CS6400 DATABASE PROJECT: PEACHTREE SAVINGS CLUB

Phase 1

	Deliverable	Textbook chapter readings	Lectures
Phase 1	Analysis & Specification: • IFD (10%) • EER Diagram (40%) • Data formatting (5%) (attributes, domains) • Constraints (5%) • Task Decomposition (10%) • Abstract code (30%)	1, 2, 3, 4	Methodology I: Analysis Methodology II: Specification

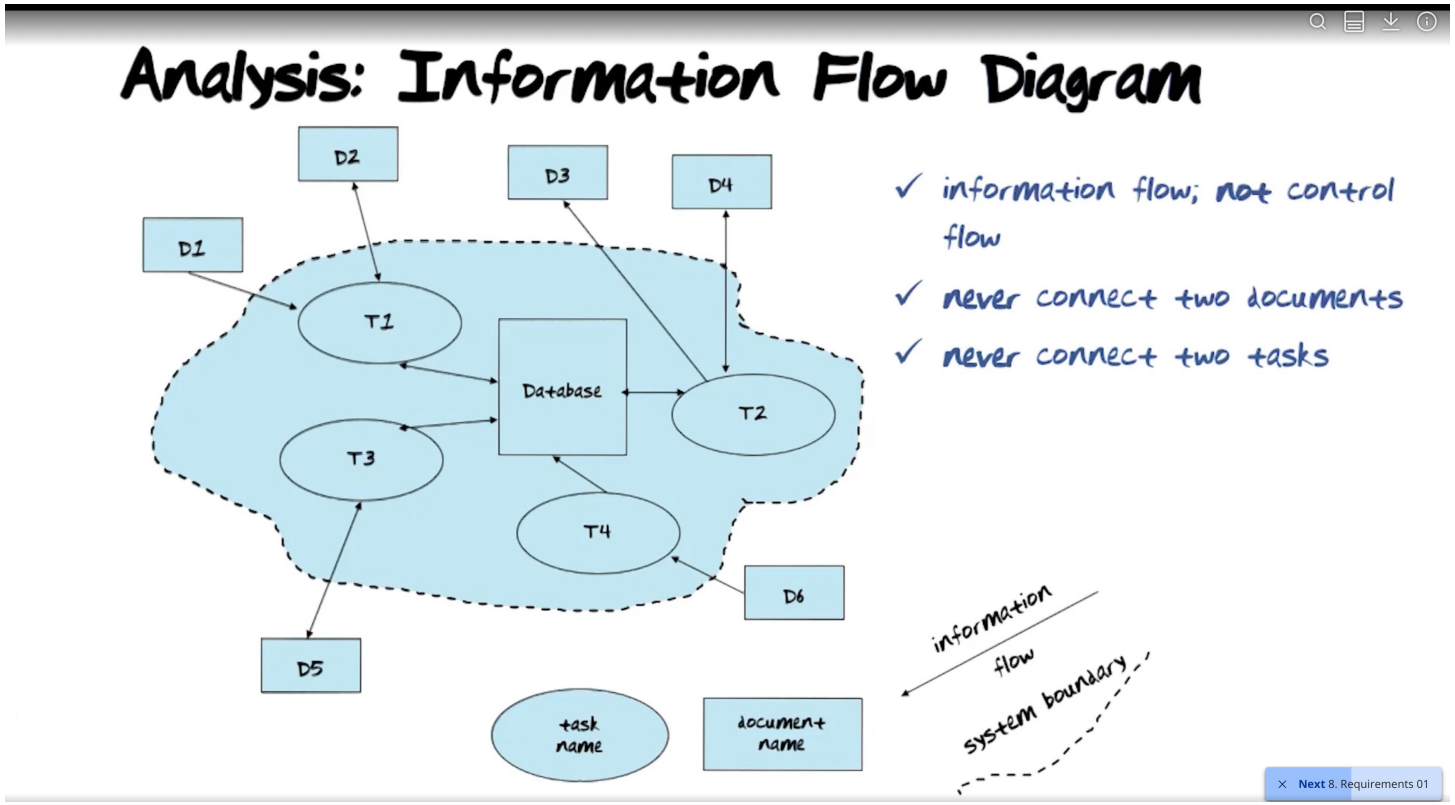
Sample Documents:

- Phase 1 Template
- EER Diagram
- IFD
- Project Spec Doc

Supported Tools:

