Part III

Entity-Relationship Model

Entity-Relationship Model

Educational Objective for Today . . .

- Knowing the concepts of the entity-relationship model
- Ability to conceptually model an application domain

Basics of Database Models

A database model is a system of concepts to describe databases. It defines the syntax and semantics of database descriptions for a database system.

- Database descriptions = database schemata

A Database Model Defines . . .

1. Static properties
   1. Objects
   2. Relationships

   including the primitive data types, which can describe data about the relations and objects,
2. Dynamic properties such as
   1. Operations
   2. Relationships between operations,
3. Integrity constraints of
   1. Objects
   2. Operations

Database Models

- Classical database models are especially suited for
  - Large amounts of data with a relatively static structure and
  - Describing static properties and integrity constraints
- Design models: (E)ER model, UML, ...
- Realization models: relational model, object-oriented models, ...

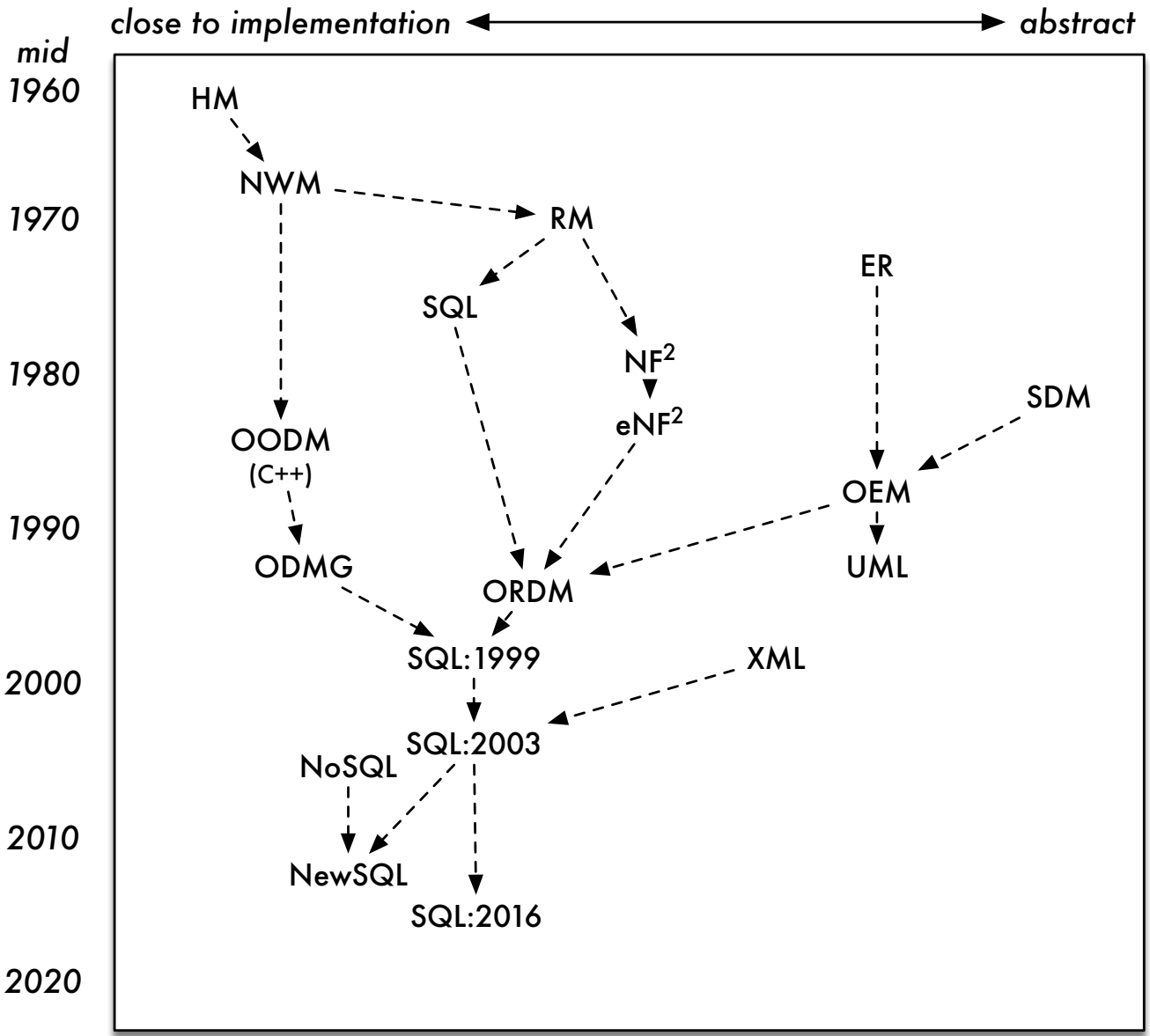
Databases versus Programming Languages

Database concept	**Type system of a programming language**
Database Model *Relation, Attribute . . .*	Type system **`int, struct ...`**
Database schema **`relation`** `WINE = (...)`	Declaration of variable **`var`** `x:` **`int`**`, y:` **`struct`** `Wine`
Database `WINE`(4961, 'Chardonnay', 'White', . . .)	Values 42, 'Cabernet Sauvignon'

Levels of Abstraction

Models	**Data**	**Algorithms**
abstract	entity-relationship model	structograms
concrete	hierarchical model network model relational model	Pascal C, C++ Java, C#

Overview of Database Models



Overview of Database Models /2

- HM: hierarchical model, NWM: network model, RM: relational model
- NF^2: model of nested (non-first-normal form $= NF^2$) relations, eNF^2: extended NF^2 model
- ER: entity-relationship model, SDM: semantic data models
- OODM / C++: object-oriented data models based on object-oriented programming languages, such as C++, OEM: object-oriented design models (e.g., UML), ORDM: object-relational data models

