ជំពូកទី ៤

Entity-Relationship Model

Entity-Relationship (E-R) model គឺជា high-level conceptual data model ដែលត្រូវបានបង្កើតឡើងដោយ Chen ក្នុងឆ្នាំ 1976 ដើម្បីជួយសំរួលការកសាង database។ Conceptual data model គឺជាសំនុំនៃ concepts ដែលពណ៌នាអំពីរចនាសម្ព័ន្ធ (structure) នៃ database និង retrieval and update transaction លើ database។ គោលបំណងជាចំបងក្នុងការកសាង high-level data model គឺដើម្បីផ្តត់ផ្តង់ការយល់ឃើញទៅលើទិន្នន័យរបស់ user និងលាក់បំពាំងចំនុចបច្ចេកទេសពាក់ព័ន្ធដល់ការកសាង database (The main purpose for developing a high-level data model is to support a user's perception of the data and to conceal the more technical aspects associated with database design.) លើសពីនេះទៀត conceptual data model គឺមិនអាស្រ័យទៅនឹង DBMS និង Hardware platform ដែលប្រើប្រាស់សំរាប់បង្កើត database ទេ។

1. The Concepts of the Entity-Relationship Model

Basic concepts នៃ entity-relationship model រួមមាន entity types, relationship types, and attributes។

1. 1. Entity Types (Entity Sets)

Entity is a person, place, object, or event in the user environment about which the organization wishes to maintain data.

Entity គឺជាមនុស្ស, ទីកន្លែង, វត្ថុ វីព្រឹត្តិការណ៍ក្នុងមជ្ឈដ្ឋានអ្នកប្រើប្រាស់ដែល organization ចង់ធ្វើការថែទាំទិន្នន័យ។

ឧទាហរណ៍មួយចំនួនទៅលើ entities:

Person : EMPLOYEE, STUDENT, PATIENT
Place : STORE, WAREHOUSE, STATE
Object : MACHINE, BUILDING, AUTOMOBILE
Event : SALE, REGISTRATION, RENEWAL

Entity type (Entity set) is a collection of entities that share common properties or characteristics.

Entity type គឺជាការប្រមូលផ្តុំ entities ដែលប្រើប្រាស់លក្ខណៈរួម។ គ្រប់ entity type ក្នុង entity-relationship diagram តែងតែមានឈ្មោះដែលស្ថិតក្នុងចតុកោណកែងមួយ (rectangle) ហើយឈ្មោះនោះច្រើនតែជាអក្សរធំទាំងអស់ ហើយមានលក្ខណៈ singular។

(រូប 4.1)

Entity instance or **Entity occurrence** is a single occurrence of an entity type.

Entity instance រឺ Entity occurrence គឺសំដៅលើការលេចឡើង រឺបង្ហាញតែម្តងក្នុង entity type។

ឧទាហរណ៍:

Entity type: EMPLOYEE
Attributes:

EMPLOYEE NUMBER	CHAR(10)
NAME	CHAR(25)
ADDRESS	CHAR(30)
ZIP	CHAR(9)
DATE HIRED	DATE
BIRTHDATE	DATE

Two Instances of EMPLOYEE:

642-17-8360	534-10-1971
Michelle Brady	David Johnson
100 Pacific Avenue	450 Redwood Drive
San Francisco	Redwood City
98173	97142
03-21-1992	08-16-1994
06-19-1968	09-04-1975

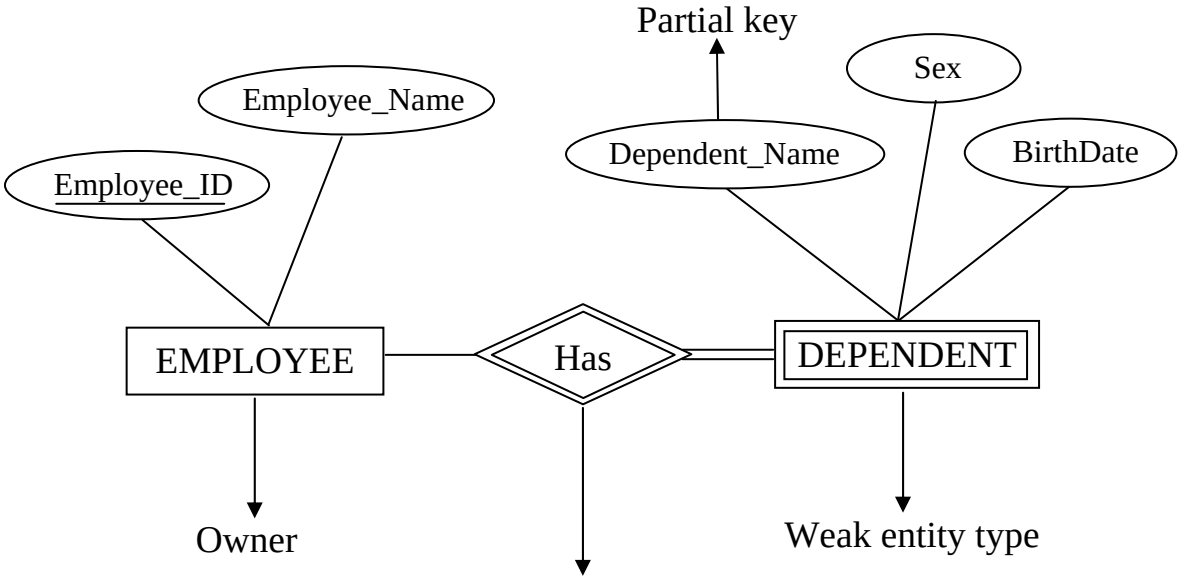
(រូប 4.2)

Strong entity type (Strong entity set) is an entity type that exists independently of other entity type.

Strong entity type គឺជា entity type ដែលកើតមានឡើងដោយមិនពឹងពាក់លើ entity type ដ៏ទៃទៀត។ ឧទាហរណ៍: STUDENT, EMPLOYEE, AUTOMOBILE និង COURSE សុទ្ធតែជា strong entity type។

Weak entity type (Weak entity set) is an entity type whose existence depends on some other entity type.

Weak entity type គឺជា entity type ដែលកើតមានឡើងដោយពឹងពាក់លើ entity type ផ្សេងទៀត។ Weak entity type គ្មានអត្ថន័យគ្រប់គ្រាន់ទេខាងវិស័យជំនួញនៅក្នុង ER diagram ប្រសិនបើគ្មានការចូលរួមពី entity type ដែលវាពឹងពាក់ទេ។ Entity type ដែលវាពឹងពាក់ គេហៅថា **identifying owner** ឬ **owner**។ Weak entity type គ្មាន identifier កើតឡើងដោយ attributes របស់វាទេ ហើយជាទូទៅក្នុង ER diagram, weak entity type មាន attributes បំរើការជា **partial key**។ Full key នៃ weak entity type កើតពីការចូលផ្សំគ្នារវាង partial key និង key នៃ owner។ Weak entity type បង្ហាញដោយរូប double rectangles។ ឧទាហរណ៍: DEPEDENT គឺជា weak entity type ព្រោះវាក៏កើតឡើងបានអាស្រ័យលើ owner entity ដែលតភ្ជាប់ជាមួយវា។



(រូប 4.3)

1. 2. Attributes

Attribute is a property of an entity or a relationship type.

Attribute គឺជាលក្ខណៈ ឬជាបំណែកព័ត៌មាននៃ entity ឬ relationship។ ជាងនេះទៅទៀត រាល់ entity instance ទាំងអស់នៃ entity type សុទ្ធតែមាន attributes ដូចគ្នាទាំងអស់។ Attribute នៃ entity ផ្តុកតំលៃដែលពណ៌នាអំពី entity នីមួយៗ ហើយតំលៃនោះបង្ហាញពីចំនុចដ៏សំខាន់នៃទិន្នន័យវក្បាទុកក្នុង database។

ឧទាហរណ៍: BRANCH entity មាន attributes ដូចជា branch number (Branch_No), address (Address), phone number (Telephone) និង fax number (Fax_No)។

Attribute domain is a set of values that may be assigned to an attribute.

Attribute domain គឺជាសំនុំតំលៃដែលបានកំនត់ទៅលើ attribute។ រាល់ attributes ទាំងអស់សុទ្ធតែមានតំលៃដែលអនុញ្ញាតិអោយបញ្ចូល ឬ domain របស់គេផ្ទាល់ ហើយជាងនេះទៅទៀតការបញ្ចូលទិន្នន័យទៅកាន់ attribute នៃ entity type សុទ្ធតែទាញចេញពី domain របស់វាទាំងអស់។

ឧទាហរណ៍: Domain នៃ attribute Sex មានប្រភេទទិន្នន័យជាតួអក្សរ (string or text) ហើយបញ្ចូលបានតែ 1 តួអក្សរគត់គឺ M ឬ F។

Data Type : Text
Field Size : 1
Values : M or F

Domain នៃ attribute Score មានប្រភេទទិន្នន័យជាលេខមានក្បៀស (float or single) ហើយអាចបញ្ចូលតំលៃចាប់ពី 0 ទៅកាន់ 100។

Data Type : Number
Field Size : Single
Values : 1 to 100

Domain នៃ attribute Employee_Name មានប្រភេទទិន្នន័យជាអក្សរ (string or text) ហើយបញ្ចូលបានតែ 50 តួអក្សរគត់។

Data Type : Text
Field Size : 50
Values : Unlimited

Attributes មួយចំនួនអាចប្រើប្រាស់ domain រួម (Attributes may share a domain)។ ឧទាហរណ៍: Address attributes នៃ Staff និង Owner entities ប្រើប្រាស់ domain រួម។

យើងធ្វើការបែកចែកចំណាត់ថ្នាក់ attribute ជា: simple or composite; single-valued or multi-valued; or derived.

A. Simple Attribute

Simple (Atomic) Attribute is an attribute that cannot be broken down into smaller components.

Simple (Atomic) attribute គឺជា attribute ដែលមិនអាចបំបែកតទៅទៀតបាន។

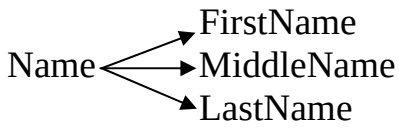
ឧទាហរណ៍ : Attribute Sex, Employee_ID, Weight, BirthDate ជា Simple attribute ពីព្រោះវាមិនអាចធ្វើការបំបែកតទៅទៀតបាន។

B. Composite Attribute

Composite Attribute is an attribute that can be broken down into component parts.

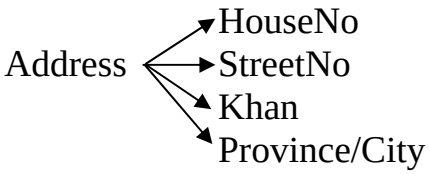
Composite attribute គឺជា attribute ដែលអាចធ្វើការបំបែកទៅជាផ្នែកតូចៗ តទៅទៀតបាន។ ជាពិសេស attribute បែបនេះគេអាចបំបែក រឺមិនបំបែកអាស្រ័យទៅតាមតំរូវការរបស់អ្នកប្រើប្រាស់ database ផ្តាល់ ហើយជាទូទៅគេតែងតែបំបែក composite attribute ទៅជា simple attribute ព្រោះវាមានភាពងាយស្រួលក្នុងការស្វែងរក (easy to search)។

ឧទាហរណ៍ : Attribute Name គឺជា Composite attribute ព្រោះវាអាចធ្វើការបំបែកទៅជា FirstName, MiddleName និង LastName បាន។



តែយើងសង្កេតឃើញថា attribute Name ជាទូទៅគេច្រើនតែបំបែកទៅជា FirstName និង LastName រឺមិនធ្វើការបំបែកតែម្តង ពីព្រោះយើងពុំទាន់ស៊ាំទៅនឹងការបំបែកចេញច្រើនបែបនេះទេ និងជាពិសេសឈ្មោះរបស់យើងកំរមាន MiddleName ណាស់។

Attribute Address យើងអាចបំបែកទៅជា HouseNo, StreetNo, Khan, Province/City,...។



C. Single-valued Attribute

Single-valued Attribute is an attribute that holds a single value for a single entity.

Single-valued attribute គឺជា attribute ទាំងឡាយណាដែលបញ្ចូលតំលៃតែមួយគត់ចំពោះមួយ single entity។

ឧទាហរណ៍ : Attribute Sex នៃ Students entity type យើងអាចធ្វើការបញ្ចូលទិន្នន័យបានតែមួយគត់ គឺ Male រឺ Female ដោយមិនអាចបញ្ចូលទាំងពីរបានទេ។ យើងដឹងហើយថាមនុស្សម្នាក់តែងតែមានអាយុរបស់ខ្លួនតែមួយគត់ បានន័យថា បើអាយុ 18 គឺ 18 មិនអាចនិយាយថា លោកម្នាក់នោះមានអាយុដល់ទៅ 2 គឺ 18 និង 21 ទេ ដូចនេះ attribute Age

នៃ Students entity ជា single-valued attribute ពីព្រោះយើងអាចបញ្ចូលបានតែមួយតំលៃតែប៉ុណ្ណោះទេ។

D. Multi-valued Attribute

Multi-valued Attribute is an attribute that holds a multiple values for a single entity.

