

CSCI235 Database Systems

BSON Design

Dr Janusz R. Getta

School of Computing and Information Technology -
University of Wollongong

BSON Design

Outline

Class

Optional attribute

Multivalued attribute

Qualification

One-to-one association

One-to-many association

Many-to-many association

Generalization

Implementation of network structures

Example

Class

Conceptual schema

CLASS A	
attribute 1	ID
...	ID
attribute k	ID
attribute m	
...	
attribute n	

Logical schema

"CLASS A"
"_id": value("attribute 1")+...+value("attribute k")
"attribute 1"
...
"attribute k"
"attribute m"
...
"attribute n"

In HTML view press 'p' to see the lecture notes

[TOP](#)

Created by Janusz R. Getta, CSCI235 Database Systems, Autumn 2024

3/55

Class

JSON Schema

```
db.createCollection("class_a",
  { "validator":{$jsonSchema:
    {"bsonType":"object",
      "properties":{"_id":{"bsonType":"string"},
        "CLASS A":{"bsonType":"object",
          "properties":{"attribute 1":{"bsonType": ... },
            ...
            "attribute k":{"bsonType": ... },
            "attribute m":{"bsonType": ... },
            ...
            "attribute n":{"bsonType": ... } }},
          "required":["attribute 1",...,"attribute k","attribute m",...,"attribute n"],
          "additionalProperties":false }
        },
      "required":["_id","CLASS A"],
      "additionalProperties":false
    } } } );
```

\$jsonSchema validator

Class

Example

STUDENT	
snumber	ID
first name	
last name	

\$jsonSchema validator

```
db.createCollection("student",
  { "validator":{$jsonSchema:
{"bsonType":"object",
  "properties":{"_id":{"bsonType":"int"},
    "STUDENT":{"bsonType":"object",
      "properties":{"snumber":{"bsonType":"int" },
        "first name":{"bsonType":"string" },
        "last name":{"bsonType":"string" } }},
      "required":["snumber","first name","last name"],
      "additionalProperties":false }
    },
  "required":["_id","STUDENT"],
  "additionalProperties":false
} } } );
```

Class

Example

```
db.student.insert({"_id":NumberInt("1234567"),
                  "STUDENT":{"snumber":NumberInt("1234567"),
                             "first name":"Harry",
                             "last name":"Potter"}
                  } );
```

STUDENT

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

[Generalization](#)

[Implementation of network structures](#)

[Example](#)

Optional attribute

Conceptual schema

CLASS A	
attribute 1	ID
...	ID
attribute k	ID
attribute m	
...	
attribute n	
attribute p	[0..1]

Logical schema

"CLASS A"
"_id": value("attribute 1")+...+value("attribute k")
"attribute 1"
...
"attribute k"
"attribute m"
...
"attribute n"
"attribute p" [0..1]

Optional attribute

JSON Schema

```
db.createCollection("class_a",
  { "validator":{$jsonSchema:
    {"bsonType":"object",
      "properties":{"_id":{"bsonType":"string"},
        "CLASS A":{"bsonType":"object",
          "properties":{"attribute 1":{"bsonType": ... },
            ...
            "attribute k":{"bsonType": ... },
            "attribute m":{"bsonType": ... },
            ...
            "attribute n":{"bsonType": ... },
            "attribute p":{"bsonType": ... } }},
          "required":["attribute 1",...,"attribute k","attribute m",...,"attribute n"],
          "additionalProperties":false }
        },
      "required":["_id","CLASS A"],
      "additionalProperties":false
    } } } );
```

\$jsonSchema validator

Optional attribute

Example

STUDENT	
snumber	ID
first name	
last name	
date of birth [0..1]	

```
db.createCollection("student",
  { "validator":{$jsonSchema:
{"bsonType":"object",
  "properties":{"_id":{"bsonType":"int"},
    "STUDENT":{"bsonType":"object",
      "properties":{"snumber":{"bsonType":"int" },
        "first name":{"bsonType":"string" },
        "last name":{"bsonType":"string" },
        "date of birth":{"bsonType":"date"} }},
      "required":["snumber","first name","last name"],
      "additionalProperties":false }
    },
  "required":["_id","STUDENT"],
  "additionalProperties":false
} } } );
```

\$jsonSchema validator

Optional attribute

Example

```
db.student.insert({"_id":NumberInt("1234567"),  
                  "STUDENT":{"snumber":NumberInt("1234567"),  
                             "first name":"Harry",  
                             "last name":"Potter",  
                             "date of birth":Date("1999-07-07")},  
                  } );
```

STUDENT

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

[Generalization](#)

[Implementation of network structures](#)

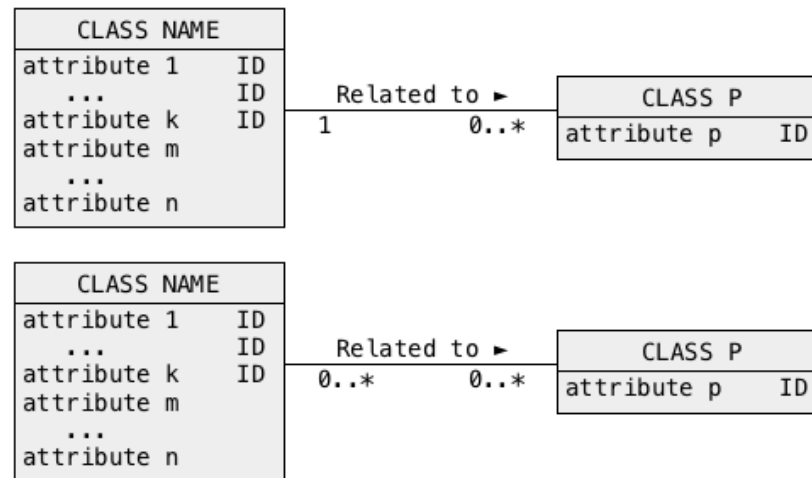
[Example](#)

Multivalued attribute

Conceptual schema

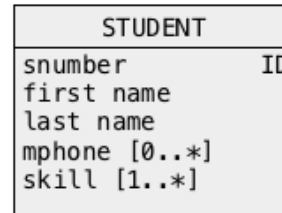
CLASS NAME	
attribute 1	ID
...	ID
attribute k	ID
attribute m	
...	
attribute n	
attribute p	[0..*]

