พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Database Systems

Pakorn Watanachaturaporn
Pakorn.Wa@KMITL.ac.th

Akkradach Watcharapupong
Akkradach.Wa@KMITL.ac.th

Bachelor Program in Computer Engineering (B.Eng.)
Faculty of Engineering

King Mongkut's Institute of Technology Ladkrabang

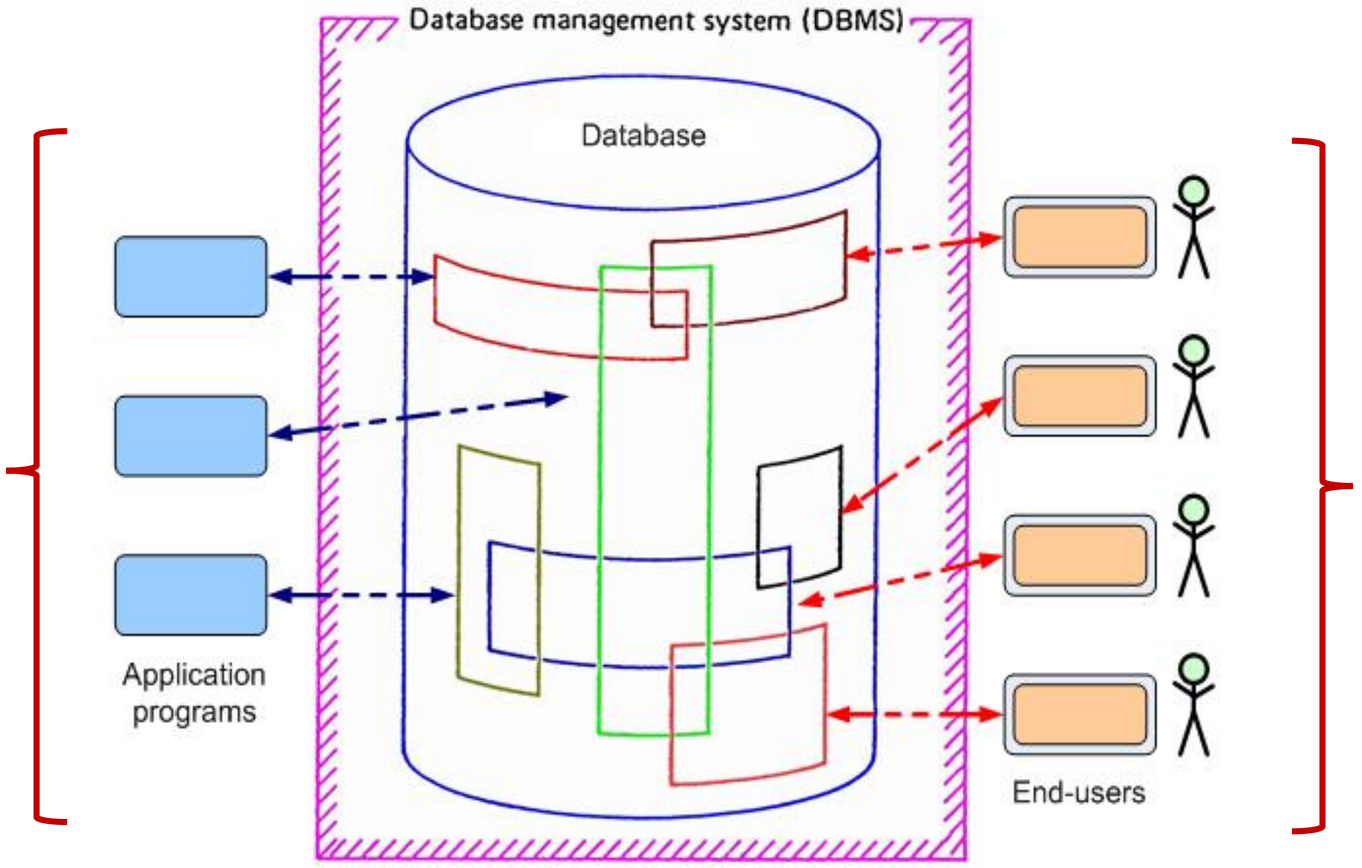
Information Systems and Databases

- A **database model** or **data model**
  - A method to present **collections of facts** and **relationships** among them.
  - Modern … present **objects** and **relationships** …
  - Use easy representation such as tables, trees, and graphs; *a.k.a.*, **logical database models** since they are perceived at the logical level.
    - *E.g.*,
      - Relational database model
      - Hierarchical database model
      - Network database model
      - Object database model
      - Object relational database model

Typical Database Systems

pakorn.wa@kmitl.ac.th

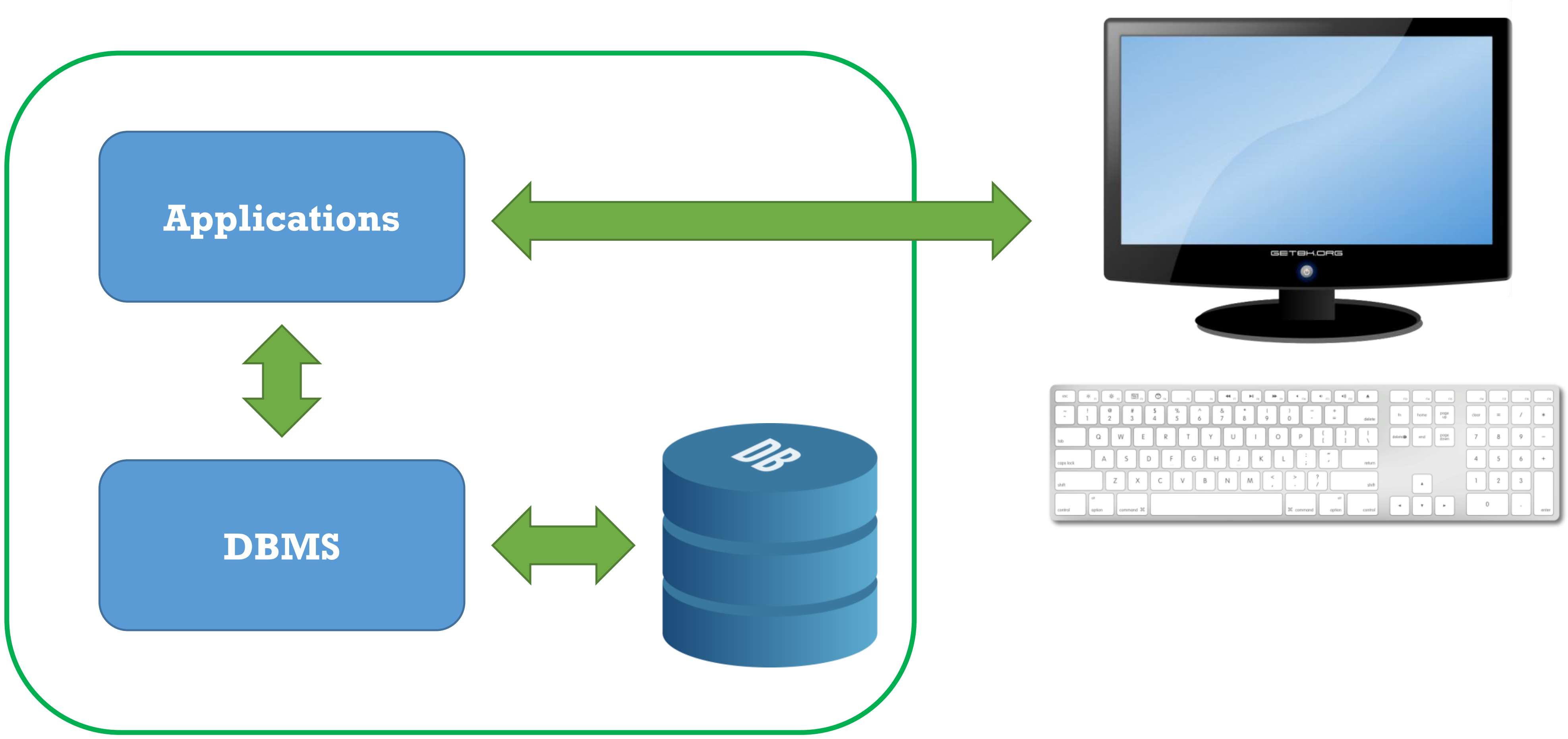
Embedded Mode



Interactive Mode

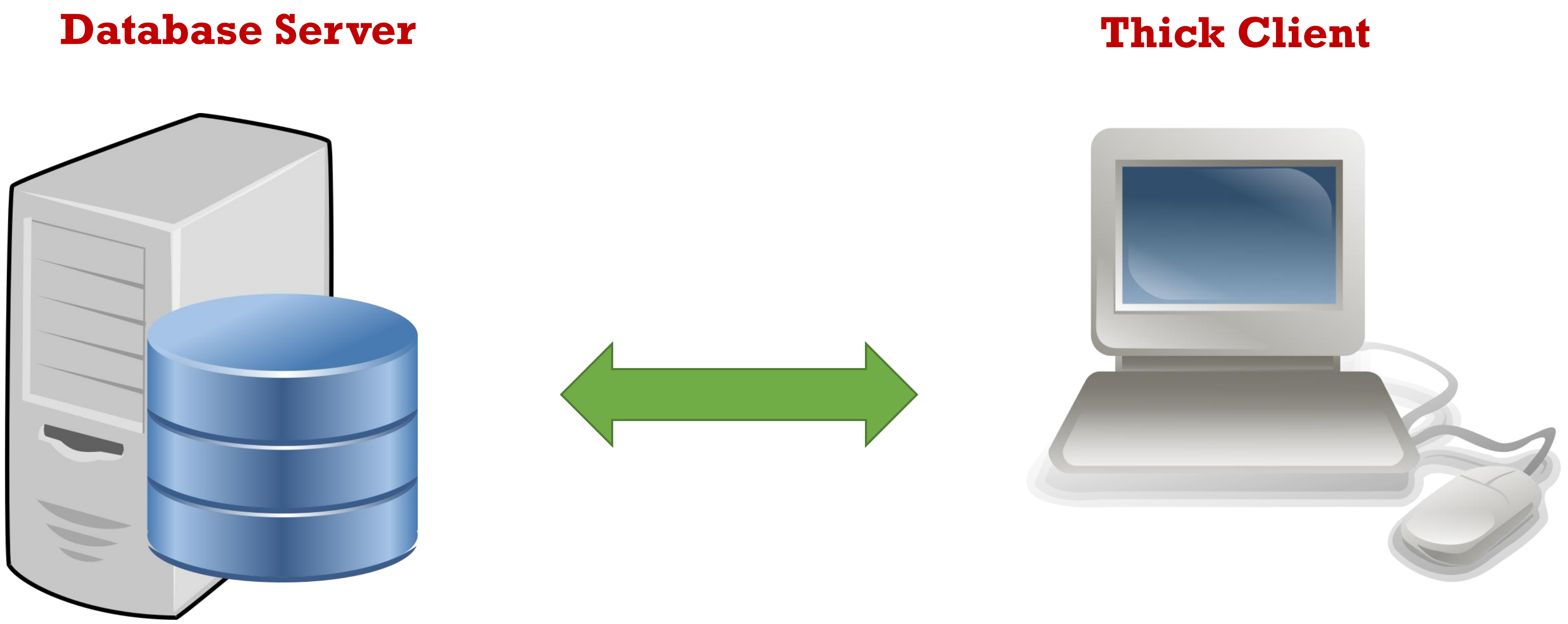
Simplified picture of a database system

Centralized Host



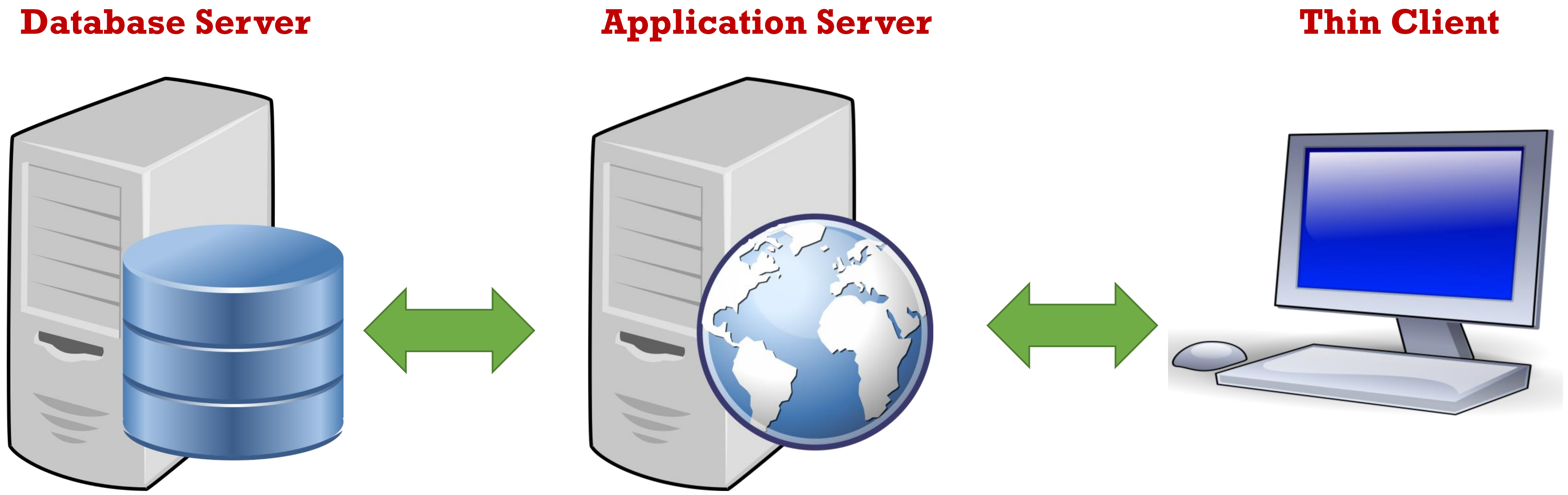
pakorn.wa@kmitl.ac.th

Two-tier Client/Server



Three-tier Client/Server

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



Sample Relational Database Tables

pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

Relation

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

Relation

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

Relation

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- **Relation**
  - *A relation is a subset of the Cartesian product of domains.*
  - The most popular representation is the tabular format.
  - Note:
    The term *relation* in the relational database model does not refer to *relationships* between tables. Instead, the *table* itself is a relation.

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

- Standard Relational Language
  - Structured Query Language (SQL)
  - Structured *English* Query Language (SEQUEL)

- ***E.g.,*** Suppose we want to find information of vendors whose status are 100 and location are in Bangkok.

```
SELECT *
FROM   VENDOR
WHERE  LOCATION = 'Bangkok' AND
       STATUS = 100;
```

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

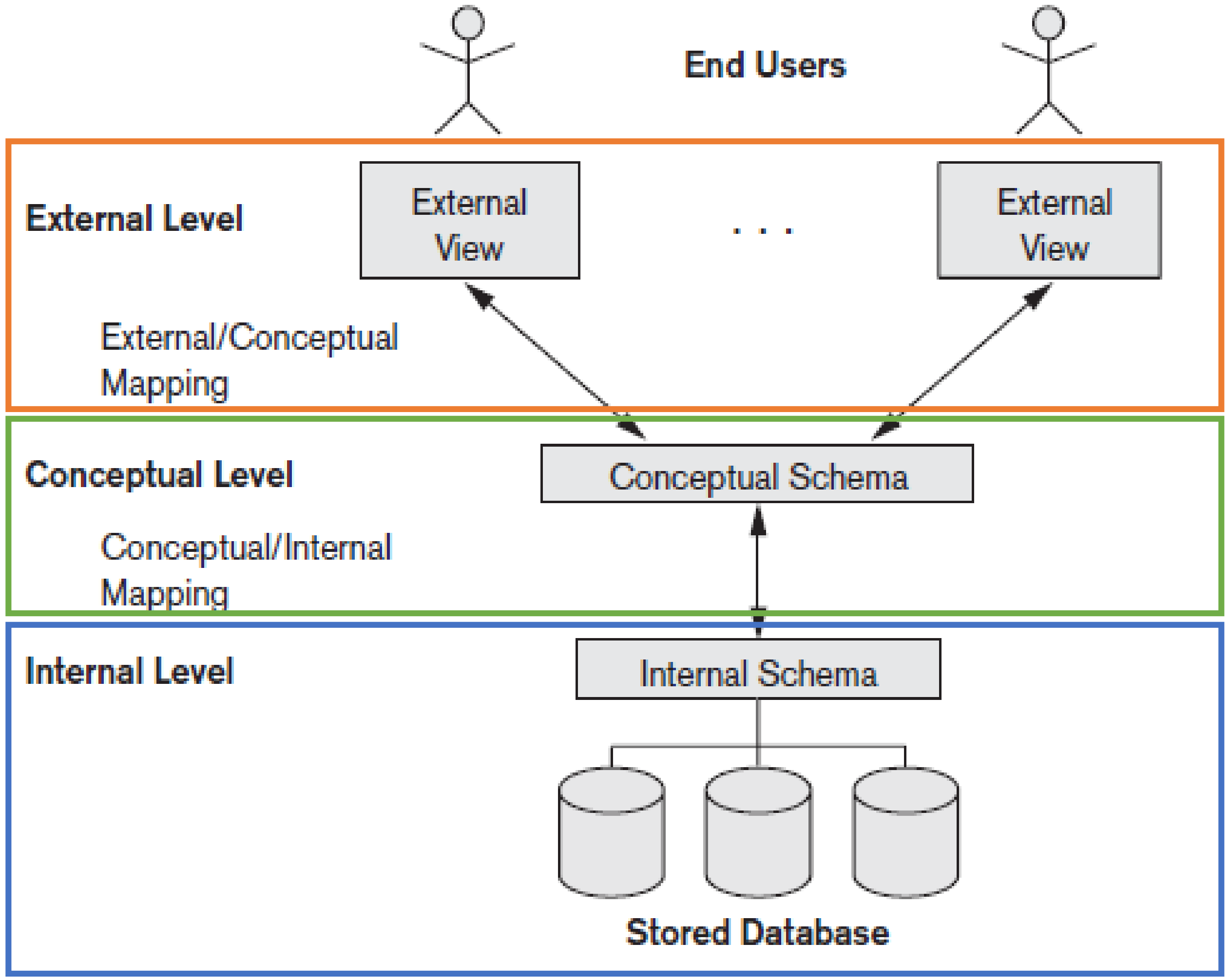
SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

- SQL
  - The table is a logical database table.
  - The SQL query is also a logical level query.
  - An SQL statement is considered a definition of the required result(s).

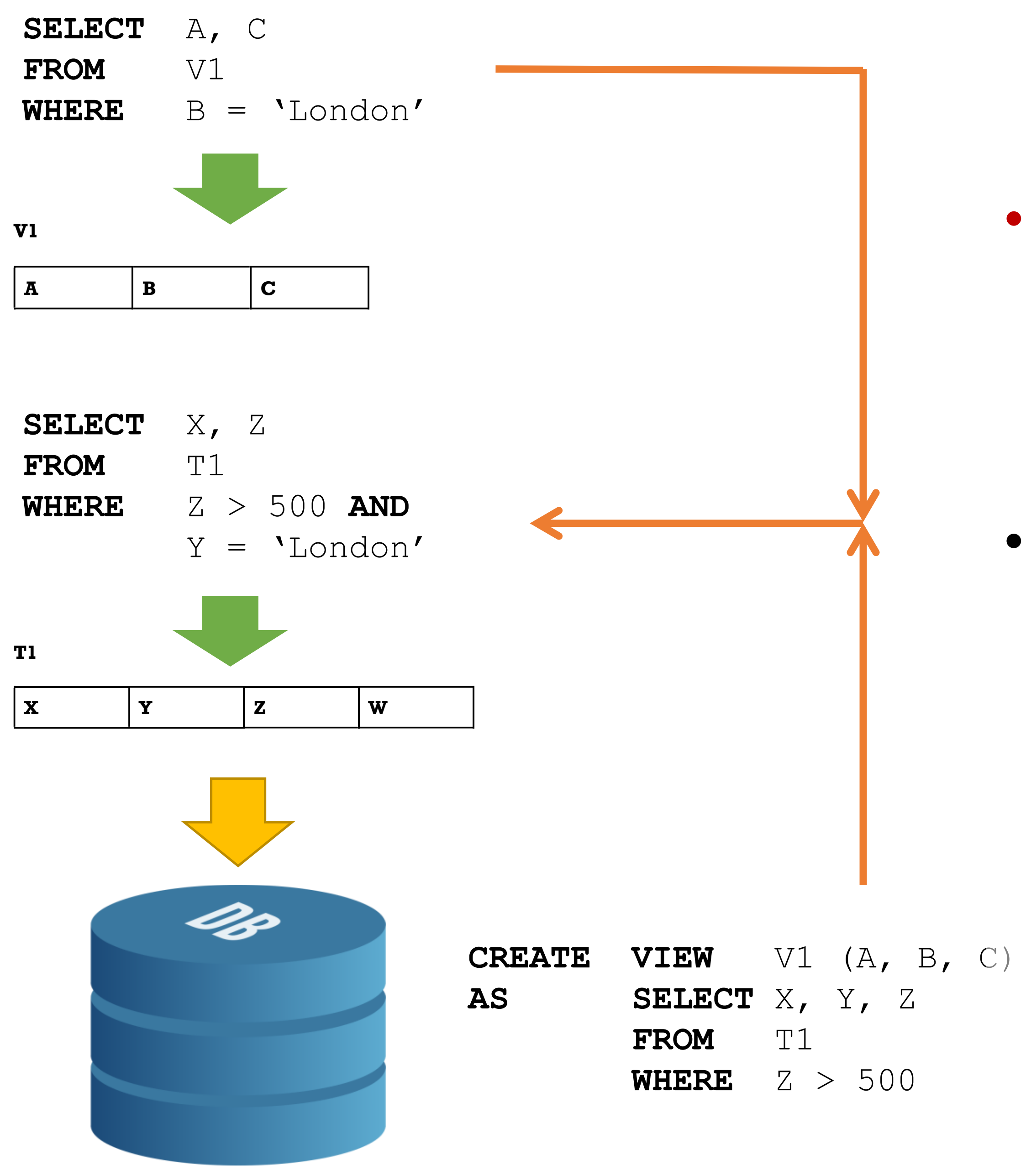
The Three-Schema Architecture

pakorn.wa@kmitl.ac.th



- External Level
  - A number of external schemas or user views.
  - Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database.
- Conceptual Level (Logical Level)
  - Describe the structure of the whole database for a community of users.
  - Hide the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints.
- Internal Level
  - Describe the physical storage structure of the database.
  - Use a physical model and describe the complete details of data storage and access paths for the database.

pakorn.wa@kmitl.ac.th



```
SELECT   A, C
FROM     V1
WHERE    B = 'London'
```

V1

A	B	C

```
SELECT   X, Z
FROM     T1
WHERE    Z > 500 AND
         Y = 'London'
```

T1

X	Y	Z	W

```
CREATE   VIEW    V1 (A, B, C)
AS       SELECT  X, Y, Z
         FROM    T1
         WHERE   Z > 500
```

- View
  - When a view is created, only the view definition is kept in the system area of the database.
  - Generally, called the system catalogs or system tables.
- From the figure,
  - The view definition of V1 is kept in the system catalogs.
  - The DBA who creates views may grant view access and manipulation permissions to programmers and users.
  - After the permissions are granted they are able to use the view V1 as if it was a table without an awareness of the existence of T1.

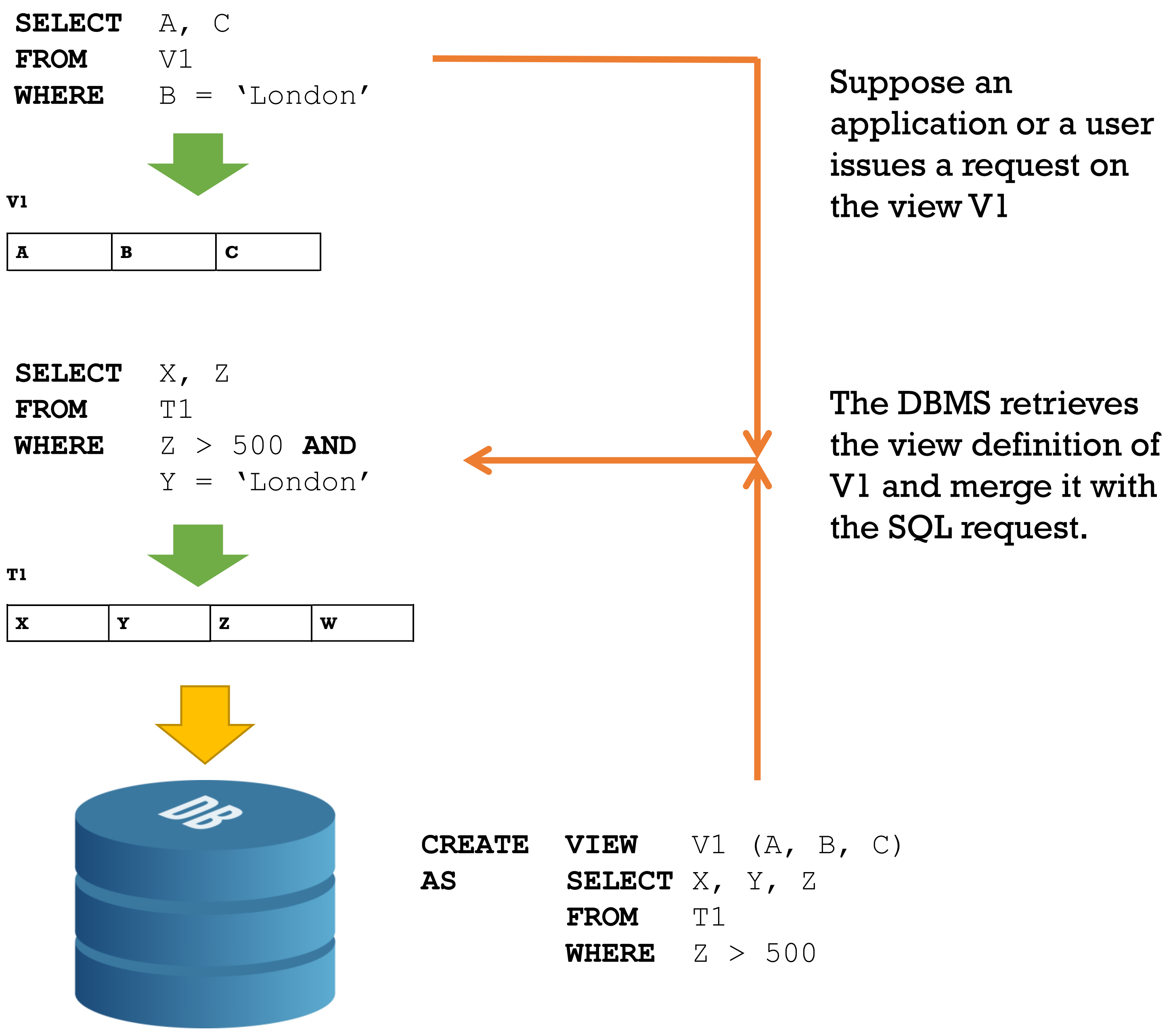
- Two major advantages of using views
  - Privacy
    - An application sees the permitted data only.
    - Data which are irrelevant to the application are hidden from the application.
  - Logical data independence
    - "data independence" means database structures can be modified without the modification of the application programs that refer to them.
    - Logical data independence allows logical-level data to be modified without the need to modify the external-level data which is seen by the applications.

Space vs. Speed

- Conventionally, NO!
- However, some DBMSs allow the creation of views which store redundant snapshot portions of base tables, called a ***materialized view***.
- In our class, "views" are referred to non-materialized views if not mentioned otherwise.

