Conceptual Database Design 1

Watch video: https://youtu.be/FeI2WrnFgjc?t=48m38s

Topics List

- Conceptual Database Design
- Build conceptual data model
- Worked Example

Conceptual Database Design

- The process of constructing a model of the data used in an enterprise, independent of *all* physical considerations.
- The goal of this phase is to produce a conceptual schema (which includes identification of the important entity types, relationship types, and attributes) for the database that is independent of a specific DBMS. We will use ER modelling during this phase.

Conceptual Database Design

Overview Database Design Methodology

- Step 1 Build conceptual data model.
- Step 2 Build and validate logical data model.
- Step 3 Translate logical data model for target DBMS.
- Step 4 Design file organisations and indexes.
- Step 5 Design user views.
- Step 6 Design security mechanisms.
- Step 7 Consider the introduction of controlled redundancy.
- Step 8 Monitor and tune the operational system.

Conceptual Database Design

Build conceptual data model

- Step 1.1 Identify entity types.
- Step 1.2 Identify relationship types.
  - 1.2.1 Cardinality.
  - 1.2.2 Participation.
- Step 1.3 Identify and associate attributes with entity or relationship types.
- Step 1.4 Determine attribute domains.
- Step 1.5 Determine candidate, primary, and alternate key attributes.
- Step 1.6 Consider use of enhanced modelling concepts (optional step).
- Step 1.7 Check model for redundancy.
- Step 1.8 Validate conceptual model against user transactions.
- Step 1.9 Review conceptual data model with user.

Topics List

- Conceptual Database Design
- Build conceptual data model
- Worked Example

Build conceptual data model

Identify entity types

- One method is to examine the users’ requirements specification for n*ou*ns or *noun phras*es.

- Also look for major objects such as people, places, or concepts of interest, excluding those nouns that are merely qualities of other objects.

- Document the entity types.

Build conceptual data model

Identify relationship types

- This is a key step in the process of building a conceptual data model.
  - One method is to examine users' requirements specification for *verbs* or *verbal expressions*. For example, if we have identified 2 entity types as *Staff* and *PropertyForRent*; a possible expression in the requirements specification would be: *Staff* ***Manages*** *PropertyForRent.*
  - Use entity–relationship (ER) modelling to understand the relationship types.
  - Determine the *multiplicity constraints of relationships*. These are used to check and maintain data quality.
  - Document the relationship types.

Build conceptual data model

Identify and associate attributes with entity or relationship types.

- Attributes can be identified where noun or noun phrase is a property, quality, identifier, or characteristic of one of the entity or relationship types previously found.
- Identify whether attributes are:
  - Simple/composite
  - Single/multi-valued
  - Derived
- Document the attributes.

Build conceptual data model

Documenting attributes

- Record the following information for each attribute:
  - attribute name and description;
  - data type and length;
  - any aliases that the attribute is known by;
  - whether the attribute must always be specified (in other words, whether the attribute allows or disallows nulls);
  - whether the attribute is multi-valued;
  - whether the attribute is composite, and if so, which simple attributes make up the composite attribute;
  - whether the attribute is derived and, if so, how it should be computed;
  - default values for the attribute (if specified).

Build conceptual data model

Determine attribute domains

- A domain is a pool of values from which one or more attributes draw their values.
- A domain specifies:
  - allowable set of values for the attribute;
  - size and format of the attribute.
- Document the attribute domains.

Build conceptual data model

Guidelines for choosing a primary key

- Select the candidate key
  - with the minimal set of attributes;
  - that is less likely to have its values changed;
  - that is less likely to lose uniqueness in the future;
  - with fewest characters (for those with textual attribute(s));
  - with the smallest maximum value (for numerical attributes);
  - that is easiest to use from the users’ point of view.

Topics List

- Conceptual Database Design
- Build conceptual data model
- Worked Example

Worked Example

- You have been asked to design a database for a garage. The manager of the garage has told you details (on next 2 slides).

- **To do:**
  - Produce an Entity Relationship diagram to clarify your understanding of the client’s requirements.

