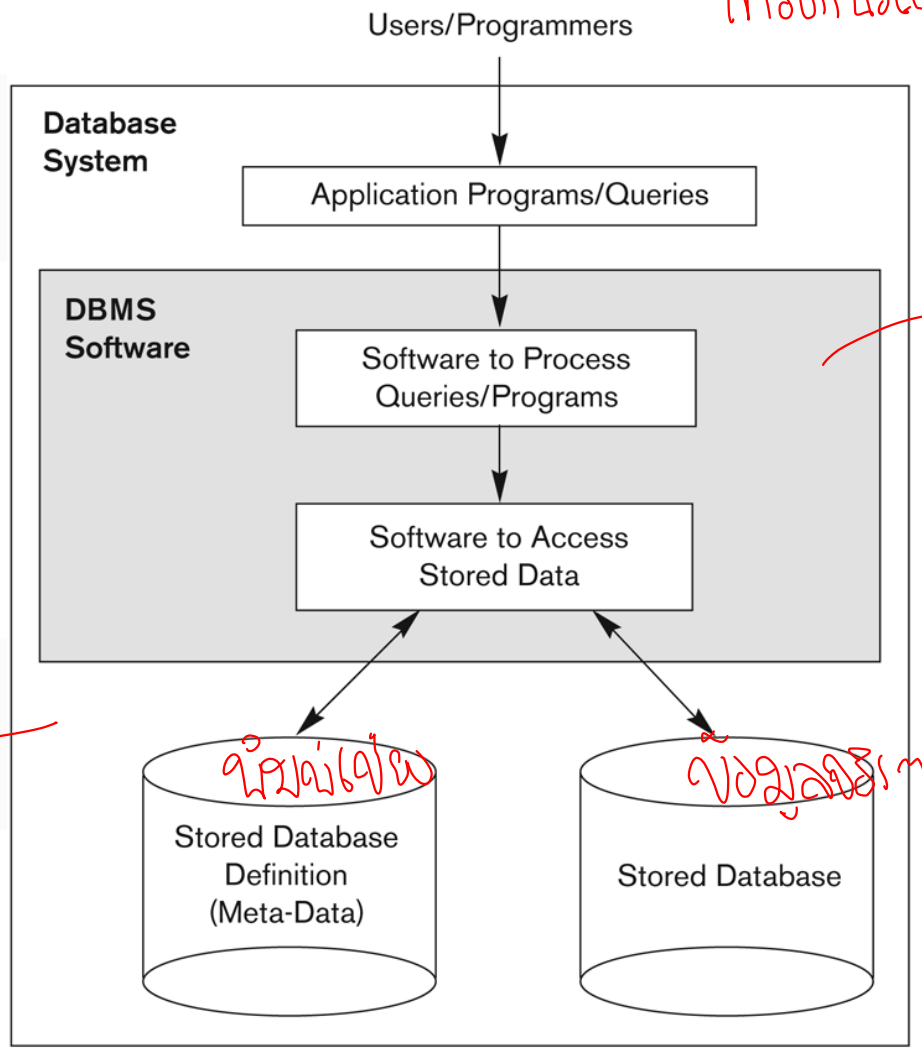


Figure 1.1
A simplified database system environment.

Typical DBMS Functionality → การทำงาน

- ***Define*** a particular database in terms of its data types, structures, and constraints → สร้างโครง → ชนิดตัวแปร → field → ชนิดข้อมูลต่างๆ
- ***Construct*** or load the initial database contents on a secondary storage medium → เริ่มการลงชื่อในครั้งแรกใน memory
- ***Manipulating*** the database: → การจัดการ db
  - Retrieval: Querying, generating reports → ดึงข้อมูล
  - Modification: Insertions, deletions and updates to its content → เปลี่ยนแปลง
  - Accessing the database through Web applications → เข้าถึง
- ***Processing*** and ***Sharing*** by a set of concurrent users and application programs – yet, keeping all data valid and consistent

สามารถร่วมการใช้งานได้หลายคน → ลบ, เพิ่ม, แก้ไข

Application Activities Against a Database

- Applications interact with a database by generating
  - Queries: that access different parts of data and formulate the result of a request
  - Transactions: that may read some data and "update" certain values or generate new data and store that in the database
- Applications must not allow unauthorized users to access data
- Applications must keep up with changing user requirements against the database

