Phase 3 Update

Michael Kramarczyk
mdk5@njit.edu
CIS 631

Functional Dependencies

Employee Table

The assumption is that each employee has a unique username in the system and there is one employee record for every employee in the company.

Field	**Value Description**	**Rules**
employee_id (Primary Key)	INTEGER	Primary Key Value Starts at 10000. Must be unique. Must be filled.
username	VARCHAR(10)	Must be unique. Must be filled.
password	VARCHAR(10)	Must be filled.
firstname	VARCHAR(15)	Must be filled.
lastname	VARCHAR(15)	Must be filled.
phonenumber1	VARCHAR(15)	Must be filled.
phonenumber2	VARCHAR(15)	
admin	INTEGER	Must be filled. (1 or 0)
active	INTEGER	Must be filled. (1 or 0)

employee_id→username
employee_id→firstname
employee_id→lastname
employee_id→phonenumber1
employee_id→phonenumber2
employee_id→password
employee_id→admin
employee_id→active

An employee_id functionally determines the employee attributes: username, firstname, lastname, phonenumber1, phonenumber2, password, admin, active. An employee_id is the super key in the Employee table. It is unique within the system.

username→employee_id
username→firstname
username→lastname
username→phonenumber1
username→phonenumber2
username→password
username→admin
username→active

A username functionally determines the employee attributes: employee_id, firstname, lastname, phonenumber1, phonenumber2, password, admin, and active. The username uniquely identifies an employee, but it is only used when logging into the system.

Job Table

The assumption is that there is a unique job_id and unique job name for each job in the system.

Field	Value Description	Rules
job_id (Primary Key)	INTEGER	Primary Key Value Starts at 1. Must be unique. Must be filled.
name	VARCHAR(15)	Must be unique. Must be filled.
active	INTEGER	Must be filled. (1 or 0)

job_id→name
job_id→active

A job_id functionally determines the job attributes: name and active. The name attribute refers to the name of the job. The name attribute can change, but it must remain unique in the system.

name→job_id
name→active

A name functionally determines a job attributes: job_id and active. The name attribute is not used throughout the system since a name can change value as long as it remains unique. This design decision was made so that job names can change and the change would only have to be changed in one place.

Employee Job Table

The assumption is that every employee can have multiple jobs, but that employee and job must exist in the system.

Field	**Value Description**	**Rules**
employee_id (Foreign Key) (Part of Primary Key)	INTEGER	Must be filled. Combined with job_id, it forms the primary key.
job_id (Foreign Key) (Part of Primary Key)	INTEGER	Must be filled. Combined with employee_id, it forms the primary key.
Active	INTEGER	Must be filled. (1 or 0)

(employee_id, job_id)→active

The attributes employee_id and job_id functionally define an Employee Job attribute active. The attribute employee_id is a foreign key which points to an employee record. The attribute job_id is a foreign key which points to a job record. Originally there was an employeeJob_id, but this was removed to simplify the design. It was decided that employeeJob_id made the code more complex and took up more space in the database.

Employee Schedule Table

The assumption is that every employee schedule record which has an employee_id must have a job_id. The employee_id and job_id combination must exist in the employee job table. The employee can only be scheduled at most once for that particular day and must not have an employee vacation record for that particular day.

Field	**Value Description**	**Rules**
dateS (Part of Primary Key)	INTEGER	Primary Key value together with position_id. The combination of dateS and position_id must be unique. Must be filled.
position_id (Part of Primary Key)	INTEGER	Primary Key value together with dateS. The combination of dates and position_id must be unique. Must be filled.
employee_id (Foreign Key)	INTEGER	Employee must not be scheduled for this day and may not be on vacation this day. Employee must have a job. The employee_id must exist.
job_id (Foreign Key)	INTEGER	Employee must be able to perform job. The job_id must exist. There must be an employee_id if there is a job_id.
startTime	VARCHAR(4)	Must be filled.
endTime	VARCHAR(4)	Must be filled.
Active	INTEGER	Must be filled. (1 or 0)

(dateS, position_id)$\rightarrow$employee_id
(dateS, position_id)$\rightarrow$job_id
(dateS, position_id)$\rightarrow$startTime
(dateS, position_id)$\rightarrow$endTime
(dateS, position_id)$\rightarrow$active

The dateS and position_id attributes functionally define the Employee Schedule attributes: employee_id, job_id, startTime, endTime, and active. The attribute employee_id is a foreign key which points to an Employee record. The attribute job_id points to a job record. The employee_id and job_id will also point to an active and existing record in the employee job table.