Equivalent conceptual schemas

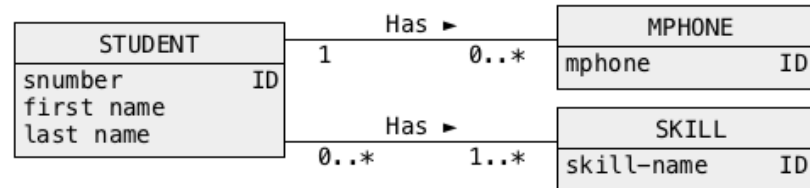


Multivalued attribute

Sample conceptual schema



Equivalent conceptual schema



BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

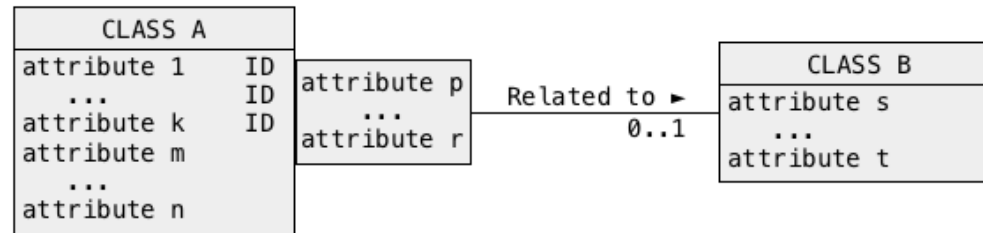
[Generalization](#)

[Implementation of network structures](#)

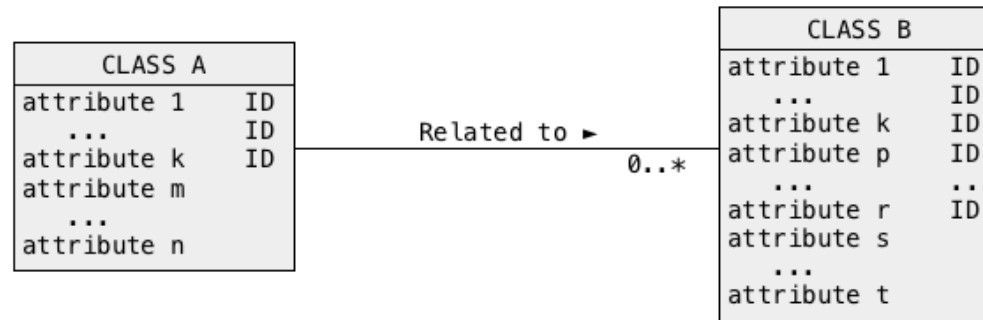
[Example](#)

Qualification

Conceptual schema

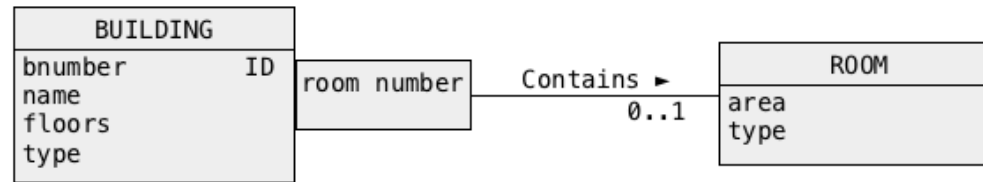


Equivalent conceptual schema

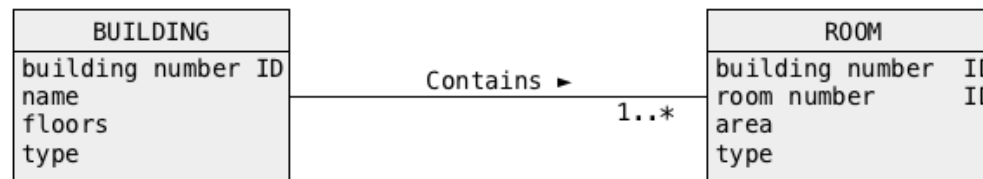


Qualification

Sample conceptual schema



Equivalent conceptual schema



BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

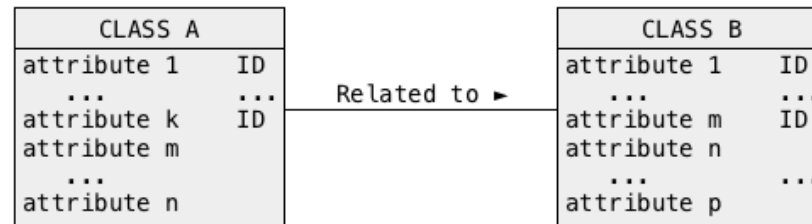
[Generalization](#)

[Implementation of network structures](#)

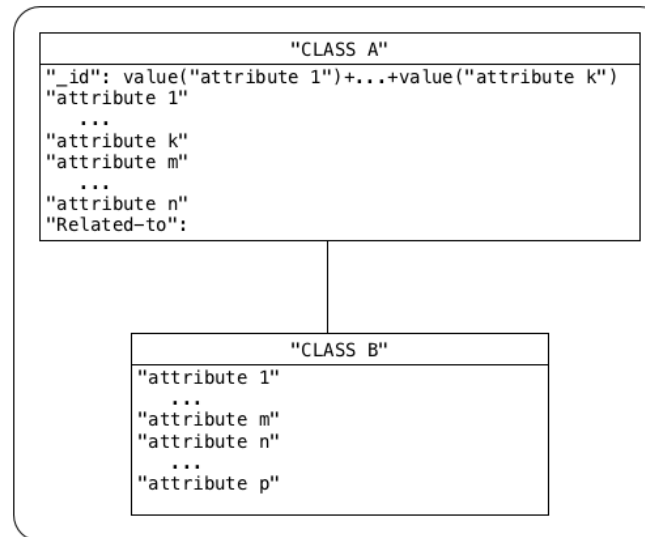
[Example](#)