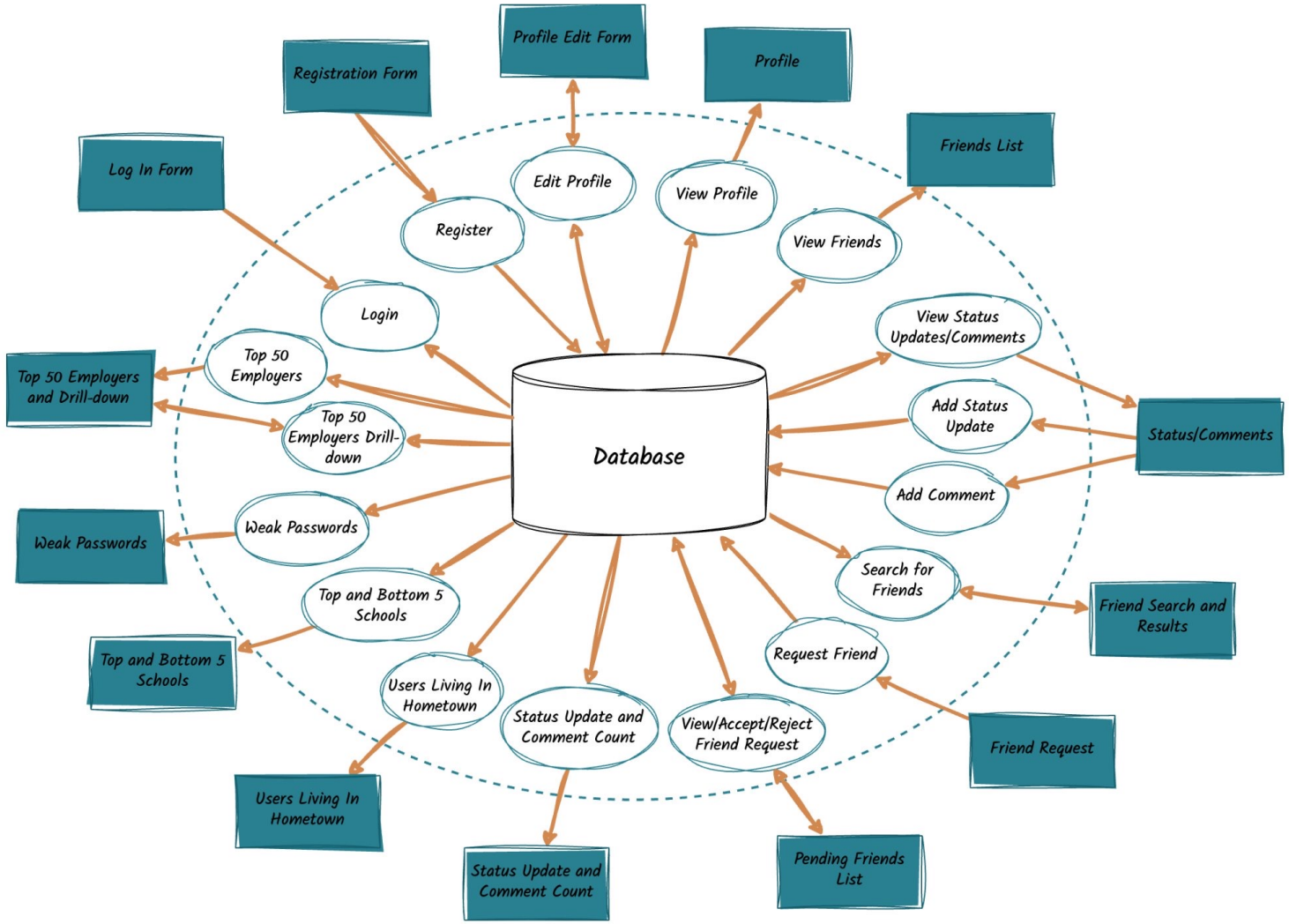
- Lucid Chart

IFD (10%)


Analysis: Information Flow Diagram
D2
D3
D4
D1
T1
Database
T2
T3
T4
D6
D5
task name
document name
information flow
system boundary
✓ information flow; not control flow
✓ never connect two documents
✓ never connect two tasks
Next 8. Requirements 01