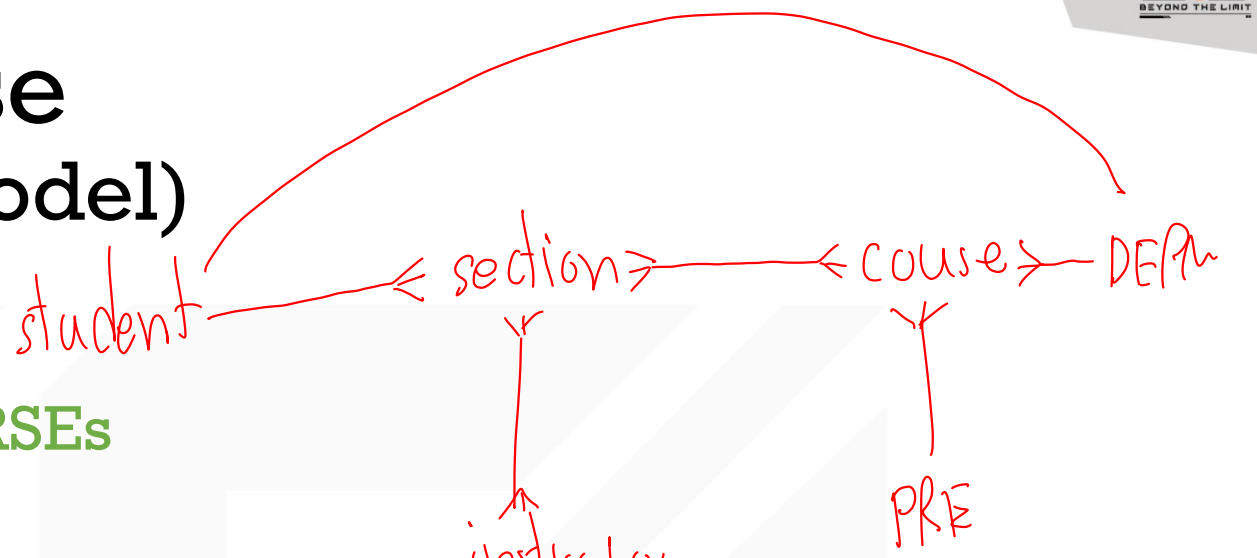
Additional DBMS Functionality

- DBMS may additionally provide:
  - Protection or Security measures to prevent unauthorized access
  - "Active" processing to take internal actions on data
  - Presentation and Visualization of data
  - Maintenance of the database and associated programs over the lifetime of the database application
    - Called database, software, and system maintenance

Example of a Database
(with a Conceptual Data Model)

- **Mini-world for the example:**
  - Part of a UNIVERSITY environment.
- **Some mini-world *entities*:**
  - STUDENTs
  - COURSEs
  - SECTIONs (of COURSEs)
  - (academic) DEPARTMENTs
  - INSTRUCTORs

Example of a Database
(with a Conceptual Data Model)



- **Some mini-world *relationships*:**
  - SECTIONs *are of specific* COURSEs
  - STUDENTs *take* SECTIONs
  - COURSEs *have prerequisite* COURSEs
  - INSTRUCTORs *teach* SECTIONs
  - COURSEs *are offered by* DEPARTMENTs
  - STUDENTs *major in* DEPARTMENTs

- Note:
  The above entities and relationships are typically expressed in a conceptual data model, such as the ENTITY-RELATIONSHIP data model

Example of a simple database

COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	04	King
92	CS1310	Fall	04	Anderson
102	CS3320	Spring	05	Knuth
112	MATH2410	Fall	05	Chang
119	CS1310	Fall	05	Anderson
135	CS3380	Fall	05	Stone

GRADE_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

Figure 1.2
A database that stores student and course information.

Main Characteristics of the Database Approach

- **Self-describing nature of a database system:**
  - A DBMS **catalog** stores the description of a particular database (e.g. data structures, types, and constraints)
  - The description is called **meta-data***.
  - This allows the DBMS software to work with different database applications.
- **Insulation between programs and data:**
  - Called **program-data independence**.
  - Allows changing data structures and storage organization without having to change the DBMS access programs.