Worked Example

- We keep a list of customers who book in repair jobs. Their customerId, name (fName, lName), address (street, town and county) and contact number are recorded.
- For each repair job, we allocate a new job number, record the date and vehicle registration number.
- We keep a list of all our mechanics, with their hourly rate of pay. We also record their staffId, and name (fName, lName).
- We keep a list of the different parts that we use. For each part we record a part number, description, quantity in stock and cost of the part.
- We record the supplier details for the supplier who supplies each part. These details include supplier number, supplier name (fName, lName), address (street, town and county), contact number, fax number and email address.

Worked Example

- Each customer will submit 0 or more repair jobs. Each repair job involves one and only one customer.
- There is only one supplier for each particular part and each supplier will supply 0 or more parts.
- Every repair job is worked on by at least one mechanic and each mechanic will work on 0 or more repair jobs. For billing purposes, we need to keep track of all the time that each mechanic spends on a given job.
- A repair job will use 0 or more parts and a part can be used in 0 or more repair jobs. We need to keep track of the quantity of the part used in each repair job.

Worked Example

- **Step 1.1 Identify entities**

Worked Example

- **Step 1.1 Identify entities**
  - Customer
  - Repair Job
  - Mechanic
  - Part
  - Supplier

Worked Example

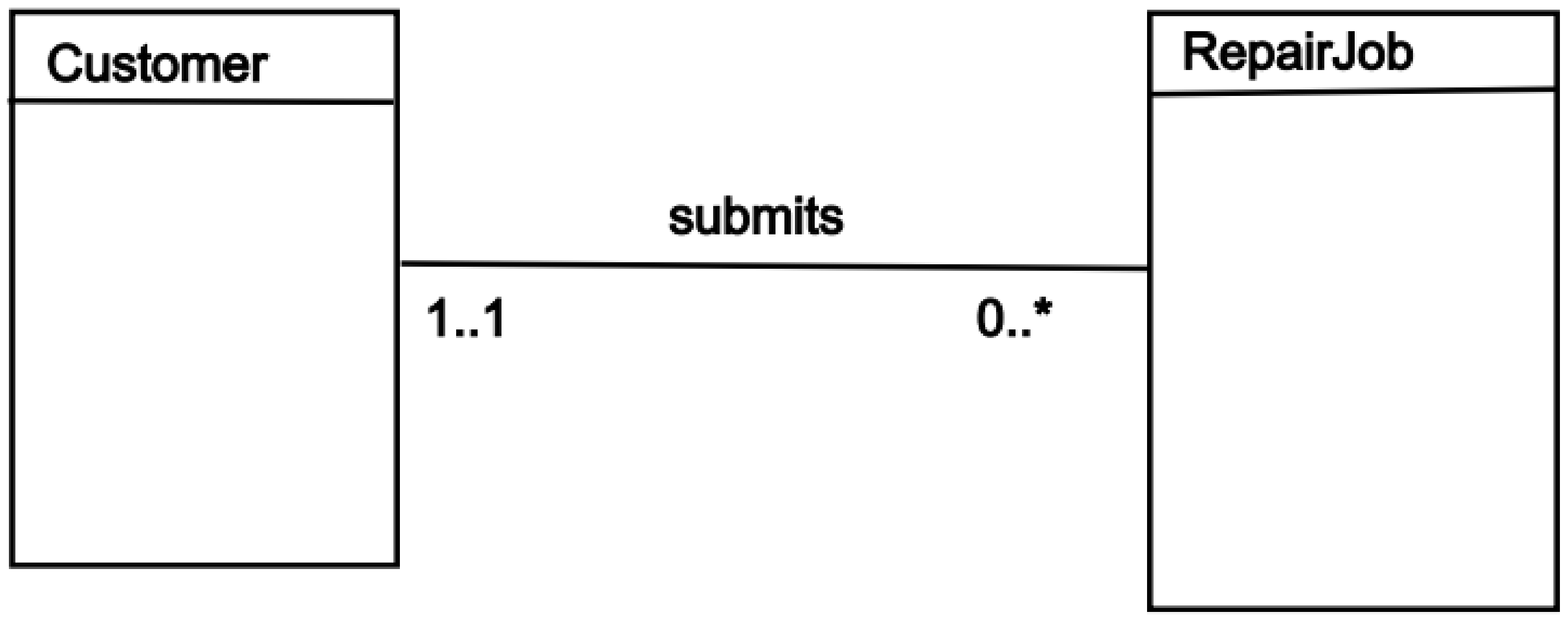
- **Step 1.2 Identify relationships**

Worked Example

- **Step 1.2 Identify relationships**
  - Each customer will submit 0 or more repair jobs. Each repair job involves one and only one customer.
    - Customer 1..1 **submits** 0..* Repair Job
  - There is only one supplier for each particular part and each supplier will supply 0 or more parts.
    - Supplier 1..1 **supplies** 0..* Part

Worked Example

- **Step 1.2 Identify relationships**
  - Every repair job is worked on by at least one mechanic and each mechanic will work on 0 or more repair jobs.
    - Mechanic 1..* **worksOn** 0..* Repair Job
  - A repair job will use 0 or more parts and a part can be used in 0 or more repair jobs.
