1.264 Lecture 5

Data modeling

---

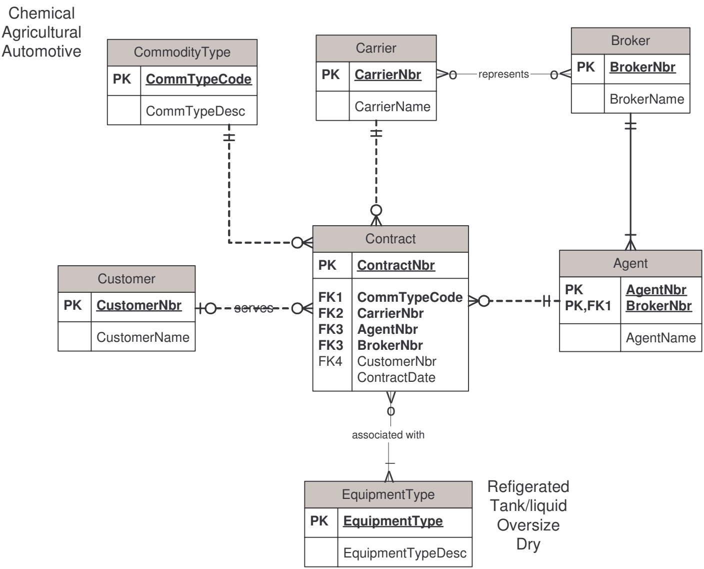
1. Data models

- Data model is representation of
  - Things (or entities or objects) of importance to a business
  - How the things relate to each other
- It is built and modified until it represents the business well enough to write a system.
- Data models are extended to become class diagrams in the Unified Modeling Language [UML] by adding the behaviors of each entity to the model
- Data models are sometimes built during requirements, and other times during design phase
  - The earlier the better. I always build them during requirements.

Logical data modeling

- Method to discover the data, relationships and rules of a business, collectively called the business rules
- Logical data models are the basis of:
  - Physical data models, or actual databases
  - Applications, parts of which can be automatically generated from the data model
- Small model for EZ Broker of transportation services
  - Small, but says a lot about EZ Broker
  - Gives good picture of what database should look like
  - Also gives good picture of underlying business rules of broker
    - Useful in requirements analysis and scrubbing!

EZ Broker Data Model



Chemical
Agricultural
Automotive

CommodityType
PK **CommTypeCode**
CommTypeDesc

Carrier
PK **CarrierNbr**
CarrierName

represents

Broker
PK **BrokerNbr**
BrokerName

Contract
PK **ContractNbr**
FK1 **CommTypeCode**
FK2 **CarrierNbr**
FK3 **AgentNbr**
FK3 **BrokerNbr**
FK4 CustomerNbr
ContractDate

Customer
PK **CustomerNbr**
CustomerName

serves

Agent
PK **AgentNbr**
PK,FK1 **BrokerNbr**
AgentName

associated with

EquipmentType
PK **EquipmentType**
EquipmentTypeDesc

Refigerated
Tank/liquid
Oversize
Dry

EZ Broker Business Rules

- A carrier can be associated with many brokers
- A broker can be associated with many carriers
- A carrier can issue many contracts
- A contract is issued by one carrier
- A broker can employ many agents
- An agent is employed by one broker
- An agent can sell many contracts
- A contract is serviced by only one agent
- A contract can serve to carry only one commodity type
- A commodity type can be carried under many contracts
- A contract can be associated with many equipment types
- An equipment type can be associated with many contracts
- A customer can be served by many contracts
- A contract covers one customer

Data model purpose

- Business needs to build logical data model so users and developers both understand business rules of company
  - Models enable users and developers to have single view of system
  - Sometimes users note this is first time they understood business rules!
  - Object modeling is small extension
    - In an object model (class diagram) we not only model data, but also the methods (behaviors, procedures) that each entity has
- Converting logical to physical data model (database) is very straightforward these days.
  - Little need for separate physical model for online databases
  - We do create separate physical models for data warehouses, read-only databases and some other special cases