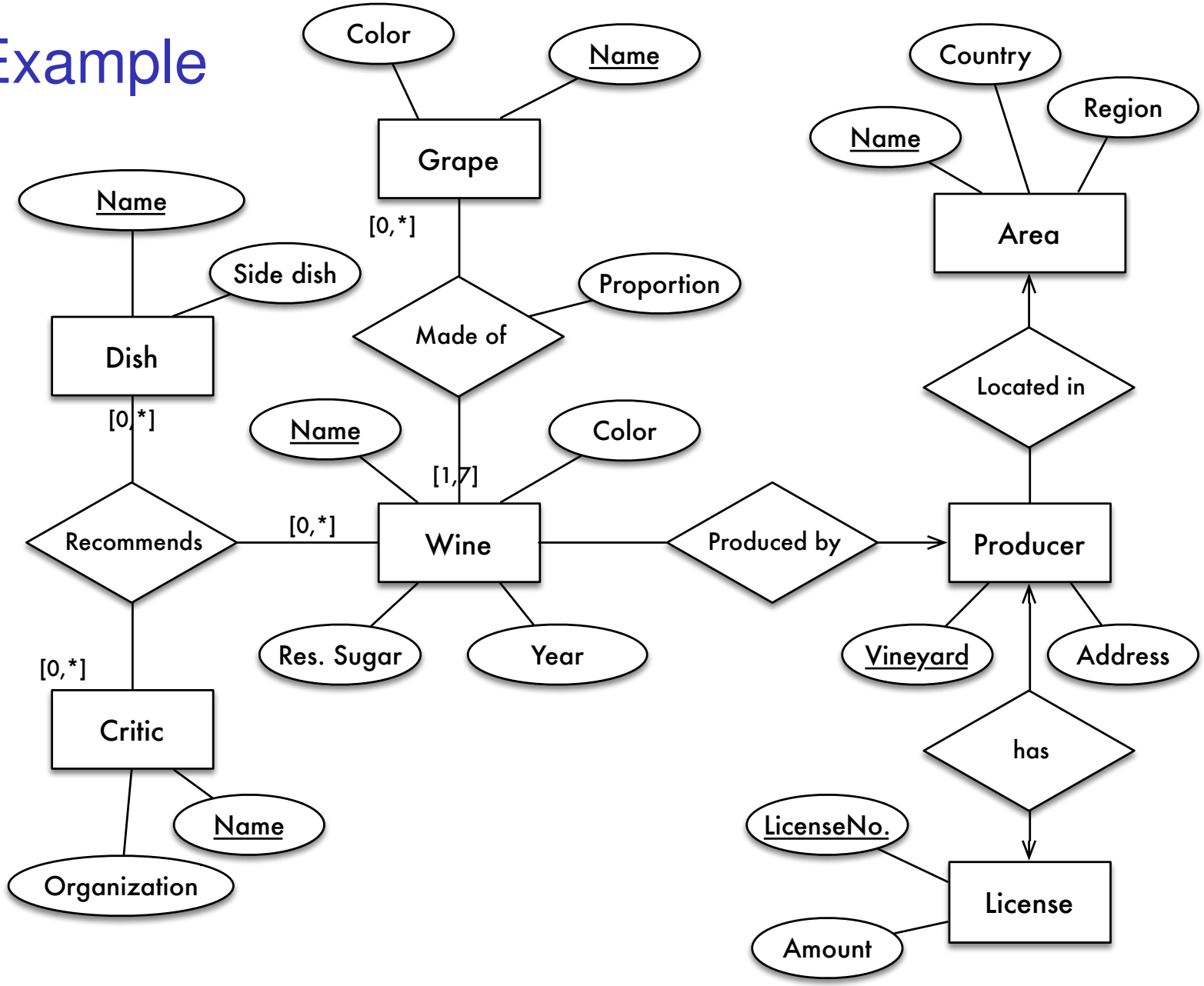
The ER Model

Entity: object of the real or a virtual world, about which information is to be stored, e.g., **Products** (wine, catalog), winemaker or critic; but also information about events, e.g., **Orders**

Relationship: describes a relationship between entities, e.g., a customer **orders** a wine or wine is being **offered** by a winemaker

Attribute: represents a property of an entity or a relationship, e.g., **Name** of customer, **Color** of a wine or **Date** of an order

ER Example


Color
Name
Grape
[0,*]
Made of
Proportion
Name
Side dish
Dish
[0,*]
Recommends
[0,*]
Name
Color
[1,7]
Wine
Res. Sugar
Year
Produced by
Country
Region
Name
Area
Located in
Producer
Vineyard
Address
has
[0,*]
Critic
Name
Organization
LicenseNo.
License
Amount

Values

- Values: primitive elements of data, which can be represented directly
- Value domains are described by datatypes, which, apart from the set of possible values, also characterize the basic operations on those values
- ER model: pre-defined primitive datatypes, such as the integers **`int`**, the character sequences **`string`**, dates **`date`** etc.
- Every datatype represents a domain, including operations and predicates on values of this domain

Entities

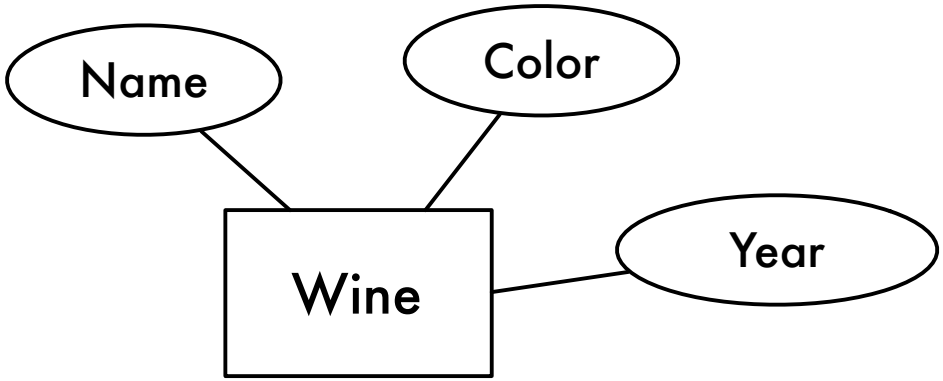
- Entities are the pieces of information to be represented in a database
- In contrast to values, entities cannot be represented directly. They can only be observed through their properties.
- Entities are grouped according to their entity types, such as $E_1, E_2 \ldots$

- Set of current entities:

$$\sigma(E_1) = \{e_1, e_2, \ldots, e_n\}$$

Attribute

- Attribute models properties of entities or relationships
- All entities of an entity type have the same kinds of properties; attributes are therefore declared for the entity type



- Textual notation $E(A_1 : D_1, \ldots, A_m : D_m)$

Key-based Identification

- Key attributes: Subset of all attributes of an entity type $E(A_1, \ldots, A_m)$

$$\{S_1, \ldots, S_k\} \subseteq \{A_1, \ldots, A_m\}$$

- In every state of the database, current values of the key attributes uniquely identify instances of the entity type E
- If multiple keys would be possible: Choice of a primary key
- Notation: Highlight by underlining:

$$E(\ldots, \underline{S_1}, \ldots, \underline{S_i}, \ldots)$$

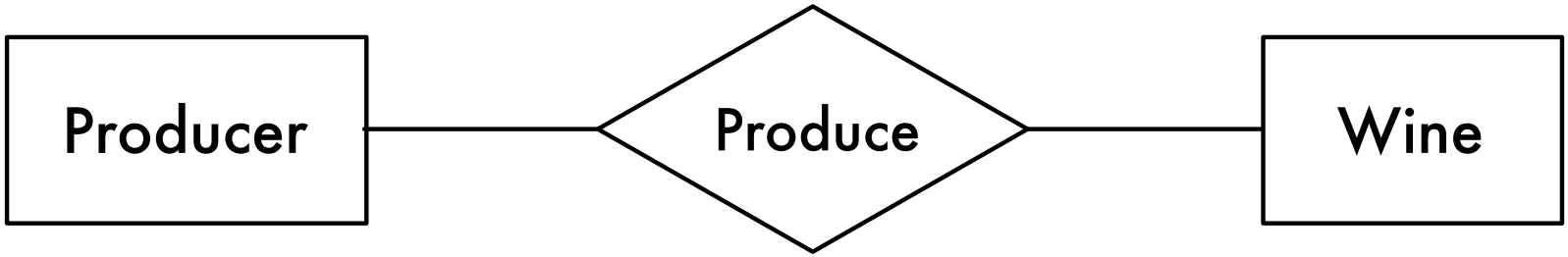
Relationship Types

- Relationships between entities are grouped into relationship types
- In general: arbitrary number $n \geq 2$ of entity types can participate in a relationship type
- Every n-ary relationship type R refers to n entity types $E_1, \ldots, E_n$
- Instances of a relationship type

$$\sigma(R) \subseteq \sigma(E_1) \times \sigma(E_2) \times \cdots \times \sigma(E_n)$$