Multi-valued attribute គឺជា attribute ទាំងឡាយណាដែលអាចបញ្ចូលតំលៃជាច្រើនចំពោះមួយ single entity។

ឧទាហរណ៍ : Branch entity អាចមានទូរស័ព្ទជាច្រើនដូចជា 011 721727, 012 754155 និង 016 876312 ដូចនេះក្នុងករណីនេះ Tel_No ជា multiple-valued attribute។ Multi-valued attribute អាចមានសំនុំតំលៃលេខកំនត់ពីតំលៃតូចបំផុតទៅដល់តំលៃធំបំផុត (Multi-valued attribute may have a set of numbers with upper and lower limits)។ ឧទាហរណ៍ : គេបានកំនត់ថា Branch នីមួយៗមានចំនួនទូរស័ព្ទពី 1 ទៅ 10 ជាដើម។

Staff entity មាន attribute Skill សំរាប់កត់ត្រា Skill ដែល Staff ចេះហើយម្យ៉ាងវិញទៀតដោយសារ Staff អាចមាន Skill លើសពី 1 ដូចជា C Programming Language, Pascal Programming Language និង Visual Basic Programming Language ដូច្នេះក្នុងករណីនេះ Skill ជា multi-valued attribute។

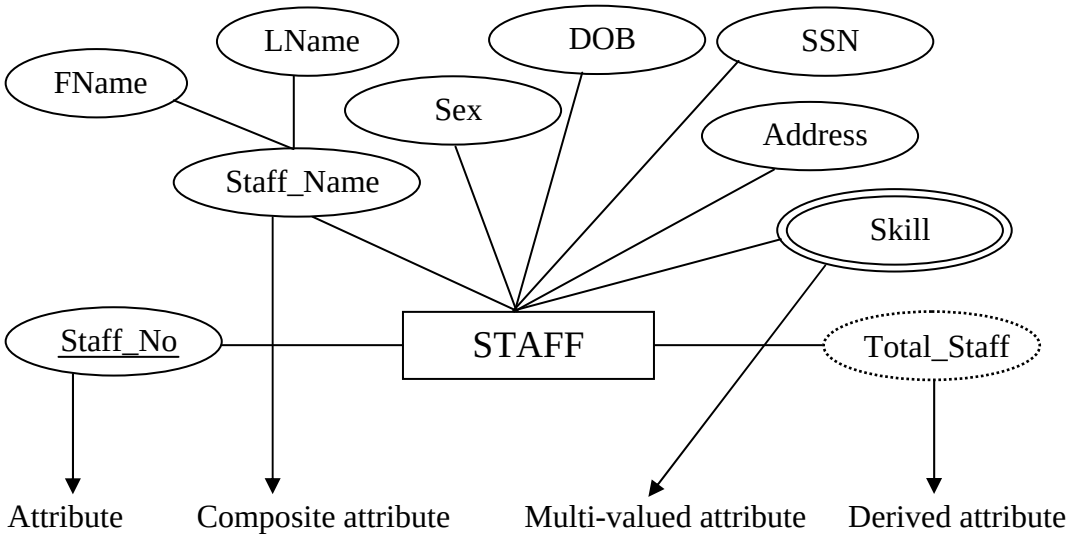
E. Derived Attribute

Derived Attribute is an attribute whose values can be calculated from related attribute values or a set of attributes values (plus possibly data not in the database, such as today's date, the current time, or a security code provided by a system user).

Derived attribute គឺជា attribute ដែលតំលៃរបស់វាគណនាចេញពីតំលៃ attribute មួយ រឺសំនុំនៃ attributes (ក្នុងករណីមួយចំនួន ទិន្នន័យនោះមិនស្ថិតក្នុង database ដូចជា today's date, the current time, or a security code ផ្តល់ដោយ system user)។

ឧទាហរណ៍ : Age attribute កើតឡើងដោយប្រើប្រាស់អនុគមន៍សំរាប់គណនារកអាយុទៅលើ BirthDate attribute ដូចនេះ attribute Age ជា derived attribute។

Total_Staff attribute កើតឡើងពីការរាប់ (count) ទៅលើ staffs ទាំងអស់ក្នុង Staff entity។



(រូប 4.4)

Keys or Identifier

យើងចាត់ទុក Key ជា data item ដែលអាចធ្វើការកំនត់អត្តសញ្ញាណនៃ entity នៅក្នុង entity type។ Key ចែកជា:

Candidate Key An attribute or set of attributes that uniquely identifies individual occurrences of an entity type.

Candidate key គឺជា attribute មួយ រឺច្រើនដែលអាចកំនត់អត្តសញ្ញាណចំពោះ entity នីមួយៗនៅក្នុង entity type។ ជាទូទៅនៅក្នុង entity type មួយអាចមាន candidate key លើសពី 1។

ឧទាហរណ៍: ចំពោះ Staff entity type ខាងលើ យើងអាចកំនត់យក attribute ធ្វើជា candidate key បានចំនួន 2 គឺ Staff_No attribute រឺ SSN attribute។

Primary Key The candidate key selected to be the primary key.

Primary key គឺជា candidate key 1 ក្នុងចំណោម candidate key ទាំងអស់ដែលយើងជ្រើសរើសជា primary key លើសពីនេះទៀតក្នុង entity type មួយមាន primary key តែមួយគត់។ ចំពោះ entity type ដែលមាន candidate key លើសពី 1 អ្នកកសាង database ត្រូវតែជ្រើសរើសមួយក្នុងចំណោមនោះមកធ្វើជា primary key។ មានលក្ខខណ្ឌមួយចំនួនក្នុងការជ្រើសរើស primary key:

- តំលៃលើ primary key attribute កំរនឹងប្រែប្រួល។
- តំលៃលើ primary key attribute ជាតំលៃត្រឹមត្រូវ (valid value) និងមិនផ្តុំតំលៃ Null (or unknown)។

➢ ជៀសវៀងការប្រើ **intelligent key** ដែលមានរចនាសម្ព័ន្ធបញ្ជាក់ពីចំណាត់ថ្នាក់ (classification), ទីកន្លែង (location)..... ឧទាហរណ៍: 2 តួអក្សរដំបូងនៃតំលៃ primary key សំរាប់បញ្ជាក់ពី warehouse location ។

➢ ពិចារណាទៅលើការជំនួស large composite key ដោយ primary key ដែលកើតឡើងតែ 1 attribute ។ ឧទាហរណ៍: Game_Number attribute នៃ Game entity type គួរតែអាចប្រើជំនួស composite key កើតពី Home_Team និង Visiting_Team ។

ឧទាហរណ៍: ចំពោះ Staff entity type ខាងលើ យើងអាចជ្រើសរើស candidate key មួយក្នុងចំណោមនោះ គឺ Staff_No attribute ធ្វើជា primary key និង candidate key SSN ដែលមិនបានជ្រើរើសជា primary key គេអោយឈ្មោះថា **alternate key** ។

Composite Key A candidate key that consists of two or more attributes.

Composite key គឺជា candidate key ដែលផ្សំឡើងដោយ attributes 2 រឺច្រើន ។

ឧទាហរណ៍: ចំពោះ Student_Course entity type ដែលមាន attributes ដូចជា Student_ID, Course_ID, Date_Start, Date_Complete, Grade ។ ដោយហេតុថា Student ម្នាក់អាចសិក្សា Course តែមួយដូចគ្នាក្នុងពេលខុសគ្នា តែ Course មួយអាចមាន Student ជាច្រើនសិក្សាក្នុងពេលតែមួយ ដូចនេះយើងអាចកំនត់បាន candidate key មួយចំពោះ entity type នេះដែលកើតឡើងដោយ 3 attributes គឺ Student_ID, Course_ID និង Date_Start ដូចនេះគេនិយាយថា Student_Course entity type មាន composite key មួយកើតពី 3 attributes ។

1. 3. Relationship Types (Relationship Sets)

Relationship is an association among the instances of one or more entity types that is of interest to the organization.

Relationship គឺជាការចងទំនាក់ទំនងរវាង instances នៃ entity type 1 រឺច្រើនដែលមានផលប្រយោជន៍ចំពោះ organization ។

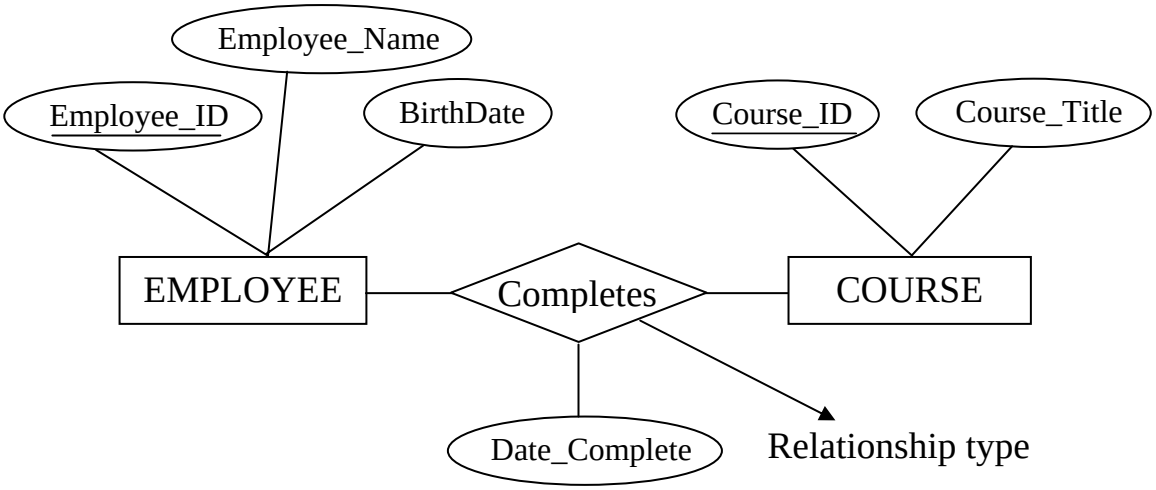
Relationship type (Relationship set) is a collection of relationship.

Relationship type គឺសំនុំនៃ relationship ។

Relationship instance or **Relationship occurrence** is a single occurrence of a relationship type.

Relationship instance រឺ **Relationship occurrence** គឺសំដៅលើការលេចឡើងរឺបង្ហាញតែម្តងក្នុង relationship type ។

ឧទាហរណ៍: ឧបមាថា យើងមាន entity types ចំនួន 2 គឺ Employee និង Course ហើយ Course មួយអាចមាន Employee ជាច្រើនសិក្សា និងបញ្ចប់វគ្គ ព្រមទាំង Employee ម្នាក់អាចសិក្សា និងបញ្ចប់ Course ជាច្រើន។ ដើម្បីតាមដាន Course ណាខ្លះដែល Employee បានបញ្ចប់វគ្គ យើងធ្វើការកំនត់ relationship រវាង Employee និង Course entity types។



(រូប 4.5)

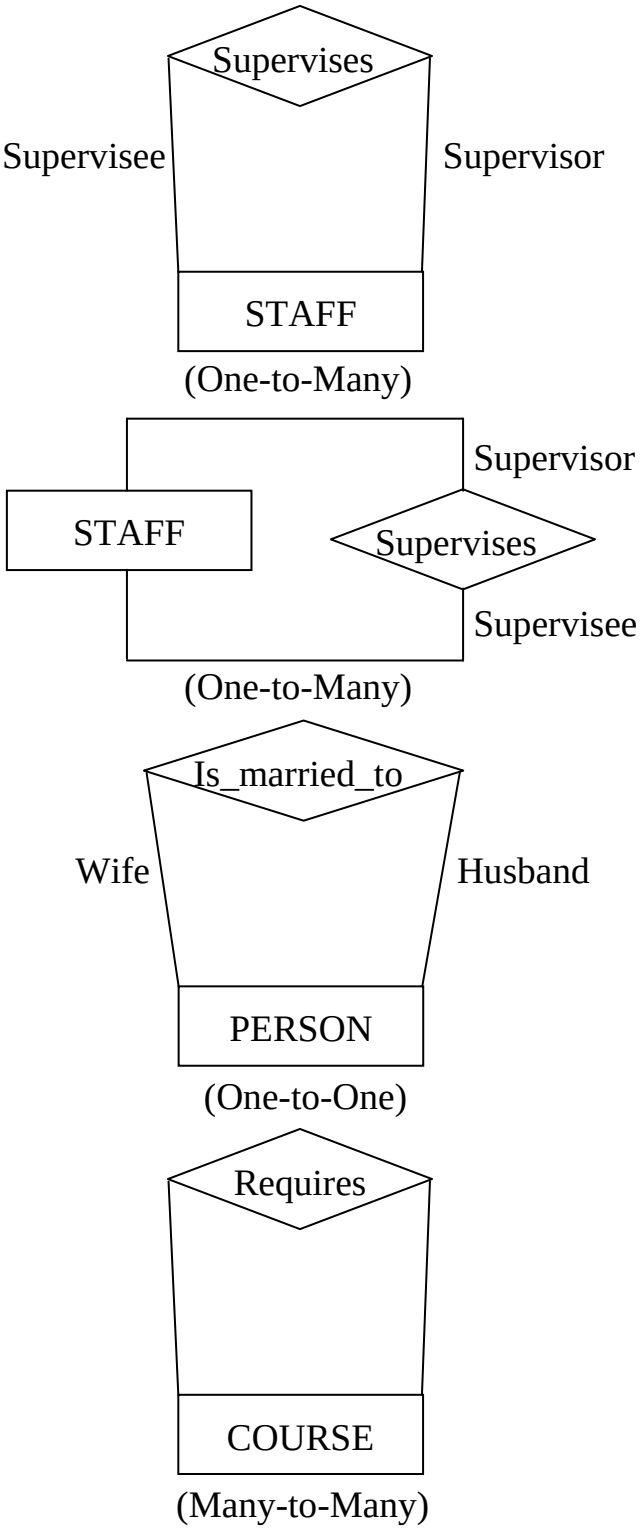
Degree of a Relationship is the number of entity types that participate in a relationship.