2. Data modeling concepts

- Entities (objects, tables)
- Attributes (properties)
- Keys (primary and foreign)
- Relationships
- Referential integrity

Entity type and entity occurrence

Entity type

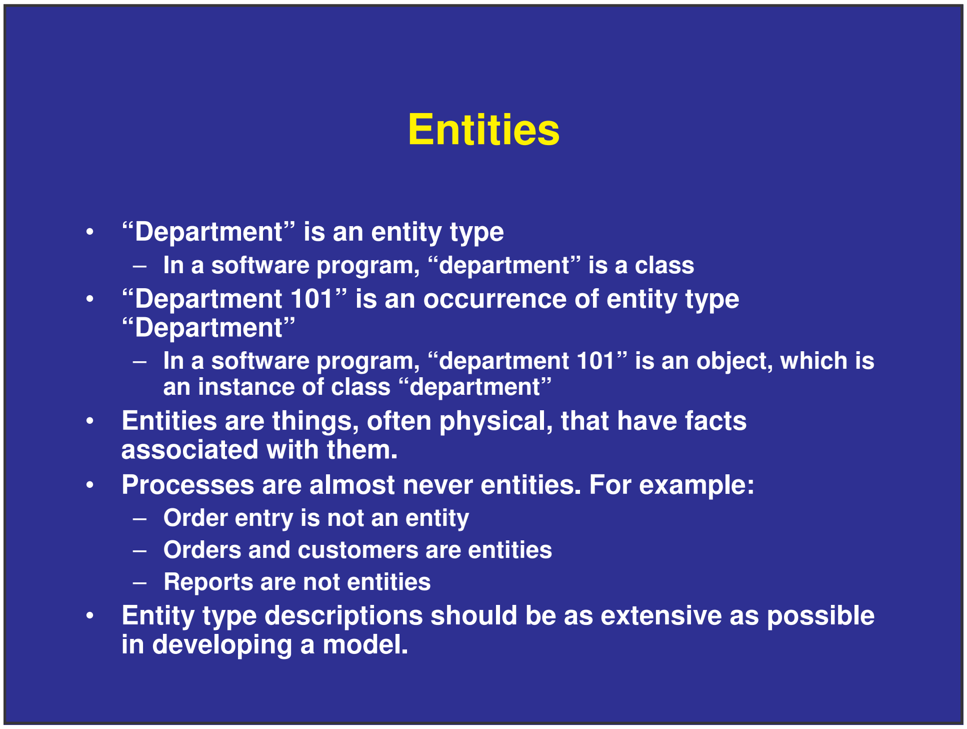
Department

DeptNbr
DeptName
DeptType
DeptStatus

Entity occurrence

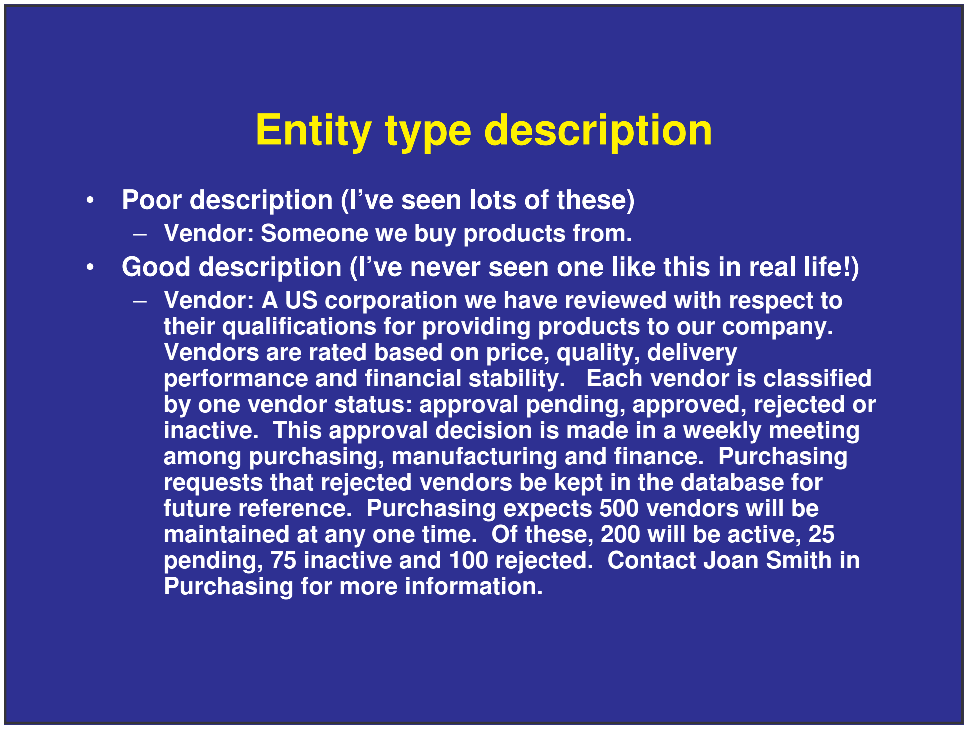
Department

DeptNbr	DeptName	DeptType	DeptStatus
930	Receiving	Mfg	Active
378	Assembly	Mfg	Active
372	Finance	Adm	Active
923	Planning	Adm	Active
483	Construction	Plant	Inactive



Entities

- “Department” is an entity type
  - In a software program, “department” is a class
- “Department 101” is an occurrence of entity type “Department”
  - In a software program, “department 101” is an object, which is an instance of class “department”
- Entities are things, often physical, that have facts associated with them.
- Processes are almost never entities. For example:
  - Order entry is not an entity
  - Orders and customers are entities
  - Reports are not entities
- Entity type descriptions should be as extensive as possible in developing a model.



Entity type description

- Poor description (I’ve seen lots of these)
  - Vendor: Someone we buy products from.
- Good description (I’ve never seen one like this in real life!)