Phase 1 IFD | CS 6400 – Fall 2023 | **Team 001**

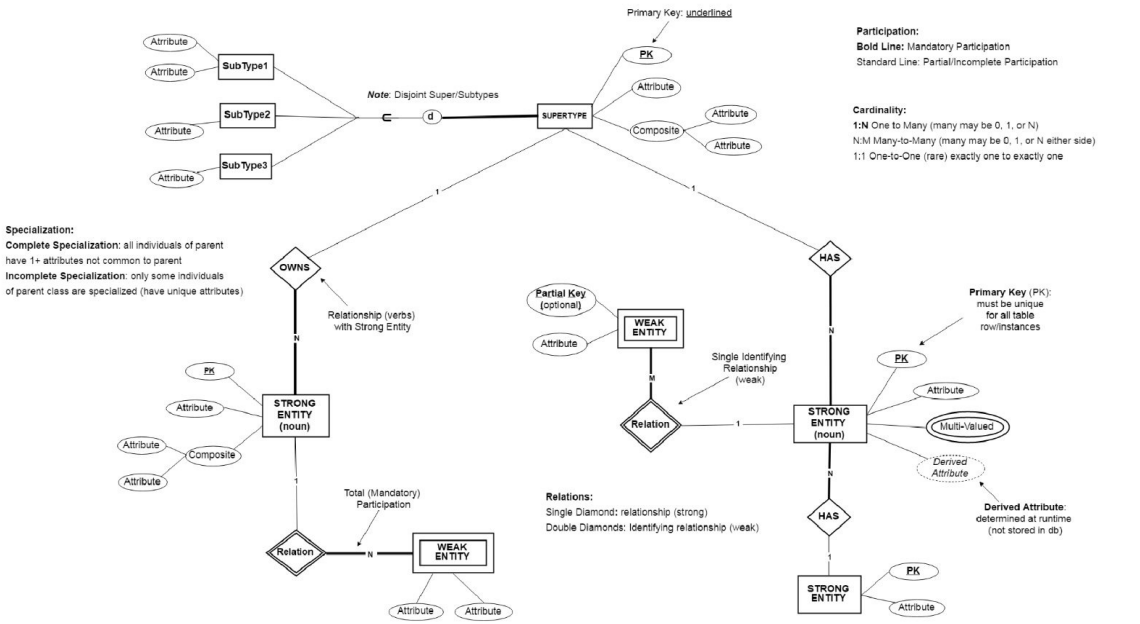
Single Page Image: format **to fit one-page *landscape***: include forms, directed arrows, dedicated tasks, DB, and system boundary dashed line. All tasks present on the IFD should be on the report in the task decmposition section. Consider the 'forms' to be like webpage UI data entry forms. See notes on wether to use dedicated tasks or combine into a "mother' task. Follow the naming convention for forms/tasks provided in the requirements document.



EER DIAGRAM (40%)

Phase 1 EER I CS 6400 - Fall 2017 I **Team 001**

Enhanced Entity-Relationship (EER) Diagram: (format example: crop/fit to one page)



Single Image:format to **fit one-page *landscape***:clearly label attributes, cardinality (1:N), strong vs. weak entities, total(bold) vs. partial participation links, underline keys. Distinguish between double ovals/rectangles/diamonds for attributes, entities, and relations respectively.
Lastly, condense attributes around entities, make sure lines do not cross over other attributes (may unintentionally look like composite attribute).

ENTITY type (**nouns**) and **relationship** type (**verbs**) names are UPPERCASE letters,
Attributes with Uppercase first letter
Primary Key (PK) underline, Parital Key (weak) dashed underline

