Intro to Database Systems

Final Deliverable

Quinten Parker
Kory Bartlett

Santa Clara, CA

December 6, 2016

Contents

1 Introduction

The requirements of the final project for COEN 178 Intro to Database System was to create the back-end and front-end of a web application for a machine repair service, So Prompt Inc. The back-end of the web application consists of PHP files which communicate to an Oracle database. The front-end of the web application consists of HTML forms that send data to the PHP files. The web application allows user to:

- create service contracts for machines
- issue repair jobs
- view a machine's status
- view a customer's bill
- view current repair jobs
- view generated revenue
- update a machine's status

Contained in this document is the entity-relationship diagram from which we derived the database relations, the functional dependencies, the resulting tables, and queries used in the back-end of the web application.

2 Entity-Relationship Diagrams

The Entity-Relationship diagrams in Fig. 2.1 and Fig. 2.2 display the relations for the SoPrompt Services Inc. application. From these diagrams, we derived the tables that made up the back-end database system.

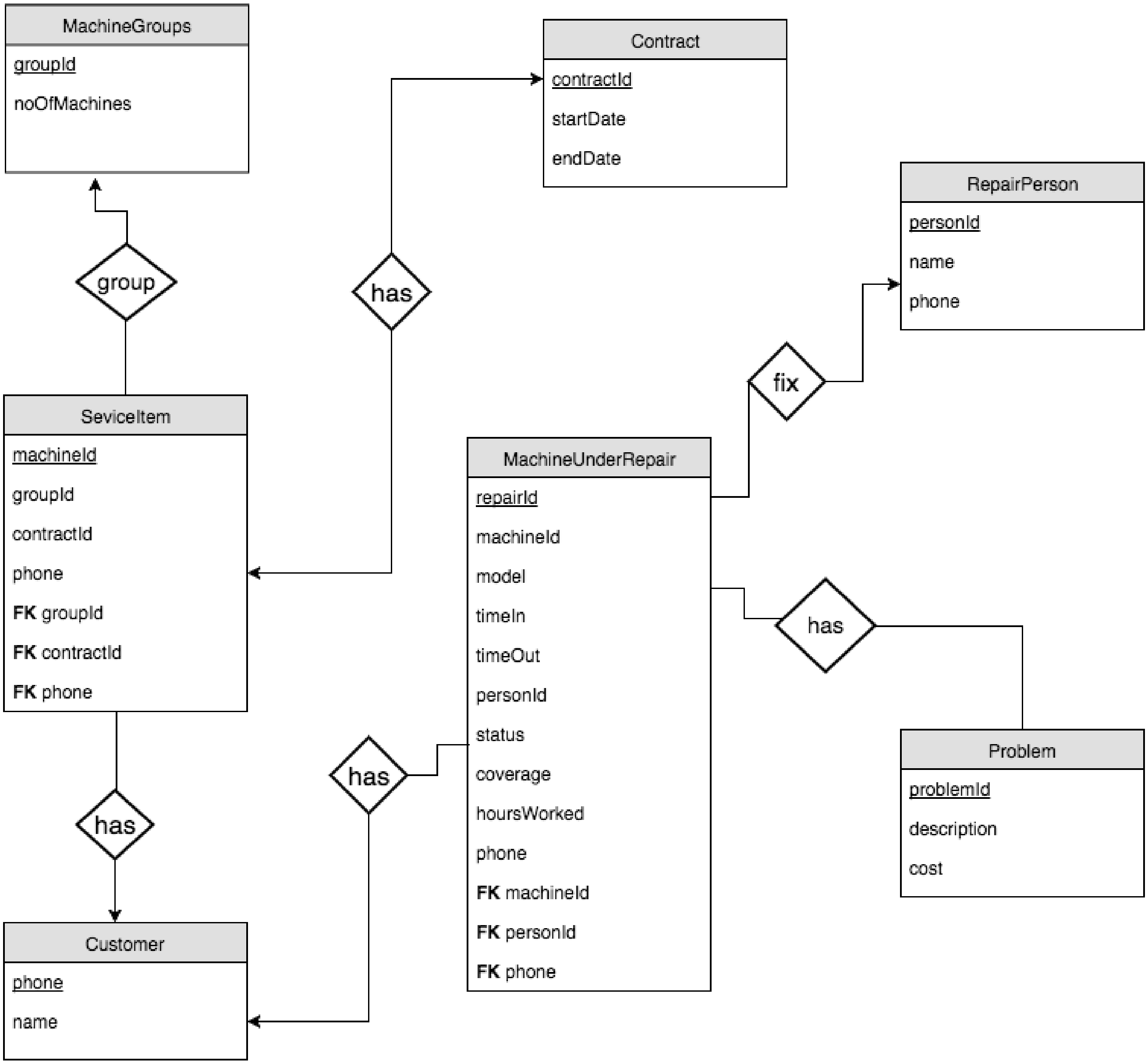

Figure 2.1: Entity-Relationship Diagram with Book Notation

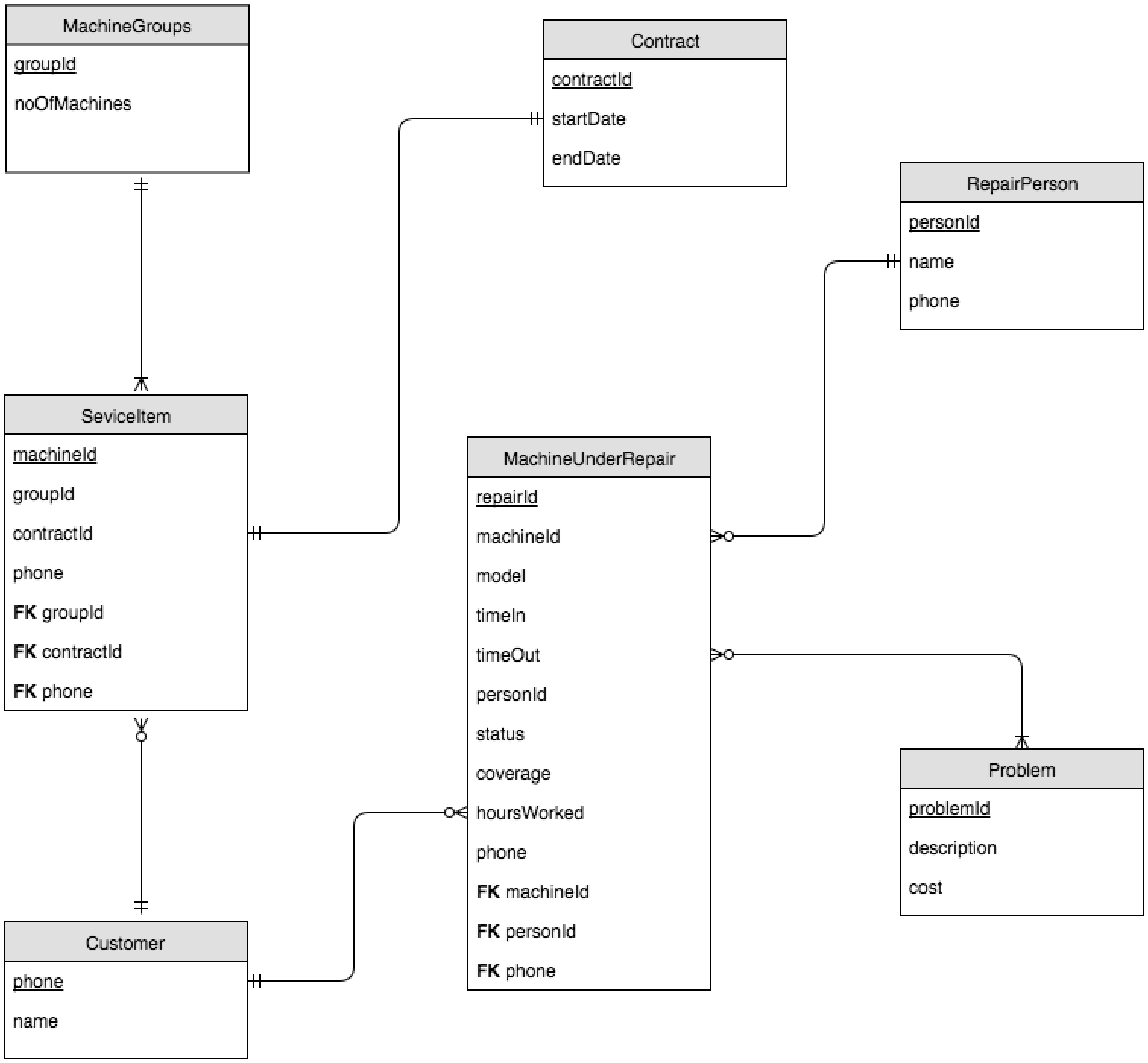

Figure 2.2: Entity-Relationship Diagram with Cardinality & Modality

3 Functional Dependencies

A functional dependency is a relationship that exists when one attribute or group of attributes uniquely determine another attribute or group of attributes. For the SoPrompt Services Inc. application, we identified the following functional dependencies:

