BPWin

What is BPWin?

- AllFusion Process Modeler (BPWin) is a comprehensive business-modeling environment that helps you to visualize, analyze, and improve business processes.

- Business process improvement includes mapping and modeling the myriad of interactions within an organization to better understand and improve its operation.

Business Modeling

- Modeling is one of the most effective techniques for understanding and communicating business rules and processes.

- Graphics (mainly boxes and arrows) are used to provide much of the structure.

BPWin

- The BPwin model provides an integrated picture of how organization gets things done, from small departments to the entire organization.

BPwin supports the following three modeling methodologies:

- IDEF0 function modeling method
- IDEF3 process description capture method
- DFD data flow diagramming

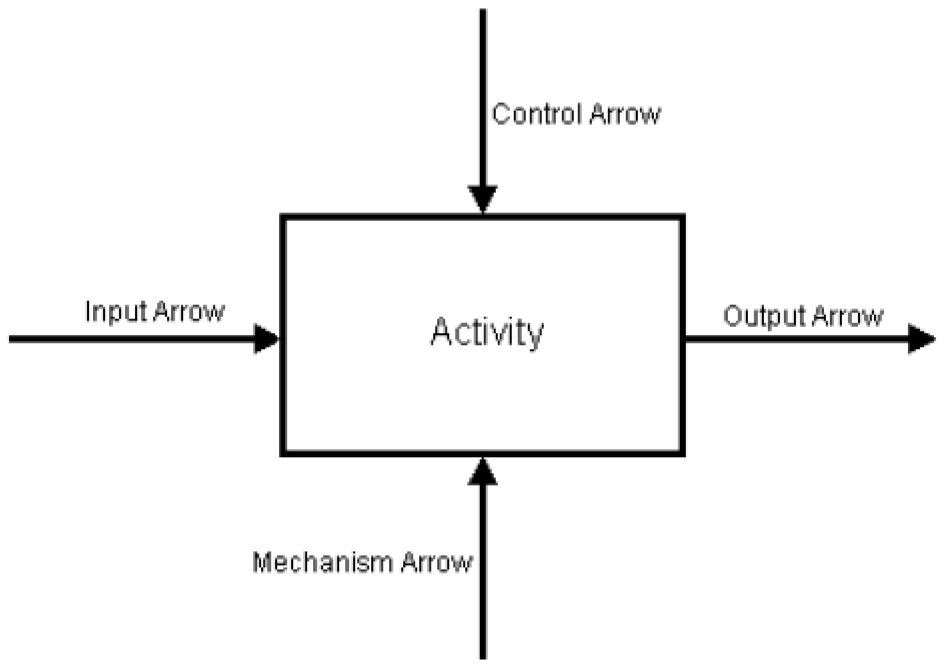
Business Process Modeling (IDEF0)

- Technique for analyzing the whole system as a set of interrelated activities or functions.
- The activities (verbs) of the system are analyzed independently of the object(s) that perform them.
- IDEF0 models a system using only two graphic symbols: boxes and arrows.

Context Activity

- The first step in creating a model is to create the highest level, the context activity. The context activity describes the system itself and is drawn as a box and given a name. Activity names generally consist of a single active verb plus a common noun that clarifies the objective of the activity from the viewpoint of the modeler

Arrows



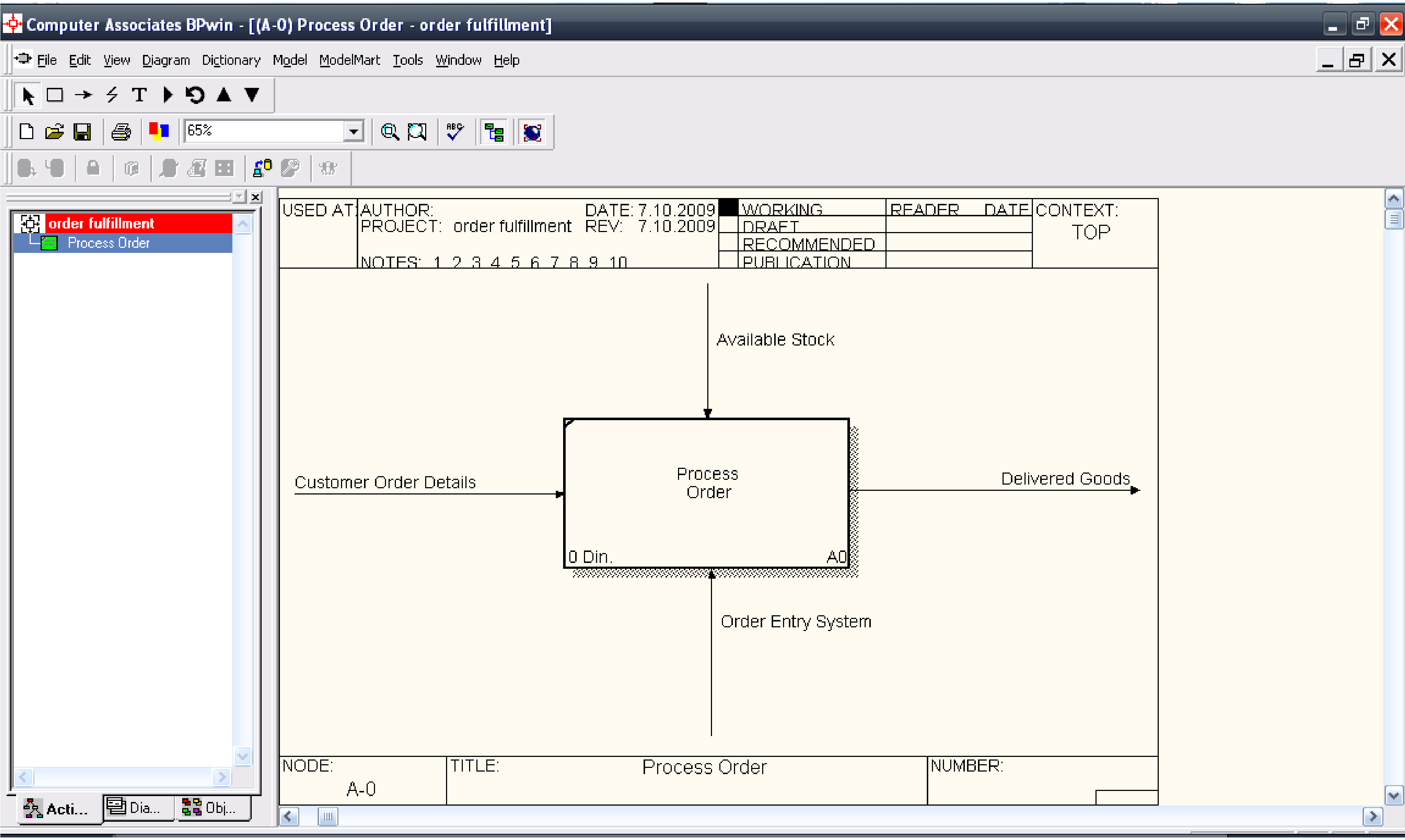
Type of Arrow	What the Arrow Represents
Input	Something consumed or modified in the process
Control	A constraint on the operation of the process
Output	Something resulting from the process
Mechanism	Something used to perform the process, but is not itself consumed

- Inputs represent material or information that is consumed or transformed by the activity to produce outputs.

- Controls impose rules that regulate how, when, and if an activity is performed and which outputs are produced. Controls are often in the form of rules, regulations, policies, procedures, or standards. They influence an activity without actually being transformed or consumed. Controls can also be used to describe items that trigger an activity to start or finish.

- Outputs are the material or information produced by the activity. An activity that does not produce a definable output should not be modeled (or, at a minimum, should be a candidate for elimination).

- Mechanisms are physical resources that perform the activity. Mechanisms could be the important people, machinery, and/or equipment that provide and channel the energy needed to perform the activity.


Computer Associates BPwin - [(A-0) Process Order - order fulfillment]
File Edit View Diagram Dictionary Model ModelMart Tools Window Help
65%
order fulfillment
Process Order
Acti...
Dia...
Obj...

USED AT: AUTHOR: DATE: 7.10.2009
PROJECT: order fulfillment REV: 7.10.2009
NOTES: 1 2 3 4 5 6 7 8 9 10

		READER	DATE	CONTEXT:
■	WORKING			TOP
	DRAFT			
	RECOMMENDED			
	PUBLICATION			

NODE: A-0 TITLE: Process Order NUMBER:

Decomposing a Model

- Decompositions are used in business process modeling to break an activity into its constituent activities. Each of these activities can in turn be decomposed into its own constituent activities. Each time you decompose an activity, you create a decomposition diagram.The number of decomposition levels is entirely up to you, and depends on the level of complexity you need to model.Note the yellow bubble on the figure. When an activity has not been decomposed, the “leaf” symbol will appear in the upper left corner of the activity box (called a "leaf-level" activity). After decomposition the leaf symbol is removed.

How to Decompose an Activity

- Click the diagram activity you want to decompose. Then click the Child tool in the BPwin Toolbox.

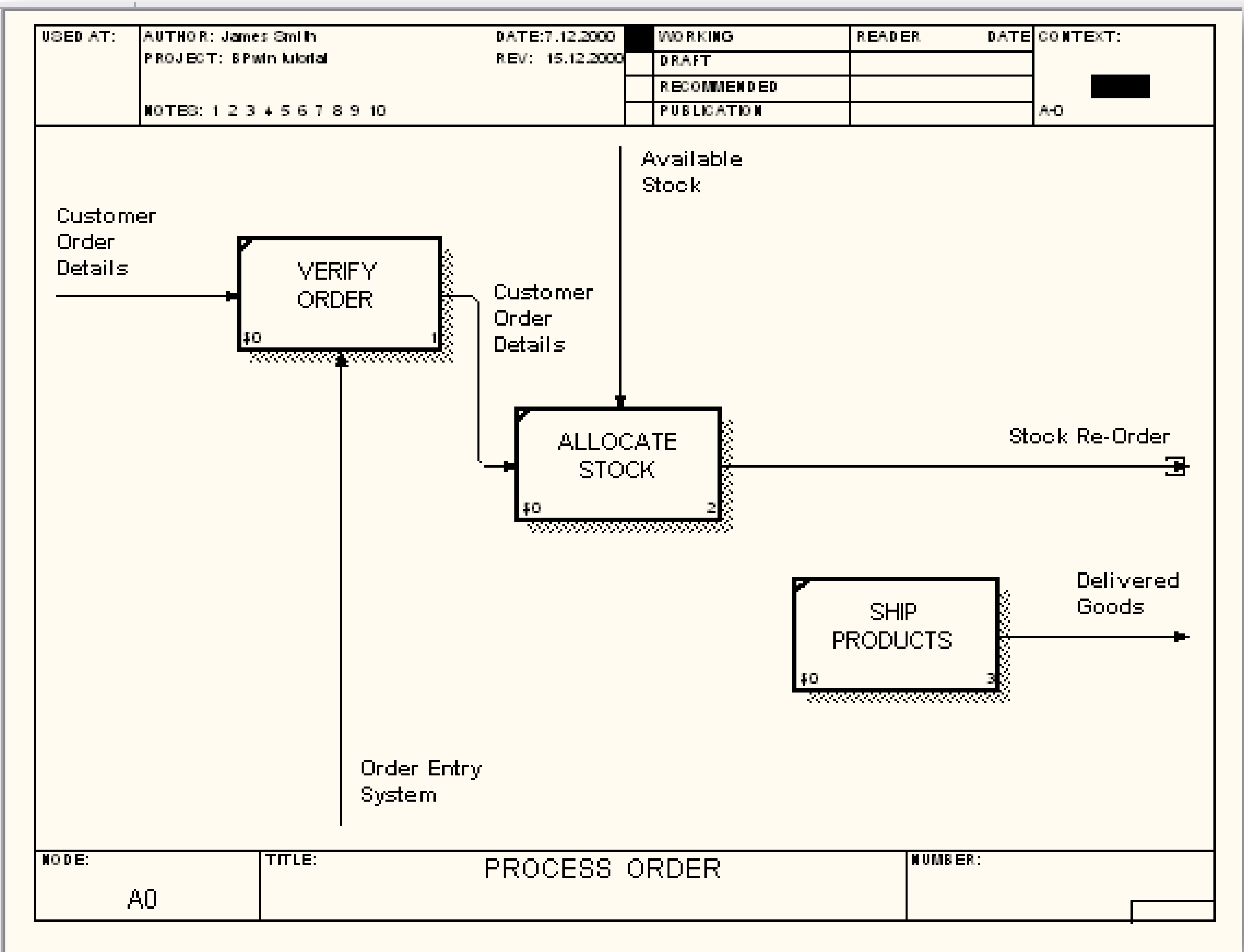
Or

- In the Model Explorer Activities tab, right-click the activity you want to decompose. Then choose Decompose on the shortcut menu. The Activity Box Count dialog will be displayed. Here you can specify the type and number of sub-activities needed. When you click OK, the new decomposition diagram will be displayed.

- The activities in the decomposition are not connected or labeled.

- The decomposition diagram automatically inherits the arrows from the parent activity.

- After naming the activities in a decomposition diagram, the next step is to draw the appropriate arrows that represent the inputs, outputs, controls, and mechanisms that describe the process.



USED AT:	AUTHOR: James Smith PROJECT: BPwin tutorial NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 7.12.2000 REV: 15.12.2000	[✓ WORKING] DRAFT RECOMMENDED PUBLICATION	READER DATE	CONTEXT: A-0

NODE: A0	TITLE: PROCESS ORDER	NUMBER:

Arrow Tunnels

- When there is a break in the arrow hierarchy in a model, BPwin creates a square arrow tunnel on the arrow stub to indicate that the arrow is unresolved within the model hierarchy. This means that there is no representation of the arrow in any other diagram in the model.
- Resolve it to a border arrow
- Resolve it with a round tunnel
- Create an external reference
- Create an off-page reference

Border Arrow

- An arrow that runs between an activity and the diagram border. Border arrows are also represented in the decomposition diagram or parent diagram.