Relationship Types /2

- Notation

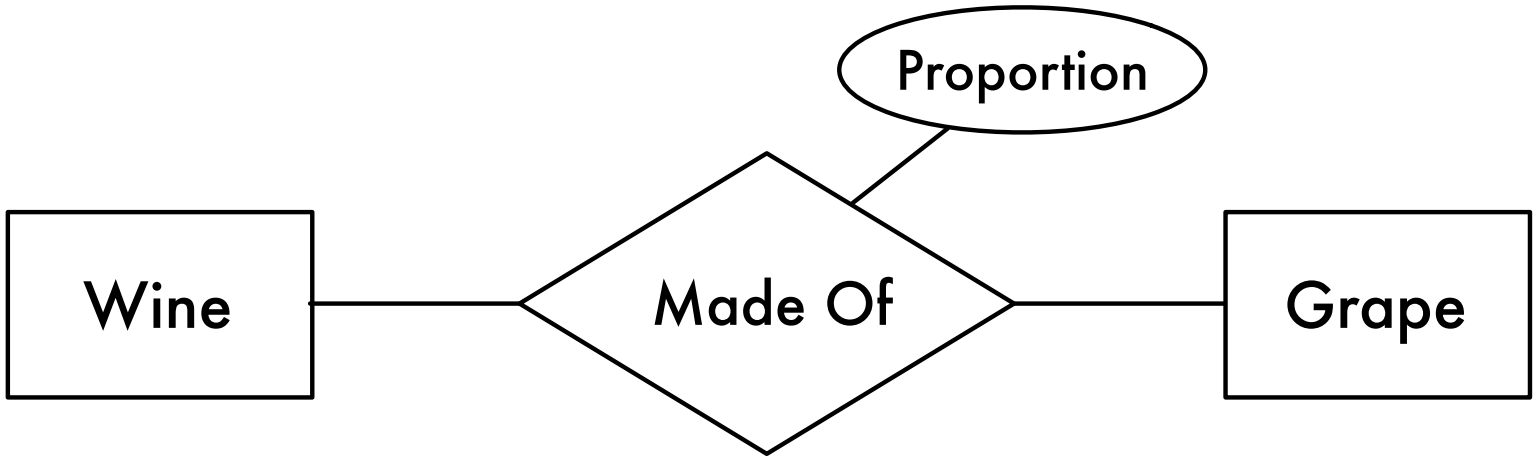


- Textual notation: $R(E_1, E_2, \ldots, E_n)$
- If an entity type participates in a relationship type multiple times: roles can be assigned

```
married(Wife:  Person, Husband:  Person)
```

Relationship Attributes

- Relationships can also have attributes
- Attribute are declared at the relationship type; this also holds for the set of possible values $\rightsquigarrow$ relationship attributes

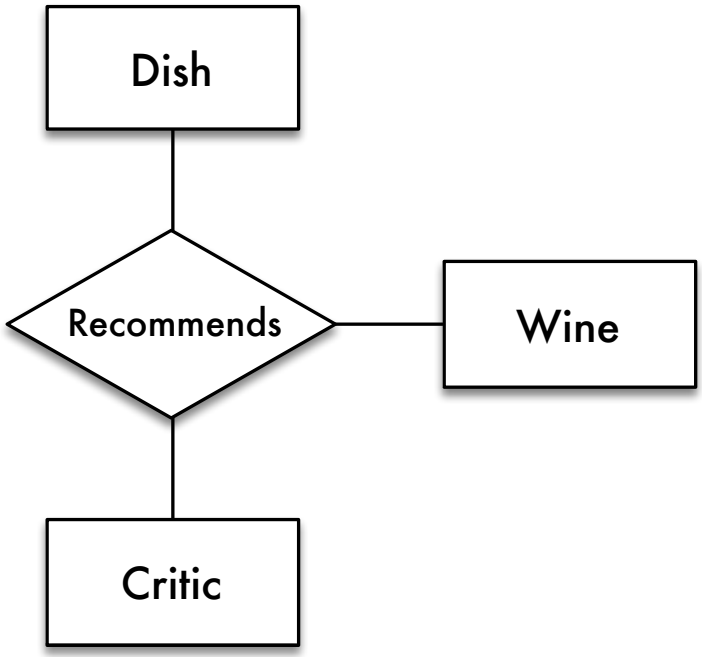


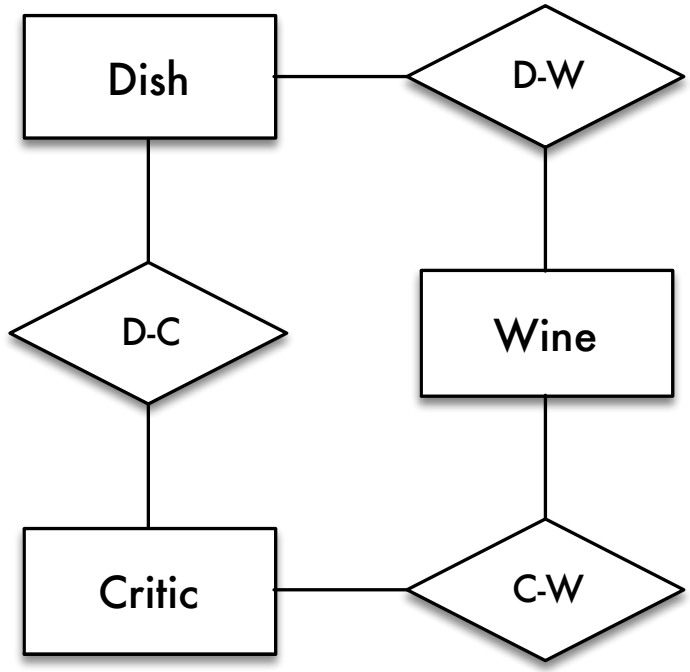
- Textual notation: $R(E_1, \ldots, E_n; A_1, \ldots, A_k)$

Characteristics of Relationships

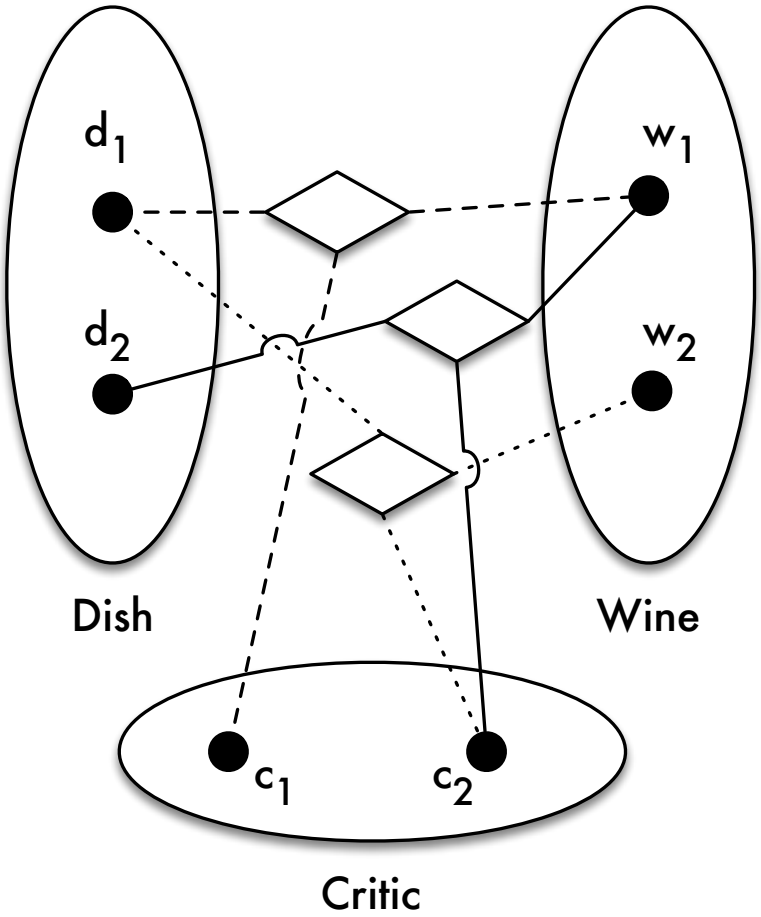
- Degree:
  - Number of participating entity types
  - Often: binary
  - Example: *Supplier* ***supplies*** *Product*
- Cardinality Constraints:
  - Number of incoming instances of an entity type
  - Typical forms: 1:1, 1:n, m:n
  - Represent integrity constraints
  - Example: ***maximum of 5*** *Products* ***per*** *Order*

