Chapter 3 DB models & data modeling

1. Introduction

development of DB sys starts w/ a modeling phase – seeks user requirements – turn into tech specifications for implementation

2. Definition & concepts

2.1 Definition of a DB model

model : a collection of concepts, language, graphics to describe the data structure & data processing operations

DB model – describes the design of DB (not the ways of constructing it)

like an architectural plan

serve as the means of communication between sponsor, DB designer, DB developer, users

modeling process

: real world → data abstraction → conceptualization → documentation

* concept : entity (relational DB model), object (OO DB model)

various DB model – hierarchical, network, relational, OO DB model

development stage – conceptual, logical, physical

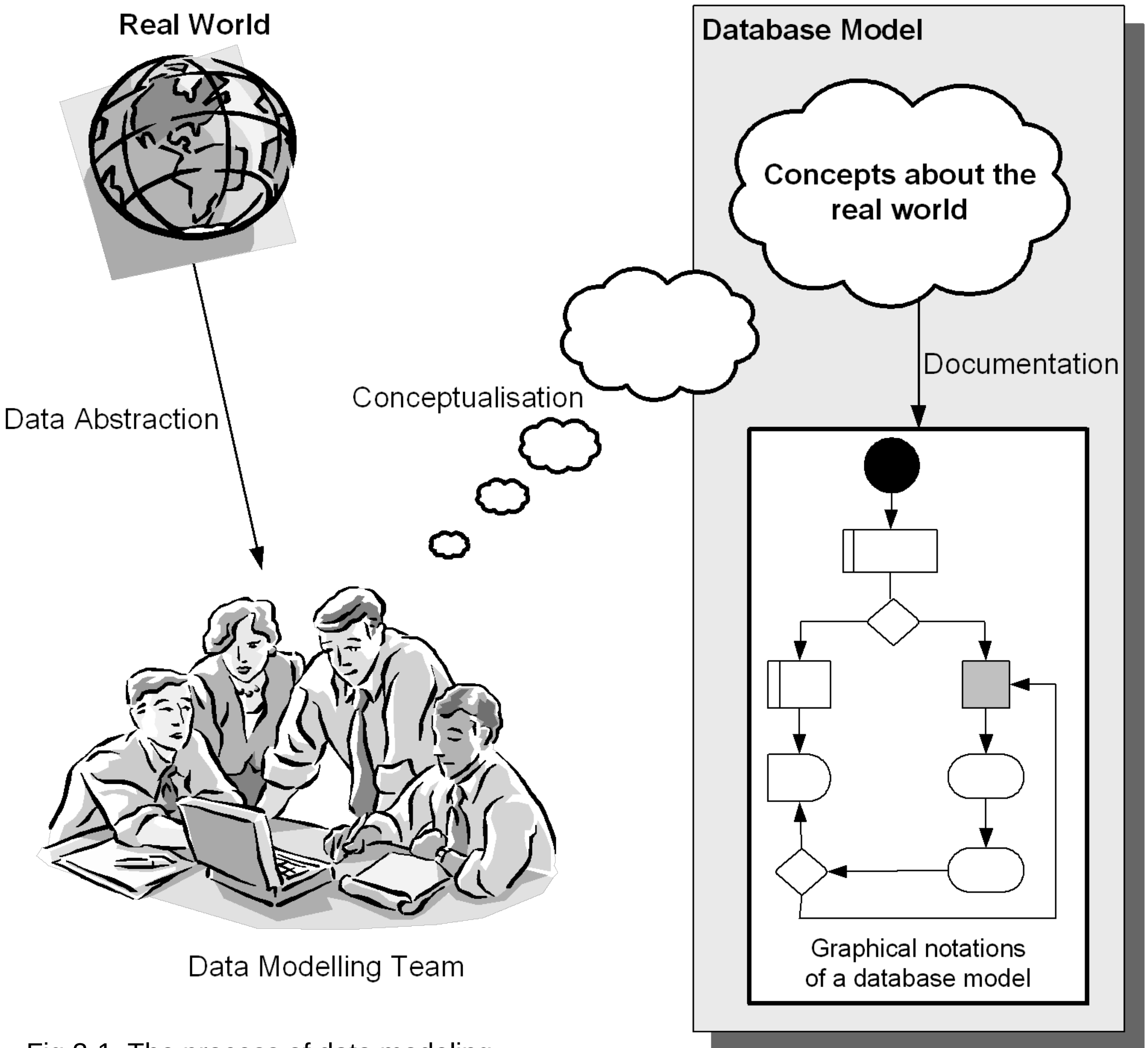


Fig 3-1 The process of data modeling

2.2 DB model, schema, instance

DB model : vocabulary & linguistic/graphic rules to describe DB – high level description

schema : collection of linguistic & graphical representations to describe the data structure

instance : an occurrence of a data object

* in this lecture, DB model + schema = DB model / schema

characteristics

schema – can contain multiple instances

instance – dynamic & time-variant as values of instances can be changed by transactions

