Matthew Killeen
Professor Phipps – ICSI 410 Assignment #8

Table residentialComplexBuilding:

The residentialComplexBuilding table only contains one record, which is the residential complex building itself. The table will not be updated unless the building changes in some way, which translates to the column complexName (building name) or the column complexPhoneNumber being changed. I cannot logically see this table being a part of any searches or queries, as the table serves to keep record of the building in which everything in the database refers to; it does not need to be a part of any searches or queries. Selectivity, index size and individual column updates are disregarded as the table only has one record (this table serves to be more of a reference table / key-value pair table). NO INDEX CREATED FOR THIS TABLE.

Table address:

The address table consists of only one record, which is the address of the residential living complex. This table is not updated frequently, as the table and therein the columns of the table would only be updated if there was a change made to the address of the building, this being extremely unlikely. I cannot see this table being searched or queried, as it serves only to hold address information of the residential complex building, which is not significant for reports. Selectivity, index size and individual column updates are disregarded as the table only has one record (this table serves to be more of a reference table / key-value pair table). NO INDEX CREATED FOR THIS TABLE.

Table recreationLounge:

The recreationLounge table is not a large table, having only 16 records. This table will almost never be updated, as only changes in the fields recLoungeName (the name of the respective recreation lounge) and recLoungePhoneNumber would be the only reason this table would be updated. Updates to these fields are extremely rare. The table itself is not searched or queried significantly on its own, as it only contains records of the individual recreation lounges in the building. Selectivity is disregarded for this table, as 16 records is not much to sift through. Taking all of this into account, I will argue however that I can definitely see the column recLoungeName being referenced in WHERE clauses to generate reports very frequently. For example, I could see which residents utilized a certain recreation lounge which could give me insight on the demographics of the residents using it, how many times they use it, why they may use one recreation lounge over another, etc. In addition, I could easily see through using a WHERE clause where any of my many building employees are stationed using the name of an individual recreation lounge. On the other hand, an index of VARCHAR(50) does take up quite a bit of space. Despite the space problem, however, I choose to create a unique index (unique since each record represents a unique recreation lounge) on recLoungeName.

CREATE UNIQUE INDEX IX_RECREATIONLOUNGE_RECLOUNGENAME ON recreationLounge(recLoungeName)

<u>Table workAndComputerCenter:</u>

The workAndComputerCenter table is not a large table, having only 16 records. This table will almost never be updated, as only changes in the fields workCompName (the name of the respective work and computer center) and workCompPhoneNumber would be the only reason this table would be updated. Updates to these fields are extremely rare. The table itself is not searched or queried significantly on its own, as it only contains records of the individual work and computer centers in the building. Selectivity is disregarded for this table, as 16 records is not much to sift through. Taking all of this into account, I will argue however that I can definitely see the column workCompName being referenced in WHERE clauses to generate reports very frequently. For example, I could see which residents utilized a certain work and computer center which could give me insight on the demographics of the residents using it, how many times they use it, why they may use one work and computer center over another, etc. In addition, I could easily see through using a WHERE clause where any of my many building employees are stationed using the name of an individual work and computer center. On the other hand, an index of VARCHAR(50) does take up quite a bit of space. Despite the space problem, however, I choose to create a unique index (unique since each record represents a unique work and computer center) on workCompName.

```
CREATE UNIQUE INDEX IX_WORKANDCOMPUTERCENTER_WORKCOMPNAME ON
workAndComputerCenter(workCompName)
```

<u>Table exerciseRoom:</u>

The exerciseRoom table is not a large table, having only 16 records. This table will almost never be updated, as only changes in the fields exerciseRoomName (the name of the respective exercise room) and exerciseRoomPhoneNumber would be the only reason this table would be updated. Updates to these fields are extremely rare. The table itself is not searched or queried significantly on its own, as it only contains records of the individual exercise rooms in the building. Selectivity is disregarded for this table, as 16 records is not much to sift through. Taking all of this into account, I will argue however that I can definitely see the column exerciseRoomName being referenced in WHERE clauses to generate reports very frequently. For example, I could see which residents utilized a certain exercise room which could give me insight on the demographics of the residents using it, how many times they use it, why they may use one exercise room over another, etc. In addition, I could easily see through using a WHERE clause where any of my many building employees are stationed using the name of an individual exercise room. On the other hand, an index of VARCHAR(50) does take up quite a bit of space. Despite the space problem, however, I choose to create a unique index (unique since each record represents a unique exercise room) on exerciseRoomName.

```
CREATE UNIQUE INDEX IX_EXERCISEROOM_EXERCISEROOMNAME ON
exerciseRoom(exerciseRoomName)
```

