

Australian/New Zealand Standard™

Information technology—CDIF semantic metamodel

Part 4: Data models



AS/NZS ISO/IEC 15476.4:2007

This Joint Australian/New Zealand Standard was prepared by Joint Technical Committee IT-015, Software and Systems Engineering. It was approved on behalf of the Council of Standards Australia on 3 November 2006 and on behalf of the Council of Standards New Zealand on 17 November 2006.
This Standard was published on 17 January 2007.

The following are represented on Committee IT-015:

Australian Computer Society
Australian Electrical and Electronic Manufacturers Association
Australian Society of Technical Communications
Australian Software Metrics Association
Engineers Australia/ACTS Joint Board in Software Engineering
Griffith University
National Association of Testing Authorities Australia
National ICT Australia
New Zealand Organisation for Quality
Software Quality Association, ACT
Software Quality Association, NSW
Systems Engineering Society of Australia
The University of Queensland
University of Auckland, NZ
University of South Australia
University of Technology, Sydney
Vendor Interests, NZ

Keeping Standards up-to-date

Standards are living documents which reflect progress in science, technology and systems. To maintain their currency, all Standards are periodically reviewed, and new editions are published. Between editions, amendments may be issued. Standards may also be withdrawn. It is important that readers assure themselves they are using a current Standard, which should include any amendments which may have been published since the Standard was purchased.

Detailed information about joint Australian/New Zealand Standards can be found by visiting the Standards Web Shop at www.standards.com.au or Standards New Zealand web site at www.standards.co.nz and looking up the relevant Standard in the on-line catalogue.

Alternatively, both organizations publish an annual printed Catalogue with full details of all current Standards. For more frequent listings or notification of revisions, amendments and withdrawals, Standards Australia and Standards New Zealand offer a number of update options. For information about these services, users should contact their respective national Standards organization.

We also welcome suggestions for improvement in our Standards, and especially encourage readers to notify us immediately of any apparent inaccuracies or ambiguities. Please address your comments to the Chief Executive of either Standards Australia or Standards New Zealand at the address shown on the back cover.

This Standard was issued in draft form for comment as DR 06512.

STANDARDS AUSTRALIA/STANDARDS NEW ZEALAND

RECONFIRMATION

OF

AS/NZS ISO/IEC 15476.4:2007

Information technology—CDIF semantic metamodel

Part 4: Data models

RECONFIRMATION NOTICE

Technical Committee IT-015 has reviewed the content of this publication and in accordance with Standards Australia procedures for reconfirmation, it has been determined that the publication is still valid and does not require change.

Certain documents referenced in the publication may have been amended since the original date of publication. Users are advised to ensure that they are using the latest versions of such documents as appropriate, unless advised otherwise in this Reconfirmation Notice.

Approved for reconfirmation in accordance with Standards Australia procedures for reconfirmation on 30 October 2015.

Approved for reconfirmation in New Zealand on behalf of the Standards Council of New Zealand on 18 May 2016.

The following are represented on Technical Committee IT-015:

Australian Computer Society
Australian Society for Technical Communication (NSW)
Charles Sturt University
Department of Defence (Australian Government)
Griffith University
Institute of IT Professionals New Zealand
IT Service Management Forum Australia
La Trobe University
National Association of Testing Authorities Australia
NEHTA
New Zealand Organisation for Quality
NSW Business Chamber
Quantitative Enterprise Software Performance
Systems Engineering Society of Australia
University of Technology, Sydney

NOTES

Australian/New Zealand Standard™

Information technology—CDIF semantic metamodel

Part 4: Data models

First published as AS/NZS ISO/IEC 15476.4:2007.

COPYRIGHT

© Standards Australia/Standards New Zealand

All rights are reserved. No part of this work may be reproduced or copied in any form or by any means, electronic or mechanical, including photocopying, without the written permission of the publisher.

Jointly published by Standards Australia, GPO Box 476, Sydney, NSW 2001 and Standards New Zealand, Private Bag 2439, Wellington 6020

ISBN 0 7337 7970 0

PREFACE

This Standard was prepared by the Joint Standards Australia/Standards New Zealand Committee IT-015, Software and Systems Engineering.

The objective of this Standard is to provide vendors and users of modelling tools and meta-data repositories with definition of the portion of the CDIF semantic metamodel that supports general Data Modelling. This includes Entity-Relationship-Attribute Modelling as well as Logical Database Design.

This Standard is identical with, and has been reproduced from ISO/IEC 15476-4:2005, *Information technology—CDIF semantic metamodel—Part 4: Data models*.

As this Standard is reproduced from an international standard, the following applies:

- (a) Its number appears on the cover and title page while the international standard number appears only on the cover.
- (b) In the source text ‘this International Standard’ should read ‘this Australian/New Zealand Standard’.
- (c) A full point substitutes for a comma when referring to a decimal marker.

References to International Standards should be replaced by references to Australian or Australian/New Zealand Standards, as follows:

<i>Reference to International Standard</i>		<i>Australian/New Zealand Standard</i>	
ISO/IEC		AS/NZS ISO/IEC	
15474	Information technology—CDIF framework	15474	Information technology—CDIF framework
15474-1	Part 1: Overview	15474.1	Part 1: Overview
15474-2	Part 2: Modelling and extensibility	15474.2	Part 2: Modelling and extensibility
15476	Information technology—CDIF semantic metamodel	15476	Information technology—CDIF semantic metamodel
15476-1	Part 1: Foundation	15476.1	Part 1: Foundation
15476-2	Part 2: Common	15476.2	Part 2: Common
15476-3	Part 3: Data definitions	15476.3	Part 3: Data definitions

CONTENTS

	<i>Page</i>
1 Scope	1
2 Conformance	2
2.1 General	2
2.2 Input conformance	2
2.3 Output conformance	2
2.4 Round-trip conformance	3
3 Normative references	3
4 Terms and definitions	3
4.1 From other International Standards	3
4.1.1 ISO/IEC 15474-1	3
4.1.2 ISO/IEC 13238-1	4
4.1.3 For this International Standard	4
5 Symbols (and abbreviated terms)	4
5.1 Naming, diagramming and definition conventions	4
5.2 Abbreviations	4
5.3 Notation for this subject area	4
6 Data models subject area overview	5
6.1 Introduction	5
6.2 Diagrams	6
6.3 Entities	11
6.4 Relationships	11
6.4.1 Binary relationship	12
6.4.2 N-ary relationship	12
6.4.3 Simple relationship	12
6.4.4 Complex relationship	12
6.5 Roles	14
6.6 Cardinalities	15
6.6.1 Outer cardinalities	16
6.6.2 Inner cardinalities	17
6.7 Multi-player roles	18
6.8 Role constraints	20
6.9 Attributes	22
6.10 Keys and access paths	23
6.11 Referential integrity	27
6.12 Supertypes and subtypes	29
6.12.1 Coverage	30
6.12.2 Exclusivity	31
6.12.3 Orthogonality	32
6.12.4 Inheritance	33
6.13 Refinement of inherited characteristics	34
6.13.1 Refinement of attribute characteristics	34
6.13.2 Refinement of relationship characteristics	35
6.14 Subsets	36
6.15 Clusters	37
6.16 Projections	38
6.17 Computable languages	41
7 Data models subject area summary	43
7.1 AttributableMetaObject hierarchy	43
7.2 MetaEntity summary	45

7.3	MetaRelationship summary.....	50
8	Data models subject area specification	54
8.1	Introduction	54
8.2	Subject area definition.....	54
8.3	Meta-entity definitions	55
8.3.1	AccessPath.....	55
8.3.2	Attribute	56
8.3.3	CandidateKey	58
8.3.4	Cluster	59
8.3.5	ComponentObject	59
8.3.6	DataModel	59
8.3.7	DataModelObject.....	60
8.3.8	DataModelSubset	61
8.3.9	DefinitionObject	61
8.3.10	Entity.....	62
8.3.11	ForeignKey	68
8.3.12	InheritableDataModelObject.....	68
8.3.13	Key	69
8.3.14	ProjectedAttribute.....	70
8.3.15	ProjectionComponent.....	71
8.3.16	Relationship	73
8.3.17	Role.....	74
8.3.18	RoleConstraint	75
8.3.19	RolePlayer	77
8.3.20	SemanticInformationObject	90
8.3.21	SubtypeSet	91
8.3.22	SubtypeSetMembershipCriterion.....	92
8.4	Meta-relationship definitions	94
8.4.1	AccessPath.Incorporates.Attribute	94
8.4.2	AccessPath.Instantiates.Key	95
8.4.3	Attribute.IsDiscriminatorFor.SubtypeSetMembershipCriterion	96
8.4.4	Attribute.IsInheritedFrom.Attribute	96
8.4.5	CandidateKey.Incorporates.ForeignKey	97
8.4.6	Cluster.Collects.DataModelObject.....	97
8.4.7	ComponentObject.References.DefinitionObject	97
8.4.8	DataModel.Collects.DataModelObject	98
8.4.9	DataModelObject.ActsAs.RolePlayer	98
8.4.10	DataModelObject.IsMemberOf.DataModelSubset.....	99
8.4.11	DataModelSubset.Excludes.Attribute.....	99
8.4.12	DataModelSubset.IsSubsetOf.DataModel	100
8.4.13	DefinitionObject.Contains.ComponentObject	100
8.4.14	DefinitionObject.IsConstructedWith.ProjectionComponent.....	100
8.4.15	Entity.IsAccessedUsing.AccessPath	101
8.4.16	Entity.IsIdentifiedBy.CandidateKey	101
8.4.17	ForeignKey.Incorporates.RolePlayer	102
8.4.18	ForeignKey.References.CandidateKey	102
8.4.19	InheritableDataModelObject.IsSubtypeIn.SubtypeSet	103
8.4.20	InheritableDataModelObject.IsSupertypeFor.SubtypeSet	104
8.4.21	Key.Incorporates.Attribute	105
8.4.22	Key.Incorporates.SemanticInformationObject	106
8.4.23	ProjectedAttribute.IsProjectionOf.Attribute.....	107
8.4.24	ProjectionComponent.IsFullProjectionOf.DefinitionObject.....	107
8.4.25	ProjectionComponent.IsProjectionOf.Attribute.....	108
8.4.26	Role.BelongsTo.Relationship	109
8.4.27	RoleConstraint.Incorporates.RoleConstraint	109
8.4.28	RoleConstraint.Incorporates.RolePlayer	110
8.4.29	RoleConstraint.Incorporates.SemanticInformationObject.....	110
8.4.30	RolePlayer.IsSupportedBy.Key.....	111
8.4.31	RolePlayer.Plays.Role	111

8.4.32	RolePlayer.Refines.RolePlayer	112
8.4.33	RolePlayer.RefinesForSubtype.DataModelObject	112
8.4.34	SubtypeSet.Specifies.SubtypeSetMembershipCriterion	113
8.4.35	SubtypeSetMembershipCriterion.Selects.InheritableDataModelObject	113

Table of Illustrations

Figure 1 – CDIF family of International Standards	1
Figure 2 – Diagrammatic Conventions - Keys	5
Figure 3 – Data Models subject area overview	6
Figure 4 – Partitioning/View mechanisms	7
Figure 5 – <i>Keys</i> and <i>AccessPath</i>	8
Figure 6 – <i>Roles</i> , <i>RolePlayers</i> and <i>RoleConstraints</i>	9
Figure 7 – Connection between Data Models and Common subject areas	10
Figure 8 – Connection between Data Models and Data Definition subject areas	11
Figure 9 – Binary Relationship	12
Figure 10 – Ternary Relationship	12
Figure 11 – Complex Binary Relationship	13
Figure 12 – Instance Diagram showing a Complex Relationship	14
Figure 13 – Roles	14
Figure 14 – Reflexive or Recursive Relationship	15
Figure 15 – Examples of Types of Binary Relationships	16
Figure 16 – Outer Cardinalities	16
Figure 17 – Inner Cardinalities	17
Figure 18 – Inner Cardinalities and Attributed Binary Relationships	18
Figure 19 – Multi-player Roles and Heterogeneous Collections	19
Figure 20 – Multi-player Roles and Common Outer Cardinality	20
Figure 21 – Role Constraint Network	21
Figure 22 – Constraints on Role Participation	21
Figure 23 – Instance Diagram showing Role Constraints	22
Figure 24 – Attributed <i>Relationships</i> , <i>Roles</i> and <i>RolePlayers</i>	23
Figure 25 – Foreign Keys	24

	<i>Page</i>
Figure 26 – <i>RolePlayer</i> as a component of a <i>ForeignKey</i>	25
Figure 27 – Key support of <i>RolePlayer</i>	26
Figure 28 – Key support for <i>RolePlayer</i> in a Reflexive <i>Relationship</i>	27
Figure 29 – Subtype Set Discriminator.....	30
Figure 30 – Abstract Supertype	31
Figure 31 – Concrete Supertype	31
Figure 32 – Non-exclusive Subtype Set	32
Figure 33 – Orthogonal Subtype Sets	32
Figure 34 – Instance Diagram showing Orthogonal Subtype Sets	33
Figure 35 – Overlapping Subtype Sets	33
Figure 36 – Inheritance	34
Figure 37 – Refinement of Cardinalities	35
Figure 38 – Instance Diagram showing refinement of Cardinalities.....	36
Figure 39 – Subsets.....	37
Figure 40 – <i>Clusters</i>	38
Figure 41 – Instance Diagram showing Simple Projection	39
Figure 42 – Instance Diagram showing Complex Projection	40
Figure 43 – Instance Diagram showing <i>Attribute</i> Renaming	41

INTRODUCTION

This International Standard will assist the vendors and users of modelling tools and meta-data repositories in developing mechanisms for interchanging information. This International Standard specifies an element of a family of related International Standards. When used together, these International Standards specify a mechanism for transferring information between tools.