cf. schema is static & time-invariant

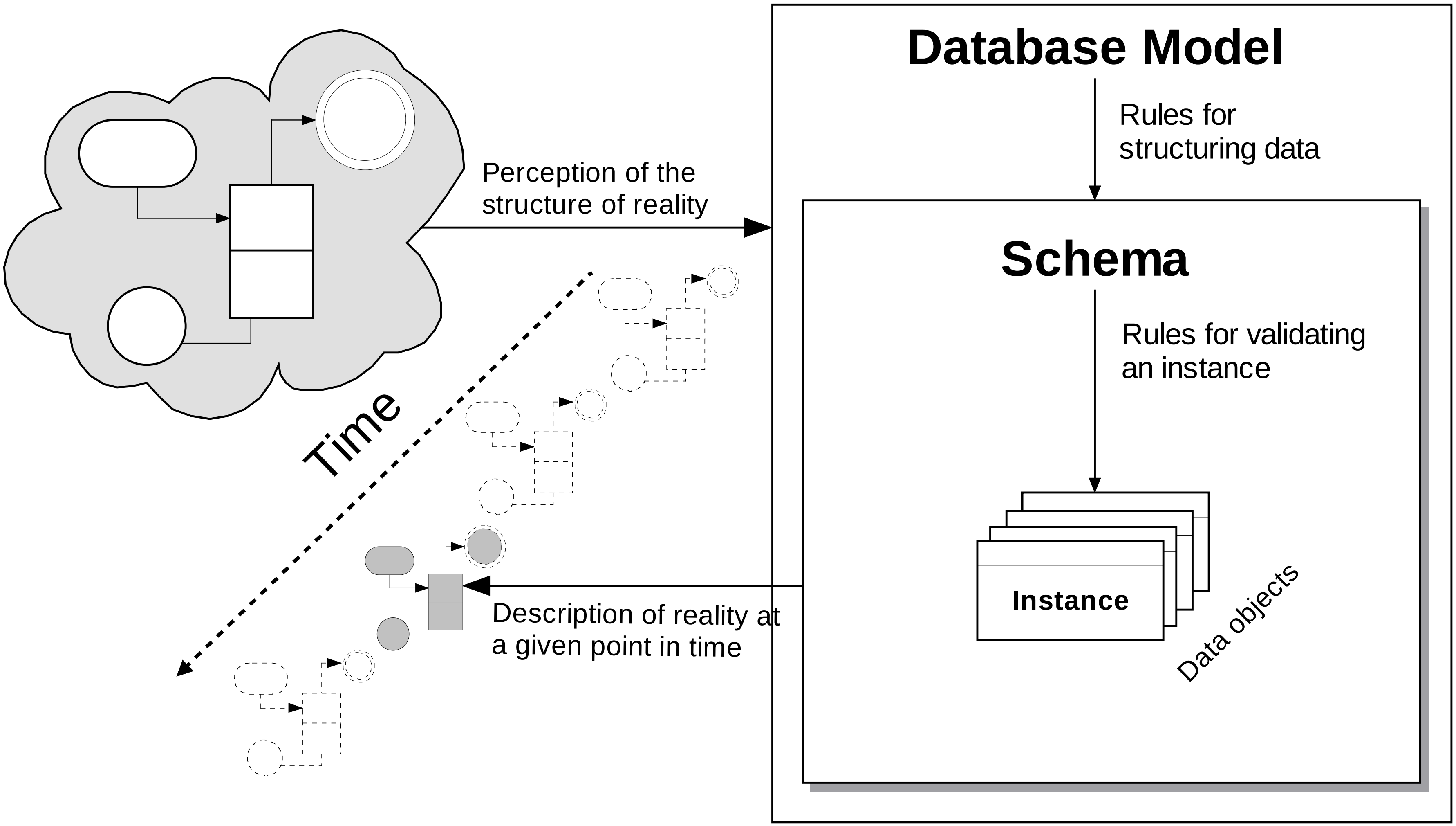


Fig 3-2 Relationship between a DB model, a DB schema, an instance of a data object

2.3 Conceptual, logical, physical data modeling

1) conceptual modeling (including schema)

a process to abstract the characteristics & properties of real world entities

HW & SW independent high-level abstraction

three types of data abstraction

a. classification abstraction : defining classes of real world features

b. aggregation abstraction : defining a new class from one / more sets of other classes

ex. residential subdivision = land parcel + road + land use zone

c. generalization abstraction : defining a set-to-subset relationship between the elements of two / more classes

ex. administrative boundary : country b., city b., provincial b.