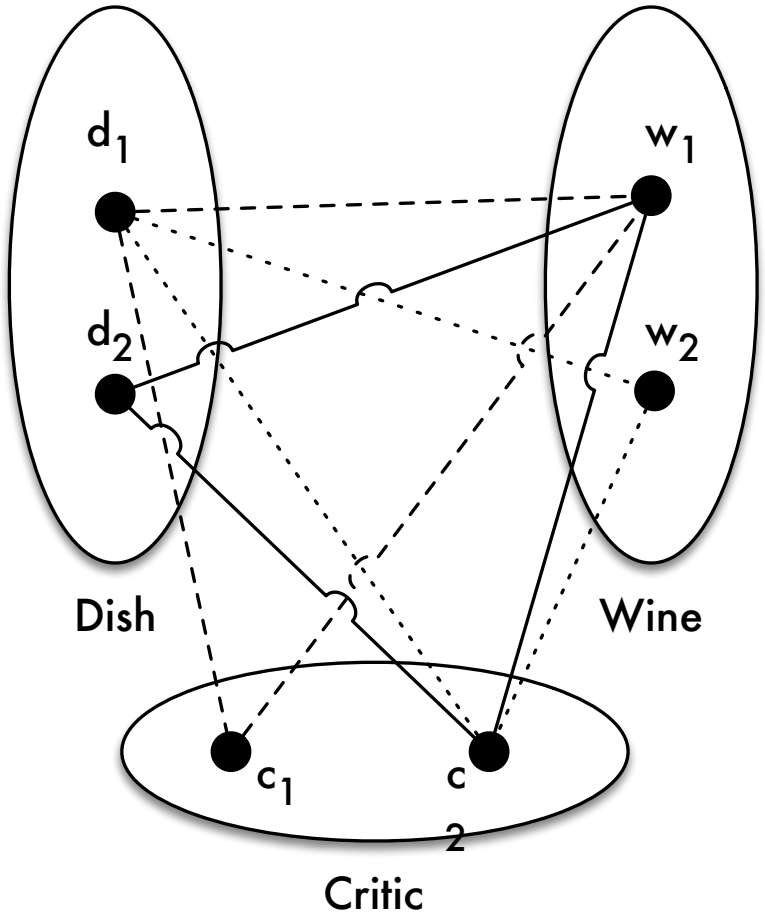
Binary vs. N-ary Relationships



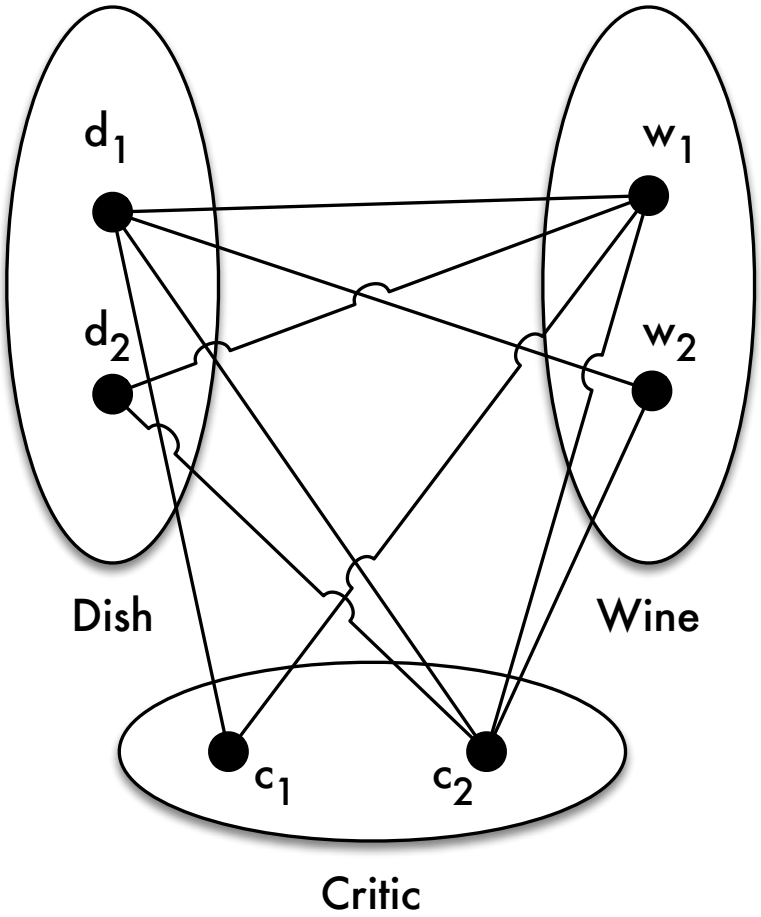


Instances in the Example





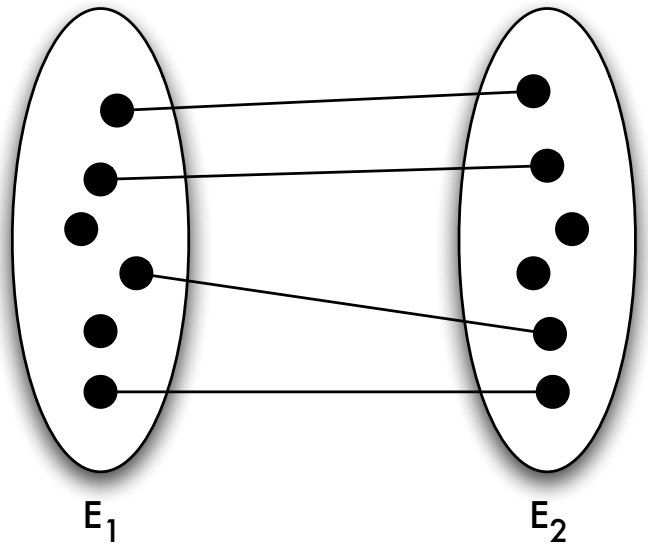
Reconstruction of Instances



- $d_1 - c_1 - w_1$
- $d_1 - c_2 - w_2$
- $d_2 - c_2 - w_1$
- But also: $d_1 - c_2 - w_1$

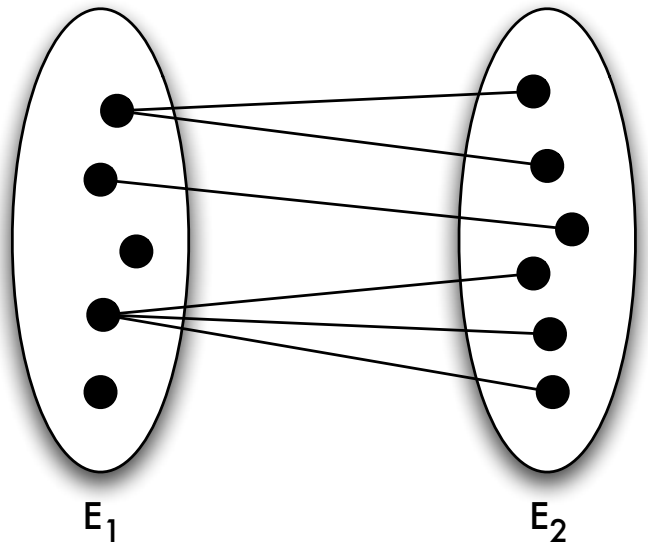
1:1-Relationships

- Every e_1 of entity type E_1 is assigned to at most one entity e_2 out of E_2 and vice versa
- Examples: *Brochure* ***describes*** *Product*, *Husband* ***is married to*** *Wife*