ISO/IEC 15474-1:2002, *Information technology - CDIF framework - Part 1: Overview* and ISO/IEC 15474-2:2002, *Information technology - CDIF framework - Part 2: Modelling and extensibility* should be read first when initially exploring CDIF. The first explains the overall CDIF architecture and how the family of standards fits together. The second explains the scope, and modelling approach in CDIF. The CDIF meta-metamodel and extensibility mechanism are also defined in that document.

This International Standard explains the Data models subject area. The CDIF semantic metamodel is used to ensure that the information held by tools communicating using CDIF is transferred with an agreed meaning. It covers the information required to express Entity-Relationship-Attribute Modelling and Logical Database Design concepts.

This International Standard has been developed with the wide support and participation of vendors, users, academia and government involved in or familiar with the CASE industry, its products and the general requirements associated with interchanging information between these products.

This document is organized into the following Clauses.

— Clause 1 to 5 are prescribed ISO/IEC Clauses;

— Clause 6: Subject area overview:

This Clause gives an overview of the coverage of this subject area.

— Clause 7: Subject area summary:

This Clause gives an overview of the content of this subject area.

— Clause 8: Subject area specification:

This Clause gives the formal specification of all the objects defined in the subject area, and the formal reference to those used, but not defined in the subject area.

This document is intended to be used by anyone wishing to understand and/or use CDIF. This document provides a definition of a single subject area of the CDIF semantic metamodel. It is suitable for:

— Those evaluating CDIF;

— Those who wish to understand the principles and concepts of a CDIF transfer; and

— Those developing importers and exporters.

This document, ISO/IEC 15474-1:2002, *Information technology - CDIF framework - Part 1: Overview*, and the framework document ISO/IEC 15474-2:2002, *Information technology - CDIF framework - Part 2: Modelling and extensibility*, should be read first when initially exploring CDIF and before attempting to read other documents in the CDIF family of standards.

While there are no specific prerequisites for reading this document, it will be helpful for the reader to have familiarity with the following:

- Entity-Relationship-Attribute modelling;
- Modelling (CASE) tools;
- Information repositories;
- Data dictionaries;
- Multiple meta-layer modelling.

Information technology — CDIF semantic metamodel —

Part 4: Data models

1 Scope

The CDIF family of International Standards is primarily designed to be used as a description of a mechanism for transferring information between modelling tools. It facilitates a successful transfer when the authors of the importing and exporting tools have nothing in common except an agreement to conform to CDIF. The language that is defined for the transfer format also has applicability as a general language for import/export from repositories. The CDIF semantic metamodel defined for CASE also has applicability as the basis of standard definitions for use in repositories.

The International Standards which form the complete family of CDIF standards are documented in ISO/IEC 15474-1. These International Standards cover the overall framework, the transfer format and the CDIF semantic metamodel.

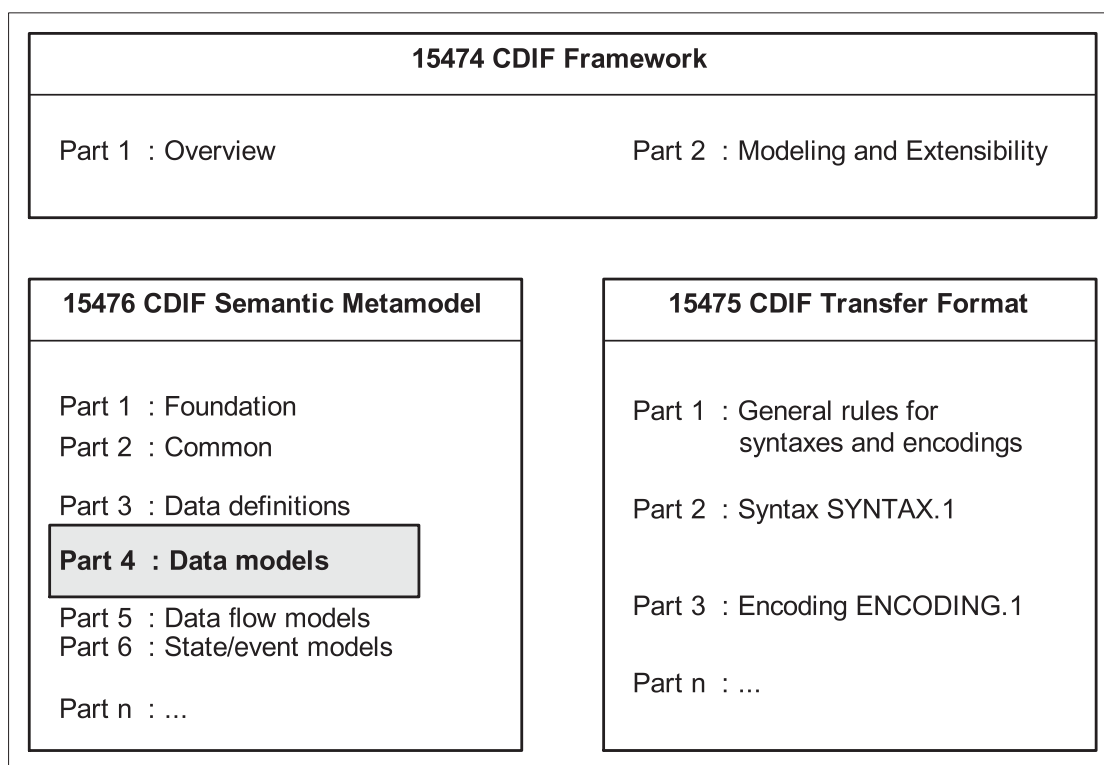


Figure 1 – CDIF family of International Standards

The diagram in Figure 1 depicts the various International Standards that comprise the CDIF family of standards. The shaded box depicts this Standard and its position in the CDIF family of standards.

This International Standard defines the portion of the CDIF semantic metamodel that supports general Data Modelling. This includes Entity-Relationship-Attribute Modelling as well as Logical Database Design.

2 Conformance

2.1 General

A product is fully standards conformant to a CDIF subject area standard if and only if it is input-conformant, output-conformant and round-trip conformant to each and every *MetaEntity*, *MetaRelationship*, *MetaAttribute*, and *AttributableMetaObject* which is defined and/or used in that standard, and it is also CDIF architecture conformant. A product may be partially input-conformant, and/or partially output-conformant, and/or partially round-trip conformant to a CDIF subject area standard.

2.2 Input conformance

Input conformance for a specific *MetaEntity*, *MetaRelationship*, *MetaAttribute*, or *AttributableMetaObject* (short: *CollectableMetaObject*) is determined by applying the following test:

A set of meta-data containing all meanings and structures standardized by a CDIF subject area is imported by the product under test. Then the meta-data which has arrived in the product is examined. The following options exist for the relation between the input (CDIF) meta-data and the imported (product) meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is input conformant if each instance of the specific *CollectableMetaObject* has arrived in the product without change of meaning or structure. If the *CollectableMetaObject* is a meta-entity or meta-relationship, its structural relationships to other *CollectableMetaObjects* have been preserved. If the *CollectableMetaObject* is a meta-attribute, the value of the meta-attribute has been preserved.
- 2 The product is input morphing conformant if each instance of the specific *CollectableMetaObject* has arrived in the product, but with some changes in meaning or structure. If the *CollectableMetaObject* is a meta-attribute, the value(s) for some instances of the meta-attribute have changed.
- 3 The product is not input conformant for that *CollectableMetaObject* if neither of the previous tests is satisfied.

2.3 Output conformance

Output conformance for a specific *CollectableMetaObject* is determined by applying the following test:

For the product being tested, a set of meta-data that includes all possible meanings and structures representable in that product is exported. Then the meta-data that has been exported is examined. The following options exist for the relation between the product's meta-data and the exported (CDIF) meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is output conformant if all of the meaning and structure for the specific *CollectableMetaObject* has been represented as meta-data in the product and has been exported as one or more instances of that *CollectableMetaObject*. If the *CollectableMetaObject* is a meta-attribute, the correct value of the meta-attribute has been exported.
- 2 The product is output morphing conformant if each instance of meta-data in the product that has the same meaning and structure as the *CollectableMetaObject* has been exported, but some instances have been exported as a different *CollectableMetaObject* or some of the meaning and structure has been changed.
- 3 If the product does not represent the meaning and structure associated with the *CollectableMetaObject*, output conformance for that *CollectableMetaObject* is not applicable to the product.

4 In all other cases, the product is not output conformant for that *CollectableMetaObject*.

2.4 Round-trip conformance

Round-trip conformance for a specific *CollectableMetaObject* is determined by applying the following test:

A set of meta-data containing all meanings and structures standardized by a CDIF subject area is imported by the product under test. Then the meta-data is exported again. The following options exist for the relation between the input meta-data and the output meta-data:

For a specific *CollectableMetaObject*:

- 1 The product is round-trip conformant if the meaning and structure of each instance of the *CollectableMetaObject* is preserved without changes during the round-trip. For a vendor to claim round-trip conformance, it is also necessary for the tool to be able to perform create, read, update, and delete operations on the imported (product) meta-data corresponding to the instances of the *CollectableMetaObject*.
- 2 The product is round-trip morphing conformant if each instance of the input *CollectableMetaObject* is preserved, but with some changes in meaning and/or structure. If the *CollectableMetaObject* is a meta-entity or meta-relationship, some of its instances' structural relationships to other *CollectableMetaObjects* have changed, or some instances have been transformed into other *CollectableMetaObjects*, or instances of other *CollectableMetaObjects* have been transformed into instances of the *CollectableMetaObject*. If the *CollectableMetaObject* is a meta-attribute, the values of some instances of the meta-attribute have changed or the domain of the meta-attribute has changed.
- 3 In all other cases, the product is not round-trip conformant for that *CollectableMetaObject*.

3 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15474-1, *Information technology — CDIF framework — Part 1: Overview*

ISO/IEC 15474-2, *Information technology — CDIF framework — Part 2: Modelling and extensibility*

ISO/IEC 15476-1, *Information technology — CDIF semantic metamodel — Part 1: Foundation*

ISO/IEC 15476-2, *Information technology — CDIF semantic metamodel — Part 2: Common*

ISO/IEC 15476-3, *Information technology — CDIF semantic metamodel — Part 3: Data definitions*

4 Terms and definitions

For the purposes of this document, the following definitions apply. Unless otherwise noted, the definitions are specific to this International Standard.

4.1 From other International Standards

4.1.1 ISO/IEC 15474-1

This part of ISO/IEC 15476 makes use of the following terms defined in ISO/IEC 15474-1:

CDIF
 CDIF family of standards
 CDIF semantic metamodel
 CDIF meta-metamodel
 CDIF transfer
 Instance
 Meta-attribute
 Meta-entity
 Metamodel
 Meta-object
 Meta-relationship
 Model
 Subject area
 Transfer
 Transfer format

4.1.2 ISO/IEC 13238-1

This part of ISO/IEC 15476 makes use of the following terms from ISO/IEC 13238-1:

Exporter
 Importer

4.1.3 For this International Standard

For the purpose of this part of ISO/IEC 15476 new terms are defined when introduced. Double quotes are used to introduce new terms (e.g., "model layer")

5 Symbols (and abbreviated terms)

5.1 Naming, diagramming and definition conventions

Conventions for naming, diagramming, describing and defining meta-objects can be found in Clause 7 of the framework document (ISO/IEC 15474-2:2002, *Information technology - CDIF framework - Part 2: Modelling and extensibility*).

5.2 Abbreviations

The following abbreviation is used in this International Standard:

CDIF CASE Data Interchange Format (originally)

5.3 Notation for this subject area

The instance diagrams appearing in Clause 6 use various techniques to depict relationships. Relationships are shown as lines and arrow-heads, or as diamonds. Keys and key components are shown as in Figure 2.