Degree of Relationship គឺជាចំនួន entity types ដែលចូលរួមក្នុង relationship នោះ។ Entity type ដែលចូលរួមក្នុង relationship នោះគេអោយឈ្មោះថា **participants**។ Degree of relationship ចែកជា 4 គឺ

Unary relationship A relationship where the *same* entity participates more than once in *different roles*.

Unary relationship គឺជា relationship ដែល entity ដូចគ្នាចូលរួមលើសពីម្តងដោយដើរតួនាទីផ្សេងគ្នា។ ការកំនត់ឈ្មោះតួនាទី (role name) មានសារៈសំខាន់ណាស់ក្នុង unary relationship (វីអាចហៅម្យ៉ាងទៀតថា **recursive relationship**) ដើម្បីកំនត់មុខងារ (function) របស់ entity ក្នុងការចូលរួមនោះ។

ឧទាហរណ៍: ចំពោះ Staff entity type ដែលមាន relationship កើតឡើងតែក្នុង entity ខ្លួនឯងមួយឈ្មោះថា Supervises ពេលនោះយើងត្រូវតែកំនត់មុខងារទៅអោយ entity ដែលចូលរួម។ មានមុខងារ 2 គឺ Supervisor និង Supervisee ដែលបង្ហាញថា Supervisor ធ្វើការគ្រប់គ្រងលើ Supervisee រី Supervisee ស្ថិតក្រោមការគ្រប់គ្រងរបស់ Supervisor។

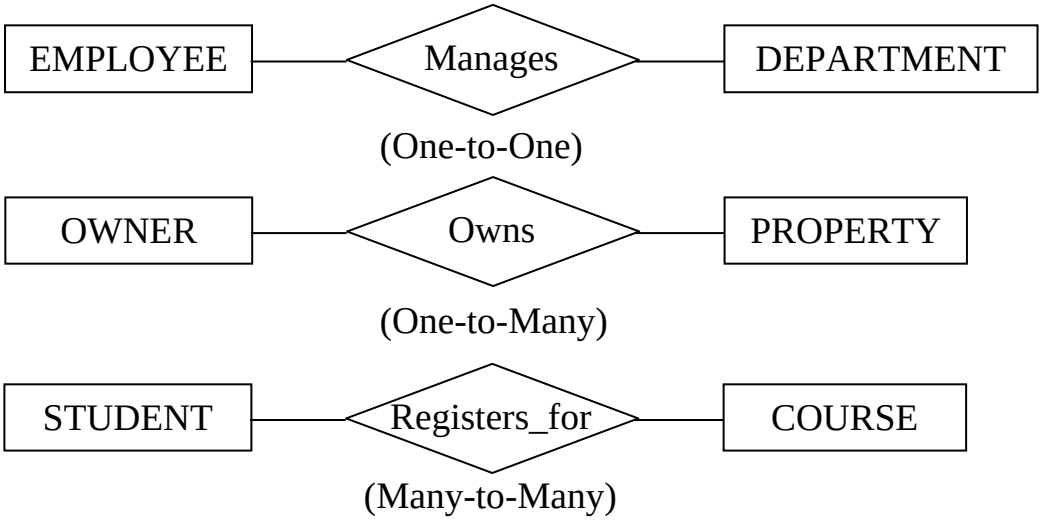


(រូប 4.6)

Binary relationship A relationship between the instances of two entities.

Binary relationship គឺជា relationship រវាង instances នៃ 2 entities។ Binary relationship គឺជា relationship ដែលជួបប្រទះច្រើនជាងគេក្នុងការកសាង database។

ឧទាហរណ៍: រូបខាងក្រោមបង្ហាញពី relationship ដែលកើតមានចំពោះ Student ធ្វើការចុះឈ្មោះ (register) លើ Course ។

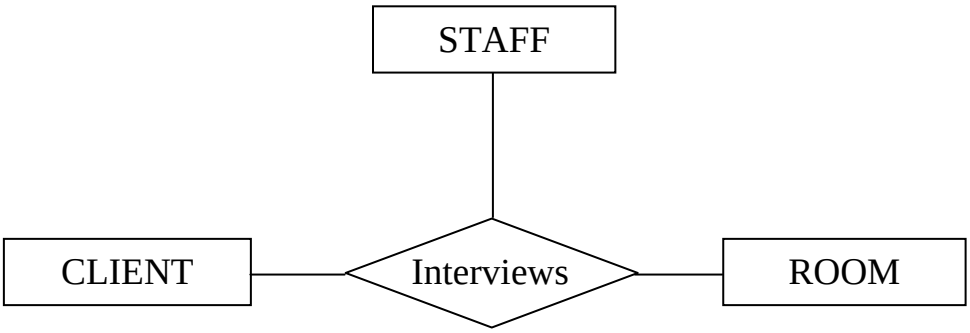


(រូប 4.7)

Ternary relationship A relationship between the instances of three entities.

Ternary relationship គឺជា relationship រវាង instances នៃ 3 entities ។

ឧទាហរណ៍: Relationship Interviews ចំពោះរូបខាងក្រោមបង្ហាញស្ថានការណ៍ដែល Staff មួយចំនួនទទួលបន្ទុកក្នុងការធ្វើសំភាសទៅលើ Client ទៅតាម Room ដែលបានកំនត់។

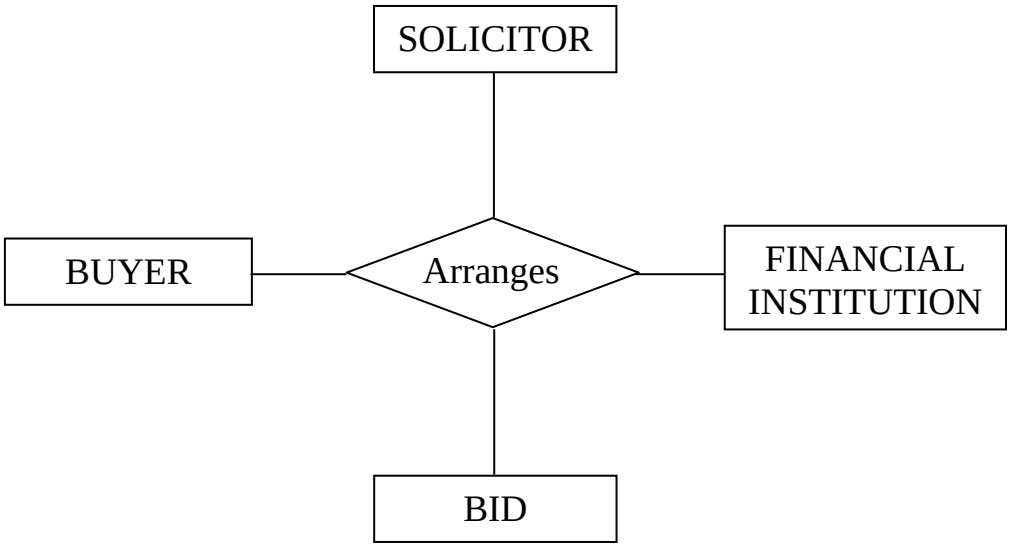


(រូប 4.8)

Quaternary relationship A relationship between the instances of four entities.

Quaternary relationship គឺជា relationship រវាង instances នៃ 4 entities ។

ឧទាហរណ៍: Relationship Arranges ចំពោះរូបខាងក្រោមបង្ហាញស្ថានការណ៍ដែលអ្នកទិញ (Buyer) ដែលទទួលការណែនាំពីស្នាក្តី (Solicitor) ក្រោមការជ្រោមជ្រែងពីស្ថាប័នហិរញ្ញវត្ថុ (Financial Institution) ដាក់តំលៃដេញថ្លៃ (Bid) ទៅលើ Property ។



(រូប 4.9)

1. 4. Attributes on Relationships

Attributes អាចធ្វើការភ្ជាប់ទៅកាន់ relationship ជាពិសេសចំពោះទំនាក់ទំនង Many-to-Many ដើម្បីបញ្ជាក់អត្ថន័យអោយកាន់តែច្បាស់ថែមទៀត។ ឧទាហរណ៍: ចំពោះ Date_Complete attribute ខាងលើភ្ជាប់ទៅកាន់ relationship Completes ដើម្បីបង្ហាញអំពីកាលបរិច្ឆេទដែល Employee បានបញ្ចប់ការសិក្សាទៅលើ Course។

2. Structural Constraints

ពេលនេះយើងក្រឡេកមើលទៅលើ constraint កំនត់ទៅលើ entities ដែលបានចូលរួមក្នុង relationship វិញម្តង។ Constraints គួរតែបង្ហាញពីការដាក់កំរិត (restrictions) ទៅលើ relationship ដូចទៅនឹងការងារជាក់ស្តែងដែលយើងតែងតែជួបប្រទះ។ ឧទាហរណ៍ទៅលើ constraint ដែលគិតថា រាល់ property ត្រូវតែមាន owner និងគ្រប់ branch office ដាច់ខាតត្រូវតែមានបុគ្គលិកយ៉ាងហោចណាស់ម្នាក់ធ្វើការដែរជាដើម។ ការដាក់កំរិតទៅលើ relationship ចែកជា 2 ប្រភេទគឺ Cardinality and Participation Constraints។

2. 1. Cardinality Constraints

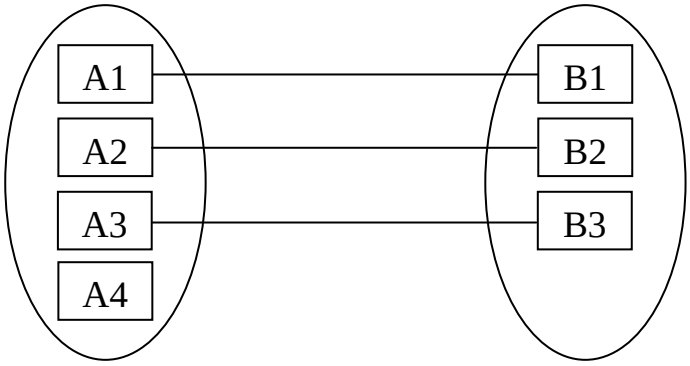
Cardinality ratio Describes the number of possible relationships for each participating entity.

Cardinality ratio បង្ហាញពីចំនួនទំនាក់ទំនងដែលអាចកើតមានចំពោះ entity ចូលរួមនីមួយៗ។ Cardinality ratio ចំពោះ relationship ចែកជា 3 ប្រភេទទៀតគឺ One-to-One (1:1), One-to-Many (1:M) និង Many-to-Many (M:N)។

Cardinality ratio រវាង entities គឺជាគោលការណ៍ដែលកំនត់នៅក្នុង organization ដើម្បីអោយសកម្មភាពការងារដំណើរបានត្រឹមត្រូវ ហើយគោលការណ៍នោះគេអោយឈ្មោះហៅថា **business rules**។ Business rules បង្ហាញជាផ្នែកដ៏សំខាន់មួយក្នុង modeling ហើយមិនមែនគ្រប់ business rules ទាំងអស់សុទ្ធតែអាចបង្ហាញក្នុង ER diagram ទេ។ ឧទាហរណ៍ទៅលើ business rules បែបនោះ គឺតំរូវការដែលសមាជិក staff ទទួលបានថ្ងៃឈប់សំរាកបន្ថែមទៀត (additional day's holiday) រៀងរាល់ឆ្នាំ។

One-to-One Relationship

សំដៅទៅលើ entity 1 ក្នុង entity type A មានទំនាក់ទំនងជាមួយ entity យ៉ាងច្រើនបំផុត 1 ក្នុង entity type B ហើយ entity 1 ក្នុង entity type B មានទំនាក់ទំនងជាមួយ entity យ៉ាងច្រើនបំផុត 1 ក្នុង entity type A វិញ។



(រូប 4.10)

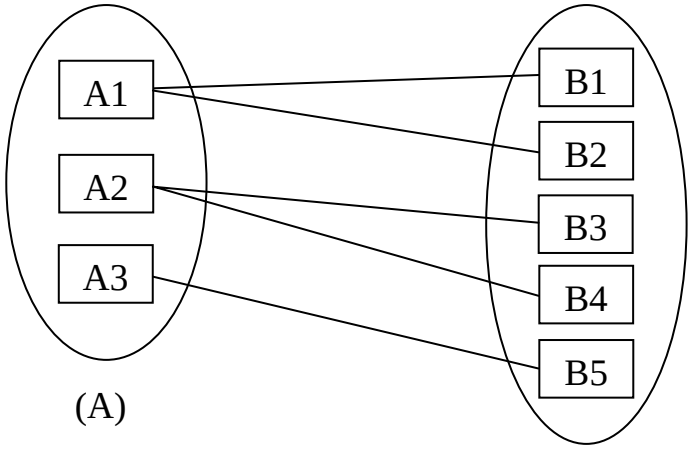
នៅក្នុងរូប 4.7 មាន binary relationship Manages ដែលតភ្ជាប់ទំនាក់ទំនងរវាង Employee entity និង Department entity។ ក្រុមហ៊ុនមានគោលការណ៍ថា Employee ម្នាក់អាចធ្វើការគ្រប់គ្រងបានតែមួយ Department ប៉ុណ្ណោះ ហើយ Department មួយមានតែ Manager គ្រប់គ្រងតែម្នាក់គត់។ យើងសង្កេតឃើញថា ទំនាក់ទំនងនេះជា ទំនាក់ទំនង One-to-One។