1:N Relationships

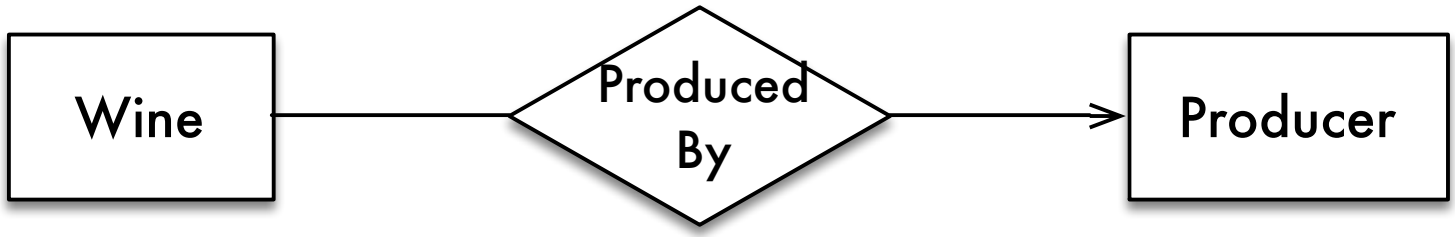
- Every entity e_1 of entity type E_1 is assigned to an arbitrary number of entities E_2, but for every entity e_2, there is at most one e_1 in E_1
- Examples: *Supplier* ***supplies*** *Product*, *Mother* ***has*** *Children*



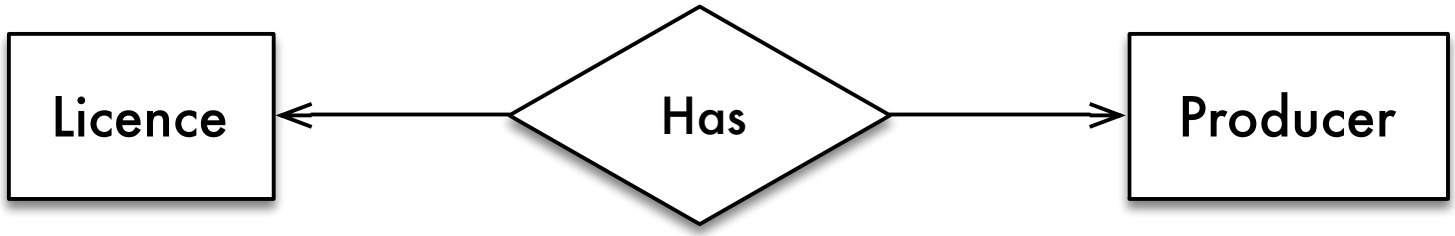
N:1 Relationship

- Inverse of 1:N, also functional relationship
- Binary relationships that define a function:
  Every entity of entity type E_1 is assigned to at most one entity of entity type E_2.

$$R : E_1 \rightarrow E_2$$

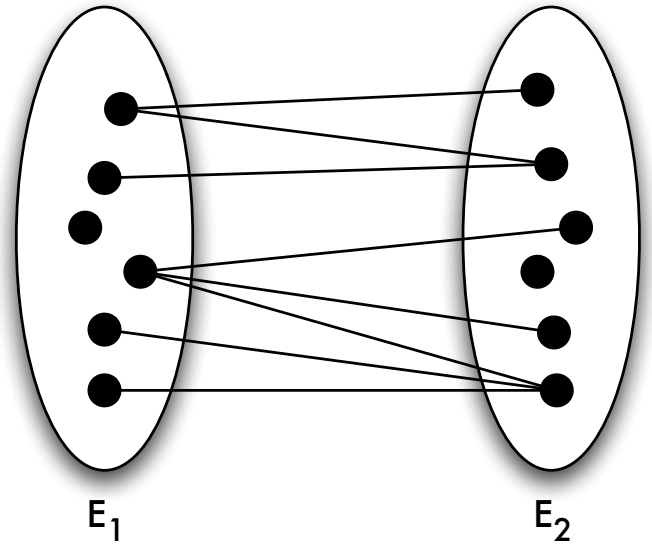


1:1 Relationship

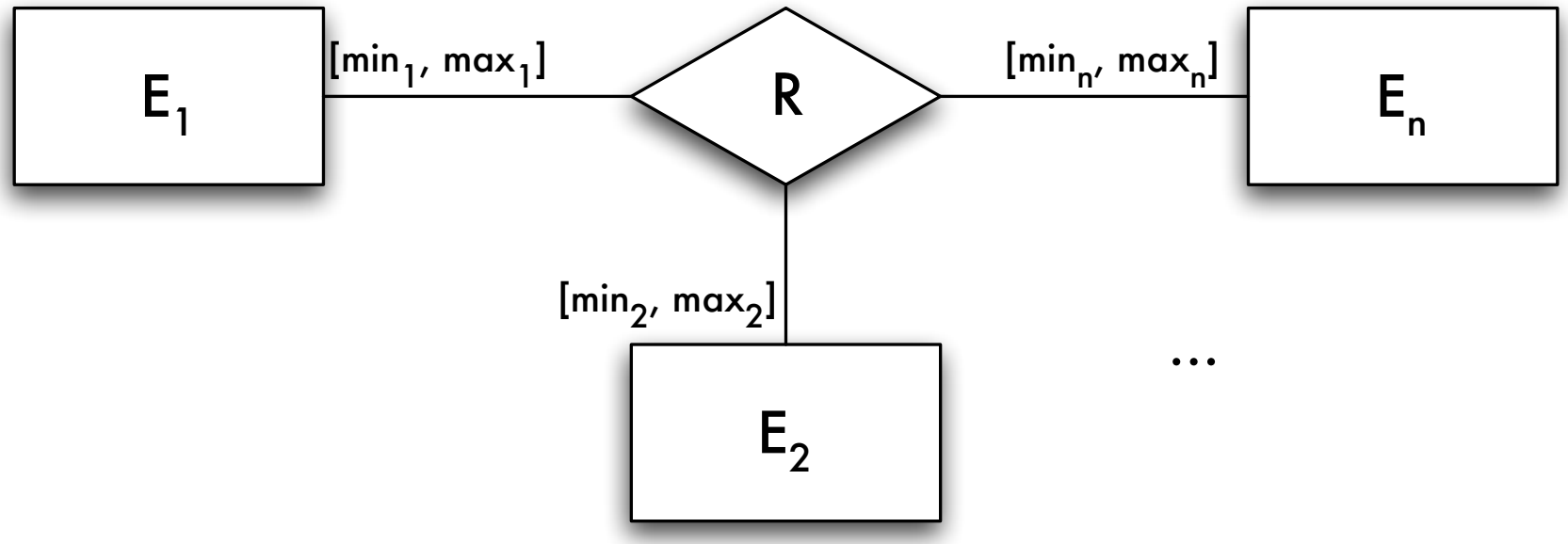


M:N Relationships

- No restrictions
- Example: *Order* ***consists of*** *Products*



[min,max] Notation



- Restricts the possible number of times an instance of an entity type can participate in a relationship by giving a minimum and a maximum value
- Notation for expressing cardinalities in a relationship type

$$R(E_1, \ldots, E_i[min_i, max_i], \ldots, E_n)$$

- Cardinality constraints: $min_i \leq |\{r \mid r \in R \wedge r.E_i = e_i\}| \leq max_i$
- Special notation for max_i is $*$

Expressing Cardinalities

- $[0, *]$ means “no restrictions” (default)
- $R(E_1[0, 1], E_2)$ corresponds to a (partial) functional relationship $R : E_1 \rightarrow E_2$, because every instance out of E_1 is assigned to at most one instance out of E_2
- Total functional relationships are modelled by $R(E_1[1, 1], E_2)$