---

* Some newer systems such as a few NOSQL systems need no meta-data: they store the data definition within its structure making it self describing

Example of a simplified database catalog

RELATIONS

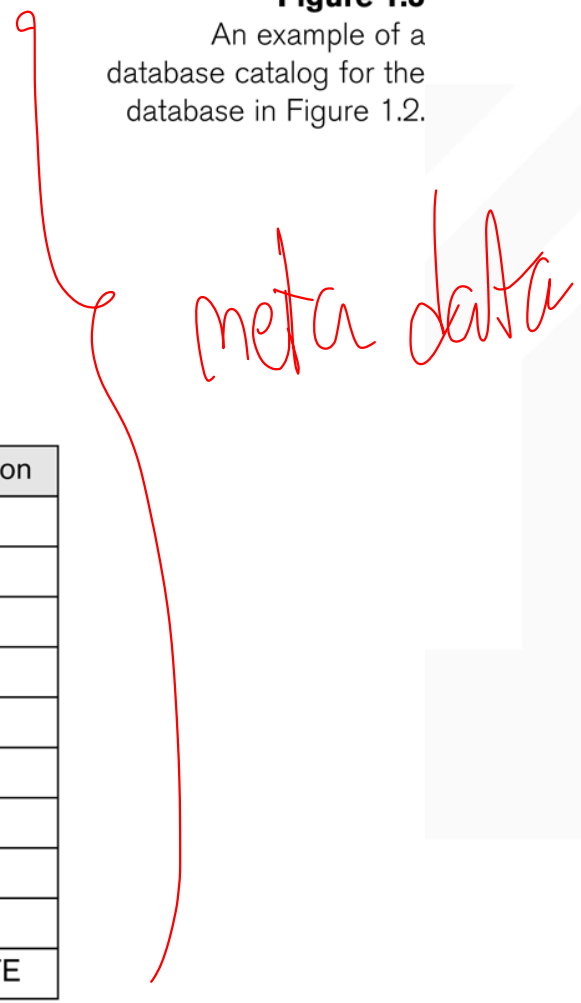
Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
….	….	…..
….	….	…..
….	….	…..
Prerequisite_number	XXXXNNNN	PREREQUISITE

Note: Major_type is defined as an enumerared type with all known majors. XXXXNNNN is used to define a type with four alpha characters followed by four digits

Figure 1.3
An example of a database catalog for the database in Figure 1.2.



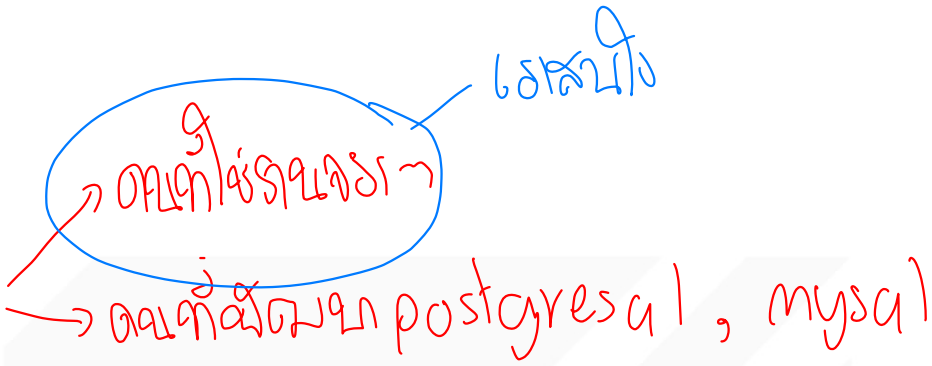
Main Characteristics of the Database Approach (continued)

- **Data Abstraction:**
  - A **data model** is used to hide storage details and present the users with a conceptual view of the database.
  - Programs refer to the data model constructs rather than data storage details
- **Support of multiple views of the data:**
  - Each user may see a different view of the database, which describes **only** the data of interest to that user.

Main Characteristics of the Database Approach (continued)

- **Sharing of data and multi-user transaction processing:**
  - Allowing a set of **concurrent users** to retrieve from and to update the database.
  - ***Concurrency control*** within the DBMS guarantees that each **transaction** is correctly executed or aborted
  - ***Recovery*** subsystem ensures each completed transaction has its effect permanently recorded in the database
  - **OLTP** (Online Transaction Processing) is a major part of database applications. This allows hundreds of concurrent transactions to execute per second.