- Yes, but should be very slightly for the response time.

pakorn.wa@kmitl.ac.th



```
SELECT   A, C
FROM     V1
WHERE    B = 'London'
```

V1

A	B	C

```
SELECT   X, Z
FROM     T1
WHERE    Z > 500 AND
         Y = 'London'
```

T1

X	Y	Z	W

```
CREATE   VIEW    V1 (A, B, C)
AS       SELECT  X, Y, Z
         FROM    T1
         WHERE   Z > 500
```

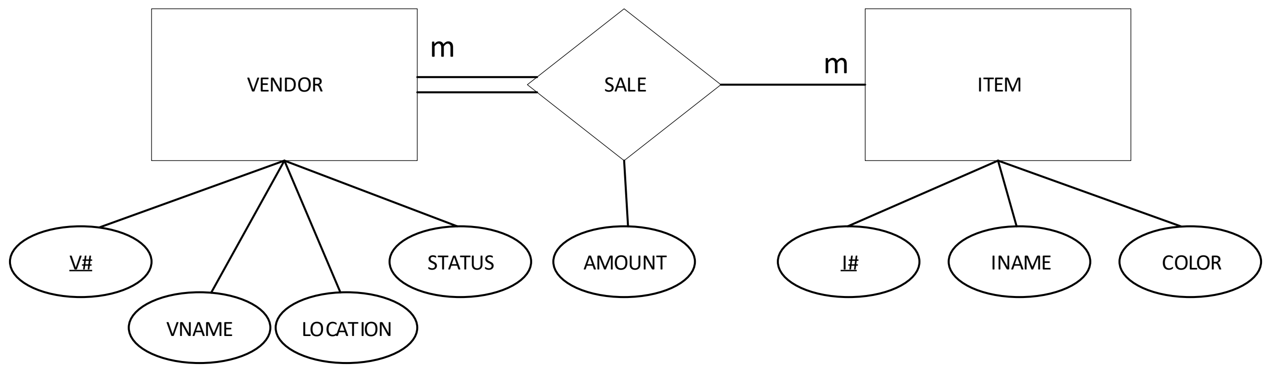
Suppose an application or a user issues a request on the view V1

The DBMS retrieves the view definition of V1 and merge it with the SQL request.

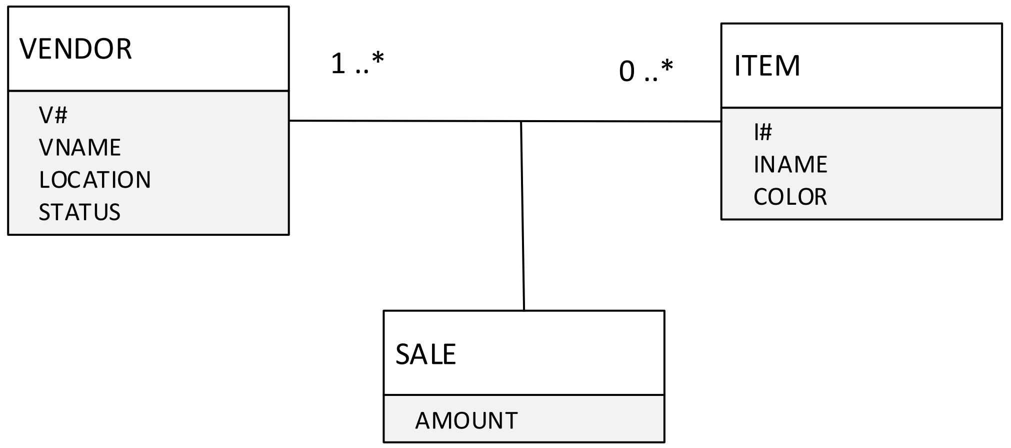
- Thus, a query to the view is slower (with exception) since the translation from a query on views to a query on tables by merging it with the view definition is required.
- In principle, the DBMS should translate the input request each time to preserve **physical data independence**.
- However, the availability of the query buffer and stored query mechanism in some DBMSs help reduced the repeated translation if a query is found to already exist in the buffer.

Conceptual Schema

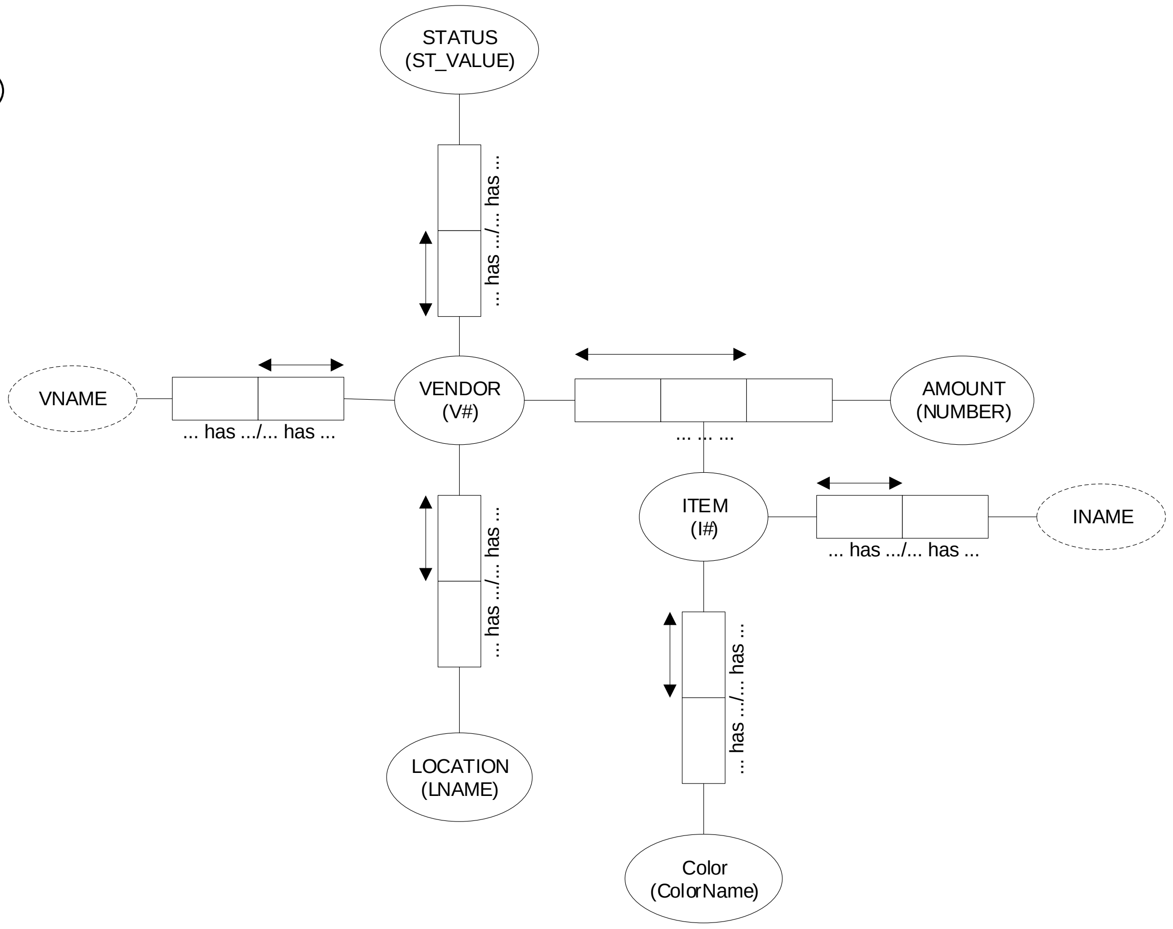
- Conceptual level or Semantic level
  - More natural representations of the data and relationships.
  - A higher level of abstraction on data and relationships.
  - Emphasize on objects (or entities) of interest and relationships among them and provides explicit symbols which distinguish them apart.
  - Conceptual schema diagrams
    - Entity Relationship Model (ER)
    - Unified Modeling Language (UML)
    - Object Role Model (ORM)



ER Diagram

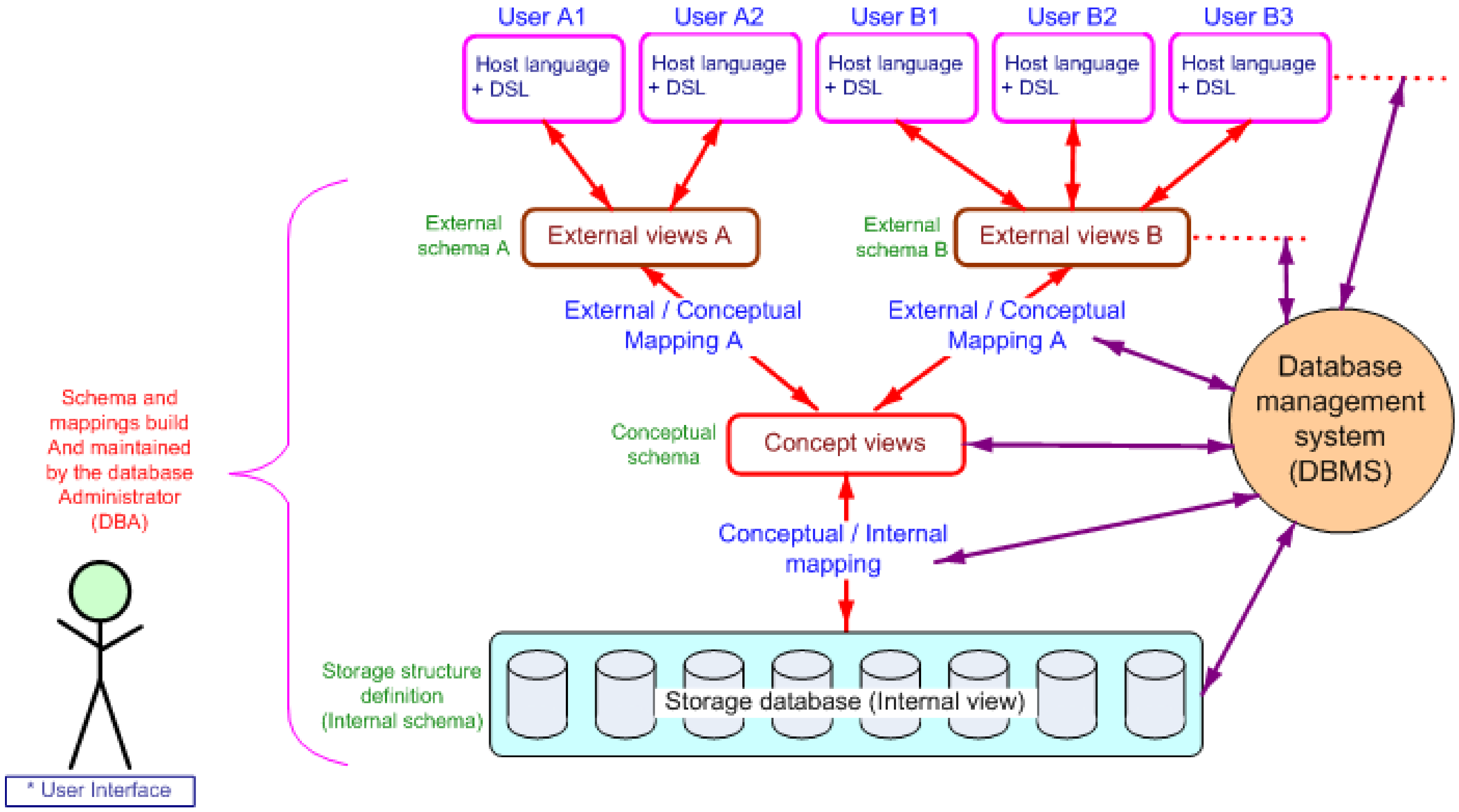


UML Class Diagram



ORM Conceptual Schema

- Conceptual models are used to represent reality, actual objects of interest and relationships among them.
- Once agreed, the conceptual diagram designed can be later transformed to logical data structures for programming purpose.


User A1
User A2
User B1
User B2
User B3
Host language + DSL
Host language + DSL
Host language + DSL
Host language + DSL
Host language + DSL
External schema A
External views A
External schema B
External views B
External / Conceptual Mapping A
External / Conceptual Mapping A
Conceptual schema
Concept views
Database management system (DBMS)
Conceptual / Internal mapping
Storage structure definition (Internal schema)
Storage database (Internal view)
Schema and mappings build And maintained by the database Administrator (DBA)
* User Interface

DBMS Major Functions

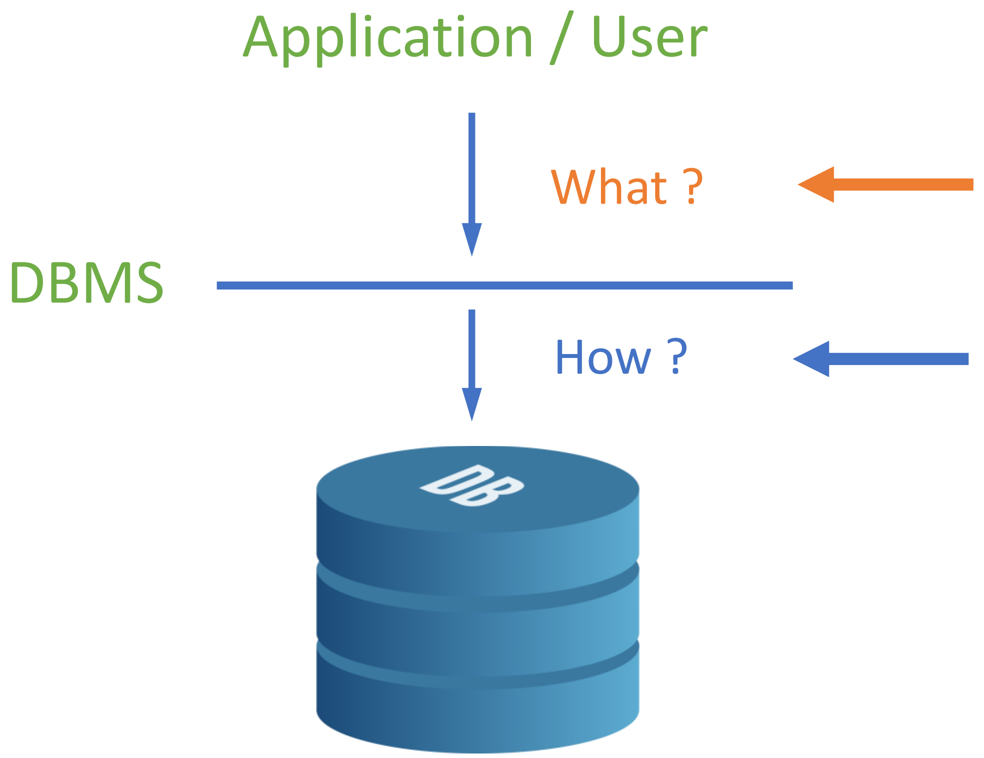
pakorn.wa@kmitl.ac.th

- Data abstraction and high-level data manipulation language support
  - Logical data structure
  - Integrity constrain definition and enforcement
  - Data definition and data manipulation languages
  - Data abstraction and hide physical storage organization and access mechanism

- Data sharing with multiple views
  - Allow logical data sharing in the multiuser environment.
  - Each application works on the external view which is relevant to the applications.
  - Extension to the underlying logical structure have no or minimum effect to the external views and application codes.

pakorn.wa@kmitl.ac.th

- Query processing and optimization
  - High level database queries of the relational database model are of the ‘what’ type. They define the results that the users want, not how to obtain the result.



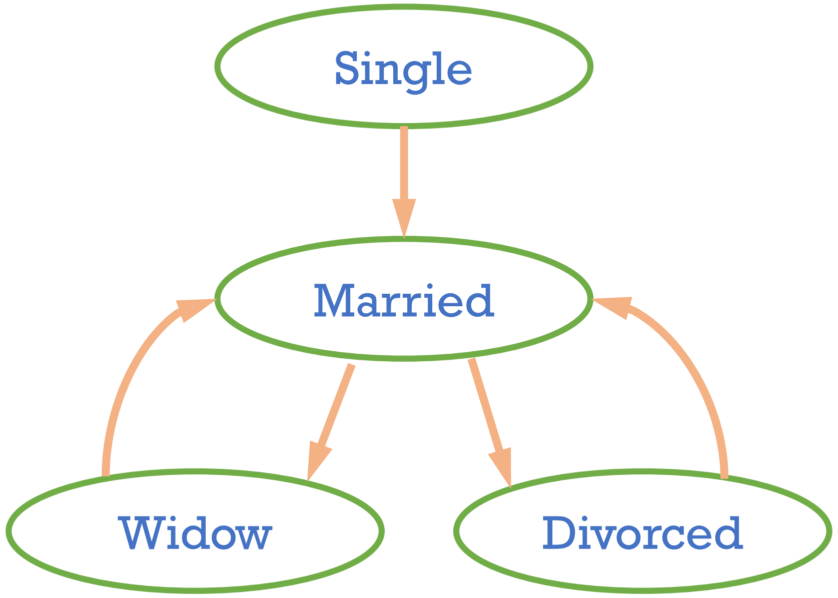
- Definition of results
- Relational Calculus
- SQL

- Relational algebra
- E.g., union, intersection, difference, product, select, project, join, divide, …

pakorn.wa@kmitl.ac.th

- Database integrity constraints enforcement
  - A basic assumption about databases is the data in the database are assumed to be correct with respect to some validation rules or integrity constrains.
    - Integrity constraints on the data structure: primary key and referential integrity constraints of the relational database model.
    - Application based constraints: *e.g.*, the maximum number of books a member can borrow from a library.

- Database integrity constraints enforcement (Cont'd)
  - Static integrity rules
    - E.g.,
      - not allow '30 February'
      - $65 \geq$ Age ≥ 18
  - Dynamic integrity rules



- Transaction processing
  - A transaction is a logical unit of work (LUW) in which integrity constraints are allowed to be violated inside.
  - An LUW comprises of one or more database requests. During a transaction execution, application-related integrity constraints or validation rules may be temporarily violated.

- Transaction processing (Cont'd)
  - *E.g.*, A company has an integrity rule (R1) states that a company car must have officers in charge. Assume that one table for cars, and one table for officers' assignment to cars.

Begin transaction

1. Insert the new car's information into the car table

   ----- *R1* is temporarily violated

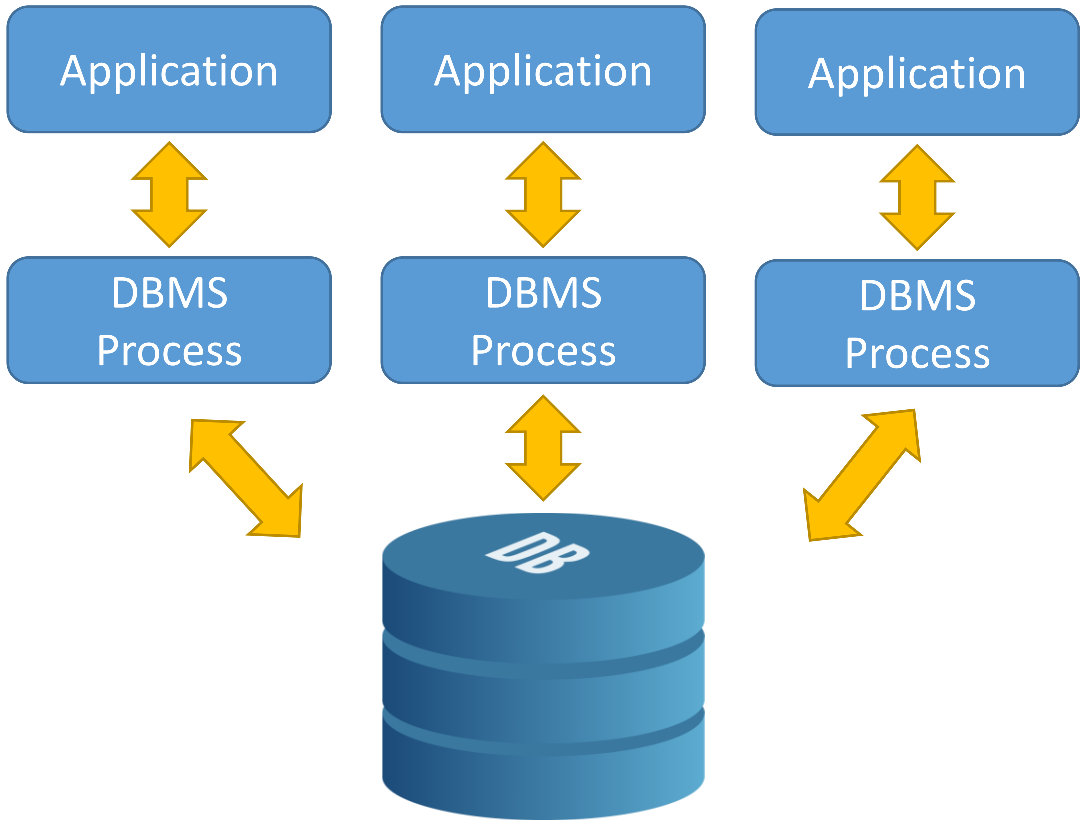
2. Insert the officer assignment to the new car

End transaction

- Centralized data management
  - To accommodate data sharing by multiusers and many application concurrently.
  - The DBMS keeps database descriptions, integrity constraints, users' information, access permission and authorization, and other system information in the system catalog. The information in the system catalog is called the metadata.
  - The database administrator (DBA) is responsible for database structures creation in all the three levels : the external schema, conceptual/logical schema, and the physical database schema.

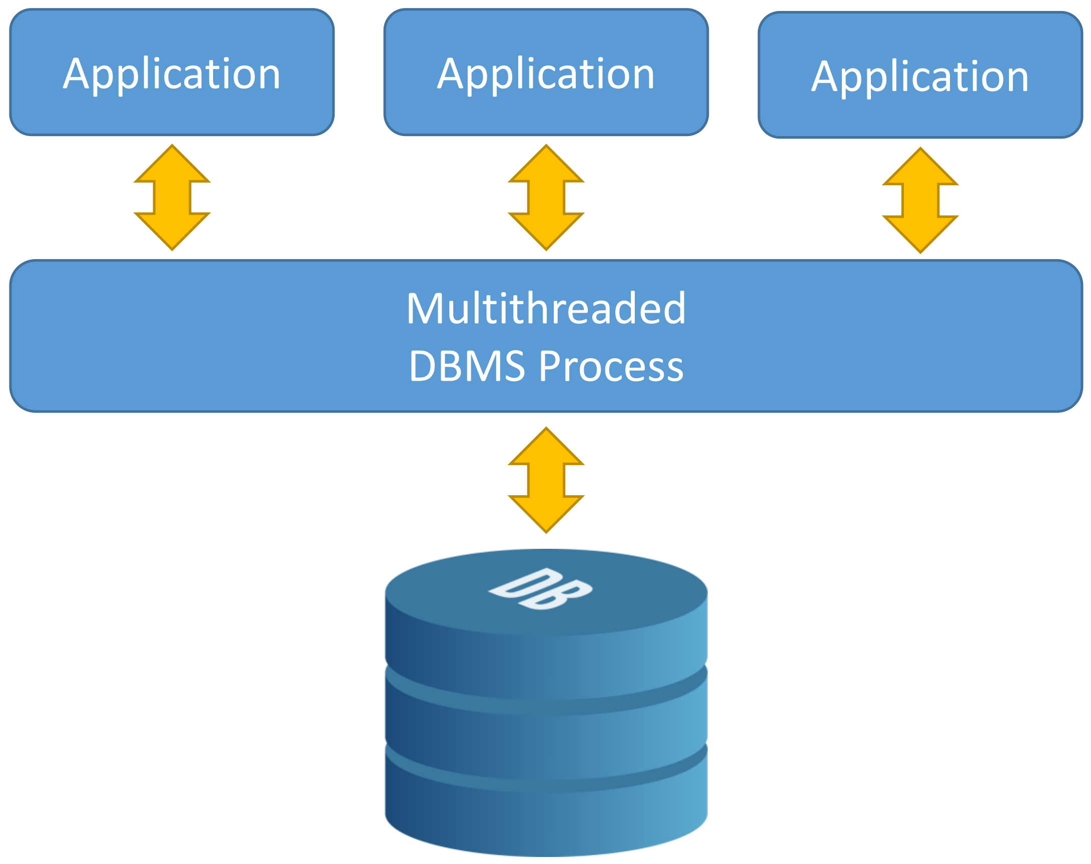
DBMS Process Architecture

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



DBMS **Process Model** Architecture

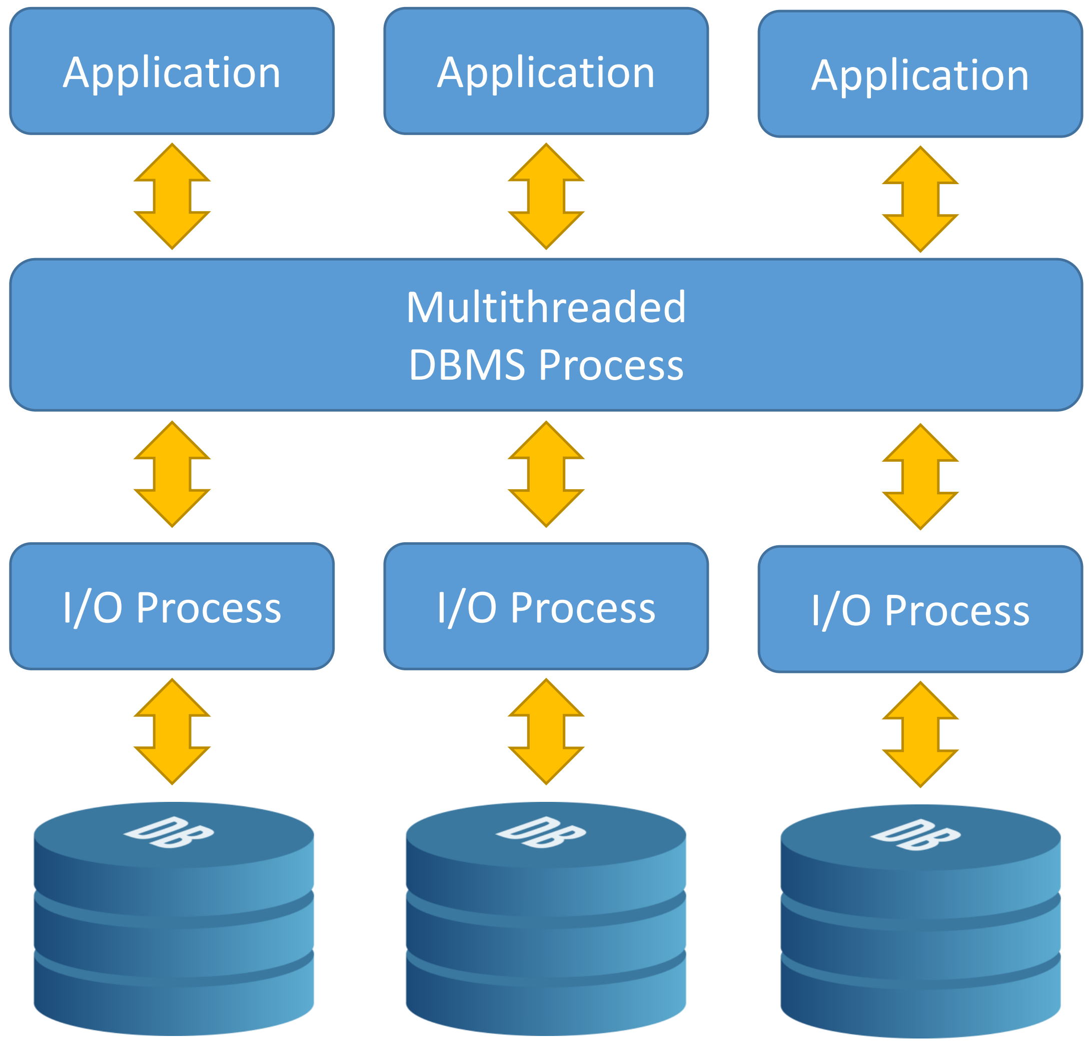
- Using a process to serve a user connection.
- The working process receives database commands such as SQL commands from the application.
- Popular among small scale open source DBMSs.
- Simple implementation and portability.
- High context switching overhead.
- The number of connection depends on the maximum number of processes.



DBMS **Multi-threaded** Architecture

- Instead of using processes, this architecture employs threads within one or more processes.
- Based on the current DBMS implementation on multiprocessor machines, one or more multithreaded server processes provide working DBMS threads for user applications.
- The number of processes depends both on the number of CPU available and the database workload on the machine.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



DBMS **Multi-threaded** Architecture,
Asynchronous I/O

- Utilize the asynchronous input/output capability of the operating systems when available.
- Allow the DBMS to work simultaneously on other tasks when the I/O devices are working on the assigned requests.

Relational Database Queries Processing

pakorn.wa@kmitl.ac.th

- Relational DB requests from users are parsed (syntax checked) and translated into internal statements before being optimized by the query optimizer.
- The SQL language is defined the results – the "what" type.
- The query optimizer is to find the best way to obtain a query result – the "how" part.
- Two common types of query optimizer
  - The rule-based optimizer
  - The cost-based optimizer
- Others are the semantic query optimizer but not common in commercial database.