Expressing Cardinalities: Examples

- Partial functional relationship

  `stored_on(Product[0,1],Shelf[0,3])`

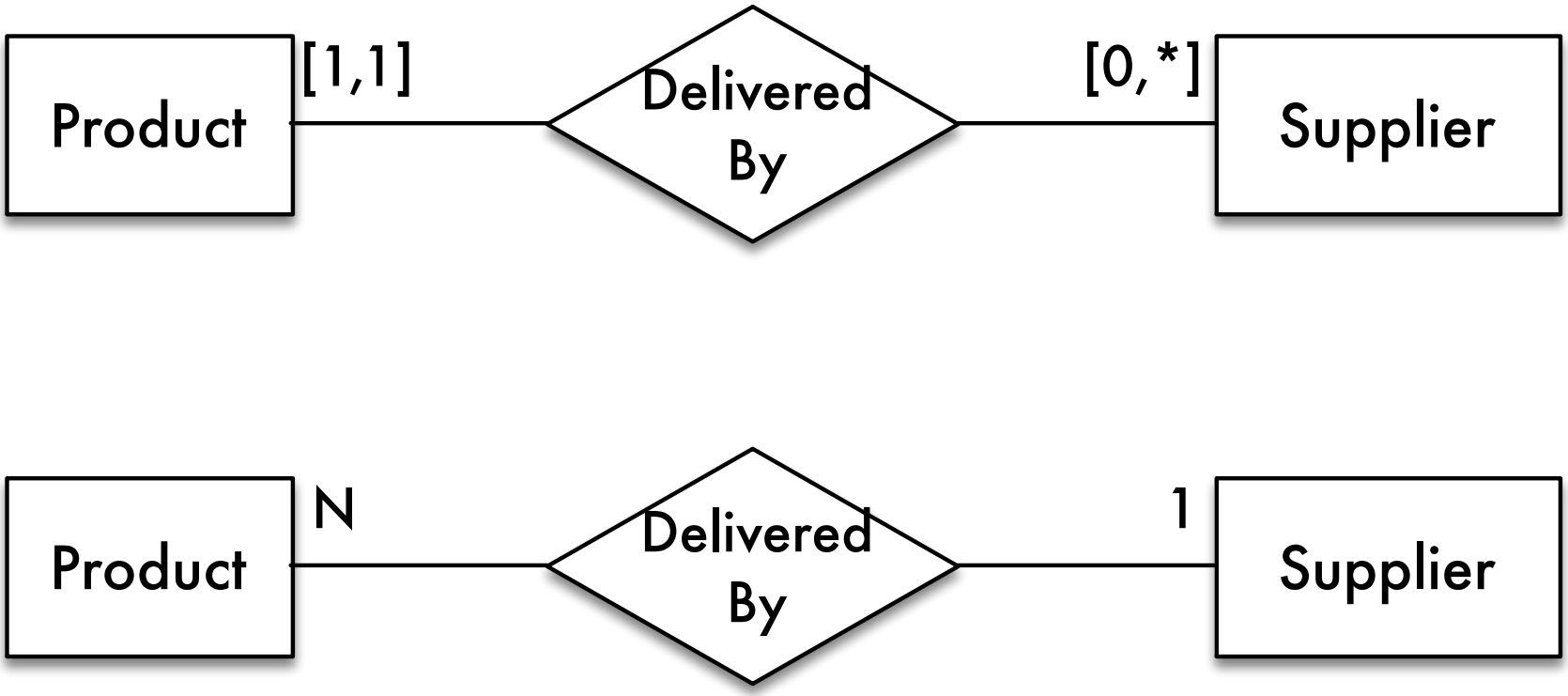
  "Every product in the warehouse is stored on one shelf. However, products that are currently out of stock are not assigned to a shelf. At most three products can share the same shelf."

- Total functional relationship

  `supplies(Supplier[0,*],Product[1,1])`

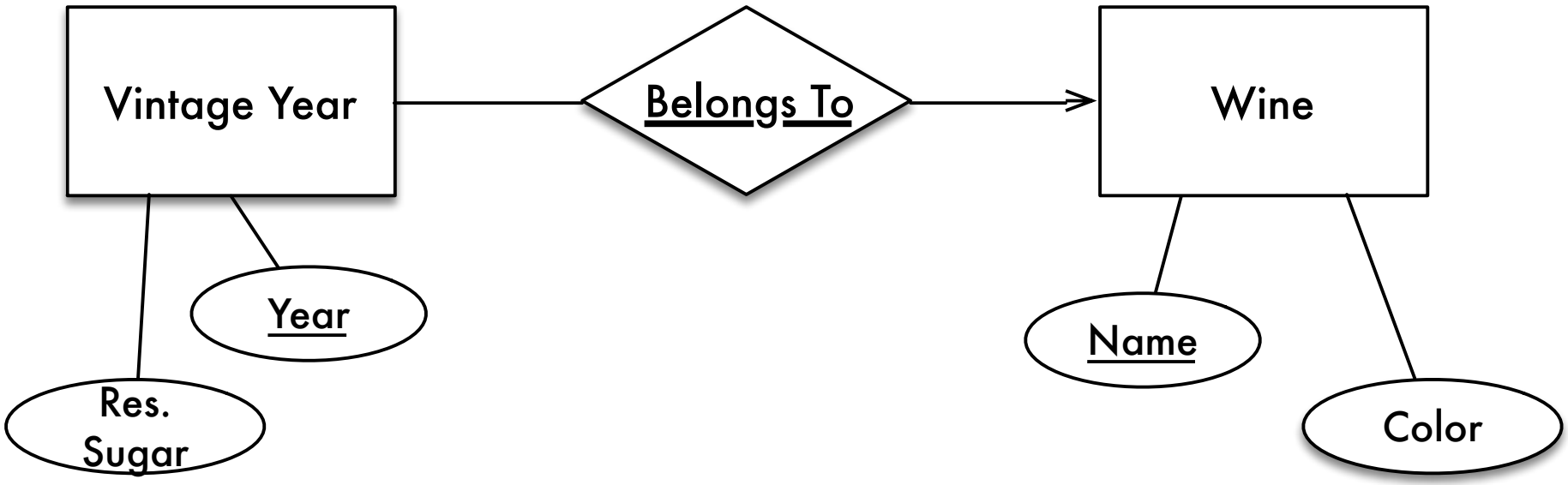
  "Every product is supplied by exactly one supplier. However, a supplier can very well supply more than one product."