Round Tunnel

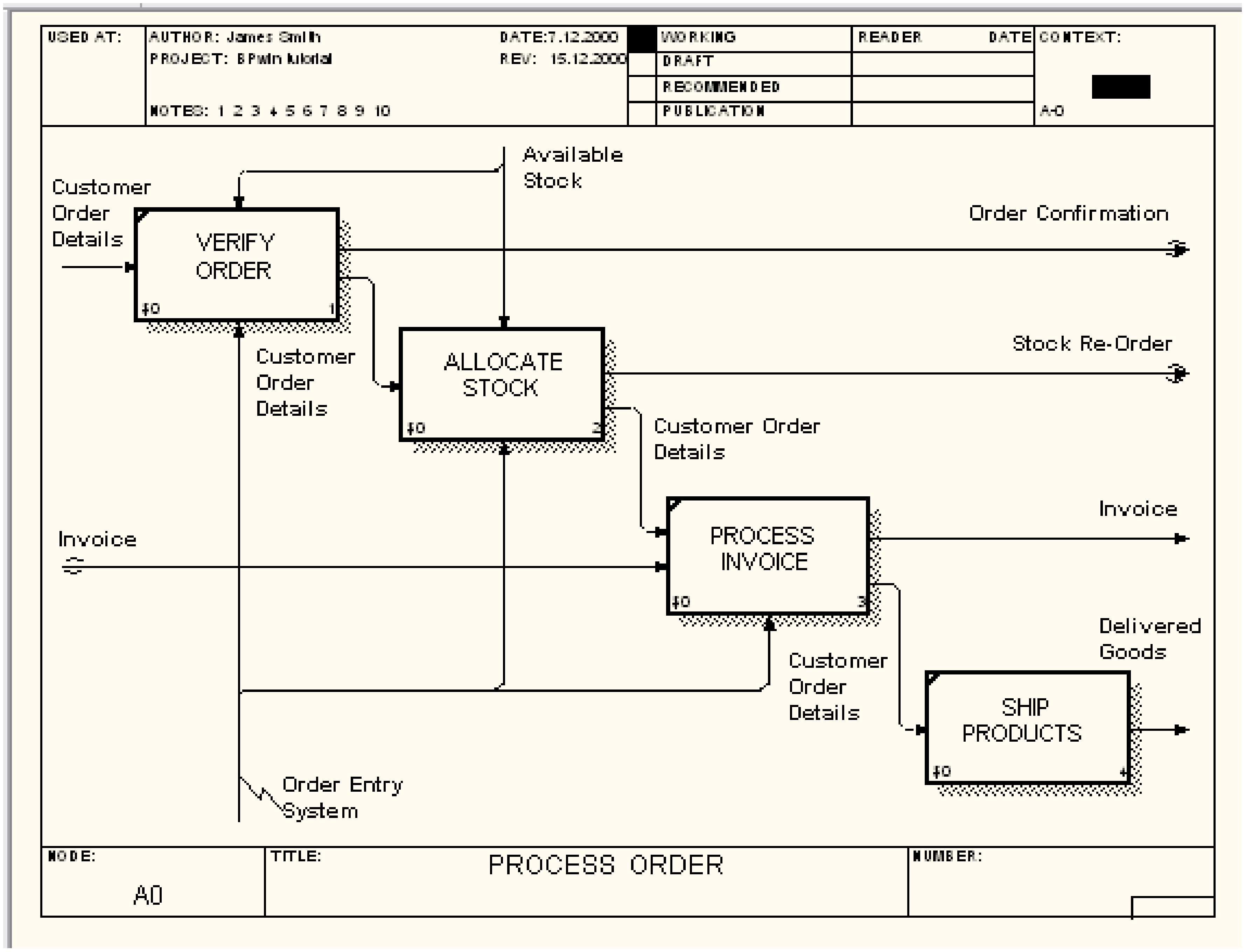
- A symbol that indicates your intention to leave the arrow unresolved in the model hierarchy. Round tunnels are usually temporary.

External Reference

- A symbol that represents a location, entity, person, or department that is a source or destination of data but is outside the scope of a diagram. An external reference can be internal to a company, such as “Accounts Payable” or outside it, such as “Vendor” or “Bank.”

Off-page Reference

- A symbol that represents a reference to another diagram in the model. You can right-click the Off-page reference and choose Go To Reference to open the referenced diagram.

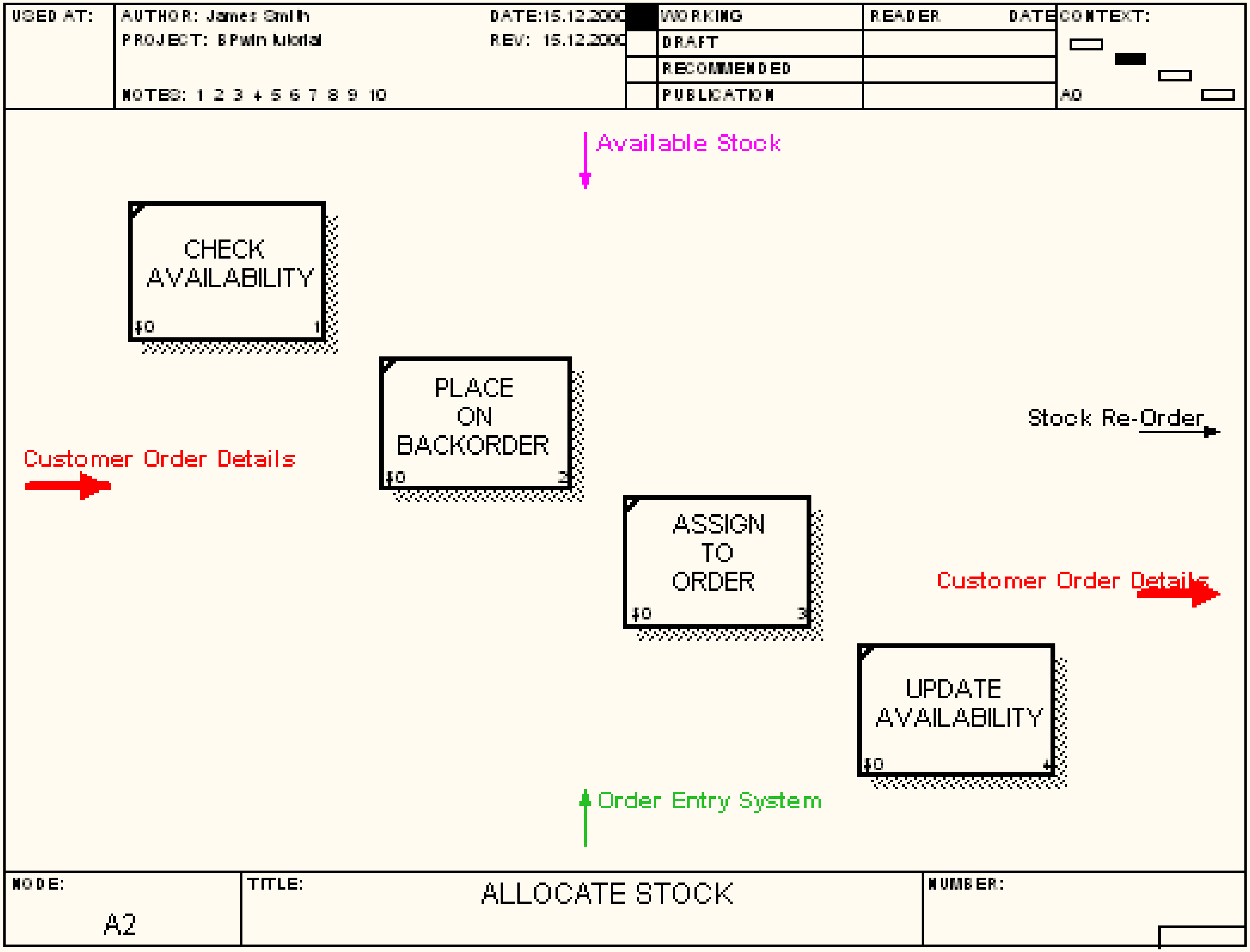


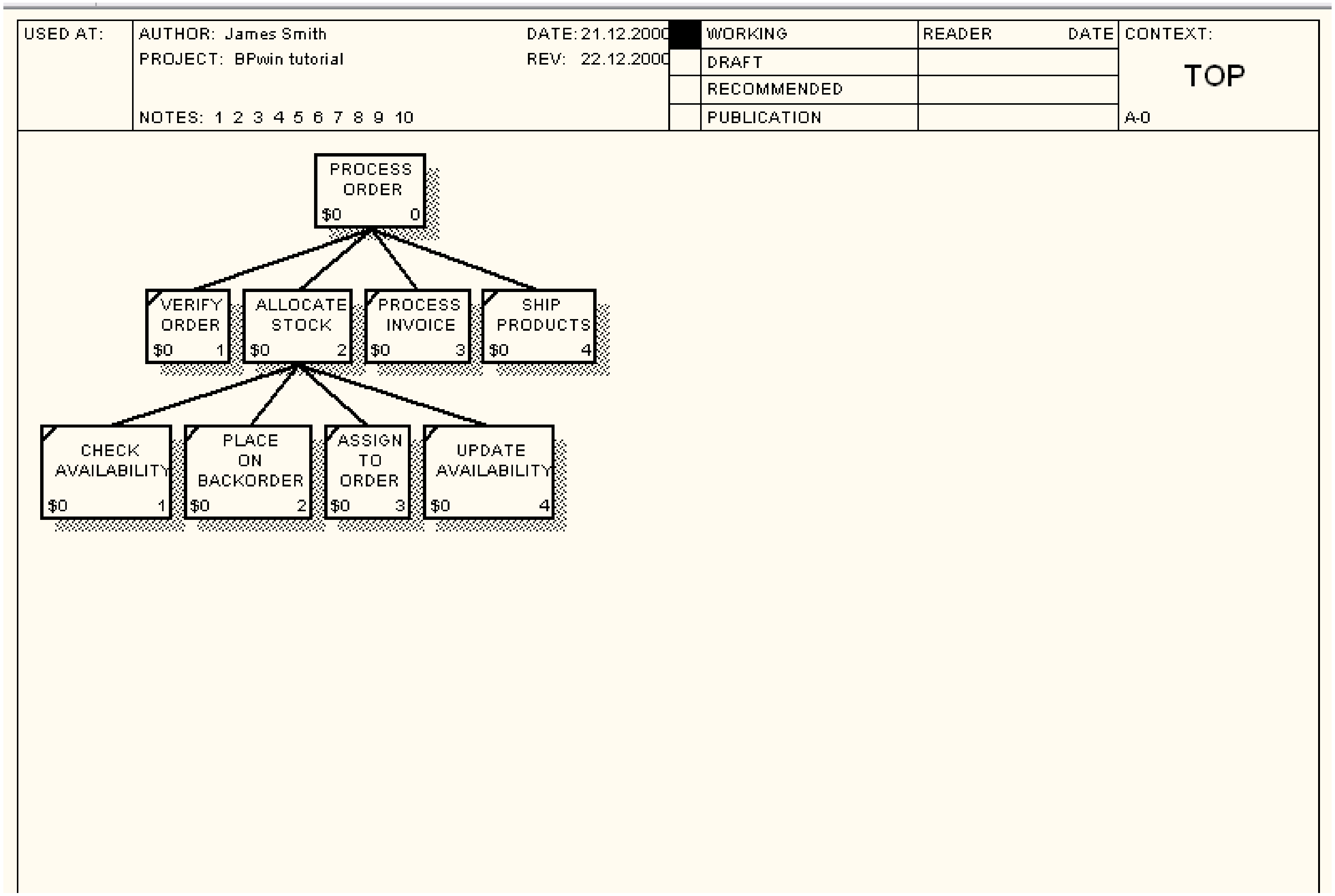
USED AT: | AUTHOR: James Smith | DATE: 7.12.2000
PROJECT: BPwin tutorial | REV: 15.12.2000
NOTES: 1 2 3 4 5 6 7 8 9 10

WORKING	READER	DATE
DRAFT		
RECOMMENDED		
PUBLICATION		

CONTEXT: A-0

NODE: A0 | TITLE: PROCESS ORDER | NUMBER:

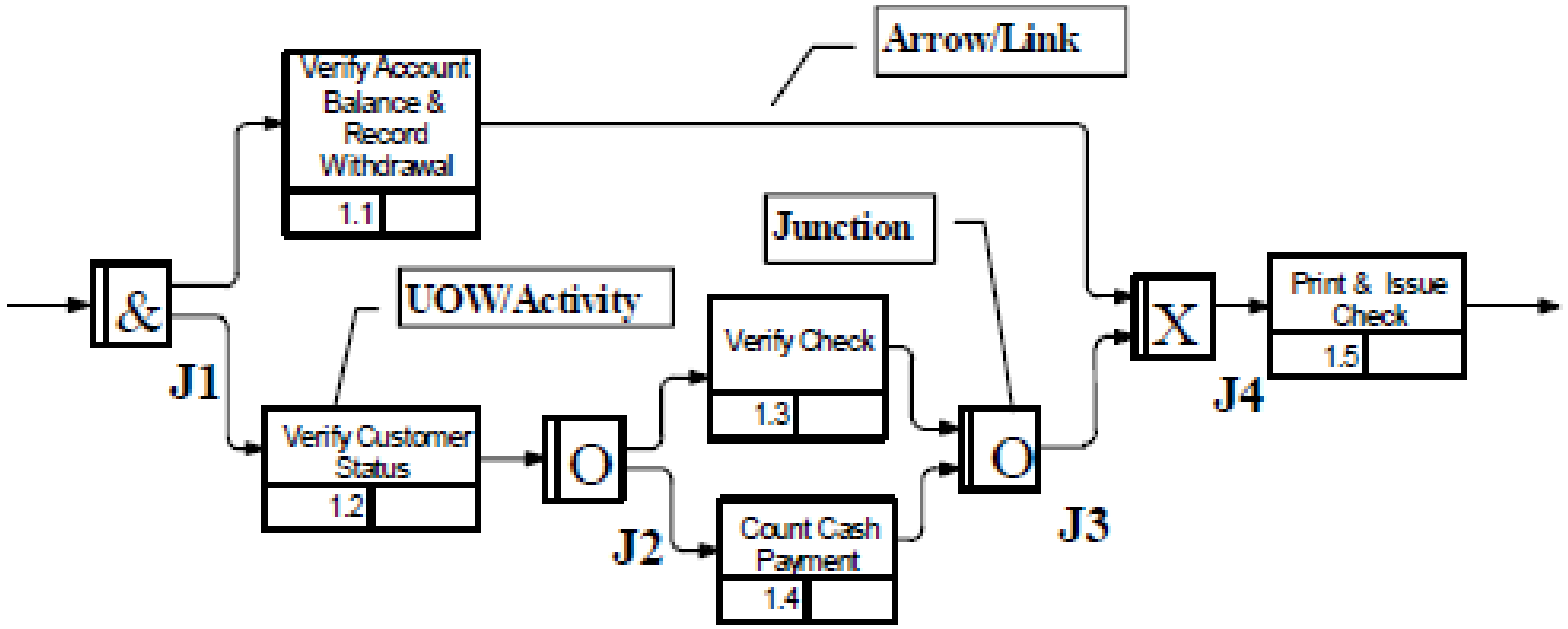

USED AT:
AUTHOR: James Smith
PROJECT: BPwin tutorial
DATE: 15.12.2000
REV: 15.12.2000
NOTES: 1 2 3 4 5 6 7 8 9 10
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
Available Stock
CHECK AVAILABILITY
0
1
PLACE ON BACKORDER
0
2
ASSIGN TO ORDER
0
3
UPDATE AVAILABILITY
0
4
Customer Order Details
Stock Re-Order
Customer Order Details
Order Entry System
NODE:
A2
TITLE:
ALLOCATE STOCK
NUMBER:



USED AT:	AUTHOR: James Smith PROJECT: BPwin tutorial NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 21.12.2000 REV: 22.12.2000	■	WORKING	READER DATE	CONTEXT:
				DRAFT		TOP
				RECOMMENDED		
				PUBLICATION		A-0

IDEF3

- IDEF3 is a process description capture method whose primary goal is to provide a structured method by which a domain expert can describe a situation as an ordered sequence of events, as well as describe any participating objects.



