Project Part 2: Transforming an ERD to a database 100/100 points

Overview

In this part of the project, you will practice database design and apply principles of database design to transform your ClassDojo ER diagram into a relational schema and then implement the relational schema in Oracle. We will also write some of the queries necessary to fill in data for the Application described in part 1 of the project.

The Appendix shows a data model that you are to use as a starting point for this assignment. (NOTE: This data model has a few small differences to the solution in Part 1) If provide an ODM file for your convenience on Canvas. Download the file Spring2021Project2Start.zip, unzip it, and open the .dmd file in ODM.

Overview of Tasks and Grading

Your grade will be computed in the following way:

Item	***Points***
Create the detailed relational schema (Task 1)	35
Create and test the CREATE and DROP table statements (Task 2)	15
Populate the tables (Task 3)	10
Write queries (Task 4)	30
Reflection (Task 5)	10

Deliverables

- Task 1:
  - One word or pdf document
    - Justification of primary keys for all entities and justification of normalization or database design changes if any
  - One word or pdf document
    - Your relational schema from Oracle data modeler (that's the one once you forward engineer your logical design)
- Task 2 and 3: An Oracle implementation of the tables including the data you entered. You don't actually hand the tables in, I will find the tables in your account on the Oracle database.
- Task 4: One .sql file with queries
- Task 5: Reflection posted on Canvas

Task 1 – From E-R Diagram to Relational Schema

1. Step (1) Convert the provided data model (see Appendix and ODM file Spring2021Project2Start.zip on Canvas), based on project part 1, into a detailed logical data model.
2. Step (2) Make sure that all of your tables are fully normalized.
3. Step (3) Convert the logical data model into a relational schema.

For Step (1), you will do the following things:

- Choose a primary key for every table.
    - Choose the PK of strong entities first (i.e., super types, strong entities in identifying relationship) since the PK of subtypes and weak entities are based on the PK of the strong entities. If a natural key exists, you have to decide whether the attribute(s) make a suitable primary key for your table or if you want to use an artificial key.
        - Please don't be worried, that you cannot see the FK from the supertype in the subtypes. That is done once you engineer your logical to the relational schema.
    - Review required and desirable qualities of a primary key. Briefly justify your choice of a primary key in writing for every entity in the provided separate word document. If you need clarification on business rules that may determine what primary key to choose, ask me.
        - Please note that there may not be *one* correct choice for a primary key but several correct ones. That's why I ask for your rationale in choosing a primary key.
- Resolve m:n relationships by creating an appropriate bridge table
- Decompose composite attributes and solve multi-valued attributes (if any).
- For more information about how to do these steps in ODM, refer to **ODM tutorial part** 4 in week 7.

Step (2) is not something a tool can help you with. The business rules from Project Part 1 are still in effect. Whichever way you choose to handle normalization, your outcome will be (we hope) a set of well-defined tables. Please also be aware that many (or all) tables may be already in BCNF. If you choose to have tables in a lower form, please provide a written justification in the provided word document. If you chose to decompose or merge tables or update tables in any other way due to other design rules, provide a justification as well.

Step (3) involves defining data types and possibly domains for your attributes (providing a domain name, a data type, and possibly additional constraints such as unique or check constraints). Please refer to **part 5 and 6 of the Oracle data modeler tutorial**. As minimum constraints, include the three domains with check constraints for attributes. The easiest way is to do this via domain constraints in ODM. If you run into troubles defining domains, then you could also choose to write the constraints directly into the ddl code that ODM produces in task 2:

- Student account status (value list)
- Feedback given to (value list)
- Weight (range)

When you choose the data type, the logical data types NUMERIC transforms to NUMBER and VARCHAR to VARCHAR2. Do not choose data type Integer as this is deprecated, use NUMBER instead. In general, use the built-in data types. Review this link for more data types: https://docs.oracle.com/en/database/oracle/oracle-

database/12.2/sqlrf/Data-Types.html. You may assume that data is specific to the US, e.g., phone numbers or addresses. Don't overcomplicate.

Make sure that your database objects names comply with the Oracle naming convention (e.g., see https://docs.oracle.com/en/database/oracle/oracle-database/12.2/sqlrf/Database-Object-Names-and-Qualifiers.html#GUID-75337742-67FD-4EC0-985F-741C93D918DA).