The Rule-based Optimizer

pakorn.wa@kmitl.ac.th

- An early technology – employ fixed heuristic rules.
- Query execution plans for SQL queries are generated according to pre-defined set of rules.
- The order of the search conditions in the `WHERE` clause is important and determines the query execution plan.
- The order of the tables which are specified in the `FROM` clause also matter.
- Different SQL techniques which give the same result may give slightly different response time.
- To gain better performance, application programmers must know the heuristic rules and write SQL codes to encourage the deployment of better rules.
- The correct choice of the rules depends on the size and the distribution nature of the data.

The Cost-based Optimizer

- A more recent technology – employ database statistics and advanced calculations to find the best way to obtain a query result.
- Common database statistics: size of the database tables, rows, indexes, number of distinct values in a column, etc.
- The way the programmers write SQL statements has minimum or no effect to the physical access path chosen by the optimizer.
- The input query is translated into possible query plans, which are described internally by the relational algebra.
  - Each plan comprises several relational algebra operations.
  - Each basic operation has several alternative algorithms to implement.
- The final implementation choice of each operators depends on the availability of physical access mechanisms, physical organization of the data, data size, and data distribution – all of which must be obtained from a pre-collected database statistics.

Example 1

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

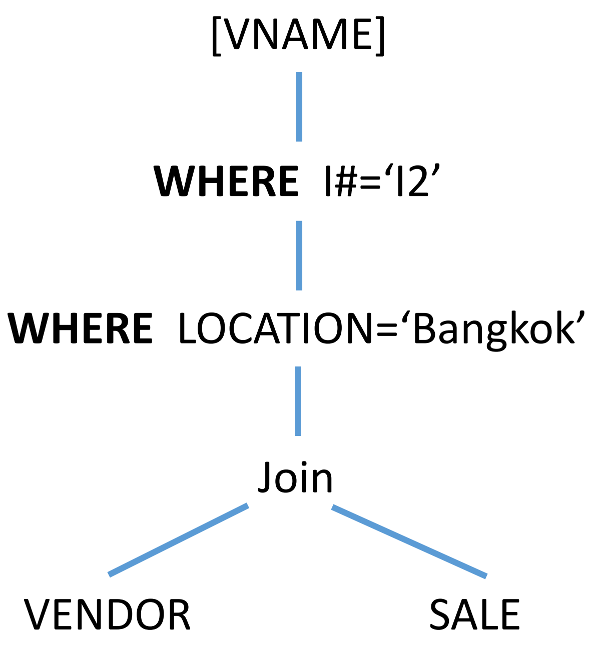
ITEM

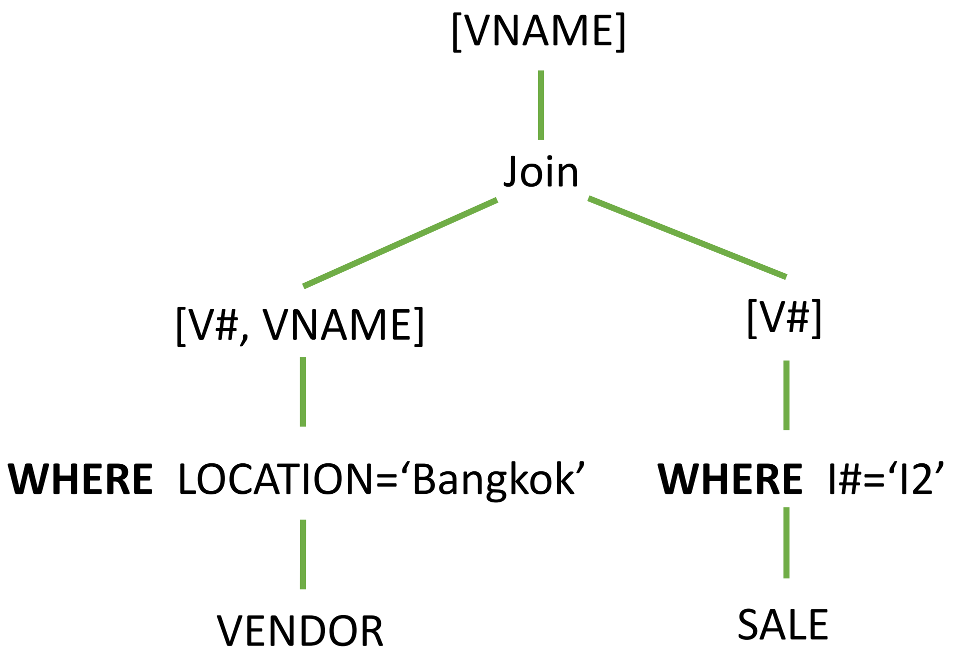
I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

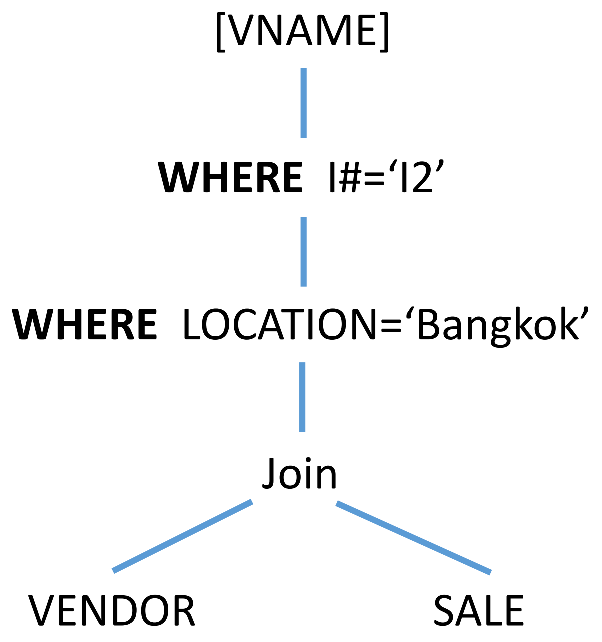
V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

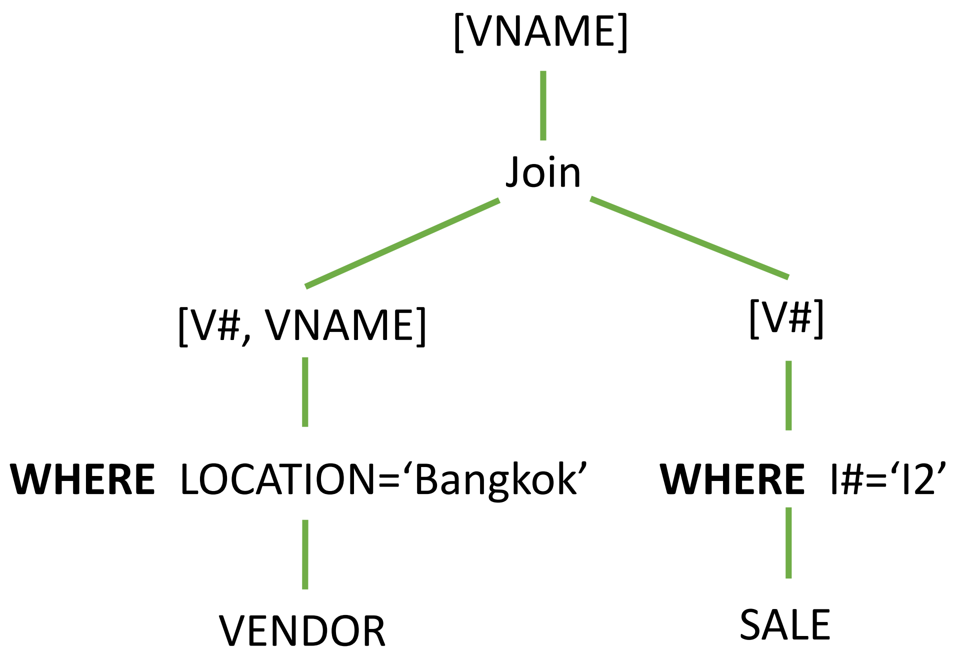
```
SELECT VNAME
FROM   VENDOR,SALE
WHERE  VENDOR.V#=SALE.V#   AND
       LOCATION='Bangkok'  AND
       I#='I2';
```





Which one is better?

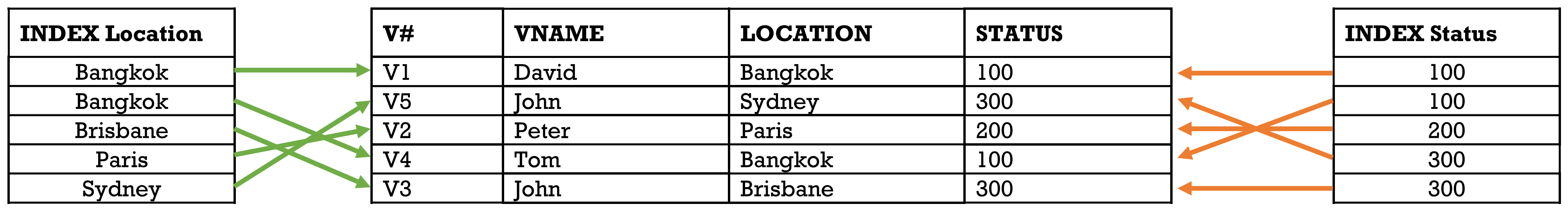




… in term of "response time"!

```
SELECT *
FROM   VENDOR
WHERE  LOCATION='Bangkok' AND
       STATUS=100
```

VENDOR FILE



INDEX Location
Bangkok
Bangkok
Brisbane
Paris
Sydney

V#	**VNAME**	**LOCATION**	**STATUS**
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

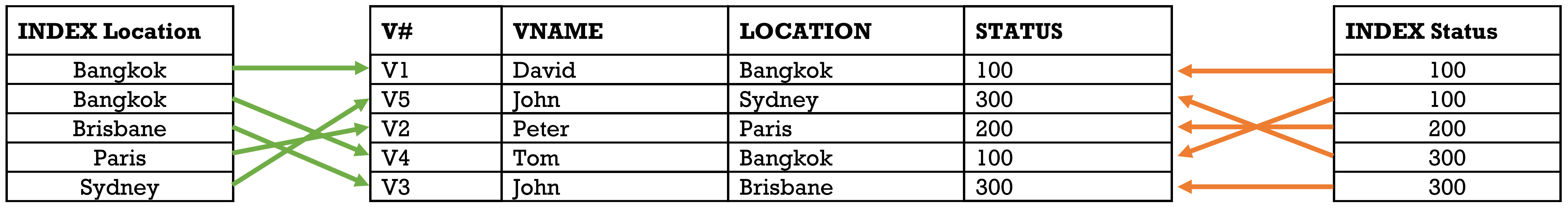
INDEX Status
100
100
200
300
300

- The physical table may not be sorted by any order.

- An optional access mechanism to reach the required row.
- The index entries are sorted.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR FILE



INDEX Location
Bangkok
Bangkok
Brisbane
Paris
Sydney

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

INDEX Status
100
100
200
300
300

```
SELECT *
FROM   VENDOR
WHERE  LOCATION='Bangkok' AND
       STATUS=100
```

- Assume that there are
  - 10,000,000 rows in the VENDOR
  - 10 rows whose LOCATION is Bangkok
  - 50,000 rows whose STATUS = 100.
- Obviously, retrieving those 10 Bangkok rows is better than the 50,000 rows whose STATUS is 100.

- In reality how does the cost based optimizer know these figures?
  - How do they know the number of rows returned for each condition?
  - Do they collect the information during the query optimization process?

- The collection of the essential information for query optimization is technically called database statistics collection or analyzing tables.
- This is the duty of database system administrator (DBA) to perform this task periodically to ensure that the database statistics are up to date and ready to be used by the cost-based optimizer.

- In practice, database statistic collection of large databases could be time consuming and only statistics of frequently used or important tables may be collected.

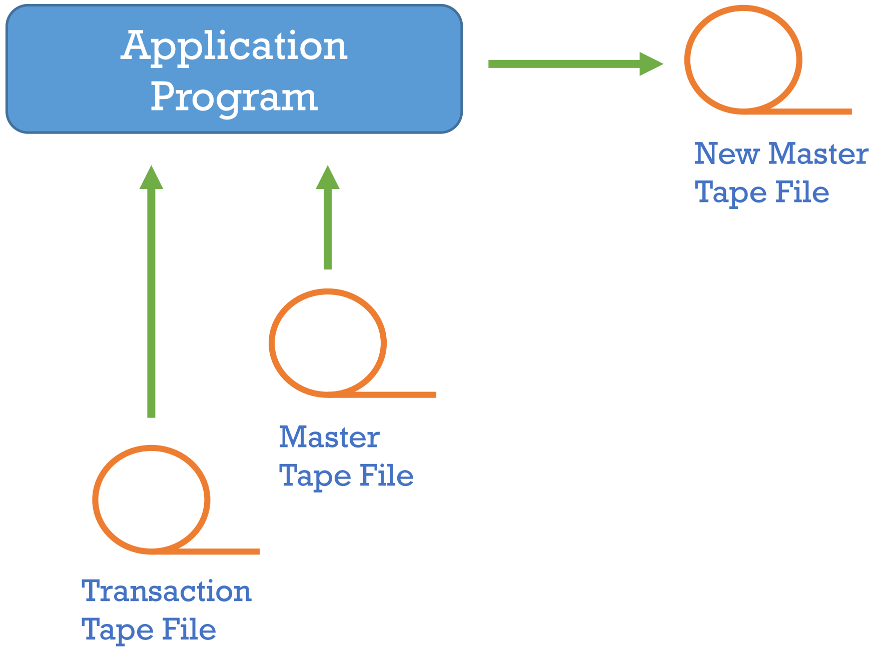
Modern Transaction Processing

pakorn.wa@kmitl.ac.th

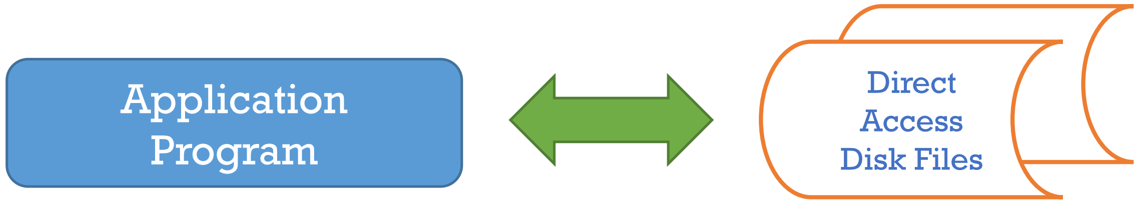
- Several different meaning based on available technologies.
  - 1960s – 1970s : A transaction is a file that collects changes.

IBM 729V
Excerpted from Wikipedia.ORG

10.5-inch diameter reel
Excerpted from Wikipedia.ORG



- 1980s – 1990s:
  A transaction is a single file operation such as an insert, delete, and modify of a record.



- 1990s - current:
  A transaction is a **logical unit of work (LUW)** in which integrity constrains are allowed to be violated.
  - LUW is a group of one or more logical database operations.
  - The integrity constraints refer to the rules which enforce correctness on the databases.
    - Rules enforce the correctness of the database content.

Integrity Constraints

- Modern DBMSs have the facility to create and maintain integrity constraints.
  - The Universal Modeling Language (UML) provides the Constraint Language (CL) for integrity constraints declaration.
- A logical database model has integrity constraints on data model's data structures, called structural constraints.
- The relational database model has two integrity constraints namely, the entity integrity and the referential integrity.

- Transaction processing
    - *E.g.*, A company has an integrity rule (R1) states that a company car must have officers in charge. Assume that one table for cars, and one table for officers' assignment to cars.

Begin transaction

1. Insert the new car's information into the car table

    ----- *R1* is temporarily violated

2. Insert the officer assignment to the new car

End transaction

- The beginning and the end of the transaction are collectively called sync points, or synchronization points.
  - In theory, integrity rules are allowed to be violated inside of the transaction.
  - In practice, rules may need to be disabled and enabled manually by using extra commands inside of the transaction.
- The SQL language has three built-in sync points:
  - `COMMIT` :
    All logical database operations since the previous sync point are confirmed.
  - `ROLLBACK`:
    All logical database operations since the previous sync points are cancelled.
  - `SET AUTOCOMMIT OFF`:
    After each successful database operations, the commit operation is confirmed only when the `COMMIT` statement is issued and confirmed explicitly.
    `SET AUTOCOMMIT OFF` is equivalent to `Begin` transaction.

- COMMIT
  - In commercial DBMSs sync point commands may vary.
    - Begin, End, Begin Work, End Work, and Start transaction.
  - Note that
    - COMMIT is a logical level statement and should not be confused with the physical writing of changes in the database buffer in the main memory to the secondary storage such as hard disks.
    - It is possible that several transactions are committed without any physical database modifications.
    - A specialized technique called the deferred database modification do not allow the transfer of database buffer content to secondary storage before commit.

pakorn.wa@kmitl.ac.th

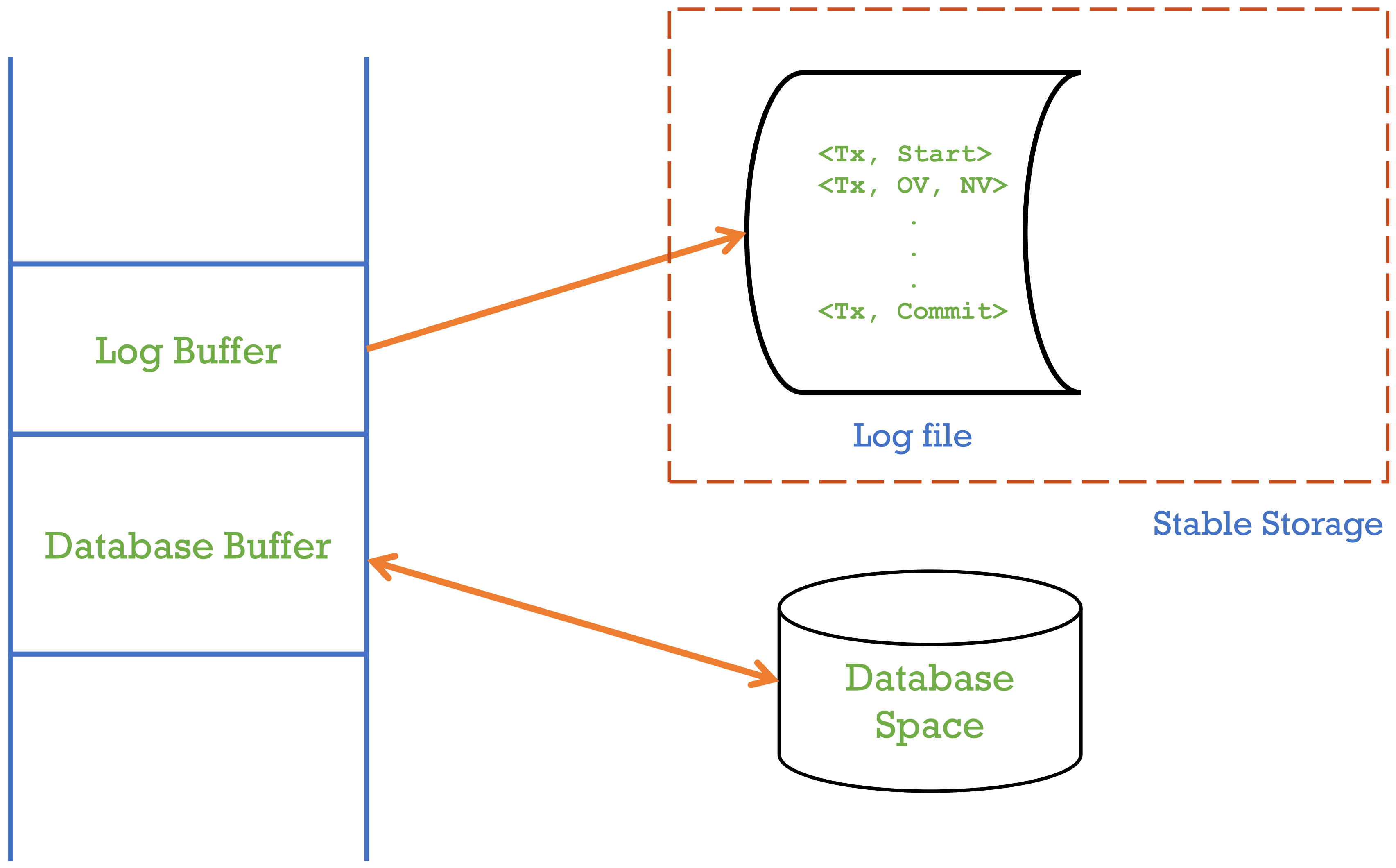
- Database Modification
  - Data manipulations and retrieval are performed in the database buffer which resides in the main memory of the database server.
  - Data are read from secondary storage into database buffer for processing.
  - After the data are no longer required by any transactions, they are transferred back to the database on the secondary storage.

A Log-based Transaction Recovery Technique

pakorn.wa@kmitl.ac.th

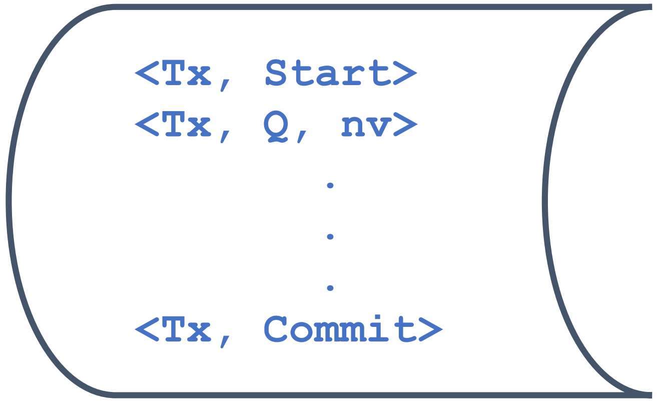
- What if there is a system crash or a transaction failure?
  - Since the actual data manipulations are performed in the database buffer whose content is lost during a system crash, how to recover the lost content so that the database is modified and consistent with the already committed transaction?
  - Transaction durability property!
- Two main problems regarding system crashes and transaction recovery.
  - Problem 1:
    Committed transaction but database modifications are not yet completed, or not even started.
  - Problem 2:
    Uncommitted transaction but some database modifications have already taken place.

- The log-based recovery technique
  - A log file keeps log records which keep tracks for transaction execution steps.
    - Transaction starts
    - Data before and after modified by instructions in the transaction
    - Transaction commit
  - Log records are buffered in the log buffer in the log file.
  - The log file should be implemented on a stable storage – a disk array or a mirror hardware.
  - A transaction physically commits when its commit record is written successfully to the log file.



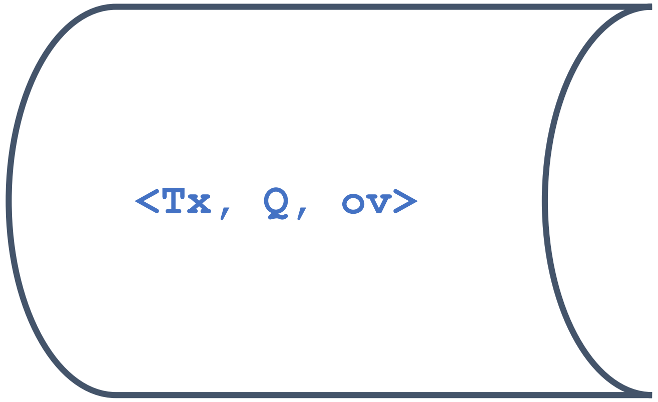
A transaction physically commits when its commit record is written successfully to the log file.

pakorn.wa@kmitl.ac.th



```
<Tx, Start>
<Tx, Q, nv>
        .
        .
        .
<Tx, Commit>
```

After Image Journal (AIJ)
(Redo log file)



```
<Tx, Q, ov>
```

Before Image Journal (BIJ)

- Upon restarting the DBMS after a system crash
  - The recovery manager scans log file backward from the end to determine which transaction committed and which ones failed by checking the existence of the commit record of each transaction.
  - After committed and failed transactions are classified, old values of failed transactions are used to undo new uncommitted value out of the database on the secondary storage and replace them with the old values before performing the transactions. (Solve Problem#2)
  - The next step is to apply new values of committed transaction to redo them to the database on the secondary storage. (Solve Problem#1)

ACID Properties of Transactions

pakorn.wa@kmitl.ac.th

- Atomicity:
  Operations in the transaction must either be executed successfully or none are.
- Consistency:
  Execution of a transaction in isolation ensures the consistency of the database according to the integrity constraints.
- Isolation:
  Immediate results obtained during processing of an uncommitted transaction must not be seen by other transactions.
- Durability:
  After a transaction commits successfully, the modifications it has made to the database persist, even if there are system failures.

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Q & A

The Relational Database Model

Relations in Relational Database

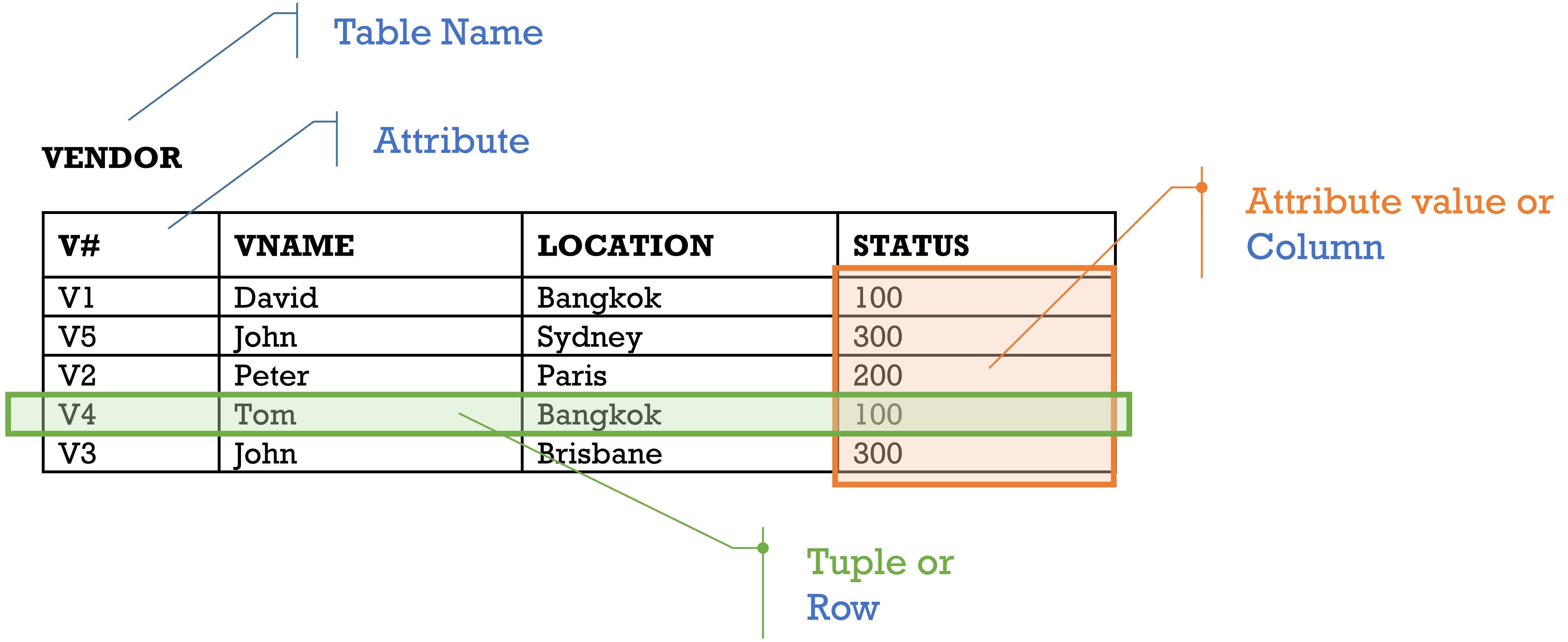
- The term **relation** is a subset of the Cartesian product of domains.
- A domain is a set of permitted values of the same type.
- *E.g.*,
  - student_name = { Somchai, Jane, Ovi, Chau, Bryan }
  - student_city = { Bangkok, Brisbane, Hanoi, Jakarta, Sydney, Denver }
  - student_dept = { CE, EE, ME, IT }
  - Then, *r* = { (Jane, Bangkok, CE) , (Ovi, Jakarta, IT) , (Chua, Hanoi, CE) , (Bryan, Denver, EE) }
- Note that the order of the values in each tuple is significant; otherwise, it is impossible to relate each value to its corresponding domain.
- The concept of **attribute** is introduced to avoid the problem of value position order.