One-to-one association

Conceptual schema

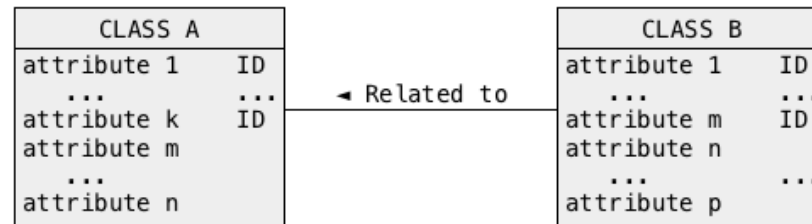


Logical schema

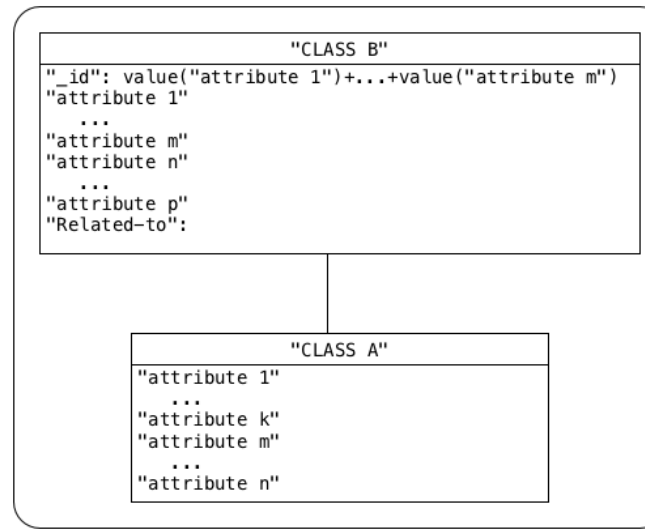


One-to-one association

Conceptual schema



Logical schema



One-to-one association

JSON Schema

```

db.createCollection("class_a",
  { "validator":{$jsonSchema:
{"bsonType":"object"
  "properties":{"_id":{"bsonType":"string"},
    "CLASS A":{"bsonType":"object",
      "properties":{"attribute 1":{"bsonType": ... },
        ...      ...      ... ,
        "attribute k":{"bsonType": ... },
        "attribute m":{"bsonType": ... },
        ...      ...      ... ,
        "attribute n":{"bsonType": ... },
        "Related-to": {"bsonType":"object",
          "properties":{"CLASS B":{"bsonType":"object",
            "properties":{"attribute 1":{"bsonType": ... },
              ...      ...      ... ,
              "attribute m":{"bsonType": ... },
              "attribute n":{"bsonType": ... },
              ...      ...      ... ,
              "attribute p":{"bsonType": ... } }},
            "required":["attribute 1",...,"attribute m","attribute n",...,"attribute p"],
            "additionalProperties":false } }},
          "required":["CLASS B"],
          "additionalProperties":false } }},
        "required":["attribute 1",...,"attribute k","attribute m",...,"attribute n","Related-to"],
        "additionalProperties":false } }},
    "required":["_id","CLASS A"],
    "additionalProperties":false
  } } } );

```

\$jsonSchema validator

One-to-one association

Example



```

db.createCollection("student",
  { "validator":{$jsonSchema:
{"bsonType":"object",
  "properties":{"_id":{"bsonType":"int"},
    "STUDENT":{"bsonType":"object",
      "properties":{"snumber":{"bsonType":"int"},
        "first name":{"bsonType":"string"},
        "last name":{"bsonType":"string"},
        "Owns":{"bsonType":"object",
          "properties":{"CAR":{"bsonType":"object",
            "properties":{"rego":{"bsonType":"string"},
              "make":{"bsonType":"string"},
              "model":{"bsonType":"string"} },
            "required":["rego","make","model"],
            "additionalProperties":false} },
          "required":["CAR"],
          "additionalProperties":false} },
        "required":["snumber","first name","last name","Owns"],
        "additionalProperties":false} },
    "required":["_id","STUDENT"],
    "additionalProperties":false
  } } } );
  
```

\$jsonSchema validator

One-to-one association

Example

```
db.student.insert({"_id":NumberInt("1234567"),
  "STUDENT":{"snumber":NumberInt("1234567"),
    "first name":"Harry",
    "last name":"Potter",
    "Owns":{"CAR":{"rego":"AL08UK",
      "make":"Rolls Royce",
      "model":"Silver Shadow"}
    }
  }
});
```

STUDENT Owns Car

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

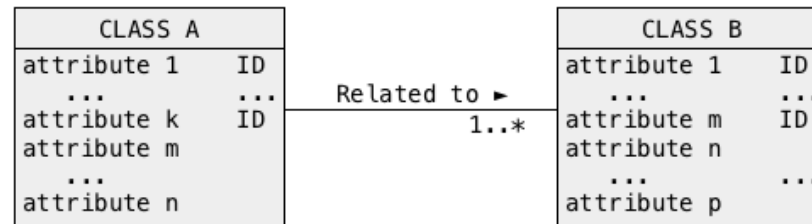
[Generalization](#)

[Implementation of network structures](#)

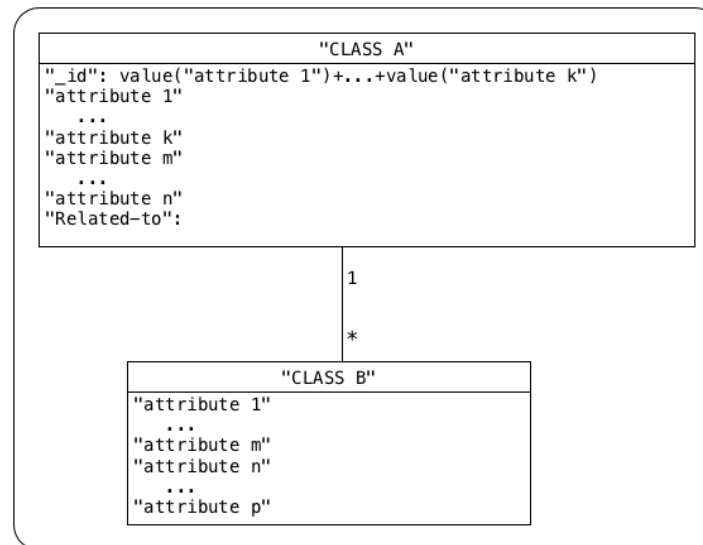
[Example](#)

One-to-many association

Conceptual schema

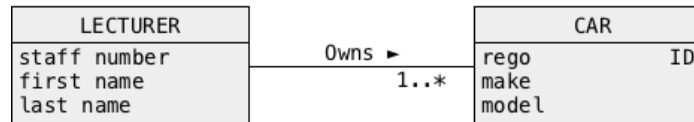


Logical schema



One-to-many association

Example



```

db.createCollection("lecturer",
  { "validator":{$jsonSchema:
{"bsonType":"object",
  "properties":{"_id":{"bsonType":"int"},
    "LECTURER":{"bsonType":"object",
      "properties":{"staff number":{"bsonType":"int"},
        "first name":{"bsonType":"string"},
        "last name":{"bsonType":"string"},
        "Owns":{"bsonType":"array",
          "items":{"bsonType":"object",
            "properties":{"CAR":{"bsonType":"object",
              "properties":{"rego":{"bsonType":"string"},
                "make":{"bsonType":"string"},
                "model":{"bsonType":"string"} },
              "required":["rego","make","model"],
              "additionalProperties":false } },
            "required":["CAR"],
            "additionalProperties":false } } },
          "required":["staff number","first name","last name","Owns"],
          "additionalProperties":false } },
    "required":["_id","LECTURER"],
    "additionalProperties":false
  } } } );
  
```

\$jsonSchema validator

One-to-many association

Example

```
db.lecturer.insert({"_id":NumberInt("007"),
  "LECTURER":{"staff number":NumberInt("007"),
    "first name":"James",
    "last name":"Bond",
    "Owns":[{"CAR":{"rego":"AL08UK",
      "make":"Rolls Royce",
      "model":"Silver Shadow"} },
    {"CAR":{"rego":"PKR856",
      "make":"Mercedes",
      "model":"SE800"} } ]
  }
});
```

LECTURER Owns CAR

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

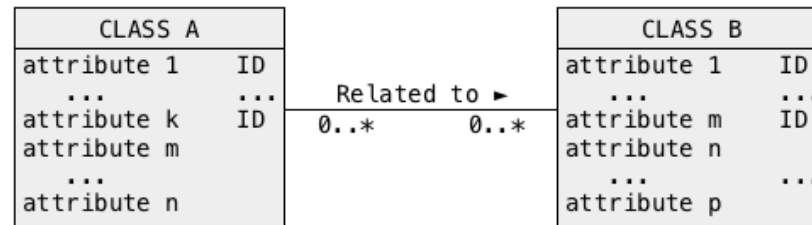
[Generalization](#)