<u>Table dayCare:</u>

The dayCare table is not a large table, having only 16 records. This table will almost never be updated, as only changes in the fields dayCareName (the name of the respective day care) and dayCarePhoneNumber would be the only reason this table would be updated. Updates to these fields are extremely rare. The table itself is not searched or queried significantly on its own, as it only contains records of the individual day cares in the building. Selectivity is disregarded for this table, as 16 records is not much to sift through. Taking all of this into account, I will argue however that I can definitely see the column dayCareName being referenced in WHERE clauses to generate reports very frequently. For example, I could see which residents utilized a certain day care which could give me insight on the demographics of the residents using it, how many times they use it, why they may use one day care over another, etc. In addition, I could easily see through using a WHERE clause where any of my many building employees are stationed using the name of an individual day care. On the other hand, an index of VARCHAR(50) does take up quite a bit of space. Despite the space problem, however, I choose to create a unique index (unique since each record represents a unique day care) on dayCareName.

CREATE UNIQUE INDEX IX_DAYCARE_DAYCARENAME ON dayCare(dayCareName)

Table parkingArea:

The parkingArea table is not large in size, as it only has two records representing the two parking areas at the residential living complex. The table is not updated regularly, as the only time it would be updated is if parkingAreaName (the name of the respective parking area) or the parkingAreaPhoneNumber was changed, which would rarely happen if at all. On its own, this table would not be searched and would seldom be queried. Selectivity is disregarded for this table as there are only 2 records (I would say it is more of a reference table). The only reason I could think of creating an index for this table would be so that I could see which residents use the two parking lots and how often, wherein I would create an index on parkingAreaName to use in WHERE clauses for reports. Since parkingAreaName is a VARCHAR(50) which is quite large for an index to be created on it, there are only two parking lots which would not give me very selective resident data and the fact that there are only two records in the table for SQL to read I am not going to create an index for this table. NO INDEX CREATED FOR THIS TABLE.

Table snackBar:

The snackBar table is not large at all, as it only holds one record which is the snack bar connected to the pool at the residential living complex. This table would only be updated in the event that snackBarName or snackBarPhoneNumber underwent changes, which would be extremely rare. On its own, this table is not searched and is seldom queried. Selectivity is disregarded for this table as there is only one record in it (I would say it is more of a reference table). The only reason I could think of creating an index for this table would be so that I could see which residents use the snack bar and how often, wherein I would create an index on snackBarName to use in WHERE clauses for reports. Since snackBarName is a VARCHAR(50) which is quite large for an index to be created on it, there is only one snack bar which would not give me very selective resident data and the fact that there is only one record in the table for SQL to read I am not going to create an index for this table. NO INDEX CREATED FOR THIS TABLE.

Table pool:

The pool table is not large at all, as it only holds one record which is the pool at the residential living complex. This table would only be updated in the event that poolName or poolPhoneNumber underwent changes, which would be extremely rare. On its own, this table is not searched and is seldom queried. Selectivity is disregarded for this table as there is only one record in it (I would say it is more of a reference table). The only reason I could think of creating an index for this table would be so that I could see which residents use the pool and how often, wherein I would create an index on poolName to use in WHERE clauses for reports. Since poolName is a VARCHAR(50) which is quite large for an index to be created on it, there is only pool which would not give me very selective resident data and the fact that there is only one record in the table for SQL to read I am not going to create an index for this table. NO INDEX CREATED FOR THIS TABLE.