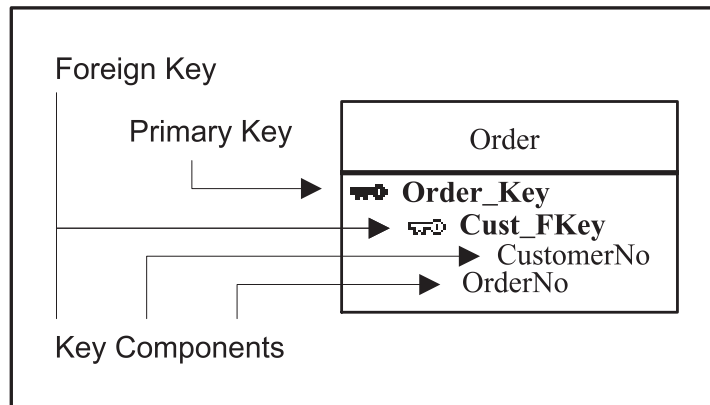


Figure 2 – Diagrammatic Conventions - Keys

Solid keys are primary keys; clear keys are foreign keys and indented items are key components. For example, Figure 2 reads, "**Orders** have a primary key called **Order_Key** that consists of a foreign key called **Cust_FKey** and an attribute **OrderNo**. **Cust_FKey** consists of a single attribute called **CustomerNo**."

6 Data models subject area overview

6.1 Introduction

The Data Models subject area addresses the major forms of entity-relationship-attribute modelling and logical database design. This subject area is technique independent but offers broad coverage, striving for a superset of the concepts represented in various techniques.

The following sub-clauses explain the major concepts supported in this subject area.