Relational Database Tables

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- Associating each tuple with its domain and distinct role by applying an **attribute** concept.
- The n distinct attributes of a relation of degree n distinguish the n roles of the domains on which the relation is defined.
- Base on the attribute concept, the order of the attributes of a relation is not significant.

pakorn.wa@kmitl.ac.th



VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

- Not all tables are relations!
- The tabular representations of a relation must have the following properties:
  1. The values of each column must belong to only one domain of interest.
  2. All table entries are atomic values.
  3. No two rows are the same.
  4. Rows order is insignificant.
  5. The order of the columns is insignificant, since each column is identified by a column name which represents an attribute.

Relational Schemas and Instances

- The **intensional** database
  - Describe the classes or types of values and classes or types of associations among them.
  - Corresponds to the structured description – relational schemas – a named set of distinct attributes.
  - A relational schema R with attributes $A_1, A_2, \dots, A_n$ is written $R(A_1, A_2, \dots, A_n)$.
  - Commonly use capital letters for schema names and attributes.

- The **extensional** database
  - Describe the actual values or instances and the associations among them.
  - Corresponds to the relations – relational instances or simply "a relation."
  - A relational instance defined on a schema $R(A_1, A_2, \dots, A_n)$ is a finite set r of tuples over $A_1, A_2, \dots, A_n$.
  - Note that there can be several relational instances defined on the same relational schema.
  - Commonly use lower-case letters for relational instances.

Keys in Relational Databases

pakorn.wa@kmitl.ac.th

- A key comprises an attribute or set of attributes whose value uniquely identify a tuple.
- These attributes are normally obtained from the application environment.

- A **superkey** is a set of attribute whose value uniquely identifies a tuple.
  - No two tuples have the same superkey value.
  - A relation may have several keys.
  - Some of which are not concise and hard to refer to. Therefore, the smallest ones are preferred, called **candidate keys**.
  - The relational schema is always a superkey of a relation since by definition no two tuples of a relation are the same.

- The **candidate keys** is defined as a superkey which has no other superkeys as a subset (or has no superkeys as its proper subset).
  - The keys have the uniqueness and minimality properties.
  - The minimum subset of the superkey.
  - They should be easy to refer to.

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

- Superkeys
  - (V#, VNAME, LOCATION, STATUS)
  - (V#, VNAME, LOCATION)
  - (V#, VNAME)
  - (V#)

The relational schema

- Which combination of attribute values are always unique?
  - From interview, from information gathering during the system analysis phase.
  - Database tables are created using known characteristics and relationships between attributes.

  - The uniqueness property of an attribute or group of attributes must be known prior to relational database table creation.

  - The database designers must specify the keys of real world objects.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

- Superkeys
  - (V#, VNAME, LOCATION, STATUS)
  - (V#, VNAME, LOCATION)
  - (V#, VNAME)
  - (V#)

key

The smallest superkey;
It has NOT other superkeys as its subsets.

Note: a key which has one attribute is called a "single key," while a key which comprises of more than one attributes is called a "composite key" or a "combined key."

SALE

V#	**I#**	**AMOUNT**
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

- The **SALE** table has (V#, I#) as a combined candidate key.
- How do we know that the combination of these two attributes is unique?
  - It has been given by the application expert or users.

pakorn.wa@kmitl.ac.th

- The **primary key**
  - The most important key in the relational database.
  - The primary identifier of a relation.
  - Defined as a **selected candidate key**.

- Candidate keys which are not selected as the primary key are called alternate keys.

- The concept of keys in relational database is about the logical identification of tuples and should not be confused with the physical access mechanism such as indexes.

- **Surrogate keys**
  - Artificial created attributes by database designers, especially when the original identifier comprises too many attributes or difficult to refer to.
  - Note that many Computer-Aided Software Engineering (CASE) tools automatically create surrogate keys during the design phase. Avoid using them if the designed primary key(s) comprises workable size of attributes.

Relational Structural Constraints

- The relational database model provides two structural constraints
  - The **Entity Integrity**:
    Primary key values must not be null.
  - The **Referential Integrity**:
    Foreign key values must match primary key values or be null.

The Entity Integrity

pakorn.wa@kmitl.ac.th

- Primary key values must not be null.
- What is a NULL value?
  - Two widely accept interpretation are
    - Attribute applicable, but value unknown – the "don't know"
    - Attribute not applicable – the "don't have"
- Therefore, the entity integrity ensures that each and every row has a unique identifier – the primary key – which is not NULL.
- Without the primary key value, a row cannot be identified.

The Referential Integrity

pakorn.wa@kmitl.ac.th

- **Foreign key** values must match primary key values or be null.
- The referential integrity rule enforces correctness of data reference between relational database tables.
- The foreign key is defined as :
  nonkey attribute(s) in a relation which is the primary key of other relation or the same relation.
- A nonkey attribute is defined as :
  an attribute which is not a candidate key.

EMP

⟷

<u>E#</u> (pk)	ENAME	ADDRESS	SALARY	D# (fk)
E1	John	London	50,000	D1
E2	Peter	Paris	45,000	D1
E3	Tom	Brisbane	50,000	D2

DEPT

⟷

<u>D#</u> (pk)	DNAME	LOCATION
D1	Finance	Bld1
D2	Engineering	Bld1
D3	Sales	Bld2
D4	Services	Bld2

- The entity and referential integrity constraints are constraints on relational database structures and independent from the applications.
- In practice, the rows of **DEPT** should be loaded into the database first. The rows of **EMP** are then followed.
- These integrity constraints are mainly validated at the data loading or insertion time. A row with a reference to non-existence primary key value is rejected.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

EMP

E# **(pk)**	**ENAME**	**ADDRESS**	**SALARY**	**D# (fk)**
E1	John	London	50,000	D1
E2	Peter	Paris	45,000	D1
E3	Tom	Brisbane	50,000	D2

?

DEPT

D20

D# **(pk)**	**DNAME**	**LOCATION**
~~D1~~	Finance	Bld1
D2	Engineering	Bld1
D3	Sales	Bld2
D4	Services	Bld2

1. Disallow primary key value update if there is at least a referencing foreign key value.
2. Allow the update of the primary key value and subsequently update all the corresponding foreign key values to the new primary key value.
3. Allow the update, but set the corresponding foreign key values to a default value including the null value.

KMITL พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

EMP

<u>E#</u> (pk)	**ENAME**	**ADDRESS**	**SALARY**	**D# (fk)**
E1	John	London	50,000	D1
E2	Peter	Paris	45,000	D1
E3	Tom	Brisbane	50,000	D2

DEPT

<u>D#</u> (pk)	**DNAME**	**LOCATION**
D1	Finance	Bld1
D2	Engineering	Bld1
D3	Sales	Bld2
D4	Services	Bld2

```
CREATE TABLE DEPT ( D#          CHAR(4),
                    DNAME       CHAR(25),
                    LOCATION    CHAR(25))
PRIMARY KEY IS D#;
```

```
CREATE TABLE EMP  ( E#          CHAR(4),
                    ENAME       CHAR(50),
                    ADDRESS     CHAR(50),
                    SALARY      NUMBER,
                    D#          CHAR(4))
PRIMARY KEY IS E#,
FOREIGN KEY IS D# REFERENCES D# OF DEPT,
DEFAULT IS NULL,
UPDATE RESTRICTED DELETE RESTRICTED;
```

pakorn.wa@kmitl.ac.th

$$\textbf{UPDATE} \begin{Bmatrix} \textbf{RESTRICTED} \\ \textbf{CASCADES} \\ \textbf{DEFAULT} \end{Bmatrix}$$

$$\textbf{DELETE} \begin{Bmatrix} \textbf{RESTRICTED} \\ \textbf{CASCADES} \\ \textbf{DEFAULT} \end{Bmatrix}$$

pakorn.wa@kmitl.ac.th

Depend on the DBMS the constraints declaration syntax may vary.

The Relational Algebra

pakorn.wa@kmitl.ac.th

- The relational algebra is a procedural language whose sole data structure is relation.
- The operands must be relations and the results are relations.
- The relational algebra is still widely used internally by the DBMS.
- All relational DBMSs translate statements in the relational calculus-based languages into equivalent relational algebra statements and further process them at the relational algebra level.

pakorn.wa@kmitl.ac.th

- The relational algebra originally comprises eight operators (later extended).
  - Relational operators: select, project, join, and divide.
  - Traditional set operations: union, intersection, difference, and Cartesian product.

- Relational operators
  1. **Select σ**
     Obtain tuples which satisfy search conditions from a specified relation.
  2. **Project π**
     Obtain attribute values of specified attributes from a specified relation.
  3. **Join $\bowtie$**
     Create a relation from two relations.
     Values of common attributes are matched and only one of the two common attributes appears at the result.
     This is also called the **natural join**.
  4. **Divide $\div$**
     From the two operand relations, one binary and one unary relation which comprises all values of one attribute from the binary relation which is not the attribute of the unary relation, that match all values in the unary relation.

- Traditional set operations:
  1. **Union ∪**
     Create a relation from two relations which comprises all tuples in either or both of the two relations.
  2. **Intersection ∩**
     Create a relation from two relations which comprises all tuples which are in both relations.
  3. **Difference −**
     Create a relation which comprises all tuples in the first relation and not in the second relation.
  4. **Product ×**
     Create a relation from two relations which comprises the concatenation of all possible tuples from the two relations.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

List only vendors who are in Bangkok

σ LOCATION='Bangkok'

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V4	Tom	Bangkok	100

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

List V# and LOCATION of vendors

π V#,LOCATION(VENDOR)

V#	LOCATION
V1	Bangkok
V5	Sydney
V2	Paris
V4	Bangkok
V3	Brisbane

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

List V# and LOCATION of vendors who are in Bangkok

π V#,LOCATION(σ LOCATION='Bangkok')

V#	LOCATION
V1	Bangkok
V4	Bangkok

pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

List V#, VNAME and LOCATION of vendors who supply the item I2.

π V#,VNAME,LOCATION(σ I#='I2' (VENDOR⋈SALE))

V#	VNAME	LOCATION
V1	David	Bangkok
V2	Peter	Paris
V3	John	Brisbane

pakorn.wa@kmitl.ac.th

List V#, VNAME and LOCATION of vendors who supply all items.

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

$$\pi_{V\#,VNAME,LOCATION}(VENDOR \bowtie (\pi_{V\#,I\#}(SALE) \div \pi_{I\#}(ITEM)))$$

V#	VNAME	LOCATION
V1	David	Bangkok

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Q & A

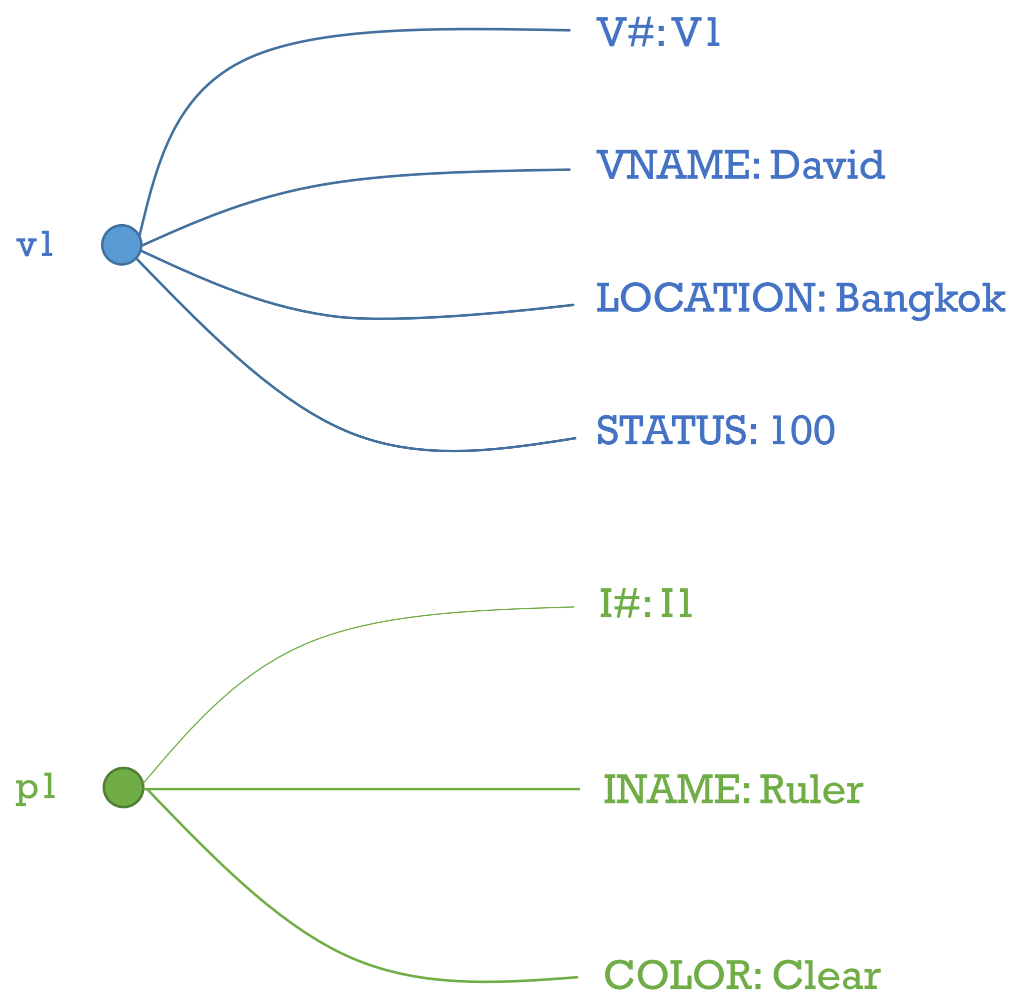
The Entity Relationship Model

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The two basic constructs of the **Entity Relationship Model (ERM)** are the entity and relationship.
  - **Entities** are objects of interest in the application domains.
  - **Relationships** are simply relationships among entities.

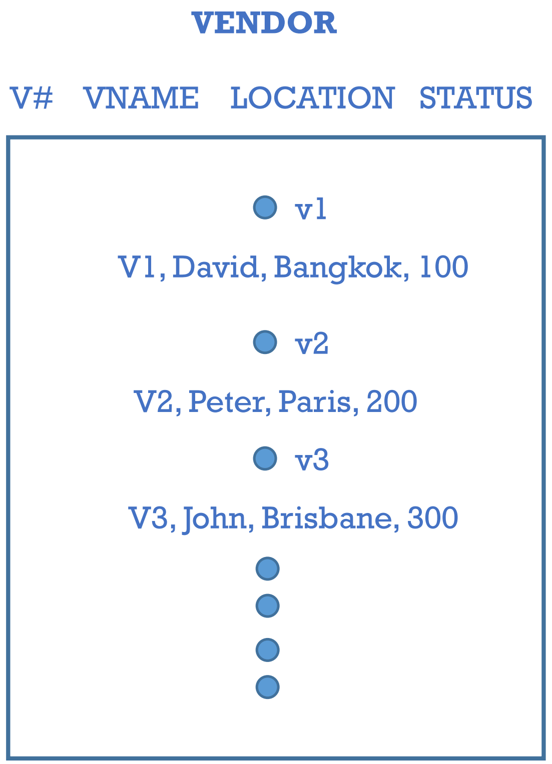
- The concept of entities can be applied to both concrete, or tangible objects, and abstract, or intangible objects.
- The term entity is actually equivalent to the term **object instance** in the **object paradigm**.
- Hence, an entity is formally called **an entity instance**.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



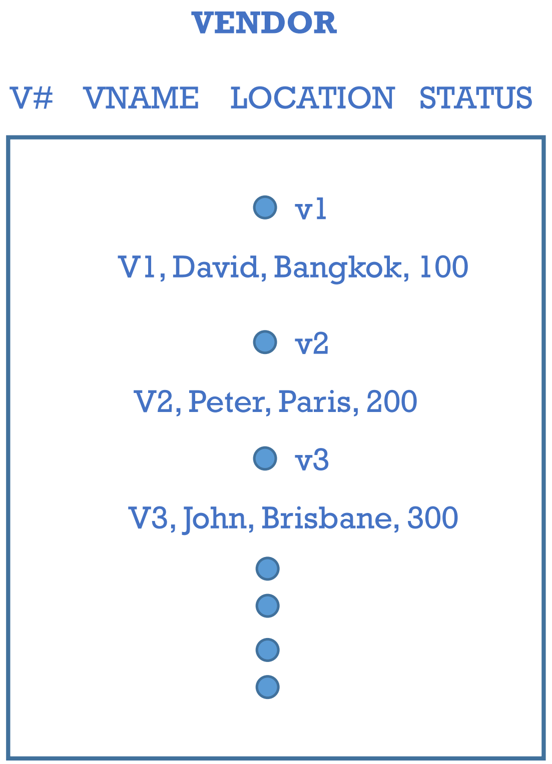
Two entities and their attributes

pakorn.wa@kmitl.ac.th



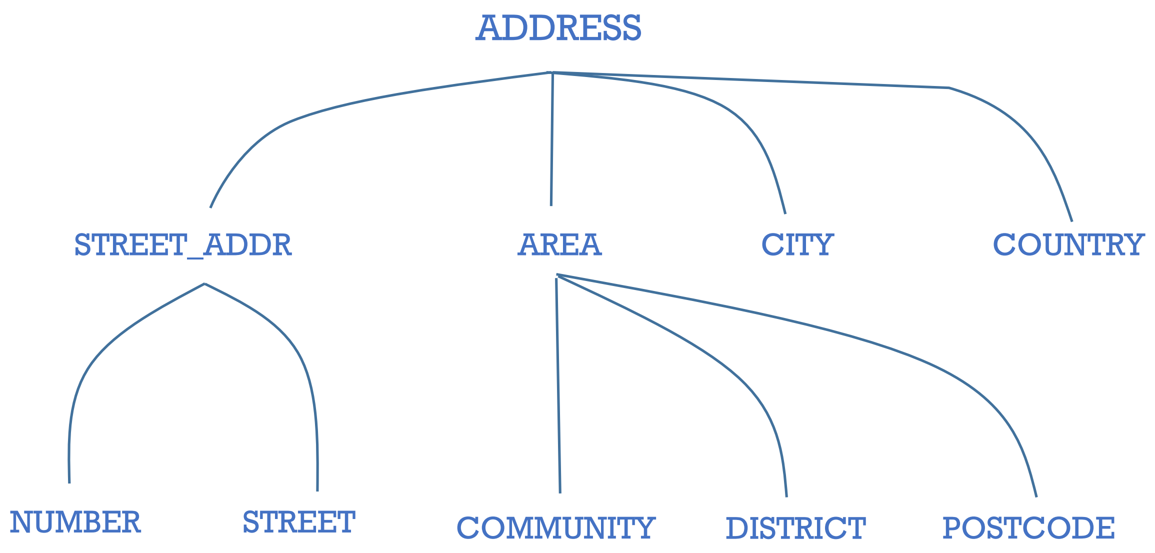
An **entity type** with some **entity instances**

- **Entity instance** which have the same set of attributes can be grouped together to be the same **entity type**.
- **v1, v2, v3, …, v_n** are entities.
- **VENDOR** is an entity type.

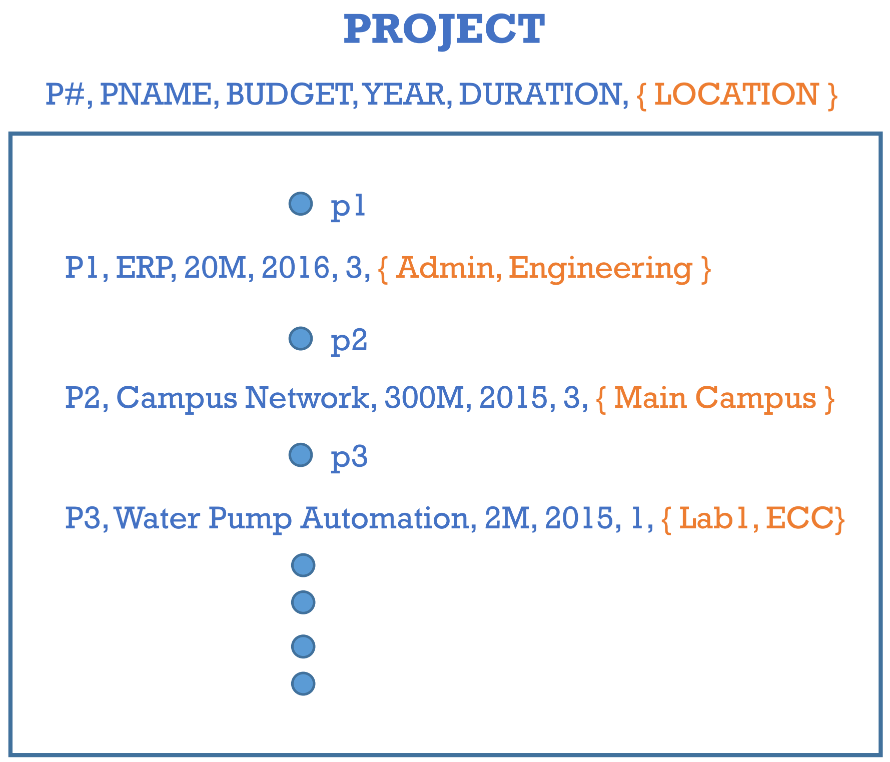


An **entity type** with some **entity instances**

- In the relational database model, the value of an attribute must be atomic.
- In the ER model, both atomic and non-atomic attributes are allowed.



A composite attributes **ADDRESS**

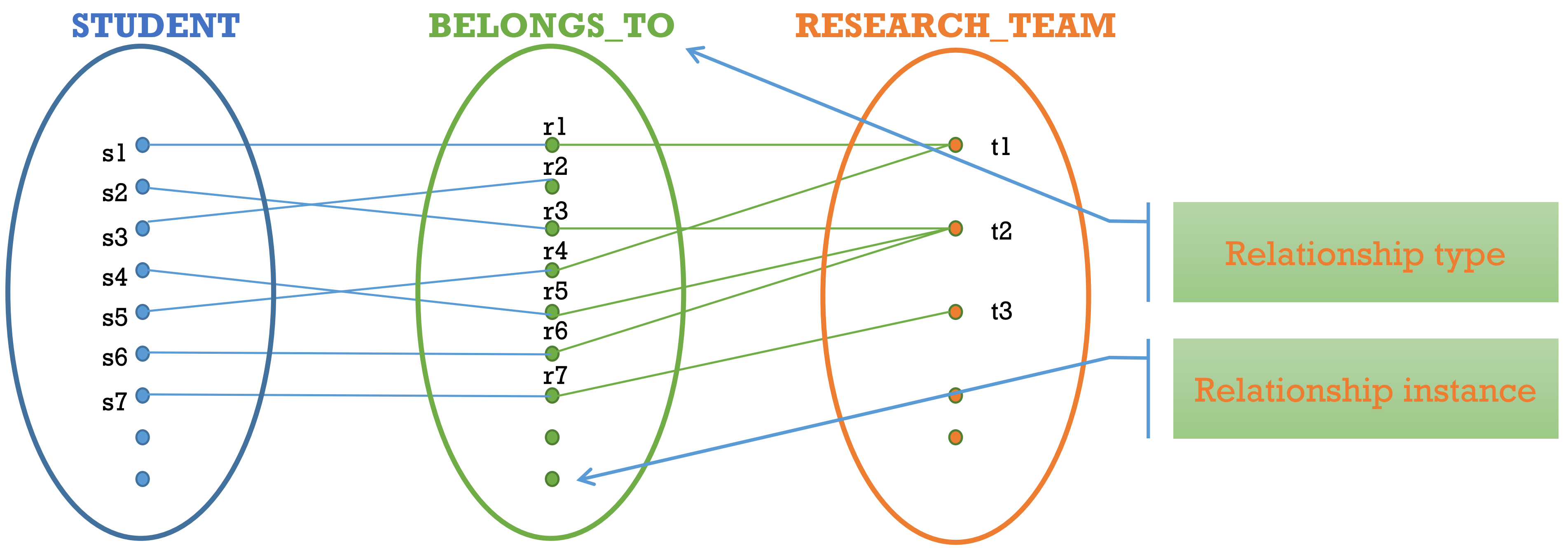

PROJECT
P#, PNAME, BUDGET, YEAR, DURATION, { LOCATION }
p1
P1, ERP, 20M, 2016, 3, { Admin, Engineering }
p2
P2, Campus Network, 300M, 2015, 3, { Main Campus }
p3
P3, Water Pump Automation, 2M, 2015, 1, { Lab1, ECC}

An entity type **PROJECT** with a multivalued attribute

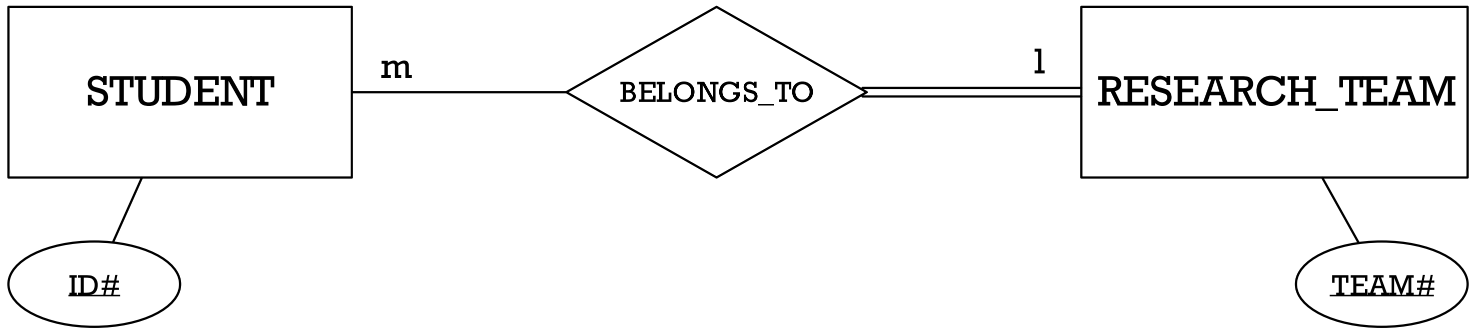
- Composite and multivalued attributes are allowed in the ERM.
- Attributes whose values are unique are called the **key attribute** of the entity type.

- Entity types must represent objects of interest
  – the main characters which are involved in the applications.
- Entity instances will be represented by some database instances such as tuples.
- Entity type are observed to have one or more of the following characteristics.
  - **Nouns** with **unique identifiers**
  - **Verbs** which are used as nouns, with **unique identifiers**
  - **Objects** which have **some details**.
  - In the case that the objects do not have much details,
    they should participate with at least two other entity types.
- Do not confuse entity type with reports.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

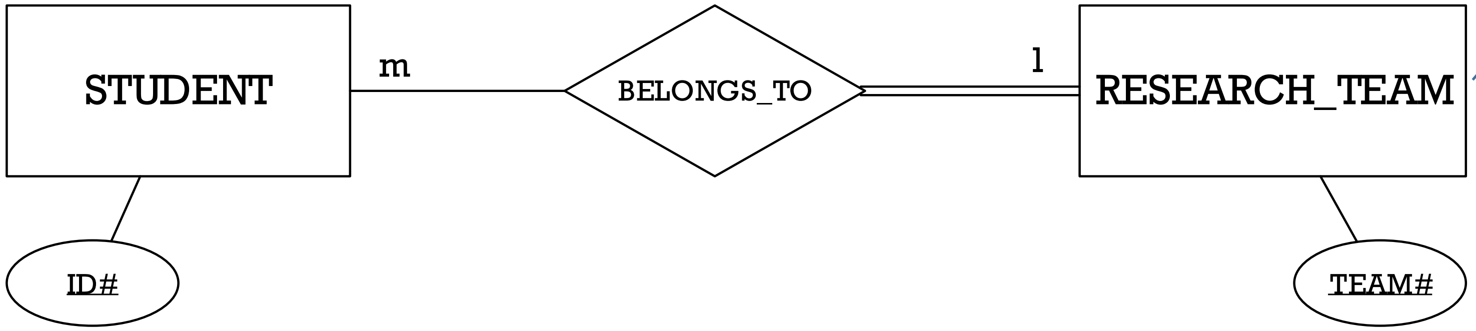


Conceptual diagram of m:1 relationship type BELONGS_TO



- Many instances of STUDENT are related to only one instance of RESEARCH_TEAM.
- The STUDENT is the many side and the RESEARCH_TEAM is the one side.