Table playground:

The playground table is not large at all, as it only holds one record which is the playground at the residential living complex. This table would only be updated in the event that playgroundName or playgroundPhoneNumber underwent changes, which would be extremely rare. On its own, this table is not searched and is seldom queried. Selectivity is disregarded for this table as there is only one record in it (I would say it is more of a reference table). The only reason I could think of creating an index for this table would be so that I could see which residents use the playground and how often, wherein I would create an index on playgroundName to use in WHERE clauses for reports. Since playgroundName is a VARCHAR(50) which is quite large for an index to be created on it, there is only one playground which would not give me very selective resident data and the fact that there is only one record in the table for SQL to read I am not going to create an index for this table. NO INDEX CREATED FOR THIS TABLE.

Table restaurant:

The restaurant table is not large at all, as it only holds one record which is the restaurant at the residential living complex. This table would only be updated in the event that restaurantName or restaurantPhoneNumber underwent changes, which would be extremely rare. On its own, this table is not searched and is seldom queried. Selectivity is disregarded for this table as there is only one record in it. The only reason I could think of creating an index for this table would be so that I could see which residents use the restaurant and how often, wherein I would create an index on restaurantName to use in WHERE clauses for reports. Since restaurantName is a VARCHAR(50) which is quite large for an index to be created on it, there is only one restaurant which would not give me very selective resident data and the fact that there is only one record in the table for SQL to read I am not going to create an index for this table. I may decide to make one in the future however if I feel it could improve performance. NO INDEX CREATED FOR THIS TABLE.

Table familyUnit:

The familyUnit table is quite a large table compared to others in the database, currently having 160 records. I wouldn't say that this table is frequently updated, however two cases must be considered: the case when a new family unit comes to live in the residential living complex wherein data for the family unit must be created via an INSERT and the case when a family unit leaves the residential living complex wherein the family unit is not dropped from the familyUnit table but dateOfDeparture is updated with a value. Taking these two cases into account, with respect to time this table would only stand to increase in size with occasional updates. I could definitely see this table being involved in quite a few searches and queries on its own. For example, you may want to see which family units arrived or left during a certain time period, you may want to see how many members the family unit consists of, you may want to join it with the resident table to see which residents are within which family units, etc. As for selectivity, the family unit table has high selectivity when you take into account that it is a table grounded in time due to the columns dateOfArrival and dateOfDeparture. I could limit the search based upon the values of these columns and the time periods I am looking for. Furthermore, I could definitely see familyUnitName being used in WHERE clauses to see the activity of members of respective family units within the residential living complex. familyUnitName being VARCHAR(50) is a bit large compared to dateOfArrival, dateOfDeparture and numberOfMembers, which are 8 bytes, 8 bytes and 4 bytes respectively. I think it would be best to create an index on both familyUnitName and dateOfArrival, which will include numberOfMembers. I am leaving dateOfDeparture out of the index as it is NULL due to the fact that it is only given a value once and if a family unit leaves the complex. familyUnitName and dateOfArrival will never be updated unless there is a data entry error made and numberOfMembers would only be updated if the size of the famiy unit increases or decreases, which is rare. I feel that I may want to update this index in time, but for now I feel that this is the best choice.

CREATE INDEX IX_FAMILYUNIT_FAMILYUNITNAME_DATEOFARRIVAL ON familyUnit(familyUnitName, dateOfArrival) include (numberOfMembers)

Table floor:

The floor table is a larger table compared to others in the database, with a total of 160 records. This coincides with the number of family units currently living within the residential living complex. The floor table is updated when a family unit starts living at the residential complex and therein starts living on a particular floor and when a family unit leaves the residential living complex and therein stops living on a particular floor. This being said, the floor table would occasionally be updated; it would be updated a bit less than the familyUnit table for a comparison. I cannot imagine this table being searched on its own, with queries being seldom; the only case scenario I could possibly see would be if I wanted to see which family units lived which floors, which would only occur every once in a while. This table has somewhat low selectivity, as the search I just described would only serve to divide the family units on each floor into 8 different groups, the number of floors that the building has. The one column present in the floor table that is not a type of key is floorNumber, which is of type TINYINT and takes up 4 bytes of memory. In addition, floorNumber would never be updated unless a mistake in data entry was made. However, making an index on floorNumber would be seemingly useless to me, as it is the only non-key column present. Therefore, I feel that there is no need to create an index for the floor table. NO INDEX CREATED FOR THIS TABLE.