  - Vendor: A US corporation we have reviewed with respect to their qualifications for providing products to our company. Vendors are rated based on price, quality, delivery performance and financial stability. Each vendor is classified by one vendor status: approval pending, approved, rejected or inactive. This approval decision is made in a weekly meeting among purchasing, manufacturing and finance. Purchasing requests that rejected vendors be kept in the database for future reference. Purchasing expects 500 vendors will be maintained at any one time. Of these, 200 will be active, 25 pending, 75 inactive and 100 rejected. Contact Joan Smith in Purchasing for more information.

Attributes

- Attributes are a data item or property associated with an entity type
  - They are typically nouns (quantity, type, color, …)
  - Example: Employee
    - ID
    - Name
    - Social security number
    - Address
    - Phone

Entity type/attribute exercise

1. Identify which are types and which are attributes:

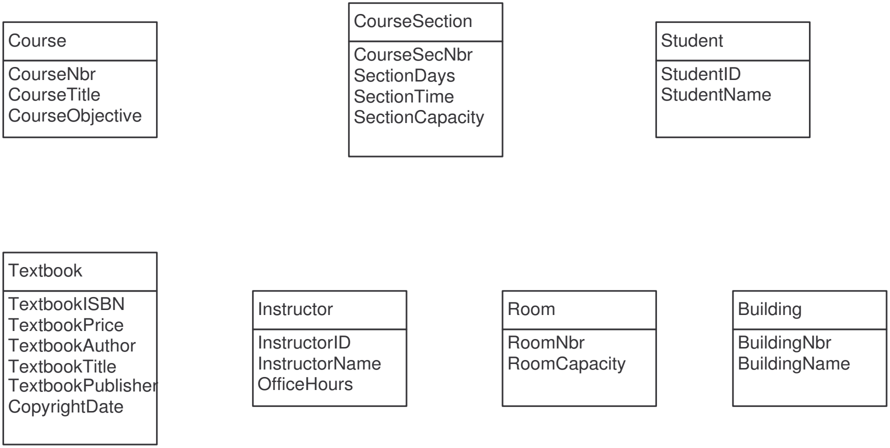
- Instructor
- Student
- Course section number
- Building name
- Course number
- Textbook price
- Student name
- Instructor ID
- Textbook author
- Course title
- Textbook
- Classroom
- Textbook ISBN
- Section days
- Office hours
- Textbook title
- Classroom number
- Student ID
- Instructor name
- Textbook publisher
- Section capacity
- Course objective
- Copyright date
- Building number
- Course section
- Course
- Building
- Section time
- Classroom capacity