[Implementation of network structures](#)

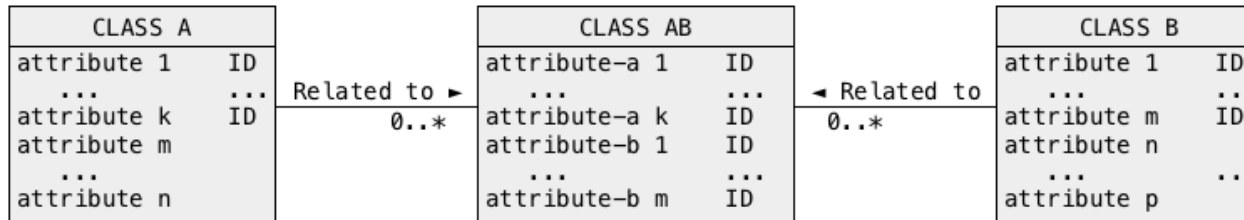
[Example](#)

Many-to-many association

Conceptual schema

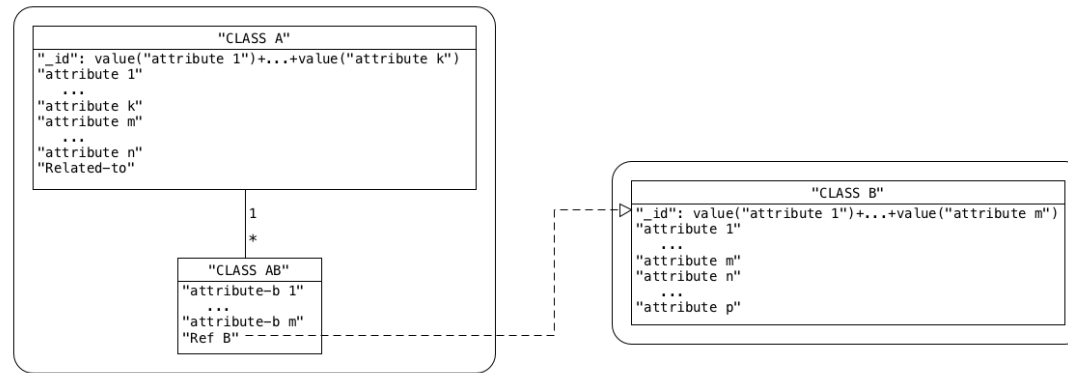


Equivalent conceptual schema

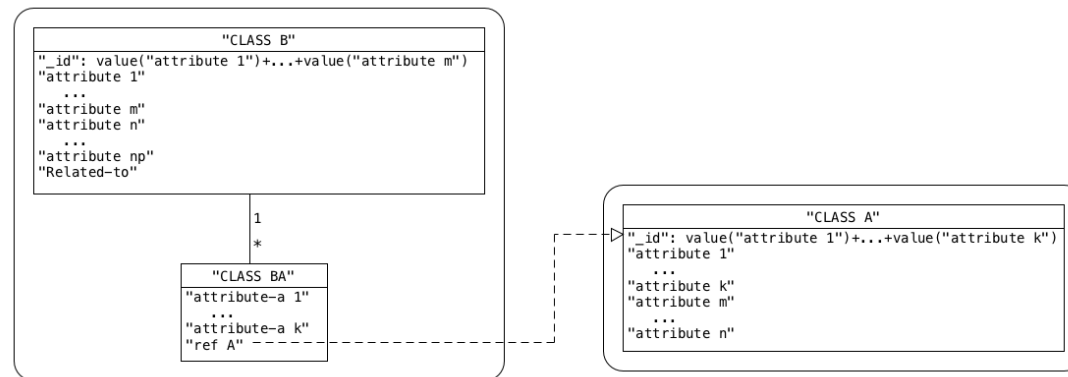


Many-to-many association

Logical schema



Equivalent logical schema



Many-to-many association

JSON Schema

```

db.createCollection("class_a",
  { "validator":{$jsonSchema:
{"bsonType":"object"
  "properties":{"_id":{"bsonType":"string"},
    "CLASS A":{"bsonType":"object",
      "properties":{"attribute 1":{"bsonType": ... },
        ...      ...      ... ,
        "attribute k":{"bsonType": ... },
        "attribute m":{"bsonType": ... },
        ...      ...      ... ,
        "attribute n":{"bsonType": ... },
        "Related-to": {"bsonType":"array",
          "items":{"bsonType":"object",
            "properties":{"CLASS AB":{"bsonType":"object",
              "properties":{"attribute-b 1":{"bsonType": ... },
                ...      ...      ... ,
                "attribute-b m":{"bsonType": ... },
                "Ref B"      :{"bsonType": ... } },
              "required":["attribute-b 1",...,"attribute-b m","Ref B"],
              "additionalProperties":false } },
            "required":["CLASS AB"],
            "additionalProperties":false } },
          "required":["attribute 1",...,"attribute k","attribute m",...,"attribute n","Related-to"],
          "additionalProperties":false } } },
    "required":["_id","CLASS A"],
    "additionalProperties":false } } } );

```

\$jsonSchema validator

Many-to-many association

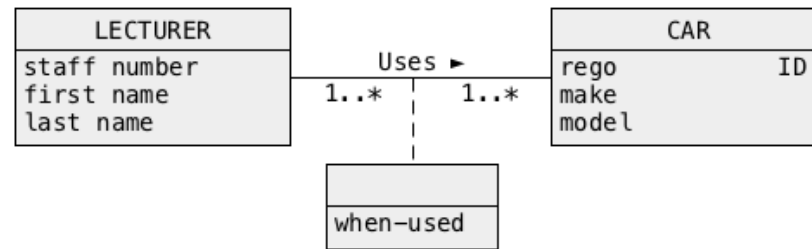
JSON Schema

```
db.createCollection("class_b",
  { "validator":{$jsonSchema:
{"bsonType":"object"
  "properties":{"_id":{"bsonType":"string"},
    "CLASS B":{"bsonType":"object",
      "properties":{"attribute 1":{"bsonType": ... },
        ...      ...      ... ,
        "attribute m":{"bsonType": ... },
        "attribute n":{"bsonType": ... },
        ...      ...      ... ,
        "attribute p":{"bsonType": ... } }},
      "required":["attribute 1",...,"attribute m","attribute n",...,"attribute p"],
      "additionalProperties":false} }},
  "required":["_id","CLASS B"],
  "additionalProperties":false } } } );
```