<u>Table roomType:</u>

The roomType table only contains 4 records, these records being descriptions of the different types of rooms within the residential living complex. The table is never updated, it is merely a reference table. The table, on its own, will never be searched and will seldom be used in queries. The selectivity for this table is high since you can separate each room type into a group, but it doesn't matter, since there are only four records for the 4 types of rooms which is not hard to separate at all. The roomDescription column takes up quite a bit of space being VARCHAR(30). Again, these columns will never be updated unless there is a data entry error when creating them. There is no need to create an index for this table. NO INDEX CREATED FOR THIS TABLE.

<u>Table room:</u>

The room table is a large table compared to others in the database, having 160 records which represent all the individual rooms in the residential complex building. This table would be updated just as frequently as the floor table, as the only reason it would be updated is if a family unit begins to occupy a room or a family unit leaves a room. This being said, it would be updated every once in a while with respect to time. On its own, I don't see this table being searched or queried often. The only case in which this table would be searched on its own would be to see if a certain room is occupied or vacant (if the room has a valid familyUnitId as a foreign key), it would be seldom queried. Selectivity for this table is somewhat high, as I could limit my search to rooms that coincide with certain numbers (for example, find rooms where roomNumber ends in '16' or begins with '1') but I could not give a reason as to why I would like to do this. On the other hand, however, I could easily see the roomNumber being a part of a lot of searches and queries when used to analyze the resident table. For example, I may want to join the resident table with roomNumber to see which room each resident inhabits, I may want to see which family unit inhabits a respective room if it is occupied, I may want to see the demographics of the family units that inhabit each respective room, I may want to see how long a family unit stayed in a respective room, etc. In addition, roomNumber will never be updated unless a data entry error is made upon insertion and roomNumber takes up only 3 bytes, being a CHAR(3). Therefore, taking all of this information into consideration, I am creating a unique index (unique since each room number is unique) on roomNumber.

```
CREATE UNIQUE INDEX IX_ROOM_ROOMNUMBER ON room(roomNumber)
```

<u>Table resident:</u>

The resident table is one of the largest tables in the database, having 400 records representing each resident living or who has lived in the complex building. This table is updated somewhat frequently, as when a family unit arrives one or more records must be inserted into the resident table. In addition, changes to employment status and age (age is a horrible thing to store in a database, will be removed in future versions) will require updates to individual records.

Therefore, this table stands to increase in size with respect to time and will therefore be updated even more frequently as there will be more records to update. I can definitely see this table being searched and queried a lot to get insight on the residents who have lived and are currently living at the residential living complex. After all, knowing information about your past and present residents is the only way you can make the residential living complex better for future occupants. This table has extremely high selectivity, as I can limit my search based on firstName, middleInitial, lastName, dateOfBirth, isEmployed and isMale (gender). Furthermore, I could definitely see it being beneficial to create an index for this table on dateOfBirth, which only takes up 8 bytes and is never updated and can be used in GROUP BY clauses, wherein I can organize residents based upon dateOfBirth. I will also include isEmployed and isMale on the index, since I would like to know the employment status and gender of those residents I am searching and since these columns are BIT and take up 2 bytes in total. I may change or delete this index and create a new one if I feel it is not being used or I see that a better one could be created to improve performance.

CREATE INDEX IX_RESIDENT_DATEOFBIRTH ON resident(dateOfBirth) include (isEmployed, isMale)

Table residentPhoneNumber:

The residentPhoneNumber table is one of the larger tables in the database, as it has 400 records representing an individual phone number for each resident. This table will be updated as new residents begin to live in the residential living complex or one of the residents gets a new phone number. The first case is more frequent than the second, with total updates being somewhat frequent. This table is not a part of any searches, and I can only imagine it being a part of a query that it with the resident table to have residents and their phone numbers side by side. Other types of queries would likely be seldom. Selectivity is high for this table as I can specify the residentId in a WHERE clause to find the exact phone number for that resident. In addition, phoneNumber only takes up 12 bytes as it is CHAR(12). The column phoneNumber will only be updated in the case that a resident changes his or her phone number, which is infrequent. Taking all of this into account, however, it is pretty useless to create an index for this table in my eyes as only has one non-key column, which is phoneNumber. Therefore, creating an index wouldn’t generate much value and would just slow down my inserts and updates. NO INDEX CREATED FOR THIS TABLE.