modeling results – graphics / verbal descriptions / both

ex. Entity-Relationship (ER) diagram – conceptual schema

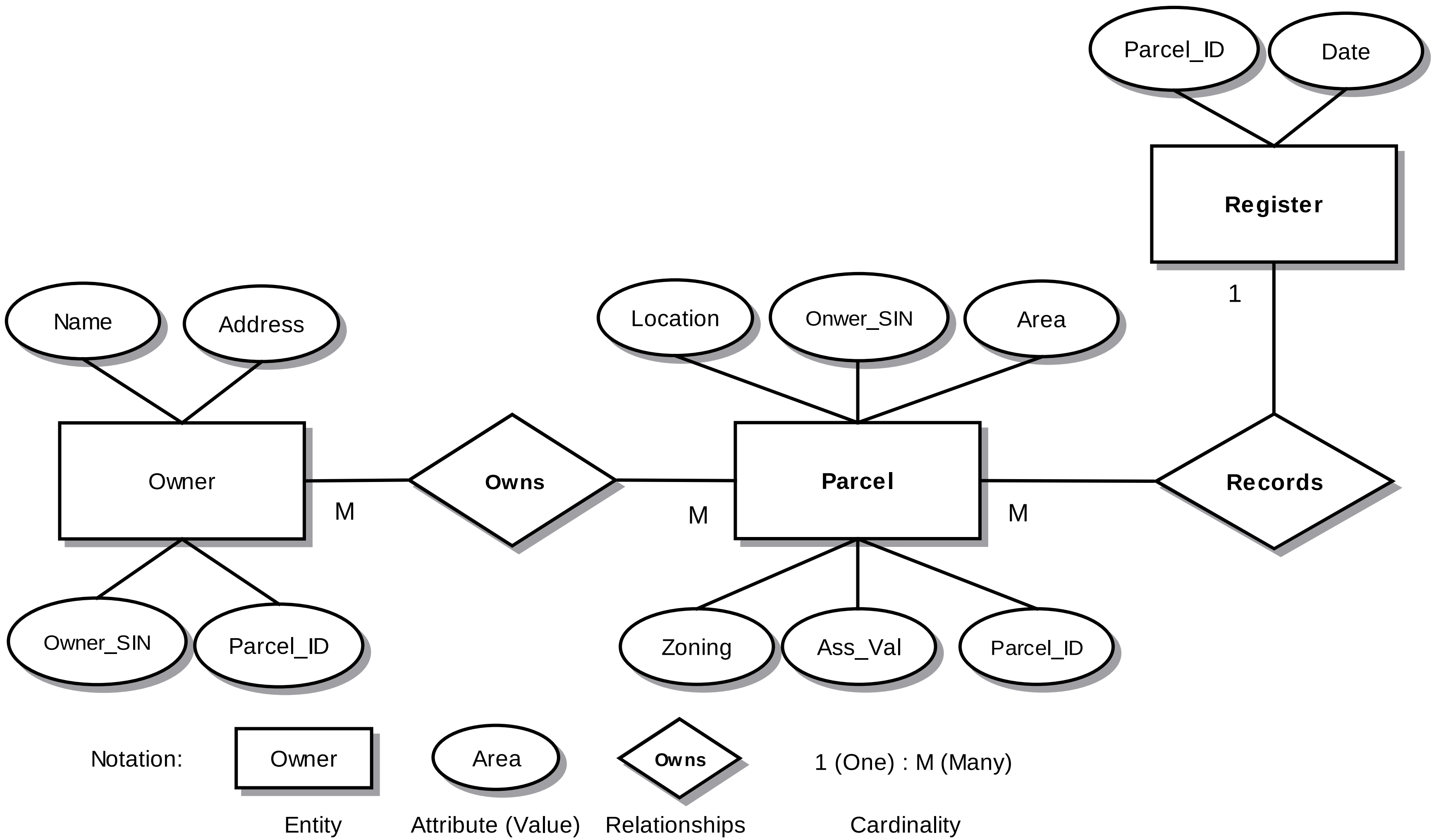


Fig 3-3 An entity-relationship (ER) diagram

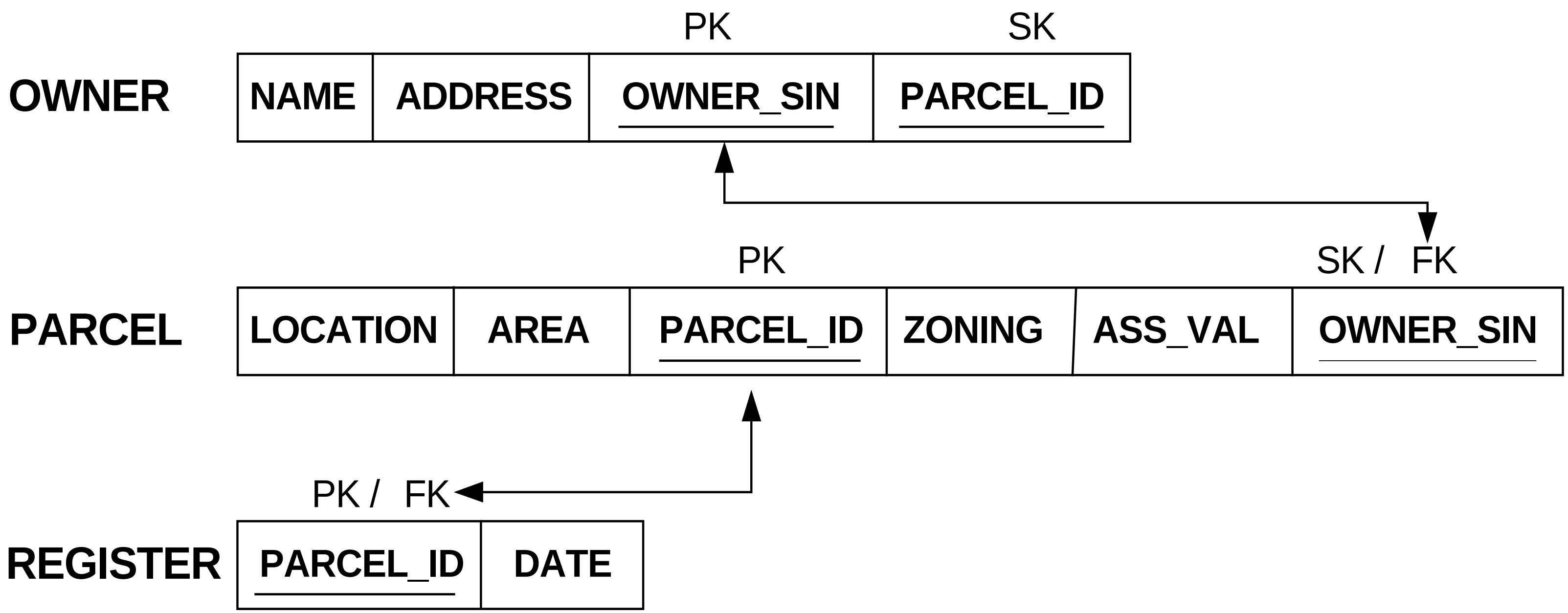
2.3 Conceptual, logical, physical data modeling

2) logical modeling (including schema)

DB implementation specification

translating a conceptual schema according to the linguistic syntax & diagrammatic notation

SW / DBMS dependent (ex. relational, OO, O-relational)



Notation: PK - Primary key; SK - Secondary key; FK - Foreign key

Fig 3-4 A relational logical schema developed from the conceptual schema in Fig 3-3

2.3 Conceptual, logical, physical data modeling

3) physical modeling

detailed specifications of the data structure of a DB considering HW requirements

* HW requirements in data modeling – computer & system arch of the DB system, physical location of data files, specific allocation of storage space to data objects

PARCEL

Definition: A taxable unit of land within city limits.
Feature type: Polygon
Implemented as layer: Cadastral_fabric
Business table: LAND_PARCEL

ATTRIBUTE DEFINITION

Name	Type	Size	Optional	Unique	Indexed	Key
LOCATION	Char	100	M	Y	N	
AREA	Num	10.2	M	N	N	
PARCEL_ID	Char	15	M	Y	Y	P
ZONING	Char	5	O	Y	N	
ASS_VAL	Num	5.2	M	N	N	
OWNER_SIN	Char	9	M	Y	Y	S / F

ATTRIBUTE DESCRIPTION

LOCATION	Municipal address of parcel, including Street Number, Street Name, Street Type, County Name, Province Name, Postal Code
AREA	Size of parcel, as obtained from survey plan, in sq. m. to 2 decimal places
PARCEL_ID	Unique identification number assigned by Property Assessor's Department for a PARCEL, used as priamry key
ZONING	Zoning code, assigned by planning department (Refer to Appendix D for Zoning Codes)
ASS_VAL	Assessed value as determined by the Property Assessor 뭘 Department
OWNER_SIN	Parcel owner 뭘 socia key and foreign key for OWNER table.

DATA LOAD / STORAGE

Initial volumn: 10000 Growth: 10% per year
Space: 25 Blocks, 50206 bytes Initial allocation: 600k Next: 60k

Fig 3-5 A portion of the physical model developed from the logical schema in Fig 3-4

3. Common DB models

3.1 Entity-relationship (ER) model

conceptual DB model describing at a high-level of abstraction

characterized by the use of diagrams to express & describe its concepts

objective is to identify the entities, relationships, attributes

terms – entity (=data object, object) : a real world feature / phenomenon of independent existence

ex. temperature, land value, contour lines

entity type (=entity class) : entities sharing common properties

relationship : association between entities

ex. belongs_to, managed_by, has

four properties

cardinality : number of occurrances of the entities participating in a relationship ex. many-to-many

optionality : whether the relationship is optional / mandatory

constraints : business rules governing a relationship ex. no one under 18 is allowed to register

attribute classification

simple / composite attributes : cannot be subdivided (ex. id) / subdivided (ex. name = first+m+last)

single-valued & multi-valued attributes : each occurrence has one value / multi value

derived attributes : computed one from another attribute

keys : index to search a DB ex. PK, SK, FK

no standardized notation for E/R - growing trend to use UML (unified modeling language)