Issue Cashier's Check

Data flow diagrams (DFDs)

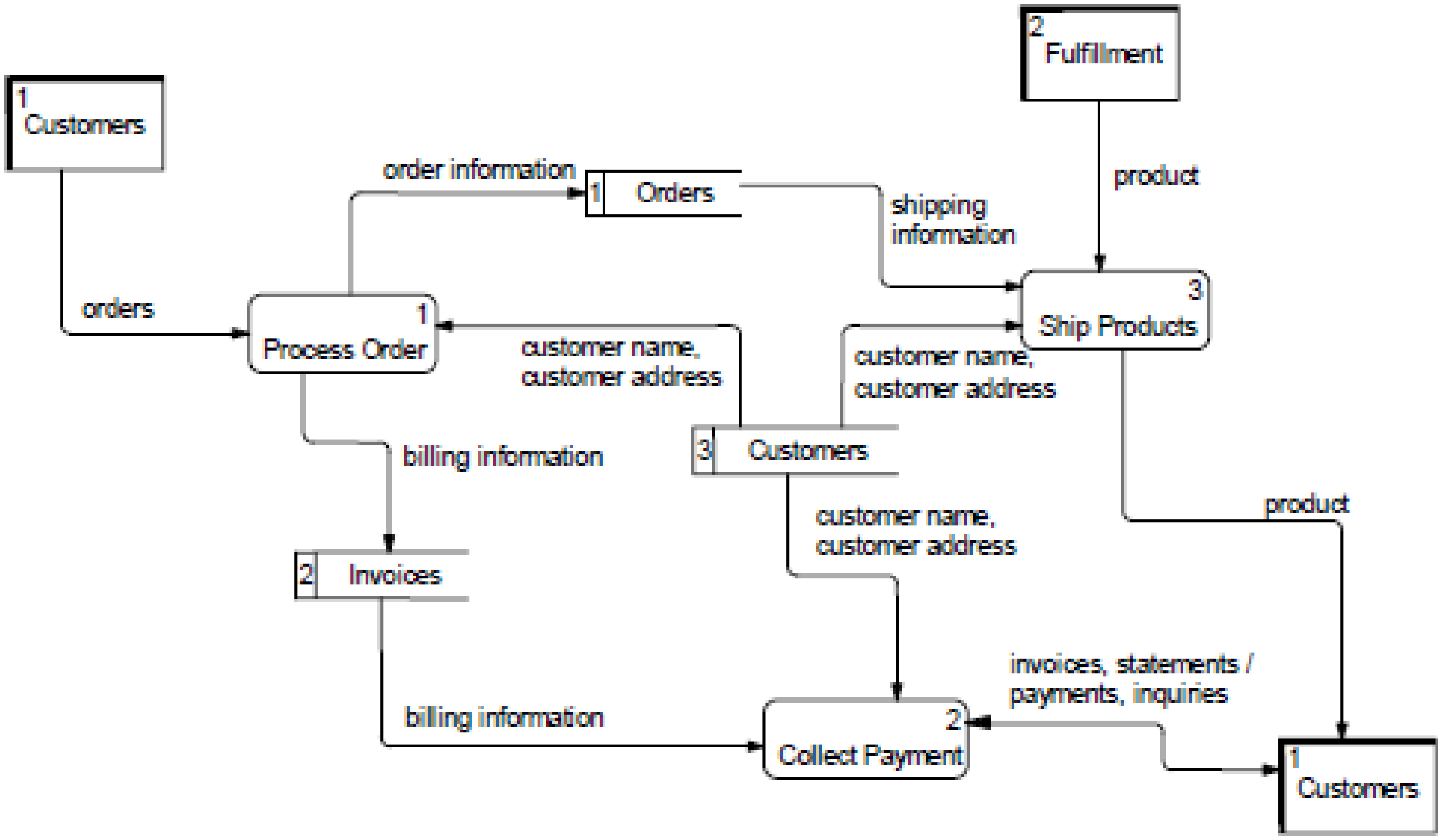
- DFDs model systems as a network of activities connected to one another by pipelines of objects.

- Data flow diagrams also model holding tanks called data stores, and external entities which represent interfaces with objects outside the bounds of the system being modeled.

- In contrast to IDEF0 arrows, which represent constraining relationships, arrows in DFDs show how objects (including data) actually flow, or move, from one activity to another.

- This flow representation, combined with data stores and external entities, gives DFD models more resemblance to some of the physical characteristics of a system.

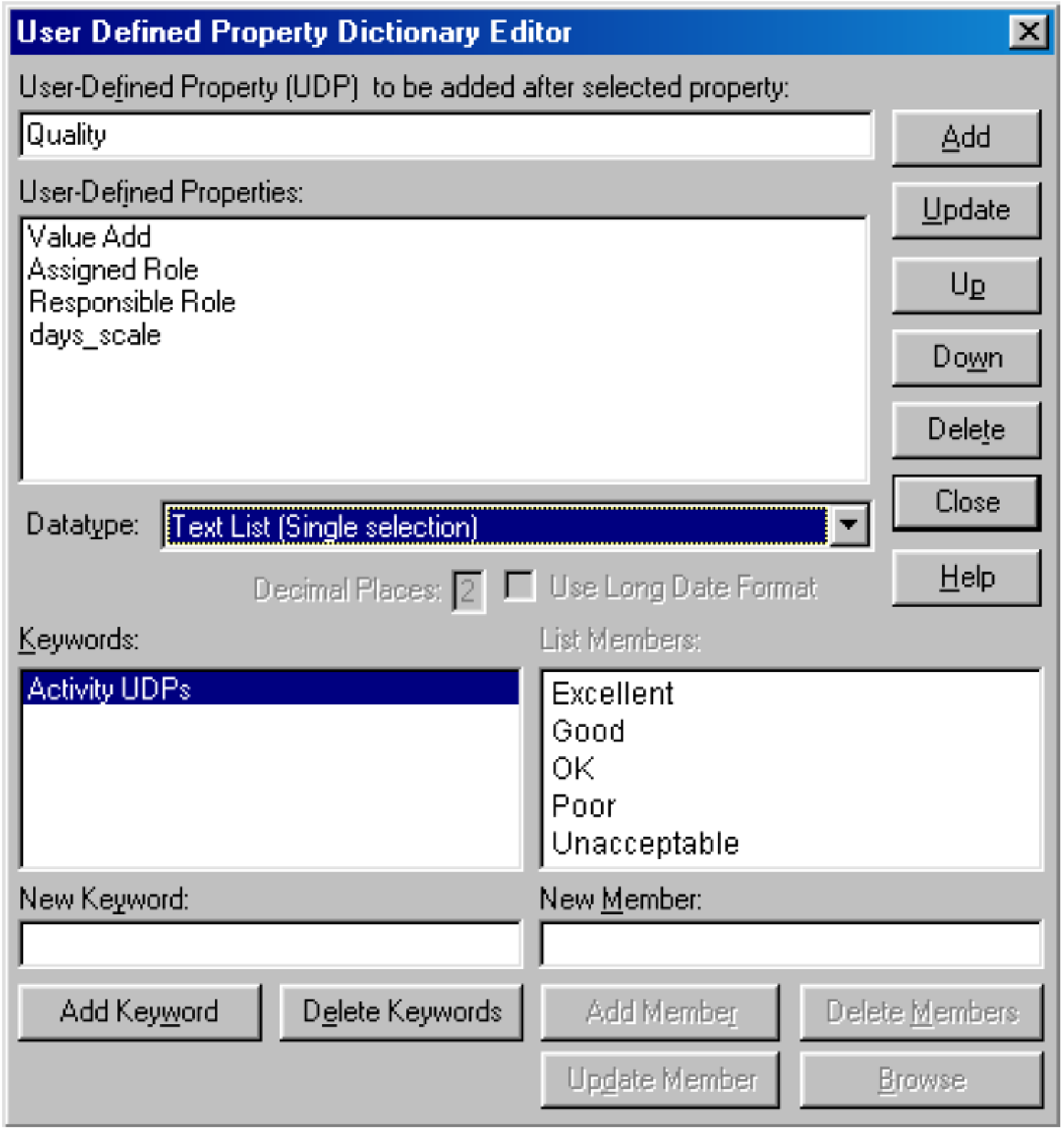
- Data flow diagramming is mostly associated with the development of software applications because it originated for that purpose.


1 Customers
orders
1 Process Order
order information
1 Orders
shipping information
2 Fulfillment
product
3 Ship Products
customer name, customer address
customer name, customer address
3 Customers
billing information
2 Invoices
customer name, customer address
product
billing information
2 Collect Payment
invoices, statements / payments, inquiries
1 Customers

UDP

- In BPwin, you can create UDPs to associate business-specific information with a diagram object such as an activity or arrow.
- Bpwin supports various types of UDPs, including pull-down lists, command UDPs, and text lists.
- The first step to creating User Defined Properties (UDPs) is to apply UDP values to diagram objects such as activities and arrows.

For example, you can create a text UDP called EMPLOYEES to list the names of employees who work in departments represented by diagram activities. You can create UDPs that use different datatypes such as text boxes, multi-select lists, and commands that run other Windows applications.


User Defined Property Dictionary Editor

User-Defined Property (UDP) to be added after selected property: Quality

User-Defined Properties:
Value Add
Assigned Role
Responsible Role
days_scale

Datatype: Text List (Single selection)

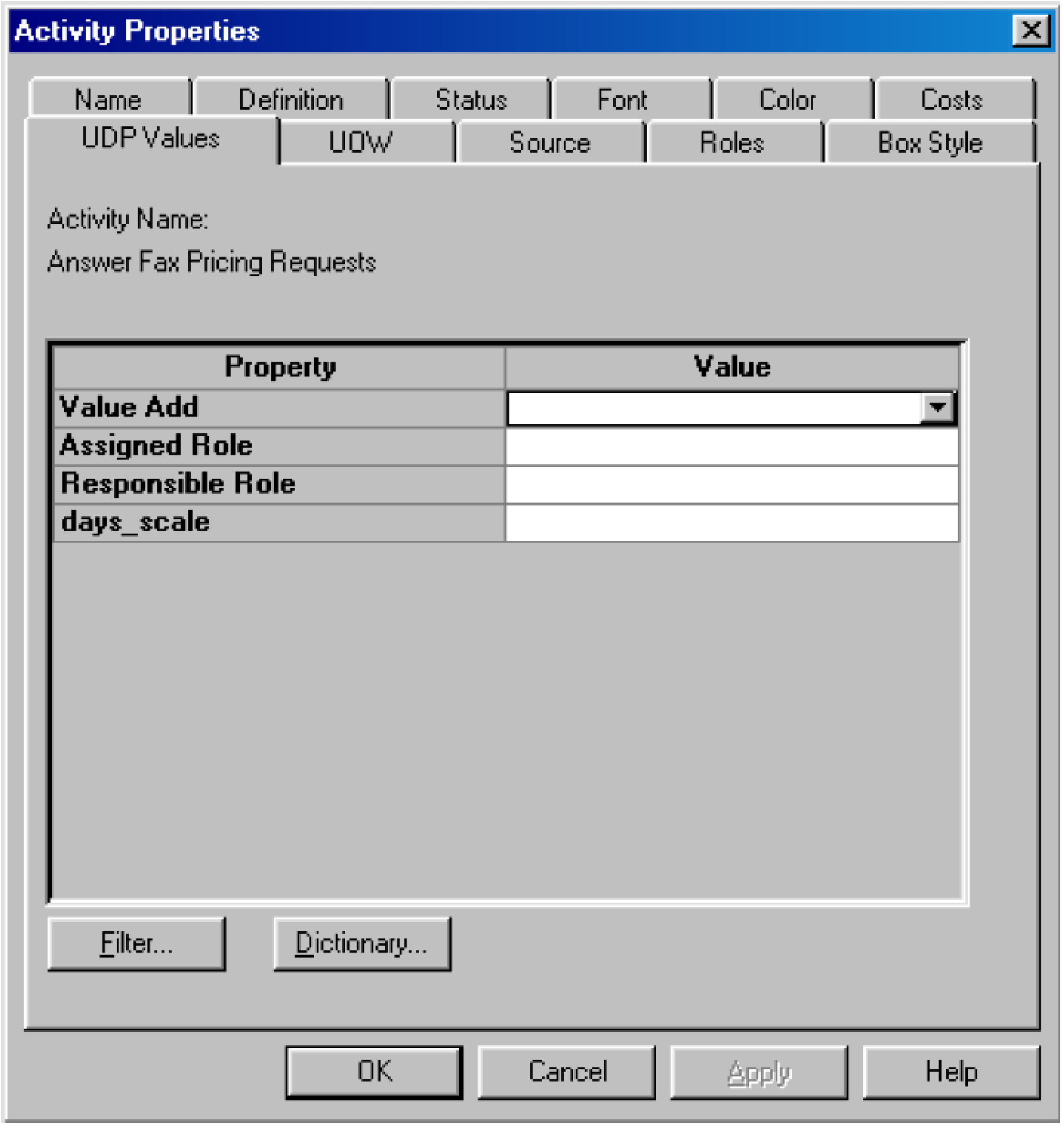
Decimal Places: 2 ☐ Use Long Date Format

Keywords:
Activity UDPs

List Members:
Excellent
Good
OK
Poor
Unacceptable

New Keyword:

New Member:



Activity Name:

