John Haney

COSC 640 – Database Systems I

Frostburg State University

**Table**

**Employee**(Employee ID, first name, last name, middle name, sex, SSN, annual salary, tax deduction, birthdate, date started at office, date started at company highest degree earned date graduated, name of spouse, number of dependents, cell phone, street address, zip code, city, state, country, marital status)

**Functional Dependencies:**

(PK) Employee ID 🡪 first name, last name, middle name, sex, SSN, annual salary, tax deduction, birthdate, date started at office, date started at company highest degree earned date graduated, name of spouse, number of dependents, cell phone, street address, zip code, city, state, country, marital status

SSN 🡪first name, last name, middle name, sex, (PK) Employee ID, annual salary, tax deduction, birthdate, date started at office, date started at company highest degree earned date graduated, name of spouse, number of dependents, cell phone, street address, zip code, city, state, country, marital status

cell phone 🡪 first name, last name, middle name, sex, (PK) Employee ID, annual salary, tax deduction, birthdate, date started at office, date started at company highest degree earned date graduated, name of spouse, number of dependents, SSN, street address, zip code, city, state, country, marital status

This table will record data relating to the employee. Update it is redundant to keep both a cell phone and home phone so we will remove home phone.

**Normalizations:**

The employee table is in the **first normal form** since all the attributes that are related to the table are atomic and cannot be broken down further.

The functional dependencies tell us that the table is also in the **second normal form** as well since the primary key is a single value. That means there cannot be a partial dependency.

The table is also in **third normal form.** We can see this in the functional dependencies since we do not gain any new information from the values returned from the dependencies, i.e. there are no transitive dependencies.

The table is also in **Boyce Codd Normal Form**. We can tell this because each determinate is fully dependent on the primary key and there are no non-prime attributes that play a part in determining a prime attribute.

The table will also be in the **fourth normal form** since there is less than two multivalued dependencies (zero in this table).

**Table (weak entity)**

**Certificate (**Employee ID, name of certificate, date earned**)**

**Functional dependencies:**

(PK) Employee ID 🡪🡪 name of certificate, date earned

This table is a weak entity i.e. an extension of the employee table. This is because the relationship is that there can be many certificates to a single employee and that there can be no certificates without an employee.

The table is in the **first normal form** since the attributes are atomic and indivisible.

The table is also in the **second normal form** since there are not any partial dependencies since the primary key is Employee ID and this is a prime key.

The table is also in **third normal form** since there are no possible transitive values. Nothing is gained from using the values returned by the primary key, as a determinate.

The table is also in **Boyce Codd normal form**. This is because the key is the only thing that determines the values. There are not any non-prime values that help to determine a unique value.

The table is also in **fourth normal form** since there is less than two *independent* multi-valued functional dependencies. We cannot have date without certificate name, so the multi-valued dependencies are dependent on name of certificate.

**Table (Weak entity)**

**Days open** (Office ID, days open)

**Functional dependencies:**

(PK) Office ID 🡪🡪 days open

This table represents a weak entity or extension of the office entity. It is used to store the days which the specified office is open.

The table is in the **First normal form.** This is because the values are all atomic and indivisible into small bits of data.

The table is also in the **second normal form** as the primary key of office ID cannot possibly have any partial dependencies.

The table is also in **third normal form.** This is obvious since there are only two attributes and therefore, we cannot have a transitive dependency.

The table is also in **Boyce Codd normal form.** This is because our candidate key is a single value attribute and chosen to be the primary key.

The table is also in the **fourth normal form** since there is less than two multi-valued dependencies.

**Table**

**Managed\_By** (Manager ID [employee ID], Employee ID, Date started as manager)

**Functional Dependencies:**

(PK)(FK) Employee ID 🡪(FK) Manager ID, Date started as manager

This table is a relationship between the employee table and itself to determine which employees are managed by who. While normalizing the table it became clear that the table should be using the employee ID’s rather than the manager ID’s as s primary key. This is because we will have a unique non-repeating value for each employee 1, 2, 3, … whereas if manager were the primary key it would repeat many times 1,1,1,1,2,2,… . Because of this reason we have renamed the table from “manages employee” to “managed by” better represent the relationship. This change will also remove the multi-valued dependency we would get from using manager as a primary key.