(One-to-One)

(រូប 4.11)

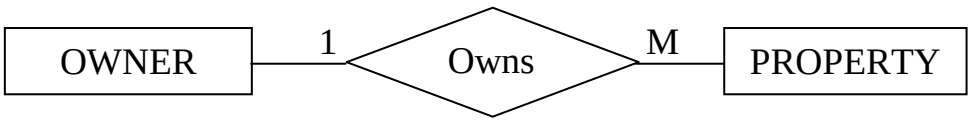
One-to-Many Relationship

សំដៅទៅលើ entity 1 ក្នុង entity type A មានទំនាក់ទំនងជាមួយ entity 1 រឺច្រើនក្នុង entity type B។ ទោះបីជាយ៉ាងណាក៏ដោយ entity 1 ក្នុង entity type B មានទំនាក់ទំនងជាមួយ entity យ៉ាងច្រើនបំផុត 1 ក្នុង entity type A វិញតែប៉ុណ្ណោះ។



(រូប 4.12)

នៅក្នុងរូប 4.7 មាន binary relationship Owns ដែលតភ្ជាប់ទំនាក់ទំនងរវាង Owner entity និង Property entity។ ក្រុមហ៊ុនមានគោលការណ៍ថា Owner ម្នាក់អាចមាន Property ជាច្រើនជាទ្រព្យសម្បត្តរបស់ខ្លួន ហើយ Property មួយមានតែ Owner តែម្នាក់គត់គ្រប់គ្រងតែប៉ុណ្ណោះ។ យើងសង្កេតឃើញថា ទំនាក់ទំនងនេះជាទំនាក់ទំនង One-to-Many។

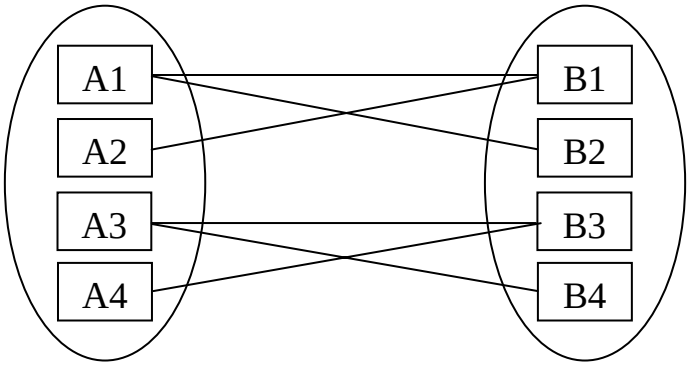


(One-to-Many)

(រូប 4.13)

Many-to-Many Relationship

សំដៅទៅលើ entity 1 ក្នុង entity type A មានទំនាក់ទំនងជាមួយ entity 1 រឺច្រើនក្នុង entity type B ហើយ entity 1 ក្នុង entity type B មានទំនាក់ទំនងជាមួយ entity 1 រឺច្រើនក្នុង entity type A វិញដែរ។



(A) (B)

(រូប 4.14)

នៅក្នុងរូប 4.7 មាន binary relationship Registers_for ដែលតភ្ជាប់ទំនាក់ទំនងរវាង Student entity និង Course entity។ សាលាមានគោលការណ៍ថា Student ម្នាក់អាចចុះឈ្មោះសិក្សាទៅលើ Course បានច្រើន ហើយ Course មួយមាន Student សិក្សាជាច្រើន។ យើងសង្កេតឃើញថា ទំនាក់ទំនងនេះជាទំនាក់ទំនង Many-to-Many។

(Many-to-Many)

(រូប 4.15)

2. 2. Participation Constraints

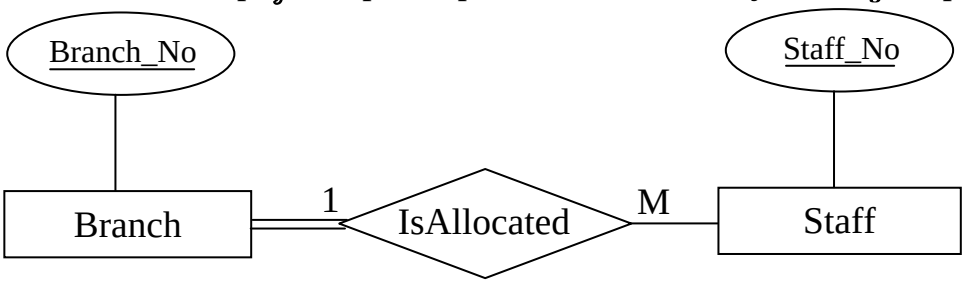
Participation Constraint Determines whether the existence of an entity depends upon it being related to another entity through the relationship.

Participation constraint កំនត់ថាតើការកើតមាន entity ណាមួយអាស្រ័យពឹងពាក់ទៅលើការទាក់ទងជាមួយ entity ផ្សេងទៀតក្នុង relationship។

Participation constraint ចែកជា 2 ប្រភេទគឺ **total** និង **partial**។ Participation មានលក្ខណៈ total ប្រសិនបើការកើតមាននៃ entity ណាមួយទាមទារការកើតមាននៃ entity ផ្សេងទៀតក្នុង relationship បើពុំដូច្នោះវាមានលក្ខណៈ partial។ Participation នៅក្នុង relationship ភ្ជាប់គ្នាដោយបន្ទាត់ (line) ដែលបន្ទាត់មួយ (single line) បង្ហាញពីលក្ខណៈ partial ហើយបន្ទាត់ពីរ (double line) បង្ហាញពីលក្ខណៈ total។

ពាក្យ total និង partial participation ជួនកាលអាចសំដៅជា **mandatory** និង **optional participation**។

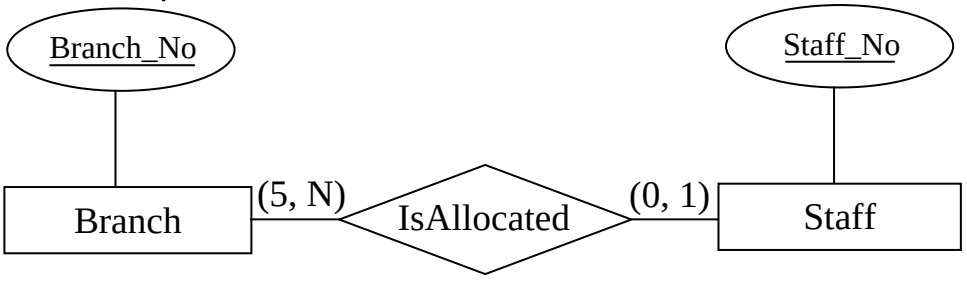
ឧទាហរណ៍: ចំពោះ Branch *IsAllocated* Staff relationship នៅក្នុងរូបខាងក្រោម រាល់ branch office តែងតែមានសមាជិក staff ជានិច្ច ពេលនោះ participation នៃ Branch entity មានលក្ខណៈ total។ ទោះបីជាយ៉ាងណាក៏ដោយ សមាជិកមួយចំនួននៃ staffs (ឧទាហរណ៍, Sales Personnel) អាចមិនស្ថិតនៅក្នុង branch office ណាមួយសោះនៅក្នុង *IsAllocated* relationship ដូចនេះ participation នៃ Staff entity មានលក្ខណៈ partial។



(រូប 4.16)

យើងអាចប្រើប្រាស់សញ្ញាជំនួស (alternative notation) សំរាប់បង្ហាញ structural constraint នៅក្នុង relationship ដោយធ្វើការបង្ហាញតំលៃតូចបំផុត (minimum value) និង តំលៃធំបំផុត (maximum value) ជាប់នឹងខ្សែបន្ទាប់តភ្ជាប់ (connecting line) សំរាប់បង្ហាញពី participation នៃ entity នៅក្នុង relationship។

ឧទាហរណ៍: យើងប្រើប្រាស់ notation ដើម្បីបង្ហាញពី structural constraints នៅក្នុង Branch *IsAllocated* Staff relationship ដូចរូបខាងក្រោមនេះ។ ផលប្រយោជន៍នៃការប្រើ ប្រាស់ notation អាចបង្ហាញពីព័ត៌មានបន្ថែមទៀតទៅលើ constraint នៅក្នុង relationship។ (5, N) notation រវាង Branch entity និង IsAllocated relationship បញ្ជាក់ថា យ៉ាងហោចណាស់ staffs 5 នាក់ស្ថិតក្នុង branch office នីមួយៗ (Min = 5) ហើយក្នុង branch office នីមួយៗមាន staffs មិនកំនត់ (Max = N)។ ដូចគ្នានេះដែរ (0, 1) notation រវាង Staff entity និង IsAllocated relationship បញ្ជាក់ថា staff ម្នាក់អាចមិនស្ថិតក្នុង branch office ណាមួយ ទាល់តែសោះ (Min = 0) និង staff ម្នាក់អាចស្ថិតក្នុង branch office យ៉ាងច្រើនបំផុត 1 (Max = 1)។



(រូប 4.17)

3. The Enhanced Entity-Relationship Model

Entity-Relationship Model (ER Model) មានលក្ខណៈគ្រប់គ្រាន់សំរាប់បង្ហាញ database schema ទូទៅភាគច្រើនចំពោះ traditional និង administrative-based database ប៉ុន្តែចាប់តាំងពីឆ្នាំ 1980 មកមានការរីកចំរើនយ៉ាងខ្លាំងក្នុងការអភិវឌ្ឍន៍ database ថ្មីៗដែលមានលក្ខណៈស្មុគស្មាញសំុញុំដែល ER Model ពុំអាចបំពេញបាន ទើបបណ្តាលអោយគេធ្វើការកែសំរួលដោយបន្ថែមលក្ខណៈមួយចំនួនទៅលើ ER model ដើម្បីធ្វើអោយវាអាចបង្ហាញទិន្នន័យស្មុគស្មាញក្នុងពេលបច្ចុប្បន្ននេះអោយបានត្រឹមត្រូវ ដោយដាក់ឈ្មោះថា Enhanced Entity-Relationship Model (EER Model)។

EER Model The model that has resulted from extending the original ER model with new modeling constructs.

EER model គឺជា model ដែលកើតចេញពីការបន្ថែមលក្ខណៈមួយចំនួនទៅលើ ER model ដែលមានស្រាប់។

Modeling construct ថ្មីហើយមានសារៈសំខាន់បំផុតដែលបានបញ្ចូលនៅក្នុង EER model គឺ Supertype/Subtype relationship។

3. 1. Supertypes (Superclasses) and Subtypes (Subclasses)

Subtype A subgrouping of the entities in an entity type which has attributes that are distinct from those in other subgroupings.

Subtype គឺជាបណ្តុំរង entity នៅក្នុង entity type ដែលមាន Attributes ខុសពីបណ្តុំរងផ្សេងទៀត។ ជាទូទៅនៅក្នុង EER model, subtype តែងតែមានតួនាទី (distinct role) ច្បាស់លាស់ខុសពី subtype ដ៏ទៃទៀត។

Supertype An generic entity type that has a relationship with one or more subtypes.

