**MSDS 7330 File Organization and Database Management**

**Mid Term Exam Answer Sheet**

Name:

Date:

1. D

D - In each answer (A, B, and C) there is a characteristic of how/what the Data Definition Language does regarding specific characteristics of a database schema. In the Java computer language (as well as C and C++), “declarations” do not create an object in a database (answer A). Declarations in Java will identify a **variable** or even a **function**. The Java language also doesn’t allow for consistency constraints to be specified directly (answer B). Java does, however, specify data types like variables (again) for the purpose of the given program being written. The language itself is not used to give specific access rights to users in a database like the DDL (answer C). Type declaration languages such as Java can be used to create a program that does each of the DDL functions, but the type definition languages themselves do not directly allow this.

2. D

D - In a file processing system, the whole issue with that system is the multitude of application programs used to access the data. For example, you may need a program to access financial information, but another to access customer data and another one to access location information and product information. Those programs may talk to one another now and days (think Google Drive apps with scripts), but access to the information doesn’t have a universal language or app. That brings the issues of inconsistent data, atomicity issues, and access anomalies, and more. DBMS are much easier and intuitive to manage.

3. A

A - The first step in designing and setting up a database is to define the business requirements to ensure the database meets the present and future needs of the business. Through the process of interacting with the business teams and subject matter experts, the user requirements are defined which is then used to design the database.

Once the enterprise requirements are known, then next step is to define the data integrity constraints; which enforce the business rules gathered from the requirements. The next step is to define the physical implementation of the database, from the logical model, which creates the schema DDL for a given target RDBMS. The last step in setting up an enterprise database is to execute the schema DDL generated from the physical model to create and initialize the database.

4. C or D (Preferred) Both Acceptable

C - This occurs during the physical-design phase, one of the last phase when designing a database. Once we have established how the database conceptually works and ensure that it meets the needs of the client, we have to create and initialize the database to test it and ensure that it works. That comes before defining how it is physically implemented, because we need to see how it works to see what the best kind of memory it should be stored in, or where the servers should be located in a company’s IT structure.

D - The database should only be initialized after all requirements are understood. The physical implementation and integrity constraints also have a strong influence on the design of a database and should be understood before starting the database.

5. C

C - The three abstraction levels in a database are: physical, logical and view. The lowest level of database abstraction is the physical level which contains low level detail of how storage and data structures are defined for a target RDBMS; this is the level that the DBA typically works at. The next level up is the logical level which contains what data is being stored and describes the relationships between the data. The Data Architect typically works at this level as a tool to communicate with the business teams to ensure business requirements and rules are accurately captured. The view is the highest level of database abstraction and used to show a high-level view of the various entities, data captured, and their interactions.

6. A

A - The Entity Relationship Model are entities (or things) and the relationships that exist between them. These related entities make up entity sets. There are also attributes that make distinguishing characteristics between entity sets. Combining all of these together creates the ER Model. SQL does an excellent job of bringing these various relationships together so that they all make sense. It also brings the various database elements together very nicely.

7. E

E - The Data Definition Language is what we use to set up the database schema (A). Part of that schema does include how the data is stored and accessed through statements in DDL (B and D) and the constraints of what the database is supposed to do (Consistency Constraints, C). That comes with domains for the values in the data, like if an account balance can be negative or if a professor can teach in multiple departments. This is all done with DDL.

8. E

E - The answer is **all of the above**, as DDL does in fact perform all of the functions listed in answers A through D. DDL is used to specify properties of the data in a database schema (answer A). There is a portion of the DDL used to specify the storage structure used by the database (answer B) which is called data storage and definition language. DDL also can specify consistency constraints (answer C). An example would be a database for a retail system that is only allowed to accept positive product prices for merchandise. DDL can do this for such a system. Finally, access methods used in the database (answer D) is done by DDL. There are various authorization methods in a database like *read*, *update*, and *delete*. DDL controls these instructions.

9. B

B - The answer is three-tier architecture for a number of reasons. Users of large applications over the World Wide Web only “need” an interface to gather the data they are trying to see. The World Wide Web is only one application anyway. I think most people confuse the Internet (which is a massive logical interconnection of computers) with the World Wide Web, which is a huge mistake. Additionally, the common user does not have enough knowledge to use a two-tier application. You need to have a certain level of sophistication to use languages that interface with databases in a two-tiered architecture. Three-tiered architectures make usability *friendlier* to the novice user by placing an application GUI in front of a user on a laptop or PC. Moreover, it’s even more common today that a user would be accessing these applications on an iPhone. Finally, three-tiered architectures allow for better performance than two-tiered architectures. Application and database performance can be tuned to deliver richer content with the structure that a three-tiered architecture provides.