Entity Relationship diagrams of the m:1 relationship type between STUDENT and BELONGS_TO with the cardinality.

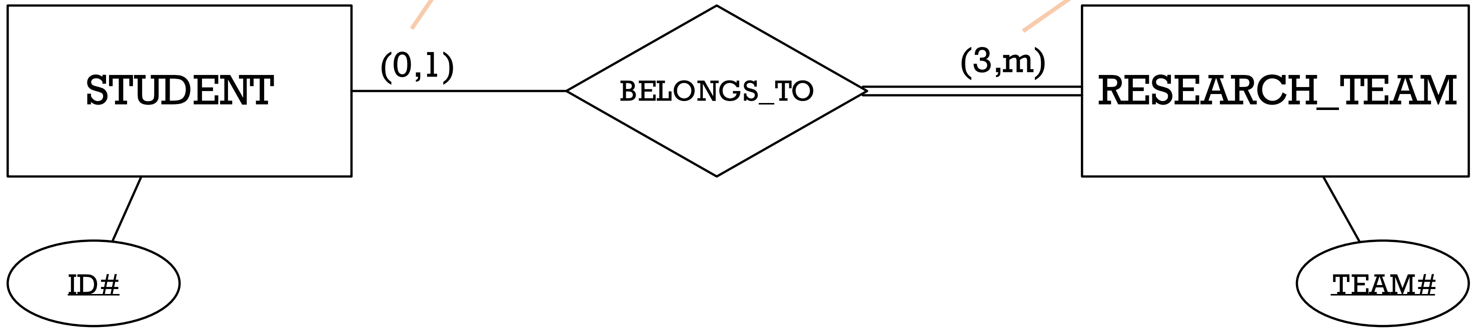


- Many instances of STUDENT are related to only one instance of RESEARCH_TEAM.
- The STUDENT is the many side and the RESEARCH_TEAM is the one side.

Entity Relationship diagrams of the m:1 relationship type between STUDENT and BELONGS_TO with the cardinality.

Participate 0 to at most 1 time

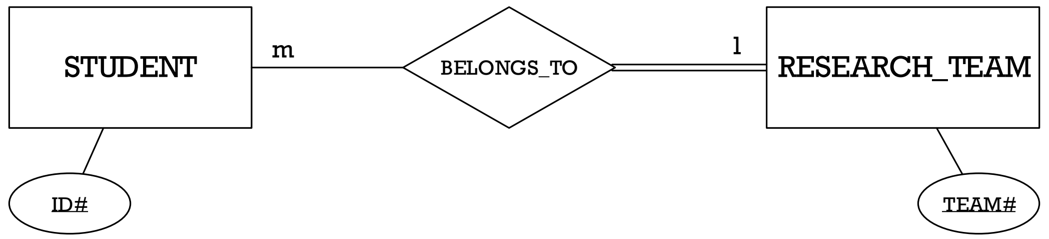
Participate at least 3 times



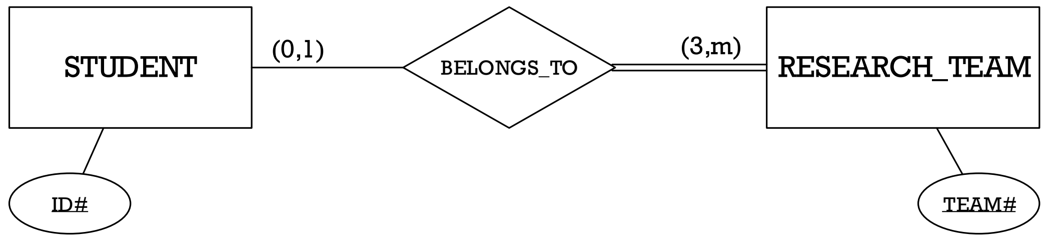
- A student instance participates only **once** with a research team instance.
- A research team many participate with **many** student instances.

Entity Relationship diagrams of the m:1 relationship type between STUDENT and BELONGS_TO with participation notations.

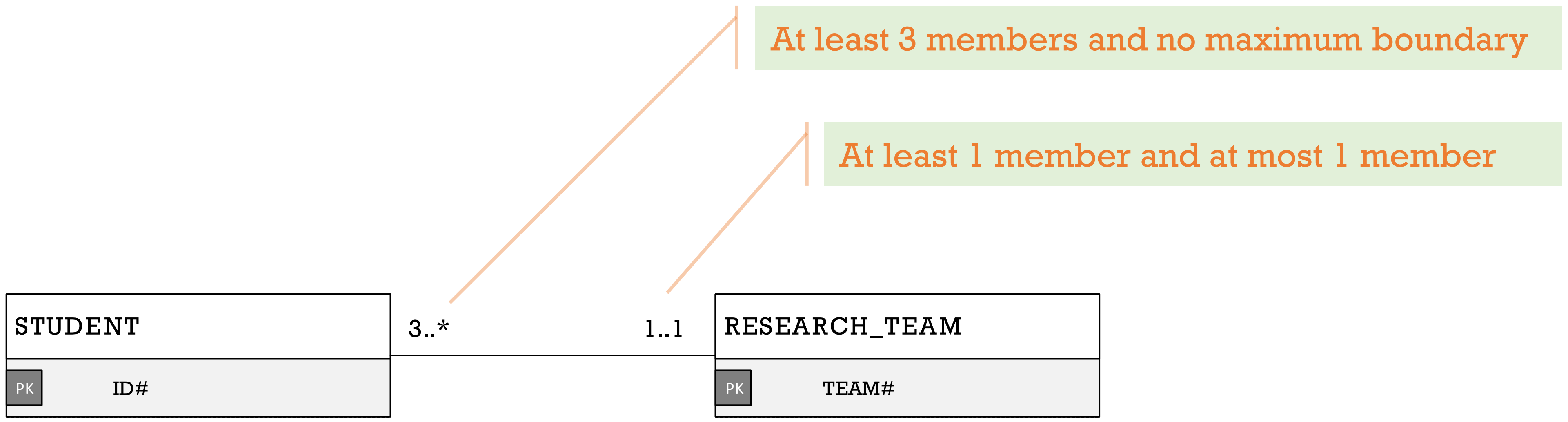
- The (min, max) in the ER diagram is the minimum and maximum participation
  - Not (min, max) cardinality
  - The cardinality of a set is the number of members in the set.
- This (min, max) format may also referred to as the **structural constraint** of the ER model.



Entity Relationship diagrams of the m:1 relationship type between STUDENT and BELONGS_TO with the cardinality notations.



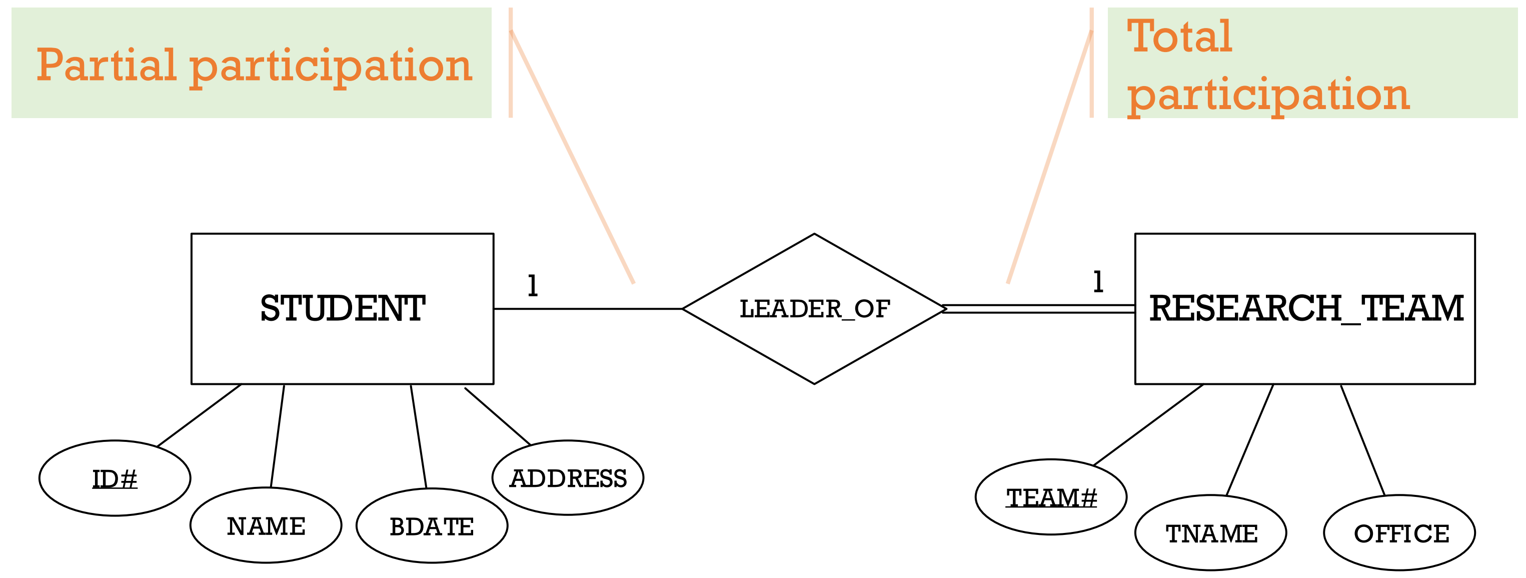
Entity Relationship diagrams of the m:1 relationship type between STUDENT and BELONGS_TO with the participation notations.


At least 3 members and no maximum boundary
At least 1 member and at most 1 member
STUDENT
PK
ID#
3..*
1..1
RESEARCH_TEAM
PK
TEAM#

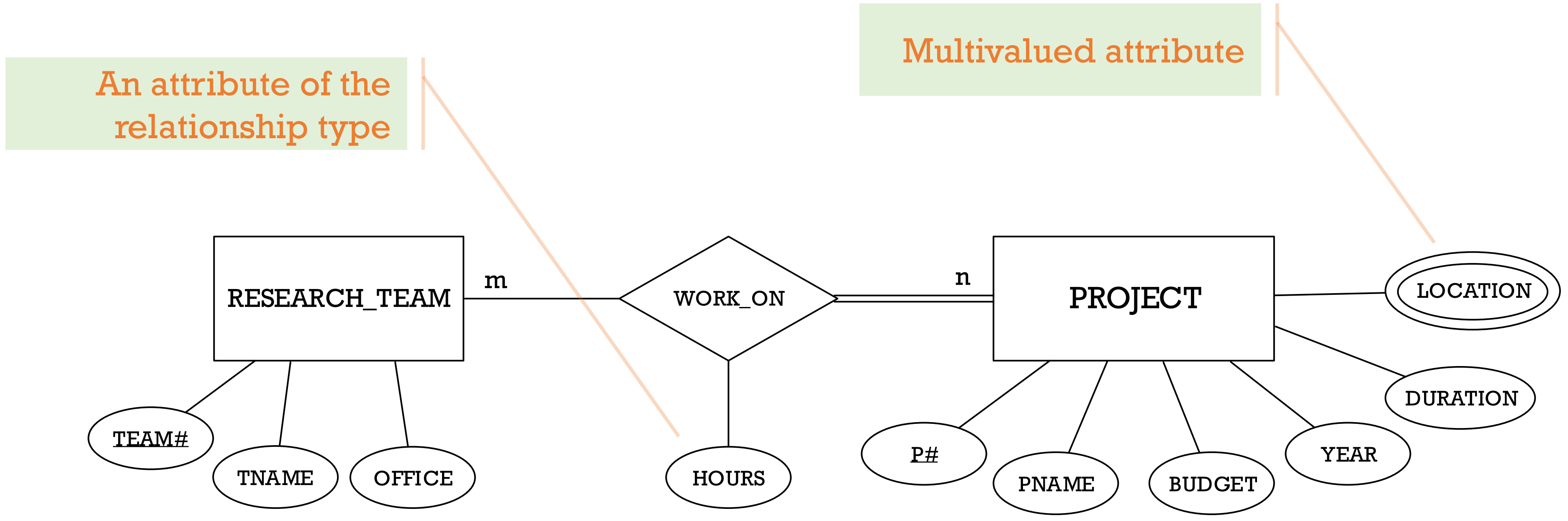
The multiplicity representation between STUDENT and RESEARCH_TEAM

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Before working with a conceptual model, it is a good practice to clarify if the particular model uses the cardinality, multiplicity notation, or participation notation.

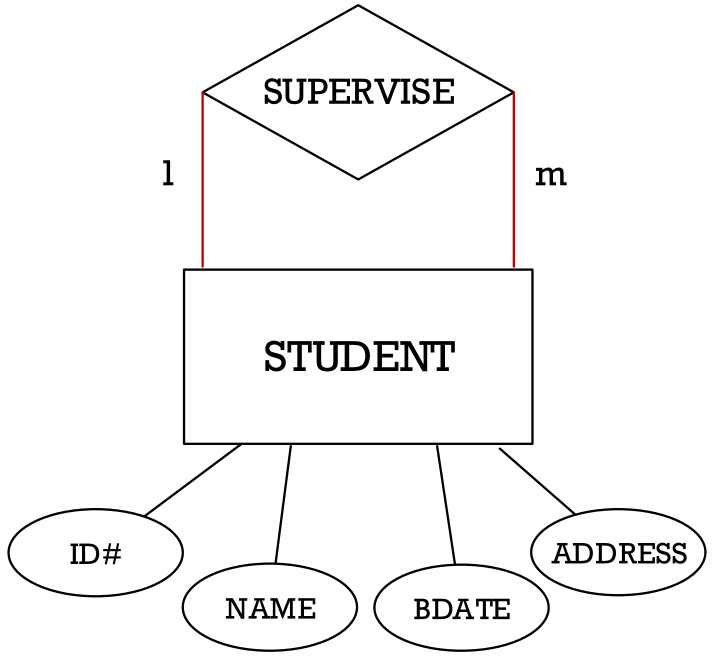


- One student is the leader of one research team.
- A research team has only one student leader.
- An entity type totally participates in a relationship type if all of its instances participate in the relationship type.
  In this case, all instances of the **RESEARCH_TEAM** must have a student leader.
- A partial participation represent that only some instances of the entity type **STUDENT** are leaders of research teams.



- A research team works on one or more project.
- A project also has one or more research team working on.
- **HOURS** is the attribute of the relationship type **WORK_ON** and represents the number of hours that a research team works on a project.
- A double ellipse represents a multivalued attribute.
- A project may have more than one location.

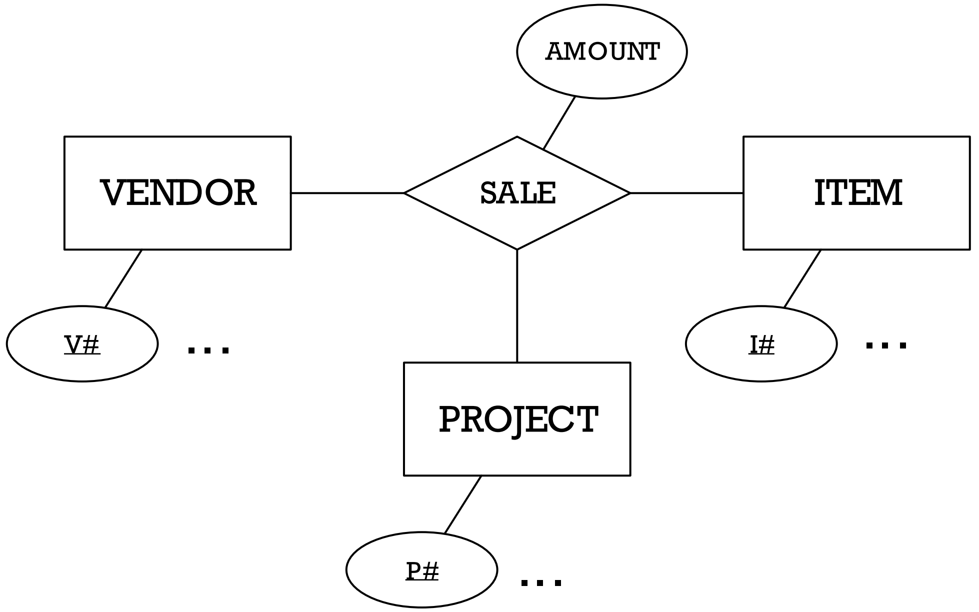
pakorn.wa@kmitl.ac.th



The ER diagram of the recursive relationship type SUPERVISE

- The recursive relationship type **SUPERVISE**, which involves only one entity type.
- Members of the same entity type are related among themselves.
- A student advisee has only one supervisor. A supervisor has one or more supervisees. Both are members of the same entity type **STUDENT**.

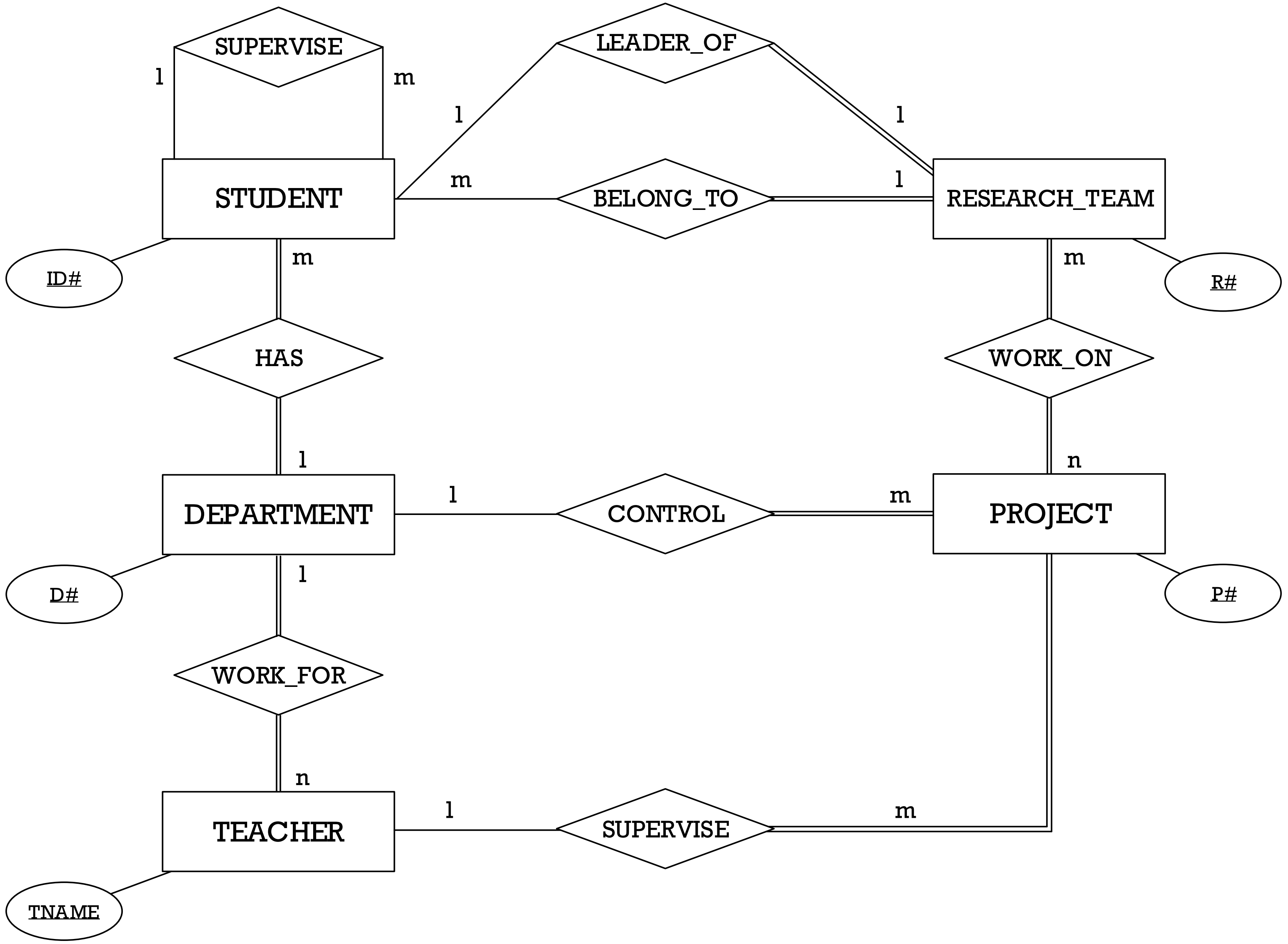
pakorn.wa@kmitl.ac.th



The ER diagram of the ternary relationship type SALE

- The ternary relationship type **SALE**.
- The three entity types, **VENDOR**, **ITEM**, and **PROJECT** are related.
- The relationship type **SALE** has an attribute **AMOUNT**.

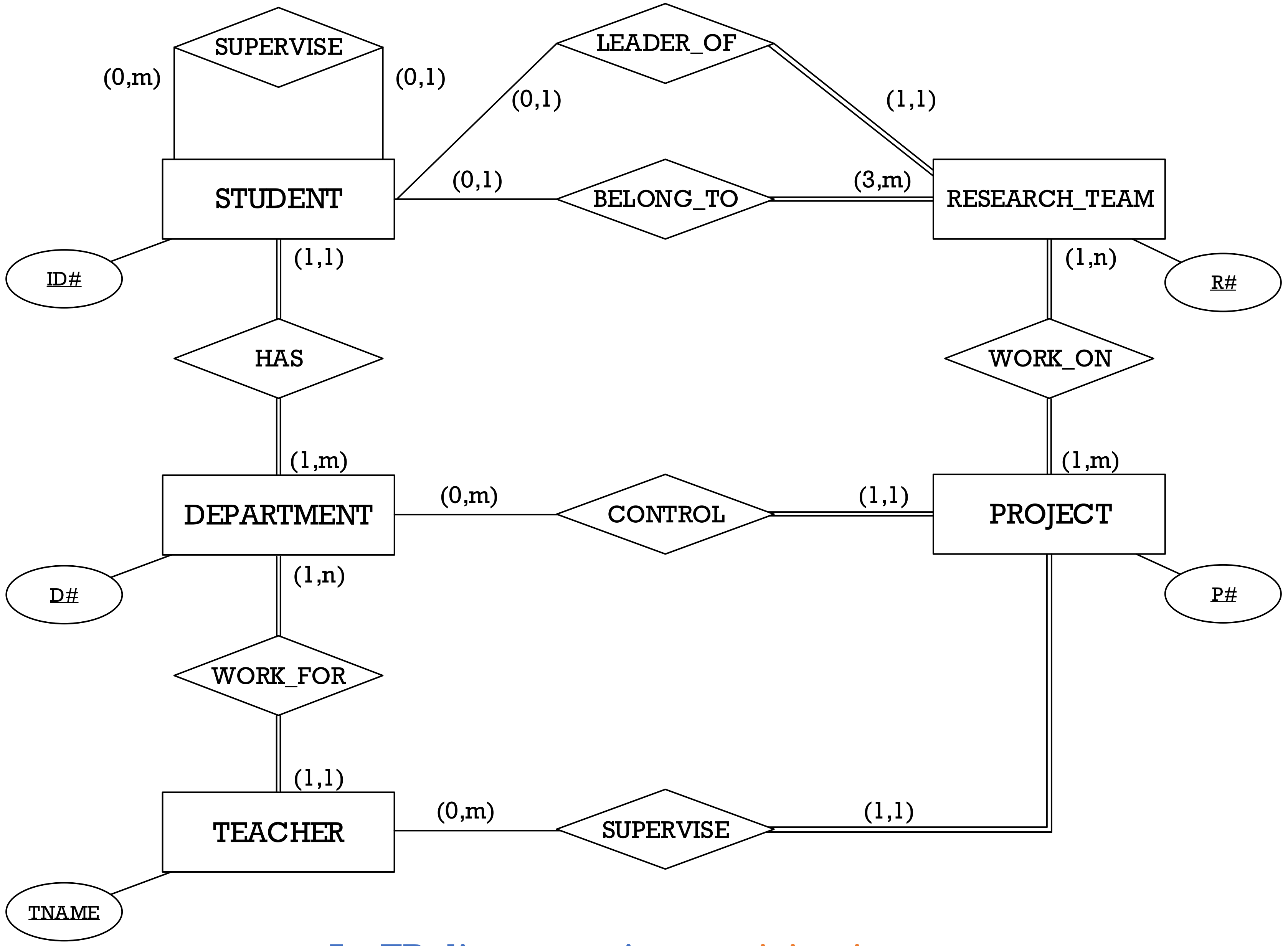
pakorn.wa@kmitl.ac.th



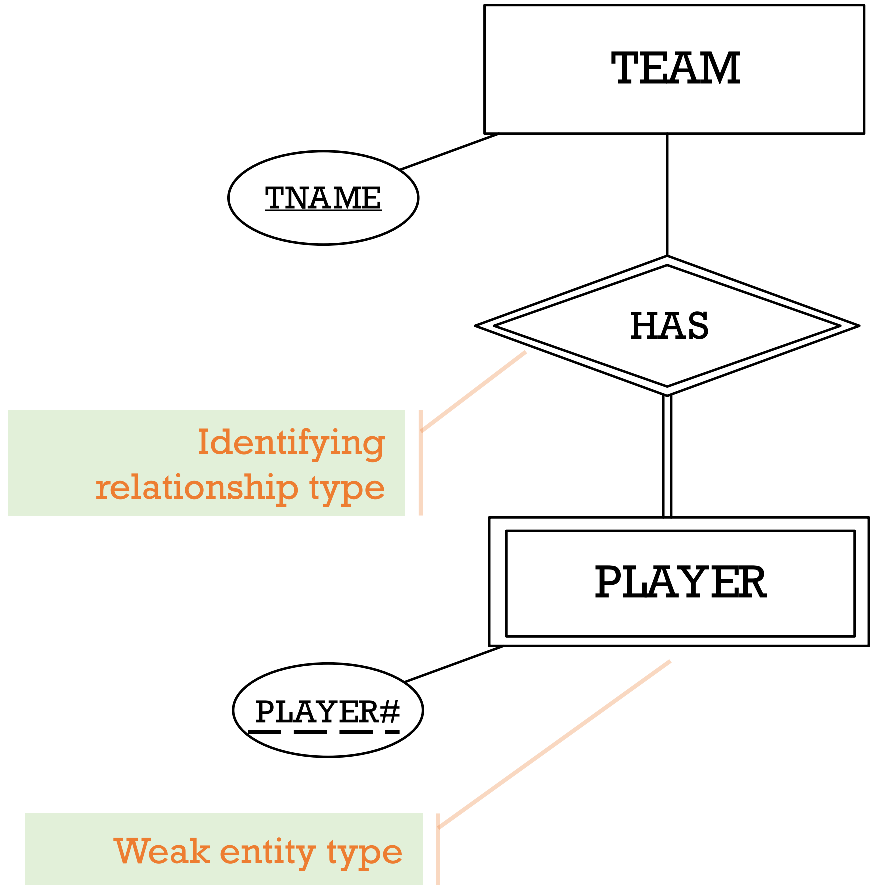
An ER diagram using cardinality

KMITL
พระจอมเกล้าลาดกระบัง

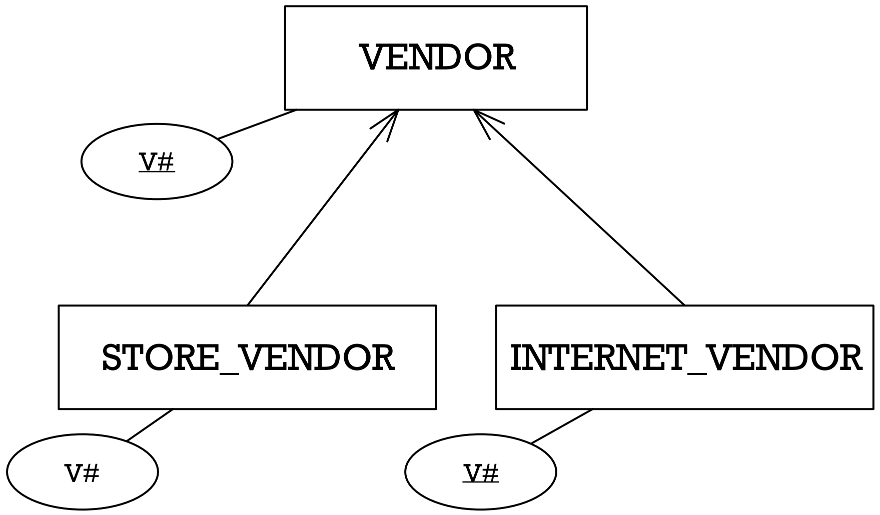
pakorn.wa@kmitl.ac.th



An ER diagram using participation



- A **weak entity type** is defined as an entity type which does not have its own unique identifier but has a **partial key**.
- The weak entity type **PLAYER** represents sport players with an **PLAYER#** attached to the back of the player's shirt and is not unique.
- Different teams may have the same player number; however, each team must have only one player with a given number.
- A player number is not unique but is unique within a team.
- The entity type **TEAM** is referred to as the **parent entity type** or **identifying entity type**.
- The unique identification is the combination of the key attribute of the parent entity type and of the weak entity type. In this case, the unique identifier of each **PLAYER** instance is **(PLAYER#, TNAME)**.