6.2 Diagrams

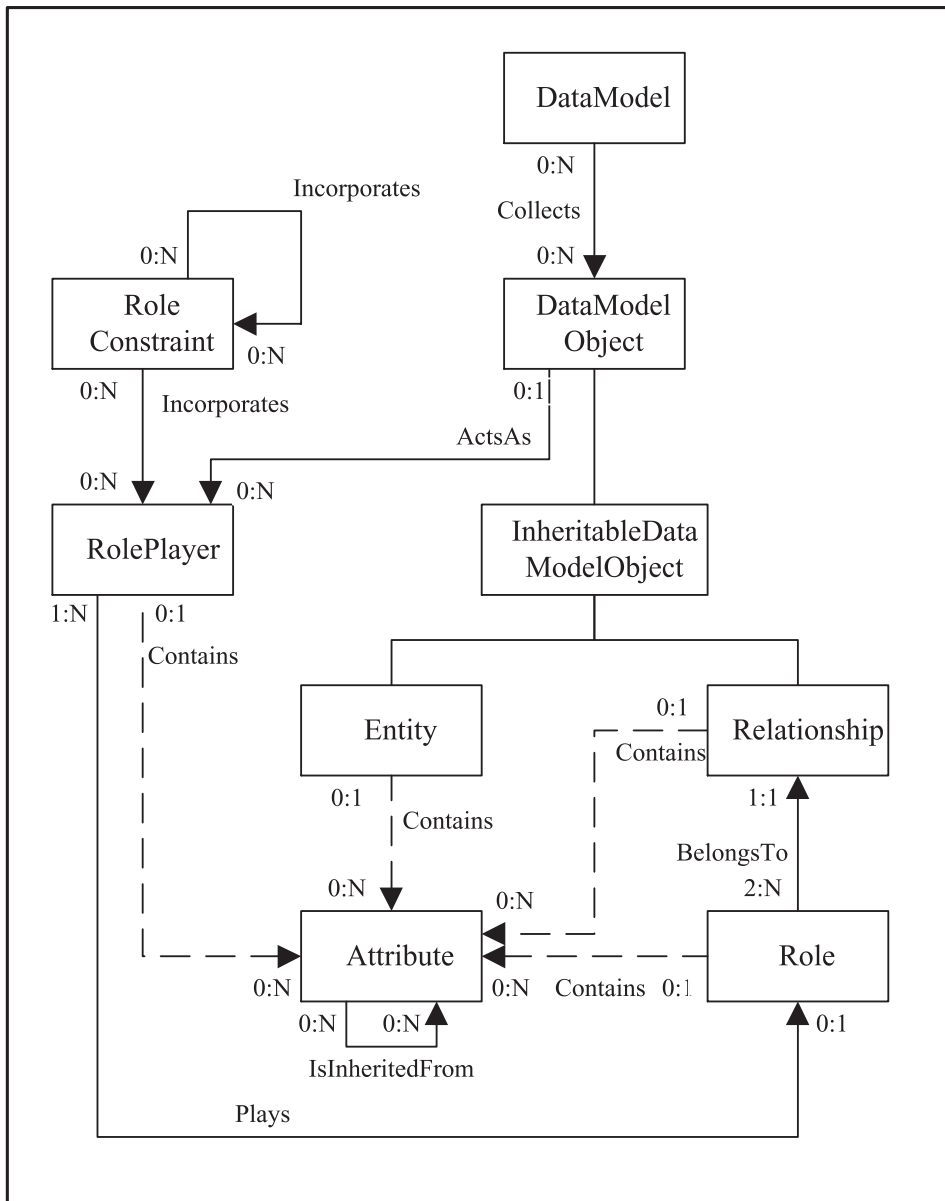


Figure 3 – Data Models subject area overview

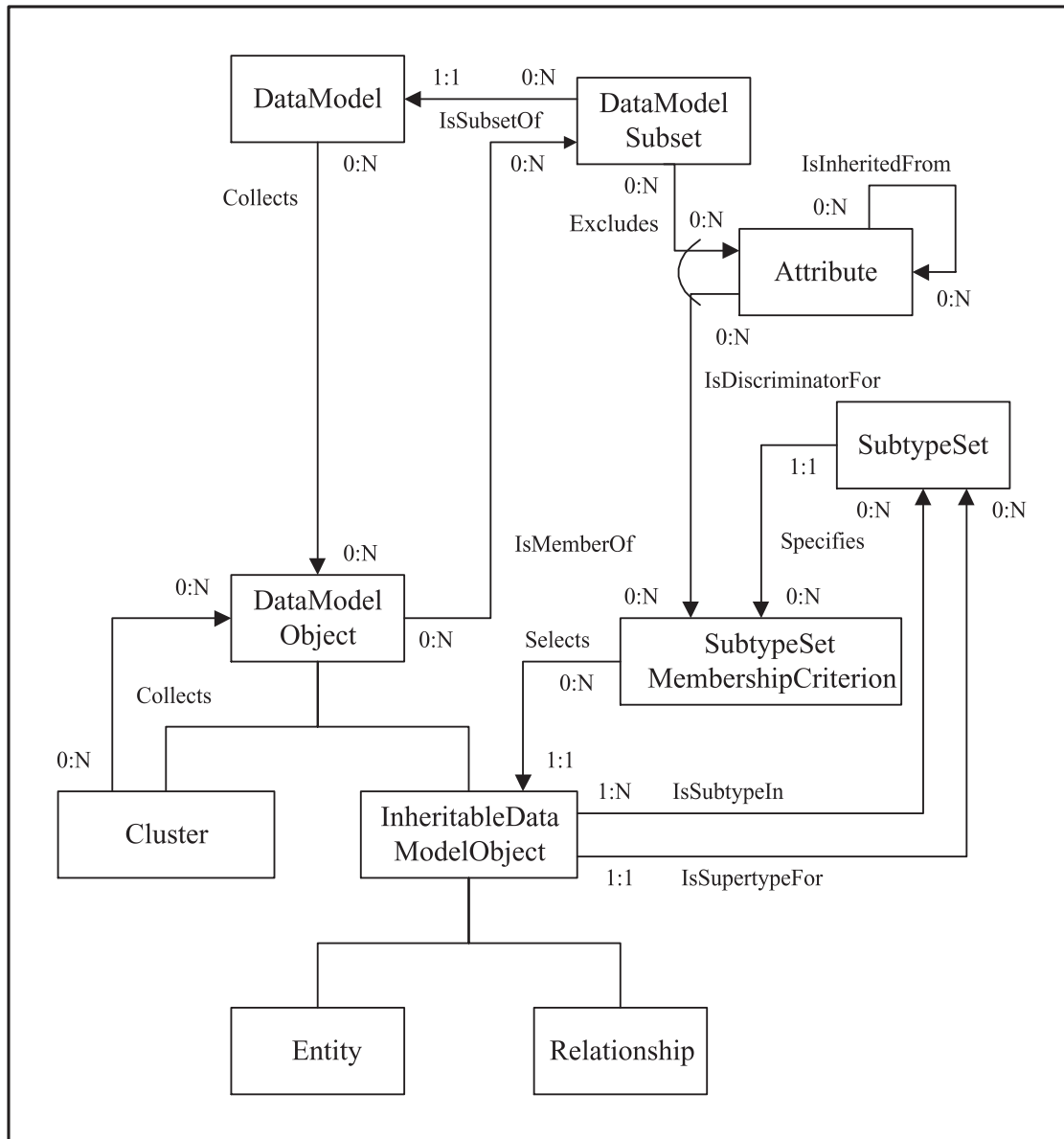


Figure 4 – Partitioning/View mechanisms

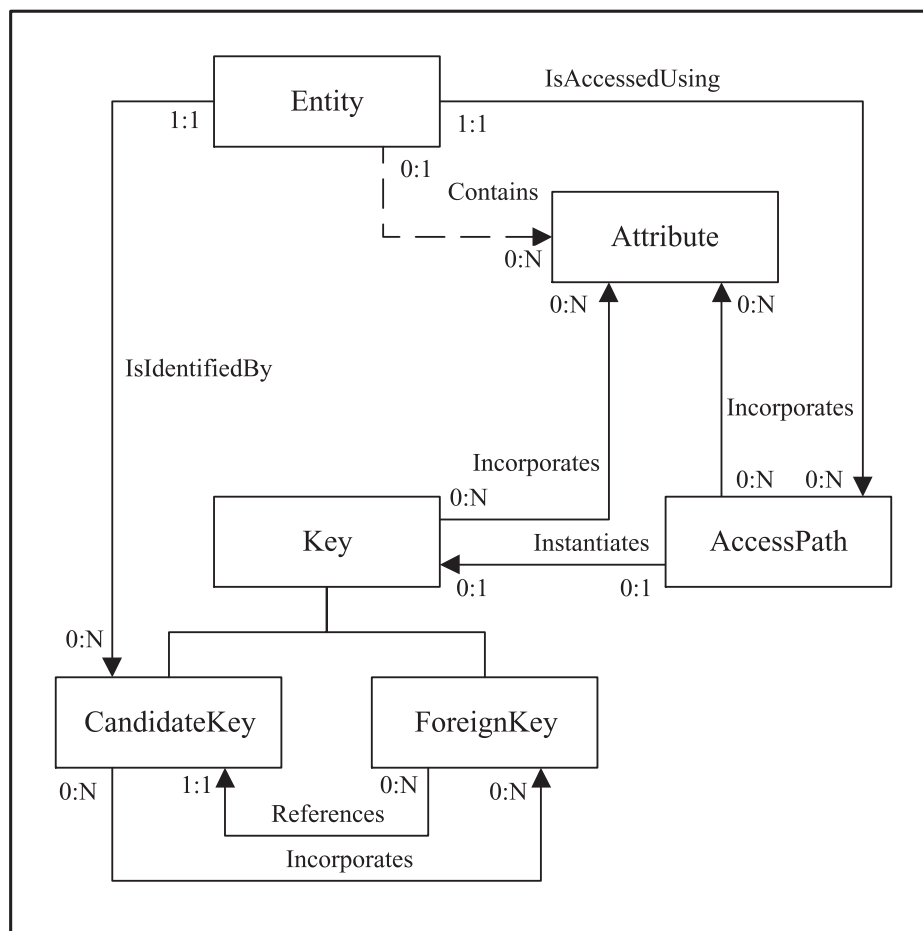


Figure 5 – Keys and AccessPath

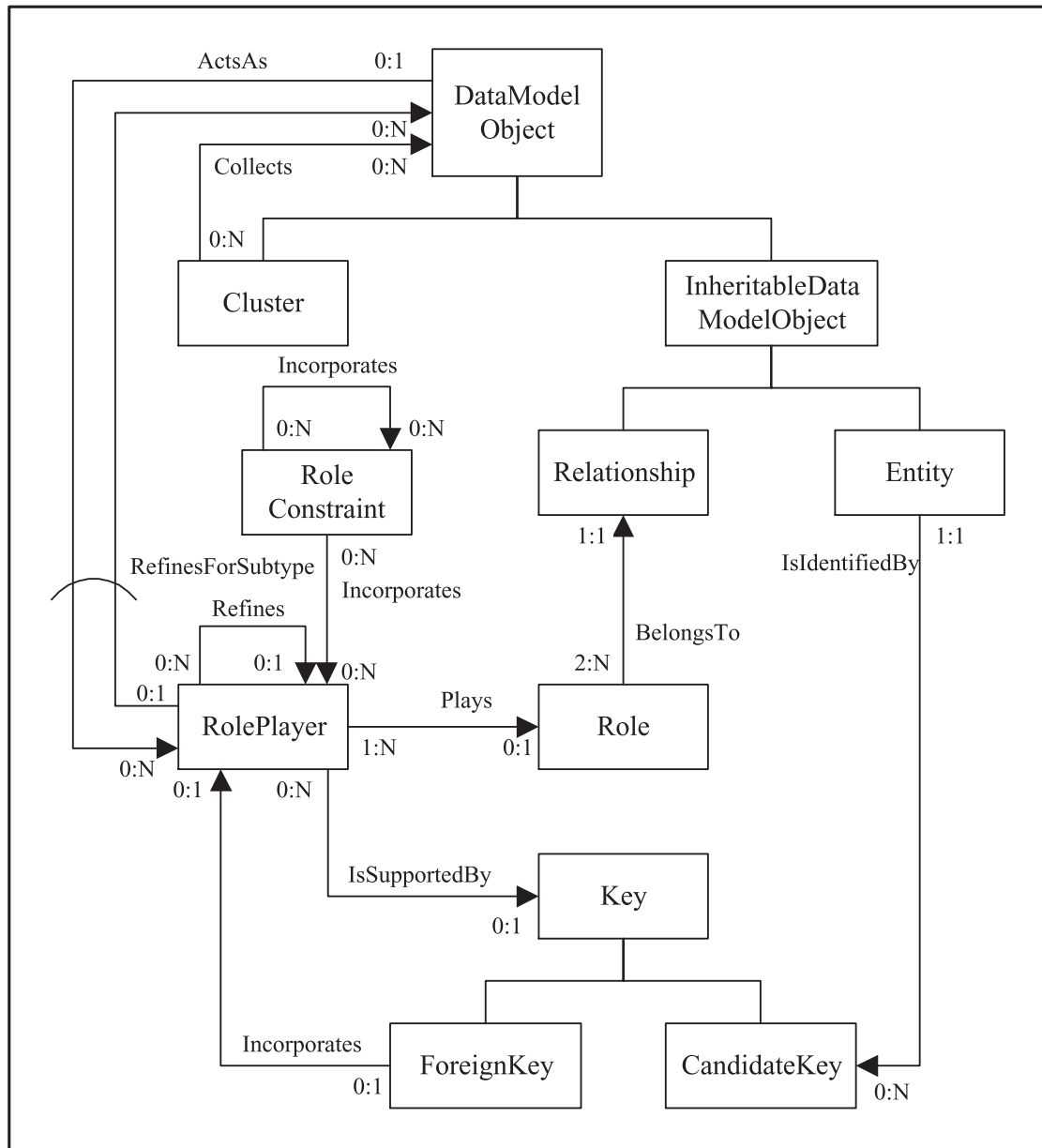


Figure 6 – Roles, RolePlayers and RoleConstraints

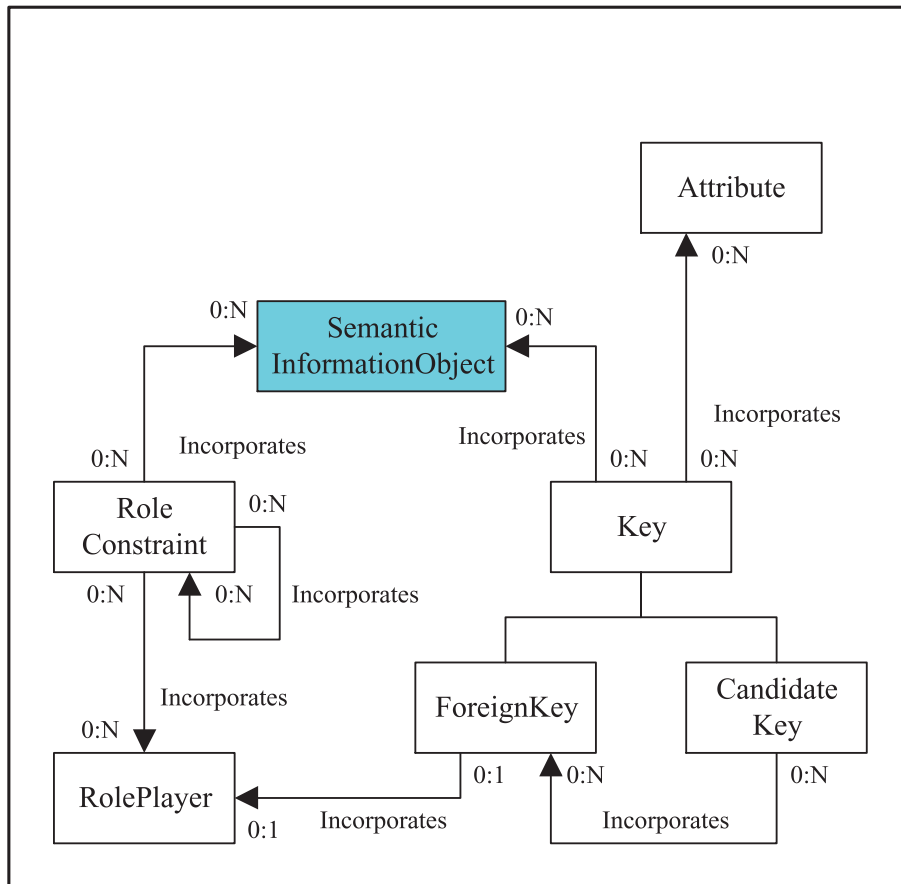


Figure 7 – Connection between Data Models and Common subject areas

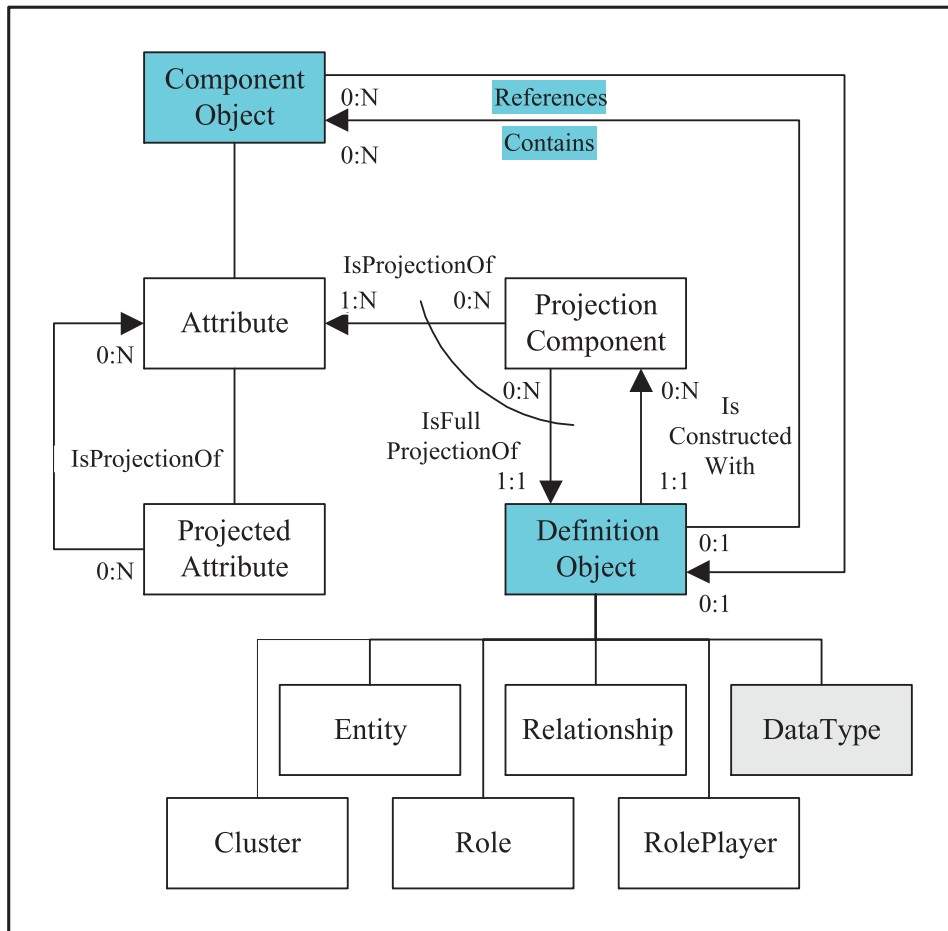


Figure 8 – Connection between Data Models and Data Definition subject areas

6.3 Entities

The concept of an "Entity," or object to be modeled, is represented directly by the meta-entity called *Entity*. A relational "Table" is also represented by *Entity*. A Table is a more concrete representation of an *Entity*.

6.4 Relationships

A relationship is an association between two or more *DataModelObjects*, expressed by the meta-entity *Relationship*. There are two orthogonal classifications of relationship, each with two subtypes:

- Arity expresses the degree or number of participants in a relationship. The two types are:
 - Binary
 - N-ary
- Intracacy categorizes the complexity of relationship association. The two types are:
 - Simple
 - Complex

6.4.1 Binary relationship

This is an association between two *DataModelObjects*. Figure 9 depicts a binary relationship between **Orders** and **LineItems** and reads, "**Order** contains **LineItem**."

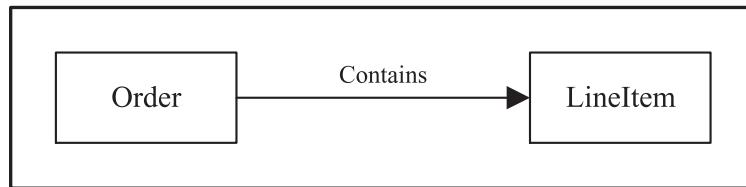


Figure 9 – Binary Relationship

6.4.2 N-ary relationship

This is an association between three or more *DataModelObjects*. Figure 10 depicts a ternary relationship which reads "An **Employee** is **Recruited** by an **Employer** using a **Recruiter**."

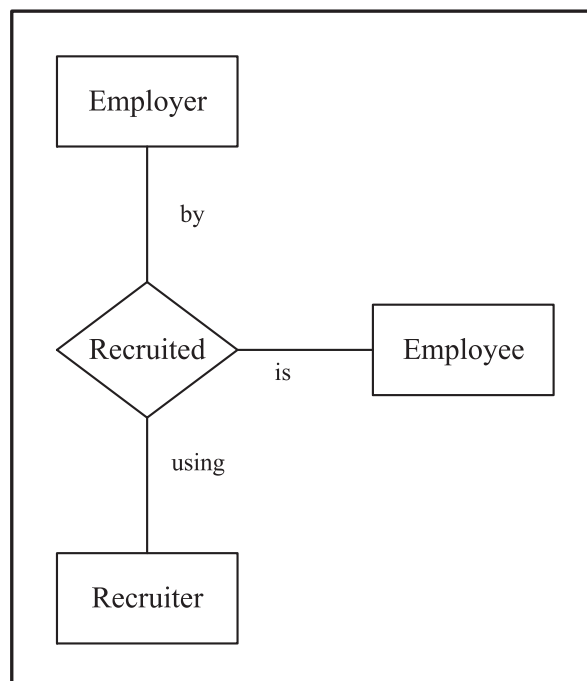


Figure 10 – Ternary Relationship

6.4.3 Simple relationship

This is an association between *Entities* or between *Clusters* or between *Entities* and *Clusters*. Figure 9 is an example of a Simple Binary Relationship and Figure 10 is an example of a Simple N-ary Relationship.

6.4.4 Complex relationship

This is an association between a *Relationship* and one or more *DataModelObjects*. Figure 11 depicts the association of an entity called **Student** and a binary relationship called **Class** using another binary relationship called **Assigned**.

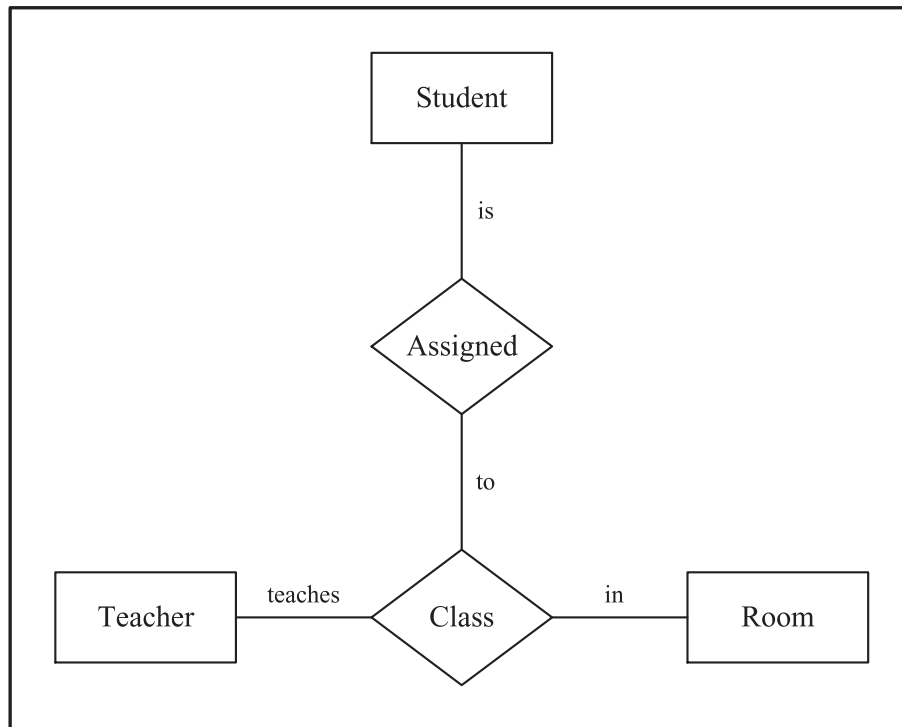


Figure 11 – Complex Binary Relationship

Figure 12 is an instance diagram for Figure 11, illustrating this concept. **Captain Bligh** is a student assigned to the class taught by the teacher **Fletcher Christian** in the room named **Bounty**.