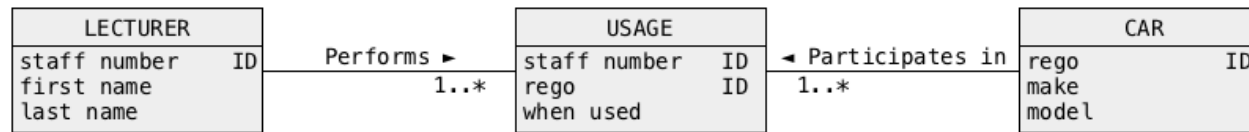
\$jsonSchema validator

Many-to-many association

Example



Equivalent conceptual schema



Many-to-many association

Example

```
db.createCollection("lecturer",
  { "validator":{$jsonSchema:
{"bsonType":"object"
  "properties":{"_id":{"bsonType":"string"},
    "LECTURER":{"bsonType":"object",
      "properties":{"staff number":{"bsonType":"int" },
        "first name":{"bsonType":"string"},
        "last name":{"bsonType":"string" },
        "Performs": {"bsonType":"array",
          "items":{"bsonType":"object",
            "properties":{"USAGE":{"bsonType":"object",
              "properties":{"rego":{"bsonType":"string"},
                "when used":{"bsonType":"date" } },
              "required":["rego","when used"],
              "additionalProperties":false } },
            "required":["USAGE"],
            "additionalProperties":false } },
          "required":["staff number","first name","last name","Performs"],
          "additionalProperties":false} } },
    "required":["_id","LECTURER"],
    "additionalProperties":false } } } );
```

Class

Example

```
db.createCollection("car",
  { "validator":{$jsonSchema:
  {"bsonType":"object",
  "properties":{"_id":{"bsonType":"string"},
  "CAR":{"bsonType":"object",
  "properties":{"rego":{"bsonType":"string" },
  "make":{"bsonType":"string" },
  "model":{"bsonType":"string" } }},
  "required":["rego","make","model"],
  "additionalProperties":false }
  },
  "required":["_id","CAR"],
  "additionalProperties":false
} } } );
```

\$jsonSchema validator

One-to-many association

Example

```
db.lecturer.insert({"_id":NumberInt("007"),
                    "LECTURER":{"staff number":NumberInt("007"),
                                "first name":"James",
                                "last name":"Bond",
                                "Performs": [ {"USAGE":{"rego":"AL08UK",
                                                       "when used":Date("2017-07-08")} },
                                              {"USAGE":{"rego":"PKR856",
                                                       "when used":Date("2017-07-09")} },
                                              {"USAGE":{"rego":"AL08UK",
                                                       "when used":Date("2017-07-09")} } ]
                    }
});
```

LECTURER Uses CAR

```
db.car.insert({"_id":"AL08UK",
              "CAR":{"rego":"AL08UK","make":"Honda","model":"Legend"}
});
```

LECTURER Uses CAR

```
db.car.insert({"_id":"PKR856",
              "CAR":{"rego":"PKR856","make":"Rolls Royce","model":"Silver Shadow"}
});
```

LECTURER Uses CAR

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

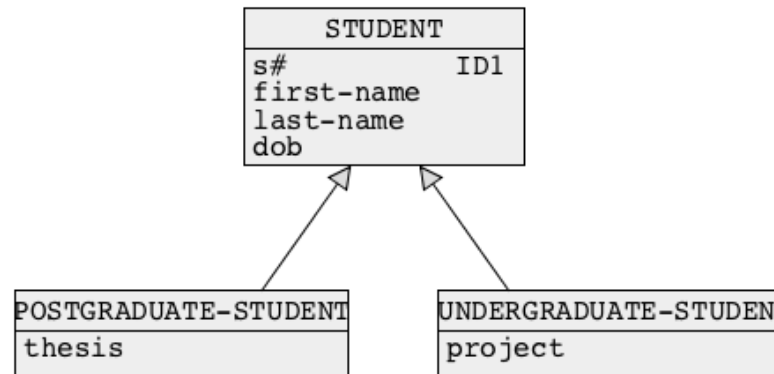
[Many-to-many association](#)

[Generalization](#)

[Implementation of network structures](#)

[Example](#)

Generalization - superset method

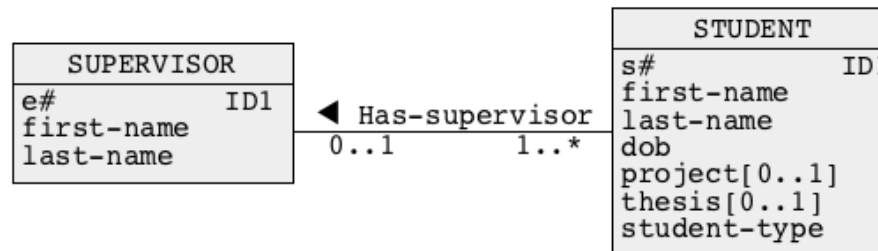


A **superset method** transforms entire generalization hierarchy into a single class of objects in the following way:

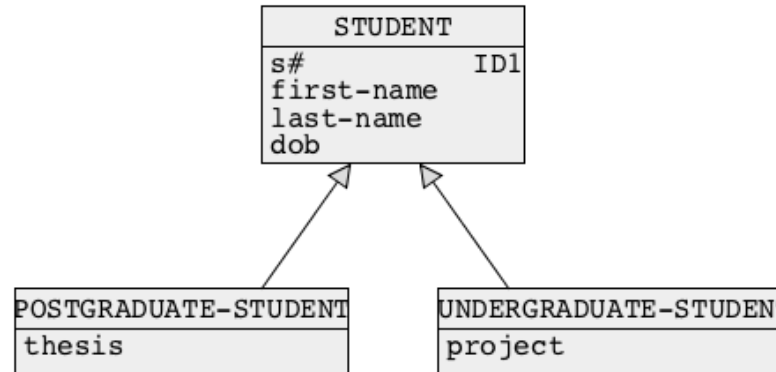
- All attributes from the classes of objects at the lowest level of generalization hierarchy are copied to an immediate higher level and become optional attributes ([0..1] tag) there, e.g. the attributes **project** and **thesis** are copied from the classes **UNDERGRADUATE-STUDENT** and **POSTGRADUATE-STUDENT** to a class **STUDENT**
- An attribute **type-of-superclass** is added to a superclass, e.g. and attribute **type-of-students** is added to a class **STUDENT**

Generalization - superset method

- All classes at the lowest level are removed
- The steps above are repeated until only one class of objects is left



Generalization - subset method



A **subset method** transforms entire generalization hierarchy into a number of classes of objects in the following way:

- All attributes from the classes of objects at the higher levels of generalization hierarchy are copied to the classes of objects at the lowest levels of generalization hierarchy e.g. the attributes **s#** and **first-name** last-name, dob are copied from a class **STUDENT** to the classes **POSTGRADUATE-STUDENT** and **UNDERGRADUATE-STUDENT**