10. D

D - Codd’s 1970 paper addressed the challenge of keeping a growing store of data consistent and accessible. As data are added to a ‘data bank’, relationships between relations can be checked for consistency. This is still a challenge in DB design, and these inconsistencies can be attenuated by eliminating redundancy with normalization and specifying constraints.

11. B (allowed C with proper justification)

B - The primary key is the candidate key chosen to uniquely identify a tuple in a relation. The primary key uniquely identifies a tuple because no two tuples in a given relation can have the same key value and a relation can only have a single primary key defined for it. For example, in the following relation (SSN) or (ID) are two possible primary keys.

Employee(ID, Name, DOB, SSN, StartDate, Salary)

C - A superkey is practically defined by the question. Superkeys are used to find a row or tuple in a relationship by using one or more attributes to eliminate all other possibilities. While there are often many possibilities of super keys, the best choices are narrowed to Candidate keys and Primary keys.

12. C

C - The set of keys which uniquely identify a tuple are the superkeys of a relation. If no proper subset of a superkey’s attributes also form a superkey, it is a minimal superkey, making it a candidate primary key.

13. A

A - The primary key of a relation is chosen from the set of candidate keys for the relation. A candidate key must be a minimal superkey for the relation. A minimal superkey is any superkey for which no proper subset of its attributes is a superkey.

14. D

D - Foreign keys are used to enforce referential integrity across the database. A foreign key is an attribute in one table that is a key in another table. The dependency on update/insert/delete between these two (or more) tables relies on the values contained to remain in sync.

15. C

C - The date is a non unique value within the table. StudentID and TutorID could be used but it is possible the student could have a same tutor assigned for different classesnand the value of TutorID would not change which could then cause duplication. The StudentID and Tutemail would risk the same issue as StudentID and TutorID. The combination of CourseID, StudentID, and TutorID is the best choice for a primary key since even if the TutorID is duplicated, the combination of CourseID and StudentID would ensure uniqueness.

16. D

D - *CourseID, StudentID,* and *TutorID* are all likely to be foreign keys because there is probably more information that the school would save regarding courses (sections, instructors), students (enrollment status, name), and tutors (name, subjects, payroll). These keys would reference tables containing information on all courses, students and tutors with *CourseID, StudentID, TutorID* as their respective primary keys.

17. Two possible answers:

17a -

First normal forma (1NF): No two rows of data can contain repeating information.

Second normal form (2NF): The relation must be in 1NF. Relations after 2NF are:

Student (StudentID, Date)

Course (CourseID, Topic, Book, Grade, Room)

Tutor (TutorID, TutEmail)

Third normal form (3NF): The relation must be in 2NF. The tables after 3NF are:

**Student\_Course** (StudentID, Date, Grade, CourseID, TutorID) [Student from 2NF is renamed to Student\_Course in 3NF]

**Course** (CourseID, Topic, Book, Room)

**Tutor** (TutorID, TutEmail)

1. Relations in 3NF is shown as follows,

Relation *Student*{ *CourseID, StudentID, TutorID, Grade* }

|  |  |  |  |
| --- | --- | --- | --- |
| *CourseID* | *StudentID* | *TutorID* | *Grade* |
| U1 | St1 | Tut1 | 4.7 |
| U2 | St1 | Tut3 | 5.1 |
| U1 | St4 | Tut1 | 4.3 |
| U5 | St2 | Tut3 | 4.9 |
| U4 | St2 | Tut5 | 5.0 |

Relation *Course*{ *CourseID, Date, Topic, Room, Book* }

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *CourseID* | *Date* | *Topic* | *Room* | *Book* |
| U1 | 23.02.03 | GMT | 629 | Deumlich |
| U2 | 18.11.02 | Gin | 631 | Zehnder |
| U4 | 04.07.03 | AVQ | 621 | SwissTopo |
| U5 | 05.05.03 | PhF | 632 | Dmmlers |

Relation *Tutor*{ *TutorID, TutorEmail* }

|  |  |
| --- | --- |
| *TutorID* | *TutorEmail* |
| Tut1 | tut1@fhbb.ch |
| Tut3 | tut3@fhbb.ch |
| Tut5 | tut5@fhbb.ch |

17b –

Functional Dependencies.

