**Question 1**

* + Relational Table

The concept of Relational Table is the relational database table storage data in relational. With each row is an instance and each column is a filed of instance. Table also can be relational with others through foreign key. The relation with others tables can be one to many, many to one, and many to many. Relational Table can solve most of data storage in most project, most model can be represented and mapped to relation tables, then can create relations with others. Relational table can represent data as traditional way, most of traditional stuff can be solved. In our class we use SQLite a smaller relational database to do all the CRUD in project. It is a good idea when there are some models can be represented by relational easily, like user account, point of information. When the big data came out, there are couple more text pattern data need to storage and query, the relational table is not a good solution.

* + Index

Index is make query operation faster methods in database, index more often created on primary key in table. After it have been created, databases can do binary search on table and query faster. Index is for speed up query problem existing. When there are millions instances in one table, scan all table needs lots of time, after create index on primary, database can binary search on tables and in LogN time can find the result. In this class, I used it in Chinbook database compare query with and without index on tables. It is a great idea to speed up query when items number is very large. However, if in a situation there are a lot of insert, delete operation in one table, index seems is not very efficient, because there needs rebuilt the index after insert and delete instances, which is also time consuming. Therefore, we need to depend the situation to decide whether there is necessary to create index on database.

* + Covering Index

Covering index is to create an index cover all filed of we need to query. Traditional index is created on only one filed, however if there needs to query more than one fields in one query, database needs to get the information back from original table. Hence, using covering index is very efficient for query more than one field, there is no necessary for database to look up information back at original tables, one query can get all needs. In our course I create covering index on Chinbook to make comparison without index and without covering index. Same as indexing, it is good idea for speed up query in lots of instances tables, additional it is good idea for multi filed query. However, it is not good idea when there are lots of insert and delete operation in this table, the covering index also need to rebuilt when there some new data be inserted.

* + Foreign Key

Foreign key is kind of constrain on one field, which in detail is about one table field can use reference another table primary key to represent on instance in that table. In some situations, in multi tables database, one table data has been change there is no necessary to change the related table data, database will through foreign key to find the modified data without errors. In additional database will check the validate for foreign key in database like the foreign key is existing in related tables. In our course I used it Chinbook, there are couples more foreign key constrains between tables, like albums, artists, songs, playlists. It is good idea to create foreign keys when there are some relations we need to create, and the database will take care all optimization for query. However, it is not good idea to create foreign key when there is tons of CRUD operations, each time the database will check this constrain and cause lots of time.

* + ACID

ACID is stand for Atomicity, Consistency, Isolation, Durability when we try to modify database. I will give concepts of all of them separately. Atomicity is the database will make guarantee that one transaction will finished all of them or none of them. Consistency means before and after the transaction, the database will still be satisficed constrained, which means all the input data meet the constrains. Isolations means one transaction will only access itself data when it is not completed, even two transactions happened in same time. Durability means that the transaction is complete, the data will be existing in the database and make storage. The ACID is the guarantee for database write and update data is correct. Through this ACID, the operations in transaction will follow some rules and make all data as we want when some concurrence stuff happened. In our class we have mentioned this concept, I use this ACID in insert data in database in my final project. It is good idea ACID exist in translations, it is good guarantee for data correctly, however it still need time to check the constrain.

* + Object Relationship Mapper

The concept of ORM is database row mapping to an object instance in program, each fields in database table will mapping to an object attribute, and the ORM will encapsulate all query in it, CURD operations will mapper SQL query statements in database. Through ORM we can treat database items like object variables in program, the program looks like more elegant, we can change the database without knowing much about how the operations happens. In our course, we have mentioned two ORM peewee and SQLAlchemy, in to-do project we used peewee as our main ORM connect with sqlite, tinydb and mongo. It is good idea when facing much mapping stuff between database and programming, without manual mapping ORM can handle all the work for you. ORM can make good solution on most of situation, however some situation is not, it will cause some performance issue, and make thing mess, hence before use ORM there need to think it carefully how it be used and how it will be used.