Entity type/attribute exercise

2. Draw an entity type box and its attributes for each:

- Instructor
- Student
- Course section number
- Building name
- Course number
- Textbook price
- Student name
- Instructor ID
- Textbook author
- Course title
- Textbook
- Classroom
- Textbook ISBN
- Section days
- Office hours
- Textbook title
- Classroom number
- Student ID
- Instructor name
- Textbook publisher
- Section capacity
- Course objective
- Copyright date
- Building number
- Course section
- Course
- Building
- Section time
- Classroom capacity

Relationships and Cardinality



Course
CourseNbr
CourseTitle
CourseObjective

CourseSection
CourseSecNbr
SectionDays
SectionTime
SectionCapacity

Student
StudentID
StudentName

Textbook
TextbookISBN
TextbookPrice
TextbookAuthor
TextbookTitle
TextbookPublisher
CopyrightDate

Instructor
InstructorID
InstructorName
OfficeHours

Room
RoomNbr
RoomCapacity

Building
BuildingNbr
BuildingName

(Classroom)

Domain entity type

- Also called pick list, validation list, etc.
- Department name example

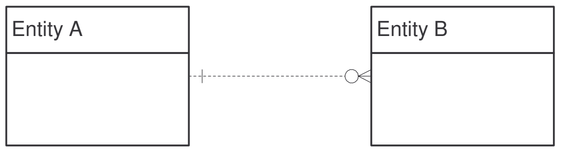
Domain entity type

Department			
DeptNbr	**DeptName**	**DeptType**	**DeptStatus**
930	Receiving	Mfg	Active
378	Assembly	Mfg	Active
372	Finance	Adm	Active
923	Planning	Adm	Active
483	Constructi	Plant	Inactive

ValidDeptType
DeptType
Mfg
Adm
Plant
Sales
Operations

Relationships

- Entities are drawn as boxes, as in the EZ Broker diagram
- Relationships are lines between boxes
- Cardinality is the expected number of related occurrences between the two entities in the relationship
- Relationships + cardinality = business rules

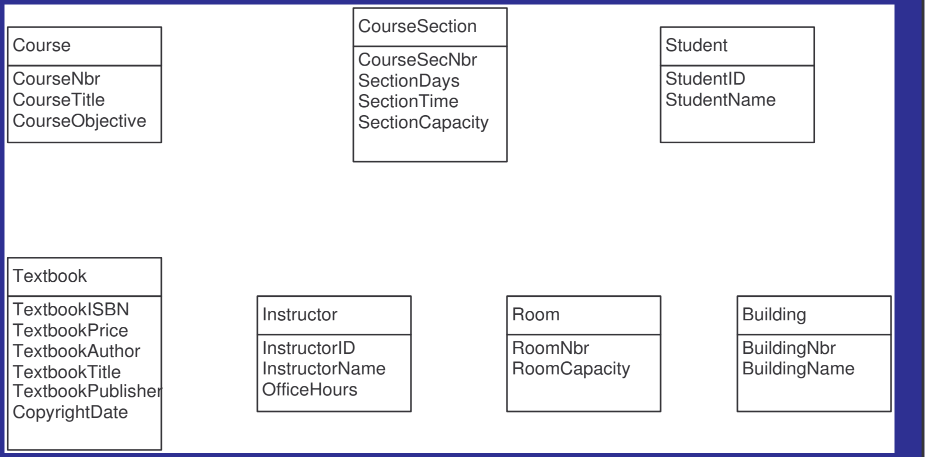


(Instructor)

(Course section)