Answer Fax Pricing Requests

Property	Value
Value Add	
Assigned Role	
Responsible Role	
days_scale	

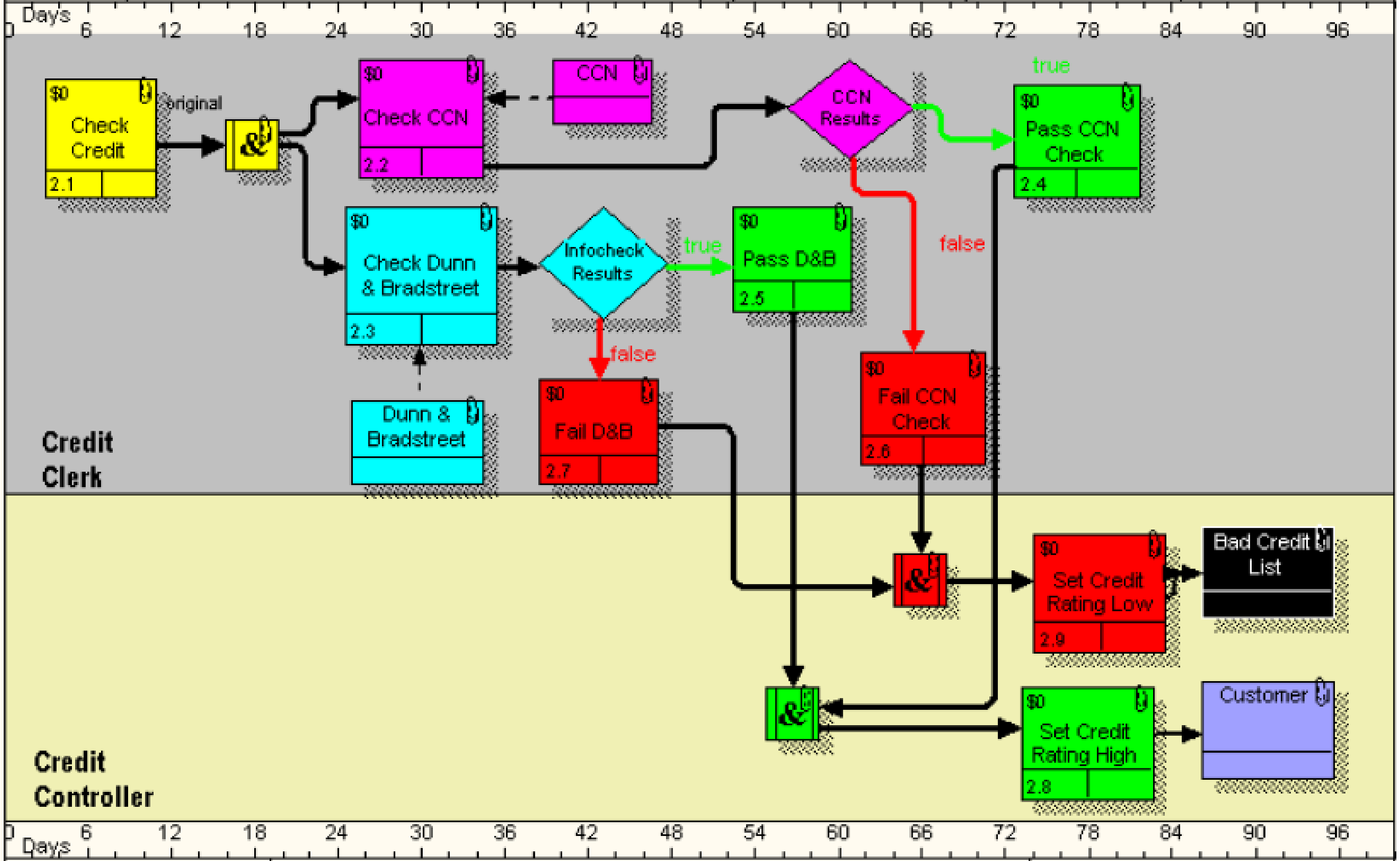
Swim Lane Diagrams

- Swim Lane diagrams can provide your organization with an efficient mechanism for visualizing and optimizing processes.

- Swim Lane diagrams organize complex processes across functional boundaries, and help you to conveniently view processes, roles, and responsibilities, and their flow.

- Swim Lane diagrams use Process Flow Network (IDEF3) methodology, and display graphical horizontal lanes that represent process dependencies called *roles*

USED AT:	AUTHOR: Norm Wold PROJECT: Reengineering Quill Computers, Inc. NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 6/14/2005 REV: 6/14/2005	■ WORKING DRAFT RECOMMENDED PUBLICATION	READER DATE



NODE: A125.2	TITLE: Credit Verification	NUMBER:

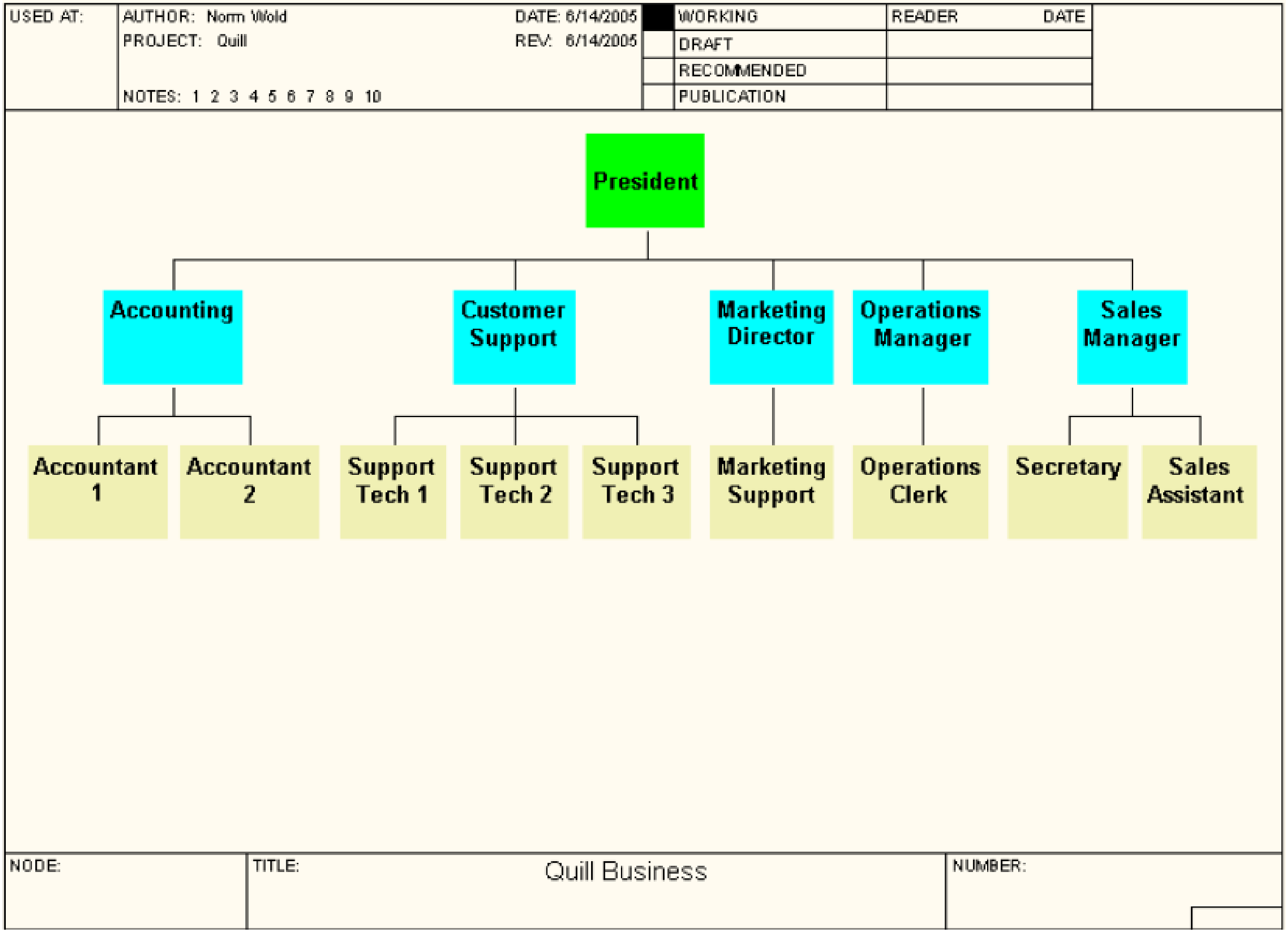
Prerequisites for Adding a Swim Lane Diagram

- Create process roles in the role dictionary.
- Create process roles in the UDP dictionary by adding list items to a text list UDP

How Process Roles are Created in the Role Dictionary

- Creating process roles involves working with the following dictionaries:
  - Role Group Dictionary
  - Role Dictionary
  - Resource Dictionary

Organization Charts



Prerequisites for Creating an Organization Chart

- You must have at least one role group defined in the Role Dictionary.
- The necessary roles must exist in the Role Dictionary and be associated with a role group.
- Any required resources must be added to the Resource Dictionary and associated with roles.

ERWin

Data Modeling

Logical Models

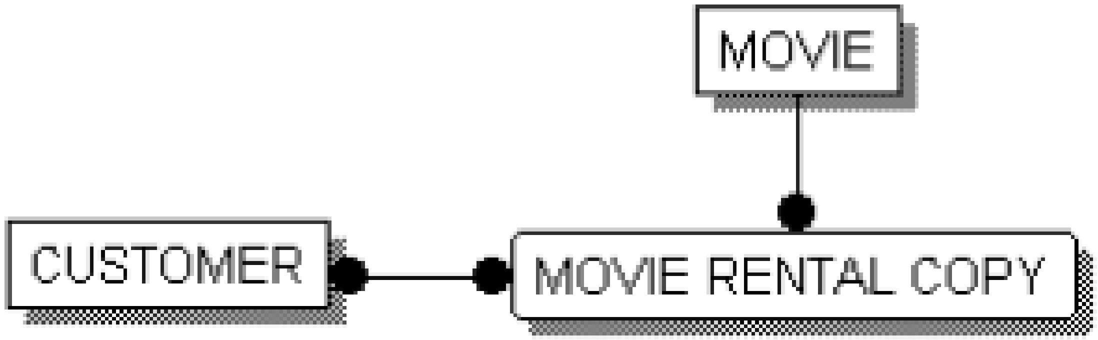
There are three levels of logical models that are used to capture business information requirements:

- the Entity Relationship Diagram,
- the Key-Based Model,
- and the Fully Attributed model.

Entity Relationship Diagram (ERD)

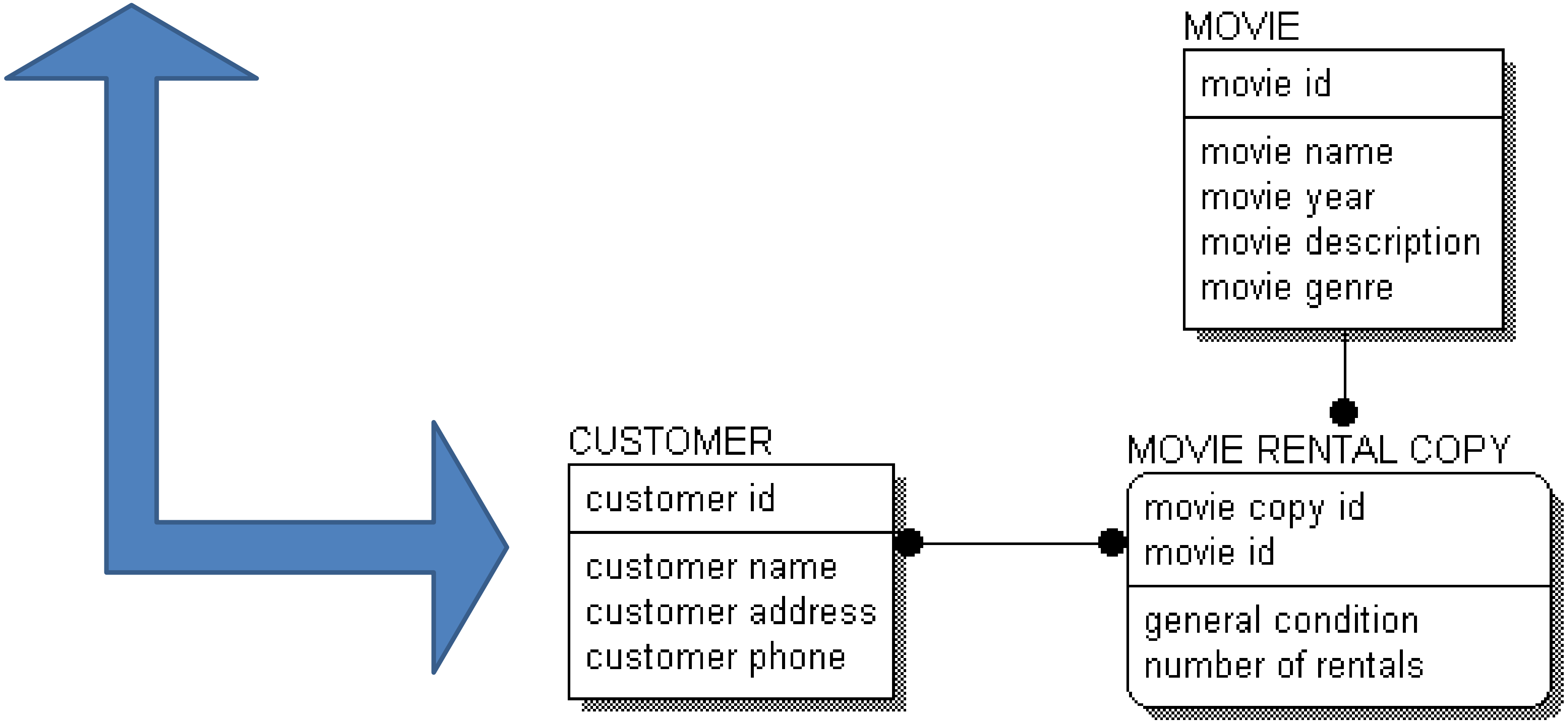
- The Entity Relationship Diagram (ERD) is a high-level data model that shows the major entities and relationships, which support a wide business area. This is primarily a presentation or discussion model.

- An ERD is made up of three main building blocks: entities, attributes, and relationships.
- The logical equivalent to a table is an entity. In an ERD, an entity is represented by a box that contains the name of the entity. Entity names are always singular.
- A relationship is represented by a line drawn between the entities.



Like tables have columns, entities have attributes.