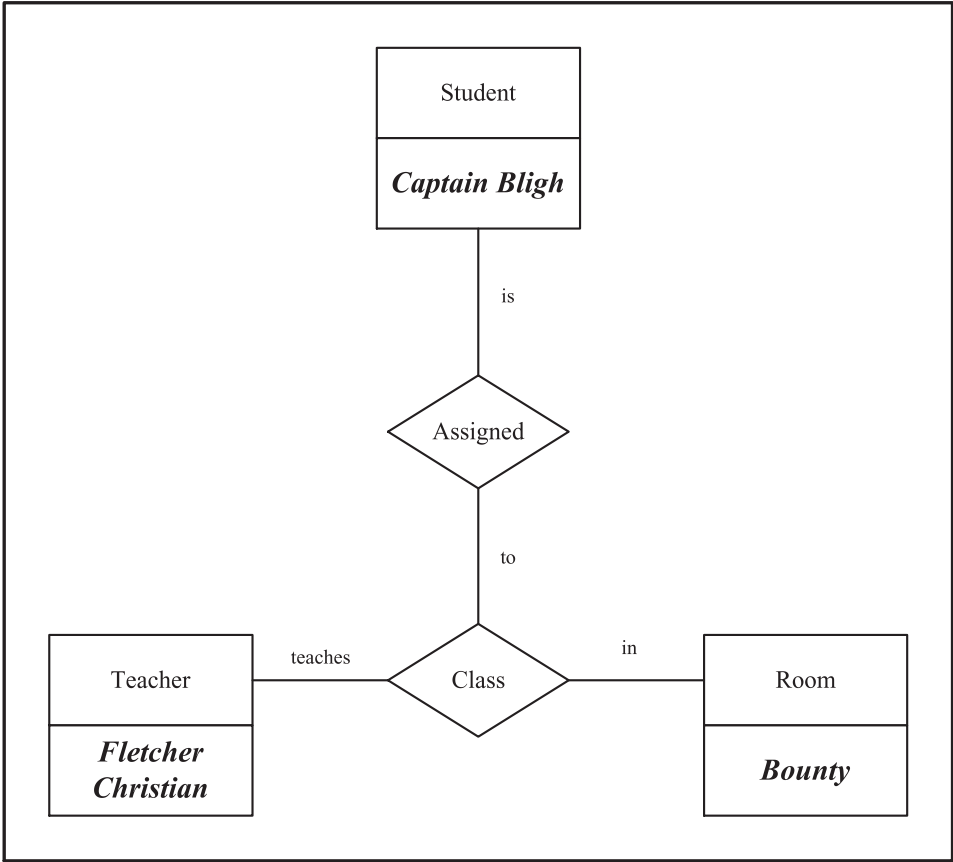


Figure 12 – Instance Diagram showing a Complex Relationship

6.5 Roles

A "Role" is the expression of an object playing a part in a relationship. The degree or arity of the relationship is a representation of the number of roles in the relationship. In a binary relationship there are two roles (Figure 13); in a ternary relationship there are three roles (Figure 10); and in an n-ary relationship there are "n" roles. In Figure 13, **Order** plays the role of **set** and **LineItem** plays the role of **member**.

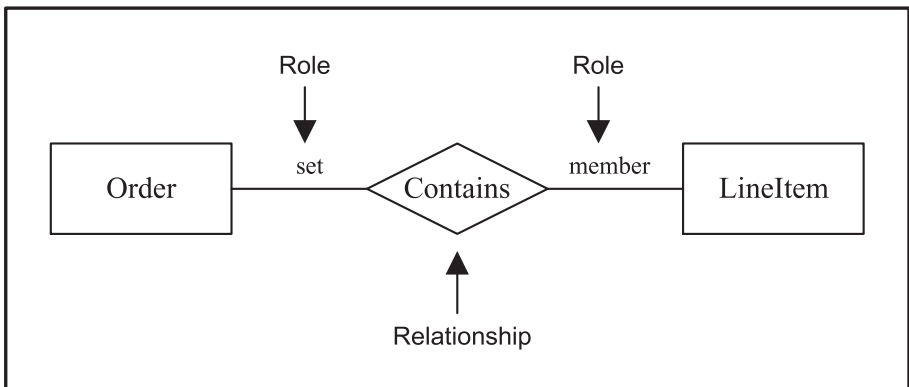


Figure 13 – Roles

The object playing a role in the relationship is represented by *DataModelObject*. The *DataModelObject* can be another *Relationship*, an *Entity* or a *Cluster*. A relationship must have at least two roles to be meaningful (a binary relationship). Incomplete relationship definitions are not supported.

The same *DataModelObject* can play both (binary) or many (n-ary) *Roles* in a *Relationship* as depicted in Figure 14. This is called a "Reflexive" or "Recursive" relationship. In this example, the relationship being modeled is that of a **Person** managing another **Person**. However, each **Person** instance plays only one role with respect to another **Person** instance - either taking the role of **Manager** or that of **Subordinate**.

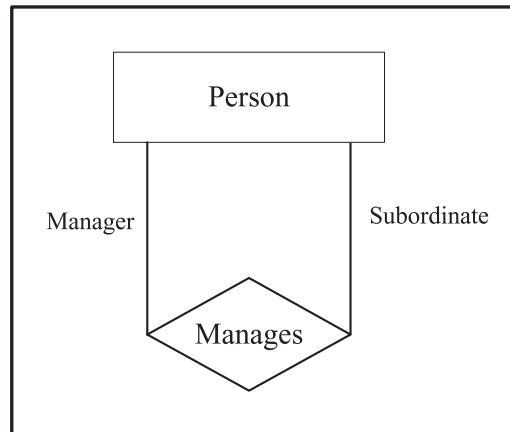


Figure 14 – Reflexive or Recursive Relationship

6.6 Cardinalities

Cardinalities are used to define static limits on the number of *DataModelObject* instances that can be associated using a *Relationship*. The limits are generally expressed as a range in the form "m:x", where "m" expresses the minimum number of object instances, and "x" expresses the maximum number of object instances. These pairings are visually associated with the object at each end of the relationship. For example, "0:N" expresses a range where the minimum participation is zero (optional participation) to a maximum value of "N" (unlimited). The combined cardinality of all participating objects expresses the type of relationship as illustrated in Figure 15.

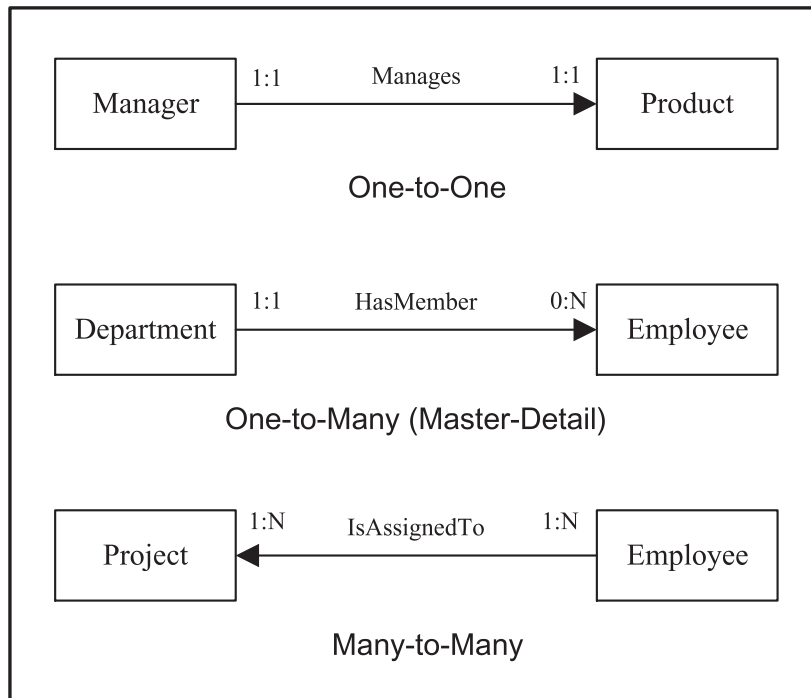


Figure 15 – Examples of Types of Binary Relationships

The Data Model Subject Area differentiates between inner and outer cardinality values for *DataModelObject*s participating in relationships. These concepts are described in the next two sub-clauses.

6.6.1 Outer cardinalities

The "Outer" cardinality defines the number of allowed instances of a participating *DataModelObject* from the viewpoint of the other participants in the *Relationship*. In the example in Figure 16, the outer cardinalities define the minimum and maximum number of instances of both the *Entity* called **LineItem** and the *Entity* called **Order** for the relationship called **Contains**. This example reads: "For a given **Order** there may be at minimum zero **LineItems** and at maximum an unlimited number of **LineItems** (0:N). For a given **LineItem** there may be one and only one **Order** (1:1)." It is assumed that for each **LineItem** there is a corresponding instance of the relationship **Contains**.

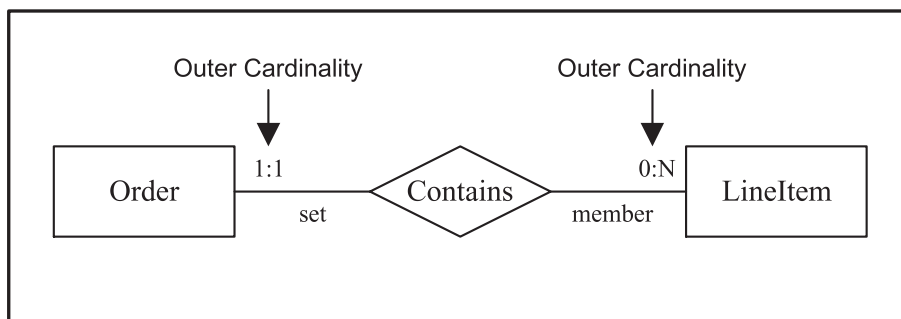


Figure 16 – Outer Cardinalities

6.6.2 Inner cardinalities

"Inner" cardinality defines the number of allowed instances of the *Relationship* from the viewpoint of a single instance of the *DataModelObject* playing a *Role*. In the context of un-attributed binary relationships, with which most people are familiar, the issue, of how many relationship instances are allowed, does not arise. It is assumed that a single instance of the relationship exists for each pair of related *DataModelObjects*. If outer cardinalities are not provided they are assumed to be **1:1**. However, in more complex Modelling, when both inner and outer cardinality values are used, the concept is important. In the example in Figure 17, the model reads "One or more **Bonuses** may be given by one **Employer** to one **Employee**. We can see that each **Bonus** may be given only once and that **Employers** and **Employees** may participate in many instances of **Given**."

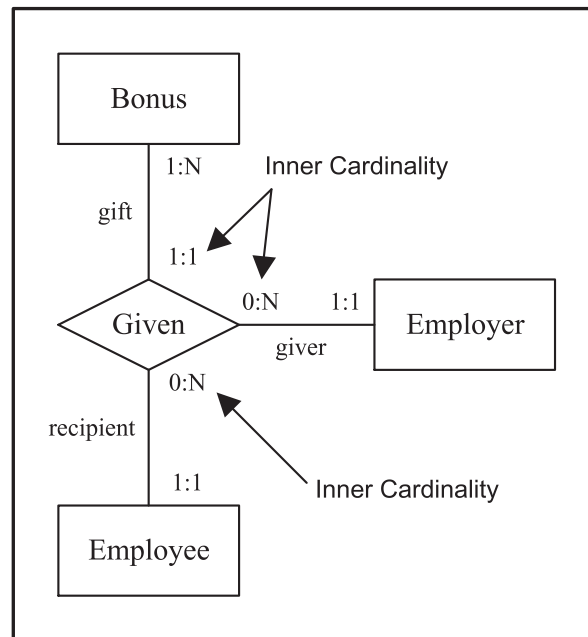


Figure 17 – Inner Cardinalities

Even where attributed relationships (either binary or n-ary) are used, a physical implementation would provide multiple instances of the relationship. But in conceptual terms, one must be able to define whether one or many relationship instances are required for any situation, since there are situations in which each is applicable.