- Before Release 12.2 table and column names could only be 30 bytes, names up to 128 bytes are now allowed. ODM may still throw an error that you should be able to simply ignore. If you want to be on the safe side, shorten long column names.
- For your table and attribute names, you should avoid reserved words and SQL keywords, (e.g., order is a reserved word and should not be used as table name, see https://docs.oracle.com/en/database/oracle/oracle-database/12.2/sqlrf/Oracle-SQL-Reserved-Words.html#GUID-55C49D1E-BE08-4C50-A9DD-8593EB925612). The list of reserved and SQL keywords, can be obtained by querying a table called V$RESERVED_WORDS. It seems student accounts cannot query this table, so I share with you that the names “class”, “comment”, and “parent" are keywords. You need to change the entity name for them. “Feedback” is not a keyword I could find, but SQL developer will color code it blue, indicating that this may be a problematic table name as well. I changed it as well. There are several ways to change entity names that could be problematic. I choose to change the entity names in a way by adding a qualifier, e.g., “story comment” instead of comment.

As a final step, forward engineer the logical model into a relational schema in Oracle data modeler. Refer to **part 7 in the Oracle data modeler tutorial**. When you forward engineer, the tutorial mentions to leave "Apply name translations" checked. In some of my model, this leads to long foreign keys (that include the table name of the originating table in the foreign key name). If this happens to you, uncheck this field as well. In order to quickly make the relational schema looking nice, select all tables and right-click on one of them and select ‘resize to visible’. Then right-click on the canvas and select Layout -> Auto Layout -> Layout 1.

Once you have the relational model, ensure that all Primary Keys are set correctly and the data types look appropriate. Unfortunately, I have found that if I set the scale to 0 to make a NUMBER an integer, this scale does not correctly reflect in the relational schema as well as ddl. If that happens to you, you can ignore this bug.

Use File -> Print Diagram -> To PDF to print the relational model to a pdf file. Arrange the tables that all column names and their data types are visible. This pdf file with your relational model and your text document with your PK justification are your deliverable for task 1.

Project Part 2: Transforming an ERD to a database 100/100 points

Task 2 – Create the database

Use forward engineering in Oracle data modeler to create the tables of your relational schema. Refer to **part 8 of the Oracle data modeler tutorial**. Make sure you have all the correct project settings as described in ODM tutorial part 1 in Week 5.

- As instructed, generate the SQL DDL statements, and save the file to a ddl or sql file. The current database is Oracle Database 12cR2. If you get an error revise your relational schema and generate the statements again. You may get an error message for names > 30 bytes, that you can ignore as 12c Release 2 allows table names length > 30 bytes.
- Open SQL developer, connect to the PKIIST database, open your newly generated ddl file in SQL developer, and run and test the SQL DDL statements. If you get error, revise your SQL statements until all tables are created.
- Your implementation needs to be consistent with your relational schema you submit for task 1 and include all referential, entity, and domain constraints as specified in the relational schema.
- Please note that the first time you run your DROP statements, you will get an error message simply because you have not yet created the tables. This error is fine. Subsequent runs should not generate this error anymore.

The tables on your PKIIST Oracle account are your deliverables of task 2. You don't actually hand them in, I will find the tables in the Oracle database.

Task 3 - Populate the database

Write INSERT statements to add data to your tables. Since there are many tables in this assignment, I decided to give you my INSERT statements for most of them as a starting point, see *Spring2021InsertStatements.sql* on Canvas. You may have to make changes if you chose a different primary key or made other design choices than I did. Adjust the INSERT statement correspondingly. I hope providing the INSERT statement will help you to fill data into your tables much faster. Inserting test data is important to test your design and queries, but can be tedious. It is part of the process.

The data you find in the INSERT statements is the minimum that I want to find in your tables from this task. You may add more data if you want.

Important: Make sure to COMMIT all data after you insert them!!!

The data in your tables are the deliverables of task 3. Again, you don't hand anything in, I will find the tables in Oracle.

Task 4 – Write additional queries

After all the set-up is done, the real fun starts. I want you to write several queries that you would need for the ClassDojo Application. Most queries will feed directly into the user interface you saw in the project part 1 ClassDojo business rules.pdf file.

Download and open the file *Spring2021Queries.sql* file. In comments, add your name at the top of the file. You will find all required queries explained in comments. Add you queries in the correct space. Test and debug them before submitting.