Customer_id	Customer Name	Customer address	Customer phone
101	James Bond	London, UK	700-007
102	Clark Kent	New York, USA	7873-1626

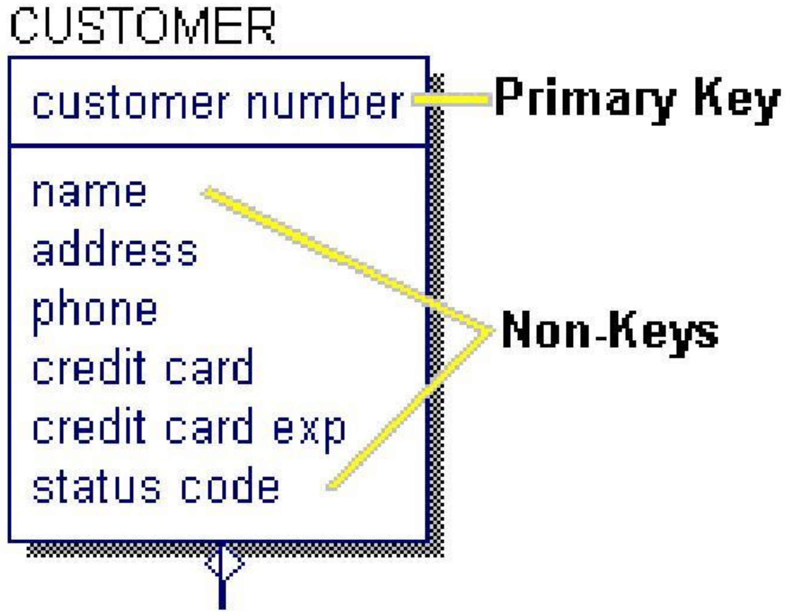


In each entity in a data model, a horizontal line separates the attributes into two groups, key areas and non-key areas.

The key area contains the primary key for the entity.

The primary key is a set of attributes used to identify unique instances of an entity.

The primary key may be comprised of one or more primary key attributes, as long as the chosen attributes form a unique identifier for each instance in an entity.



Logical Relationships

- Relationships represent connections, links, or associations between entities. They are the *verbs* of a diagram that show how entities relate to each other.

■ A TEAM <has> many PLAYERs.

■ A PLANE-FLIGHT <transports> many PASSENGERs.

■ A DOUBLES-TENNIS-MATCH <requires> exactly 4 PLAYERs.

■ A HOUSE <is owned by> one or more OWNERs.

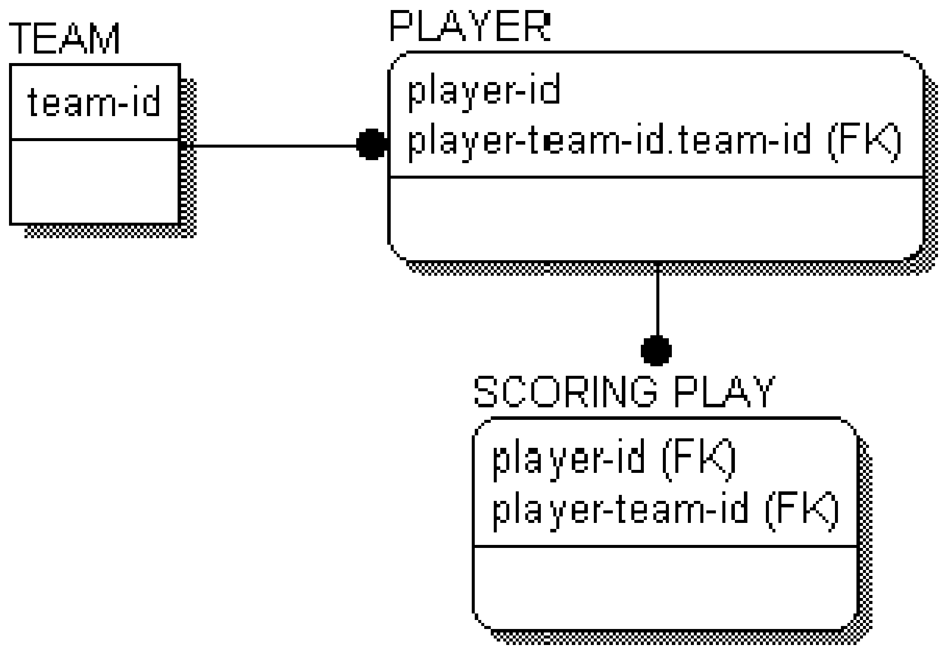
■ A SALESPERSON <sells> many PRODUCTs.

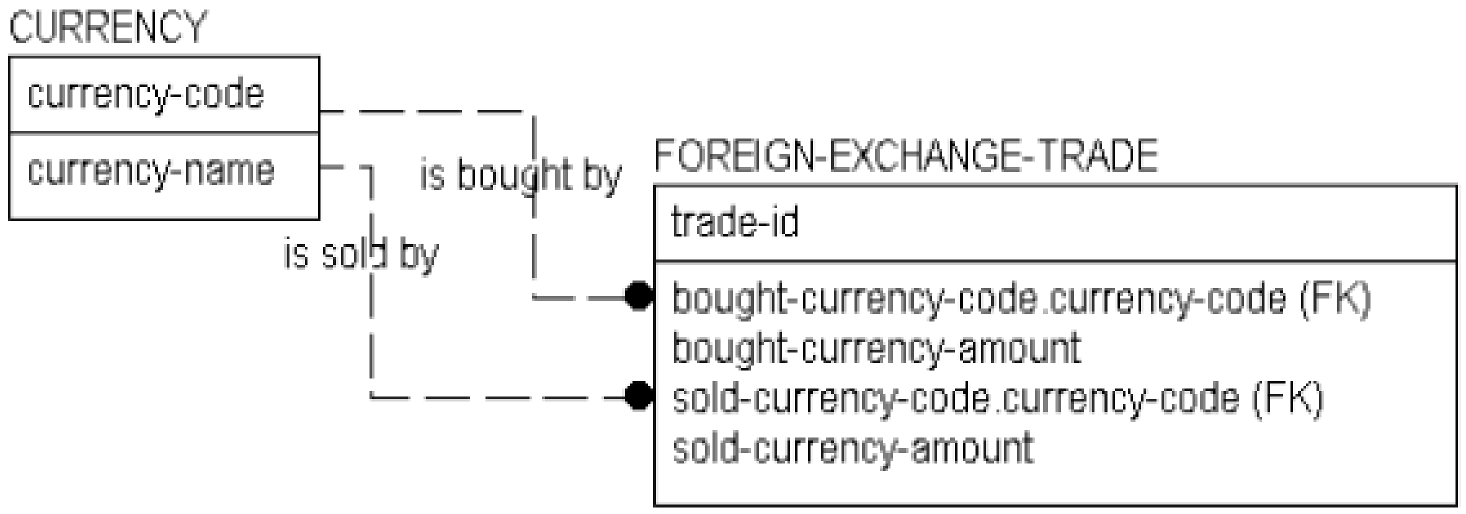
Foreign Key

- A foreign key is the set of attributes that define the primary key in the parent entity and that migrate through a relationship from the parent to the child entity. In a data model, a foreign key is designated by the symbol (FK) after the attribute name.
- When you create a relationship between entities, ERwin automatically migrates the primary key attributes of the parent entity to the child entity.

Rolenames

- When foreign keys migrate from the parent entity in a relationship to the child entity, they are serving double-duty in the model in terms of stated business rules. To understand both roles, it is sometimes helpful to rename the migrated key to show the role it plays in the child entity.



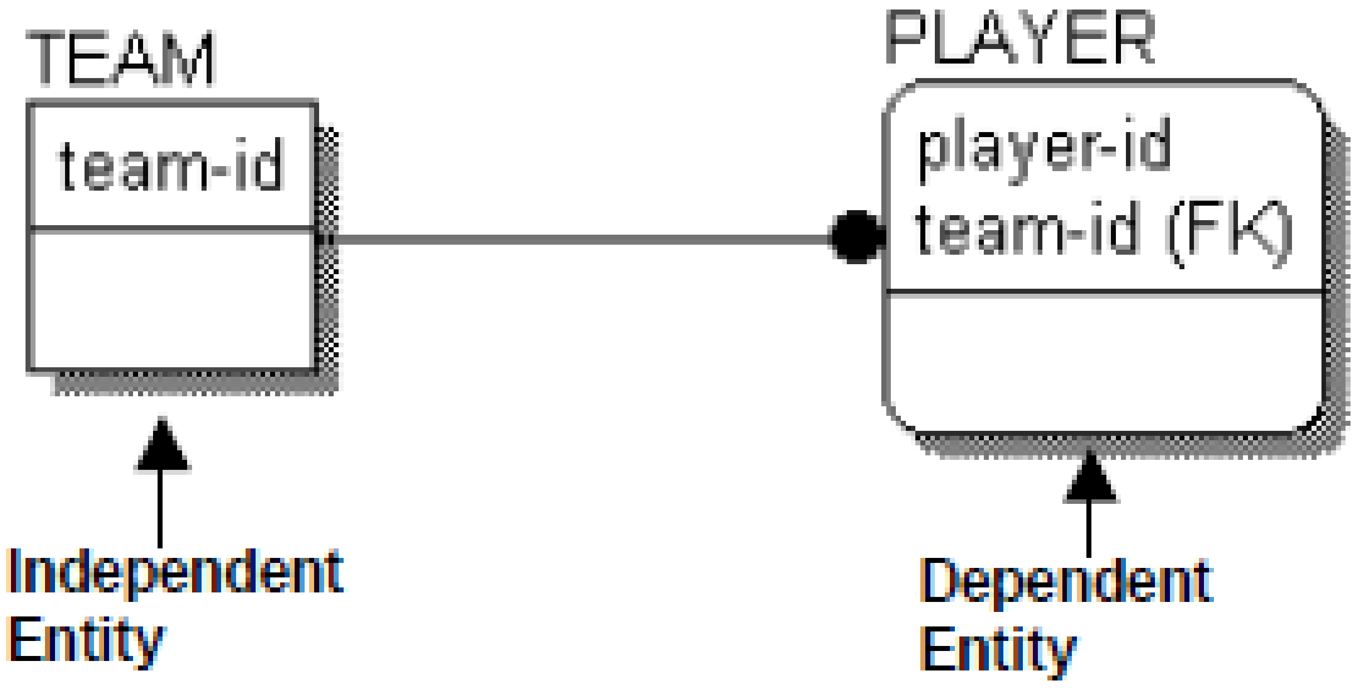


Attribute/Rolename	**Attribute Definition**
currency-code	The unique identifier of a CURRENCY.
bought-currency-code	The identifier ("currency-code") of the CURRENCY bought by (purchased by) the FOREIGN-EXCHANGE-TRADE.
sold-currency-code	The identifier ("currency-code") of the CURRENCY sold by the FOREIGN-EXCHANGE-TRADE.

Types of Entities

- An *independent entity* is an entity whose instances can be uniquely identified without determining its relationship to another entity. It is represented as a box with square corners.

- A *dependent entity* is an entity whose instances cannot be uniquely identified without determining its relationship to another entity or entities. It is represented as a box with rounded corners.

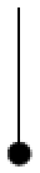
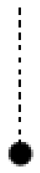
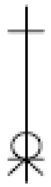
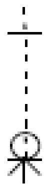
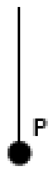
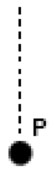
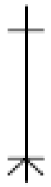
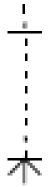
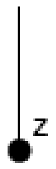
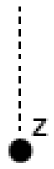
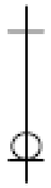
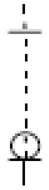
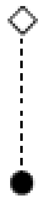
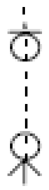
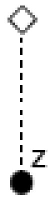
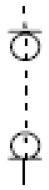

TEAM
team-id
PLAYER
player-id
team-id (FK)
Independent Entity
Dependent Entity

Types of Relationships

The type of relationship determines how a primary key of the parent entity migrates to the child entity as a foreign key. There are two basic types of relationships:

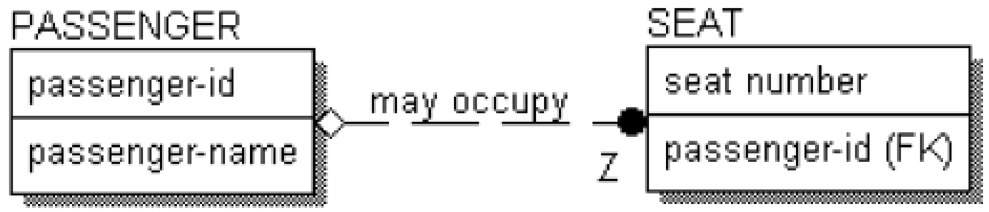
- An *identifying relationship* is represented by a solid line and through it the primary key of the parent migrates to the primary key area of the child entity and becomes part of the primary key of the child entity.

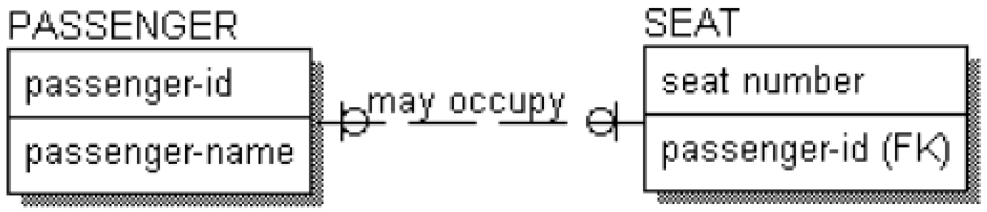
- A *non-identifying relationship* is represented by a dashed line and through it the primary key of the parent migrates to the non-key area of the child entity and becomes a non-key attribute in the child entity.