The table is in the **first normal form** since none of the values can be broken down further into more atomic values.

The table is also in the **second normal form** since there are no possible partial dependencies from prime key.

The table is also in the **third normal** **form** because we do not have transitive dependencies. We get no new information from either employee ID or date started as a manager.

The table is also in **Boyce Codd Normal Form**. We can see this is the case since our prime valued primary key is needed to get all the unique results.

The table is also in the **fourth normal form** since there is only a single multi-valued dependency.

**Table**

**Office** (Office ID, office name, phone number, fax number, email, cost of location, number of employees, country, state, city, street address, zip code)

**Functional dependencies:**

(PK) Office ID 🡪 office name, phone number, fax number, email, cost of location, country, state, city, street address, zip code

office name 🡪 (PK) Office ID, phone number, fax number, email, cost of location, country, state, city, street address, zip code

phone number 🡪 (PK) Office ID, office name, fax number, email, cost of location, country, state, city, street address, zip code

fax number 🡪 (PK) Office ID, office name, phone number, email, cost of location, country, state, city, street address, zip code

email 🡪(PK) Office ID, office name, phone number, fax number, cost of location, country, state, city, street address, zip code

This table us all the data for an office. Update there is no need to keep track of number of employees since that can be queried.

The table is in **first normal form** since all the attributes we can gather are atomic/indivisible into smaller pieces of data.

The table is also in the **second normal form** since there is a prime attribute primary key. This means there cannot be any partial dependencies

The table is also in the **third normal form** because there are no transitive dependencies. Nothing new is gained from applying any of the dependent values returned by the primary key.

The table is also in **Boyce Codd normal form**. This is because we have chosen a single value key to be the primary key and it is not disjoint.

The table is also finally in the **fourth normal** since there are no multi-valued dependencies.

**Table**

**Manages\_Office** (Manager ID [Employee ID], Office ID, Date started at office)

**Functional dependencies:**

(PK) (FK) Manger ID 🡪 date started at office, (FK) office ID

(FK) office ID 🡪 (PK) (FK) Manger ID, data started at office

This table will describe the relationship between the general manager of each office. Since it is a one-to-one relationship it will not have the same modifications as the “Manages employee” / “managed by”. This means that either office or manager ID’s would be acceptable as a primary key since they will both equally yield unique values.

The table is in the **first normal form** since there are only atomic values that are recorded.

The table is also in the **second normal form** since there is a single value primary key. This mean there cannot be a partial dependency.

The table will also satisfy the **third normal form**. This is because there is not a possible transitive dependency.

The table is also in **Boyce Codd normal form** since the primary key is single valued and no non-prime attributes determine the primary key.

The table will also be in the **fourth normal form** since there are no multi-valued dependencies in this one-to-one relationship.

**Table**

**Works\_at\_office** (office ID, Employee ID, date started at office)

**Functional dependencies:**

(PK) (FK) Employee ID 🡪 (FK) Office ID, date started at office

This table describes the relationship between the office and employee entity more specifically which employee works at which office and when they started there. This relationship is a one-to-many and will be much like the “Managed by table” wherein we will take the many side employee ID as the primary key. This will mean that each row of Employee ID will provide unique values.

The table is in the **first normal form** since there are only indivisible atomic values being stored here.

The table is also in the **second normal form** since there is a single valued primary key we have chosen. This mean there cannot possibly be a partial dependency.

The table will also satisfy the **third normal form.** This is because we do not have any possible transitive dependencies. Neither Office ID nor date started at office by themselves will yield any new values that are not trivial.

The table is also in **Boyce Codd normal.** this is because we have decided on a single/prime attribute as a primary key and it is not disjoint.

The table is also in the **fourth normal form** since there are no multi-valued dependencies.