Figure 18 depicts the concept of relationship instances. A given relationship has either one instance or multiple instances and each has its uses. Inner cardinality specifies which case is being used. In this example we have two depictions of the relationship between an **Order** and its **LineItems**. The left hand side depicts a model that allows multiple relationship instances and the right, a single instance.

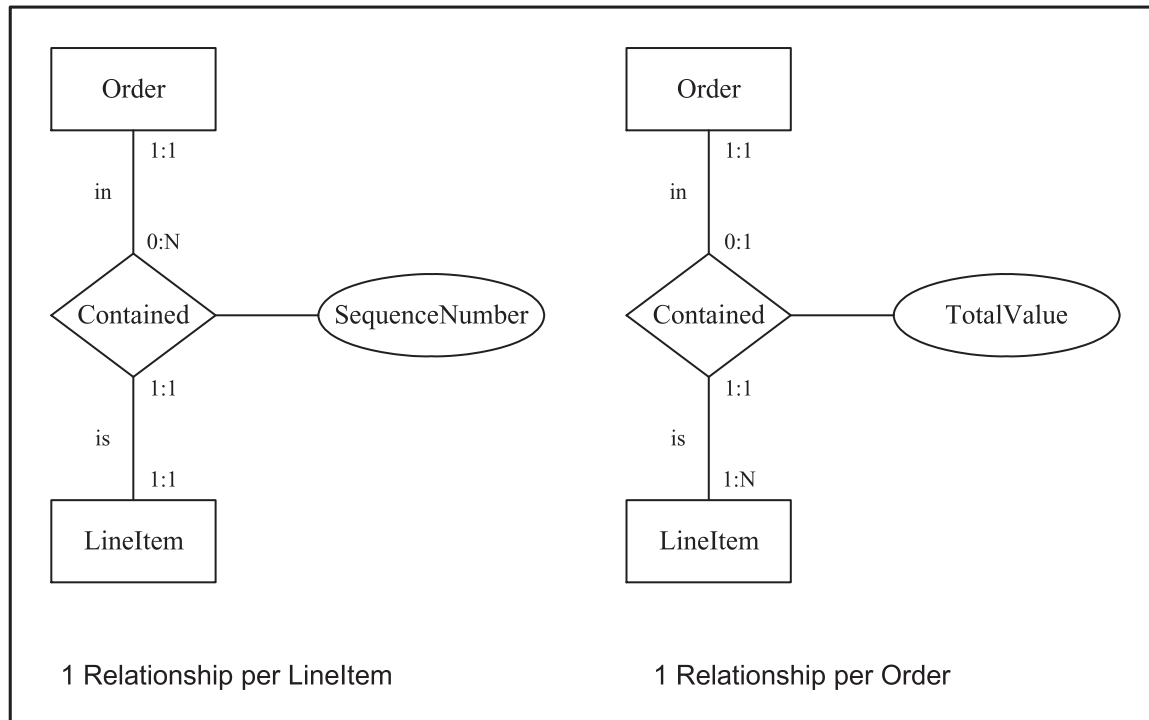


Figure 18 – Inner Cardinalities and Attributed Binary Relationships

In the case of the left hand side of Figure 18, the relationship holds an attribute called **SequenceNumber** to order the **LineItems**. The inner cardinality for **Order** is **0:N** and the outer cardinality for **LineItems** of **1:1**, giving one instance of the relationship for each **LineItem**.

In the right hand side the **TotalValue** attribute on the relationship shall occur only once for the **Order**, rather than once per **LineItem** so there shall only be one instance of the relationship for each **Order**. The **TotalValue** attribute can be considered as a property of the relationship between the two entities, rather than a property of the **Order**, since its value is dependent on the **LineItem**, rather than solely on the **Order** itself. Thus the inner cardinality from the viewpoint of an **Order** instance is **0:1**, and the outer cardinality for **LineItem** is **1:N**, giving only one instance of the relationship for each **Order**, and thus only one instance of **TotalValue**.

In both cases the inner cardinality for **LineItem** is **1:1** as each **LineItem** can only relate to a single **Order**.

In the absence of inner cardinality values, the assumed semantic is multiple relationship instances. In the absence of outer cardinality values, the assumed semantic is mandatory participation of exactly one instance from each of the associated *DataModelObjects*.

6.7 Multi-player roles

For each object participating in a relationship there shall be a *Role*. However, several different types of object may play the same *Role* in a *Relationship*. This is called a multi-player *Role* and it is modeled using the meta-entity *RolePlayer*. It is this mechanism that enables heterogeneous collections to be modeled. For example, in Figure 19 an **Office** is furnished with a **Desk**, **Chair**, **BookCase** and **FilingCabinet**. Note the specification of outer cardinalities on each *RolePlayer*, as each may have a different participation. There shall be only one **Desk** in an **Office** but one or more **Chairs** and zero or more **BookCases** and **FilingCabinets**.

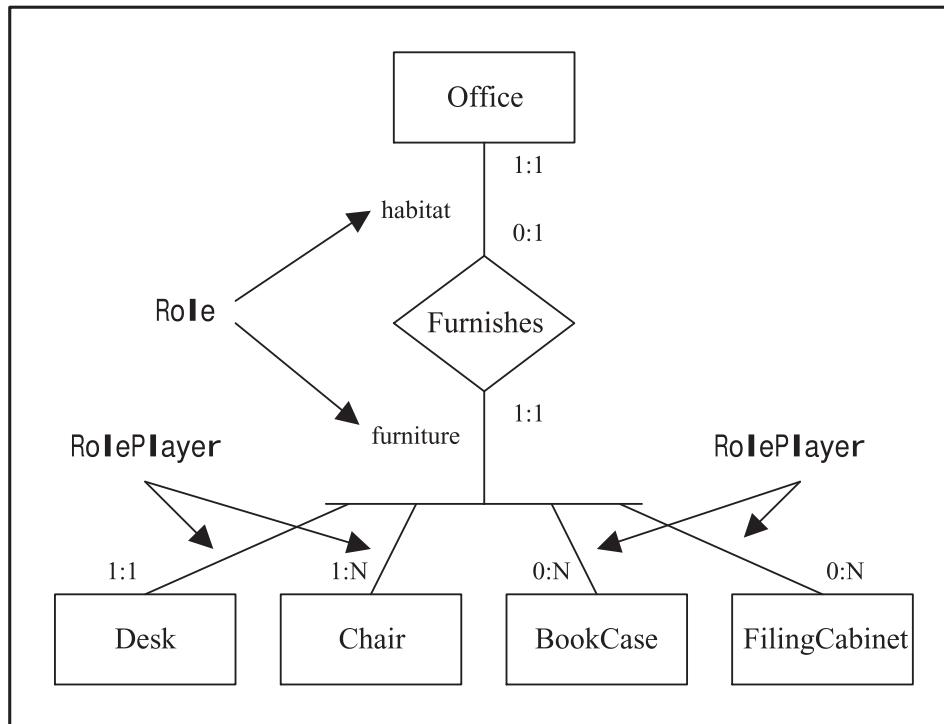


Figure 19 – Multi-player Roles and Heterogeneous Collections

In contrast, in Figure 20 the *Role recruitingAgent* specifies the cardinalities of the *RolePlayers Recruiter* and *Employee*. Note that this is merely a graphical convention; cardinality values are properties of the *RolePlayer* meta-entity, and thus in this case the *RolePlayers* called *serviceProvider* and *reference* both contain the same cardinality values.

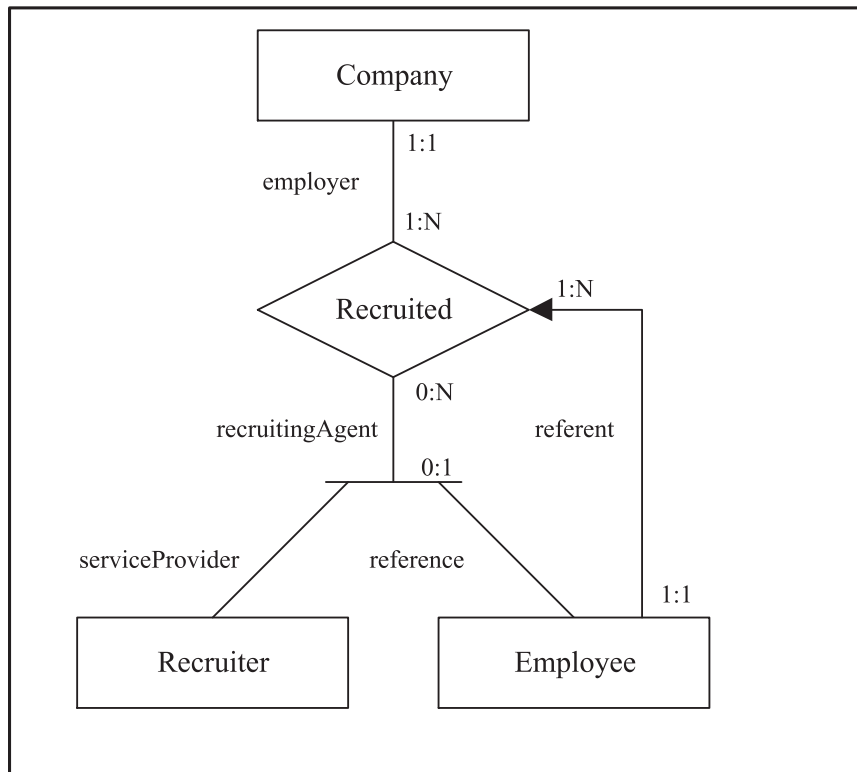


Figure 20 – Multi-player Roles and Common Outer Cardinality

All relationships use *RolePlayer*, even for simple binary relationships. The *RolePlayer* meta-entity contains meta-attributes that describe cardinalities and volumetrics concerning the participating object. The volumetric meta-attributes specify the number of insertions, deletions, updates and reads over a given time period that are expected for the participating object.

6.8 Role constraints

RoleConstraints provide a mechanism for expressing complex permissible groupings of relationships. This constrains the participation of an instance of a *DataModelObject* (*Entity*, *Relationship* or *Cluster*) in a set of relationships. The meta-attribute *Operator* in the meta-entity *RoleConstraint* expresses the constraint, whose values are the boolean operators inclusive **OR**, exclusive **XOR** and **AND**. The groupings are expressed on the *RolePlayers* which make up the relationships, rather than the relationships themselves, as multiple *DataModelObjects* may play a single role. The meta-relationship *RoleConstraint.Incorporates.RolePlayer* allows constraint groups to be constructed. Note that a role may participate in more than one constraint group. Figure 21 illustrates these concepts.

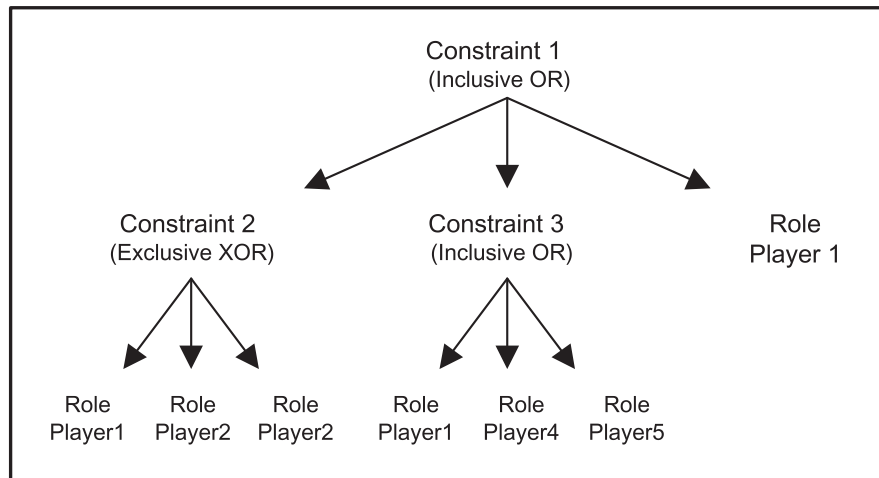


Figure 21 – Role Constraint Network

A single object can act as more than one *RolePlayer* in a *Relationship* with a need to constrain the participation of its instances to only one *Role*. Figure 22 illustrates this point. In this example, **Employee** is constrained to being either a player in the role **recruitingAgent** or a player in the role **referent**. This example models employee recruitment and is read "An **Employee** participates in one or more acts of recruitment. For each recruitment, there is one **Company**, and optionally, either a single **Recruiter** or **Employee** is used. **Recruiters** and **Employees** can be used in zero or more acts of recruitment. An **Employee** is constrained from recruiting him/herself."