A subtype hierarchy

- A **subtype hierarchy** comprises an entity supertype and multilevel subtype.
- An important property that the entity type and all entity subtypes must have the common unique identifier, the common key attribute.

Transformation from an ER diagram to Relational Table Structures

พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

STEP 1 Regular Entity Type

- Transform an entity type into a relational table structure.
- All attributes of the entity type, except the multivalued attributes, become attributes of the corresponding relational table structure.
- A key attribute of the entity type is selected to be the primary key of the relational table structure.

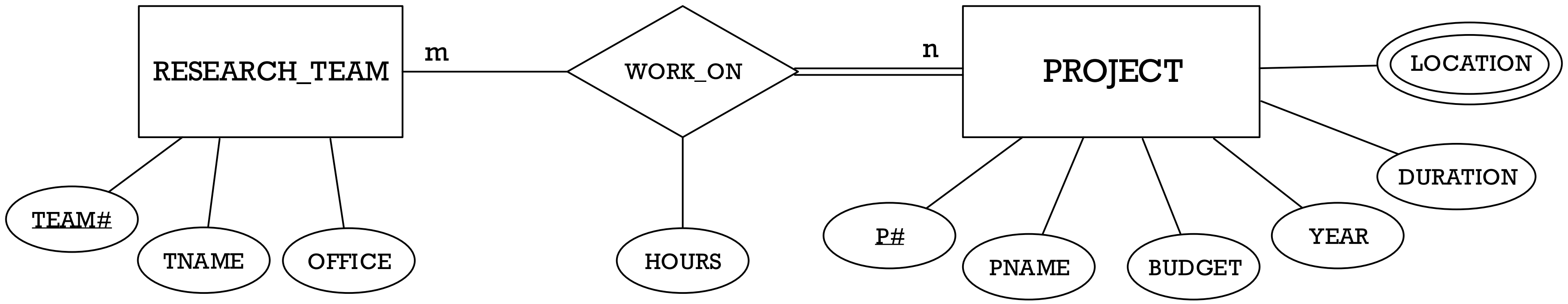
STEP 2 Multivalued Attribute

- Transform a multivalued attribute into a relational table structure.
- The primary key of the relational table structure is the combination of the multivalued attribute and the key attribute from the entity type to which the multivalued attribute is attached.

STEP 3 Many-to-Many Relationship Type

- Transform each m:n relationship type into a relational table structure.
- The primary key is the combined key from the primary key of each participating entity type.
- Attribute(s) of the relationship type, if any, becomes a non-key attribute of the relational table structure.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



STEP 1

RESEARCH_TEAM

TEAM#	**TNAME**	**OFFICE**

PROJECT

P#	**PNAME**	**BUDGET**	**YEAR**	**DURATION**

STEP 2

PROJECT_LOCATION

P#	**LOCATION**

STEP 3

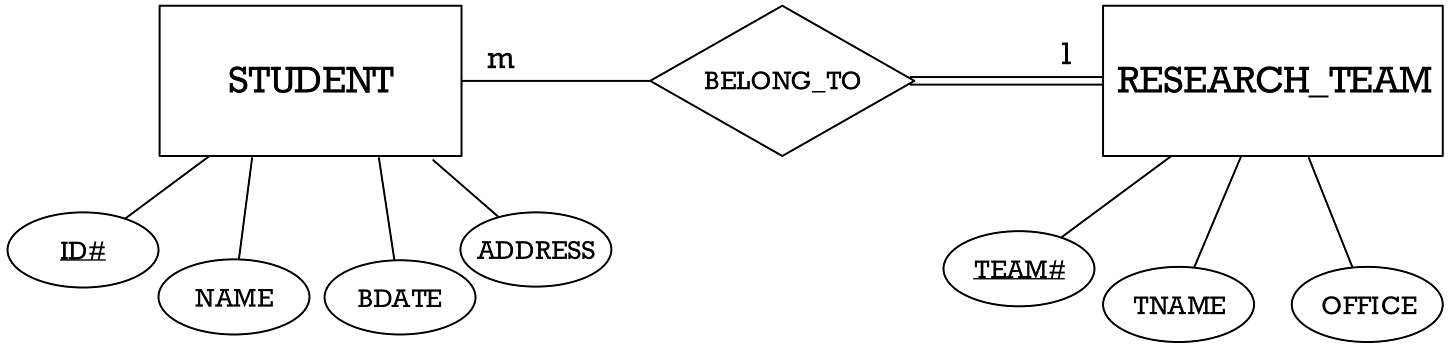
WORK_ON

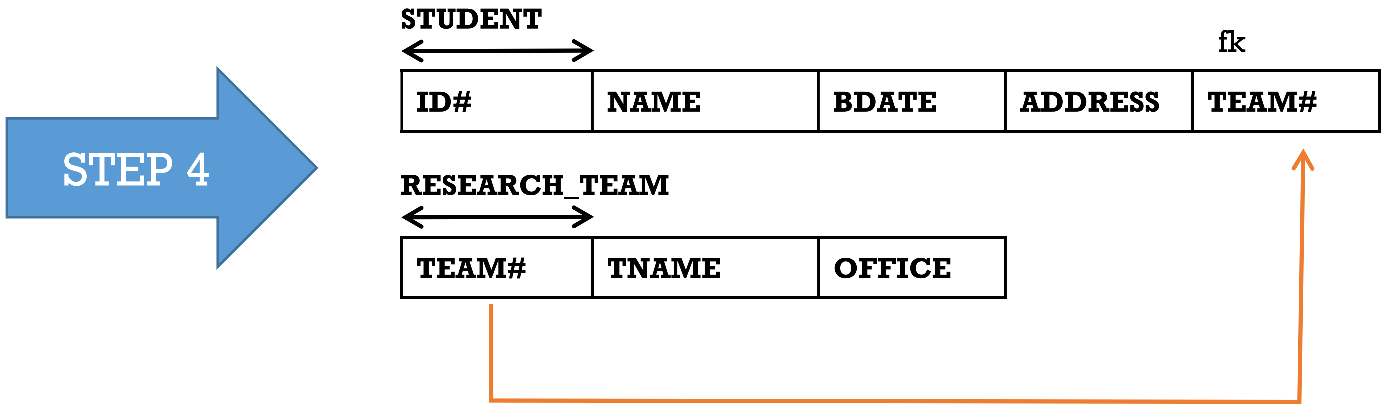
TEAM#	**P#**	**HOURS**

STEP 4 Many-to-One Relationship Type

- Transform each m:1 relationship type into a primary key and foreign key pair at the corresponding relational table structures.
- Use the corresponding relational table structure of the many side as the main side, bring the primary key of the one side to be the foreign key of the many side.
- The attribute(s) of the relationship type become non-key attribute of the main side.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th





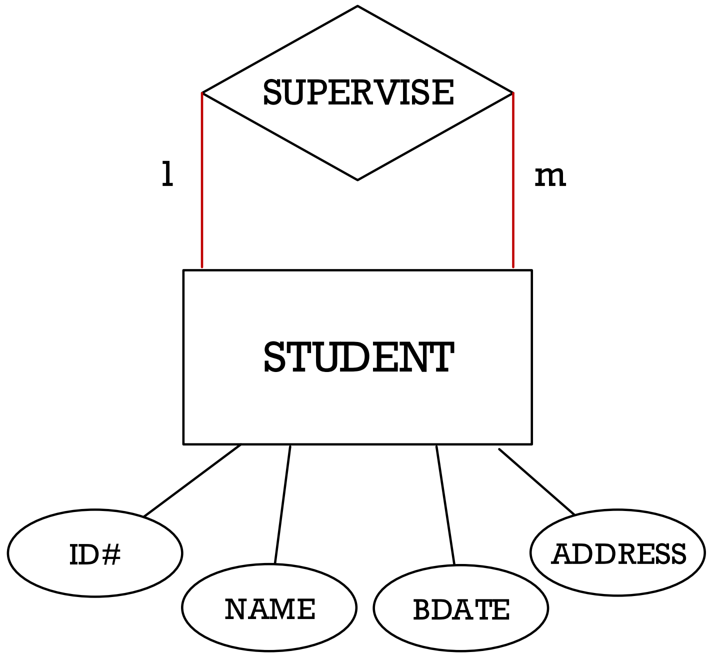
STUDENT

ID#	NAME	BDATE	ADDRESS	TEAM#

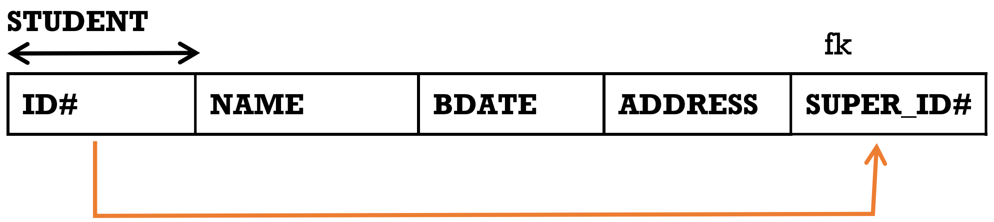
RESEARCH_TEAM

TEAM#	TNAME	OFFICE

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



STEP 4



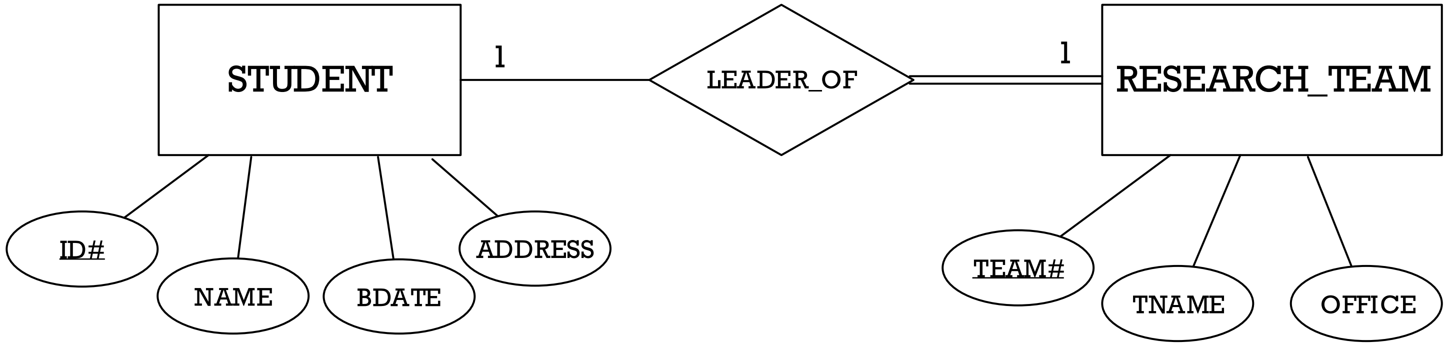
ID#	NAME	BDATE	ADDRESS	SUPER_ID#

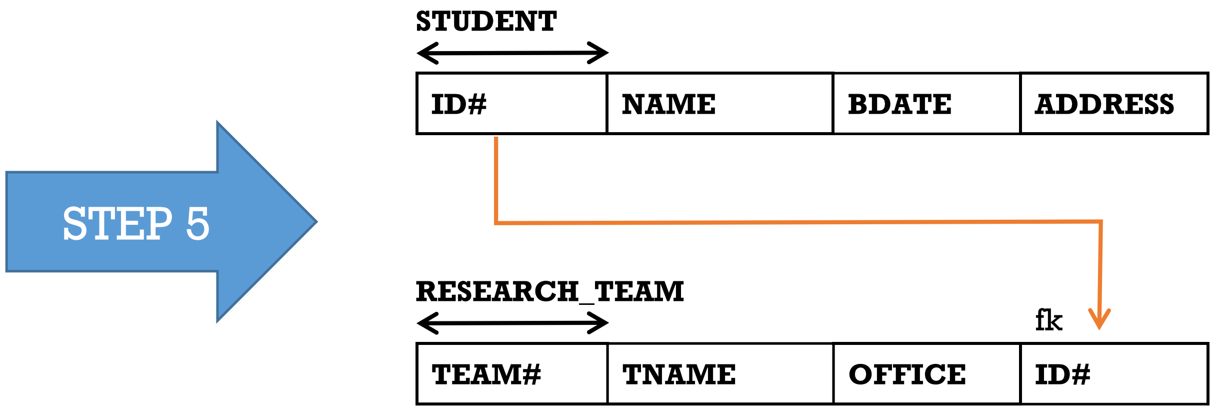
STEP 5 One-to-One Relationship Type

- Transform each 1:1 relationship type into a primary key and foreign key pair at the corresponding relational table structures.
- Use the corresponding relational table structure of the one side as the main side, bring the primary key of the one side to be the foreign key of the many side.
- The attribute(s) of the relationship type become non-key attribute of the main side.

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th





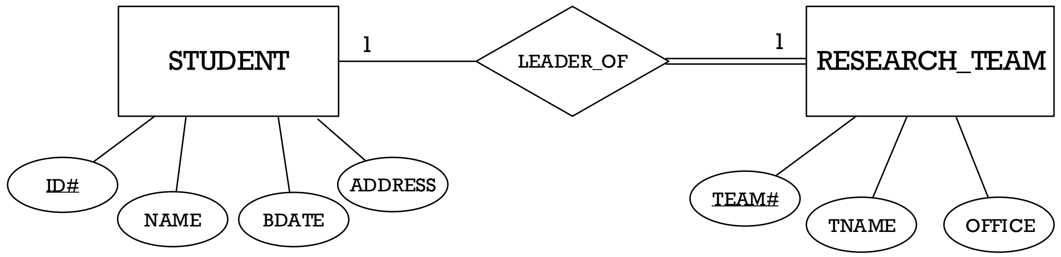
STUDENT

ID#	NAME	BDATE	ADDRESS

RESEARCH_TEAM

TEAM#	TNAME	OFFICE	ID#

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



STUDENT

ID#	NAME	BDATE	ADDRESS

RESEARCH_TEAM

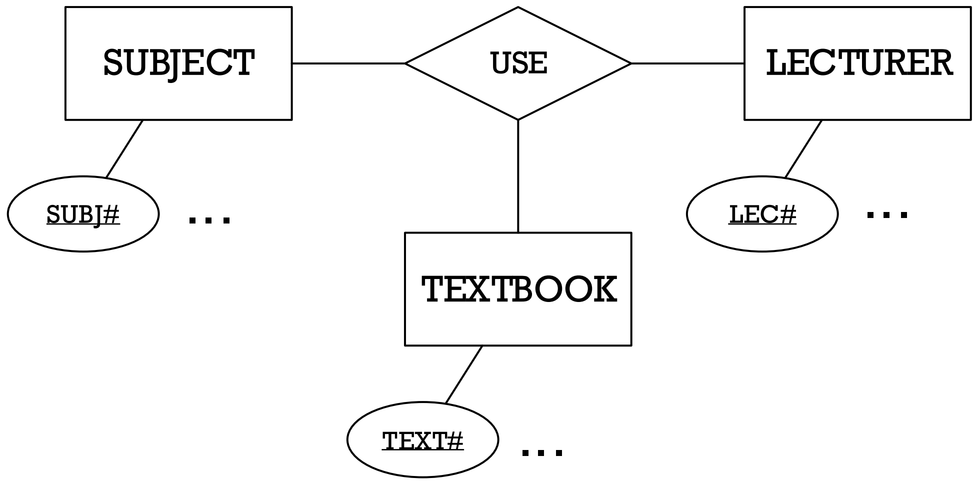
TEAM#	TNAME	OFFICE	ID# (fk)

- The choice of the main side in the 1:1 relationship type should involve the consideration of high priority queries. These queries should have minimum number of join operations. If the queries concentrate more on a corresponding table of an entity type, this entity type should be selected as the main side.
- The total participation constraint has no effect on the choice of the main side in the case of 1:m and m:n relationship type.

STEP 6 N-ary Relationship Type

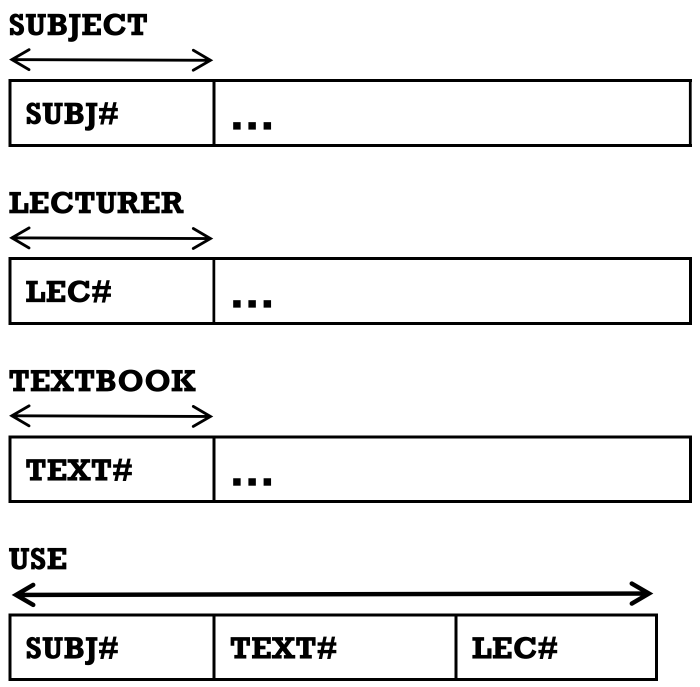
- Transform each n-ary relationship type into a relational table structure.
- The primary key is the combined key from the primary key of each participation entity type.
- Attribute(s) of the relationship type becomes a non-key attribute of the relational table structure.

pakorn.wa@kmitl.ac.th



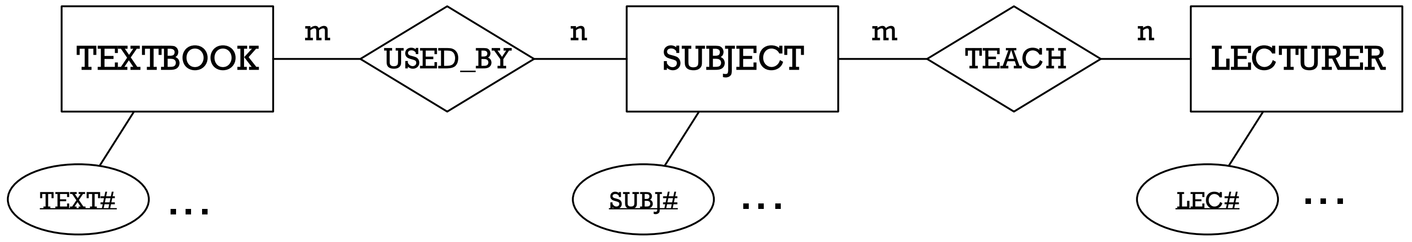
- Notice that all three entity types are related.
  - Teachers use texts in the subjects that they teach. Teachers have the right to choose.

STEP 6



pakorn.wa@kmitl.ac.th

- A common mistake in practice is to draw an n-ary relationship type when the participating entity types are not related.
- E.g.,
  - The teachers cannot choose their own text books, and the text books are selected by a committee.
  - The relationship of the three is no longer a ternary relationship type since **LECTURER** and **TEXTBOOK** are not related.
  - These are two binary relationship types.



SUBJECT

SUBJ#	**...**

LECTURER

LEC#	**...**

TEXTBOOK

TEXT#	**...**

USE

TEXT#	**SUBJ#**

TEACH

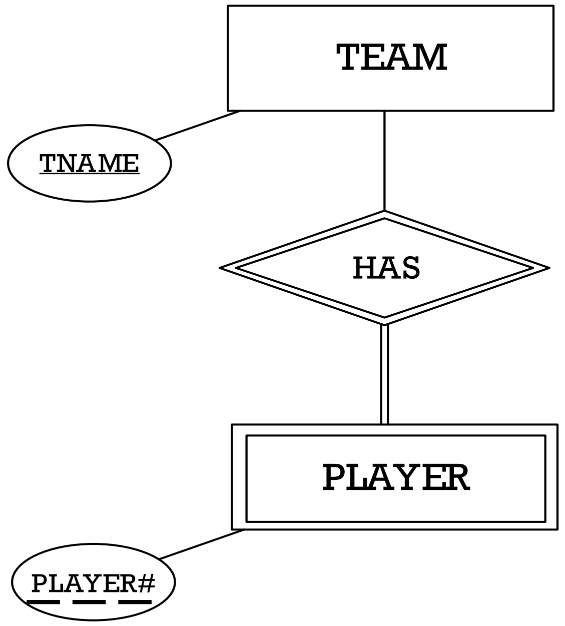
SUBJ#	**LEC#**

pakorn.wa@kmitl.ac.th

STEP 7 Weak Entity Type

- Transform a weak entity type into a relational table structure.
- All attributes of the entity type except the multivalued attributed become attributes of the corresponding relational table structure.
- The key attribute of the identifying (or parent) entity type combined with the partial key of the weak entity type is the primary key of the corresponding relational table structure.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th



STEP 7

TEAM

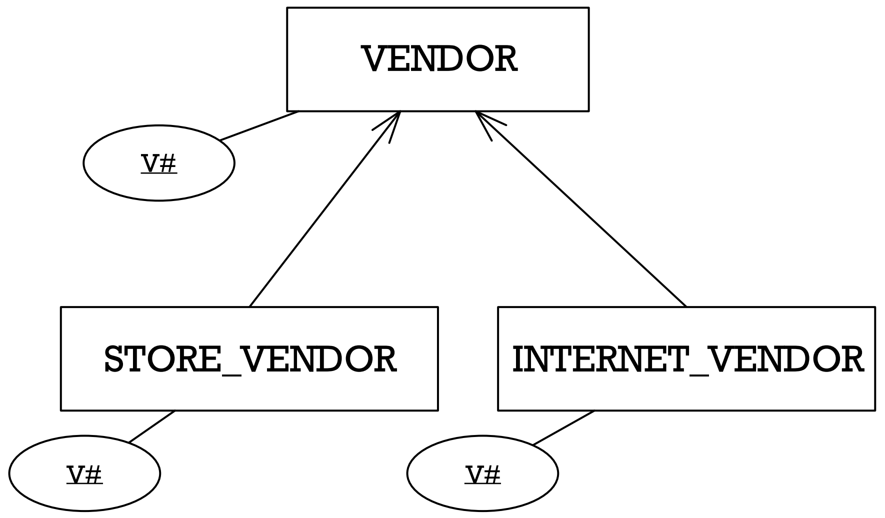
TNAME	...

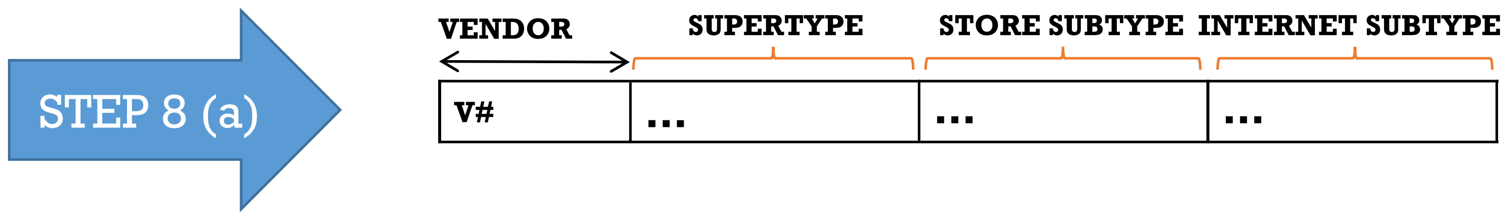
PLAYER

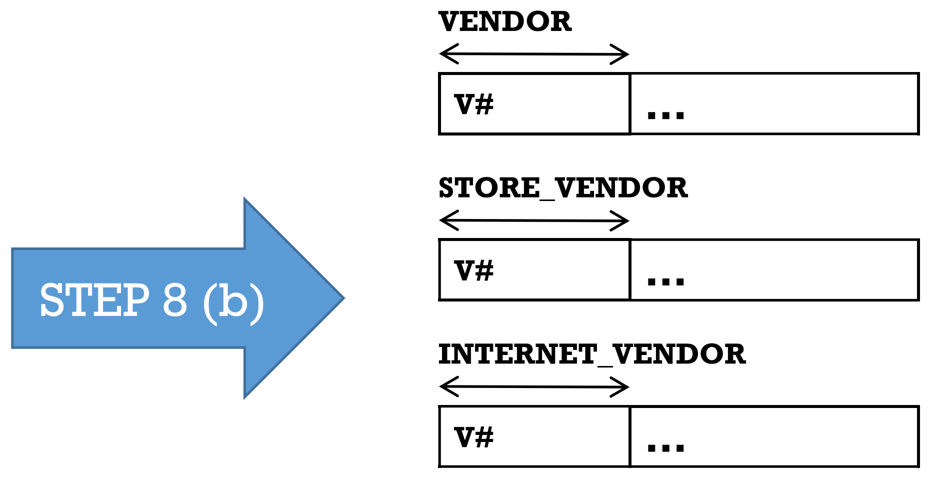
PLAYER#	TNAME	...

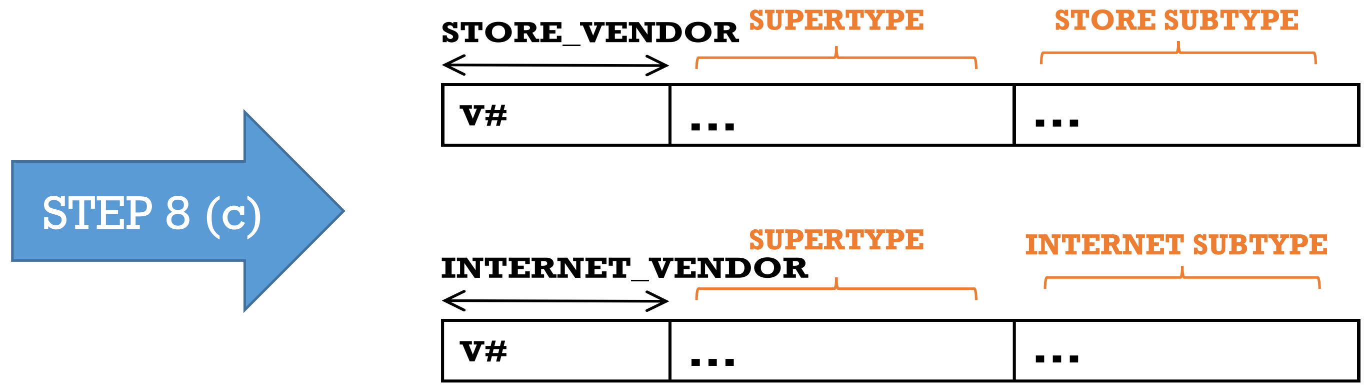
STEP 8 Subtype Hierarchy

- Transform a subtype hierarchy entity type into a relational table structure(s) by
  a) Collapsing the subtype hierarchy into one relational table structure
  b) Having a corresponding relational table structure for each entity subtype and the entity supertype
  c) Having a corresponding relational table structure for each supertype/subtype path


VENDOR
V#
STORE_VENDOR
INTERNET_VENDOR
V#
V#


STEP 8 (a)
VENDOR
SUPERTYPE
STORE SUBTYPE
INTERNET SUBTYPE
V#
...
...
...


STEP 8 (b)
VENDOR
V#
...
STORE_VENDOR
V#
...
INTERNET_VENDOR
V#
...