Cardinality Description	IDEF1X Notation		IE Notation	
	Identifying	Non-identifying	Identifying	Non-identifying
One to zero, one, or more				
One to one or more				
One to zero or one				
Zero or one to zero, one, or more (non-identifying only)				
Zero or one to zero or one (non-identifying only)				

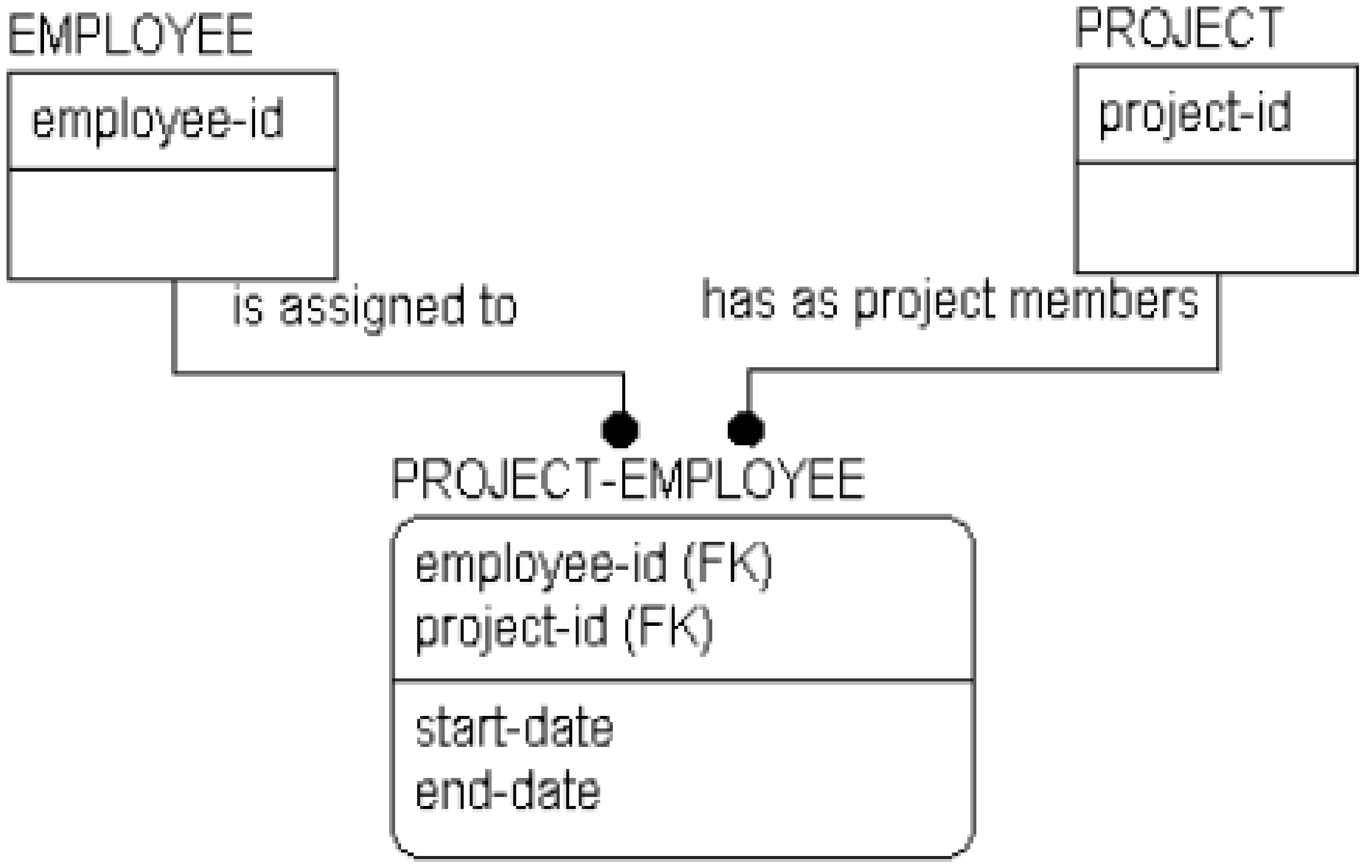
The child is neither existence nor identification-dependent with respect to that relationship

The attribute “passenger-id” is a foreign key attribute of SEAT. Because the “passenger-id” does not identify the SEAT, it identifies the PASSENGER occupying the SEAT, the relationship is non-identifying. The SEAT can exist without any PASSENGER, so the relationship is optional. When a relationship is optional, the diagram includes either a diamond in IDEF1X or a circle in IE notation.





The cardinality for the relationship, indicated here with a Z in IDEF1X and a single line in IE, states that a PASSENGER <may occupy> zero or one of these SEATs on a flight. Each SEAT can be occupied, in which case the PASSENGER occupying the seat is identified by the “passenger-id,” or it can be unoccupied, in which case the “passenger-id” attribute is empty (NULL).



What would happen if you were to delete an instance of PROJECT?

Referential Integrity

- Cascade—Each time an instance in the parent entity is deleted, each related instance in the child entity must also be deleted.
- Restrict—Deletion of an instance in the parent entity is prohibited if there are one or more related instances in the child entity, or deletion of an instance in the child entity is prohibited if there is a related instance in the parent entity.
- Set Null—Each time an instance in the parent entity is deleted, the foreign key attributes in each related instance in the child entity are set to NULL.
- Set Default—Each time an instance in the parent entity is deleted, the foreign key attributes in each related instance in the child entity are set to the specified default value.
- <None>—No referential integrity action is required. Not every action must have a referential integrity rule associated with it.

In each relationship there are six possible actions for which referential integrity can be defined:

- PARENT INSERT
- PARENT UPDATE
- PARENT DELETE
- CHILD INSERT
- CHILD UPDATE
- CHILD DELETE

Reading Referential Integrity Options

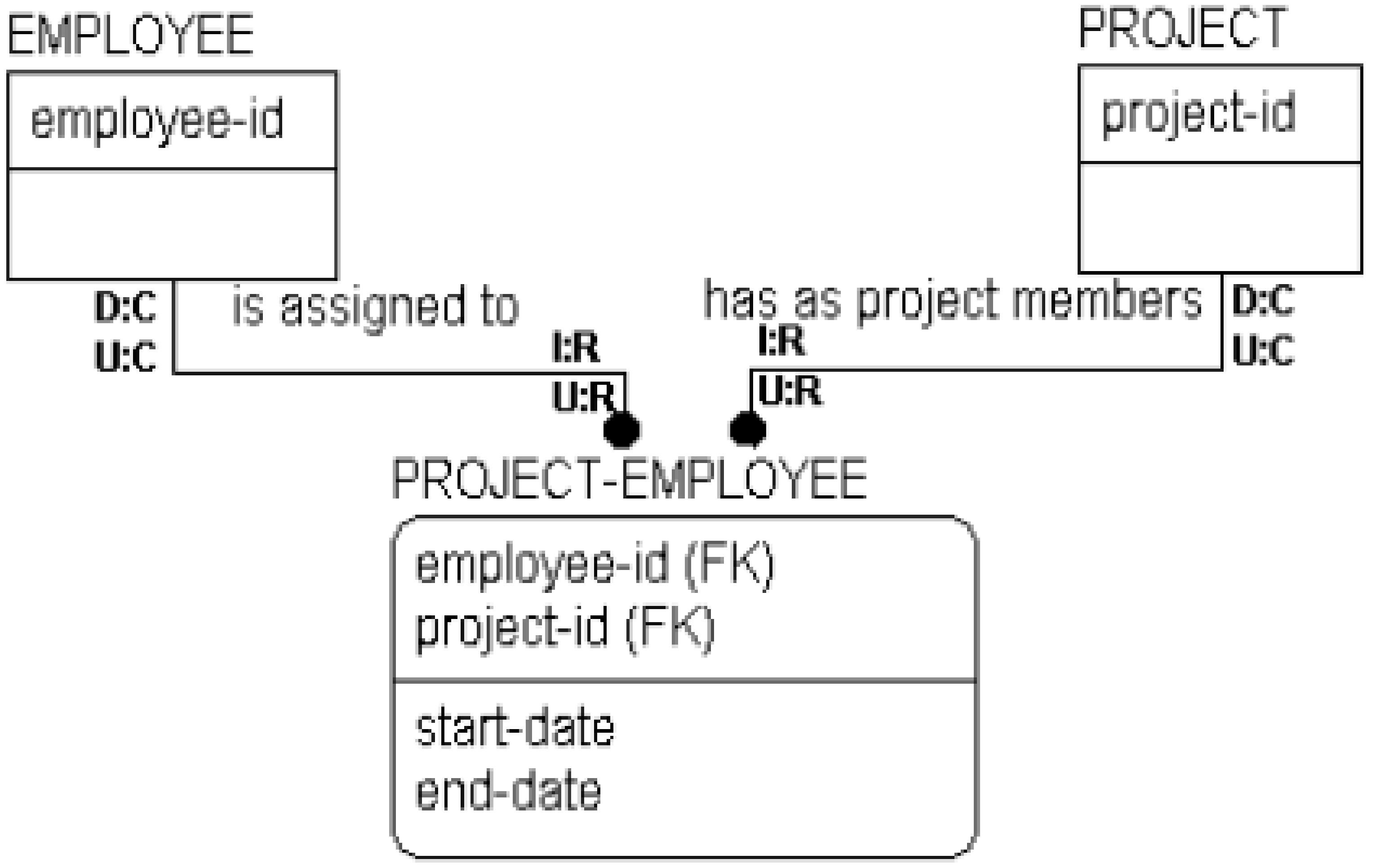
- The first letter in the referential integrity symbol always refers to the database action:

  I(Insert), U(Update), or D(Delete).

- The second letter refers to the referential integrity option:

  C(Cascade), R(Restrict), SN(Set Null),

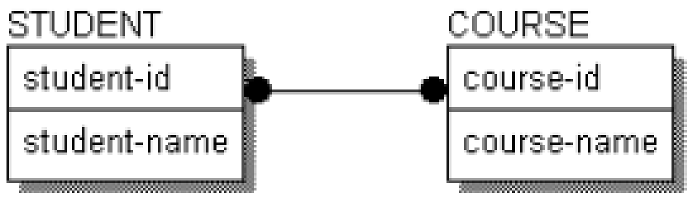
  and SD(Set Default).

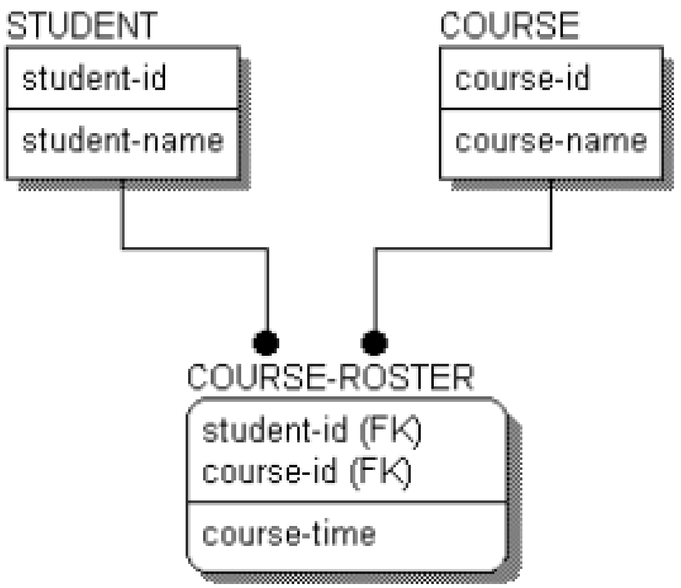

EMPLOYEE
employee-id
PROJECT
project-id
D:C
U:C
is assigned to
I:R
U:R
has as project members
I:R
U:R
D:C
U:C
PROJECT-EMPLOYEE
employee-id (FK)
project-id (FK)
start-date
end-date

Additional Relationship Types

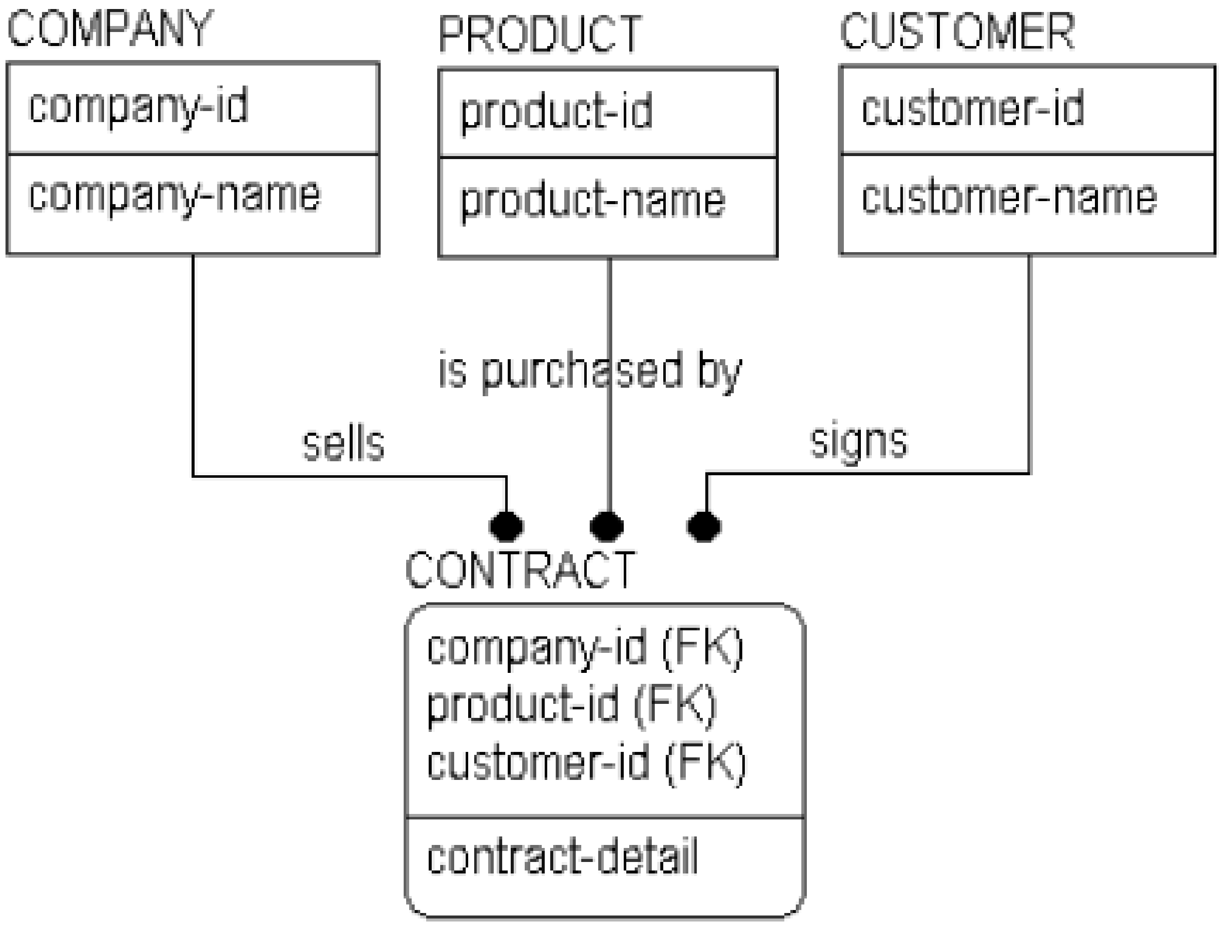
- Many-to-many relationships—A relationship where one entity <owns> many instances of a second entity, and the second entity also <owns> many instances of the first entity.
- N-ary relationships—A simple one-to-many relationship between two entities is termed binary. When a one-to-many relationship exists between two or more parents and a single child entity, it is termed an *n-ary relationship.*
- Recursive relationships—Entities that have a relationship to themselves take part in recursive relationships. This type of relationship is used for bild-of-materials structures, to show relationships between parts.
- Subtype relationships—Related entities are grouped together so that all common attributes appear in a single entity, but all attributes that are not in common appear in separate, related entities.