- Customer (phone, name)
  - FD: phone → name
  - The relation is in BCNF form as the closure of the FD, phone, consists of all the possible values of the relation
- MachineGroups (groupId, noOfMachines)
  - FD: groupId → noOfMachines
  - The relation is in BCNF form as the closure of the FD, groupId, consists of all the possible values of the relation
- Contract (contractId, startDate, endDate)
  - FD: contractId → startDate, endDate
  - The relation is in BCNF form as the closure of the FD, contractId, consists of all the possible values of the relation
- repairPerson (personId, name, phone)
  - FD: personId → name, phone
  - The relation is in BCNF form as the closure of the FD, personId, consists of all the possible values of the relation
- problem (problemId, description, cost)
  - FD: problemId → description, cost
  - The relation is in BCNF form as the closure of the FD, problemId, consists of all the possible values of the relation
- serviceItem (machineId, groupId, contractId, phone)

  - FD: machineId → groupId, contractId, phone
  - FD: groupId → contractId, phone
  - The relation is not in BCNF form as the closure of the FD, groupId, violates BCNF form as it does not consist of all the possible values of the relation

- MachineUnderRepair (repairId, machineId, model, timeIn, timeOut, personId, status, coverage, hoursworked, phone)
  - FD: repairId → machineId, model, timeIn, timeOut, personId, status, coverage, hoursworked, phone
  - FD: timeIn, timeOut → status, hoursWorked
  - FD:endDate, contractId → coverage
  - The relation is not in BCNF form as the closure of the FDs, timeIn-timeOut and endDate-contractId, violate BCNF form as it does not consist of all the possible values of the relation

4 Tables

The following tables were derived from the Entity-Relationship diagrams from Chapter 2 (a Primary Key is denoted by PK, a Foreign Key is denoted by FK):

- ServiceItem

  PK: machineId

  PK: groupId

  PK: contractId

  phone

- MachineGroups

  PK: groupId

  NoOfMachines

- Customer

  PK: phone

  name

- Contract

  PK: contractId

  startDate

  endDate

- RepairProblem

  PK,FK: machineId

  PK,FK: problemId

- MachineUnderRepair

  PK: repairId

  FK: machineId

  model

  FK: personId

  timeIn

  timeOut

  status

  coverage

  hoursWorked

  FK: phone

- RepairPerson

  PK: personId

  name

  phone

- Problem

  PK: problemId

  description

  cost

5 Queries

The following queries were used in the application to store and retrieve data:

View Machine Status

```
SELECT status FROM MachineUnderRepair WHERE machineId = ’$machineId’;
```

View Customer Bill

```
SELECT genBill(’$repairId’) from dual;
```

View Repair Jobs

```
SELECT * FROM MachineUnderRepair;
```

View Generated Revenue

```
SELECT genRevenue(’$startDate’, ’$endDate’) FROM dual;
```

Create Service Contract

```
INSERT INTO Contract VALUES (’$contractId’, ’$startDate’, ’$endDate’)

INSERT INTO MachineGroups VALUES (’$groupId’, ’1’);

INSERT INTO ServiceItem VALUES (’$machineId’, ’$groupId’, ’$pNum’, ’$contractId’);
```

Create Repair Job

```
INSERT INTO MachineUnderRepair (repairId, machineId, model, personId, timeIn, status,
    coverage, hoursWorked, phone) VALUES (’$repairId’, ’$machineId’, ’$model’, ’$repPersonId’,
    ’$timeIn’, ’$status’, ’$covered’, ’$hWorked’, ’$pNum’);
```

Update Machine Status

```
UPDATE MachineUnderRepair SET status=’$newMachStat’, hoursWorked=hoursWorked+’$hours’
    WHERE machineId=’$machineId’;

UPDATE MachineUnderRepair SET status=’$newMachStat’, timeOut=’$curTime’, hoursWorked=hoursWo
    WHERE machineId=’$machineId’;
```

6 Assumptions

We made the following assumptions about the system in order to complete the project:

- In table MachineGroups, noOfMachines could be no larger than 3 (used a check constraint to ensure this).
- In table MachineUnderRepair, the status of a machine could only be 1, 2, 3, or 4 (used a check constraint to ensure this).
- In table MachineUnderRepair, the coverage could only be Y or N (used a check constraint to ensure this).
- The coverage attribute was not kept up to date, but only indicated if a machine had a contract at some point. Therefore, the end date of the contract always had to be checked.