Relationships and Cardinality

Exercise: Draw the relationships among these entities



Course
CourseNbr
CourseTitle
CourseObjective

CourseSection
CourseSecNbr
SectionDays
SectionTime
SectionCapacity

Student
StudentID
StudentName

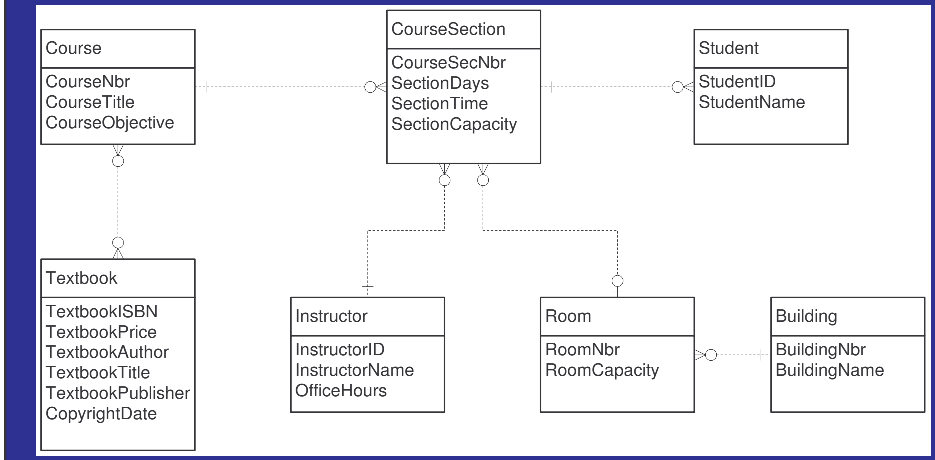
Textbook
TextbookISBN
TextbookPrice
TextbookAuthor
TextbookTitle
TextbookPublisher
CopyrightDate

Instructor
InstructorID
InstructorName
OfficeHours

Room
RoomNbr
RoomCapacity

Building
BuildingNbr
BuildingName

Relationships and Cardinality



Course
CourseNbr
CourseTitle
CourseObjective

CourseSection
CourseSecNbr
SectionDays
SectionTime
SectionCapacity

Student
StudentID
StudentName

Textbook
TextbookISBN
TextbookPrice
TextbookAuthor
TextbookTitle
TextbookPublisher
CopyrightDate

Instructor
InstructorID
InstructorName
OfficeHours

Room
RoomNbr
RoomCapacity

Building
BuildingNbr
BuildingName

We're getting there: we've defined entities, attributes and relationships. We still have to add keys and more entities

Course example

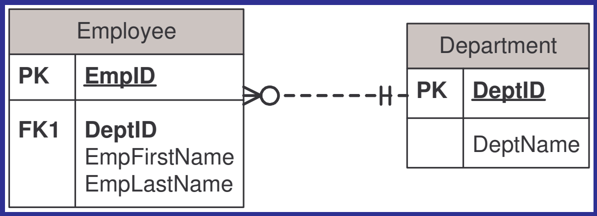
- Course may be offered in many (0,1 or more) sections
- Course section must be associated with a course
- Course section may be attended by many (0,1 or more) students
- Student may enroll in 1 course section
- Course section must be taught by 1 instructor (??)
- Instructor may teach many sections
- Course may use many textbooks (all sections use same)
- Textbook may be used in many courses
- Building may contain many rooms
- A room is in only one building
- A course section may use a room
- A room may be used by many course sections (not at same time)

Keys

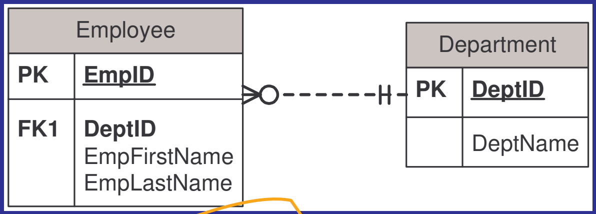
- Primary key: one or more attributes that uniquely identify a record.
  - What would you use in a customer database of 100,000 people and no unique customer id?
    - Name not unique
    - Add birthdate, but not guaranteed to be unique
    - Address can change
    - Can use social security number, but not everyone has one
    - Privacy is an issue
  - Issues in choosing a primary key
    - Stability
    - Control
    - Use a system generated key if possible in many cases