Many-to-many

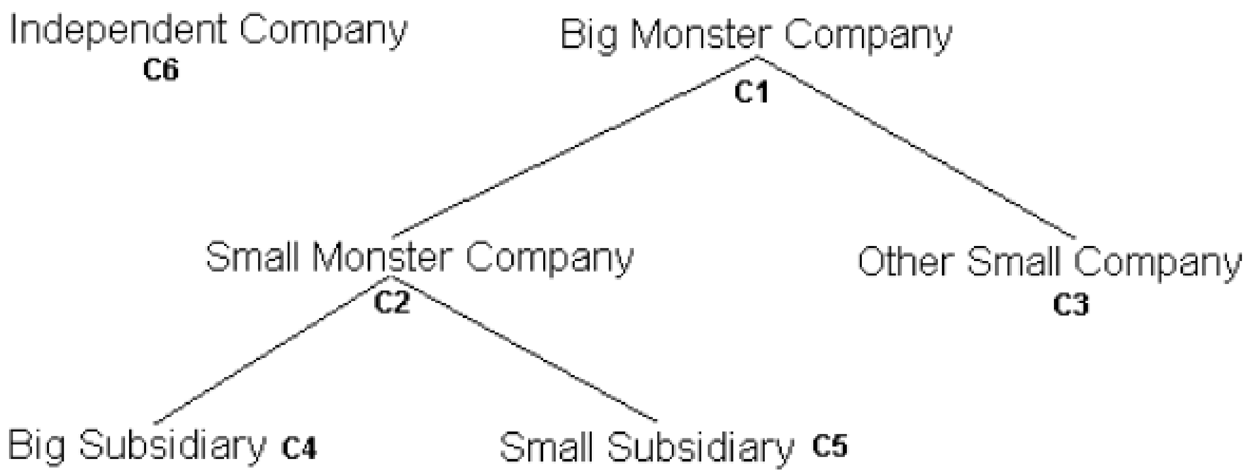

STUDENT
student-id
student-name
COURSE
course-id
course-name

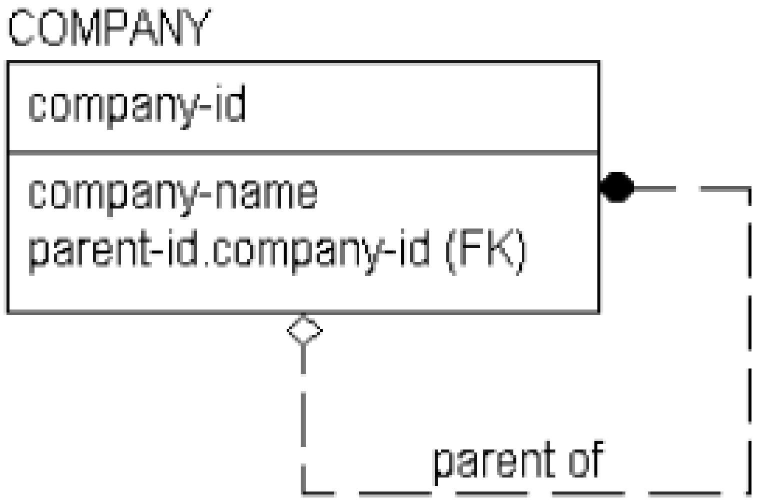

STUDENT
student-id
student-name
COURSE
course-id
course-name
COURSE-ROSTER
student-id (FK)
course-id (FK)
course-time

N-ary Relationships



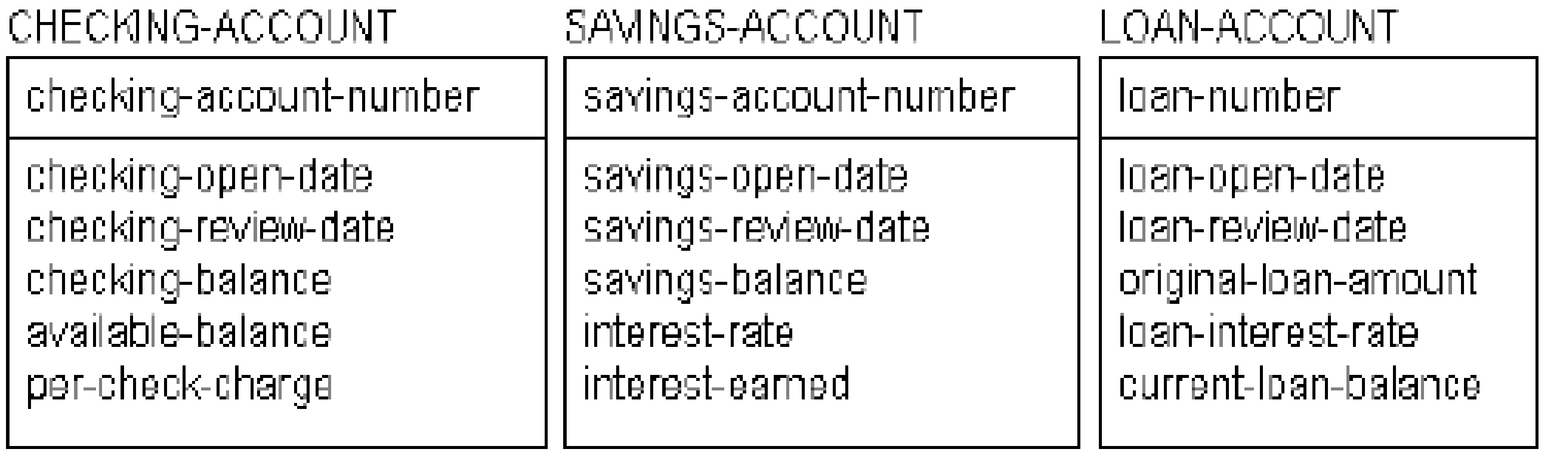
Recursive Relationship

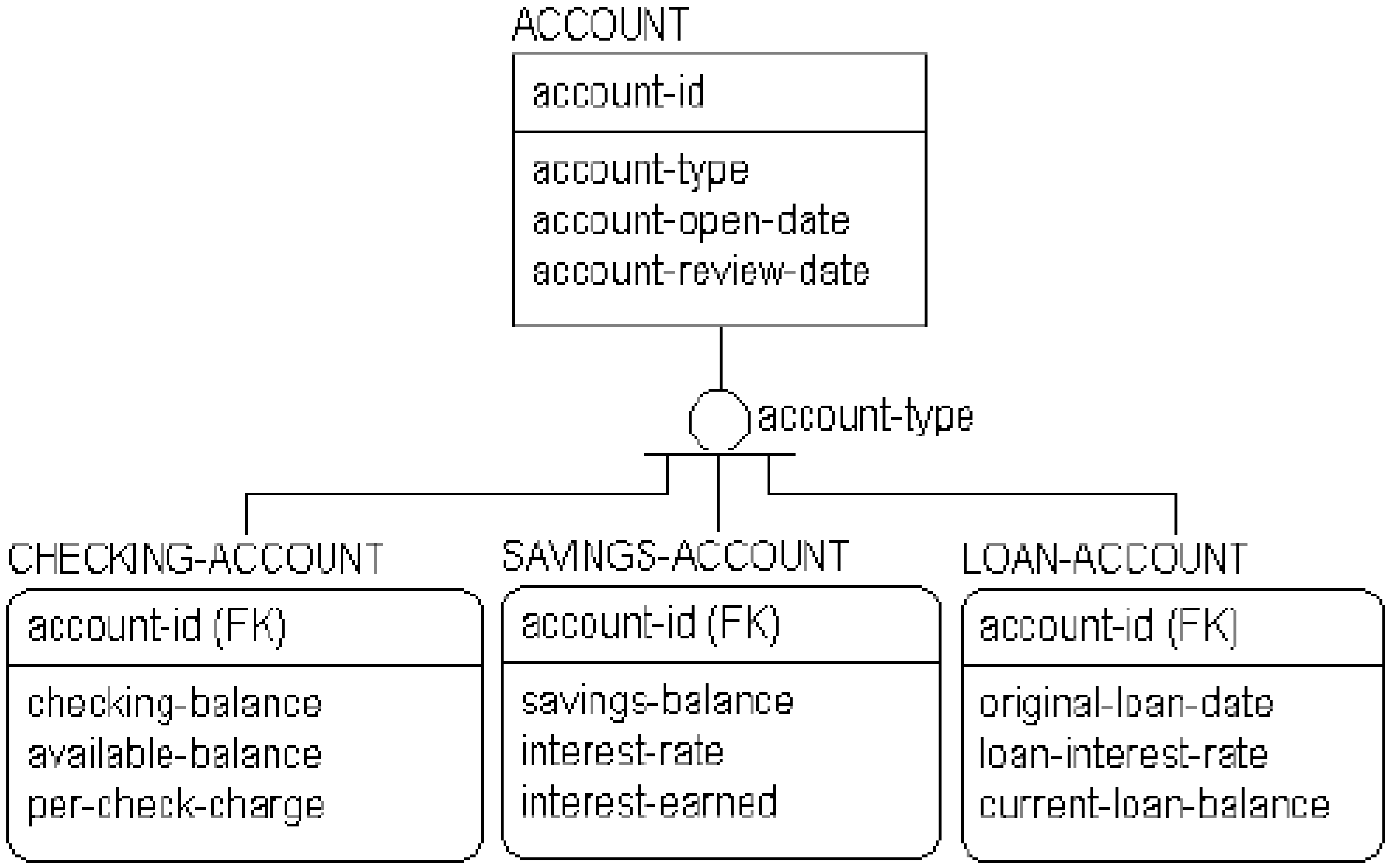




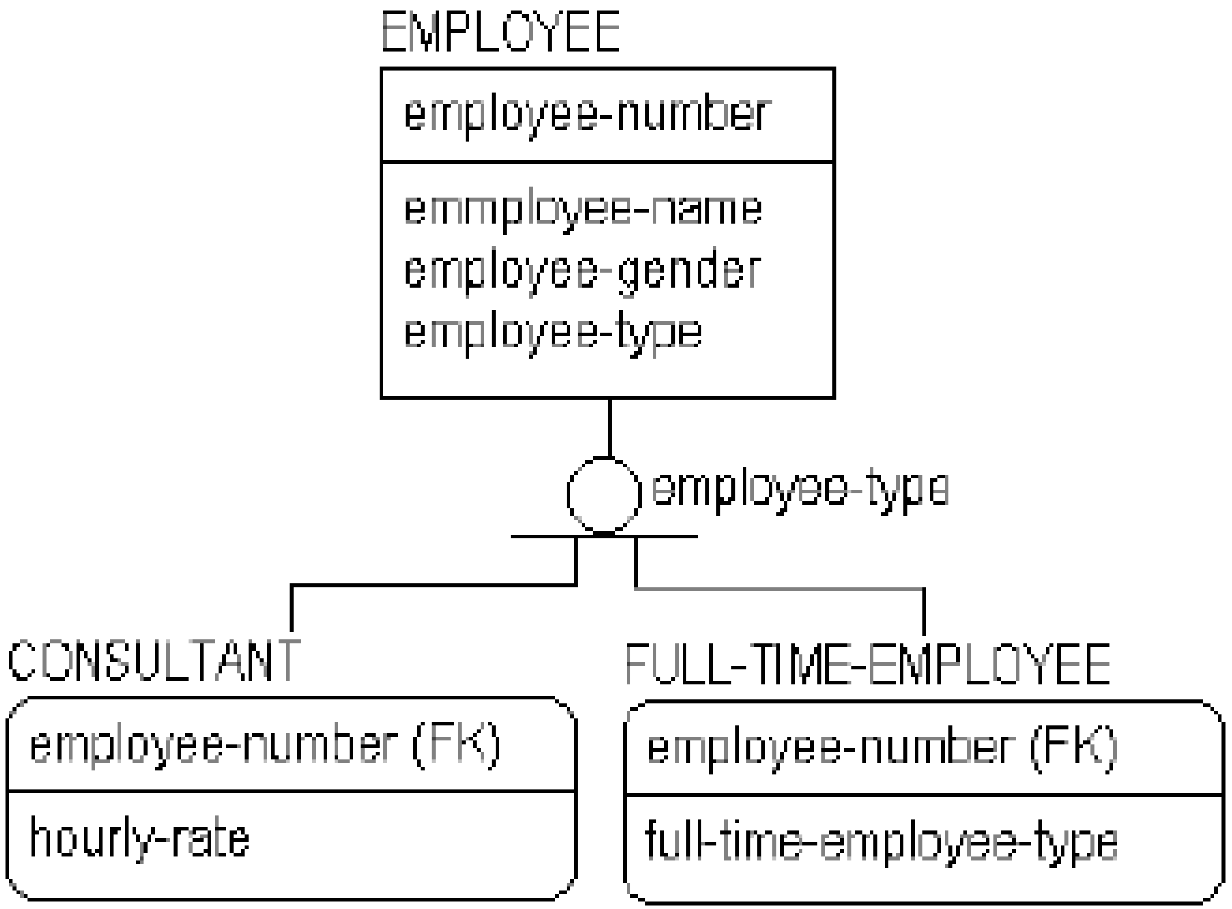
Subtype Relationships

- A subtype relationship, also referred to as a generalization category, generalization hierarchy, or inheritance hierarchy, is a way to group a set of entities that share common characteristics.