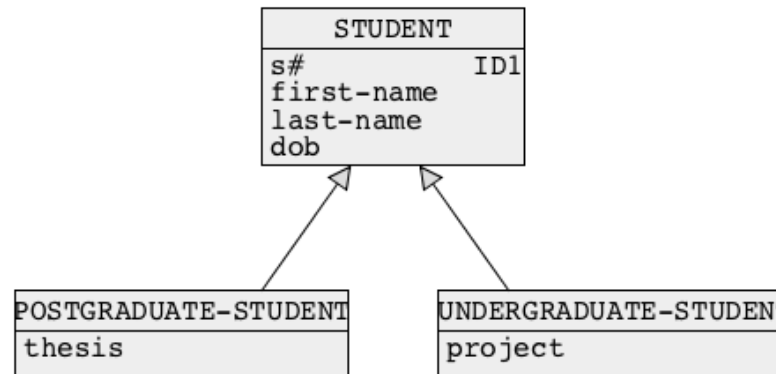
Generalization - subset method

- All classes of objects except those at the lowest levels of generalization hierarchy are removed, e.g. a class **STUDENT** is removed

POSTGRADUATE-STUDENT	
thesis	
s#	ID1
first-name	
last-name	
dob	

UNDERGRADUATE-STUDENT	
project	
s#	ID1
first-name	
last-name	
dob	

Generalization - association method

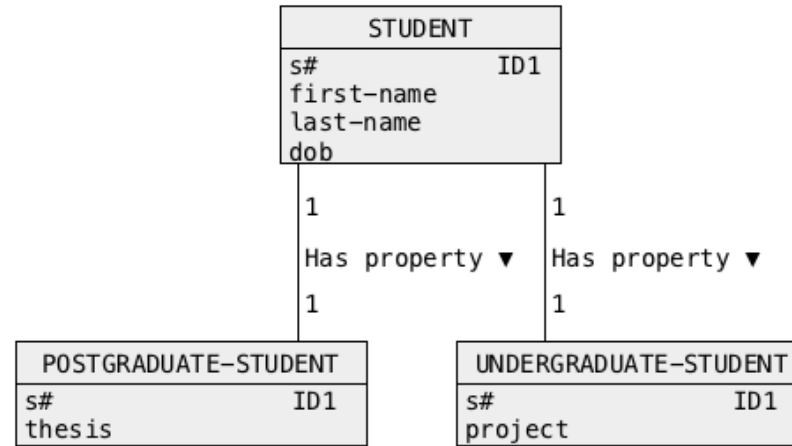


An **association method** transforms entire generalization hierarchy into a number of classes of objects in the following way:

- One of the identifiers from a superclass is copied to subclasses one level below a superclass, e.g. an attribute `s#` is copied from a class **STUDENT** to the classes **UNDEGRADUATE-STUDENT** and **POSTGRADUATE-STUDENT**

Generalization - association method

- A generalization level is removed from a diagram



BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

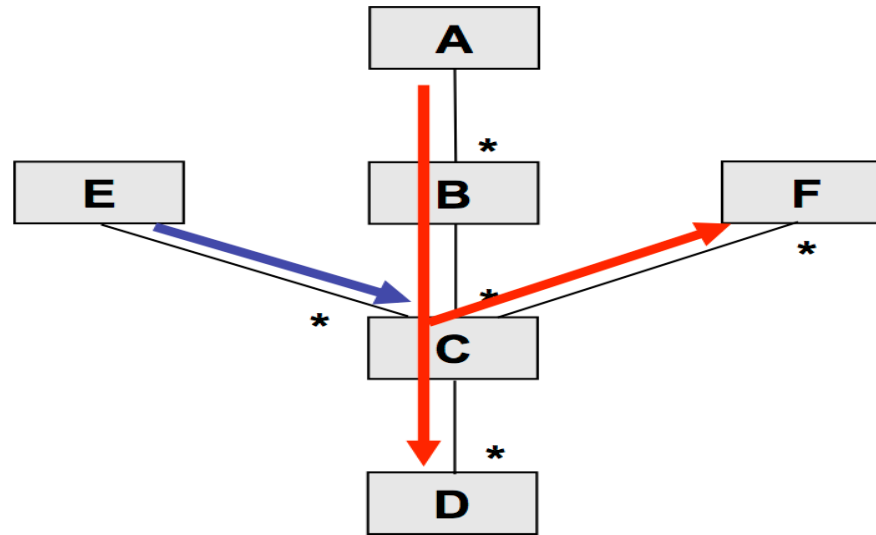
[Many-to-many association](#)

[Generalization](#)

[Implementation of network structures](#)

[Example](#)

Implementation of network structures

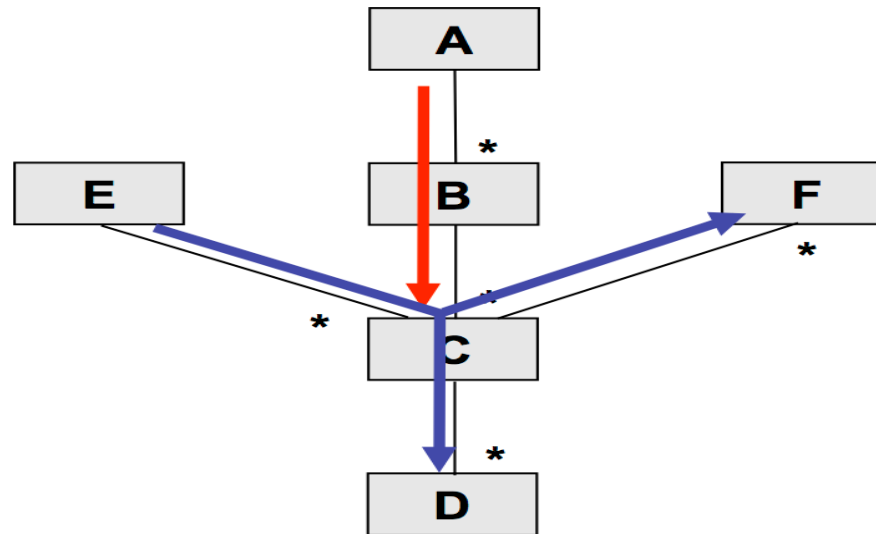


Implementation of sample hierarchy

```

"A"
...
"B"
...
"C":{"REF":"e"},
...
"D"...
"F"...
"E":{"_id":"e", ... }
  