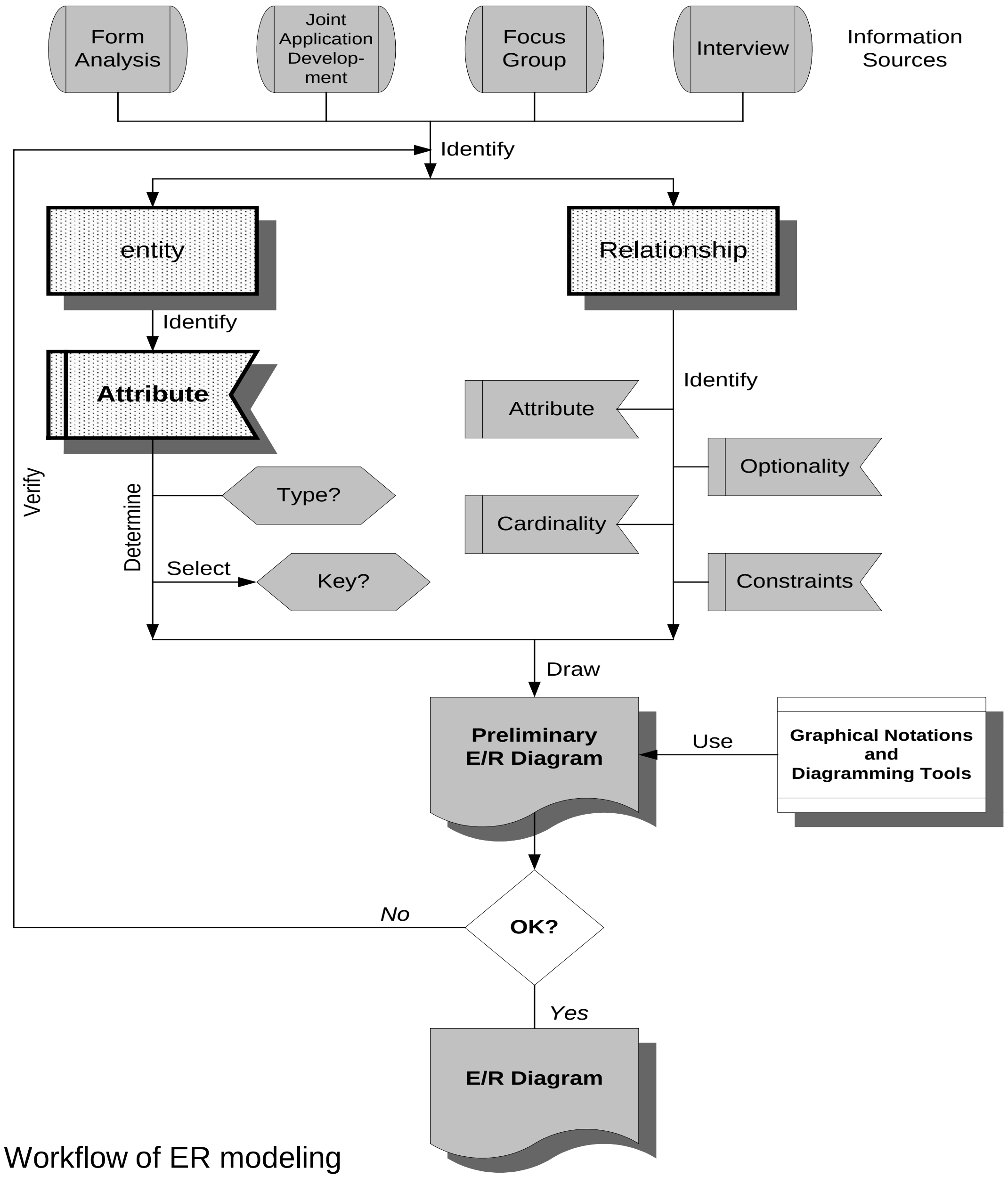


Fig 3-6 Workflow of ER modeling

3.2 Relational model

data are logically structured within *tables (=formally called relation)*

table characteristics

- unique name – distinguishes it from other tables
- column (=field) – represents one of its attributes
- key – work as an index   ex. PK(primary key), S(secondary)K, F(foreign)K
- row (=tuple / record) – represents an instance / occurrence of an entity
- domain – type of values stored in the cells

integrity constraints - a relational table must conform

- domain constraint : limit what values can be permissible values
- entity constraint : a primary key cannot be null   (*null : missing value case)
- referential constraint : insertion of a new row w/ particular value whenever it is in another table FK

business rules – a condition that the use of a relational table must satisfy

- ex. land parcel DB – registered only when a document is verified & a fee is paid

another rules governing the structure – called *normal form*

- 1NF(normal form), 2NF, 3NF

Permissible values:

Polygon ID	Land cover code	Area (m^2)	Gradient (^{o})	Soil type	Landuse Suitability
	List of land cover codes		*0 to 90*	*List of soil types*	*List of codes*

Column → Polygon ID; Attribute → header row; Tuple (Row) → first row; Relation → whole table; A domain value → 3.5

Polygon ID	Land cover code	Area (m^2)	Gradient (^{o})	Soil type	Landuse Suitability
0123	Wd	2370	2.5	Sand	M
0124	Grs	3450	3.0	Loam	H
0125	Shrb	5900	5.5	S-loam	H
1209	CGrs	3700	4.0	Sand	M
1345	Wd	6000	3.0	Loam	M
5434	Otcrp	7500	2.0	Rock	XL
2234	Grs	2500	4.0	Clay	L
3000	DWd	4550	3.5	Loam	H

Fig 3-7 Features of a relational table

Table 3-1. First, second and third normal forms

Normal Forms	***Rules***
First (1NF)	○ There are no repeating attributes in the table (that is, no two columns are allowed to store identical attributes, for example, the land use status of a parcel at different points in time)
Second (2NF)	○ The table is in 1NF, and ○ All non-key attributes are functionally dependent on the primary key
Third (3NF)	○ The table is in 1NF and 2NF, as well as ○ There is no transitive dependency of attributes on the non-primary key ("transitive" in this context means indirect)