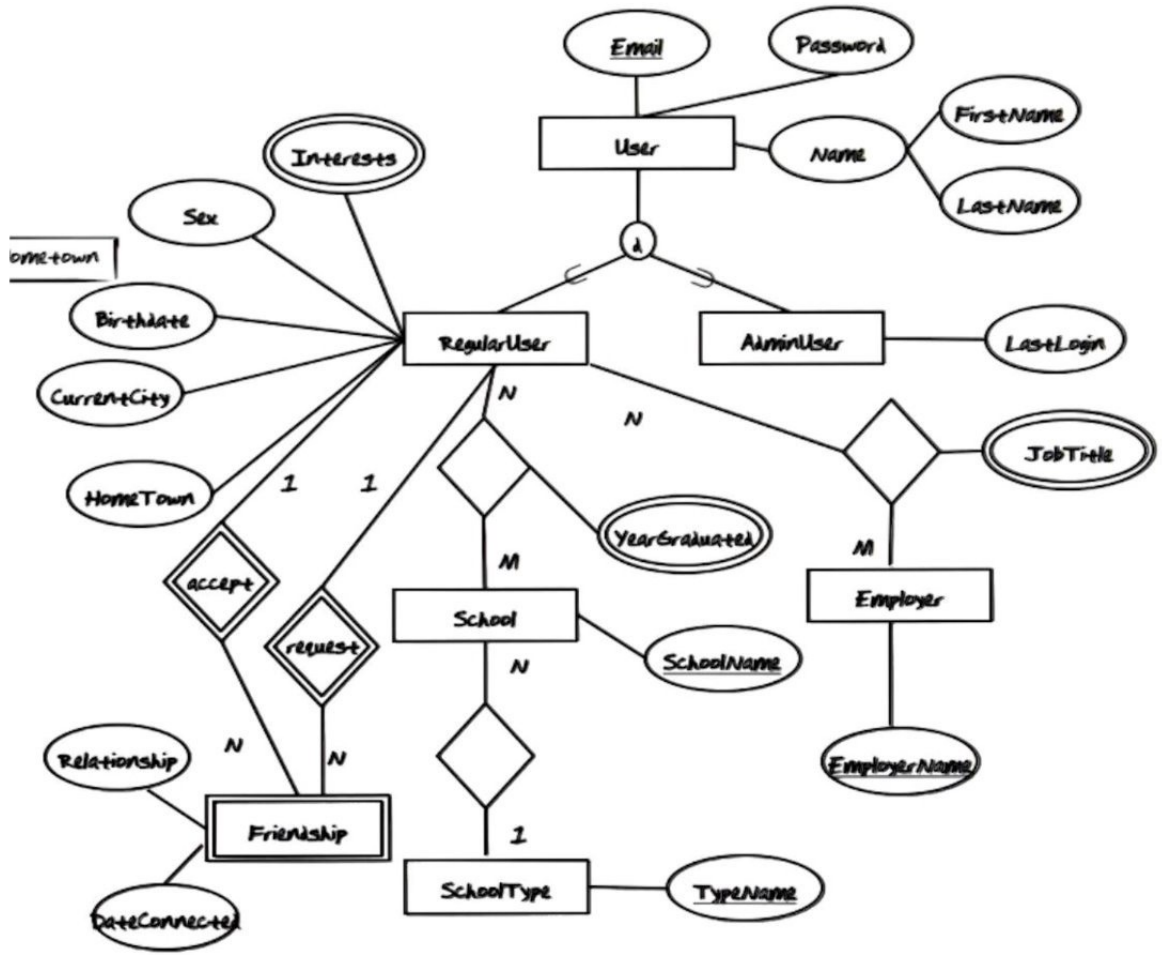
Total/Complete specialization (existence dependency) demands that every entity in the superclass belong to some subclass- represented with a bold line link/connection
Partial/Incomplete specialization(optional all or none) rule allows an entity to not belong to any of subclasses- represented with a **single line** link/connection

ⓓ **Disjoint**:(exclusive either/or but not both) an individual of the parent class may be a member of ***exactly one*** specialized subclass.
ⓞ **Overlapping**: (both are possible) individual of parent class may be a member of ***more than one*** of specialized subclasses.
∪ **Union**: joining of two super types/classes

Revised: 6/27/2017

Phase 1 EER I CS 6400 - Fall 2017 I **Team 001**

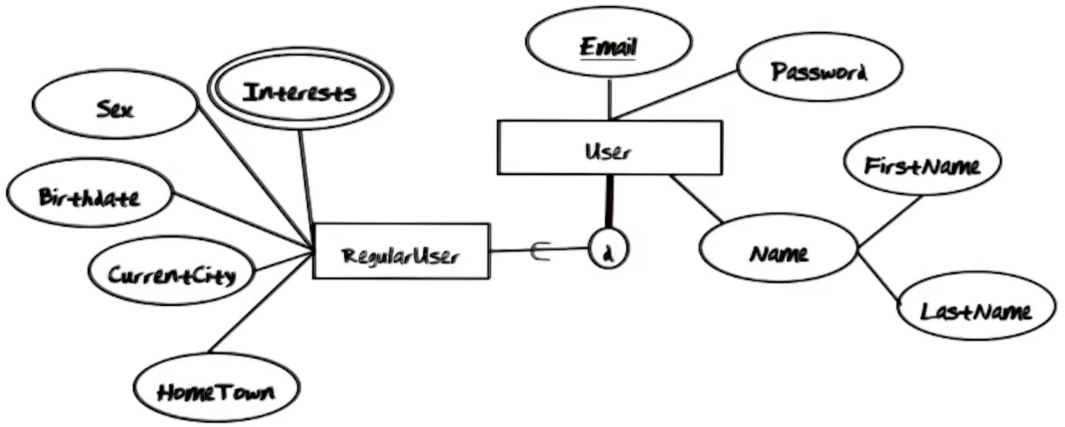
EER Submission details:
Single Image: format to **fit one-page *landscape***, clearly label attributes, cardinality (1:N), strong vs. weak entities, total (bold) vs. partial participation links, underline keys. Distinguish between double ovals/rectangles/diamonds for attributes, entities, and relations respectively. Lastly, condense attributes around entities, make sure lines do not cross over other attributes (may unintentionally look like compound attribute).
Do NOT include Surrogate Keys for Phase 1, these will be added for Phase 2 submission.