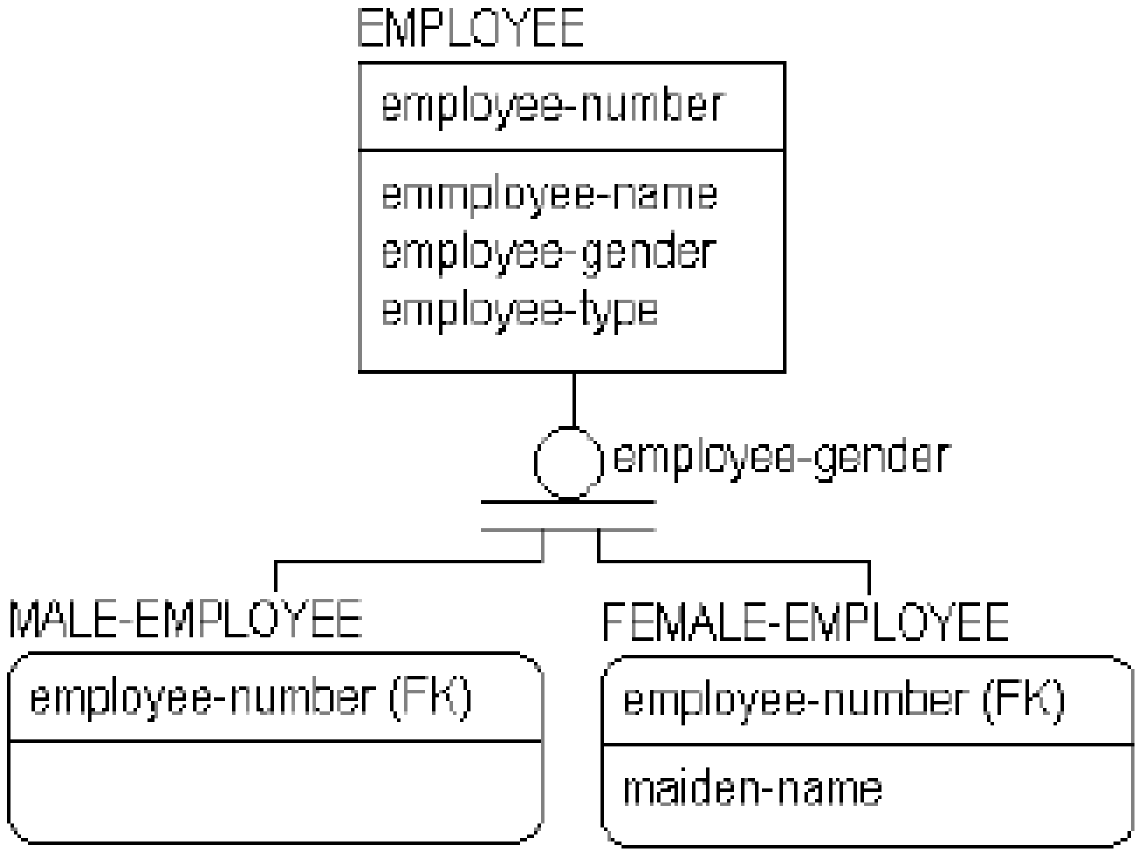



ACCOUNT
account-id
account-type
account-open-date
account-review-date
account-type
CHECKING-ACCOUNT
account-id (FK)
checking-balance
available-balance
per-check-charge
SAVINGS-ACCOUNT
account-id (FK)
savings-balance
interest-rate
interest-earned
LOAN-ACCOUNT
account-id (FK)
original-loan-date
loan-interest-rate
current-loan-balance

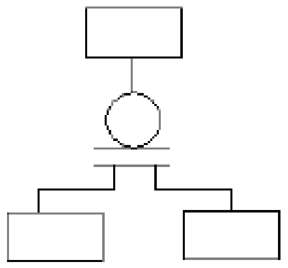
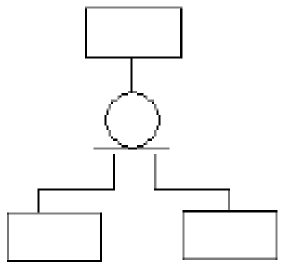
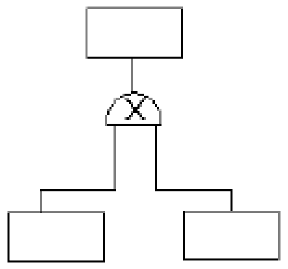
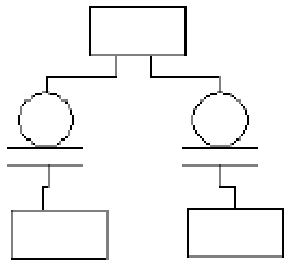
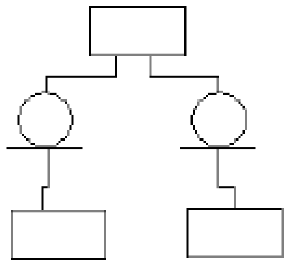
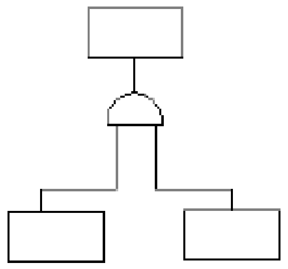
Incomplete


EMPLOYEE
employee-number
emmployee-name
employee-gender
employee-type
employee-type
CONSULTANT
employee-number (FK)
hourly-rate
FULL-TIME-EMPLOYEE
employee-number (FK)
full-time-employee-type

Complete

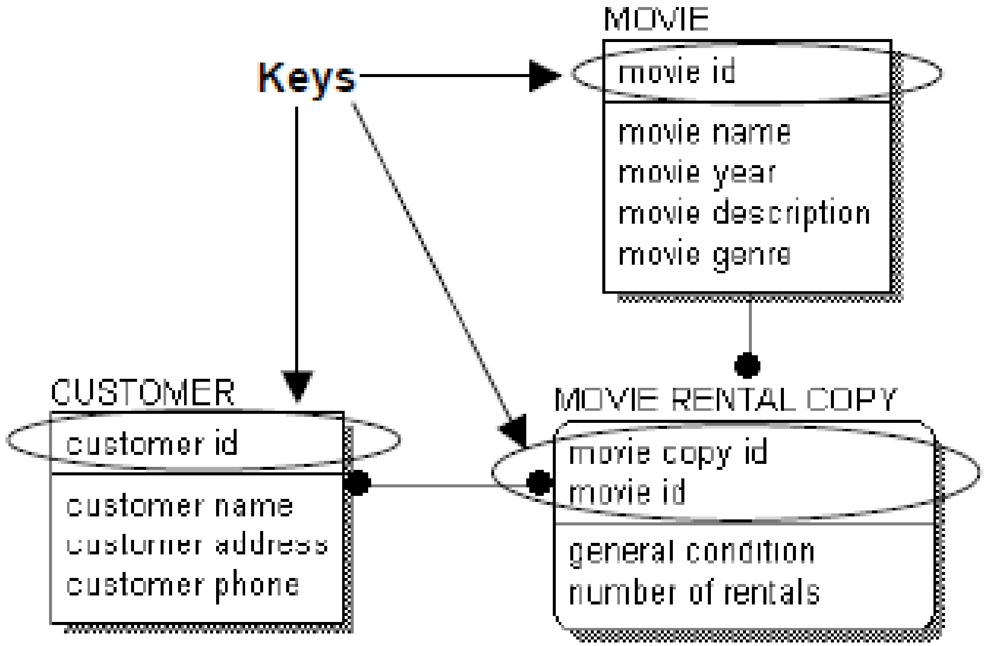


EMPLOYEE
employee-number
emmployee-name
employee-gender
employee-type
employee-gender
MALE-EMPLOYEE
employee-number (FK)
FEMALE-EMPLOYEE
employee-number (FK)
maiden-name

	IDEF1X Subtype Notation		
	Complete	Incomplete	IE Subtype Notation
Exclusive Subtype			
Inclusive Subtype			

Key-based model

- The key-based model covers the same scope as the Entity Relationship Diagram (ERD) but exposes more of the detail, including the context in which detailed implementation level models can be constructed.

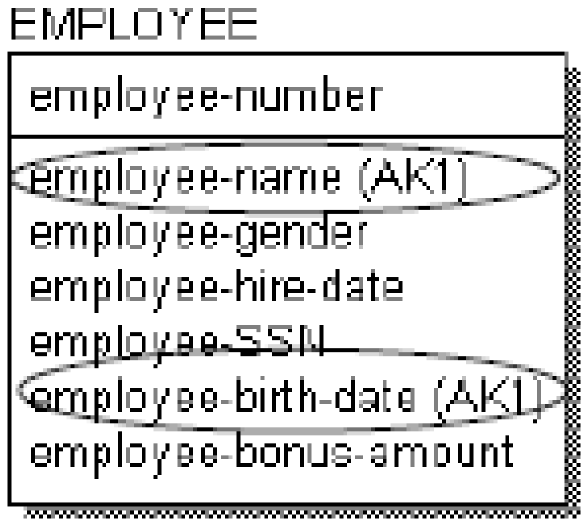


Selecting a Primary Key

- Uniquely identify an instance.
- Never include a NULL value.
- Not change over time. An instance takes its identity from the key. If the key changes, it is a different instance.
- Be as short as possible, to facilitate indexing and retrieval. If you need to use a key that is a combination of keys from other entities, make sure that each part of the key adheres to the other rules.

Designating Alternate Key Attributes

Alternate keys are often used to identify the different indexes, which are used to quickly access the data. In a data model, an alternate key is designated by the symbol (AKn), where n is a number that is placed after the attributes that form the alternate key group.



Fully-attributed model

- A Fully-Attributed (FA) Model is a third normal form data model that includes all entities, attributes, and relationships needed by a single project. The model includes entity instance volumes, access paths and rates, and expected transaction access patterns across the data structure.

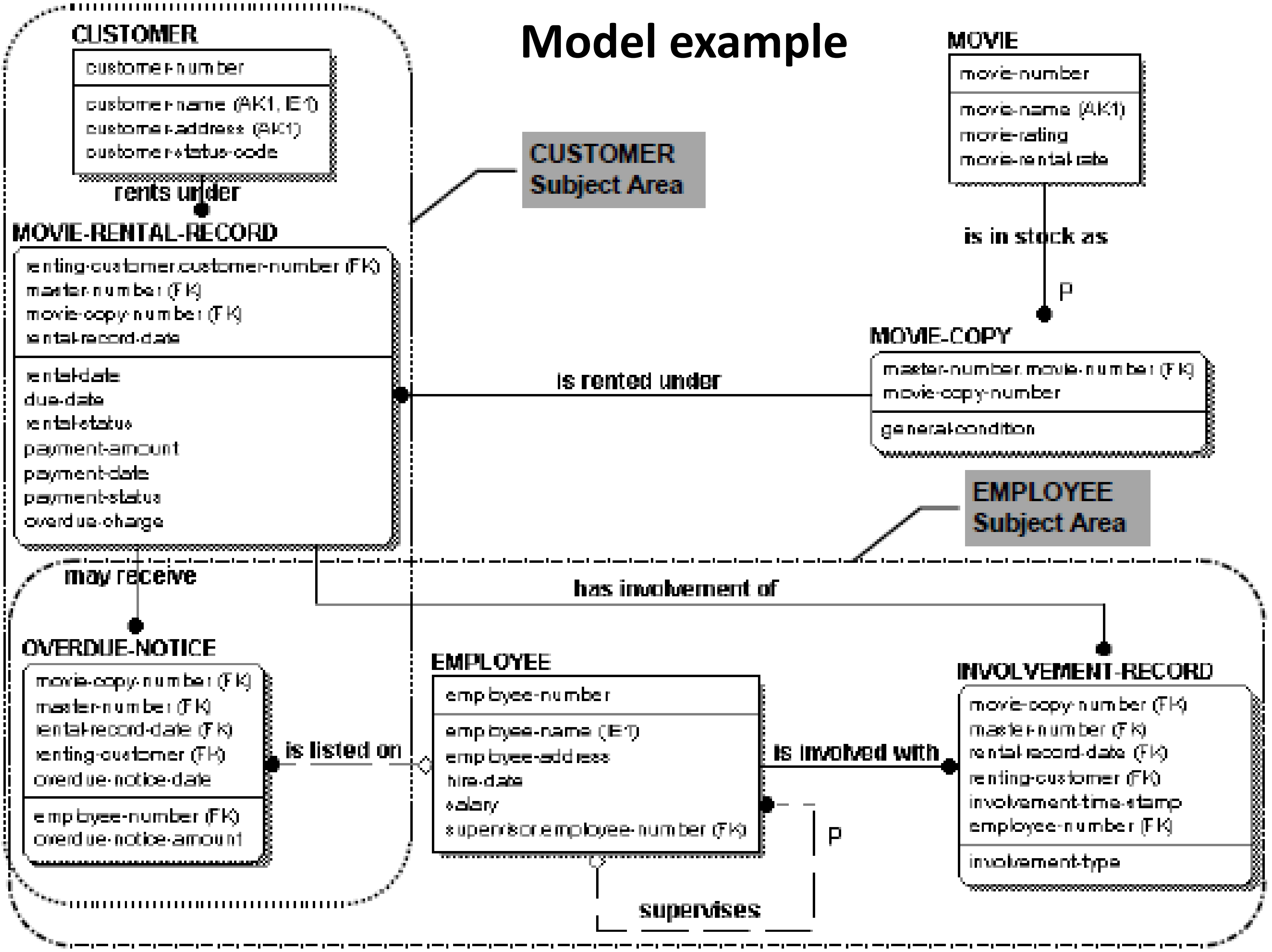
Physical model

- The objective of a physical model is to provide a database administrator with sufficient information to create an efficient physical database. The physical model also provides a context for the definition and recording in the data dictionary of the data elements that form the database, and assists the application team in choosing a physical structure for the programs that will access the data.

Physical model

- You can create a physical model from an ERD, key-based, or fully attributed model simply by changing the view of the model from Logical Model to Physical Model. Each option in the logical model has a corresponding option in the physical model. Therefore, each entity becomes a relational table, attributes become columns, and keys become indices.

Model example


CUSTOMER
customer-number
customer-name (AK1, IE1)
customer-address (AK1)
customer-status-code
rents under
MOVIE-RENTAL-RECORD
renting-customer.customer-number (FK)
master-number (FK)
movie-copy-number (FK)
rental-record-date
rental-date
due-date
rental-status
payment-amount
payment-date
payment-status
overdue-charge
CUSTOMER Subject Area
MOVIE
movie-number
movie-name (AK1)
movie-rating
movie-rental-rate
is in stock as
P
MOVIE-COPY
master-number.movie-number (FK)
movie-copy-number
general-condition
is rented under
EMPLOYEE Subject Area
may receive
has involvement of
OVERDUE-NOTICE
movie-copy-number (FK)
master-number (FK)
rental-record-date (FK)
renting-customer (FK)
overdue-notice-date
employee-number (FK)
overdue-notice-amount
is listed on
EMPLOYEE
employee-number
employee-name (IE1)
employee-address
hire-date
salary
supervisor.employee-number (FK)
is involved with
INVOLVEMENT-RECORD
movie-copy-number (FK)
master-number (FK)
rental-record-date (FK)
renting-customer (FK)
involvement-time-stamp
employee-number (FK)
involvement-type
P
supervises