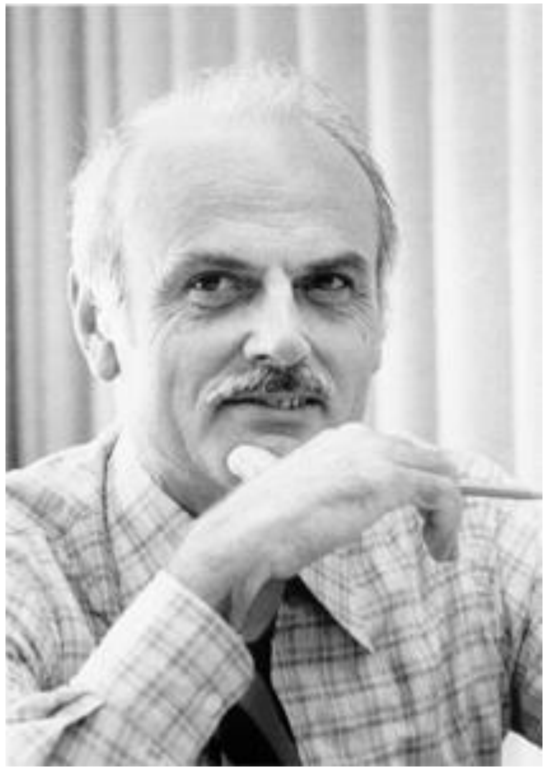

STEP 8 (c)
STORE_VENDOR
SUPERTYPE
STORE SUBTYPE
V#
...
...
INTERNET_VENDOR
SUPERTYPE
INTERNET SUBTYPE
V#
...
...

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Q & A

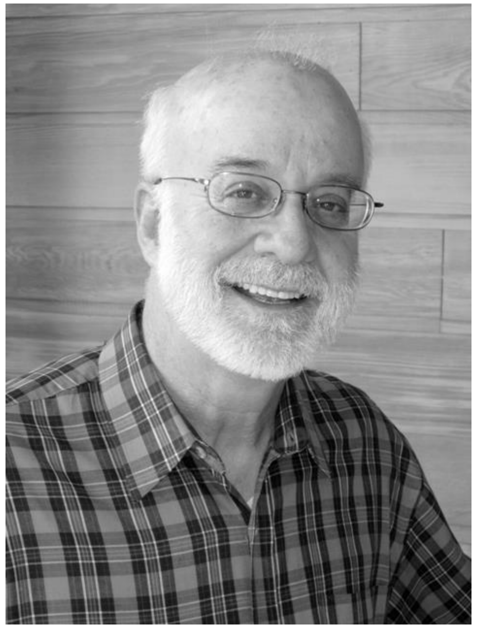
Attribute Analysis using the Normalization Process


Edgar Frank "Ted" Codd
(19 August 1923 – 18 April 2003)
Excerpted from Wikipedia.ORG

- **What is a good relational database structures?**
  - **Free** from operational problems, or **at least reduce** them to be controllable level.
- **Normal Forms**
  - Regarded as **quality control** levels along with its process called the **normalization process**.
  - Invented by E.F. Codd.
  - **1NF**, **2NF**, and **3NF**

pakorn.wa@kmitl.ac.th

Raymond 'Ray' Boyce
(1947–1974)
Excerpted from
alchetron.com/Raymond-F-Boyce


Ronald Fagin
(1945 – present)
Excerpted from Wikipedia.ORG

- **Normal Forms (cont'd)**
  - The new **3NF** or **BCNF** (Boyce/Codd NF), regarding
    the **Functional Dependency** (**FD**)
  - The **4NF** regarding
    the **Multi-Valued Dependency** (**MVD**)
  - The **5NF** regarding
    the **Join Dependency** (**JD**) and
    the **Project/Join NF** (**PJNF**)

- The normalization process alone does not give the number of relational table structures required for a given application.
- It tells us if a given relational table structure is good enough or ready to be deployed or not.
- The normalization process intends to **reduce** insert, delete, and update problems. These problems are caused by misplacements of dependencies between attributes.

- Data redundancies cause the data manipulation problems.
- Once the redundancies and the problems are detected, the solution is to split the relational table structures which have the problems into smaller ones.
- However, some queries may require the join of these smaller table structures back and the query performance may drop.
- *Why do we split the big table into smaller ones, and then join them back for queries?*
    - These insert, delete, and update programs cannot be solved by simply using better hardware.
    - Join operation performance can be improved with better hardware or better physical access mechanism.
    - *We solve problems that the hardware cannot solve first*.

Insert, delete, and update problem

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
V1	David	Bangkok	100	I6	Eraser	White	100

- No more join operations!
- Full of insert, delete, update anomalies.

pakorn.wa@kmitl.ac.th

- Note that, at the database design stage, the database designer do not see relational tables; only relational structures are perceived.
- Database design concentrates on the types, not on the instances. The instances occur during database operations.
- The population instances help visualize how good a relational table is.

Insert anomalies

pakorn.wa@kmitl.ac.th

- An insert operation inserts the entire row(s) into the relational table.
- Two possible problems
    1. Some information cannot be inserted since the row which contains the information violates the primary key not NULL constraint.
    2. Redundant information, or conflict information, can be inserted to the same relational table.

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
V1	David	Bangkok	100	I6	Eraser	White	100

What is the primary key of the FIRST?

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
V1	David	Bangkok	100	I6	Eraser	White	100

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- What if a vendor who does not supply anything is insert?
  - Since the PK of FIRST is (V#, I#) and the PK cannot have null value, the insertion of a vendor without an I# is therefore rejected.
  - Similarly, items that no vendor supplies cannot be inserted; since the V# of the row will be null.
  - When consider the initial data loading case, where only information on vendors and items are available without any sales, these information cannot be inserted at all.

pakorn.wa@kmitl.ac.th

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
V1	David	Bangkok	100	I6	Eraser	White	100
V3	Thomas	…	…	…	…	…	…

- What if a row which states that the vendor V2 supplies the item I3 is inserted but incorrectly states that V3 is Thomas instead of John?
  - This row can be inserted and the DBMS will allow the insertion.
  - However, the inconsistent information about V3's name appears undetected!
- **Inconsistent information** is undesirable especially in data-intensive applications.

- The three relational tables VENDOR, ITEM, and SALE do not have the same insertion problem.
  - Initial loading of vendors and items can be done to the corresponding tables without any sales.
  - Information about particular vendors cannot be duplicated, since V# is the primary key of VENDOR and the primary key values cannot be duplicated.

pakorn.wa@kmitl.ac.th

FIRST_SEQ

SEQ#	V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
1	V3	John	Brisbane	300	I2	Pen	Red	100
2	V2	Peter	Paris	200	I2	Pen	Red	100
3	V1	David	Bangkok	100	I2	Pen	Red	200
4	V1	David	Bangkok	100	I4	Pen	Black	250
5	V4	Tom	Bangkok	100	I1	Ruler	Clear	200
6	V3	John	Brisbane	300	I1	Ruler	Clear	600
7	V2	Peter	Paris	200	I1	Ruler	Clear	50
8	V1	David	Bangkok	100	I1	Ruler	Clear	300
9	V3	John	Brisbane	300	I3	Pencil	Black	200
10	V1	David	Bangkok	100	I3	Pencil	Black	50
11	V4	Tom	Bangkok	100	I5	Ruler	White	100
12	V1	David	Bangkok	100	I5	Ruler	White	400
13	V1	David	Bangkok	100	I6	Eraser	White	100

- Poor fix of the insertion problem
  1. Ignore the primary key concept by not enforcing the primary key constrain at all. This would allow null values, duplicated, and conflict information to be inserted freely into the table.
  2. The incorrect introduction of **surrogate keys** – a system generated or application generated unique key.
     - Since the surrogate key SEQ# is unique and not null, it is chosen to be the primary key instead of (V#, I#) of the original FIRST.
     - Conflict information and null values in correct V# and I# can now be freely inserted undetected! The information in the table is not trustworthy and cannot be used for further analysis.

Deletion anomalies

pakorn.wa@kmitl.ac.th

- The deletion operation removes the entire row(s) from relational tables.
- When only an information on the row needs to be removed,
  a deletion operation may remove other information on the same row unintentionally.

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
~~V1~~	~~David~~	~~Bangkok~~	~~100~~	~~I6~~	~~Eraser~~	~~White~~	~~100~~

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

- What if vendor V1 wants to cancel the sale of the item I6?
  - The actual information which needs to be removed is (V1, I6, 100).
  - The UPDATE operation to remove these values from the row does not work because the primary key V# and I# will be null.
  - The only possible other option is the delete operation which removes the entire row.
  - The removal of (V1, I6, 100) by the deletion operation will also remove the only row which has the details of I6.

pakorn.wa@kmitl.ac.th

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
~~V1~~	~~I6~~	~~100~~

FIRST_SEQ

SEQ#	V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
1	V3	John	Brisbane	300	I2	Pen	Red	100
2	V2	Peter	Paris	200	I2	Pen	Red	100
3	V1	David	Bangkok	100	I2	Pen	Red	200
4	V1	David	Bangkok	100	I4	Pen	Black	250
5	V4	Tom	Bangkok	100	I1	Ruler	Clear	200
6	V3	John	Brisbane	300	I1	Ruler	Clear	600
7	V2	Peter	Paris	200	I1	Ruler	Clear	50
8	V1	David	Bangkok	100	I1	Ruler	Clear	300
9	V3	John	Brisbane	300	I3	Pencil	Black	200
10	V1	David	Bangkok	100	I3	Pencil	Black	50
11	V4	Tom	Bangkok	100	I5	Ruler	White	100
12	V1	David	Bangkok	100	I5	Ruler	White	400
~~13~~	~~V1~~	~~David~~	~~Bangkok~~	~~100~~	~~I6~~	~~Eraser~~	~~White~~	~~100~~

- The insertion and deletion anomalies on the FIRST relational table cannot be solved by using better hardware.
- The performance of join operations on the VENDOR, ITEM, and SALE relational tables, which are required by some queries, can be improved by using better hardware and other performance tuning techniques.

Update anomalies

- **Update anomaly** is the problem of modifying multiple occurrences of the same information within a relational database table but does not have all occurrences modified; thus leads to conflict information of information inconsistencies within a relational table.

pakorn.wa@kmitl.ac.th

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT
V3	John	Brisbane	300	I2	Pen	Red	100
V2	Peter	Paris	200	I2	Pen	Red	100
V1	David	Bangkok	100	I2	Pen	Red	200
V1	David	Bangkok	100	I4	Pen	Black	250
V4	Tom	Bangkok	100	I1	Ruler	Clear	200
V3	John	Brisbane	300	I1	Ruler	Clear	600
V2	Peter	Paris	200	I1	Ruler	Clear	50
V1	David	Bangkok	100	I1	Ruler	Clear	300
V3	John	Brisbane	300	I3	Pencil	Black	200
V1	David	Bangkok	100	I3	Pencil	Black	50
V4	Tom	Bangkok	100	I5	Ruler	White	100
V1	David	Bangkok	100	I5	Ruler	White	400
V1	David	Bangkok	100	I6	Eraser	White	100

- Want to update that V1 moves from Bangkok to Brisbane.
  - SQL statement updates

```
UPDATE    FIRST
SET       LOCATION = 'Brisbane'
WHERE     V# = 'V1'
```

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

ITEM

I#	INAME	COLOR
I2	Pen	Red
I4	Pen	Black
I1	Ruler	Clear
I3	Pencil	Black
I5	Ruler	White
I6	Eraser	White

SALE

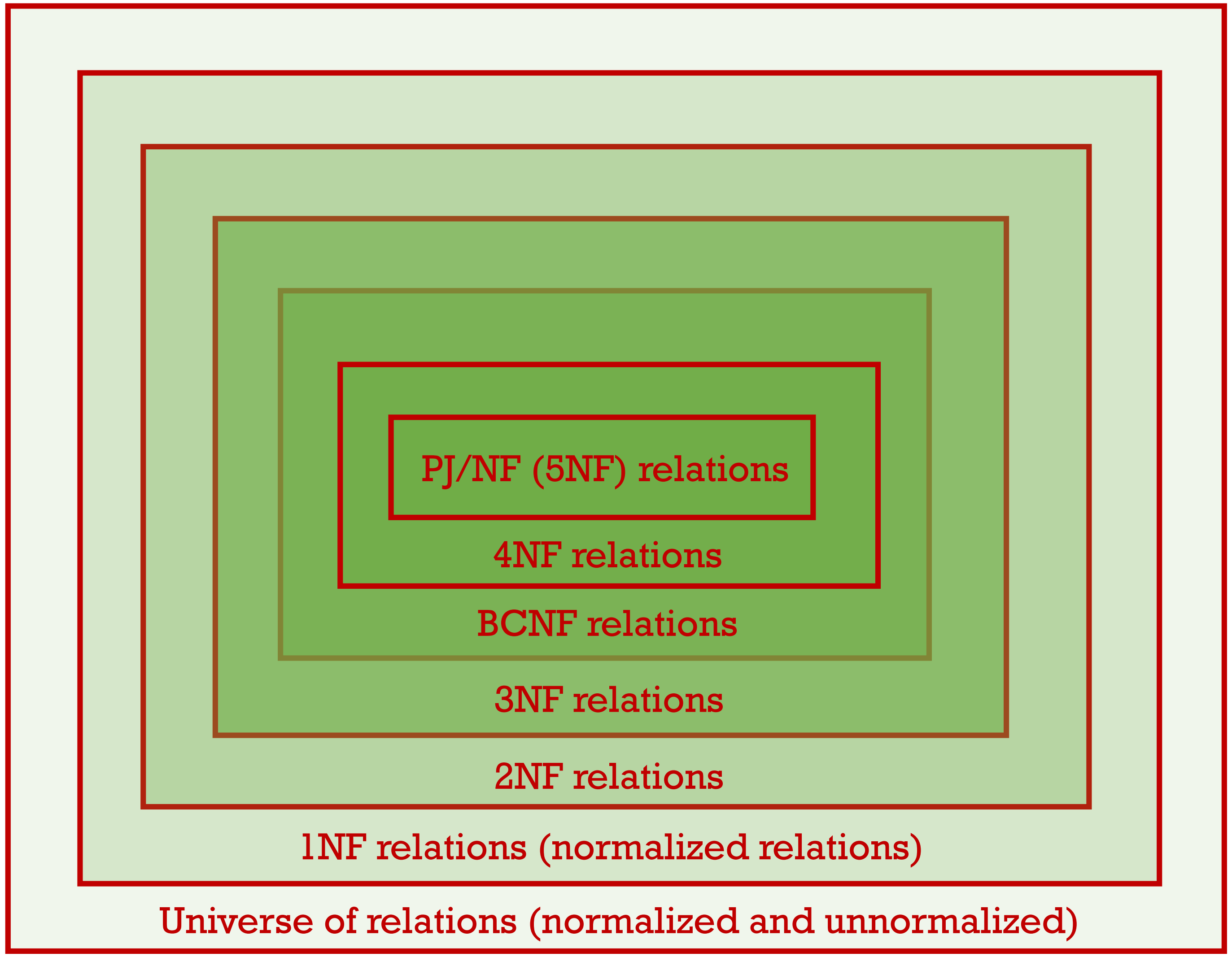
V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

- It seems that the problems are less when the relational table is split into smaller ones.
- In fact, the normalization process **does not encourage splitting** of relational table structures.
- Instead, the normalization process encourages database designers not to split relational tables. It suggests that the number of relational table structures be kept minimum.

The normalization process steps

pakorn.wa@kmitl.ac.th

- The input comprises all attributes and all dependencies between attributes – functional dependency, multivalued dependency, and join dependency.
- The output is a set of relational table structures which guarantee to be free from the insert, delete, and update anomalies, if performed up to the final normal form level, the 5NF.
- In practice, is it necessary to perform the normalization process up to the 5NF?
  - The answer is to perform the normalization process up to the normal form level that the design team understands.
  - There is no need to do things that one does not clearly understand; especially in professional tasks.


PJ/NF (5NF) relations
4NF relations
BCNF relations
3NF relations
2NF relations
1NF relations (normalized relations)
Universe of relations (normalized and unnormalized)

Functional dependence and related normal forms

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- **Definition**
  - A **functional dependency (FD)** is a relationship between attributes in the form of function.
  - Given a relational table structure T,
    attribute B of T is functionally dependent on attribute A of T,
    symbolized by $A \rightarrow B$,
    if and only if,
    whenever two tuples of a relational table of T have the same A value,
    they must have the same B value.
  - A and B may be composite attributes.

- $A \rightarrow B$
  - A is called a **determinant**.
  - B is called a **dependent**.
  - Read A determines B, or
  - B depends on A

VENDOR

V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V5	John	Sydney	300
V2	Peter	Paris	200
V4	Tom	Bangkok	100
V3	John	Brisbane	300

- From the VENDOR table structure,
  - $V\# \rightarrow VNAME$
  - $V\# \rightarrow LOCATION$
  - $V\# \rightarrow STATUS$
- The interpretation
  - Given a V# value,
    there must be only one corresponding VNAME, LOCATION, and STATUS value.
- The FDs must be given.
  They must be known from the application domain.

- **Definition**
  - Attribute B of relational table structure T is **fully functionally dependent** on attribute A of relational table structure T if it is functionally dependent on A and not functionally dependent on any proper subset of A.
  - A and B may be composite attributes.

  - According to the definition, the case where A is a simple attribute (comprises only one attribute) is always a **full FD**.
  - The case where the determinant is a composite may lead to a functional dependency which is not **full FD**.

pakorn.wa@kmitl.ac.th

SALE

V#	I#	AMOUNT
V1	I1	300
V1	I2	200
V3	I2	100
V1	I3	50
V2	I1	50
V2	I2	100
V4	I1	200
V4	I5	100
V3	I3	200
V3	I1	600
V1	I4	250
V1	I5	400
V1	I6	100

- From the SALE table structure
  - $(V\#, I\#) \rightarrow AMOUNT$ is a **full FD**
    since both V# and I# must be known in order to obtain AMOUNT.
  - Only V# or I# alone cannot determine AMOUNT.
  - Said otherwise, AMOUNT fully depends on both (V#, I#)
    since it depends on both of them and not only one of them.

pakorn.wa@kmitl.ac.th

- **Definition**
  - A relational table structure T is in the **first normal form (1NF)**, if and only if, all of its attributes contain atomic values only, and each attribute belongs to only one domain.

BEFORE

S#	PO	
	P#	QTY
S1	P1	300
	P2	200
	P3	400
	P4	200
	P5	100
	P6	100
S2	P1	300
	P2	400
S3	P2	200
S4	P2	200
	P4	300
	P5	400

AFTER

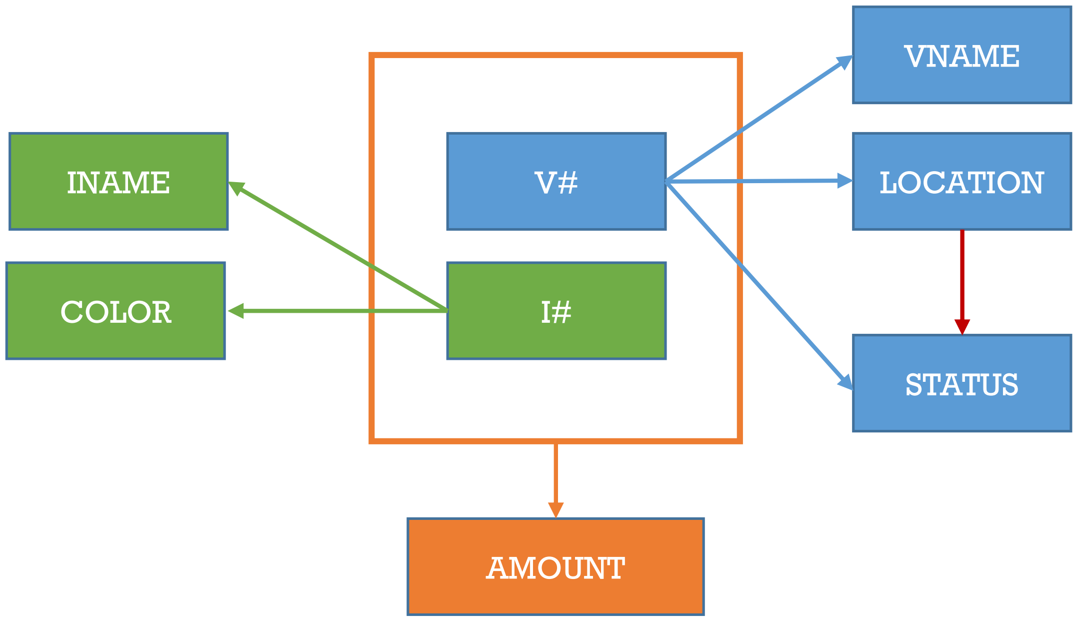
S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

- **Definition**
  - A relational table structure T is in the **second normal form (2NF)**, if and only if, the relationship between the primary key and every non-key attribute is in full functional dependence.
  - The term non-key attribute in this normalization process means an attribute which is not part of a candidate key.

pakorn.wa@kmitl.ac.th

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT



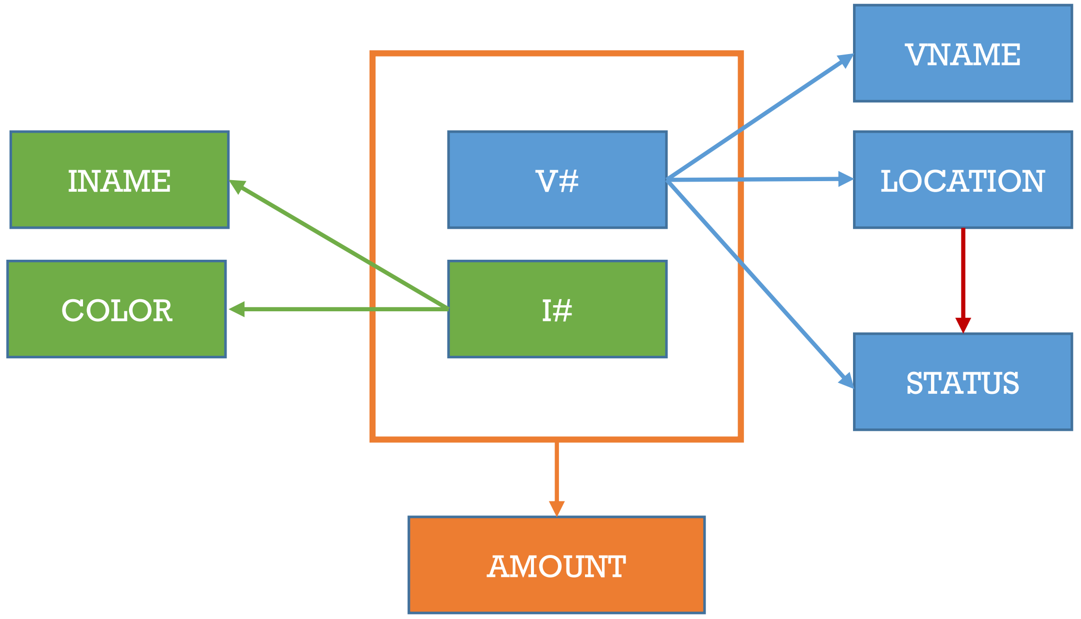
The FD diagram of the table structure in FIRST. The FD diagram is a diagram which shows all functional dependencies in a relational table structure.

- Only AMOUNT is fully dependent on the entire primary key (V#, I#).
  - The other non-key attributes are dependent on a subset of it.
  - VNAME, LOCATION, and STATUS depend on V#.
  - INAME and COLOR depend on I#.
- Therefore, FIRST is NOT in 2NF.

- What should we do now?
  1. Leave FIRST as it is and tolerate the insert, delete, and update problems.
  2. Split the table structure to smaller ones so that all of them agree with the 2NF definition, which leads to less insert, delete, and update problems.

FIRST

V#	VNAME	LOCATION	STATUS	I#	INAME	COLOR	AMOUNT



VENDOR

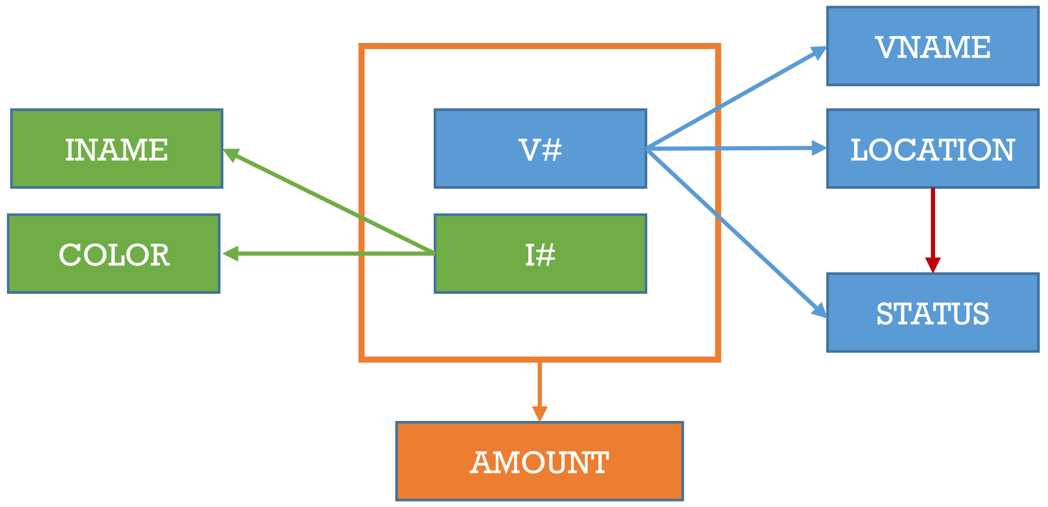
V#	VNAME	LOCATION	STATUS

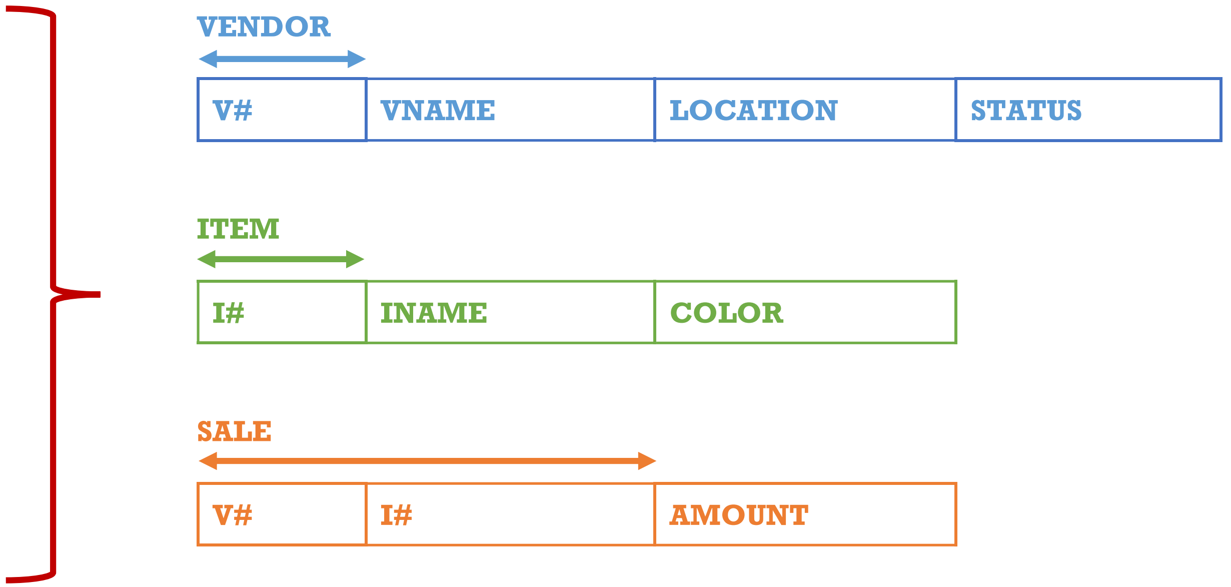
ITEM

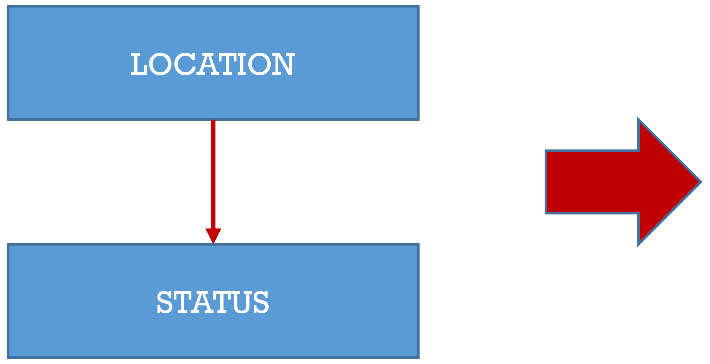
I#	INAME	COLOR

SALE

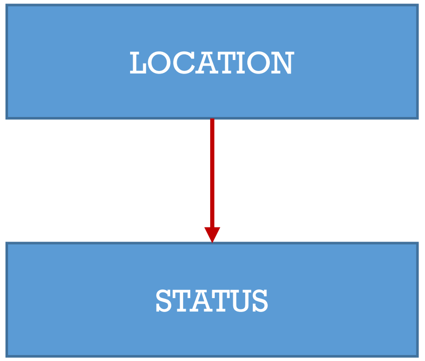
V#	I#	AMOUNT







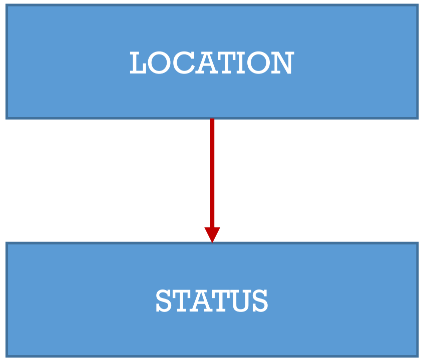
- The status number depends on the location.
- Suppliers who are in the same city must therefore have the same status number.