Revised: 6/27/2017

DATA FORMATS AND DOMAIN (5%)

Data Formats- beg, steal, borrow



User:

- Email: max 36 chars. Example: leomark@cc.gt.edu
- Password: max 20 chars. Example: qwerty
- Name: FirstName: max 25 chars; Last: max 40 chars
- Addresses (when needed) are very very difficult.
- E.g., look at the 208 page guideline at http://pe.usps.com/cpim/ftp/pubs/pub28/pub28.pdf

RegularUser:

- Birthdate: Date: 'YYYY-MM-DD'
- Sex: {M, F}
- CurrentCity, HomeTown: max 20 chars, each
- Interests: multi-value with 16 chars, each

DATA FORMATS AND DOMAIN (5%)

Data Types:

User

Attribute	Data type	Nullable
username	String	Not Null
email	String	Not Null
password	String	Not Null
last_name	String	Not Null
middle_name	String	Not Null
first_name	String	Not Null
interests	List<String>	Null
…	…	…

Friendship

Attribute	Data type	Nullable
relationship	String	Not Null
date_connected	Date	Null
…	…	…

Do not include surrogate keys or foreign keys in phase 1 (e.g. ‘friend_email’)

CONSTRAINTS (5%)

Constraints

Examples:

- DateConnected is NULL until request is accepted.
- Cannot be Friend with yourself.
- Can only comment on Status of Friends.

Already covered:

- Data formatting constraints
- Constraints that can be expressed in the EER Diagram

Relational Model - Constraints

Constraints express rules that cannot be expressed by the data structures alone:

- Email's must be unique
- Email's are not allowed to be NULL
- BirthDate must be after 1900-01-01
- Hometown must be cities in the US

RegularUser

Email varchar(50)	BirthDate datetime	Hometown Varchar(50)	Salary integer
user1@gt.edu	1985-11-07	Atlanta	10,000
user2@gt.edu	1969-11-28	Austin	11,000
user3@gt.edu	1967-11-03	Portland	12,000
user4@gt.edu	1988-11-15	Atlanta	13,000
user5@gt.edu	1973-03-12	Portland	14,000
user6@gt.edu	1988-11-09	Atlanta	15,000

Business Logic Constraints:

GTOnline User

- Users who are new to GTOnline must register first.
- Users who have an existing GTOnline account will not be able to register.
- Both users must send friend requests to each other and both requests must be accepted.
- …

BUSINESS LOGIC CONSTRAINTS

Instructors' Answer

Updated 45 minutes ago by Alex Yanovsky

Business logic constraints should be statements on the form of constraints which can be found in the specs and which cannot be expressed in EER

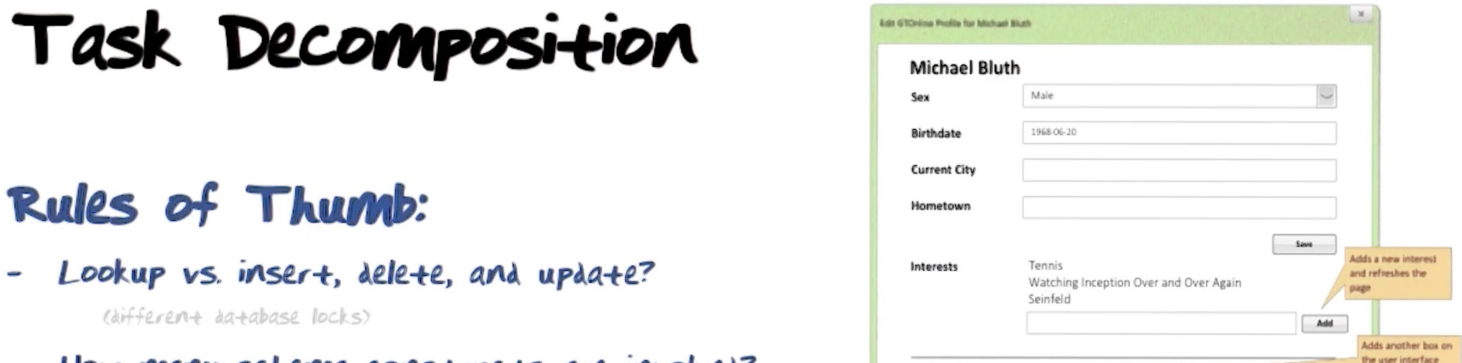
there should be a limited number of these statements , not more than 10, usually even less. You should not put there anything about format , and no trivial points , like price should be positive etc.