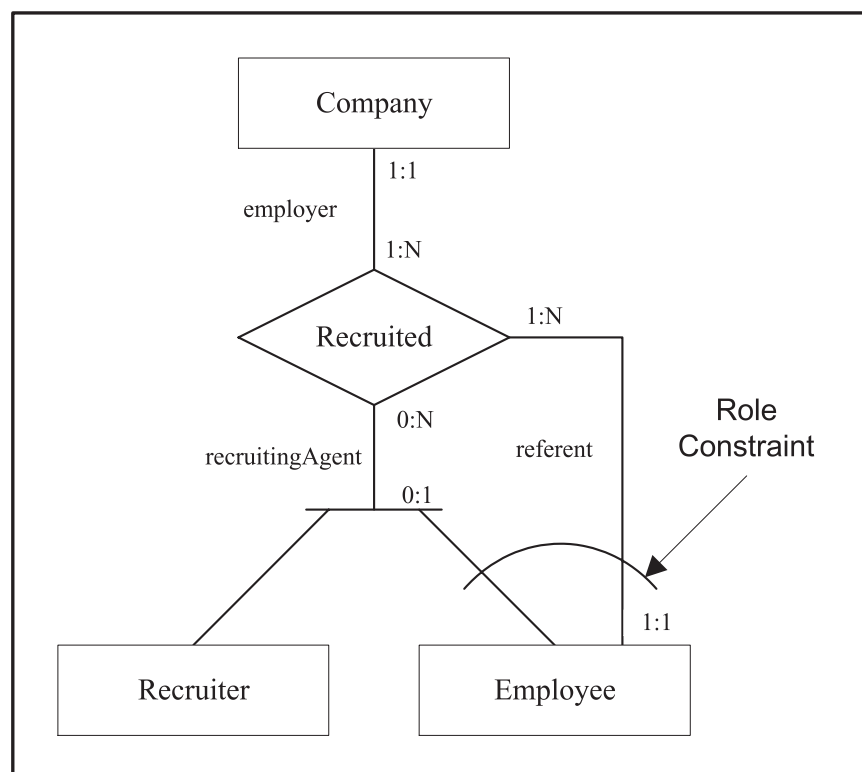


Figure 22 – Constraints on Role Participation

Figure 23 is an instance diagram of Figure 22, illustrating the concept of *Role* constraints.

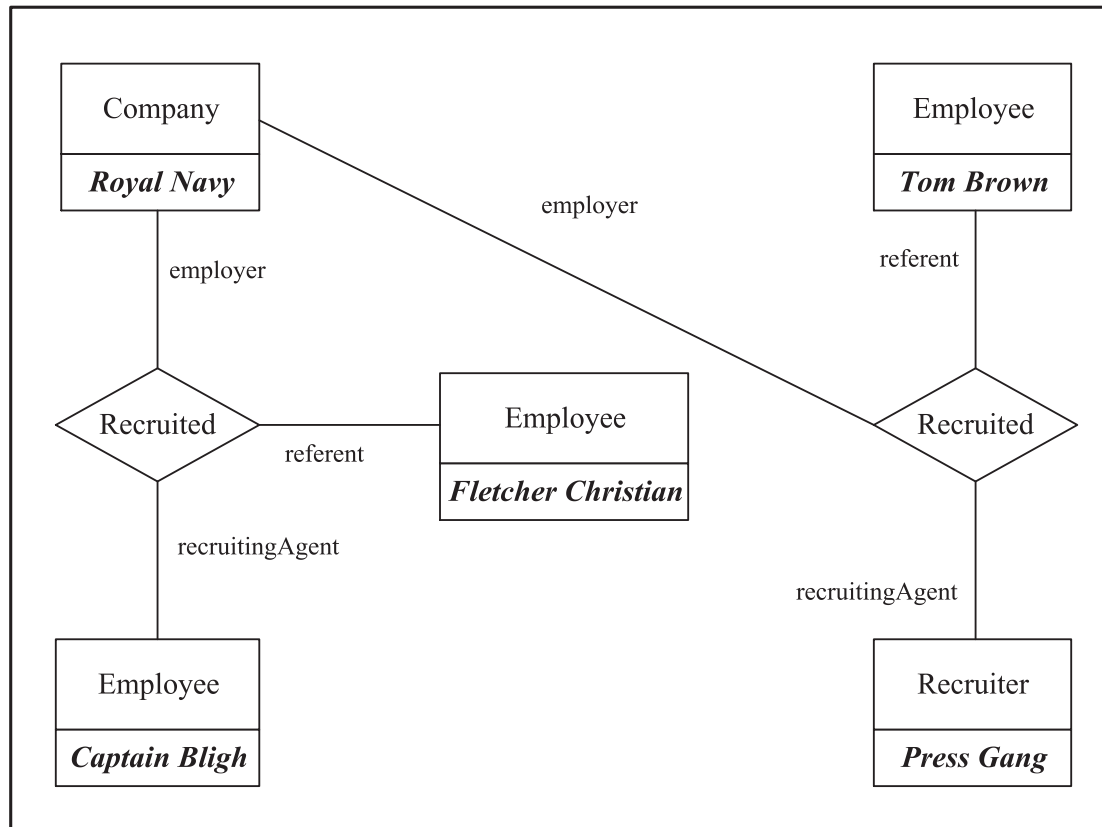


Figure 23 – Instance Diagram showing Role Constraints

6.9 Attributes

In this subject area the data content of *DefinitionObjects* is described by the use of *Attribute* or *ProjectedAttribute* (which is a subtype of *Attribute*). These are assigned to the subtypes of *DefinitionObjects* through the meta-relationship *DefinitionObject.Contains.ComponentObject*. For example, an *Entity* called **Customer** contains attributes **Name**, **Address** and **PhoneNumber**.

ProjectedAttributes are dependent attributes that exist as a result of a projection operation such as computation. For example, the *ProjectedAttribute* called **Total** exists as a result of multiplying the *Attributes* called **Price** and **Quantity**. For a detailed discussion on projection see Sub-clause Projections.

The detailed type information about attributes is not covered by this subject area; the Data Definition subject area is used to describe this. The connections between the Data Models subject area and the Data Definition (see ISO/IEC DIS 15476-3:2000, *Information Technology - CDIF Semantic Meta-model - Part 3: Data Definition*) subject area is shown in Figure 8.

Since the Data Definition subject area has its own decomposition mechanism of a *DataType* into its subtypes, an *Entity* should be modelled as it only contains top-level *Attributes* when the Data Models subject area is used along with the Data Definition subject area.

Roles, *RolePlayers*, and *Relationships* may also have attributes defined for them using the meta-relationship *DefinitionObject.Contains.ComponentObject*. Figure 24 illustrates the use of each of these. This models how an **Employee** is recruited by a **Company**. The *Relationship* called **Recruited** has an *Attribute* of **ApprovedBy**. Sometimes the services of a **Recruiter** are used and sometimes an existing **Employee** refers someone. In the case of the former the *RolePlayer* called **serviceProvider** is paid a **Fee** and the *RolePlayer* called **reference** is paid a **Bonus**.

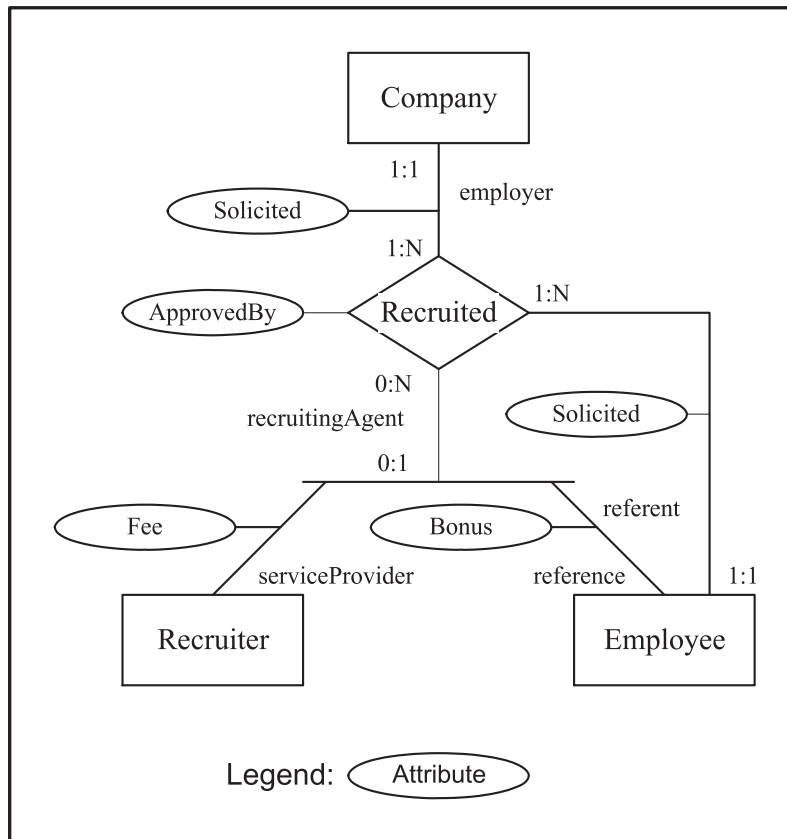


Figure 24 – Attributed Relationships, Roles and RolePlayers

6.10 Keys and access paths

Entity instances are identified by *Key* values and are accessed using *AccessPath* values. A *Key* is a means of identification of *Entity* instances, providing the basis of normalization and referential integrity (see 6.11). *Keys* are used to support relationships. *AccessPaths*, sometimes known as Alternate Keys, are used to specify efficient access to *Entity* instances and are often implementations of *Keys* (using the meta-relationships *Entity.IsAccessedUsing.AccessPath* and *AccessPath.Instantiates.Key*).

Keys are classified into two types:

a) Candidate Keys

CandidateKeys are a means of identification of the containing *Entity*. The concatenated values of *CandidateKey* components contain unique values and thus uniquely identify *Entity* instances. A primary key is simply one of the set of candidate keys for a given *Entity* and signifies that this is the key that is the major identifier. This concept of unique identification is the basis on which *Entity* normalization takes place. *CandidateKeys* used in conjunction with *ForeignKeys* provide the basis of referential integrity, which is the mechanism for maintaining consistency between related *Entities* after Delete, Update or Insert operations (See 6.11).

b) Foreign Keys

ForeignKeys are a replication of a *CandidateKey* from a related *Entity*. The *ForeignKey*'s purpose is to provide a referential integrity constraint to ensure that any *ForeignKey* values have matching *CandidateKey* values affecting creation, modification and deletion of *Entity* instances. It is also used to support the concept of a relationship. The *ForeignKey* refers to the *CandidateKey* by using the meta-relationship called

ForeignKey.References.CandidateKey. For example, from Figure 25, in the relationship **Customer.Places.Order** if there is a *ForeignKey* value in **Order** that does not have an equivalent *CandidateKey* value in **Customer** then there is an integrity failure.

Keys contain one or more *Attributes* drawn from the set that describes the related *Entity*. Multiple attributes comprising a key characterize the key as a concatenated key. Note that *CandidateKeys* may also contain *ForeignKeys* as in the example in Figure 25. **Order_Key** consists of both a foreign key, **Cust_FKey**, and an attribute **OrderNo**. The foreign key is **Cust_FKey** which contains a single attribute **CustomerNo**. This is modeled through the meta-relationship *CandidateKey.Incorporates.ForeignKey*. This implies that the attributes that describe the foreign keys are the same attributes (and attribute values) as those describing the associated candidate key, and thus the attributes are "global" in scope. However, *Attributes* are local in their ability to describe a *DefinitionObject* and thus *ForeignKeys* must contain *Attributes* that describe the entity that contains the *ForeignKey*. The association between a *CandidateKey*'s attributes and those of its *ForeignKeys* is accomplished by defining for each *CandidateKey* attribute a *DataType* that is shared between the *CandidateKey* attribute and the corresponding *ForeignKey* attribute. Additionally, the sequence of attributes that comprise the *ForeignKey* must be the same sequence that comprise the *CandidateKey*.