* + Document Database

Document database is unlike relational database, it storage like store all object, like a json file or xml file. There is some object cannot be represented by relational database, therefore we came up the document database to store data. Some unstructured data also can be stored in database. We use TinyDB and MongoDB in our course, in my project I use mongoDB on mongoLab as NoSQL database to store geojson data. It is good idea to make storage some unstructured data like document and json file in document database, it can handle it perfectly. As traditionally, the structure still need relational database.

* + Database replication

Database replication means same data set existing on different database server. The reason why need replication is we need increasing guarantee data do not loss, and increasing the read capacity of same data. When database server stores some important data, of course we do not want loss it, replication like back up all the data in to another server, and we some one of them down, we can make resotre. At the same time is a database has a heavy read load, increasing replication number can increase the capacity of read data, because client can get data from one of them, the read load can be averaged by all of them. We have demo it in mongoDB it shows the 2 different replication on Codio. It is good idea for back up all data and make it reliable, but also it will cost your money to set up an new server make it more reliable.

* + Sharding

Sharding means in document oriented database, one collection can be divided into multi part in to different server, make a master node knows each parts existing location. As data growth there is some requirement for make storage in different server. Share is for these problem existing, it through shard key to arrange item should be store at which server. In our course we have demo a MonogoDB sharding in two machine. With sharing each query statement will be rearranged into master server, master server query in multi machine and return the result. Sharding is good for performance in large data set, and when the shard key is good, the distribution will be good. However, if there is no good option for shard key, the sharding is not very efficient.

* + Map/Reduce

Mapreduce concept is for parallel programming in big data, it contains two parts. First is mapping, which means divide whole document into word, after shuffling all same word came into one reducer. Second is reducing, it will combine all same words together, count the number or do some other jobs. Mapreduce pattern create a new way to do parallel computing in multi machine, at same time different can do different jobs, according the name server, the data can be separated into different machine, then do mapping and reducing. The big data process needs this kind of pattern it will cause cluster working make whole process faster. I have take Big Data course in last semester, I use mrjobs in python to do project in analysis yelp academic dataset. It is really good and efficient pattern for handle big data process, as the node increasing the whole part job can be arranged into them and working together to get the final result.

**Question 2**

1. Imagine you were asked to create a database to hold the Kent State University student records.

1. What are some questions you would want to ask?

What is features will be supported about this database?

What kind of information will be storage in this database?

How many the records will be in this database?

Which data in this database needs to do backup?

How many servers will be in database cluster?

How many DBA will be use this database?

Do you prefer free or commercial database?

2. Pretend you are a stakeholder, and answer those questions.

This database will support web portal system, provide query information from web server.

This database will store students’ information, staff information, and students information, in additional if students have work in campus, there should support in database.

The number of students will be 30,000 and the staff is 10,000, consider we have 5000 newcomers, the database show has extendibility, the course will be 1000.

The database needs backup every week, some important data should backup per day.

We can provide 10 at most database server for our system.

There will less 5 people for manage this database.

This university don’t want provide any funding on software support, so we only choose free.

3. Given the answer to those questions, in a few paragraphs, describe your approach to creating the database, including:

* + Type of database

The database will be relational database, using MySQL.

* + Design of the schema

This database will contain these tables:

-Student information table, it contains (id, last, first, bannerID, password, email, address, phone, major)

-Employee information table (id, last, first, bannerID, password, email, address, phone, position)

-Course table (id, name, credits, instructor id)

-Take course table (stu.id, stu.name, major, course.id, course.name, grade)

-Jobs in campus table (id, name, description, salary, location, level)

-Take job table (employee.id, employee.name, startdate, job.id, job.name, benefits)

There are some FK at TakeCourse, TakeJobs table, which relate to student, employee, course table’ primary key id.