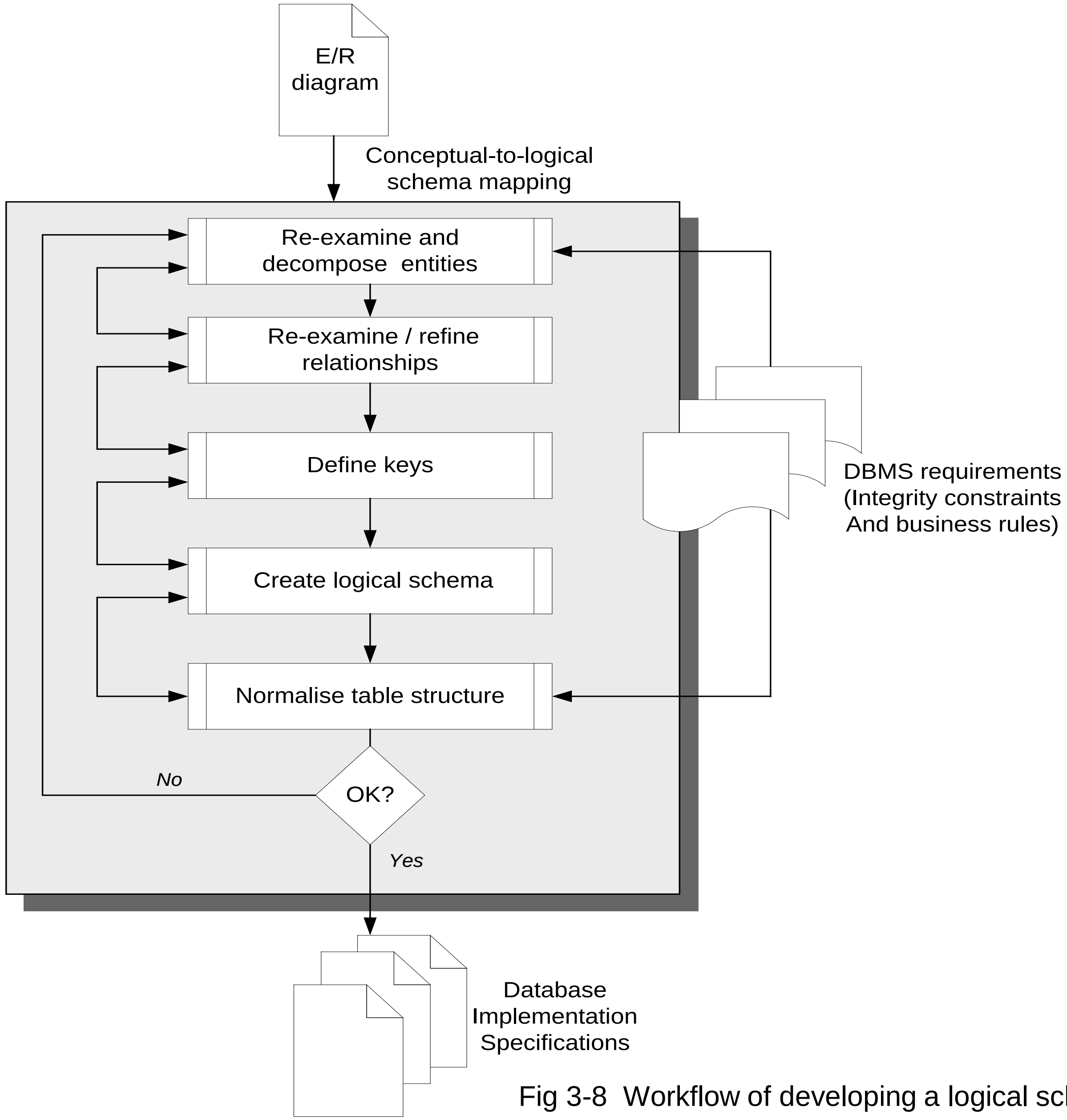


Fig 3-8 Workflow of developing a logical schema

3.3 Object-oriented (OO) model

current DB requires to store & process text + graphics + video + sound + maps → leads to OODB

need to understand *object*

- conceptually autonomous data item representing a real world entity
- can include tasks it performs - act upon itself & interact w/ other objects

important components & characteristics of objects

- name : assigned by DB designer
- unique identity : object ID (OID)
- attributes : instance variables
- object state : set of values for an object's attribute at a given point in time
- base data type : conventional data types – string, real, integer – use predefined arithmetic operators
- abstract data type : user defined data type – has user defined operations (methods)
- method : a program to perform a specific operation (also called *service*)
- message : used to invoke a method by specifying object · method (and parameters)
- type : specification of an interface that an object will support
- control & business rules : govern the use of an object (i.e. its behavior)

3.3 Object-oriented (OO) model

class is a major building block in OODB

all classes are organized into a class hierarchy (Fig 3-9) – super class vs sub class

→ inheritance is possible & overriding, polymorphism

OO model describes data + DB operations + processes within a single object

OO model is a more complete & meaningful description of a DB than relational models

OMG (object management group)

produces & maintains vendor-independent standards & specifications for OO models, systems, DBs

ex. OMA (object management arch) : standards for interoperation of objects across different sys

UML (unified modeling language) : diagrammatic language for modeling, designing, visualizing

CORBA (common object request broker arch) : standard of the OMA for client/server DB sys