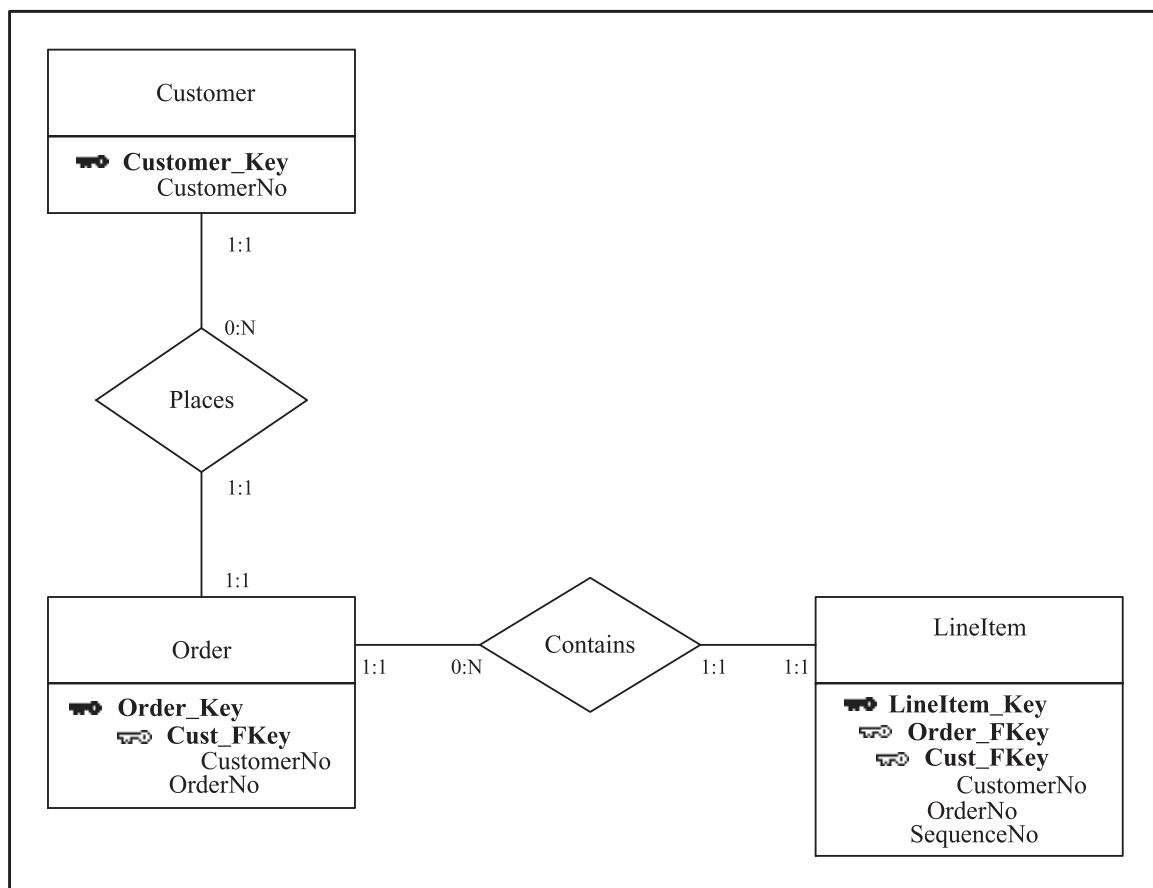


Figure 25 – Foreign Keys

In some techniques a relationship can be a component of a *ForeignKey*. This is modeled by relating the relevant *RolePlayer* to the *ForeignKey* using *ForeignKey.Incorporates.RolePlayer*. In this technique attributes describing the key are not specified - they are implied from the candidate key; the foreign key is simply composed of the *RolePlayer* reference. Figure 26 illustrates this where **Employee_FKey** in **Software** is described by the *RolePlayer* **version** and **Employee_FKey** in **Hardware** is described by the *RolePlayer* **model**. This means that both **Hardware** and **Software** are not described by the attribute **Name** that is the description of the primary key in **Employee**.

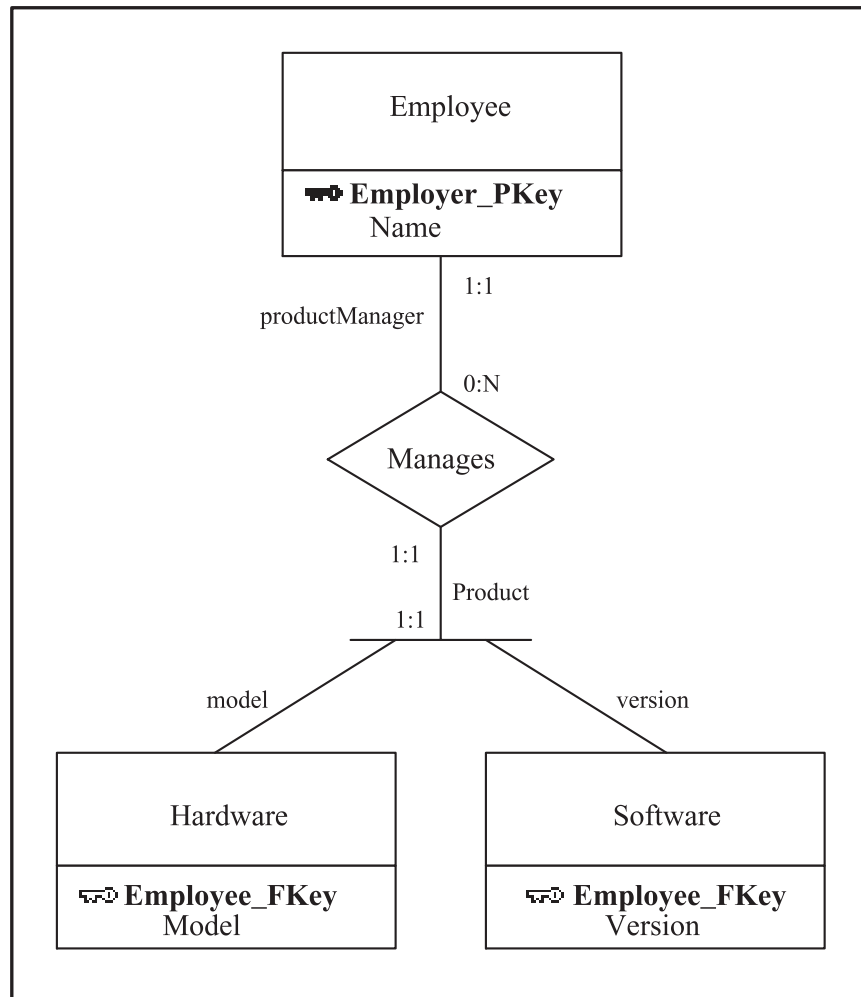


Figure 26 – RolePlayer as a component of a ForeignKey

Pairings of *CandidateKeys* and *ForeignKeys* are used to more concretely express the *Relationships* defined between *Entities*. The meta-relationship *RolePlayer.IsSupportedBy.Key* is used as the mechanism to express this, by selecting a particular *CandidateKey* (usually the primary key) or *ForeignKey* from an entity to be used in support of the entity's participation as a *RolePlayer* in a particular *Relationship*. Where more than one *Entity* can play the same *Role* in a *Relationship*, there is one *Key* for each *RolePlayer* in the *Role*. Figure 27 illustrates this concept, where **Recruiter** and **Employee** act as *RolePlayers*, playing the role of **recruitingAgent** in the relationship. In this case the primary key **Recruiter_PKey** supports the *RolePlayer* **serviceProvider** and **Employee_PKey** supports the *RolePlayer* **reference**.

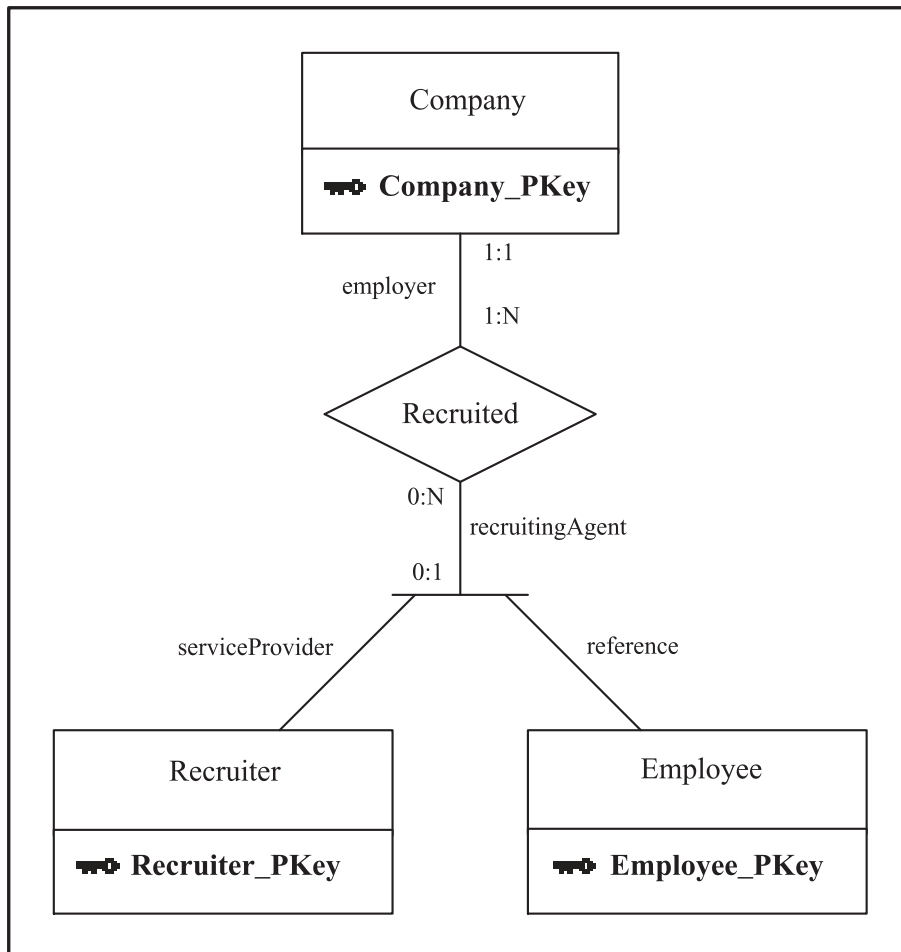


Figure 27 – Key support of *RolePlayer*

The same *DataModelObject* can play both (binary) or many (n-ary) *Roles* in a reflexive or recursive *Relationship* as depicted in Figure 14. In this example, the relationship being modeled is that of a **Person** managing another **Person**. However, each **Person** instance plays only one role with respect to another **Person** instance - either taking the role of **Manager** or that of **Subordinate**. The meta-relationship *RolePlayer.IsSupportedBy.Key* identifies which role is played by which instance - the *CandidateKey* **Person_PKey** would play the **Manager** role and the *ForeignKey* **Manager_Key** would play the **Subordinate** role.

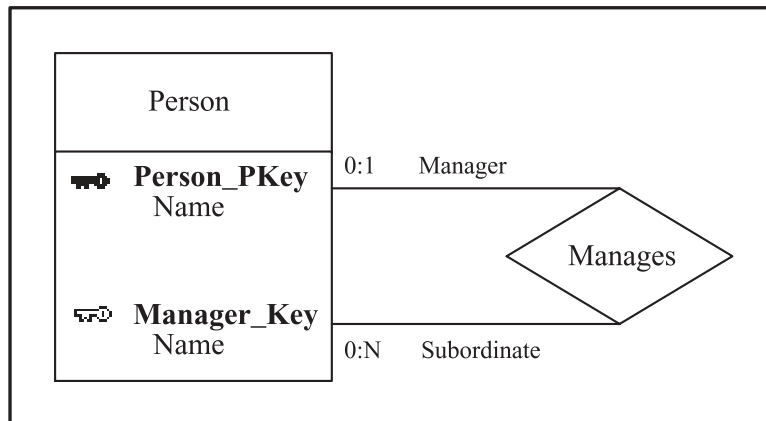


Figure 28 – Key support for *RolePlayer* in a Reflexive Relationship

6.11 Referential integrity

Keys are used to enforce referential integrity. Referential integrity rules are associated with *RolePlayer* as the mechanism for binding *CandidateKey* and *ForeignKey* pairs using the meta-relationship *RolePlayer.IsSupportedBy.Key*.

The meta-attributes *InsertEffect*, *DeleteEffect*, and *UpdateEffect*, belonging to *RolePlayer*, govern the behavior associated with Insert, Delete, and Update events on an *Entity* type. If the referenced *Entity* instance is inserted, deleted, or its *Key* is updated, then the rules outlined in Table 1 apply for each type of key when each event occurs:

Table 1 Referential Integrity Rules

Key	Insert	Delete	Update
Candidate	RESTRICTS CASCADES	RESTRICTS CASCADES SETNULL SETDEFAULT	RESTRICTS CASCADES SETNULL SETDEFAULT
Foreign	RESTRICTS CASCADES	RESTRICTS CASCADES	RESTRICTS CASCADES

CandidateKey InsertEffect

- RESTRICTS**

The insert operation cannot complete if there are no matching *ForeignKey* entity instances. This occurs where the *ForeignKey* entity's participation in the relationship is mandatory. For example, ***Employee.Has.WorkSpace*** (0:1 to 1:1), where an instance of *Employee* is inserted.

- CASCADES**

An instance of the *ForeignKey* entity shall be inserted. This is the equivalent of deferred checking of integrity. This applies to a mandatory relationship. For example, ***ProductManager.IsResponsibleFor.Product*** (0:N to 1:N), where an instance of ***ProductManager*** is inserted.