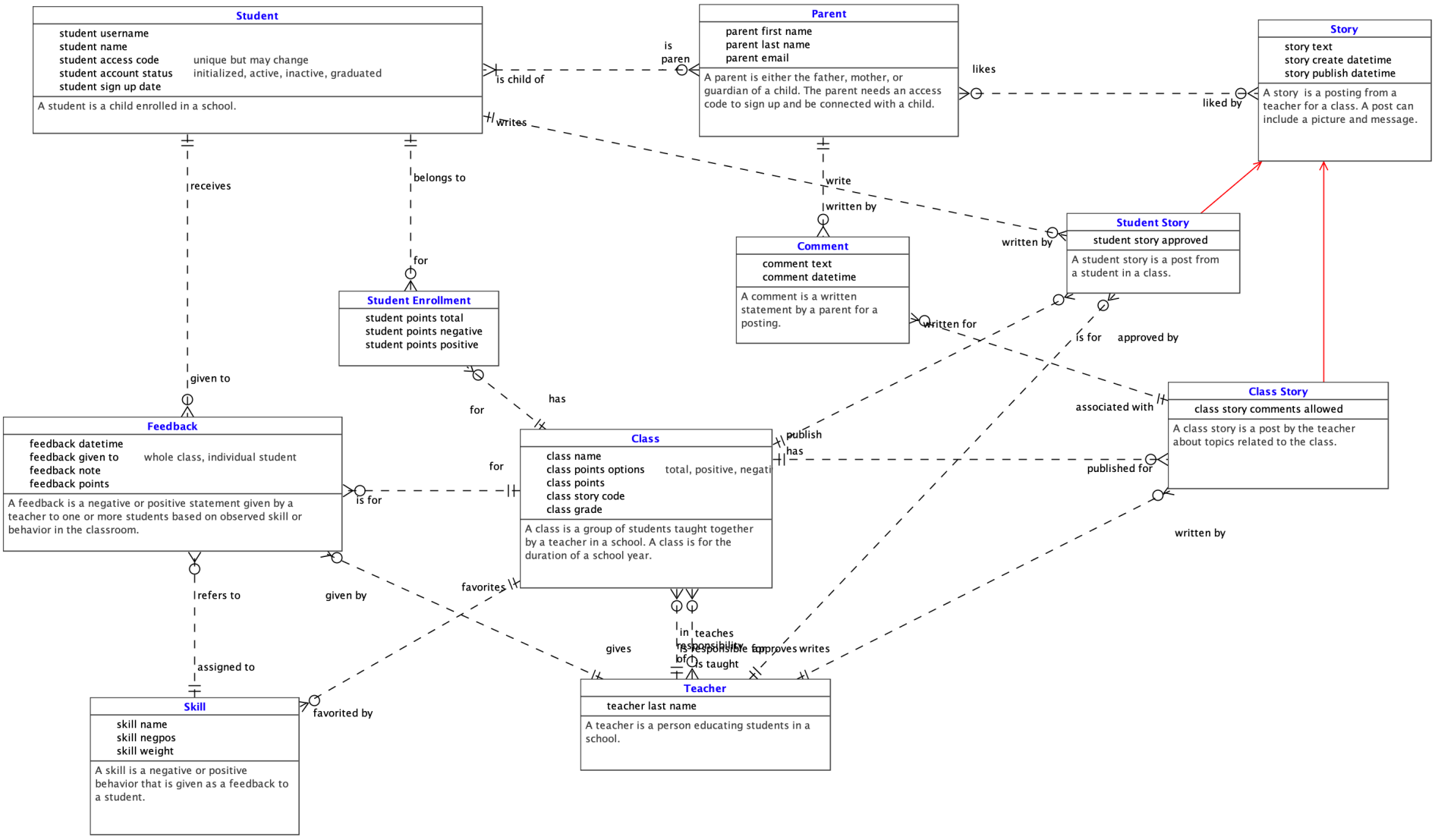
You are done for now! We will visit our Class Dojo case again in project part 3 when designing one or more star schema!

Submit your *Spring2021Queries*.sql on Canvas.

Task 5: Reflection

See Canvas.

Appendix 1


Student
student username
student name
student access code
student account status
student sign up date
unique but may change
initialized, active, inactive, graduated
A student is a child enrolled in a school.
Parent
parent first name
parent last name
parent email
A parent is either the father, mother, or guardian of a child. The parent needs an access code to sign up and be connected with a child.
Story
story text
story create datetime
story publish datetime
A story is a posting from a teacher for a class. A post can include a picture and message.
is child of
is paren
likes
liked by
writes
receives
belongs to
write
written by
written by
Student Story
student story approved
A student story is a post from a student in a class.
Comment
comment text
comment datetime
A comment is a written statement by a parent for a posting.
for
Student Enrollment
student points total
student points negative
student points positive
written for
is for
approved by
given to
has
for
Class Story
class story comments allowed
A class story is a post by the teacher about topics related to the class.
associated with
Feedback
feedback datetime
feedback given to
feedback note
feedback points
whole class, individual student
A feedback is a negative or positive statement given by a teacher to one or more students based on observed skill or behavior in the classroom.
Class
class name
class points options
class points
class story code
class grade
total, positive, negati
A class is a group of students taught together by a teacher in a school. A class is for the duration of a school year.
publish
has
for
published for
is for
written by
refers to
given by
favorites
in
teaches
responsibility
of
is taught
gives
approves
writes
assigned to
Skill
skill name
skill negpos
skill weight
A skill is a negative or positive behavior that is given as a feedback to a student.
favorited by
Teacher
teacher last name
A teacher is a person educating students in a school.