The SQL statement is like below:

CREATE TABLE IF NOT EXISTS `Course` (

`id` int(11) NOT NULL,

PRIMARY KEY (`id`),

`name` int(11) NOT NULL,

`credits` double NOT NULL,

`instructorid` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `Employee` (

`id` int(11) NOT NULL,

PRIMARY KEY (`id`),

`last` varchar(100) NOT NULL,

`first` varchar(100) NOT NULL,

`bannerid` varchar(100) NOT NULL,

`password` varchar(100) NOT NULL,

`email` varchar(100) NOT NULL,

`phone` varchar(100) NOT NULL,

`position` varchar(100) NOT NULL,

`address` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `Jobs` (

`id` int(11) NOT NULL,

`name` varchar(100) NOT NULL,

`description` varchar(100) NOT NULL,

`salary` double NOT NULL,

`location` varchar(100) NOT NULL,

`level` int(11) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `Student` (

`id` int(11) NOT NULL,

`last` varchar(100) NOT NULL,

`first` varchar(100) NOT NULL,

`bannerid` varchar(100) NOT NULL,

`email` varchar(100) NOT NULL,

`address` varchar(100) NOT NULL,

`phone` varchar(100) NOT NULL,

`major` varchar(100) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `TakeCourse` (

`stu.id` int(11) NOT NULL,

`stu.name` varchar(100) NOT NULL,

`stu.major` varchar(100) NOT NULL,

`course.id` int(11) NOT NULL,

`course.name` varchar(100) NOT NULL,

`grade` double NOT NULL,

`instructor.id` int(11) NOT NULL,

FOREIGN KEY (`instructor.id`) REFERENCES `Employee` (`id`),

FOREIGN KEY (`stu.id`) REFERENCES `Student` (`id`),

FOREIGN KEY (`course.id`) REFERENCES `Course` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