    - Repair Job 0..* **uses** 0..* Part


Customer
submits
1..1
0..*
RepairJob

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**

- **Entity Type attributes:**
  - **Customer**:customerId, name (fName, lName), address (street, town, county), contactNumber
  - **Repair Job**: jobNumber, jobDate, regNumber
  - **Mechanic**: staffId, name (fName, lName), hourlyRate
  - **Supplier**: supplierNumber, supplierName (fName, lName), address (street, town, county), contactNumber, faxNumber, emailAddress
  - **Part**: partNumber, description, stockQuantity, partCost

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**
    - **Relationship Type attributes:**
        - There is a requirement to record the *time* that each Mechanic spends on a given Repair Job. This attribute *time* is not an attribute of Mechanic because there would be multiple *time* values for each Repair Job that the Mechanic works on. Likewise, it is not an attribute of Repair Job because there would be multiple *time* values for each Mechanic working on the Repair Job. It represents the amount of *time* one Mechanic works on one Repair Job. So for each relationship occurrence between Mechanic and Repair Job there will be a *time* value.

Worked Example

- **Step 1.3 Identify and associate attributes with entities or relationships**

  - **Relationship Type attributes:**
    - There is also a requirement to record the *quantity* of a Part used in each Repair Job. This attribute *quantity* is not an attribute of Part because there would be multiple *quantity* values for each Repair Job that the Part is used in. Likewise, it is not an attribute of Repair Job because there would be multiple *quantity* values for each Part being used in the Repair Job. It represents the *quantity* amount of one Part being used in one Repair Job. So for each relationship occurrence between Part and Repair Job there will be a *quantity* value.

Worked Example

- **Step 1.4 Determine attribute domains**
- In the data dictionary record the allowable set of values for the attribute; and the size and format of the attribute.

Worked Example

- **Step 1.5 Determine candidate, primary, and alternate key attributes**

  - **Customer**: Candidate keys: customerId, name and address, contactNumber
    Primary key: customerId