Piazza NOTE @247

TASK DECOMPOSITION (10%) & ABSTRACT CODE (30%)

Task Decomposition:
Include the rules of thumb outlined in the lectures for each task and determine if a mother task is needed (lock types, enabling conditions, frequency, schemas, indices, consistency, and subtasks). Task names on the IFD should match those in the TD. Please follow the report names provided in the requirements document, so TAs can easily follow your logic.

Per Leo Mark: "*The rules of thumb for task decomposition give you an indication of whether to sub-divide or not. You stop the decomposition when no more decomposition is called for by the rules. You need a mother task if the sub-tasks need to be sequenced, all executed together as an atomic unit.*"
Per Jay Summet: "*If your task has no sub-tasks you would only have the single oval in the IFD and the abstract code in the report. If your tasks do have subtasks, then you show how it breaks down and show the abstract code for each subtask in the report.*"



SUBTASKS

Instructors' Answer

Updated 1 day ago by Alex Yanovsky

Task is a set of operations performed by our application. The data flow to or from the task can touch attributes of some entities or relationships (mind that relationship is not the same as relation in this field , so we should mind which words we are using)

but of task should or should not be decomposed is not connected to how many entities or relationships are involved in the data flows which we show in ifd.

as task is set of operations , the subtask

sill involve subset of such operations.

in some cases it makes sense to distinguish some subsets of operations into subtasks.

please watch the methodology lecture which talks about task decomposition.

ifd is separate disgram very different from EER which is about entities and relationships

Piazza NOTE @230

LOCK TYPES, NUMBER OF LOCKS, ENABLING CONDITIONS, FREQUENCY, CONSISTENCY (ACID) AND SUBTASKS

Instructors' Answer

Updated 1 day ago by Alex Yanovsky

The only source for those are lectures on methodology where Leo explains task decomposition

This is about the locks:

There can be read locks and write locks.
you should look at your EER and at your ifd and fir each task in ifd you should determine how many entities and many to many relationships and multivalued attributes it needs to read to accomplish given task. That would be number of read locks.

the similar number (replacing reading by writing in the explanation above) is the number of write locks.
so in the lock section for in thr task decomposition part you want to write number of read locks, number of write locks and enumerate all objects in your EER which you counted for reading and all objects which you counted for writing. In this way you will show that you had some considerations of how your task will address its function when relying on the data stored according to your EER.

ACID is critical or not critical. It will be critical for the task which can potentially make such changes which if we assume two users are doing simultaneously may affect adversely the business of our company.

Piazza NOTE @184

COMPLEX QUERY IN ABSTRACT CODE

For complex query, it means some reports need (aggregate, left join, case when) to get result ”column”, example of Revenue.

Instructors' Answer

Updated 21 minutes ago by Alex Yanovsky

Yes, that is a good question

mind this is our first phase , so if you at least enumerate correctly what this query is supposed to do and which objects of our EER will be involved that would be ok.

and please do not use the words timetables, columns etc.

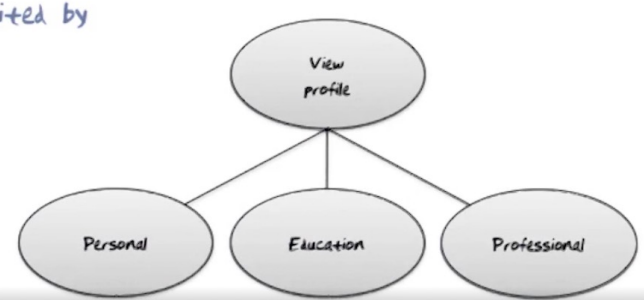
Piazza NOTE @270

EXAMPLE 1: VIEW PROFILE

Task Decomposition ViewProfile

View Profile:

- three lookups of Personal, Education, and Professional information for a RegularUser
- All three are read-only.
- All three are enabled by a user's login or a friend's lookup.
- All three have the same frequency.
- Several different schema constructs are needed.
- Consistency is not critical, even if the profile is being edited by the user while a friend is looking at it.
- They can be done in any order.
- All three must be done, so mother task is needed.
- **Should be decomposed into three sub-tasks.**

