CIS250 Final Project

Overview and Grading Sheet

Introduction:
So far this semester you have been playing the role of a programmer or developer. Now it's time to switch roles and become a DBA (Database Administrator)! Over the next few weeks you will be working to design a database, create the database, and load the database with data. Once that is done you will switch roles and become a developer once again to run various statements against your database.

The database system you are designing is for Leroy's Used Record Store. Leroy needs a way to keep track of all of the albums he has in stock and all of the songs on those albums. This means you will need to store data about albums (cd's, too!), the groups that recorded the albums and the songs. His store is organized by rows of albums, and he would also like to know what row each album has been placed in …

Leroy would also like to keep a mailing list of potential customers. He would like to keep their name, address and e-mail address so that he can send them information about special events.

Initially you should be familiar with the album/song structure that is used by the recording industry and you'll use as the basis for designing/creating a logical data model, aka and entity relationship diagram (ERD). You will use this ERD to create the physical database. The information you have gathered by being a music user represents what you would normally gather in real life from business users as you interview them to find out what data they use to do their jobs and thus what data you'll need to store in the database. If questions come up (which more than likely they will), you will have to check with the user (me!).

Overview of Final Project Tasks:
The following table lists the main tasks involved in this project. <u>You may work individually or in pairs (no more than 2 people!)</u> If you work in pairs you will each will still be required to turn in your own complete packet of work.
Refer to 'CIS250 Final project Grading Sheet' for specific info on items to be turned in. All items will be turned in together during the last week of the semester.

Task
1. Design and create a logical data model (ERD) based on reports, forms and other information provided by business user. • Record info gathered from users or other research upon which you base your design (like notes taken during design phase as a result of discussion/questions asked of users…things like why you omitted certain data, business rules the user gave you, etc…)
2. Create the physical database based on your ERD • Using DDL (data definition language) • Record what you have learn as you create tables (for example, you have to create parent table before you can create related child tables)
3. Load Data into the Database • Record what you learn as you load data (for example, you may have used the wrong data type for a column and have to fix it before you can load data)

4. Run queries against your database
5. Turn in all required documentation

Getting started:
To get started on designing/creating a logical data model, refer to the PowerPoint sides for Chapter 10, the book, or the Chapter 10 exercises. Here is a quick refresher on the design process:

1. Identify Entities and Attributes
    - Identify primary keys for entities
    - Subdivide attributes into smallest useful data components
2. Identify Relationships (use a verb like 'submits /is submitted by'; 1:M, M:M, 1:1)
3. Normalize Data
    - 1st normal form: remove repeating attributes and place in their own entity
    - 2nd normal form: every non-key attribute must depend on the **whole** primary key; if not, it indicates another entity exists or remove such attributes to their own entity
    - 3rd normal form: every non-key attribute must depend **only** on the primary key; if not, it indicates attribute is assigned to wrong entity or can be derived.
4. Develop ERD to hand over to DBA (you!) to create physical design (data may or may not be denormalized when creating physical tables)

Remember – designing and creating a database is an iterative process so don't be surprised if you have to re-visit/tweak your design as you progress through the tasks listed in the table on the previous page. The goals of normalization are:

- To reduce data redundancy
- To accurately depict business entities, attributes and relationships between entities
- To minimize re-work down the road based on poor design (i.e. to do a thorough job of analysis and design up front to avoid having to re-do the design in the future)

Don't worry if you find bugs once you start running queries against your database and discover you can't answer a user's question based on the design of your database. Just learn from it – realize where/why the design is wrong and figure out how you could change your design to fix it. At least it's in the classroom and not in real life!

CIS250 Final Project Grading Sheet

Deliverable Description	*Max Points	Your Points
Data Model Turn in a copy of your data model • Data model should match tables created in your database Note: If you end up tweaking your database late on independently of your teammate, you will need to give me an updated data model that reflects YOUR database.	15	
Tables created in your database • Tables include appropriate primary keys, required constraints and default values (see list on next page), and unique indexes (10 pts) • Tables follow naming conventions as mentioned in class handout (5pts) ○ Do NOT use nulls unless you absolutely have to ○ Use mixed case names and capitalize the 1st letter of each ○ Use singular nouns for table names (Student vs Students) ○ All names must be 18 characters or less ○ Do NOT use spaces in table or attribute names ○ Use the same column name for primary keys and related foreign keys ○ Use meaningful/commonly use names and avoid abbreviations other than those listed below: ○ Code – cd ○ Flag – Flg ○ Date – Dt ○ Number – No ○ Time – Tm ○ Amount – Amt ○ Perent – Pct ○ Flag - Flg ○ Prefix index names with IX_ ○ Prefix check constraint names with CK_ ○ Prefix unique constraint names with UN_ ○ Contains should use meaningful names vs names assigned by SQL	15	

Tables loaded with data ○ Six releases (albums) with track (song) and band data ○ Two rows of releases ○ A mixture of Vinyl, CD and DVD ○ Four potential customers	15	
SQL queries run against team's DB to answer user's questions (SQL queries will be graded based on attempt to run queries and items below. It's ok if you are not able to answer all questions as long as you turn in notes as specified in #3 below) Items to turn in: 1. Printout of all of your SQL queries (whether they ran successfully or not) 2. For questions successfully answered via SQL queries, turn in a screen shot of the results 3. For each question that could not be answer by SQL queries, turn in notes indicating what is wrong with the DB design or data & your guess as to how you could fix it to be able to answer the question ("I don't know how to fix it" is not acceptable as a guess!)	15	
Misc Documentation ○ Turn in a copy of DDL statements used to create two tables chosen by instructor ○ Turn in documentation describing what you learned from your table creation and data entry experience. ○ Turn in documentation of all design decisions (like notes taken during design phase as a result of discussion/questions asked of users…things like why you omitted certain data, business rules the user gave you, etc…)	15	
Total	75	

* I reserve the right to modify max points at any time

Check Constraints You Must Implement: (use exact names as shown)

- CK_Media for media – Vinyl, CD, DVD are valid options
- CK_Email – valid format is *@*
- CK_StateAbbr – value format must be aa (2 alpha chars)

Other Constraints and Indexes You Must Implement:

- Album published date – use default value of Jan 1st 1901
- Foreign key constraints
- One index on album name and one index on customer e-mail