**Table**

**Client** (Client ID, first name, middle name, last name, cell phone, SSN, birthdate, Email, credit score number, Zip code, city, state Country)

**Functional dependencies:**

(PK) Client ID 🡪 first name, middle name, last name, cell phone, SSN, birthdate, Email, credit score number, Zip code, city, state Country

SSN 🡪 first name, middle name, last name, cell phone, (PK) Client ID, birthdate, Email, credit score number, Zip code, city, state Country

Email 🡪 first name, middle name, last name, cell phone, (PK) Client ID, birthdate, SSN, credit score number, Zip code, city, state Country

Cell phone 🡪 first name, middle name, last name, cell phone, (PK) Client ID, birthdate, SSN, credit score number, Zip code, city, state Country

This table will record all the information that we care about involving a client.

The table is in the **first normal form** this is evident since there are no attributes we can divide further into smaller values.

The table is also in the **second normal form.** This is clear since there is a single value attribute chosen for a primary key and cannot therefore have partial dependencies.

The table is also in **third normal form.** we can see this since we cannot get any new values from any of the returned values i.e. there are no transitive dependencies.

The table happens to also be in **Boyce Codd normal form** this is because we have chosen a single value attribute to be the primary key. This key is also not disjoint.

The table is also in **fourth normal form** since there are no multi-valued dependencies.

**Table**

**Checks\_client** (Employee ID, Client ID, date checked)

**Functional dependencies:**

(PK) (FK) Client ID 🡪 (FK) Employee ID, date checked

This table will record which employee helped to check which client on which date. This is a one-to-many relationship and therefore the primary key will have to be the primary key from the many side. This will ensure that each row can be targeted accurately while limiting the multi-valued dependencies.

The table is in the **first normal form** since there are only atomic values that are being recorded.

The table is also in the **second normal form** since there are no possible partial dependencies on a single value primary key.

The table is also in the **third normal form.** This is because there are no transitive dependencies. We do not get any new information from either employee ID or date checked.

The table is also in the **Boyce Codd normal form.** Every determinate is a single value/prime candidate key we have chosen a primary key from this group.

The table is also in the **fourth normal form** since there are no multi-valued dependencies.

**Table**

**Vehicle** (Car ID (VIN), license plate number, year registered, class, manufacturer, color, additional cost per mile, car description, Listed, state registered, current milage, number of doors, car model, model year, weekly price, monthly price)

**Functional dependencies:**

(PK) Car ID (VIN) 🡪 license plate number, year registered, class, manufacturer, color, additional cost per mile, car description, Listed, state registered, current milage, number of doors, car model, model year, weekly price, monthly price

License plate number 🡪 (PK) Car ID (VIN), year registered, class, manufacturer, color, additional cost per mile, car description, Listed, state registered, current milage, number of doors, car model, model year, weekly price, monthly price

This table contains the information needed to record a new vehicle into the system. We will use the VIN number as they are all unique and are used by many other databases to record information about vehicles.

The table is in the **first normal form** this is because we have separated all the data into atomic values.

The table is also already in the **second normal form.** We can tell this since we can clearly see there is no possibility of a partial dependency.

The table also stratifies the requirement of the **third normal** since no new information is gained by looking at any of the attributes returned form the candidate keys i.e. there are no transitive dependencies.

The table also fulfills **Boyce Codd normal form**. We can see that each determinate is a candidate key and that those are all single value attributes. From this group we have chose the Car ID (VIN) to be the primary key.

Finally, the table is in the **fourth normal form** since there are no multi-valued dependencies.

**Table**

**List Vehicle** (Client ID, Car ID, date listed)

**Functional dependencies:**

(PK) (FK) Car ID 🡪 (FK) Client ID, date listed

This table is the result of a relationship between the vehicles and their owners. The type of relationship is a many-to-one which means that we will choose the foreign key from the many side as primary key.

The table is clearly in the **first normal form** since the only three values we are keeping track of are two ID’s (single integers) and a date.