Alternative Ways to Express Cardinalities



Dependent Entity Types

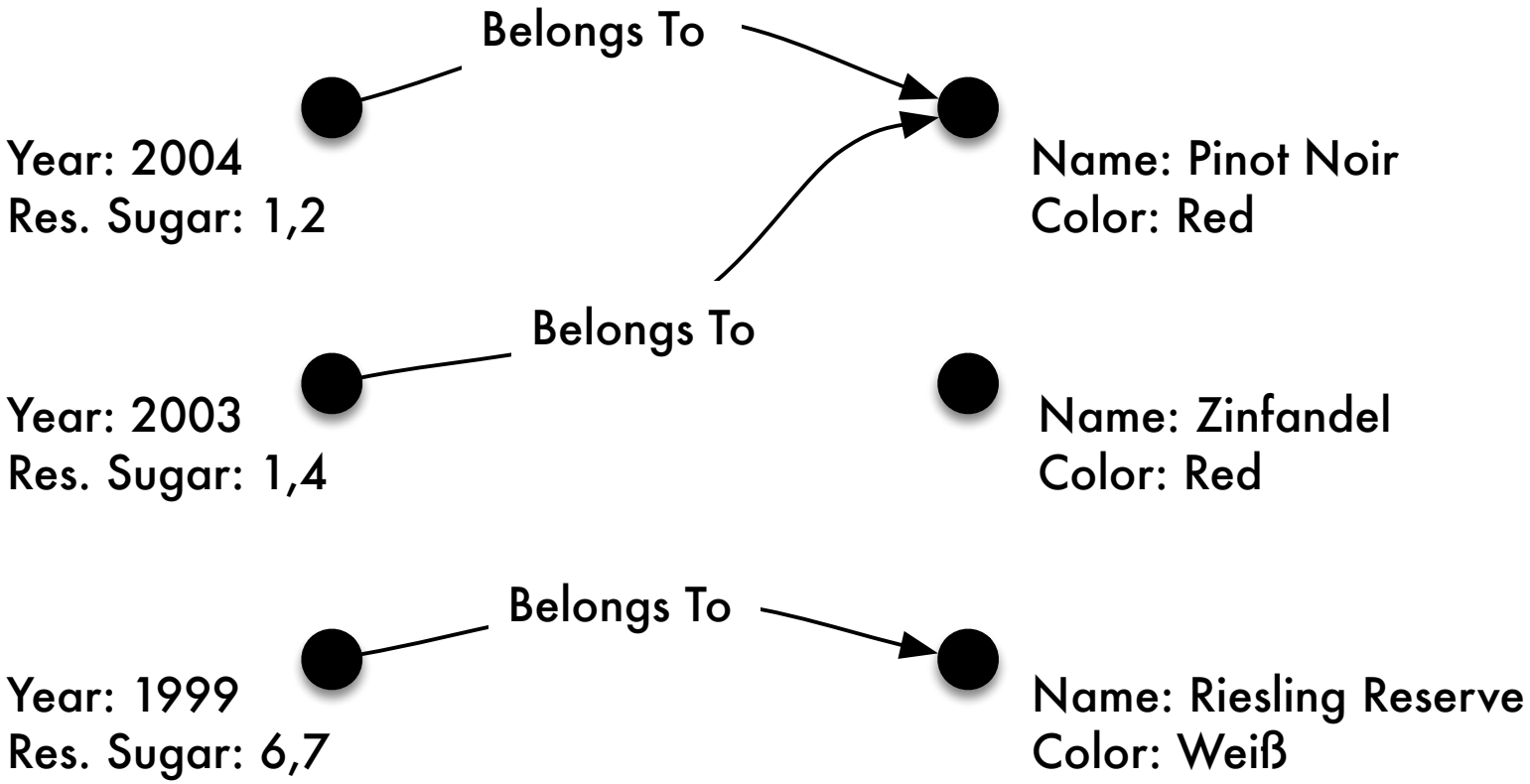
- *Dependent Entity Type*: Identification through functional relationship



- Dependent entities in the ER model: Functional relationship used as key

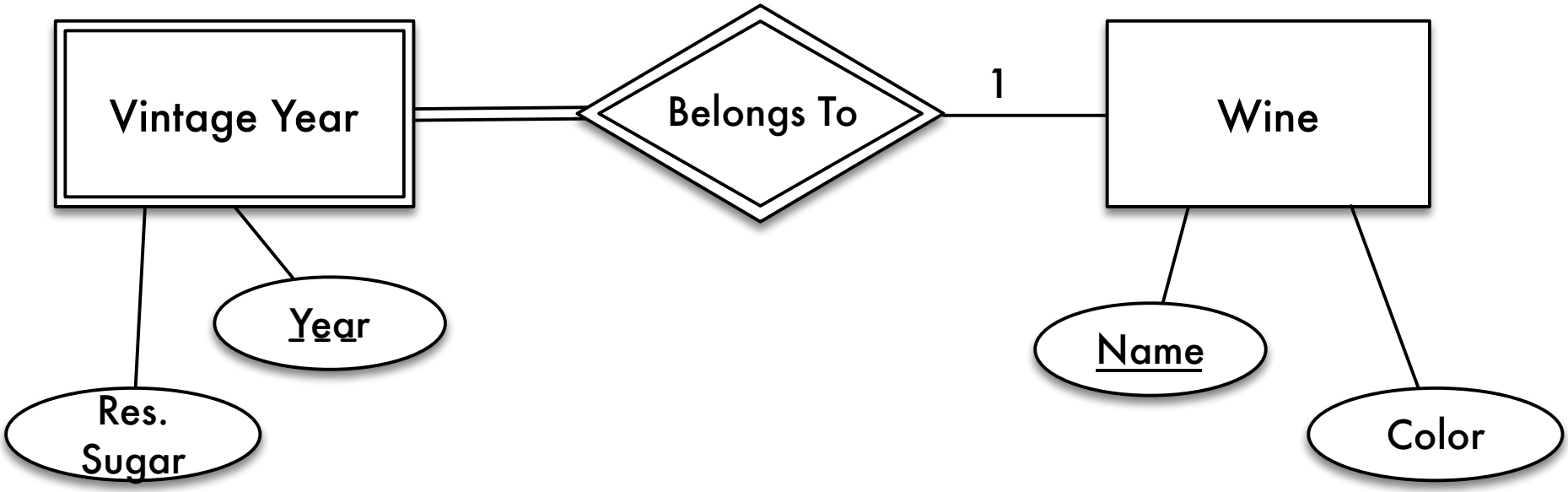
Dependent Entity Types /2

- Possible instantiations for dependent entities



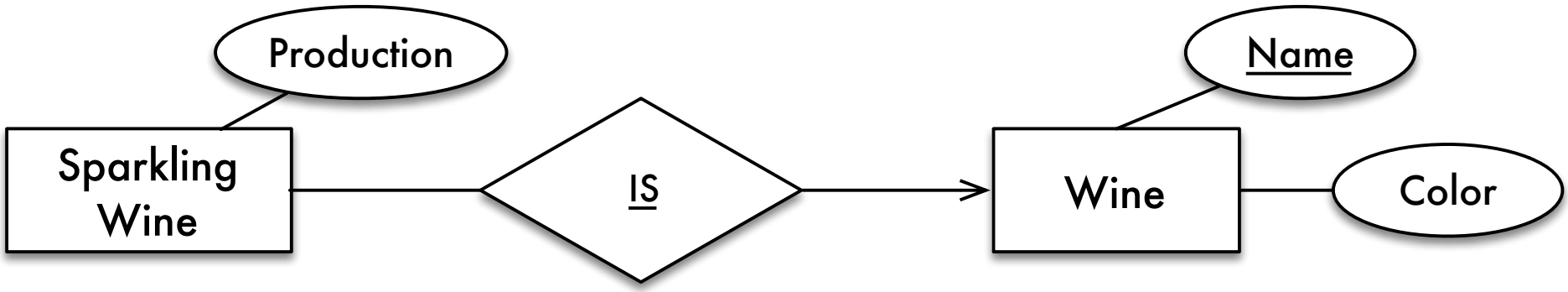
Dependent Entity Types /3

- Alternative notation



The IS-A Relationship

- Specialization/generalization relationship or IS-A relationship
- Textual notation: E_1 IS-A E_2
- IS-A relationship semantically corresponds to an injective functional relationship