Database Users



- Users may be divided into
  - Those who actually use and control the database content, and those who design, develop and maintain database applications (called "Actors on the Scene"), and
  - Those who design and develop the DBMS software and related tools, and the computer systems operators (called "Workers Behind the Scene").

Database Users – Actors on the Scene

- Actors on the scene
  - **Database administrators**:
    - Responsible for authorizing access to the database, for coordinating and monitoring its use, acquiring software and hardware resources, controlling its use and monitoring efficiency of operations.
  - **Database Designers**:
    - Responsible to define the content, the structure, the constraints, and functions or transactions against the database. They must communicate with the end-users and understand their needs.

Database End Users

- Actors on the scene (continued)
  - **End-users**: They use the data for queries, reports and some of them update the database content. End-users can be categorized into:
    - **Casual**: access database occasionally when needed
    - **Naïve** or **Parametric**: they make up a large section of the end-user population.
      - They use previously well-defined functions in the form of "canned transactions" against the database.
      - Users of Mobile Apps mostly fall in this category
      - Bank-tellers or reservation clerks are parametric users who do this activity for an entire shift of operations.
      - Social Media Users post and read information from websites

Database End Users (continued)

- **Sophisticated**:
  - These include business analysts, scientists, engineers, others thoroughly familiar with the system capabilities.
  - Many use tools in the form of software packages that work closely with the stored database.
- **Stand-alone**:
  - Mostly maintain personal databases using ready-to-use packaged applications.
  - An example is the user of a tax program that creates its own internal database.
  - Another example is a user that maintains a database of personal photos and videos.

Database Users – Actors on the Scene (continued)

- **System Analysts and Application Developers**

  This category currently accounts for a very large proportion of the IT work force.

  - System Analysts: They understand the user requirements of naïve and sophisticated users and design applications including canned transactions to meet those requirements.
  - Application Programmers: Implement the specifications developed by analysts and test and debug them before deployment.
  - Business Analysts: There is an increasing need for such people who can analyze vast amounts of business data and real-time data ("Big Data") for better decision making related to planning, advertising, marketing etc.

Database Users – Actors behind the Scene

- **System Designers and Implementors:** Design and implement DBMS packages in the form of modules and interfaces and test and debug them. The DBMS must interface with applications, language compilers, operating system components, etc.
- **Tool Developers:** Design and implement software systems called tools for modeling and designing databases, performance monitoring, prototyping, test data generation, user interface creation, simulation etc. that facilitate building of applications and allow using database effectively.
- **Operators and Maintenance Personnel:** They manage the actual running and maintenance of the database system hardware and software environment.

Historical Development of Database Technology

ประวัติความเป็นมา

- Early Database Applications:
  - The Hierarchical and Network Models were introduced in mid 1960s and dominated during the seventies.
  - A bulk of the worldwide database processing still occurs using these models, particularly, the hierarchical model using IBM's IMS system.
- Relational Model based Systems:
  - Relational model was originally introduced in 1970, was heavily researched and experimented within IBM Research and several universities.
  - Relational DBMS Products emerged in the early 1980s.