The table is also in **second normal form** since the primary key is a single value attribute and cannot have partial dependencies.

The table is also in the **third normal form**. We can verify this by checking the dependent attributes and seeing if we get any new information. We do not, therefore we are normalized for 3NF.

The table will also satisfy the **Boyce Codd normal form.** This is because we really only have one possible candidate key. This key is a single value attribute and we have chosen it as a primary key.

The table is also normalized for **fourth normal form.** This is because we do not have any multi-valued dependencies.

**Table**

**Customer** (Customer ID, Driver’s license number, state driver’s license was issued, email, first name, last name, middle name, birth date, cell phone)

**Functional dependencies:**

(PK) Customer ID 🡪 Driver’s license number, state driver’s license was issued, email, first name, last name, middle name, birth date, cell phone

Email 🡪 Driver’s license number, state driver’s license was issued, (PK) Customer ID, first name, last name, middle name, birth date, cell phone

cell phone 🡪 Driver’s license number, state driver’s license was issued, (PK) Customer ID, first name, last name, middle name, birth date, Email

This table will record all the information that is involved with registering and saving a customer type user. There will be less attributes here then in the client since they will not need to have a credit check and we will not need to know their location.

The table is in the **first normal form** since the values we took from the initial proposal were carefully broken into their most basic data components.

The table is also in the **second normal form** since there are no possible partial dependencies when a primary key is a single value attribute.

The table is also in the **third normal form.** This is clear when we look at the non-candidate keys to see if they return new data. They do not and therefore we do not have any transitive dependencies.

The table happens to also be in **Boyce Codd normal form**. We can see that all determinates are single value candidate keys and from this group we have chosen a primary key.

The table also fulfills the **fourth normal form** since we do not have any multi-valued dependencies.

**Table**

**Accident** (Accident ID, Customer ID, Vehicle ID, country, state, city, street address, zip code, cost of damage, date of accident, description of accident, time of accident, reporting office first name, reporting office last name, summary of police report)

**Functional dependencies:**

(PK) Accident ID 🡪 (FK) Customer ID, (FK) Vehicle ID, country, state, city, street address, zip code, cost of damage, date of accident, description of accident, time of accident, reporting office first name, reporting office last name, summary of police report

This table will record all the relational data between a vehicle and a customer regarding accidents. One change made from the previous design was to make this relationship as a many-to-many This table has a many-to many type relationship and therefore we will create a unique auto-incrementing primary key. This will allow for us to identify each row without the use of a composite key and will prevent any multi-valued dependencies. Update, there will be no need to save a separate “time” attribute.

This means the table is in the **first normal form** since we took the original proposal attribute and split the non-atomic one so that we only record atomic data.

The table is also in **second normal form.** This is because we have chosen to include a unique auto-incrementing, single value, primary key to identify rows. This mean that there cannot be any partial dependencies.

The table is also in the **third normal form**. We can verify this by seeing if any of the dependencies get us any new or transitive information. They do not so we are in 3NF.

The table is also in **Boyce Codd normal form.** This is because our set of candidate keys, which is the minimum attribute(s) needed to uniquely identify rows. We only have one the assigned table primary key this will handle the many-to-many relationship between customer and vehicle. Since the candidate keys are single value and not disjoint BCNF is satisfied.

Finally, the table is in **fourth normal form.** This is because out primary key is only giving single valued dependencies and no multi-valued dependencies.

**Table**

**Review** (Review ID, customer ID, client ID, review body, rating)

**Functional dependencies:**

(PK) Review ID 🡪 (FK) customer ID, (FK) client ID, review body, rating

This table is a relationship between the customer and client entities. One change made from was that it became obvious that, much like the accident table, will be a many-to-many type relationship. This change meant that we needed to create a review ID primary key attribute to uniquely identify the tuples.

The table is in the **first normal form** since there are only atomic indivisible attributes that are being recorded.

The table is also in the **second normal form**. We know this since the primary key we chose/added is a single value attribute and therefore, cannot have any partial dependencies.