  - **Repair Job**: Candidate key: jobNumber
    Primary key: jobNumber

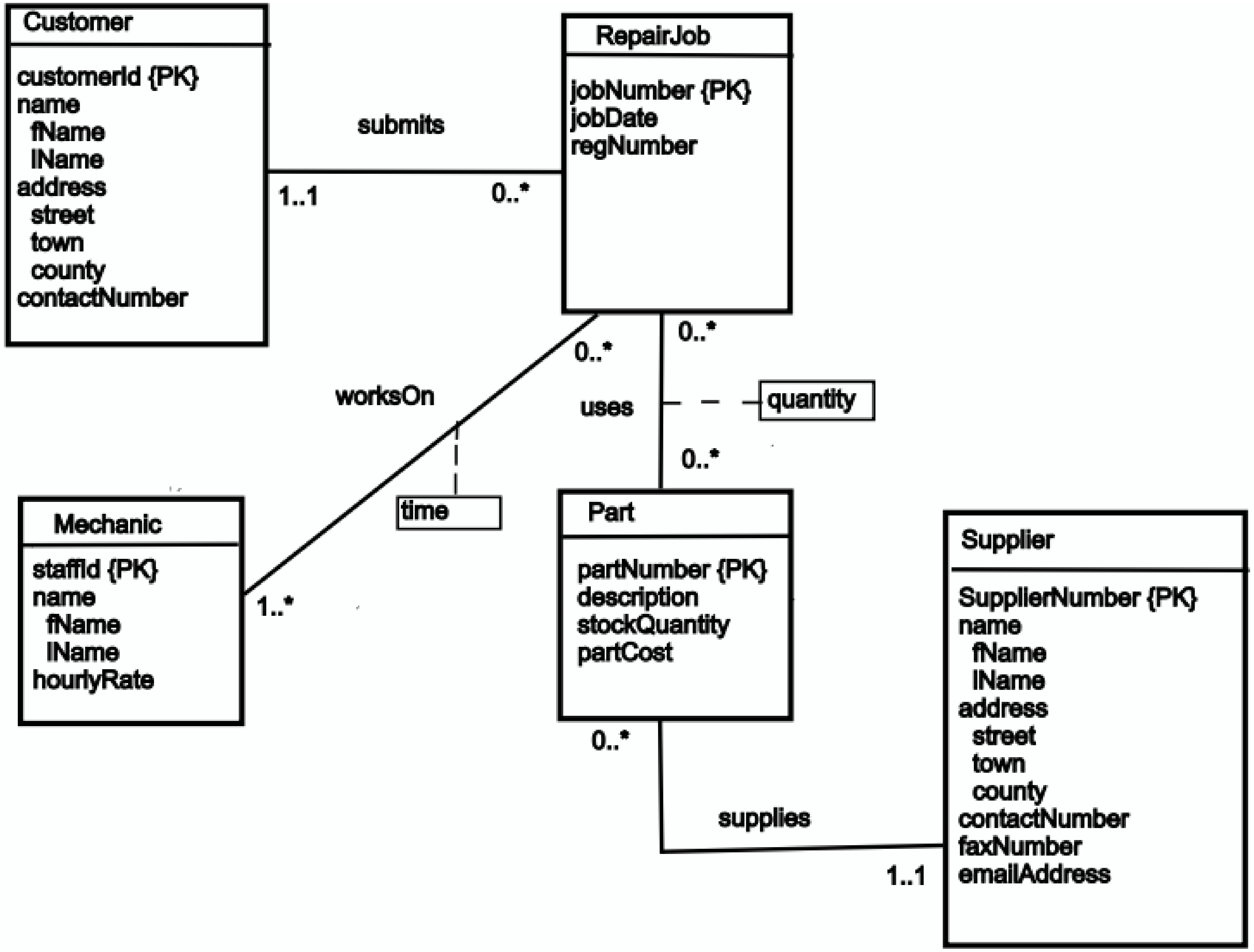
  - **Mechanic**: Candidate key: staffId
    Primary key: staffId