CREATE TABLE IF NOT EXISTS `TakeJob` (

`emp.id` int(11) NOT NULL,

`emp.name` varchar(100) NOT NULL,

`startdate` varchar(100) NOT NULL,

`job.id` int(11) NOT NULL,

`job.name` varchar(100) NOT NULL,

`benefits` varchar(100) NOT NULL,

`TakeJob\_ibfk\_2` FOREIGN KEY (`job.id`) REFERENCES `Jobs` (`id`),

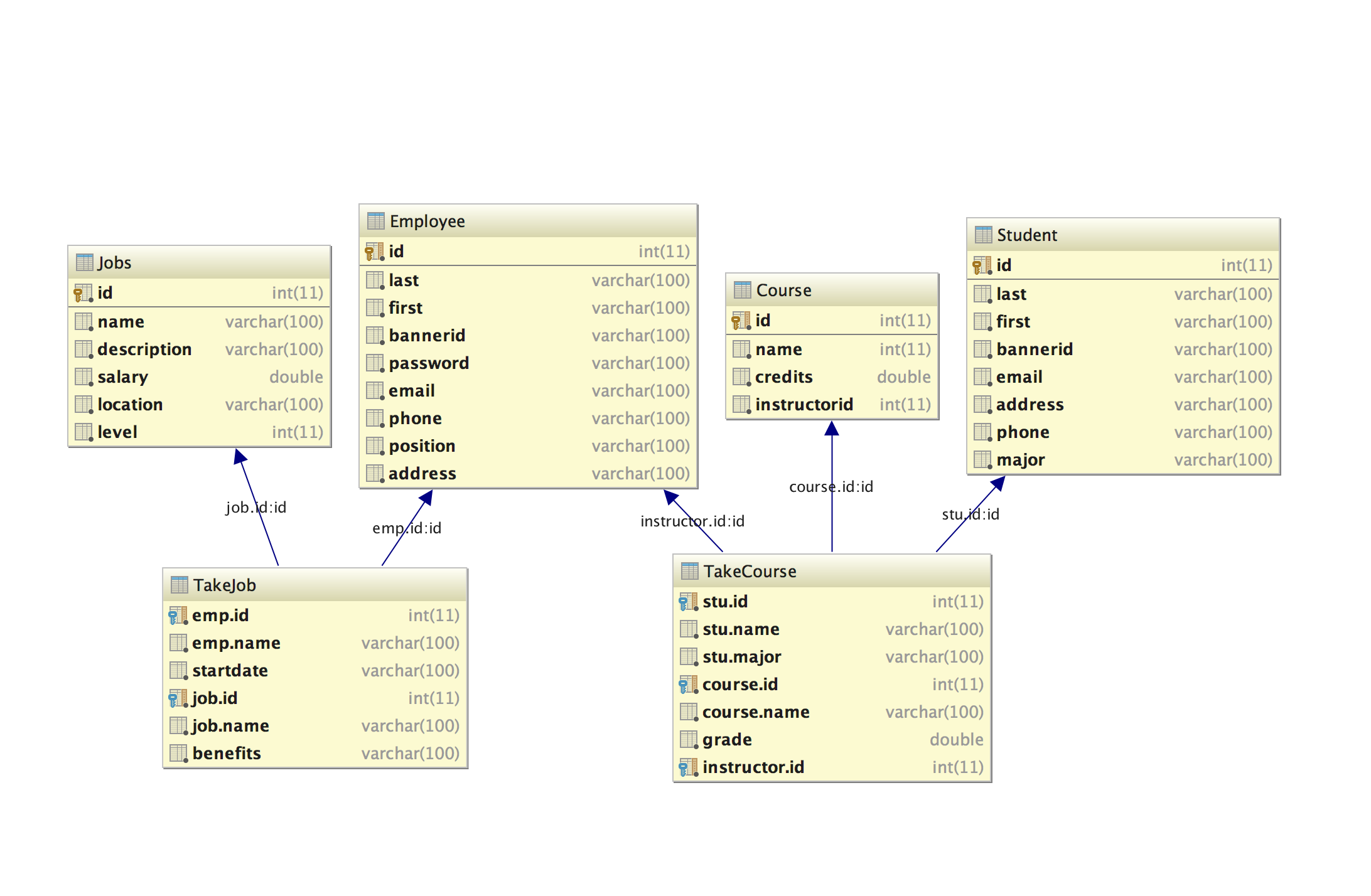
`TakeJob\_ibfk\_1` FOREIGN KEY (`emp.id`) REFERENCES `Employee` (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=latin1;

* + Details of database inputs and outputs

This database Input including: student info, employee info, course info, jobs info

Output including: query student info, query employee info, query course info, query jobs info, query student take course info, query employee take jobs info.



* + Reasons for these decisions

I choose use this database design based on these reasons: The schema is extendable I can add more information into each table without changing the schema. These tables are enough to store all the information we need to store. In future if there are any increasing need, we can add more tables and create relation with existing tables. We can see the most relation is existing in the TakeCourse table and TakeJobs table. If there are some new relation need, we can create more table to take care of that. On each table there is an index according the primary key, hence the query speed is acceptable under 100,000 records. It can represent most of requirement of university. It is simple enough to do a lot query. It is easy to backup and restore in different machine. The backup can be located on 10 duplicate servers on every day.

* + Expected risks and benefits from selecting your recommendations.

With the number of student keep increasing, the records might go over 100,000 records, it will cause some performance problem. The system should do some performance optimization if the query speed is low at future. There are lots of data existing in TakeCourse table and TakeJob table, therefore if the course and jobs number is very large, there are some performance issue will happen. Then we can think about other relational schema to redesign it and make it more optimizer.

4. Briefly propose one alternative approach, and explain why your initial approach was preferable.

An alternative plan is using documented oriented database. According the requirement, the free database, I will try Apache Cassandra, it an open source NoSQL database. This database system contains a lot of documented oriented attributes. Tt can replace some features in MySQL, the backup of Cassandra is also simple and easy. This whole information system still is good for traditional relational database. The schema is still strong and flexible. Some well developed methods have been deployed on thousand of server. There are a lot of solutions in relational database as NoSQL database. I prefer use this relational database in this project. I think well design on the table and do some optimization in high frequency query table will make this project running perfectly.

|  |
| --- |
|  |