The table is also in the **third normal form.** This is verifiable by seeing if there are any new unique values returned by the looking at the dependencies. Nothing new can uniquely be identified by the values returned by the dependencies and therefore the table is normalized to 3NF.

The table is also in **Boyce Codd normal form.** We can check this by looking at our candidate keys. They are all single value attributes and there is only one which we have chosen to be the primary key. This means the table is normalized to BCNF.

The table is also in **fourth normal form.** This is because the primary key we added prevents any multi-valued dependencies by connecting and uniquely identifying the relationship between customer and client.

**Table**

**Payment** (Payment ID, subtotal, payment date, credit card number, expiration date, fees, taxes status, payment date)

**Functional dependencies:**

(PK) payment ID 🡪 total, payment date, credit card number, expiration date, fees, taxes payment date

This table will record all the information involved in the payment. There were a few modification that we made at this stage first was the status was not needed. This is because we will be able to verify that with the table “receives\_payment”. Another small change that was made here was changing subtotal to total. This is because in the “transaction” table we will record a “price” which will be the same as a subtotal and we therefore will not need to record it twice in two different locations.

The table is in the **first normal form.** This is because we have made all the attributes atomic i.e. they are indivisible.

The table is also in the **second normal form.** This is because the primary key we have chosen is not composite i.e., prime.

The table is also in **third normal form.** This is verifiable by taking all the values that are returned (dependencies) from the primary key and seeing if they can return any new unique values. They cannot so the table is in 3NF.

The table is also in **Boyce Codd normal form.** We know this because there is only one candidate key and this candidate key is a single value attribute. In turn we have chose this candidate key to be the primary key.

The table is also in the **fourth normal form.** This is because we don’t have any multi-valued dependencies to worry about.

**Table**

**Transaction** (Transaction ID, Customer ID, Payment ID, price, client approval, return status, request start date, request end date)

**Functional dependencies:**

(PK) transaction ID 🡪 (FK) Customer ID, (FK) Payment ID, (FK) vehicle ID, price, client approval, return status, request start date, request end date

(FK) payment ID 🡪 (FK) Customer ID, (PK) transaction ID, (FK) vehicle ID, price, client approval, return status, request start date, request end date

This table will store all the data related to the transaction relationship. This relationship is special as it is a junction table between payment, vehicle, and customer. Many customers can create transactions among many different vehicles. Each transaction will involve exactly one payment. Client approval will be a yes or no and will be determined by other back-end logic. Additionally we can remove the return status since we can just verify that the end date has passed with some logic.

The table satisfies **first normal form** since the values being recorded are atomic.

The table is also in the **second normal form** as it cannot possibly have a partial dependency with a single value attribute for a primary key.

The table is also in **third normal form.** This is because none of the values returned by the candidate keys will not get us any new or unique results. So, there are no transitive dependencies.

The table also is in **Boyce Codd normal form.** We have chosen a primary key from the set of candidate keys which are all singular in value, is not disjoint, and is our primary key.

The table will also be in the **fourth normal form** since we do not have multi-valued dependencies.

**Table**

**Receives\_payment** (payment ID, client ID, date payment was received)

**Functional dependencies:**

(PK) (FK) Payment ID 🡪 client ID, date payment was received

This table is to keep track of the relationship between clients and the payments they will receive. It is a many-to-one relationship as many different payments can be received by a single client.

The table is in **first normal form** since there are no attributes that are storing multiple types of data i.e. the attributes are atomic.

The table is also in the **second normal form** because there is no possible partial dependency if the determinate is a single valued attribute.

The table is also in **third normal form** since neither the client ID, or date payment was received will get us any new or unique information and the combination of the two is trivial.

The table is also in **Boyce Codd normal form** which is obvious since there is only one candidate key. The candidate key is a single value attribute, that is not disjoint, and we have chosen it to be out primary key.

The table will also clearly be in **fourth normal form** as well since we do not have any multi-valued dependencies.