(PK) CourseID, StudentID, TutorID

TutorID -> TutEmail

CourseID -> Book

StudentID, CourseID -> Grade

TutorID -> Topic

New Tables

1) STUDENT-TUTOR

|  |  |  |
| --- | --- | --- |
| StudentID | CourseID | TutorID |
| St1 | U1 | Tut1 |
| St1 | U2 | Tut3 |
| St2 | U5 | Tut3 |
| St2 | U4 | Tut5 |
| St4 | U1 | Tut1 |

2) TUTOR-SESSIONS

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Topic | Room | TutorID |
| 18.11.02 | GIn | 631 | Tut1 |
| 23.02.03 | GMT | 629 | Tut3 |
| 23.02.03 | GMT | 629 | Tut1 |
| 05.05.03 | PhF | 632 | Tut3 |
| 04.07.03 | AVQ | 621 | Tut1 |

3) CANDIDATE-TUTORS

|  |  |
| --- | --- |
| Tutor | Topic |
| Tut1 | GMT |
| Tut3 | GIn |
| Tut3 | PhF |
| Tut5 | AVQ |

4) TUTORS

|  |  |  |
| --- | --- | --- |
| StudentID | CourseID | Grade |
| St1 | U1 | 4.7 |
| St1 | U2 | 5.1 |
| St2 | U4 | 5.0 |
| St2 | U5 | 4.9 |
| St4 | U1 | 4.3 |

|  |  |
| --- | --- |
| TutorID | TutEmail |
| Tut1 | tut1@fhbb.ch |
| Tut3 | [tut3@fhbb.ch](mailto:tut3@fhbb.ch) |
| Tut5 | tut5@fhbb.ch |

5) TAKES

6) COURSES

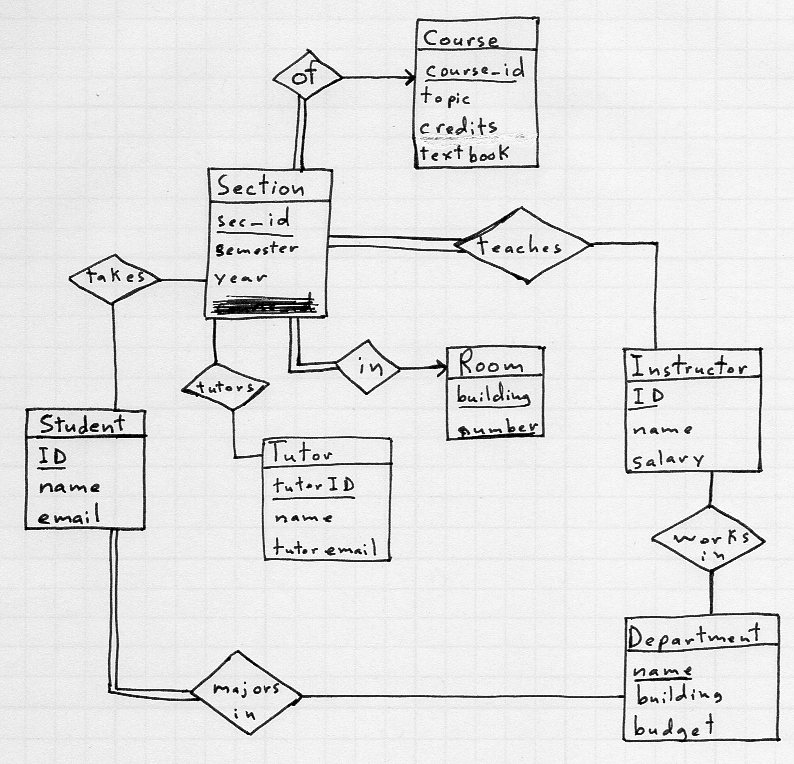
|  |  |
| --- | --- |
| CourseID | Book |
| U1 | Deumlich |
| U2 | Zehnder |
| U4 | SwissTopo |
| U5 | Dmmlers |

7) STUDENTS

|  |
| --- |
| StudentID |
| St1 |
| St2 |
| St4 |

18.