Supertype គឺជា entity type ទូទៅដែលមានទំនាក់ទំនងជាមួយ subtypes មួយរឺច្រើន។

នៅក្នុងករណីមួយចំនួន entity type មួយប្រហែលជាអាចមាន subtypes ជាច្រើនដែលមានតួនាទីខុសៗគ្នា។ ឧទាហរណ៍: Entities ដែលជាសមាជិកនៃ Staff entity type អាចចាត់ជាក្រុមទៅតាមតួនាទីជា Manager, Secretary និង Sales Personnel ដែលក្នុងនោះគេចាត់ទុក Staff entity set ជា supertype នៃ subtypes: Manager, Secretary និង Sales Personnel។ Relationship ដែលកើតឡើងរវាង supertype មួយនិង

subtype ណាមួយក្នុងចំណោម subtype ទាំងអស់ គេហៅថា Supertype/Subtype relationship (Superclass/Subclass relationship)។ ឧទាហរណ៍: Staff/Manager គឺជា supertype/subtype relationship។

រាល់សមាជិកនៃ subtype ក៏គឺជាសមាជិកនៃ supertype ផងដែរ។ លើសពីនេះទៅទៀត សមាជិកនៃ subtype គឺដូចទៅនឹង entity នៅក្នុង supertype ប៉ុន្តែមានតួនាទីខុសគ្នា (Each member of a subtype is also a member of the supertype. In othe words, the subclass member is the same as the entity in the supertype, but has a distinct role.)។ ប៉ុន្តែមានសមាជិកមួយចំនួននៃ supertype អាចមិនមែនជាសមាជិកនៃ subtype ណាមួយទាល់តែសោះ។ Relationship រវាង supertype និង subtype គឺ One-to-One (1:1) relationship។

មានហេតុផលសំខាន់ 2 ដែលនាំអោយមានការនែនាំអោយស្គាល់នូវ concepts នៃ supertypes and subtypes គឺ

- ហេតុផលទី 1: ជៀសវាងការពណ៌នា concepts ស្រដៀងគ្នាច្រើនលើកច្រើនសារ ដោយហេតុនេះហើយ ធ្វើអោយកាត់បន្ថយពេលវេលាក្នុងការកសាង និងធ្វើអោយយើងមើល ER diagram កាន់តែឆាប់យល់ (The first reason is that it avoids describing similar concepts more than once, thereby saving time for the designer and making the ER diagram more readable.)។

- ហេតុផលទី 2: បន្ថែមព័ត៌មានថែមទៀតទៅកាន់ការកសាងក្នុងទំរង់មួយដែលមនុស្សភាគច្រើនអាចជឿង ឺស្គាល់យ៉ាងច្បាស់ (The second reason is that it adds more semantic information to the design in a form that is familiar to many people.)។

3. 2. Attribute Inheritance

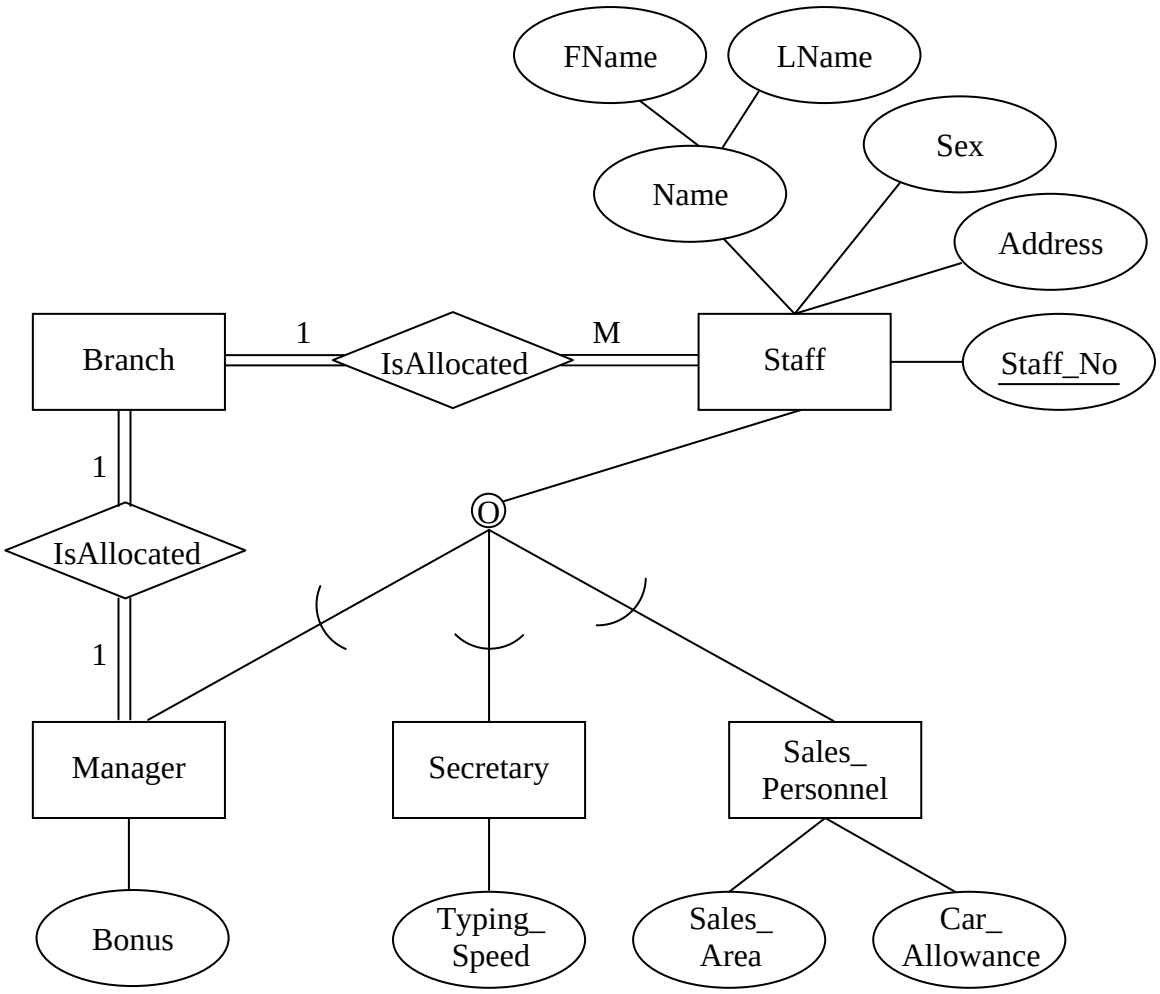
Subtype entity inherits values of all attributes of the supertype.

Subtype entities ធ្វើការទទួលយករាល់ attributes ទាំងអស់នៃ Supertype។

ឧទាហរណ៍: Sales_Personnel subtype ទទួលយករាល់ attributes ទាំងអស់នៃ Staff supertype ដូចជា Staff_No, Name, Address និង DOB ព្រមទាំង attributes ផ្តាល់របស់វាដូចជា Car_Allowance និង Sales_Area។

Subtype គឺជា entity ដោយហេតុដូចនេះវាប្រហែលជាអាចមាន subtypes ជារបស់វាផងដែរ។ An entity, and its subtypes, and their subtypes, and so on, is called a **type hierarchy**។ ឈ្មោះដែលគេកំនត់ទៅលើ type hierarchy មានដូចជា:

specialization hierarchy (ឧទាហរណ៍: Manager is a specialization of staff), **generalization hierarchy** (ឧទាហរណ៍: Staff is a generalization of Manager) និង **IS-A hierarchy** (ឧទាហរណ៍: Manager IS-A (member of) Staff)។



(រូប 4.18)

3. 3. Specialization

Specialization The process of maximizing the differences between members of an entity by identifying their distinguishing characteristics.

Specialization គឺជាប្រតិបត្តិការក្នុងពង្រីកភាពខុសគ្នារវាងសមាជិកនៃ entity ដោយធ្វើការកំនត់លក្ខណៈខុសប្លែកគ្នារបស់ពួកវា។ Specialization ជាប្រតិបត្តិការពីលើចុះមកក្រោម (top-down process) ក្នុងការកំនត់សំនុំនៃ supertypes និង subtypes របស់ពួកគេ ហើយការកំនត់សំនុំនៃ subtypes ដោយឈរទៅលើមូលដ្ឋាននៃលក្ខណៈខុសប្លែកគ្នា (distinguishing characteristics) នៃ entities នៅក្នុង supertype។ បន្ទាប់ពីយើង

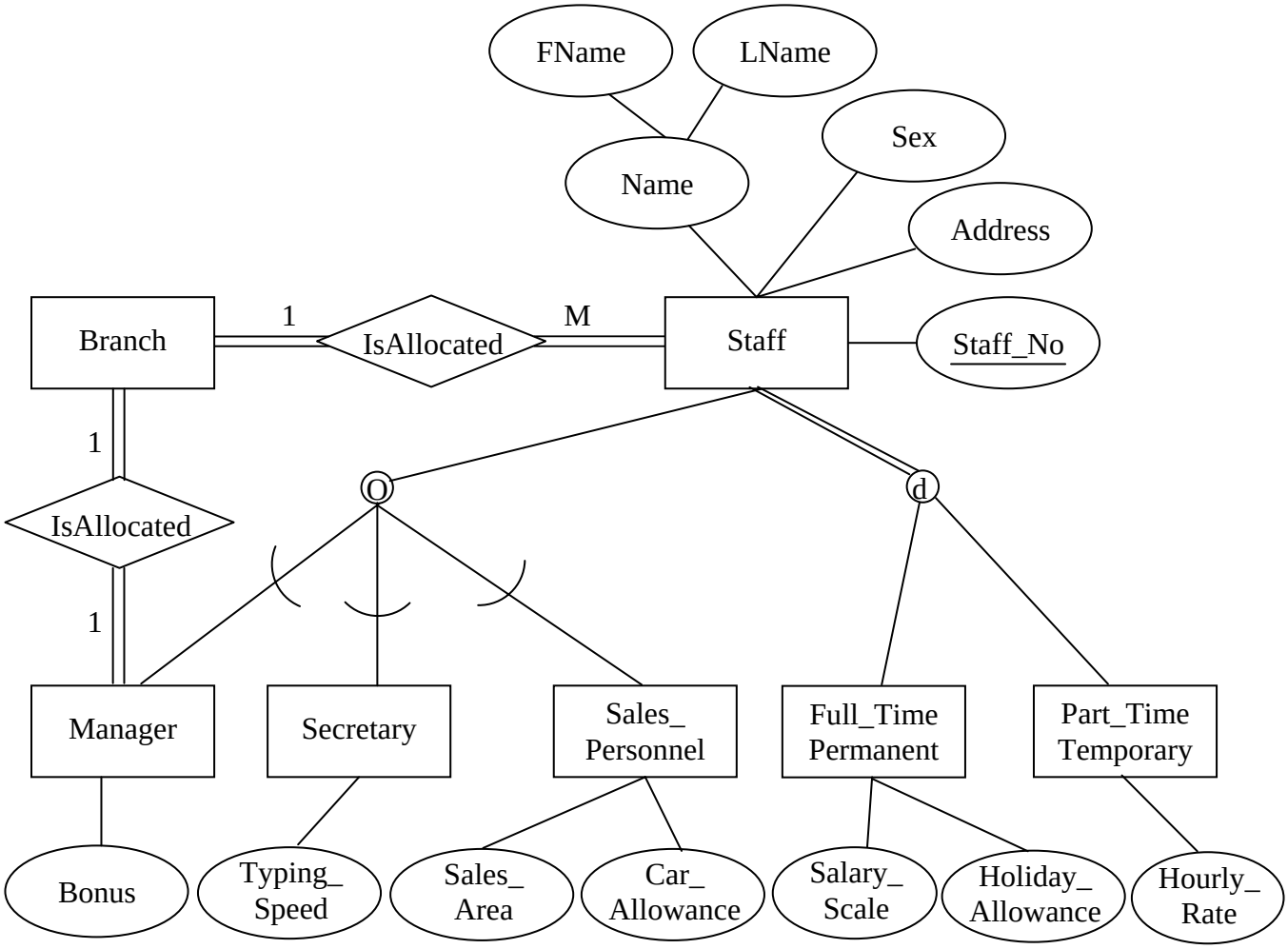
កំនត់ចានសំនុំ subtypes នៃ entity type រួចមក យើងធ្វើការភ្ជាប់ specific attributes ទៅកាន់ subtype នីមួយៗ (នៅពេលចាំបាច់) និងកំនត់ relationship ផងដែររវាងរាល់ subtypes នីមួយៗ និង entity types រឺ subtypes ដ៏ទៃទៀត (នៅពេលចាំបាច់)។

ឧទាហរណ៍: ធ្វើការត្រួតពិនិត្យទៅលើ specialization ដែលកំនត់សំនុំ subtypes រួមមាន Manager, Secretary និង Sales Personnel ចេញពី Staff supertype។ យើងប្រើប្រាស់ EER diagram ដើម្បីបង្ហាញជារូបសំគាល់ specialization ដូចរូប 4.18។ យើងកត់សំគាល់ឃើញថា Staff supertype និង subtypes ផ្សេងទៀតបង្ហាញជារាងចតុកោណកែង (rectangle) ហើយ subclasses នៃ specialization ភ្ជាប់ទៅកាន់រង្វង់មូល (circle) តាមរយះបន្ទាត់មួយ (single line) ហើយបន្ទាត់នោះក៏ភ្ជាប់ទៅកាន់ supertype តទៅទៀត។ Subset symbol or U-Shaped symbol ($\subset$) ស្ថិតនៅលើបន្ទាត់នោះសំរាប់បញ្ជាក់ពីទិសដៅនៃ supertype/subtype relationship (ឧទាហរណ៍: Manager ($\subset$) Staff)។ សញ្ញា 'O' នៅក្នុង specialization circle បង្ហាញពី constraints លើ supertype/subtype relationship។

Specific attributes ភ្ជាប់ទៅកាន់ subtypes របស់គេ ឧទាហរណ៍: ដូចជា Car_Allowance និង Sales_Area ក្នុងរូប 4.18 ដែលភ្ជាប់ទៅកាន់ Sales_Personnel subtype តែមួយគត់។

យើងប្រហែលជាអាចមាន specializations មួយចំនួនទៅលើ entity តែមួយដោយឈរមូលដ្ឋានទៅលើលក្ខណៈខុសប្លែកគ្នា។ ឧទាហរណ៍: ចំពោះរូបខាងក្រោម (រូប 4.19) បង្ហាញពី specialization មួយផ្សេងទៀត ដែលបង្កើត subtypes 2 ទៀតគឺ Full_Time_Permanent និង Part_Time_Temporary ដោយបែងចែកលក្ខណៈខុសគ្នាលើកិច្ចសន្យាការជួលរបស់បុគ្គលិក (employment contract)។

តាមរយះរូបនេះ យើងក៏បង្ហាញពី specific attributes នៃ Full_Time_Permanent (Salary_Scale និង Holiday_Allowance) និង Part_Time_Temporary (Hourly_Rate) subtypes។ សញ្ញា 'd' ក្នុង specialization circle បង្ហាញពី constraints លើ supertype/subtype relationship។



(រូប 4.19)

3. 4. Generalization

Generalization The process of minimizing the differences between entities by identifying their common features.