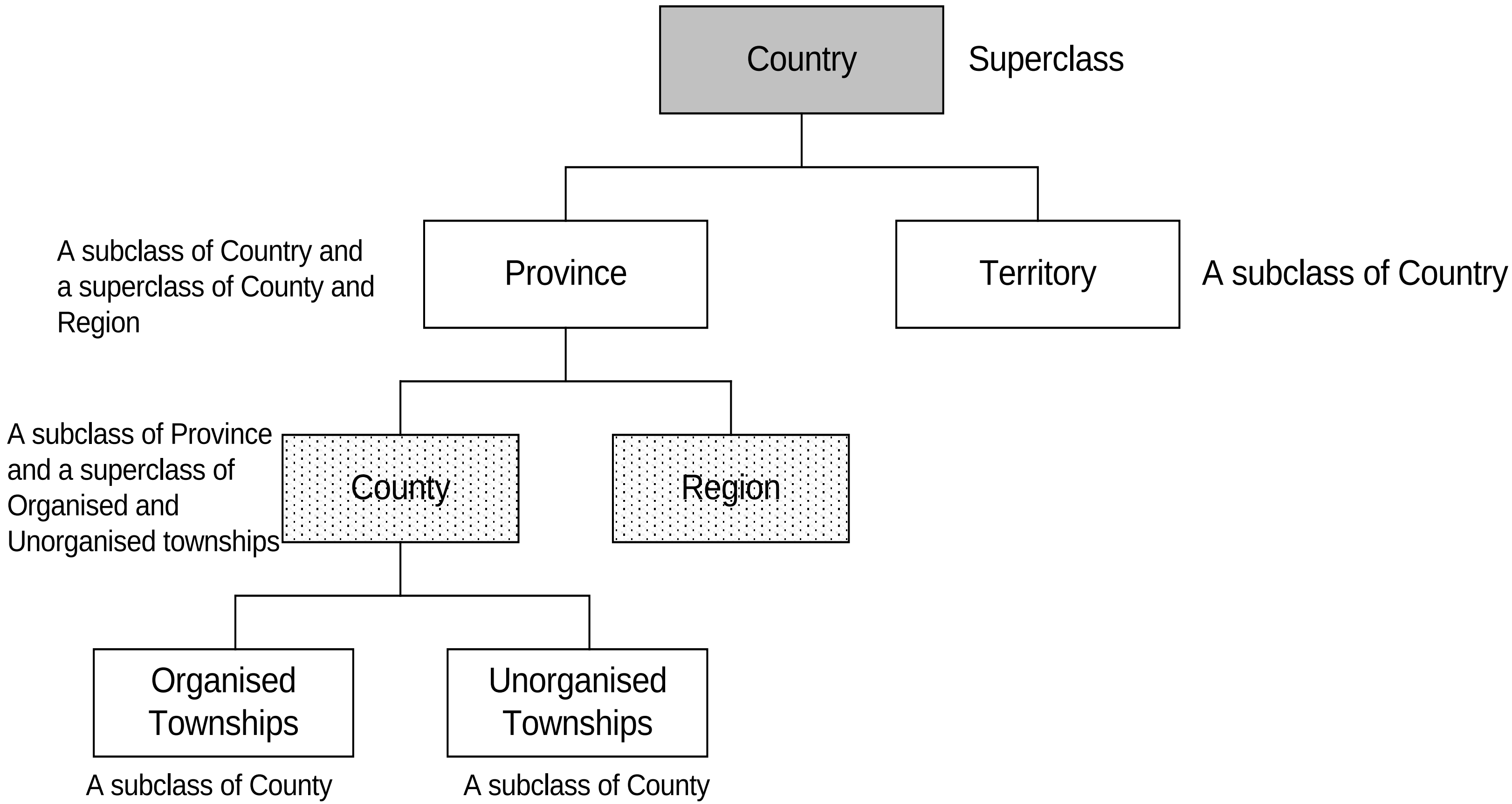


Fig 3-9 The concept of class hierarchy in OO

3.3 Object-oriented (OO) model

OO modeling processes

structural modeling : to identify all the things for an application ex. Land parcel – owner, registrar

to identify attributes, operations, associations, interdependencies bet things

output – object diagram, class d., component d., deployment d.

→ conceptual modeling process similar to ER modeling

behavioral modeling : to identify the dynamic aspects of the sys

- defining roles of objects (classes), interactions among them, flows of control

  concerned w/ methods, messages associated w/ objects

output – activity diagram, sequence d., interaction d., collaboration d.

→ logical modeling process

arch modeling : model implementation aspect of the sys

divide the sys into physical parts (called components of HW & SW)

cover a wide spectrum of modeling tasks

  user interfaces, data files, tables, executables, code libraries

output – component diagrams, deployment d.

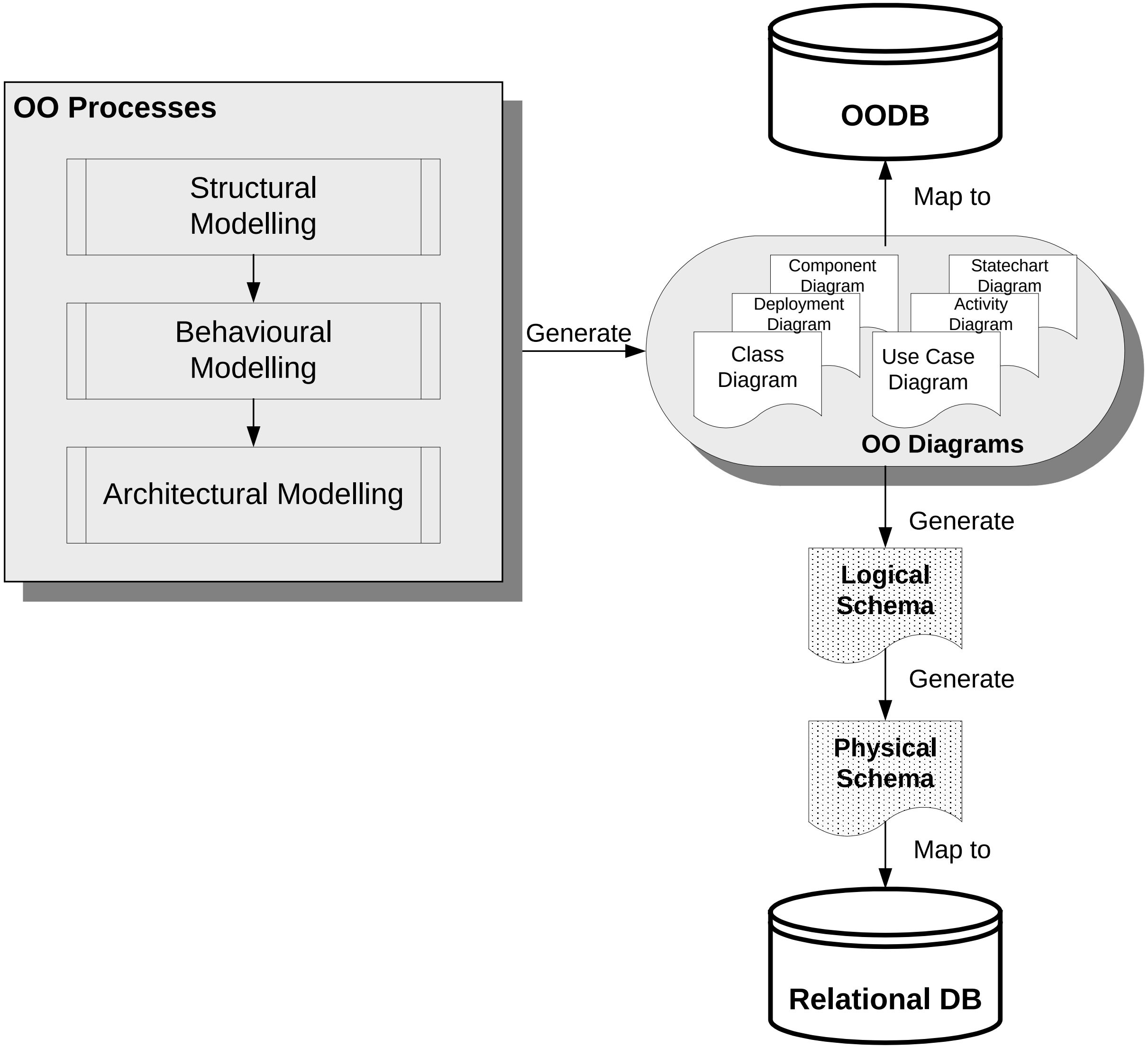


Fig 3-10 Workflow of OO modeling

3.4 Object-relational model

developed to overcome the limitations of relational sys to handle complex data

introduce many of the concepts of OO sys
: object storage, user-defined data types, inheritance, encapsulation

main characteristics :