```

Implementation of network structures

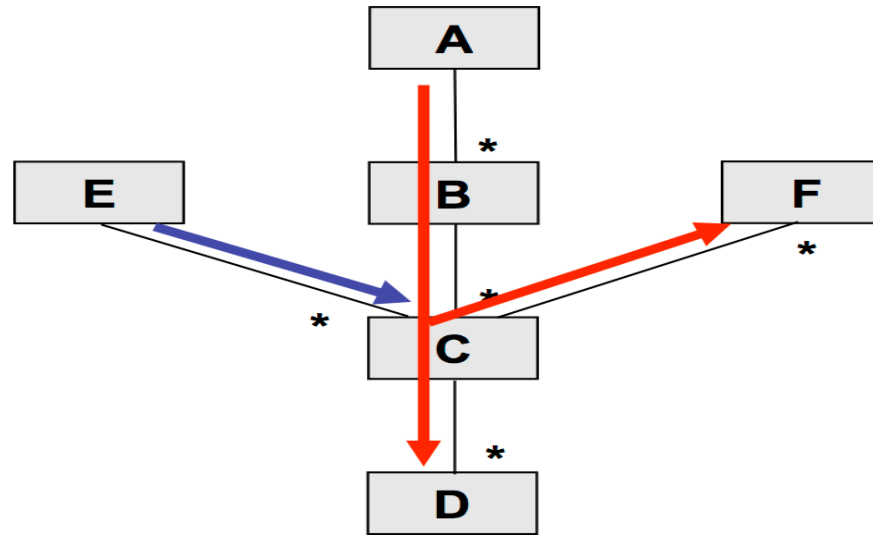


Implementation of sample hierarchy

```

"E"
...
"C":{"REF":"b"}
...
"D"
"F"
"A"
...
"B":{"ID":"b", ... }
  
```

Implementation of network structures

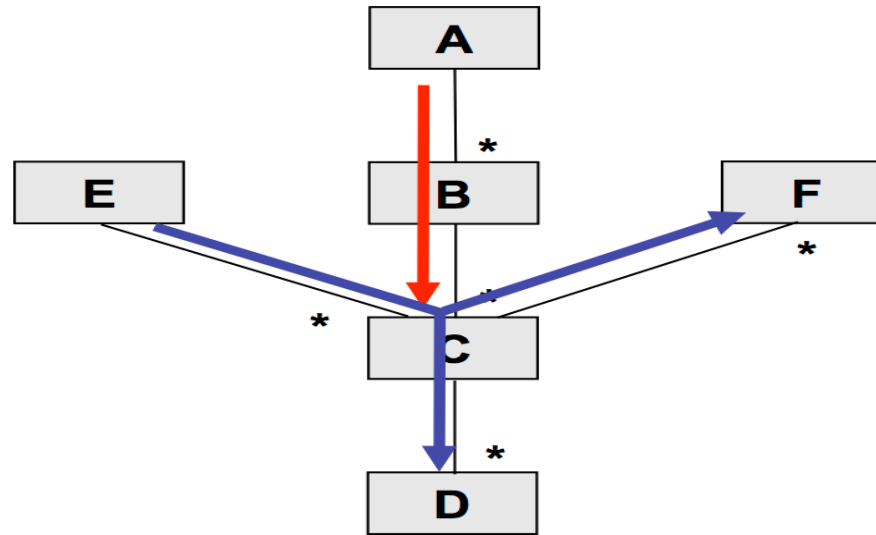


Implementation of sample hierarchy

```

"A"
...
"B"
...
"C":{"ID":"c"},
...
"D"...
"F"...
"E":[{"REF":"c"}, ...]
  
```


Implementation of network structures



Implementation of sample hierarchy

```

"E"
...
"C": {"ID": "c"}
...
"D"
"F"
"A"
...
"B": [{"REF": "c"}, ... ]
  
```

BSON Design

Outline

[Class](#)

[Optional attribute](#)

[Multivalued attribute](#)

[Qualification](#)

[One-to-one association](#)

[One-to-many association](#)

[Many-to-many association](#)

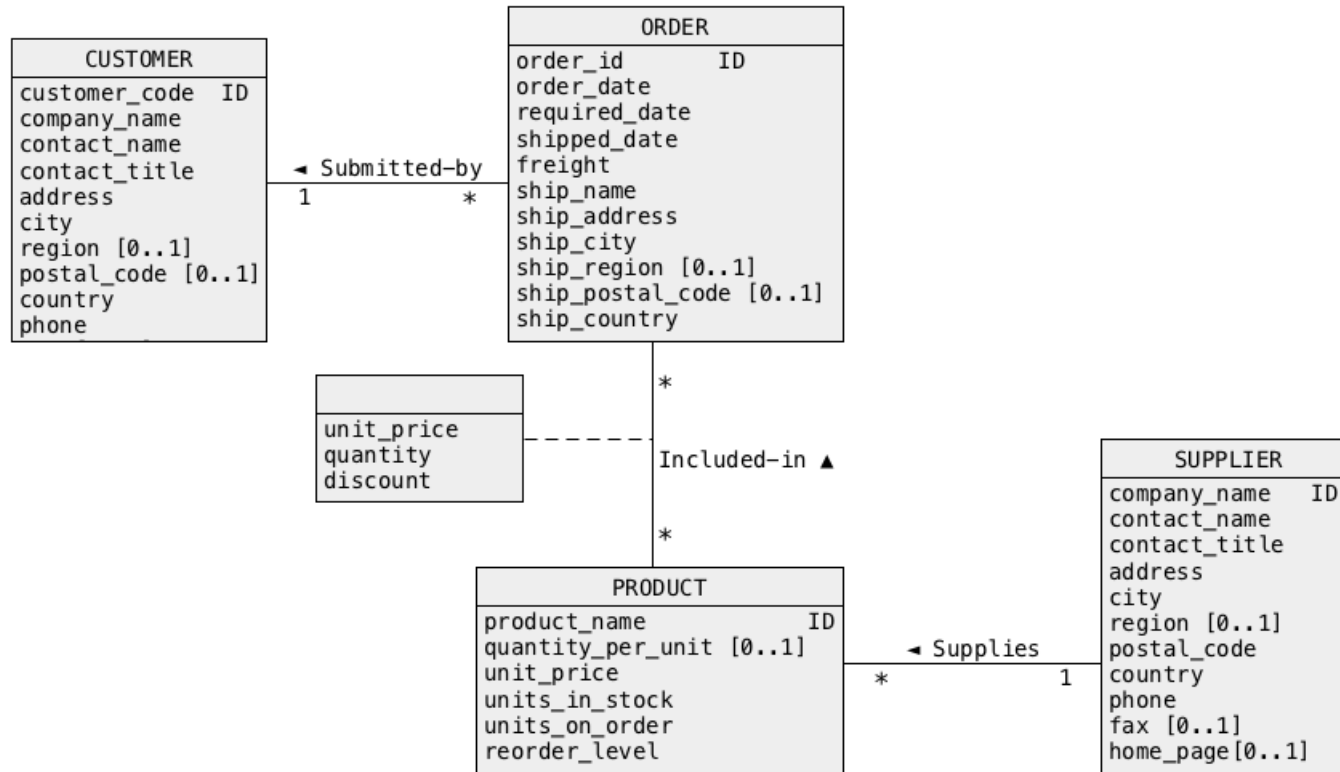
[Generalization](#)

[Implementation of network structures](#)

[Example](#)