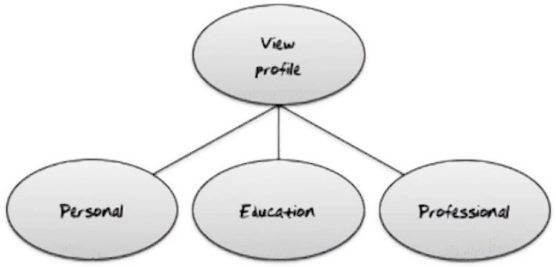


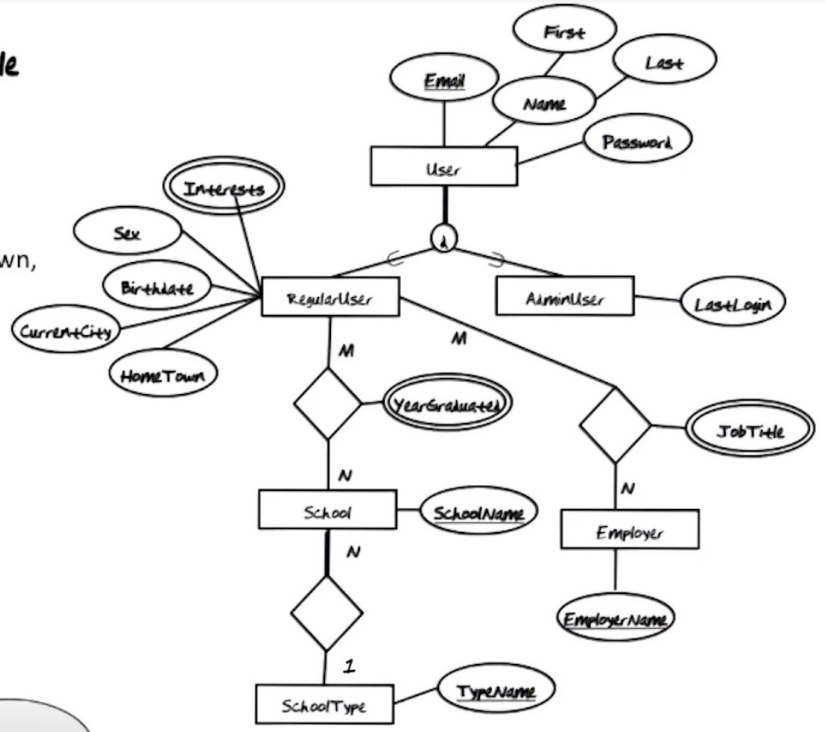
EXAMPLE 1: VIEW PROFILE

Abstract Code ViewProfile

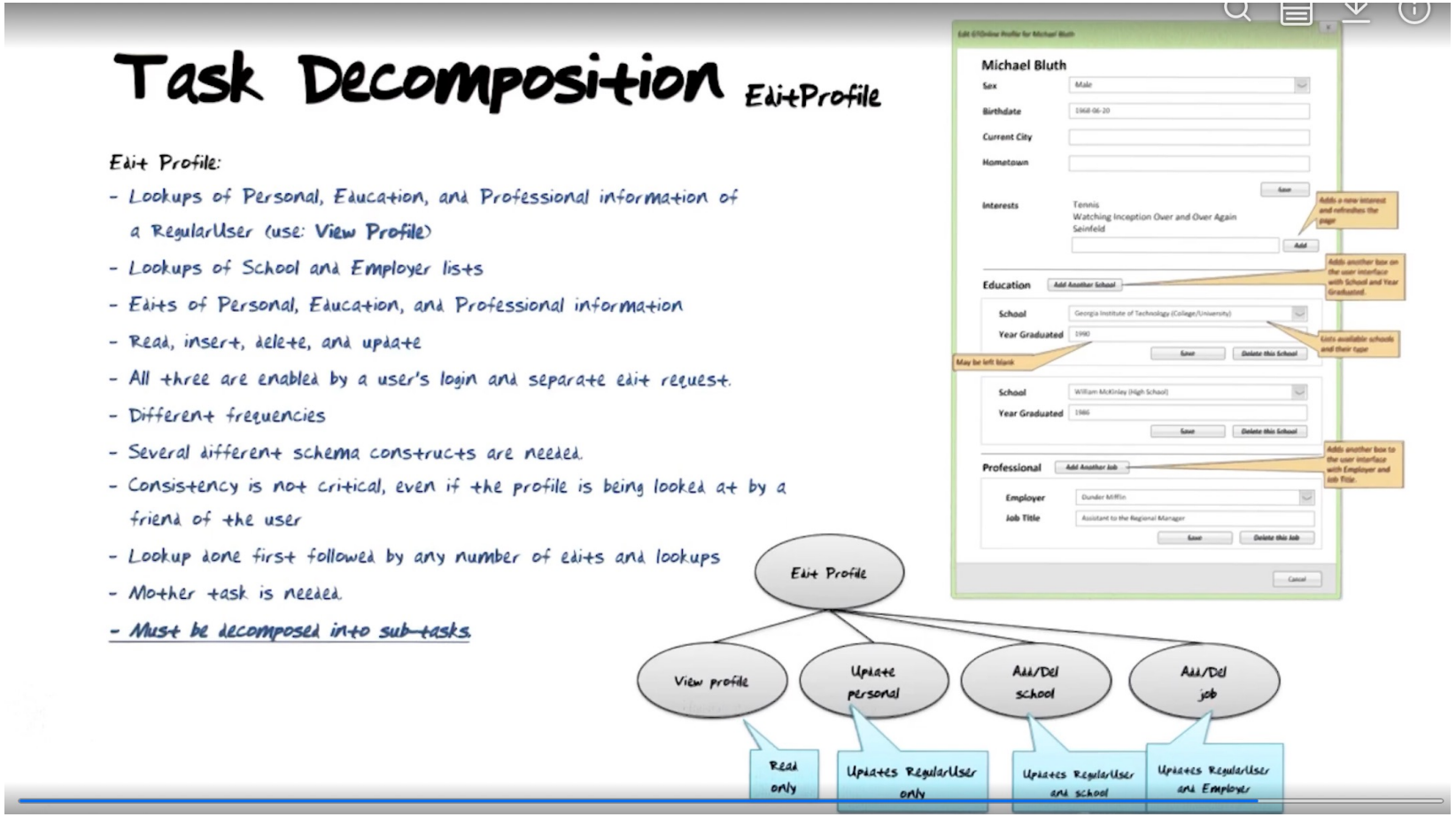
View Profile

Find the current User using the User Email;
Display User Name;
Find the current RegularUser using the User Email;
Display RegularUser Sex, Birthdate, CurrentCity, Hometown, and Interests;
Find each School for the RegularUser
    {Display School Name and YearsGraduated;
    Find SchoolType;
    Display SchoolType Name};
Find each Employer for the RegularUser
    Display Employer Name and JobTitles;