A simple model to describe a school. A *student* **takes** a *section* **of** a *course* **in** a *room*, and **majors** in a *department*. An *instructor* **teaches** one or more *sections* **of** a *course* **in** a *room* and **works** in a *department*. A *tutor* **tutors** a *section* **of** a *course*.



19. A

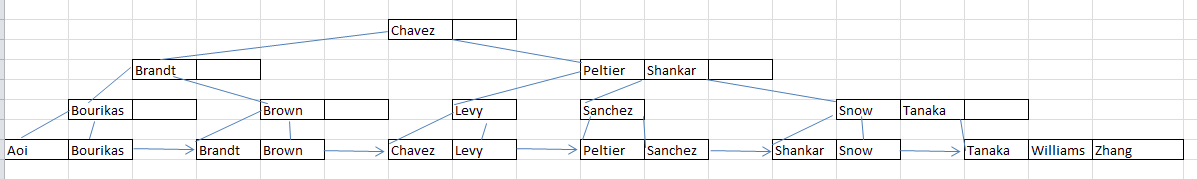
A - PersonName is the best choice given the available answers. The companyName, salary, and hireDate could all be duplicates given the data. The key of PersonName, hireDate would further the uniqueness since a common name such as John Smith could be duplicated in a large company but the hireDate would most likely mitigate duplicates.

20. C

C - The key of PersonName, hireDate would further the uniqueness since a common name such as John Smith could be duplicated in a large company but the hireDate would most likely mitigate duplicates.

21.

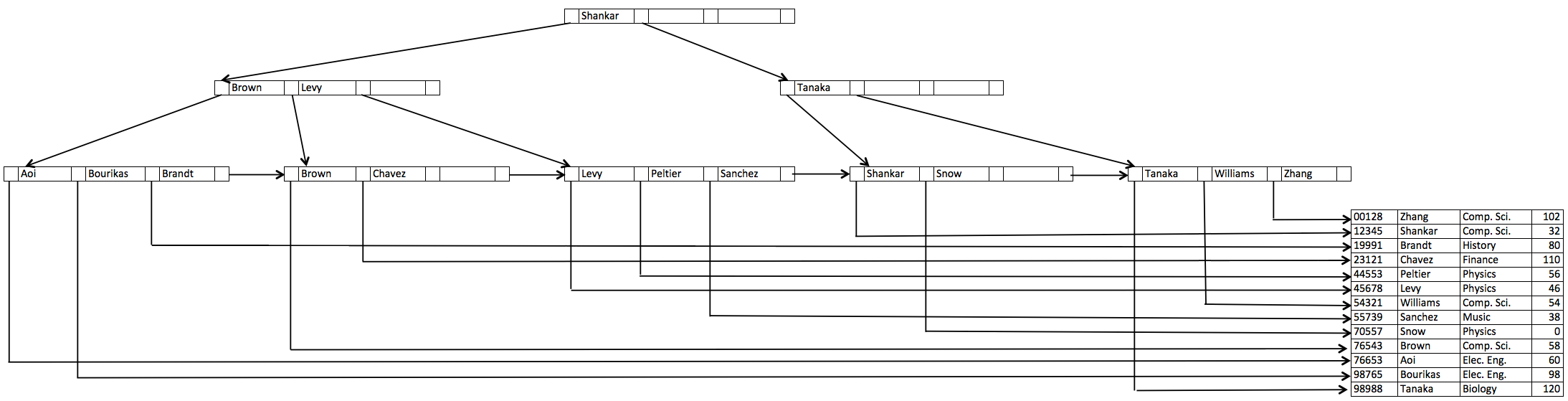
Hand calculated



Created by mySQL

In mySQL : CREATE UNIQUE INDEX NameIndex ON student(name) USING btree;

select name from student use index (NameIndex) order by name;



22.

SELECT DISTINCT student.name

FROM student, course, takes

WHERE student.ID = takes.ID

AND takes.course\_id = course.course\_id

AND course.dept\_name = ‘Comp.Sci.’;

23.

SELECT DISTINCT student.name, student.id

FROM student

WHERE student.id NOT IN

(SELECT student.ID

FROM student NATURAL JOIN takes AS joined

WHERE joined.year < 2009);

24.

SELECT instructor.dept, max(instructor.salary)

FROM instructor

GROUP BY instructor.dept\_name;

25.

SELECT min(maxsalary)

FROM (SELECT instructor.dept\_name, max(instructor.salary) as maxsalary

FROM instructor

GROUP BY instructor.dept\_name);