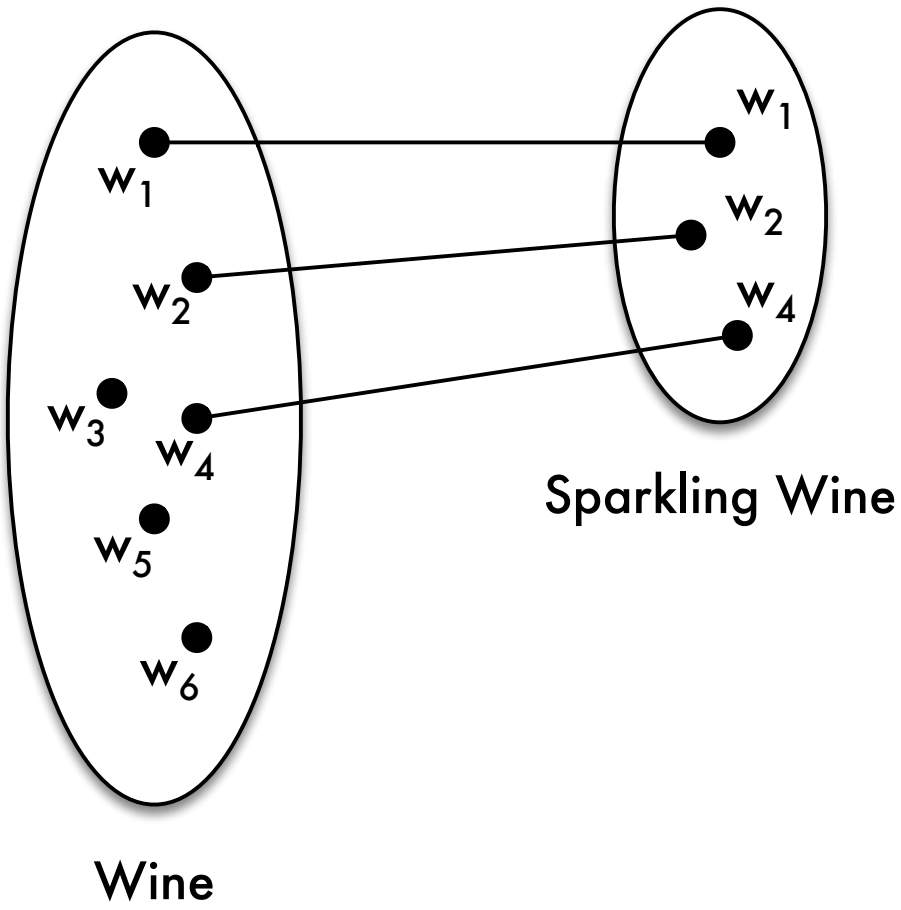
Properties of the IS-A Relationship

- Every sparkling wine instance is assigned to exactly one wine instance
  $\rightsquigarrow$ sparkling wine instances are identified by their functional IS-A relationship
- Not every wine is a sparkling wine
- Attributes of the entity type `Wine` also apply to sparkling wines: "inherited" attributes

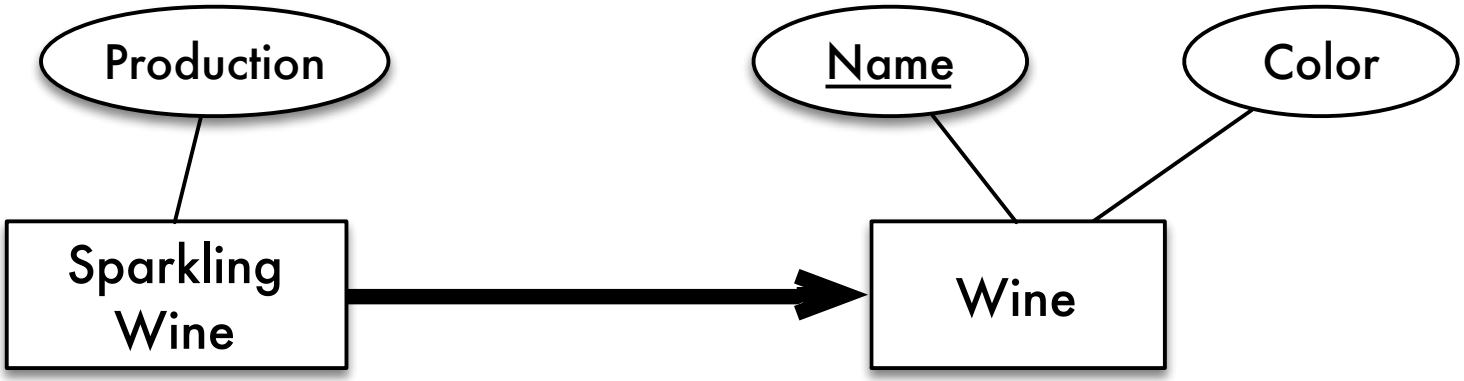
$$\texttt{Sparkling\_wine}(\underbrace{\texttt{Name},\texttt{Color}}_{\textit{of Wine}},\texttt{Production})$$

- Not only attribute declarations are inherited, but also the current values of each instance

Instantiations of IS-A Relationship



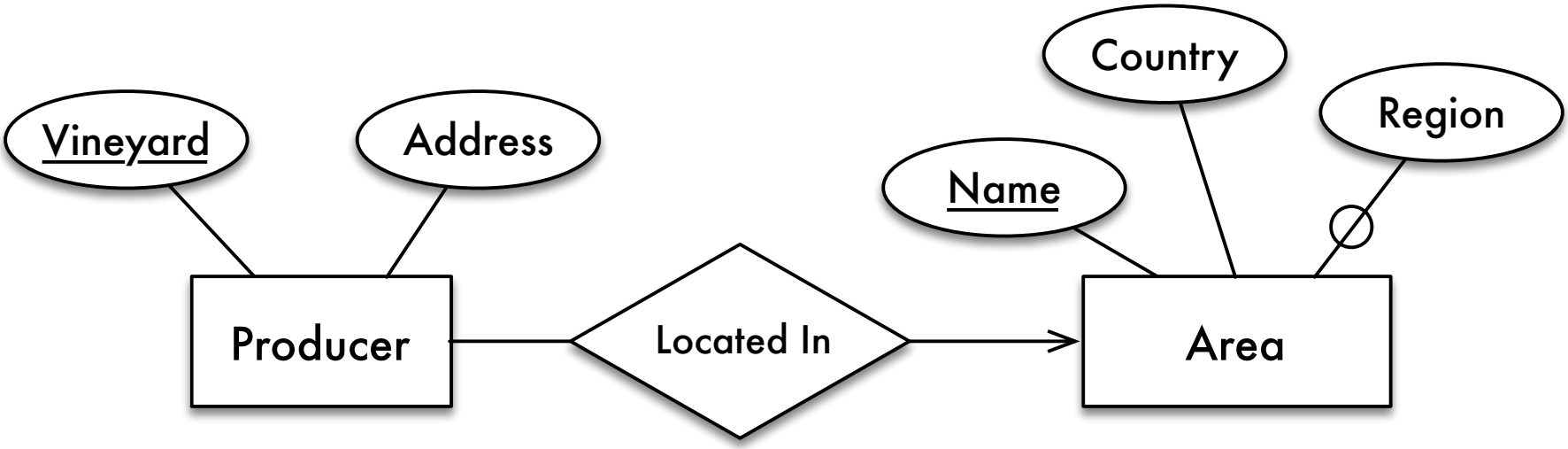
Alternative Notation for IS-A Relationship



Expressing Cardinalities: IS-A

- It holds for every relationship E_1 IS-A E_2 that: IS-A$(E_1[1,1], E_2[0,1])$
- Every instance of E_1 participates exactly once in the IS-A relationship, whereas instances of the supertype E_2 do not have to participate
- This does not affect aspects like attribute inheritance

Optionality of Attributes



Overview of Concepts

Term	Informal Meaning
Entity	The piece of information to be represented
Entity type	Grouping of entities with the same properties
Relationship type	Grouping of relationships between entities
Attribute	Property value of an entity or a relationship
Key	Identifying property of an entity
Cardinalities	Restrict relationship types with regards to the number of times an entity can participate in a relationship
Degree	Number of entity types that participate in a relationship type
Functional relationship	Relationship Type with functional property
Dependent entities	Entities that cannot exist independently from other entities
IS-A relationship	Specialization of entity types
Optionality	Attribute or functional relationships as partial functions

Summary

- Database model, database schema, database (instance)
- Entity-relationship model
- Further concepts of the ER model
- *Based on: chapter 3 in Datenbanken - Konzepte und Sprachen von Gunter Saake, Kai-Uwe Sattler und Andreas Heuer* and chapter 7 in Fundamentals of Database Systems by Ramez Elmasri and Shamkant B. Navathe

Control Questions

- What defines a database model? What is the distinction between model an schema?
- Which concepts does the ER model define?
- Which properties characterize relationship types?
- How are dependent entity types different from regular entity types?