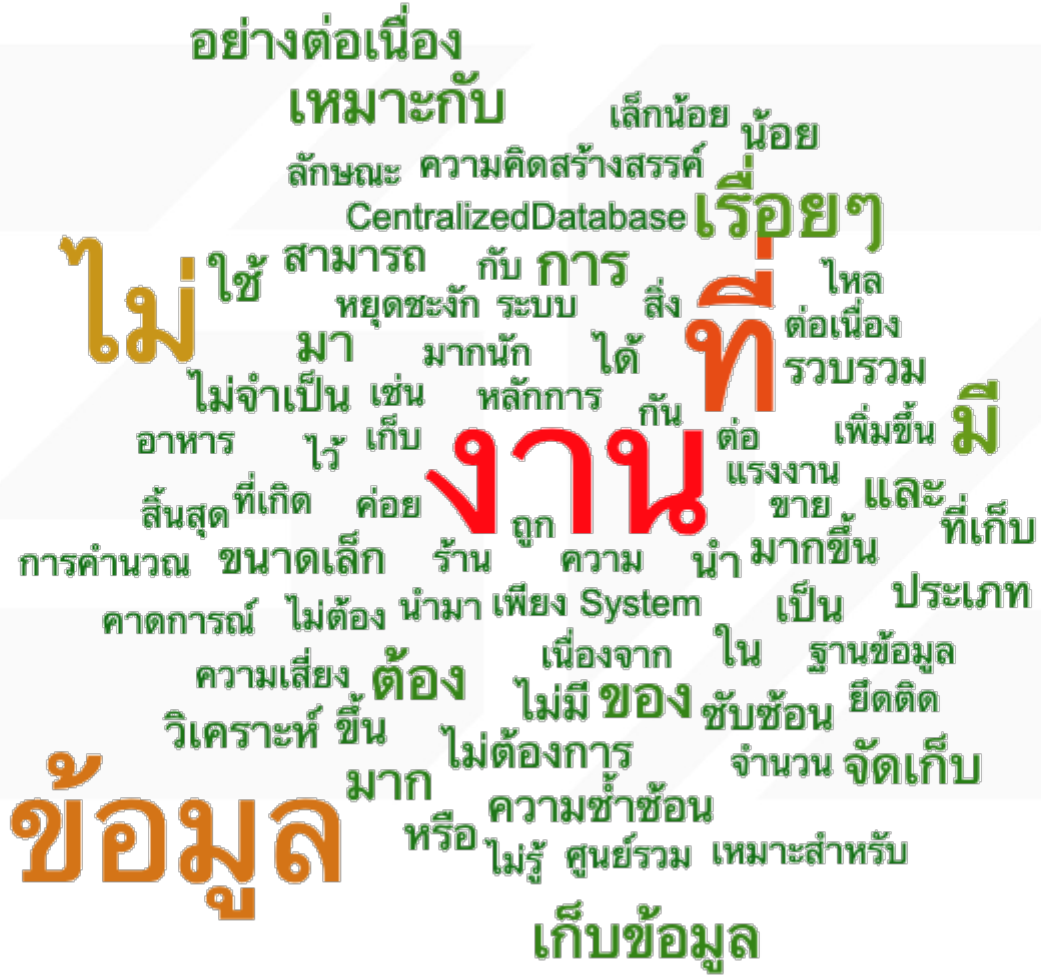
Historical Development of Database Technology (continued)

- Object-oriented and emerging applications:
  - Object-Oriented Database Management Systems (OODBMSs) were introduced in late 1980s and early 1990s to cater to the need of complex data processing in CAD and other applications.
    - Their use has not taken off much.
  - Many relational DBMSs have incorporated object database concepts, leading to a new category called *object-relational* DBMSs (ORDBMSs)
  - Extended relational systems add further capabilities (e.g. for multimedia data, text, XML, and other data types)

When not to use a DBMS

- Main inhibitors (costs) of using a DBMS:
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- When a DBMS may be unnecessary:
  - If the database and applications are simple, well defined, and not expected to change.
  - If access to data by multiple users is not required.
- When a DBMS may be infeasible:
  - In embedded systems where a general-purpose DBMS may not fit in available storage

When not to use a DBMS?



NoSQL vs SQL Databases

TLDR: NoSQL ("non SQL" or "not only SQL") databases were developed in the late 2000s with a focus on scaling, fast queries, allowing for frequent application changes, and making programming simpler for developers. Relational databases accessed with SQL (Structured Query Language) were developed in the 1970s with a focus on reducing data duplication as storage was much more costly than developer time. SQL databases tend to have rigid, complex, tabular schemas and typically require expensive vertical scaling.

If you're not familiar with what NoSQL databases are or the different types of NoSQL databases, start here.

Overview

Below is an overview of what this article covers.

Differences between SQL and NoSQL

The table below summarizes the main differences between SQL and NoSQL databases.

	SQL Databases	NoSQL Databases
Data Storage Model	Tables with fixed rows and columns	Document: JSON documents, Key-value: key-value pairs, Wide-column: tables with rows and dynamic columns, Graph: nodes and edges
Development History	Developed in the 1970s with a focus on reducing data duplication	Developed in the late 2000s with a focus on scaling and allowing for rapid application change driven by agile and DevOps practices.