(dateS, employee_id)→job_id
(dateS, employee_id)→startTime
(dateS, employee_id)→endTime
(dateS, employee_id)→active
(dateS, employee_id)→position_id

The dateS and employee_id attributes functionally define the Employee Schedule attributes: job_id, startTime, endTime, active, and position_id. This functionally dependency is true when the employee_id exists. When the employee_id does not exist (employee_id equals zero.), there might not be a unique record.

Employee Vacation Table

The assumption is that that every employee vacation record with an employee_id must not have an employee schedule record for that particular day. The employee_id must also have at most one employee vacation record for that particular day.

Field	**Value Description**	**Rules**
dateS (Part of Primary Key)	INTEGER	Primary Key value together with position_id. The combination of dateS and position_id must be unique. Must be filled.
position_id (Part of Primary Key)	INTEGER	Primary Key value together with dateS. The combination of dates and position_id must be unique. Must be filled.
employee_id (Foreign Key)	INTEGER	Employee must not be scheduled for this day. The employee_id must exist.
Active	INTEGER	Must be filled. (1 or 0)

(dateS, position_id)→employee_id
(dateS, position_id)→active

The dates and position_id attributes functionally define the Employee Vacation attributes: employee_id and active.

(dateS, employee_id)→active

(dateS, employee_id)$\rightarrow$position_id)

The dates and employee_id attributes functionally define the Employee Vacation attributes: active and position_id. When the employee_id does not exist (employee_id equals zero.), there might not be a unique record. Hence this functional dependency is only true if the employee_id exists.

3-NF Analysis

The design satisfies the 1-NF property since each field has a single value (not a set). This can be clearly seen. The design is 2-NF since no proper subset of the key functionally determines an attribute. There is no instance of duplicated information.

The employee_id is used as a foreign key in Employee Job table, Employee Schedule table, and Employee Vacation table. The employee_id foreign key uniquely identifies an employee. A bad implementation of these tables (Employee Job, Employee Schedule, and Employee Vacation) would have used employee firstname and employee lastname instead of employee_id. This would create a lot of duplicate information and be expensive to change. If the person changed their last name, it would have to be changed in multiple places with the bad implementation. Using the employee_id minimizes the changes to just the employee table.

The job_id is used as a foreign key in the Employee Job table and Employee Schedule table. The job_id foreign key uniquely identifies a job. A bad implementation of these tables (Employee Job and Employee Schedule) would use the job name instead of job_id. This would pose a problem if the job name changed. Using the job_id minimizes the changes to just the job table.

The design is 3-NF since it is 2-NF and there are no transitive dependencies.

Status Report

The Employee Schedule project will meet the July 28th, 2005 delivery date. Currently the GUI Interface and Database has been completed. The project is currently in the debugging and testing phase. There appears to be some problems related to adding an employee schedule record which is being investigated.

- Adding employee: Implemented and Tested
- Edit employee: Implemented and Tested
- Add Job: Implemented and Tested
- Edit Job: Implemented and Tested
- Add Employee Job: Implemented and Tested
- Edit Employee Job: Implemented and Tested
- Add/Edit Vacation: Implemented and Tested
- Add/Edit Schedule: Implemented, Fixing bugs
- Test Database Connection: Implemented and Tested
- Login: Implemented and Tested
- Logout: Implemented and Tested
- Help: Needs to be Implemented
- Version: Implemented and Tested
- Print Staff Schedule: Implemented, needs Testing
- Print Staff Available: Implemented, needs Testing
- Print Staff Info Active: Implemented, needs Testing

- Print Staff Info InActive: Implemented, needs Testing
- Print Staff Info: Implemented, needs Testing
- Print Staff Schedule Order by start time, employee name, job name: Implemented, needs Testing
- Print Staff Schedule Order by start time, job name, employee name: Implemented, needs Testing
- Print Staff Schedule Order by job name, start time, employee name: Implemented, needs Testing

  Print Staff Schedule Order by job name, employee name, start time: Implemented, needs Testing
- Print Staff Schedule Order by employee name, start time, job name: Implemented, needs Testing
- Print Staff Schedule Order by employee name, job name, start time: Implemented, needs Testing
- Print Employee Schedule: Implemented, needs Testing
- Print Employee Info: Implemented, needs Testing
- Manual: Completed First Version, need to edit