Generalization គឺជាប្រតិបត្តិការក្នុងបង្រួមភាពខុសគ្នារវាងសមាជិកនៃ entity ដោយធ្វើការកំនត់លក្ខណៈទូទៅរបស់ពួកវា។ Specialization ជាប្រតិបត្តិការពីក្រោមឡើងលើ (bottom-up process) ដោយធ្វើការកំនត់ generalized supertype ចេញពី original subtypes។ ប្រតិបត្តិការនៃ generalization មានលក្ខណៈផ្ទុយពី specialization។

ឧទាហរណ៍: ធ្វើការត្រួតពិនិត្យទៅលើ Manager, Secretary និង Sales Personnel ដែលបង្ហាញជា entities ដាច់ដោយឡែកពីគ្នា។ ប្រសិនបើយើងធ្វើប្រតិបត្តិការ generalization ទៅលើ entities ទាំងនោះ គឺយើងចង់កំនត់ភាពដូចគ្នារវាងពួកវាដូចជា common attributes និង relationships។ ដូចបានរៀបរាប់ពីខាងលើ entities ទាំងនេះប្រើប្រាស់ common

attributes ចំពោះ staffs ទាំងអស់ ហើយហេតុដូច្នេះយើងនឹងកំនត់ Manager, Secretary និង Sales_Personnel ជា subtypes នៃ supertype Staff ដូចរូប 4.18។

3. 5. Constraints on Specialization and Generalization

នៅក្នុងចំនុចនេះ យើងនឹងពណ៌នាអំពី constraints ដែលកំនត់ទៅលើ specialization រឺ generalization។

Constraint ទី 1 ហៅថា **Disjoint constraint**។ Constraint នេះកំនត់ថា ប្រសិនបើ subtypes នៃ specialization មានលក្ខណៈ disjoint នោះ entity អាចជាសមាជិកនៃ subtype មួយក្នុងចំណោម subtypes ទាំងអស់នៃ specialization (This constraint specifies that if the subtypes of a specialization are disjoint, then an entity can be a member of only one of the subtypes of the specialization.)។ សញ្ញាតំណាងអោយ disjoint constraint តាងដោយតួអក្សរ ‘d’ (មានន័យថា disjoint) ដាក់ក្នុងរង្វង់មូល (circle) ដែលភ្ជាប់ subtypes ទៅកាន់ supertype។ Subtypes ចំពោះ specialization កិច្ចសន្យាក្នុងការជួល (Full_Time_Permanent, Part_Time_Temporary) ក្នុងរូប 4.19 បង្ហាញពី disjoint constraint មានន័យថា កិច្ចសន្យាក្នុងការជួលចំពោះសមាជិកនៃ staff បើមិនមានលក្ខណៈ full-time permanent គឺមានលក្ខណៈ part-time temporary គឺមិនអាចមានលក្ខណៈទាំង 2 ក្នុងពេលតែមួយបានទេ។

ប្រសិនបើ subtypes នៃ specialization គ្មានលក្ខណៈ disjoint នោះមានន័យថា entity អាចជាសមាជិកនៃ subtype ក្នុង specialization លើសពី 1 (If subtypes of a specialization are not disjoint, then an entity may be a member of more than one subtype of a specialization.)។ ដើម្បីបង្ហាញពី **nondisjoint constraint** គេប្រើតួអក្សរ ‘o’ (មានន័យថា overlapping) ដាក់ក្នុងរង្វង់មូល (circle) ដែលភ្ជាប់ subtypes ទៅកាន់ supertype។ Subtypes ចំពោះ specialization តួនាទីការងារ (Manager, Secretary, Sales_Personnel) ក្នុងរូប 4.19 បង្ហាញពី nondisjoint constraint មានន័យថា staff មួយអាចជាសមាជិកនៃ Manager និង Sales_Personnel subtypes ទាំង 2 តែម្តង។

Constraint ទី 2 លើ specialization ហៅថា **participation constraint** ដែលអាចជា total រឺ partial។ Specialization ដែលមាន total participation កំនត់ថា រាល់ entity នីមួយៗនៅក្នុង supertype ត្រូវតែជាសមាជិកនៃ subtype ក្នុង specialization (A specialization with a total participation specifies that every entity in the

supertype must be a member of a subtype in the specialization.)។ ដើម្បីបង្ហាញពី **total participation** គេប្រើខ្សែបន្ទាត់ពីរ (double line) គូរភ្ជាប់ពី supertype ទៅកាន់ specialization circle។ នៅក្នុងរូប 4.19 កិច្ចសន្យាក្នុងការជួលមានលក្ខណៈ total participation ដែលមានន័យថា រាល់សមាជិកទាំអស់នៃ staff សុទ្ធតែ part-time រឺ full-time។

Specialization ចំពោះ **partial participation** កំនត់ថា entity មួយមិនចាំបាច់ជាកម្មសិទ្ធ រឺជាសមាជិកនៃ subtypes ណាមួយទាល់តែសោះក្នុងចំណោម subtypes ទាំងអស់ទេ (A specialization with partial participation specifies that an entity need not belong to any of the subtypes of a specialization.)។ Partial participation បង្ហាញដោយខ្សែបន្ទាត់មួយ (single line) គូរភ្ជាប់ពី supertype ទៅកាន់ specialization circle។ នៅក្នុងរូប 4.19 specialization តួនាទីការងារមានលក្ខណៈ partial participation ដែលមានន័យថា សមាជិកនៃ staff មិនចាំបាច់មានតួនាទីការងារបន្ថែមទៀតទេដូចជា Manager, Secretary or Sales_Personnel។

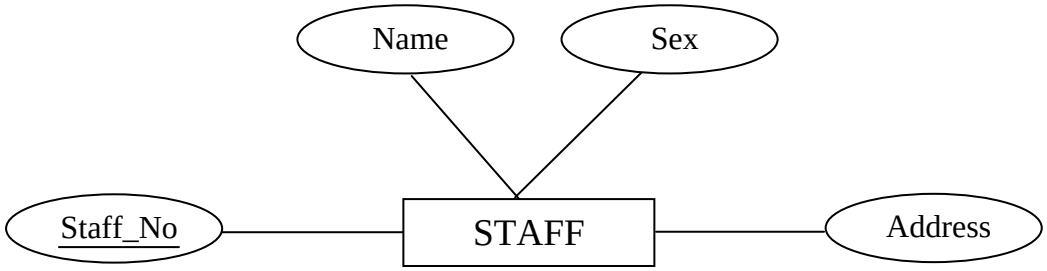
ការចូលរួមគ្នារវាង disjoint និង participation constraints បានបង្កើតជា 4 ប្រភេទគឺ disjoint and total, disjoint and partial, overlapping and total និង overlapping and partial។

4. Transforming EER Diagrams into Relations

ក្រោយពីបង្កើត EER diagram រួច យើងត្រូវបំលែងវាទៅជា relations (tables) ដោយប្រើប្រាស់សំនុំនៃច្បាប់មួយចំនួនសំរាប់កំនត់ដូចតទៅ:

Step 1: Map Regular Entities

រាល់ entity type ធម្មតាទាំងអស់នៅក្នុង ER diagram ត្រូវបំលែងទៅជា relation (table) ហើយឈ្មោះ relation ដូចនឹងឈ្មោះ entity type។ រាល់ simple attribute ទាំងអស់នៃ entity type នឹងបំលែងទៅជា attribute នៃ relation លើសពីនេះទៀត key រឺ identifier នៃ entity type នឹងបំលែងទៅជា primary key នៃ relation។

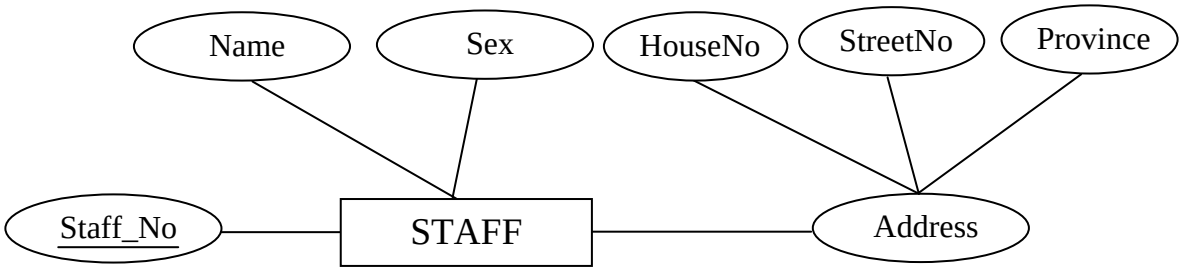


(រូប 4.20a)

Staff (Staff_No, Name, Sex, Address)

(រូប 4.20b)

Composite Attributes នៅពេលធ្វើការបំលែងចំពោះ entity ធម្មតាដែលមាន composite attribute យើងគួរតែបញ្ចូល simple attributes ទាំងអស់ដែលជាបំណែកនៃ composite attribute ទៅកាន់ relation វិរក្សាតែ composite attribute នោះ ប៉ុន្តែវិធីដែលគេប្រើប្រាស់ញឹកញាប់ជាងគេគឺបញ្ចូល simple attributes ទាំងអស់នៃ composite attribute ទៅកាន់ relation ។

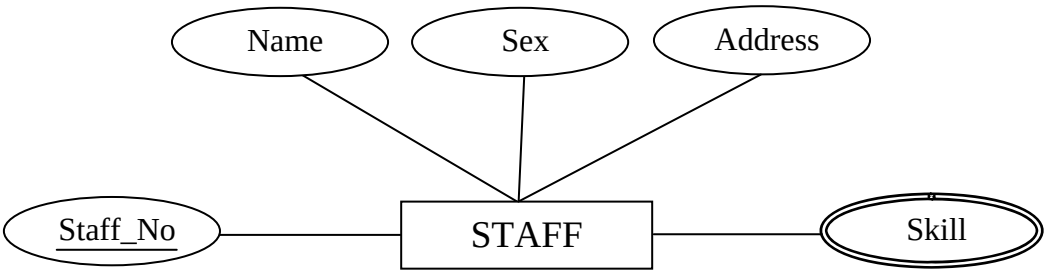


(រូប 4.21a)

Staff (Staff_No, Name, Sex, HouseNo, StreetNo, Province)

(រូប 4.21b)

Multivalued Attributes នៅពេលធ្វើការបំលែងចំពោះ entity ធម្មតាដែលមាន multivalued attribute យើងត្រូវតែបំលែងវាទៅជា 2 relations ដោយ relation ទី 1 ផ្ទុកទូវ simple attributes ទាំងអស់នៃ entity type លើកលែងតែ multivalued attribute ហើយ relation ទី 2 ផ្ទុក attributes ចំនួន 2 ដែល attribute 1 ក្នុងចំណោមនោះគឺជា primary key នៃ relation ទី 1 និង attribute 1 ទៀតគឺជា multivalued attribute លើសពីនេះទៀត primary key នៃ relation ទី 2 ជ្រុំឡើងដោយ attributes ទាំង 2 នេះ។



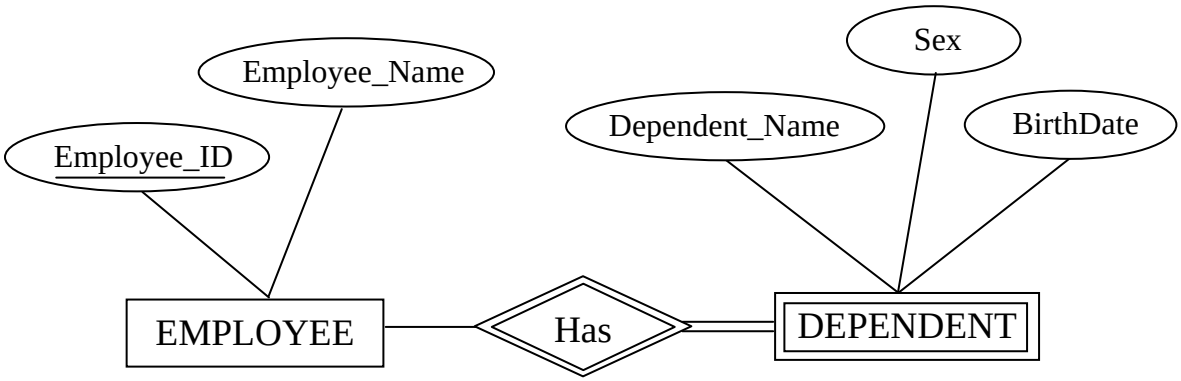
(រូប 4.22a)

Staff (Staff_No, Name, Sex Address)

Staff_Skill (Staff_No, Skill) (រូប 4.22b)