Examples	Oracle, MySQL, Microsoft SQL Server, and PostgreSQL	Document: MongoDB and CouchDB, Key-value: Redis and DynamoDB, Wide-column: Cassandra and HBase, Graph: Neo4j and Amazon Neptune
Primary Purpose	General purpose	Document: general purpose, Key-value: large amounts of data with simple lookup queries, Wide-column: large amounts of data with predictable query patterns, Graph: analyzing and traversing relationships between connected data
Schemas	Rigid	Flexible
Scaling	Vertical (scale-up with a larger server)	Horizontal (scale-out across commodity servers)
Multi-Record ACID Transactions	Supported	Most do not support multi-record ACID transactions. However, some—like MongoDB—do.
Joins	Typically required	Typically not required
Data to Object Mapping	Requires ORM (object-relational mapping)	Many do not require ORMs. MongoDB documents map directly to data structures in most popular programming languages.

What are the Benefits of NoSQL Databases?

NoSQL databases offer many benefits over relational databases. NoSQL databases have flexible data models, scale horizontally, have incredibly fast queries, and are easy for developers to work with.

- **Flexible data models**

  NoSQL databases typically have very flexible schemas. A flexible schema allows you to easily make changes to your database as requirements change. You can iterate quickly and continuously integrate new application features to provide value to your users faster.

- **Horizontal scaling**

  Most SQL databases require you to scale-up vertically (migrate to a larger, more expensive server) when you exceed the capacity requirements of your current server. Conversely, most NoSQL databases allow you to scale-out horizontally, meaning you can add cheaper, commodity servers whenever you need to.

- **Fast queries**

  Queries in NoSQL databases can be faster than SQL databases. Why? Data in SQL databases is typically normalized, so queries for a single object or entity require you to join data from multiple tables. As your tables grow in size, the joins can become expensive. However, data in NoSQL databases is typically stored in a way that is optimized for queries. The rule of thumb when you use MongoDB is **Data is that is accessed**

together should be stored together. Queries typically do not require joins, so the queries are very fast.

- **Easy for developers**

  Some NoSQL databases like MongoDB map their data structures to those of popular programming languages. This mapping allows developers to store their data in the same way that they use it in their application code. While it may seem like a trivial advantage, this mapping can allow developers to write less code, leading to faster development time and fewer bugs.

What are the Drawbacks of NoSQL Databases?

One of the most frequently cited drawbacks of NoSQL databases is that they don't support ACID (atomicity, consistency, isolation, durability) transactions across multiple documents. With appropriate schema design, single record atomicity is acceptable for lots of applications. However, there are still many applications that require ACID across multiple records.

To address these use cases MongoDB added support for multi-document ACID transactions in the 4.0 release, and extended them in 4.2 to span sharded clusters.

Since data models in NoSQL databases are typically optimized for queries and not for reducing data duplication, NoSQL databases can be larger than SQL databases. Storage is currently so cheap that most consider this a minor drawback, and some NoSQL databases also support compression to reduce the storage footprint.

Depending on the NoSQL database type you select, you may not be able to achieve all of your use cases in a single database. For example, graph databases are excellent for analyzing relationships in your data but may not provide what you need for everyday retrieval of the data such as range queries. When selecting a NoSQL database, consider what your use cases will be and if a general purpose database like MongoDB would be a better option.

How to Try a NoSQL Database

Now that you understand the basics of NoSQL databases, you're ready to give them a shot.

You can check out the Where to Use MongoDB whitepaper to help you determine if MongoDB or another database is right for your use case. Then hop on over to What is a Document Database? to learn about the document model and how it compares to the relational model.

For those who like to jump right in and learn by doing, one of the easiest ways to get started with NoSQL databases is to use MongoDB Atlas. Atlas is MongoDB's fully managed, global database service that is available on all of the leading cloud providers. One of the many handy things about Atlas is that it has a generous, forever-free tier so you can create a database and discover all of the benefits of NoSQL databases first hand without providing your credit card. If you'd like to try a paid tier, apply code NOSQLEXPLAINED for $200 of Atlas credits.

For those who prefer structured learning, MongoDB University is completely free online training that will walk you step-by-step through the process of learning MongoDB.

When you're ready to interact with MongoDB using your favorite programming language, check out the Quick Start Tutorials. These tutorials will help you get up and running as quickly as possible in the language of your choice.

This article was written by Lauren Schaefer, a MongoDB Developer Advocate.