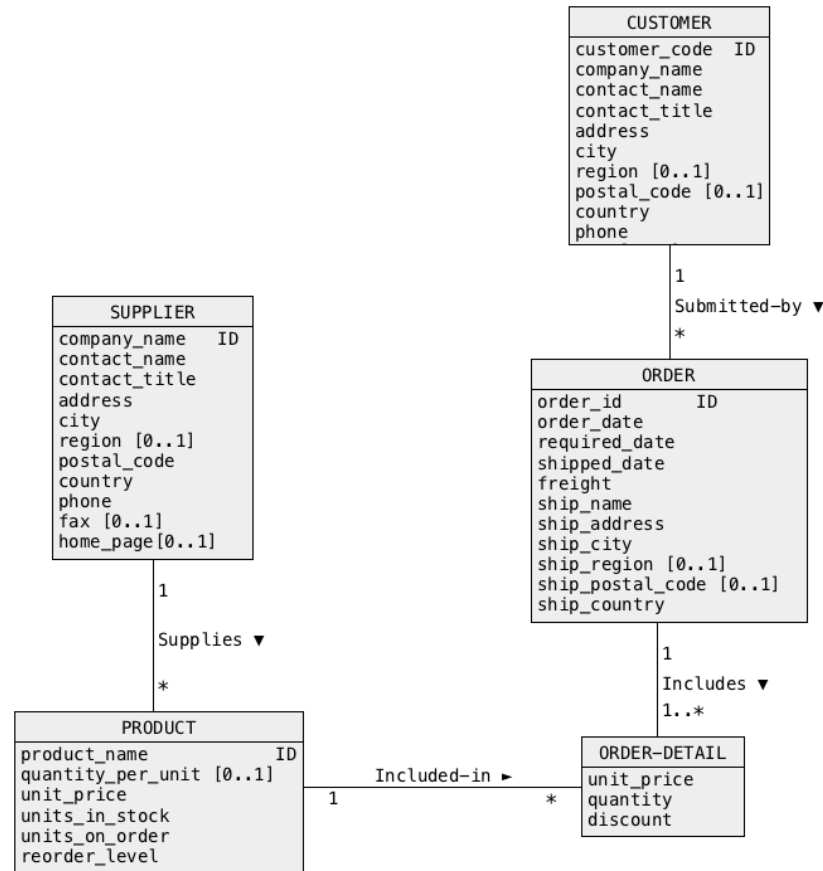
Example

Consider the following conceptual schema



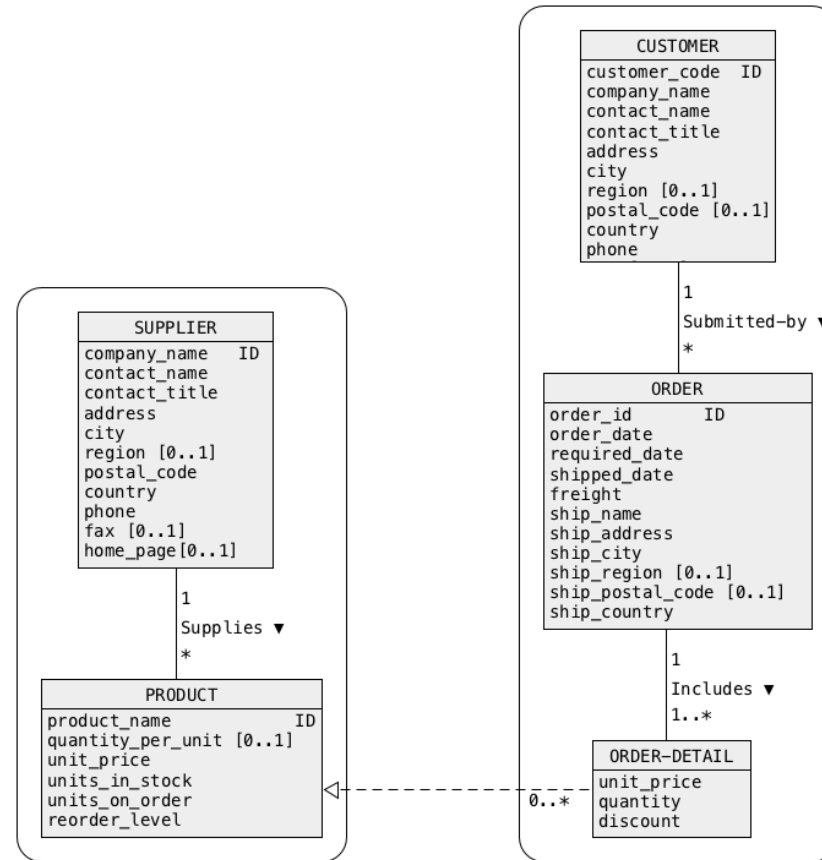
Example

Replacement of "many-to-many" association with "one-to-many" and "many-to-one" associations provides the following conceptual schema



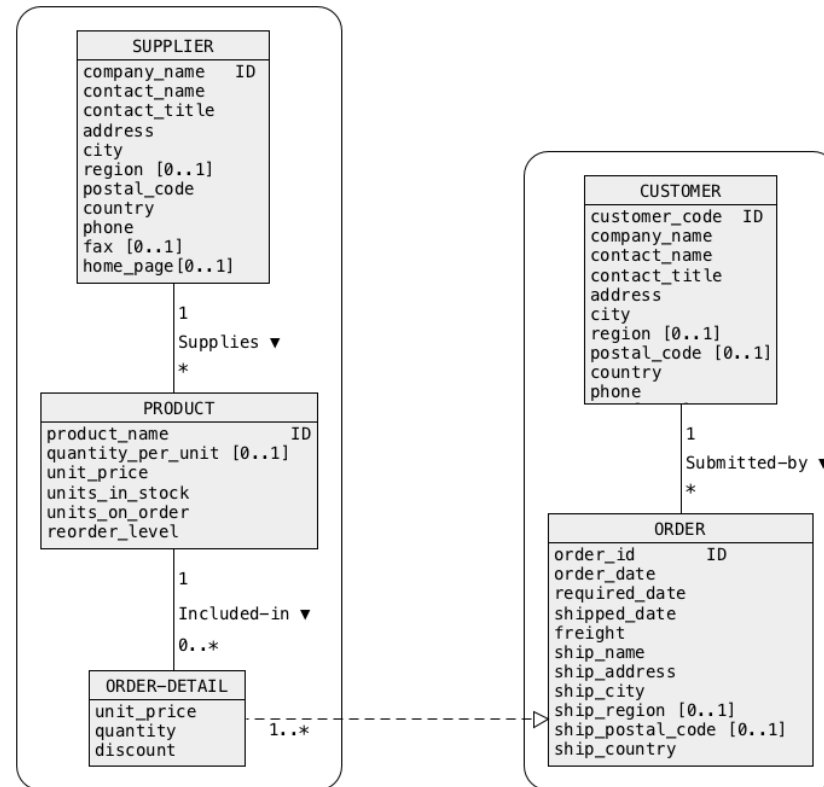
Example

Transformation into hierarchical structures and links provides the following a logical schema



Example

Another transformation into hierarchical structures and links provides the following a logical schema



References

[JSON Schema](#)

[Understanding JSON Schema](#)

[MongoDB - JSON Schema validation](#)

[MongoDB - \\$jsonSchema operator](#)