Step 2: Map Weak Entities

គេធ្វើការបំលែង weak entity ដោយធ្វើការបង្កើត relation ថ្មីមួយផ្ទុក simple attributes ទាំងអស់នៃ weak entity ព្រមទាំង primary key នៃ identifying owner ដែល attribute នោះនឹងក្លាយទៅជា foreign key។ Primary key នៃ relation ថ្មីនេះផ្សំដោយ primary key នៃ identifying owner រួមជាមួយ partial key នៃ weak entity។



(រូប 4.23a)

Employee (Employee_ID, Employee_Name)
Dependent (Employee_ID, Dependent_Name, Sex, BirthDate)

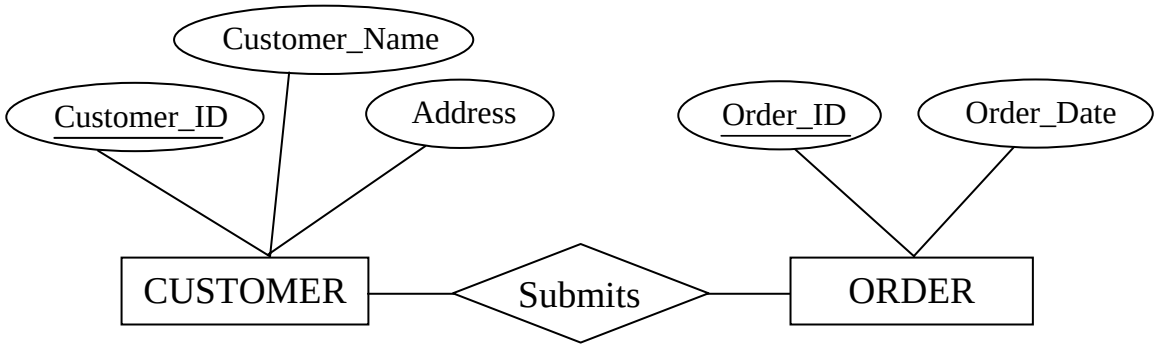
(រូប 4.23b)

Step 3: Map Binary Relationships

Cardinality ratio ចំពោះ binary relationship ចែកចេញជា 3 ប្រភេទទៀតគឺ One-to-One, One-to-Many និង Many-to-Many។

Map Binary One-to-Many Relationships

គេធ្វើការបំលែង binary 1:M relationship ដោយបង្កើត relation ផ្សេងៗគ្នាសំរាប់ entity types ទាំង 2 ដែលបានចូលរួមនៅក្នុង relationship។ បន្ទាប់មកបន្ថែម primary key នៃ one-side entity ទៅកាន់ many-side entity ដើម្បីអោយវាក្លាយទៅជា foreign key ទើប relations ទាំង 2 អាចធ្វើការទាក់ទងគ្នាបាន។



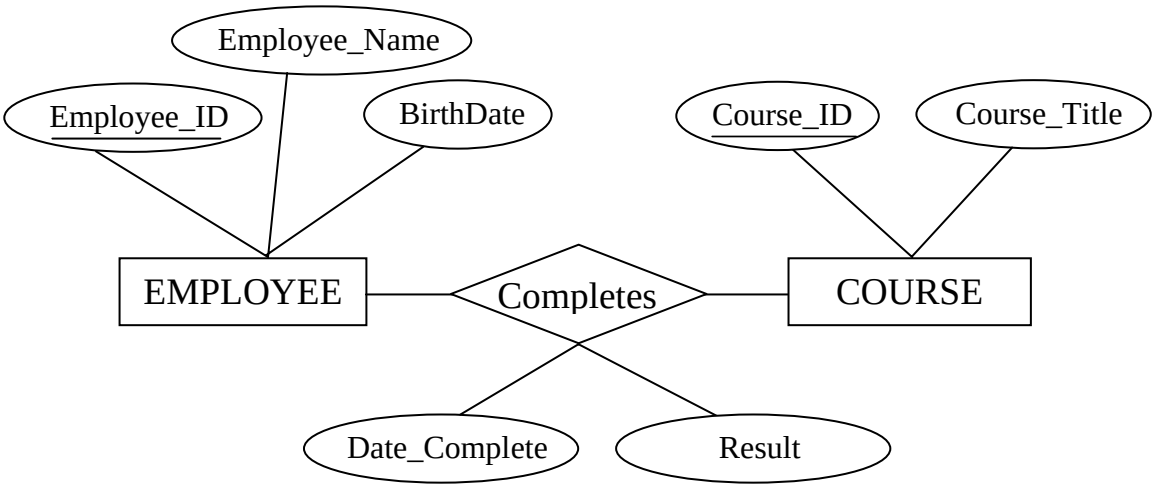
(រូប 4.24a)

Customer (Customer_ID, Customer_Name, Address)
Order (Order_ID, Order_Date, Customer_ID)

(រូប 4.24b)

Map Binary Many-to-Many Relationships

គេធ្វើការបំលែង binary M:N relationship ដោយបង្កើត relation ផ្សេងៗគ្នាសំរាប់ entity types ទាំង 2 ដែលបានចូលរួមនៅក្នុង relationship និងបង្កើត relation ថ្មីមួយផ្សេងទៀតដោយផ្ទុក primary key នៃ relations ទាំង 2 ហើយ attributes ទាំង 2 នោះ ភាគច្រើន ច្រើនតែចូលរួមជ្បំគ្នាធ្វើការជា primary key នៃ relation ថ្មីនោះ។



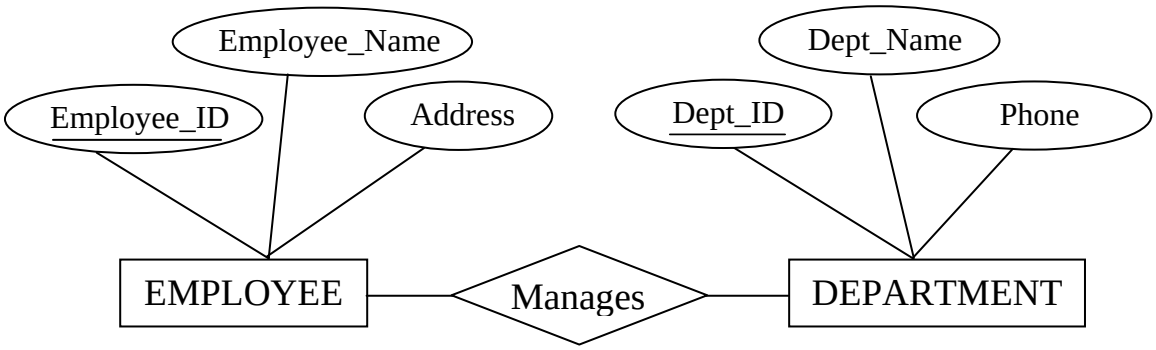
(រូប 4.25a)

Employee (Employee_ID, Employee_Name, BirthDate)
Course (Course_ID, Course_Title)
Completes (Employee_ID, Course_ID, Date_Complete, Result)

(រូប 4.25b)

Map Binary One-to-One Relationships

គេធ្វើការបំលែង binary one-to-one relationship ដោយអនុវត្តជា 2 ជំហាន។ ជំហានទី 1 ត្រូវធ្វើការបង្កើត relation នីមួយៗសំរាប់ entity types ទាំង 2និងជំហានទី 2 ទាញ primary key នៃ relation ណាមួយទៅដាក់ក្នុង relation មួយទៀតដើម្បីក្លាយទៅជា foreign key ។



(រូប 4.26a)

Employee (Employee_ID, Employee_Name, Address)
Department (Dept_ID, Dept_Name, Phone, Mgr_ID)

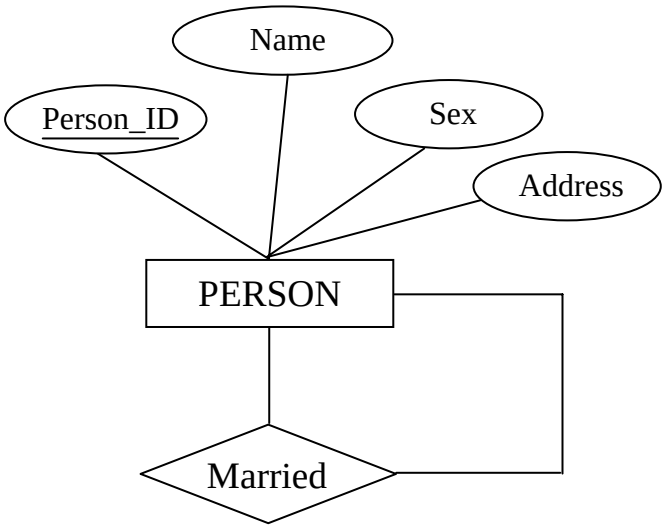
(រូប 4.26b)

Step 4: Map Unary Relationships

Cardinality ratio ចំពោះ unary relationship ចែកចេញជា 3 ប្រភេទទៀតគឺ One-to-One, One-to-Many និង Many-to-Many ។

Map Unary One-to-One and One-to-Many Relationships

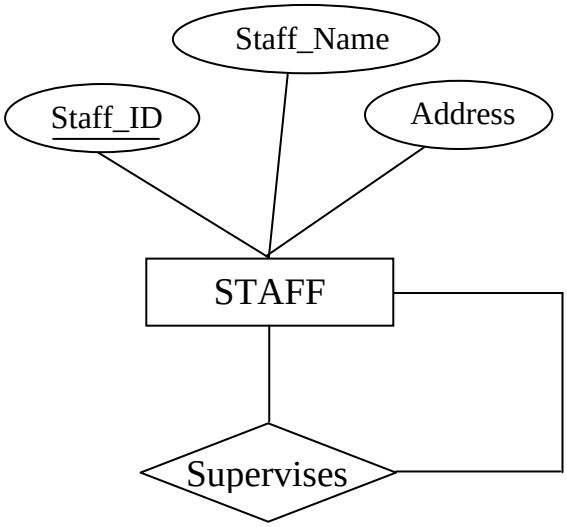
គេធ្វើការបំលែង unary 1:M relationship ដោយបង្កើត relation មួយដោយបញ្ចូល simple attributes ទាំងអស់ បន្ទាប់មកបន្ថែម foreign key ទៅកាន់ relation ដដែលនោះ ដែលចង្អុលទិទាញតំលៃ (references) ទៅកាន់ primary key នៅក្នុង relation តែមួយ។



(រូប 4.27a) For One-to-One relationship

Person (Person_ID, Name, Sex, Address, Partner_ID)

(រូប 4.27b)



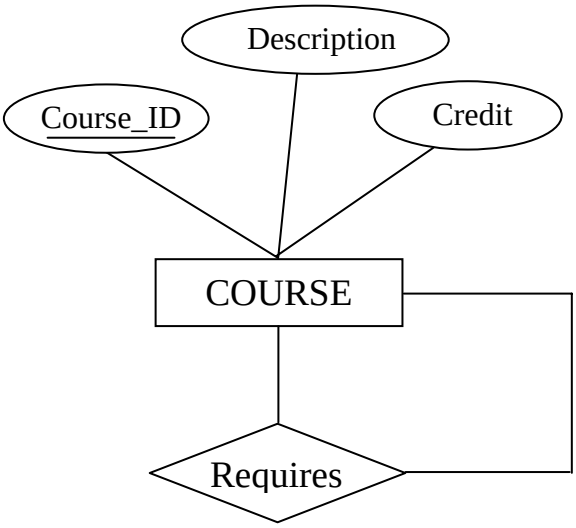
(រូប 4.28a) For One-to-Many relationship

Staff (Staff_ID, Staff_Name, Address, Supervisor_ID)

(រូប 4.28b)

Map Unary Many-to-Many Relationships

គេធ្វើការបំលែង unary M:N relationship ដោយបង្កើត relation ថ្មីចំនួន 2 ដោយ relation ទី 1បង្ហាញពី entity type នៅក្នុង relationship ហើយ relation មួយផ្សេងទៀត បង្ហាញពីទំនាក់ទំនង M:N ជាមួយខ្លួនឯងផ្ទាល់។



(រូប 4.29a)

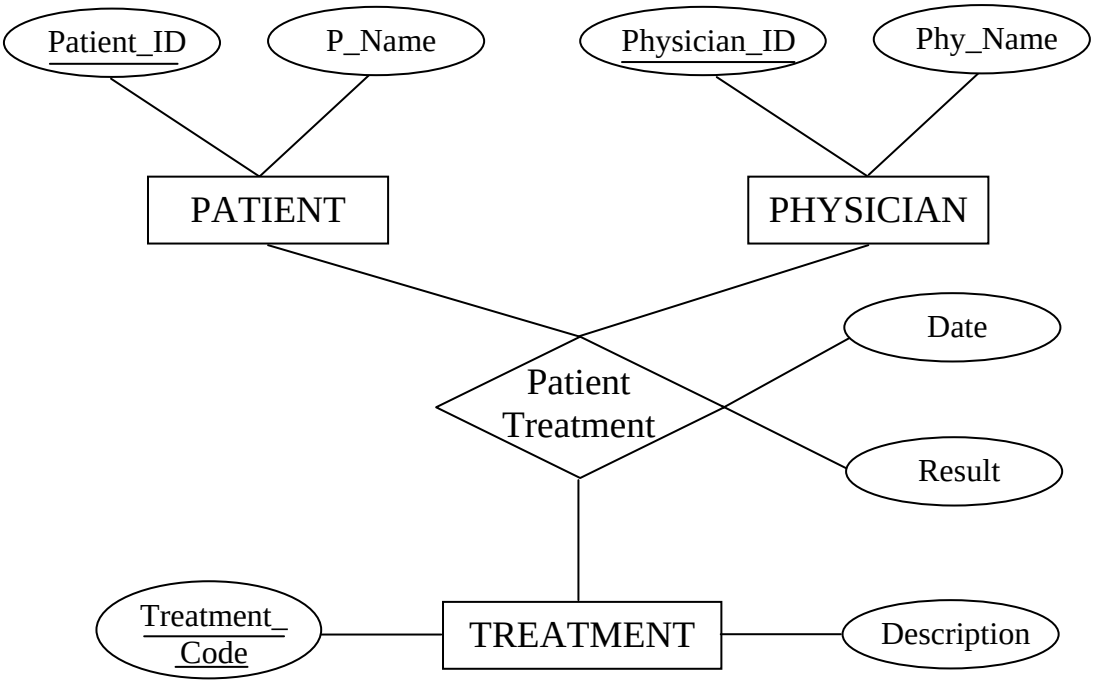
Course (Course_ID, Description, Credit)