V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V2	Peter	Paris	200
V4	Tom	Bangkok	100

pakorn.wa@kmitl.ac.th

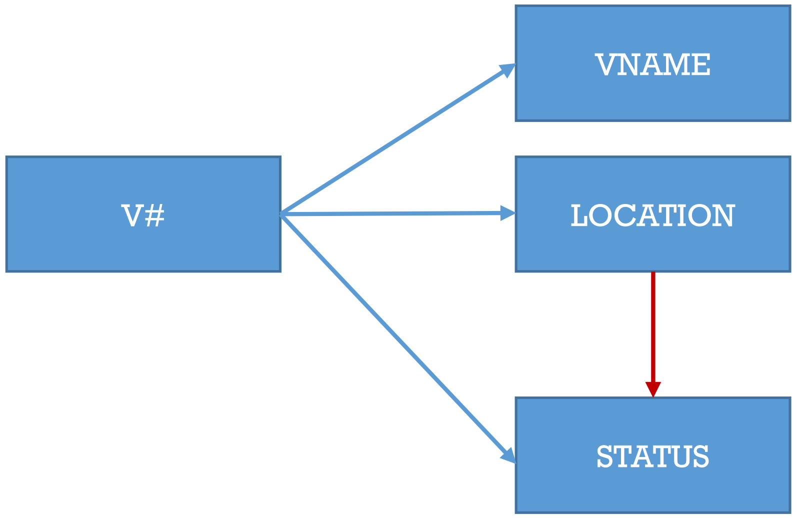
- Because of the FD, the information that Bangkok has status 100 appears as many times as the number of vendors who are in Bangkok.
- An incorrect insertion of a vendor in Bangkok whose status is not 100 is possible and will cause the insertion of conflict information problem.
- Since the PK is V#, the insertion of the information such as (Denver, 500) is not possible if there is no vendor in the city.
- Assume V2 is the only vendor in Paris, what if one day V2 is no longer our vendor and needs to be removed.
- The deletion of the row V2 will result in the loss of the (Paris, 200) information.

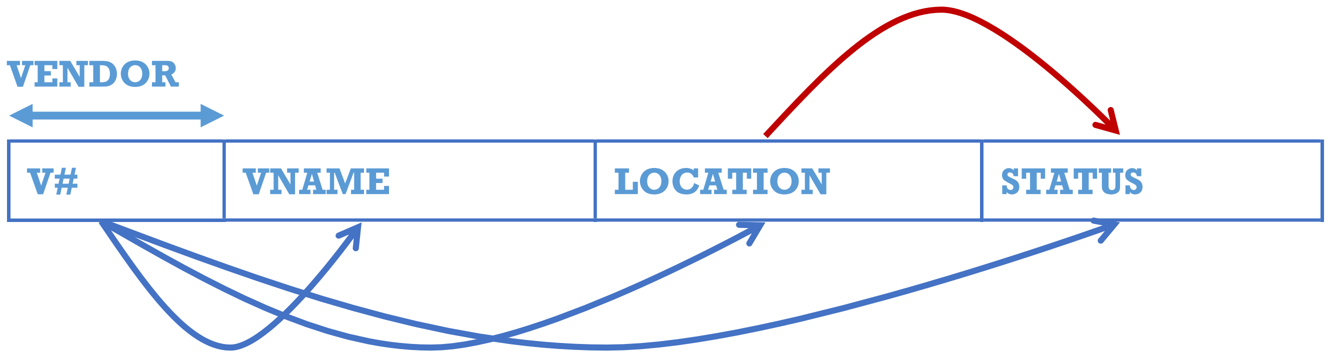


V#	VNAME	LOCATION	STATUS
V1	David	Bangkok	100
V2	Peter	Paris	200
V4	Tom	Bangkok	100

- The FD LOCATION → STATUS between non-key attributes is therefore a misplacement that the 2NF does not take care of.

- **Definition**
  - A relational table structure T is in the **third normal form (3NF)**, if and only if, T is in 2NF and there is no functional dependencies between non-key attributes of T.





VENDOR

V#	VNAME	LOCATION	STATUS

ITEM

I#	INAME	COLOR

SALE

V#	I#	AMOUNT

VENDOR

V#	VNAME	LOCATION

LOCATION

LOCATION	STATUS

ITEM

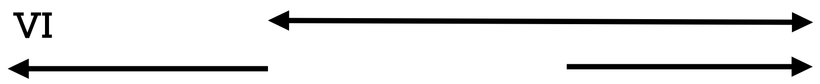
I#	INAME	COLOR

SALE

V#	I#	AMOUNT

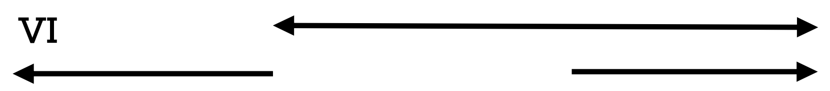
- The 3NF definition is supposed to correctly place FDs to the right relational table structures.
- Later found that problems from FD misplacements still exists in 3NF relational tables.
- The case where the 3NF definition needs to be improved is the case that a relational table structure has multiple candidate keys, which are composite and have at least one attribute in common (overlapped).

pakorn.wa@kmitl.ac.th



V#	VNAME	I#	AMOUNT
V1	David	I1	300
V1	David	I2	200
V1	David	I3	50
V1	David	I4	250
V1	David	I5	400
V1	David	I6	100

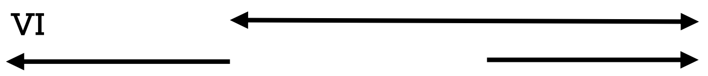
- Assume that V# and VNAME are unique.
- No two vendors have the same V#.
- No two vendors have the same VNAME.
- This relational structure has two candidate keys (V#, I#) and (VNAME, I#), one of them can serve as the primary key.
- Only one non-key attribute AMOUNT which is fully dependent on the PK; therefore, it is in 2NF.
- Since there is only one non-key attribute, there is no FD between non-key attributes; therefore, it is in 3NF.



V#	VNAME	I#	AMOUNT
V1	David	I1	300
V1	David	I2	200
V1	David	I3	50
V1	David	I4	250
V1	David	I5	400
V1	David	I6	100

- (V1, David) appears many times.
- A conflict information such as (V1, John) could still be inserted into the table and causes information inconsistencies.
- A deletion of a row could also lead to an information loss in the case that the vendor only has a sale row.

- **Definition**
  - A relational table structure T is in the **Boyce/Codd normal form (BCNF)**, if and only if, every determinant in T is a candidate key of T.



V#	VNAME	I#	AMOUNT
V1	David	I1	300
V1	David	I2	200
V1	David	I3	50
V1	David	I4	250
V1	David	I5	400
V1	David	I6	100

- There are four FDs and therefore four determinants.
  - (V#, I#) → AMOUNT
  - (VNAME, I#) → AMOUNT
  - V# → VNAME
  - VNAME → V#
- The first two determinants are candidate keys.
- The relational table structure V1 should be split into two smaller structures.
  - (V#, VNAME) , (V#, I#, AMOUNT) or
  - (V#, VNAME) , (VNAME, I#, AMOUNT)

- (V#, VNAME) , (V#, I#, AMOUNT) or
- (V#, VNAME) , (VNAME, I#, AMOUNT)

- The (V#, VNAME) has two determinants and both of them are candidate keys.
- The other two relational table structures have one determinant; each of which is a candidate key.
- Note that most 3NF relational table structures in practice are already in BCNF. The aforementioned relational table structure is uncommon.

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Q & A

pakorn.wa@kmitl.ac.th

Multivalued, Join Dependencies, and Related Normal Forms

- Considering relationships between attributes which are not in the functional format.
  - The functional dependency is a relationship between attributes in the form of function; it is either m:1 or 1:1.
  - What about m:1 or 1:m relationships between attributes?

pakorn.wa@kmitl.ac.th

CTX

COURSE	TEACHER	TEXT
Database	{ John, Smith }	{ Set Theory, DMBS Technology }
Math	{ John }	{ Set Theory, Algebra, Calculus }

- *E.g.*,
  - Each university course is taught by a set of teachers and uses a set of texts.
  - Texts are assigned by the university; teacher has no choice.
  - **TEACHER** and **TEXT** are therefore independent.
  - A text can be used in more than one course, and a course may use several texts.
  - A course may have several teachers and a teacher may teach several courses.

pakorn.wa@kmitl.ac.th

CTX

COURSE	TEACHER	TEXT
Database	{ John Smith }	{ Set Theory DMBS Technology }
Math	{ John }	{ Set Theory Algebra Calculus }

The CTX unnormalized report

CTX

COURSE	TEACHER	TEXT
Database	John	Set Theory
Database	John	DBMS Technology
Database	Smith	Set Theory
Database	Smith	DMBS Technology
Math	John	Set Theory
Math	John	Algebra
Math	John	Calculus

The CTX normalized relational database table

CTX

COURSE	TEACHER	TEXT
Database	John	Set Theory
Database	John	DBMS Technology
Database	Smith	Set Theory
Database	Smith	DMBS Technology
Math	John	Set Theory
Math	John	Algebra
Math	John	Calculus

The CTX normalized relational database table

- What is the PK of the CTX?
- Not everyone can insert the row correctly!
- What if
  - Prof.White teaches Database?
  - The university wants to assign two text books to the Physics course but do not know who will teach the course?
- The PK will contain NULL values and the insertion will be rejected.
- What if
  - Prof.John would like to quit teaching Math?
- All Math text books will be deleted. This is the information loss problem.

pakorn.wa@kmitl.ac.th

CTX

COURSE	TEACHER	TEXT
Database	John	Set Theory
Database	John	DBMS Technology
Database	Smith	Set Theory
Database	Smith	DMBS Technology
Math	John	Set Theory
Math	John	Algebra
Math	John	Calculus

The CTX normalized relational database table

- Some relationship between attributes which are not FDs.
  - → One value of COURSE gives a set of values of TEACHER and a set of values of TEXT.
  - → This kind of dependencies is called **Multivalued Dependencies (MVD)**.

- **Definition**
  - Given a relational table structure T with attribute X, Y, and Z,
    the **multivalued dependence (MVD)** T.X →→T.Y hold in T
    if and only if
    the set of Z-values matching a given (X-value, Y-value) pair in T
    depends only on the X-value and is independent of the Y-value.
  - The X, Y, and Z may be composite.
- **Note**
  - $T.X \twoheadrightarrow T.Y$
  - Read as T.X multidetermines T.Y
  - MVD can exist only if the relational table structure T has at least three attributes.

CTX

COURSE	TEACHER	TEXT
Database	John	Set Theory
Database	John	DBMS Technology
Database	Smith	Set Theory
Database	Smith	DMBS Technology
Math	John	Set Theory
Math	John	Algebra
Math	John	Calculus

The CTX normalized relational database table

- The CTX have the MVD
  - **COURSE** ↠ **TEACHER** and **COURSE** ↠ **TEXT** if and only if the set of values of **TEXT**, which look like depending on both **COURSE** and **TEACHER**, is in fact depends only on **COURSE** and independent of **TEACHER**.
  - This is precisely the given condition of this university.

- Note that other universities where the choice of texts depends on both course and teachers, there will be no MVDs.

- MVDs always come in a pair!
- Given the relational structure T(X, Y, Z), the MVD, the MVD T.X ↠ T.Y hold if and only if the MVD T.X ↠ T.Z also holds.
- MVDs always go together in pairs in this way.
- Therefore, it is common to represent both in a single joint statement, using the notation: $X \twoheadrightarrow Y \mid Z$.

- Relational table structure T, with attributes X, Y, and Z, can be **nonloss-decomposed** into its two projection T1(X,Y) and T2(X,Z) if and only if the MVDs $X \twoheadrightarrow Y \mid Z$ hold in T.

CTX.COURSE ↠ CTX.TEACHER

CTX.COURSE ↠ CTX.TEXT

pakorn.wa@kmitl.ac.th

CTX

COURSE	TEACHER	TEXT
Database	John	Set Theory
Database	John	DBMS Technology
Database	Smith	Set Theory
Database	Smith	DMBS Technology
Math	John	Set Theory
Math	John	Algebra
Math	John	Calculus

CT

COURSE	TEACHER
Database	John
Database	Smith
Math	John

CX

COURSE	TEXT
Database	Set Theory
Database	DBMS Technology
Math	Set Theory
Math	Algebra
Math	Calculus

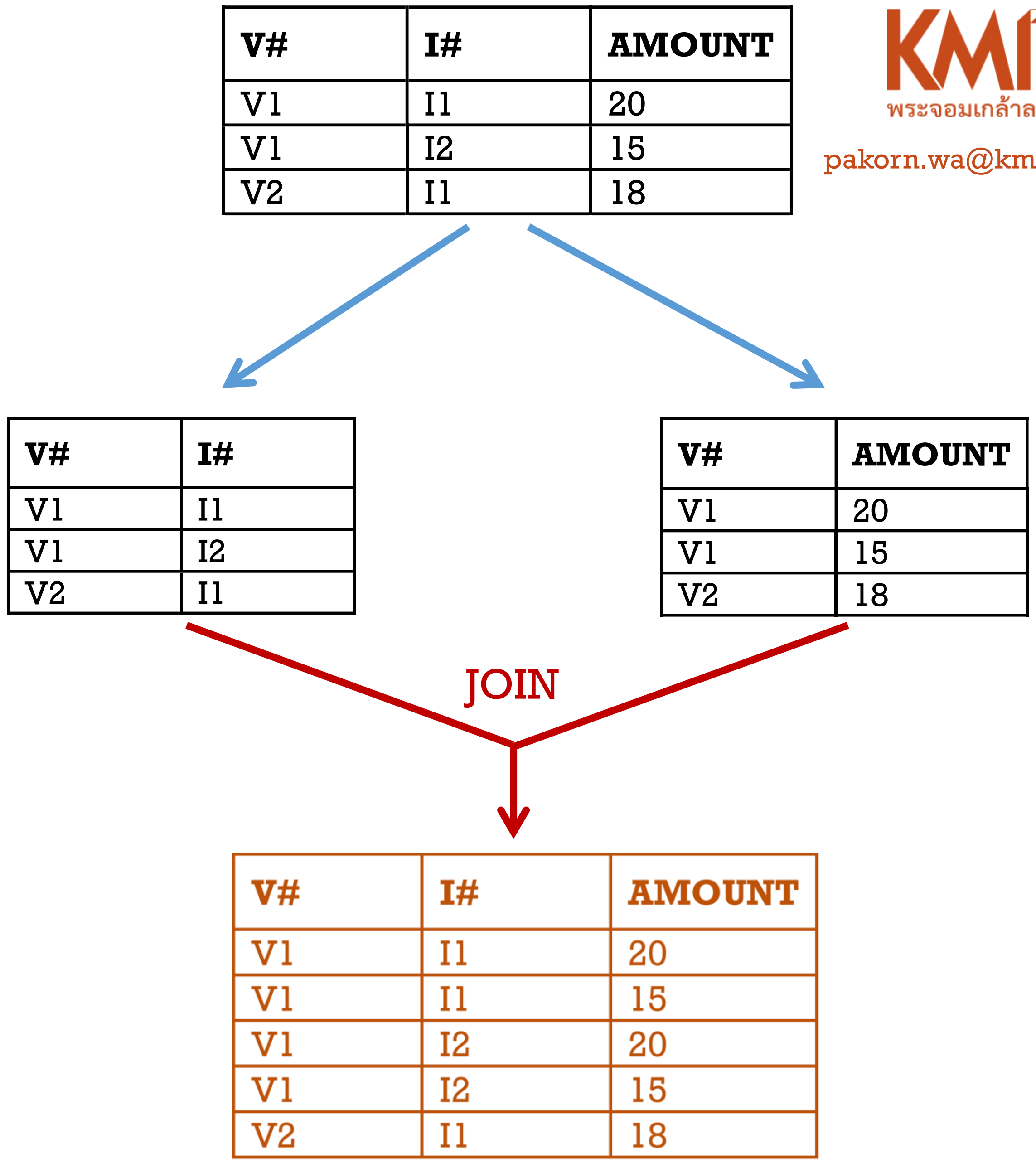
What are the PK of the **CT** and **CX**?

pakorn.wa@kmitl.ac.th

- **Definition**
  - A relational table structure T is in the **fourth normal form (4NF)** if it is in BCNF and all MVDs in T are FDs only.

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The normalization process does not encourage splitting of relational table structures. If the relational table structures are already good, according to a normal form definition, keep them as they are.
- **The lossless join property**
  - A relational table can be split if and only if the projections (small tables after the split) can be joined back to produce the original relational table.



V#	I#	AMOUNT
V1	I1	20
V1	I2	15
V2	I1	18

V#	I#
V1	I1
V1	I2
V2	I1

V#	AMOUNT
V1	20
V1	15
V2	18

V#	I#	AMOUNT
V1	I1	20
V1	I1	15
V1	I2	20
V1	I2	15
V2	I1	18

- The **4NF** is based on the concepts of FDs and MVDs.
- MVDs involve three set of attributes in a relational table structures that can be split into two smaller ones.
- What if
  - There are more than three set of attributes in the relational table structure?
  - There are relational table structures which can be split into three or more smaller ones but not into two?

- **Definition**
  - A relational table structure T satisfies the **join dependency (JD)**: *(A, B, C, …, N) if and only if T is equal to the join of its projections on A, B, C, …, N, where A, B, C, …, N are subsets of the set of attributes of T.

- In other words,
  - "T is equal to the join of its projections on A, B, C, …, N" means T can be split into A, B, C,….,N.
  - Thus, a join dependency is the ability to split a relational table.

- **JDs are the most general form of dependency possible.**

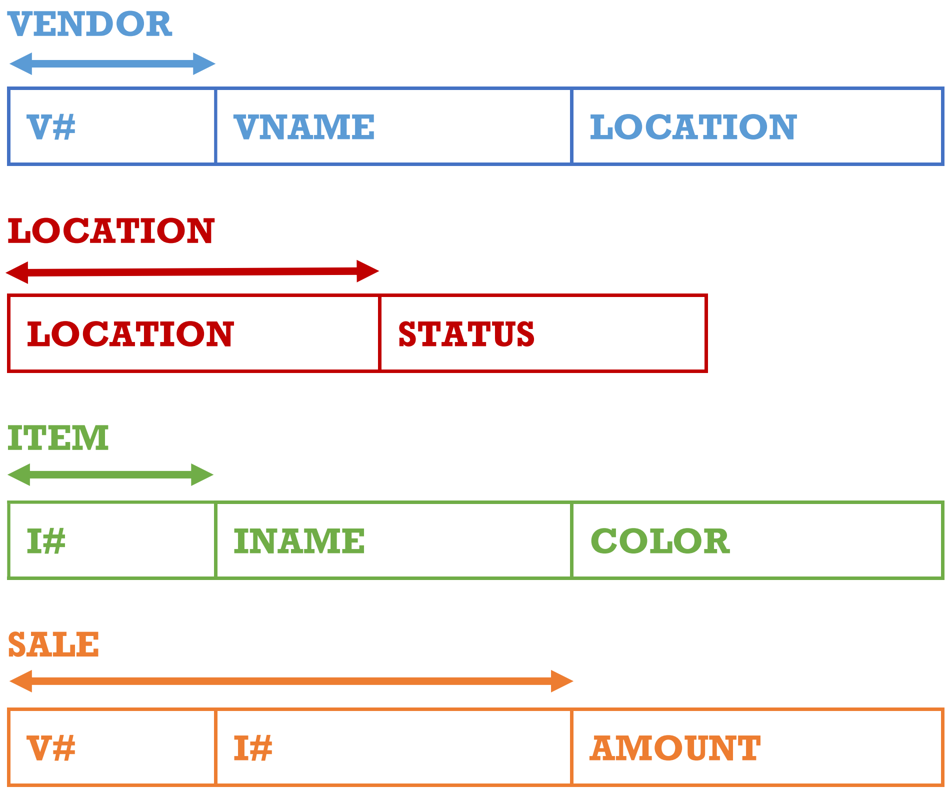
KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

- The relational table structure **VENDOR(<u>V#</u>, VNAME, LOCATION)** has a join dependency ***((V#, VNAME) , (V#, LOCATION)).**
- Suppose email is added to **VENDOR** and become **VENDOR(<u>V#</u>, VNAME, LOCATION, EMAIL).**
  Some JDs are
  - ***((V#,VNAME) , (V#,LOCATION) , (V#,EMAIL))**
  - ***((V#, VNAME, LOCATION) , (V#, EMAIL))**
  - ***((V#, VNAME) , (V#, LOCATION, EMAIL))**
- These splitability properties can be used to decide if a splitable table should be split or should be left as it is.

- **Definition**
  - A relational table structure T is in **projection-join normal form (PJ/NF)** or the **fifth normal form (5NF)** if and only if
    every join dependency in T is implied by the candidate keys of T.
  - Consequently, if a relational table structure cannot be split,
    it is already in 5NF since it has no JD.

- If a relational table structure can be split, should we leave it as it is, or split it?

- The answer is
  if the small tables after the split always have a candidate key of the big table, then leave the big table as it is.



VENDOR

V#	VNAME	LOCATION

LOCATION

LOCATION	STATUS

ITEM

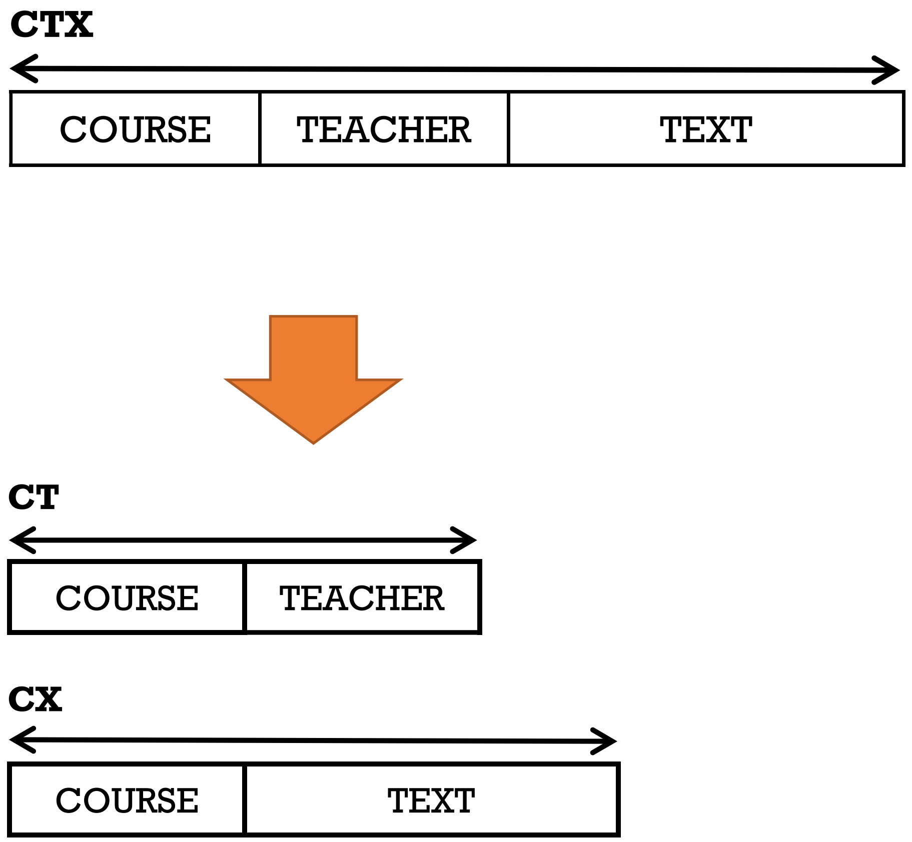
I#	INAME	COLOR

SALE

V#	I#	AMOUNT

- The **LOCATION** and **SALE** cannot be split.
- **VENDOR** and **ITEM** can be split but always with the primary key, so we keep them as they are.

pakorn.wa@kmitl.ac.th



- **COURSE** →→ **TEACHER | TEXT** can be represented by a JD: *((**COURSE**, **TEACHER**) , (**COURSE**,**TEXT**))
- **CTX** is not in the 5NF since each projection (**COURSE**, **TEACHER**) and (**COURSE**,**TEXT**) does not have a candidate key of **CTX**. The candidate key of **CTX** is the three attributes combined.
- The projections are smaller than the candidate key of **CTX** and should be split!

KMITL
พระจอมเกล้าลาดกระบัง
pakorn.wa@kmitl.ac.th

Cyclic Dependency

VIJ

V#	**I#**	**J#**
V1	I1	J2
V1	I2	J1
V2	I1	J1
V1	I1	J1

- The **VIJ** represents the information of vendors supply items to projects.
- What is the PK of the **VIJ**?
- It is in the 4NF – no FDs, no MVDs.
- Not in the 5NF since the projections in the JD are all smaller than the candidate key of the **VIJ**.
- Note that in 5NF every JD is implied by the candidate key – meaning that each projection must have a candidate key of the relational table structure.

VIJ has a JD:
*((V#,I#),(I#,J#),(J#,V#))
but no MVDs

VIJ

V#	I#	J#
V1	I1	J2
V1	I2	J1
V2	I1	J1
V1	I1	J1

VI

V#	I#
V1	I1
V1	I2
V2	I1

IJ

I#	J#
I1	J2
I2	J1
I1	J1

JV

J#	V#
J2	V1
J1	V1
J1	V2

Join over I#

V#	I#	J#
V1	I1	J2
V1	I1	J1
V1	I2	J1
V2	I1	J2
V2	I1	J1

Join over (J#, V#)

VIJ

V#	I#	J#
V1	I1	J2
V1	I2	J1
V2	I1	J1
V1	I1	J1

- The VIJ, since it does not satisfy the 5NF definition, it should then be split into three smaller relational table structures, VI, IJ, and JV – all of which are in 5NF.

- Note that the cyclic dependency is a rare form of JD.

- The VIJ relational table actually combines three separate information that are not related.
- If properly designed, an ER diagram would show relationship types between entity type VENDOR, ITEM, and PROJECT.
- The three relational table structures VI, IJ, and JV are therefore obtained from the three corresponding relationship types.

pakorn.wa@kmitl.ac.th

VIJ

V#	**I#**	**J#**
V1	I1	J2
V1	I2	J1
V2	I1	J1
V1	I1	J1

VI

V#	**I#**
V1	I1
V1	I2
V2	I1

IJ

I#	**J#**
I1	J2
I2	J1
I1	J1

JV

J#	**V#**
J2	V1
J1	V1
J1	V2

- The VIJ table, in fact, should not be a table.
- Instead, it should be a "**view**" for producing reports or even an output result file which is obtained from the join of the three relation tables.
- This view or output result file should not be modified.
- Never allow the modification of a view, or a report file, which is not in 5NF and is the result of the joins.

FDs, MVDs, and JDs are integrity constraints that database operations must adhere to.

KMITL
พระจอมเกล้าลาดกระบัง

pakorn.wa@kmitl.ac.th

Q & A