- user-defined data types - manage complex data types encapsulating data structure & attributes
- user-defined functions – create, manipulate, access data stored as user-defined data types
- extensible optimizer – help the DBMS determine the best way to access data

inherited robust transaction management capability (relational sys) + flexibility of data storage, access (OO sys)

4. Principles & tech of data modeling

4.1 The four principles of data modeling

choice of a model has a profound influence on how a problem is approached & how a solution is formed

every model may be expressed at different levels of precision

best models are connected to reality

no single model is sufficient & every non-trivial sys is best approached thru a small set of nearly independent models

4.2 The sys & DB development life cycle

SDLC is a generic description of the process of developing a DB sys

six phases – planning, analysis, design, building, implementation, maintenance

DB DLC is a generic description of the process of developing a DB

six phases – DB initial study, DB design, implementation& data loading, testing& evaluation, operationalization, maintenance& monitoring

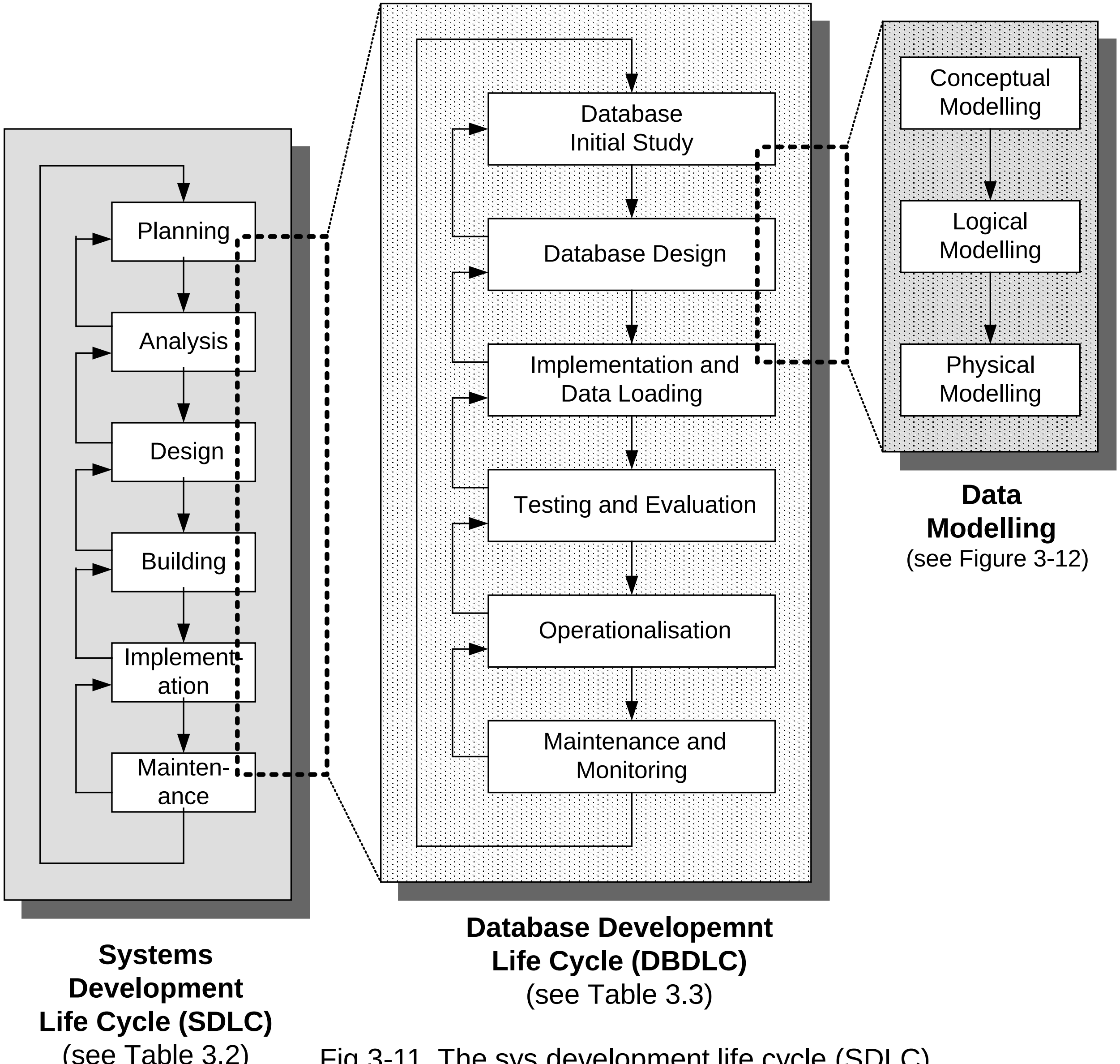


Fig 3-11 The sys development life cycle (SDLC)

Table 3-2. Activities of the systems development life cycle (SDLC)

SDLC Phases	*Activities*
Planning	○ Initial understanding of business functions ○ Initial assessment of user requirements ○ Feasibility study to implement the database
Analysis	○ Systematic assessment of user requirements ○ Evaluation of existing business practices and operations ○ Evaluation of existing data resources
Design	○ Development of hardware/software architecture ○ Development of systems performance standards ○ Development of data structure
Build	○ Application programming (on a development computer) ○ Database programming (on a development computer)
Implementation	○ Hardware/software installation of the production computer/server ○ Data loading to the production computer/server ○ Systems testing and fine tuning ○ User education and training
Maintenance	○ Performance monitoring and evaluation ○ Regular maintenance including database backup ○ Continuing user education and training

Table 3-3. The database development life cycle (DBDLC)

DBDLC Phases	***Activities***
Initial Database Study	○ Analysis of business functions and information needs ○ Identifying problems and constraints ○ Defining goals and objectives of the database project ○ Defining scope and database performance standards
Database Design	○ Conceptual database modelling ○ Hardware and software (DBMS) selection ○ Logical database modelling ○ Physical database modelling
Implementation and data load	○ Hardware and software (DBMS) installation ○ Creating data structure ○ Data loading, including any data conversion
Testing and evaluation	○ Testing and fine-tuning database ○ Testing and fine-tuning application programs
Operationalisation	○ Putting the database into production mode ○ User education and training
Maintenance and monitoring	○ Regular maintenance of hardware and software, including change management in hardware and software upgrades ○ Database backup and replication ○ Continuing user education and training