Course_Pre (Course_ID, Pre_Take)

(រូប 4.29b)

Step 5: Map Ternary, Quaternary and *n*-ary Relationships

គេធ្វើការបំលែង ternary, quaternary and n-ary relationship ដោយបង្កើត relation ផ្សេងៗគ្នាសំរាប់ entity types ដែលចូលរួមក្នុង relationship និងបង្កើត relation ថ្មីមួយទៀតដោយផ្គុក primary key នៃ relations ទាំងនោះ ក្នុងករណីមួយចំនួន attributes ទាំងនោះនឹងក្លាយទៅជា primary key សំរាប់ relation ថ្មីនោះ។



(រូប 4.30a)

Patient (Patient_ID, P_Name)
Physician (Physician_ID, Phy_Name)
Treatment (Treatment_Code, Description)
Patient_Treatment (Patient_ID, Physician_ID, Treatment_Code, Date, Result)

(រូប 4.30b)

Step 6: Map Supertype/Subtype Relationships

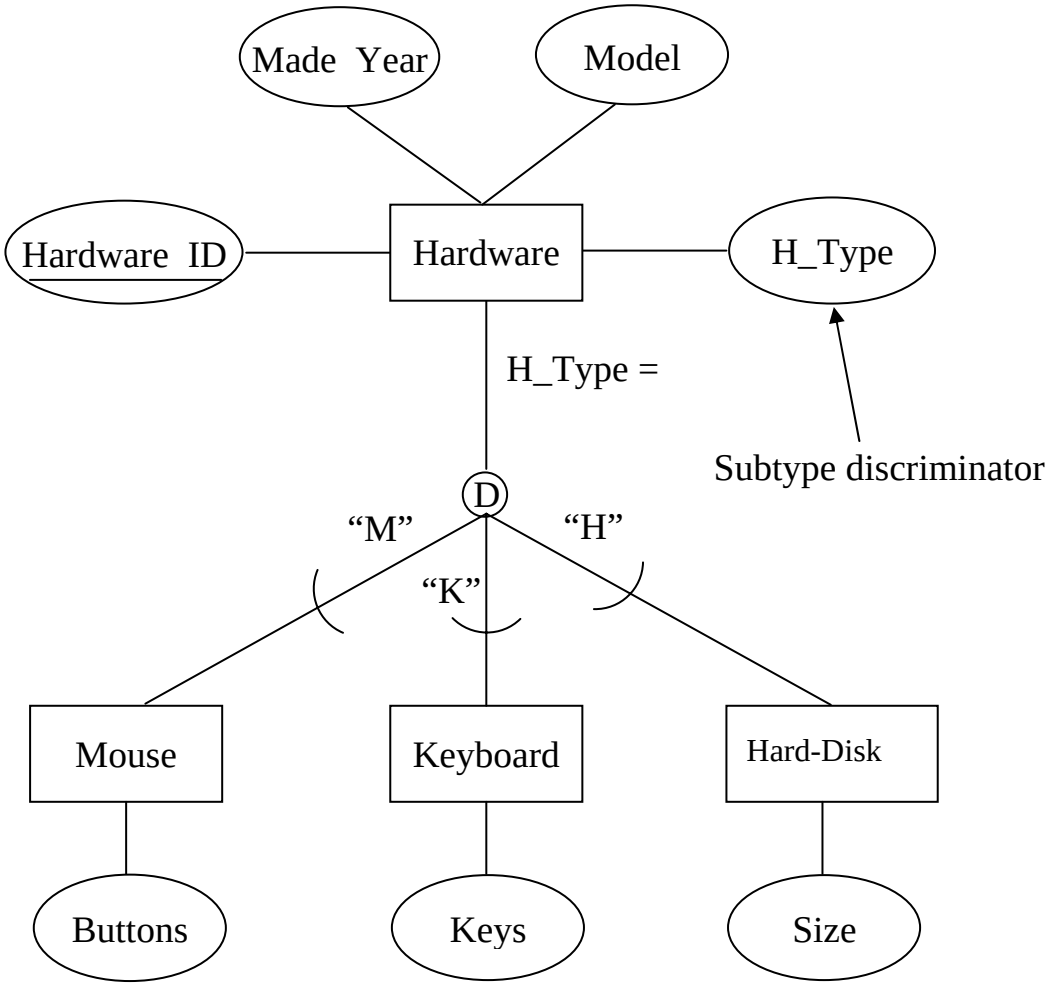
មានវិធីជាច្រើនក្នុងការបំលែង supertype/subtype relationship ទៅជា relations ប៉ុន្តែមិនទាន់មានវិធីណាមួយគេចាត់ទុកជា standard នៅឡើយ ប៉ុន្តែយើងលើកយកវិធីណាដែលគេប្រើច្រាស់ញឹកញាប់ជាងគេតែប៉ុណ្ណោះ។

1. បង្កើត relation នីមួយៗសំរាប់ supertype និង subtypes នីមួយៗ
2. បញ្ចូល common attributes ទៅកាន់ supertype relation ដោយរួមទាំង primary key ផងដែរ

3. បញ្ចូល primary key នៃ supertype relation ទៅកាន់ subtype relation នីមួយៗ ព្រមទាំង specific attributes ចំពោះ subtype relation នីមួយៗ
4. បញ្ចូល attribute 1 (វីច្រើន) ទៅកាន់ supertype អោយជើរតួនាទីជា subtype discriminator

Subtype discriminator គឺ attribute ស្ថិតក្នុង supertype entity ដែលតំលៃរបស់វាប្រើប្រាស់សំរាប់បង្ករ វីបង្ហាញពី subtype ។

ក្នុងករណីដែល relationship មានលក្ខណៈ disjoint ពេលនោះគេប្រើប្រាស់ subtype discriminator តែមួយហើយតំលៃវាប្រើប្រាស់សំរាប់ចង្អុល subtype ឧទាហរណ៍: តាមរយៈរូប 4.31a យើងឃើញថា បើយើងបញ្ចូលតួអក្សរ “M” ទៅកាន់ subtype discriminator មានន័យថា hardware នោះមានប្រភេទជា Mouse ប៉ុន្តែបើបញ្ចូលតួអក្សរ “K” មានន័យថា hardware នោះមានប្រភេទជា Keyboard ហើយបើបញ្ចូលតួអក្សរ “H” វិញ មានន័យថា hardware នោះមានប្រភេទជា Hard disk ។

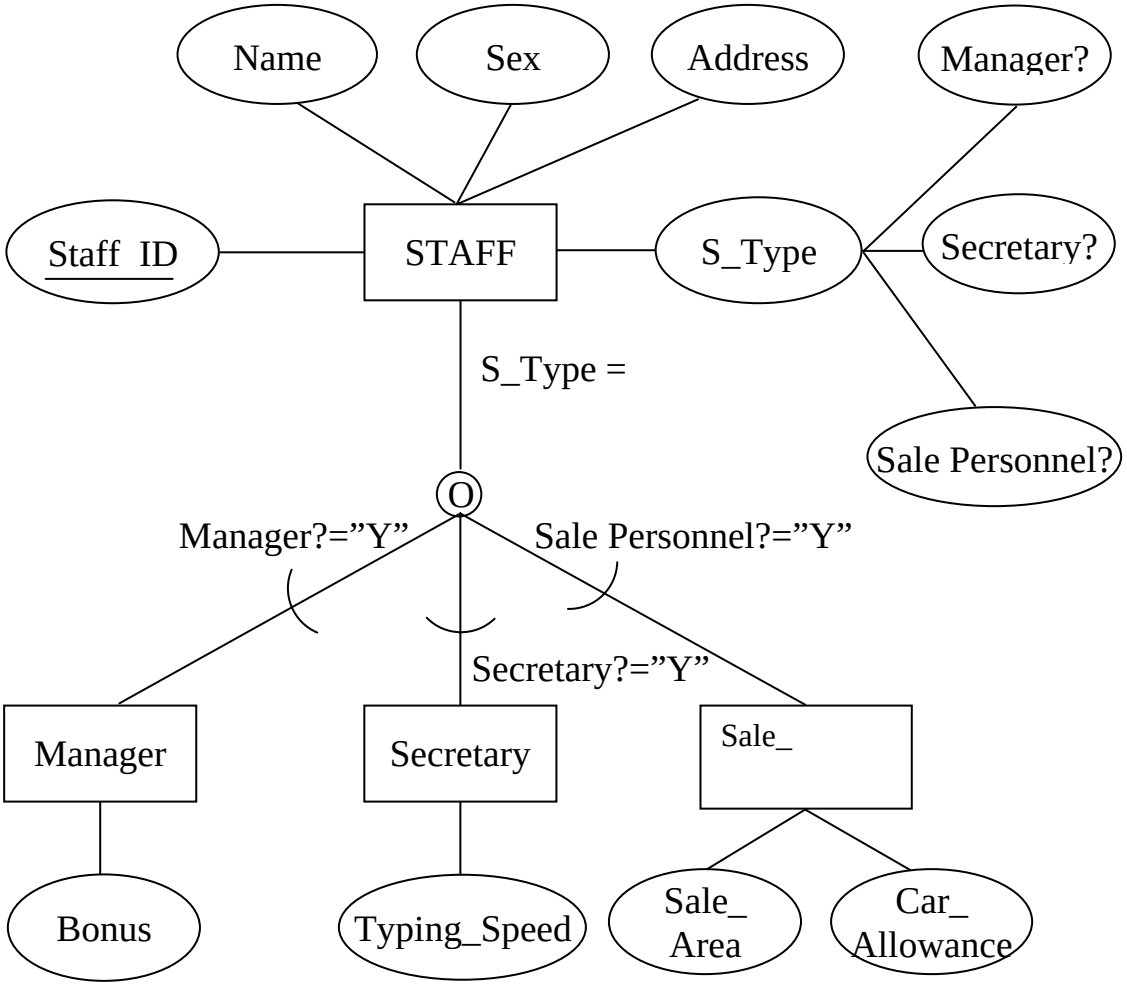


(រូប 4.31a)

Hardware (Hardware_ID, Made_Year, Model, H_Type)
Mouse (Hardware_ID, Buttons)
Keyboard (Hardware_ID, Keys)
Hard_Disk (Hardware_ID, Size)

(រូប 4.31b)

ផ្ទុយទៅវិញ ប្រសិនបើ relationship មានលក្ខណៈ overlap គេត្រូវប្រើប្រាស់ subtype discriminator លើសពីមួយ ហើយ subtype discriminators ទាំងនោះអាចផ្ទុកតំលៃបានតែ 2 គត់គឺ True និង False។ ឧទាហរណ៍: តាមរយៈរូប 4.32a យើងឃើញថា បើយើងបញ្ចូល staff ណាមួយដែលមានតួនាទីដល់ទៅ 2 ជា Manager និង Sale Personnel នោះយើងត្រូវបញ្ចូលចំពោះ record នោះនូវ MFlag ស្មើ True ហើយ SFlag ស្មើ False និង PFlag ស្មើ True។



(រូប 4.32a)

Staff (Staff_ID, Name, Sex, Address, MFlag, SFlag, PFlag)
Manager (Staff_ID, Bonus)
Secretary (Staff_ID, Typing_Speed)
Sale_Personnel (Staff_ID, Sale_Area, Car_Allowance)

(រូប 4.32b)

Review Questions

1. ចូរពន្យល់អោយបានក្បោះក្បាយនូវពាក្យខាងក្រោម

(a) entity
(b) relationship
(c) weak entity

2. អ្វីទៅជា attribute? ចែកជាប៉ុន្មានប្រភេទ? ចូរពន្យល់

3. អ្វីទៅជា degree of relationship? ចែកជាប៉ុន្មានប្រភេទ? ចូរពន្យល់

4. អ្វីទៅជា cardinality ratio? ចែកជាប៉ុន្មានប្រភេទ? ចូរពន្យល់

5. ចូរពន្យល់ពី Attribute inheritance?

6. ចូរពន្យល់ពី Generalization, Specification និង constraint លើ supertype/subtype relationship?

7. ចូរពន្យល់ពីជំហានក្នុងការបំលែង EER diagrams ទៅជា relations?