Foreign keys

- Primary key of the independent or parent entity type is maintained as a non-key attribute in the dependent or child entity type



Foreign keys

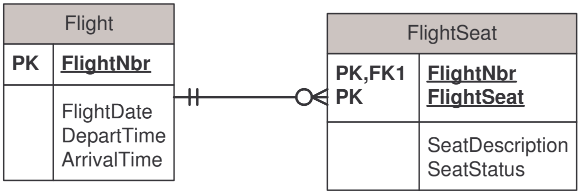


EmpID	DeptID	EmpFirstName	EmpLastName
4436	483	Brown	John
4574	483	Jones	Helen
5678	372	Smith	Jane
5674	372	Crane	Sally
9987	923	Black	Joe
5123	923	Green	Bill
5325	483	Clinton	Bob

DeptID	DeptName
930	Receiving
378	Assembly
372	Finance
923	Planning
483	Construction

Database requires a valid department number when employee is added

Identifying foreign keys

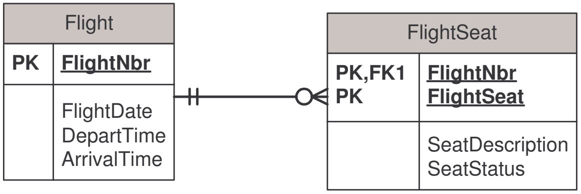


Independent/parent

Dependent/child (must contain, as a foreign key, the primary key of the independent entity)

To keep it simple, assume this is a charter airline, and every flight has a different number

Identifying foreign keys



Flight

FlightNbr	FlightDate	DepartTime	ArrivalTime
243	9/24/00	9:00am	11:00am
253	9/24/00	10:00am	12:30pm
52	9/24/00	11:00am	2:00pm

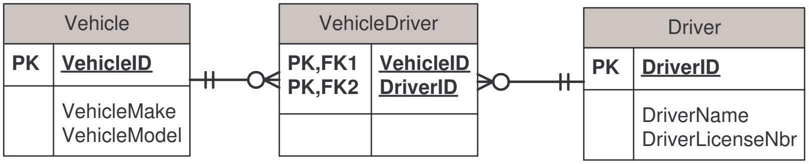
FlightSeat

FlightNbr	SeatNbr	SeatStatus	SeatDescription
243	8A	Confirmed	Window
243	7D	Reserved	Aisle
243	14E	Open	Center
253	1F	Open	Window
253	43A	Confirmed	Window

(This violates normalization, the next topic. Can you see the problem?)

Foreign keys (many-many relationships)

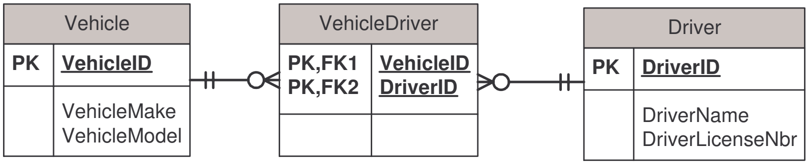
- **Primary key of parent is used in primary key of child**



Independent **Dependent** **Independent**

Vehicle can be driven by many drivers; driver can drive many vehicles

Many-to-many relationships with foreign keys



Vehicle

VehicleID	VehicleMake	VehicleModel
35	Volvo	Wagon
33	Ford	Sedan
89	GMC	Truck

Vehicle Driver

VehicleID	DriverID
35	900
35	253
89	900

Driver

DriverID	DriverName	DriverLicenseNbr
253	Ken	A23423
900	Jen	B89987

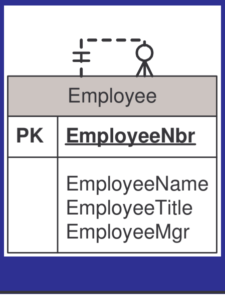
Never create an entity with vehicle1, vehicle2,… !