Table buildingManager:

The buildingManager table only contains one record, which is the record for the building manager. The table will not be updated unless the buildingManager changes any part of his or her name (firstName, middleInitial, lastName columns respectively), he or she changes his or her phone number or a new building manager takes the old building manager’s place. These are all extremely unlikely or extremely infrequent. I cannot logically see this table being a part of any searches or queries, as the table serves to keep record of the building manager; it does not need to be a part of any searches or queries. Selectivity, index size and individual column updates are disregarded as the table only has one record (this table is more of a reference table / key-value pair table). NO INDEX CREATED FOR THIS TABLE.

Table employee:

The employee table is one of the largest tables in the database, having 300 records representing each employee who has worked or is currently working for the residential living complex. This table is updated somewhat frequently, as when a new employee is hired a record for the employee must be inserted into the table. Therefore, this table stands to increase in size with respect to time. I can definitely see this table being searched and queried a lot to get insight on the employees who have worked and are currently working for the residential living complex. This table has extremely high selectivity, as I can limit my search based on firstName, middleInitial, lastName dateOfBirth and isMale (gender). Furthermore, I could definitely see it being beneficial to create an index for this table on dateOfBirth, which only takes up 8 bytes and is never updated and can be used in GROUP BY clauses, wherein I can organize employees based upon dateOfBirth. I will also include isMale on the index, since I would like to know the gender of those residents I am searching and since this column is of type BIT and takes up only 1 byte. I may change or delete this index and create a new one if I feel it is not being used or I see that a better one could be created to improve performance.

```
CREATE INDEX IX_EMPLOYEE_DATEOFBIRTH ON employee(dateOfBirth) include
(isMale)
```

Table facilityCheckIn:

The facilityCheckIn table is the largest table in the database, consisting of about 1400 records. This table will be updated constantly due to the fact that records are inserted every time a resident uses a particular facility within the database. With respect to time, this table could grow exponentially ultimately having tens of thousands of records, most likely more. This table would be searched and queried to view the activity of residents within the residential living complex, aiming to see which facilities they are utilizing and how often they are utilizing them. Doing this also gives insight into the demographics of residents who use certain facilities. This table has extremely high selectivity, as I can limit my search based upon the foreign keys that are present within the table representing individual residents and individual facilities within the residential living complex. Once a record is inserted into the table, it will not be altered as it there to keep a historical record of a resident using a particular facility. In addition, none of the columns are particularly large as they are all INT and take up 4 bytes respectively. This is a tough decision, as I would like to avoid searching through all of the records when trying to track the facility usage of residents within the residential complex though the table is inserted at such a rapid rate I can't justify creating an index for this table as it would slow that process down. NO INDEX CREATED FOR THIS TABLE.

Table residentEmployee:

The residentEmployee table is one of the larger tables within the database, having 100 rows. This table will be updated regularly as it is meant to keep track of notable interactions between an individual resident and an individual employee. Throughout the course of time, hundreds if not thousands of notable interactions could take place, therefore needing to be inserted as records within the table. This table, on its own, would not be searched or queried too frequently as I could only see this table being searched to see which residents have interacted with which employees and vice versa, which in itself is somewhat of a niche search. Due to the potentially large size of this table, selectivity is very high as I can narrow my search down to an individual resident's interactions with various employees and again vice versa. In addition, the columns present in the table other than the primary key are two foreign keys referring to resident and employee respectively. These are of type INT and altogether only take up 8 bytes. These columns will also never be updated once the record is created unless there is an error in data entry. I find myself in the same situation as with the facilityCheckIn table: I would like to dwindle my search down using an index since there will be many records in this table, though this will compromise the speed of insertions into the table. This being said, I am going to choose not to create an index for this table for that reason, though I may choose to create an index for this table in the future if it seems more advantageous as time goes on. NO INDEX CREATED FOR THIS TABLE.