Worked Example

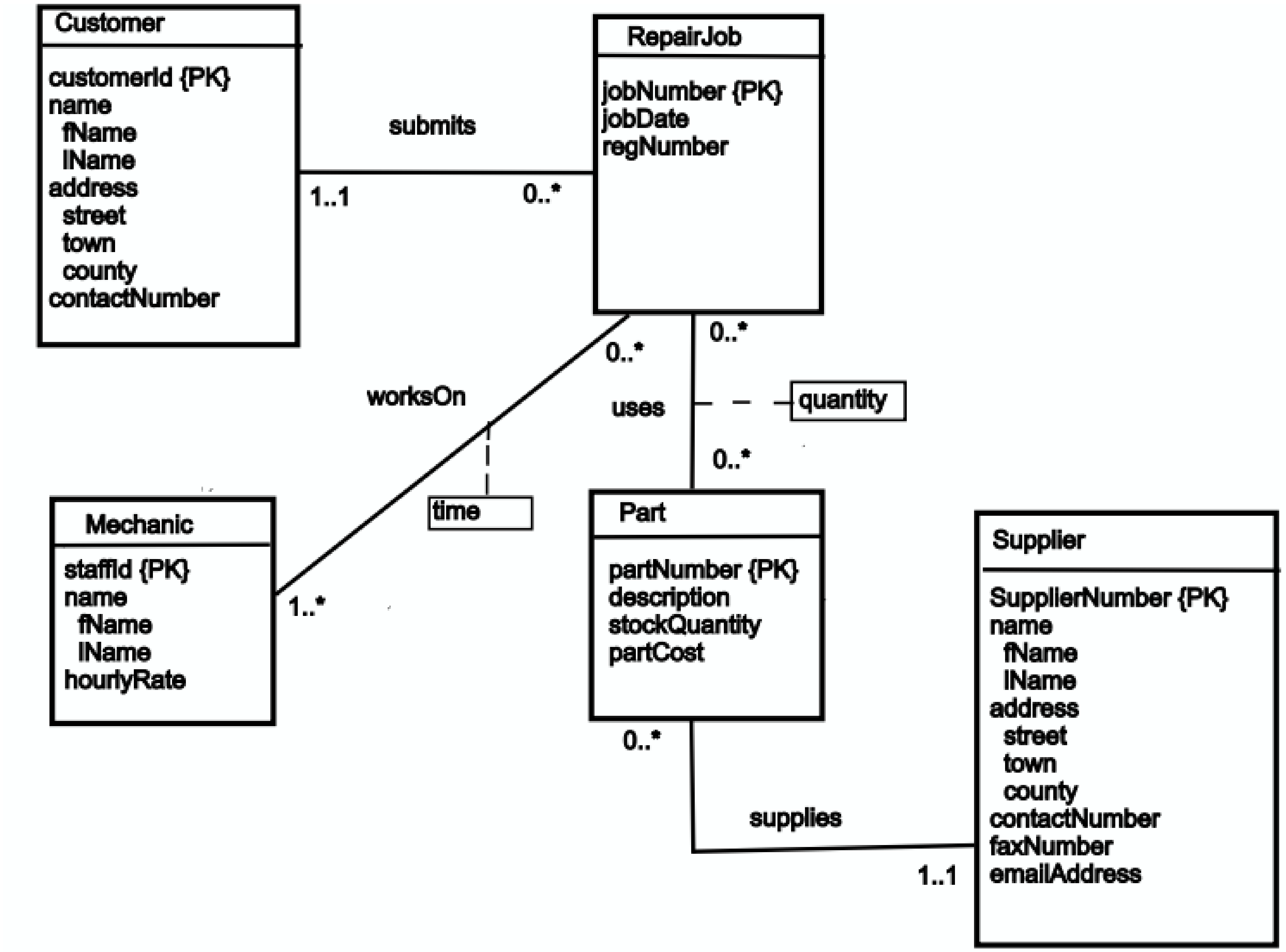
- **Step 1.5 Determine candidate, primary, and alternate key attributes**

  - **Supplier**: Candidate keys: supplierNumber, supplierName and address, contactNumber, faxNumber, emailAddress
    Primary key: supplierNumber

  - **Part**: Candidate keys: partNumber, description
    Primary key: partNumber


Customer
customerId {PK}
name
fName
lName
address
street
town
county
contactNumber
submits
1..1
0..*
RepairJob
jobNumber {PK}
jobDate
regNumber
0..*
0..*
worksOn
uses
quantity
0..*
time
1..*
Mechanic
staffId {PK}
name
fName
lName
hourlyRate
Part
partNumber {PK}
description
stockQuantity
partCost
0..*
supplies
1..1
Supplier
SupplierNumber {PK}
name
fName
lName
address
street
town
county
contactNumber
faxNumber
emailAddress

Worked Example (Solution)


Customer
customerId {PK}
name
fName
lName
address
street
town
county
contactNumber
submits
1..1
0..*
RepairJob
jobNumber {PK}
jobDate
regNumber
0..*
0..*
worksOn
uses
quantity
0..*
time
Mechanic
staffId {PK}
name
fName
lName
hourlyRate
1..*
Part
partNumber {PK}
description
stockQuantity
partCost
0..*
supplies
1..1
Supplier
SupplierNumber {PK}
name
fName
lName
address
street
town
county
contactNumber
faxNumber
emailAddress