Referential integrity

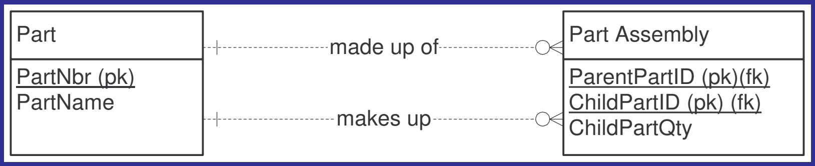
- Referential integrity maintains the validity of foreign keys when the primary key in the parent table changes
  - Every foreign key either matches a primary key or is null
- Cascade rules. Choose among two delete options:
  - Cascade restrict: Rows in the primary key table can't be deleted unless all corresponding rows in the foreign key tables have been deleted.
    - E.g., when deleting a department, don't delete all the employees
  - Cascade delete: When rows in the primary key table are deleted, associated rows in foreign key tables are also deleted
    - E.g. When deleting an order, delete all items in the order
  - Cascade update: When rows (keys) in the primary key table are updated, associated rows in foreign key tables are also updated
    - E.g., when changing a department number, change the employee department numbers

Recursive relation

- We'll cover this in more detail under SQL. Pretend the recursive relation is between two tables, the real one and a virtual copy. In this case, a manager table and an employee table. Proceed as usual, with a small syntax change.



Sets of recursive relations



Part

PartNbr (pk)
PartName

made up of

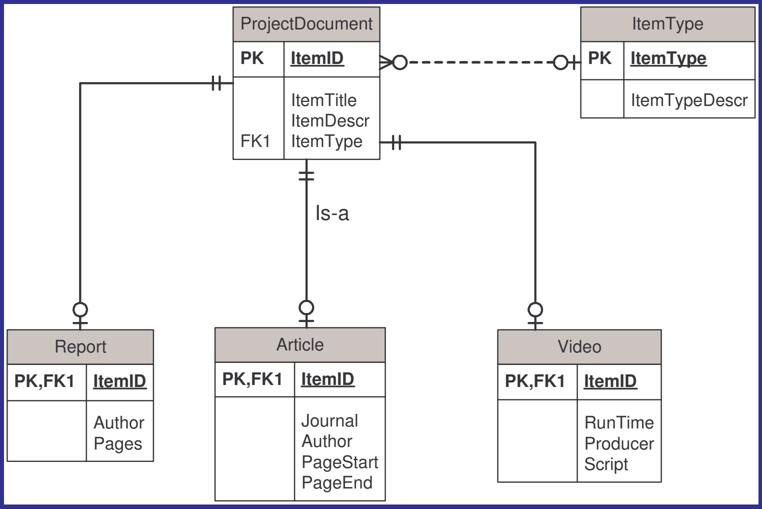
makes up

Part Assembly

ParentPartID (pk)(fk)
ChildPartID (pk) (fk)
ChildPartQty

Steering column makes up part of steering system
Steering column is made up of shafts, linkages, etc.

Variant (category) relations



ProjectDocument

PK	**ItemID**
	ItemTitle ItemDescr
FK1	ItemType

ItemType

PK	**ItemType**
	ItemTypeDescr

Is-a

Report

PK,FK1	**ItemID**
	Author Pages

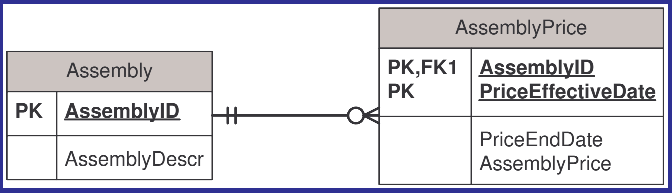
Article

PK,FK1	**ItemID**
	Journal Author PageStart PageEnd

Video

PK,FK1	**ItemID**
	RunTime Producer Script

Time dependent relation



Assembly	
PK	**AssemblyID**
	AssemblyDescr

AssemblyPrice	
PK,FK1 **PK**	**AssemblyID** **PriceEffectiveDate**
	PriceEndDate AssemblyPrice