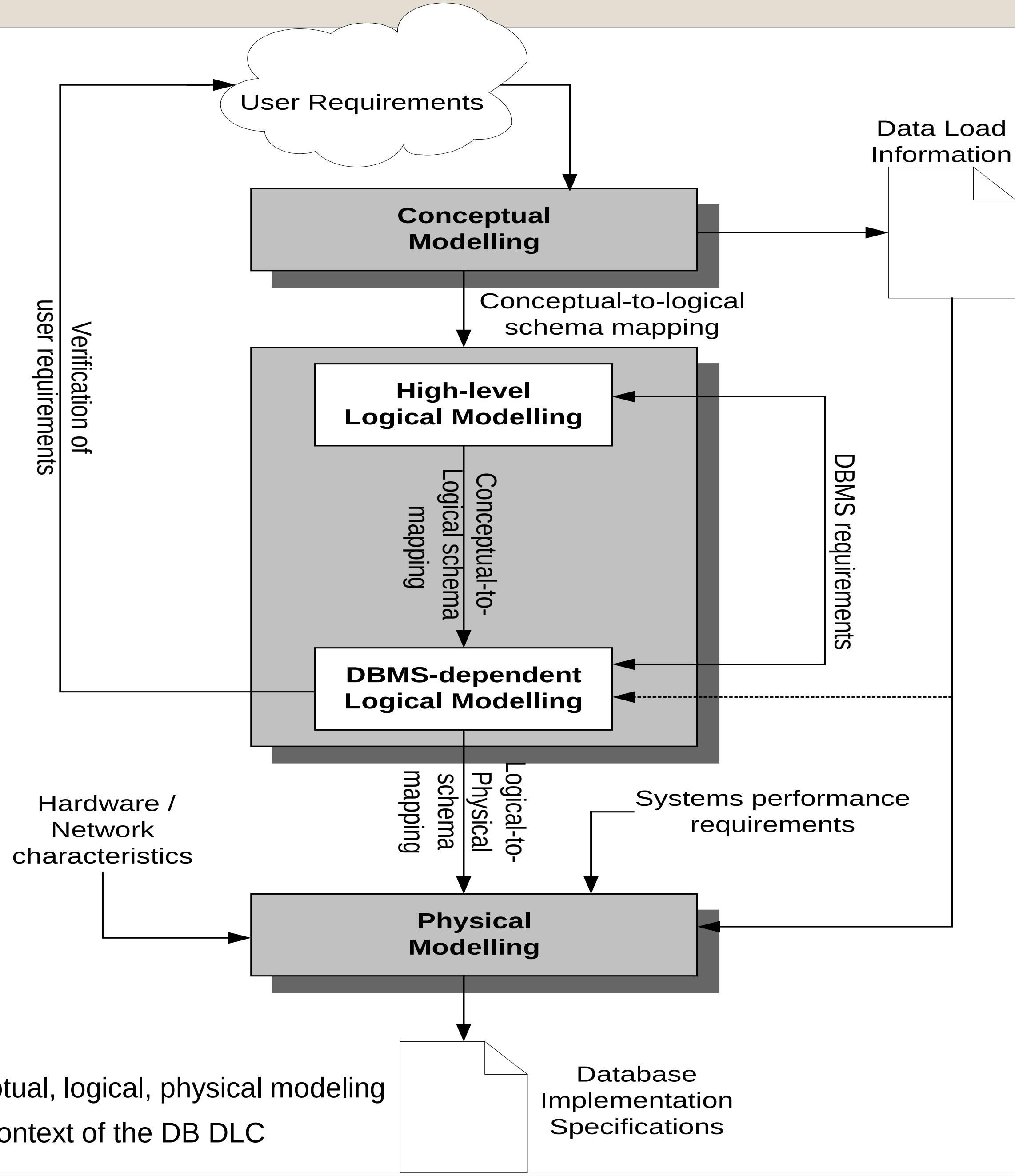


Fig 3-12 Conceptual, logical, physical modeling in the context of the DB DLC

4.3 Case tools (CASE : computer aided SW engineering)

used to automate the sys development activities in a SDLC, DB DLC

typical CASE tool components

- sys development environment – a set of drawing tools (describe & document DB schema, flows of data, application processes, user interface)
- repository – stores & integrate all sys development decisions & results of sys & design activities
- data dictionary – keeps track of all objects created (ex. entity descriptions, attribute definitions, data store, screen interface formats)

  also records the relationships among these objects, rules

three classes of CASE tools

- front-end tools – support planning, analysis, design phase
- back-end tools – support building & implementation phase
- cross life cycle tools – support all the activities across the entire SDLC

4.4 User-centric DB design

traditional sys developments – technology centered, application driven

thus, user centered design (UCD) methodology is developed

UDC is driven by a) clearly specified task-oriented business objectives

b) recognition of user needs, preferences, constraints

merits of UDC

- providing a well-structured framework compatible w/ the concepts of the SDLC
- user participation in the modeling process
- high-level of user-designer interaction during the modeling process
- iterative approach to data modeling

4.5 Data modeling documentation

documentation is the only tangible outcome of data modeling

general guidelines for documentations – expressiveness, simplicity, minimalism, formality

UML – now emerged as de facto industry standard for documentation

- non proprietary standard that is open to all users
- created by fusing OMT (object modeling tech)+OOSE (object oriented SW engineering)

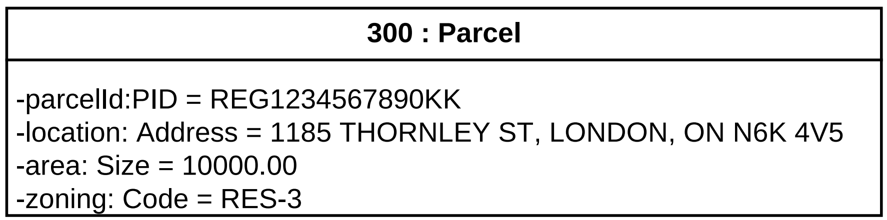
UML meta model : a set of definitions to describe the meaning of each element used

- has four layer architecture
  - user object : object diagrams populated w/ the facts from problem space of the DB (Fig 3-13 b)
  - model : explains the classes (Fig 3-13 c)
  - meta model : define class & attribute definitions

    (ex. class name, data type, default value, constraints)
  - meta meta model : abstract definitions to serve as templates

UML Layer	Description	Example
Metametamodel	Defines the language for speficying metamodels	Meta-class, Meta-attribute Meta-operation, etc.
Metamodel	Defines the language for specifying models	Class, Attribute, Operation
Model	Defines the language for describing subject domains	Parcel, Owner, Registers
User object	Defines specific subject domain information	Parcel (id, location, area, zoning) Owner (name, address, SIN) Registers (sin, pid, date)

(a) The Four-layer Metamodel Architecture of UML



(b) An object diagram of PARCEL

(c) A Class Diagram

Fig 3-13 UML layers & features