EXAMPLE 2: EDIT PROFILE



EXAMPLE 2: EDIT PROFILE

Abstract Code

Edit Profile

Edit Profile
View Profile. Populate School and Employer dropdowns.
While no buttons are pushed, do nothing.
When a button is pushed, then do the following:

{If *SAVE PERSONAL:* **Update Personal. View Profile.**

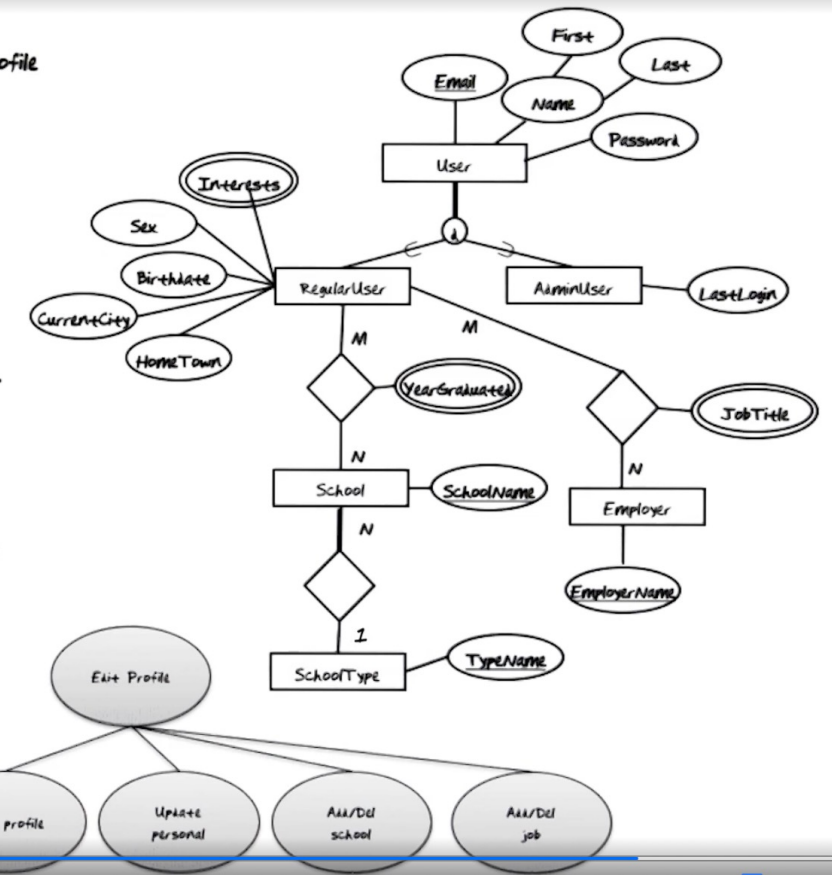
If *DELETE THIS SCHOOL*: **Delete School. View Profile.**
If *DELETE THIS JOB*: **Delete Job. View Profile.**

If *ADD ANOTHER SCHOOL*: **View Profile;** display the form with room for another School;
If *ADD ANOTHER JOB*: **View Profile;** display the form with room for another Job

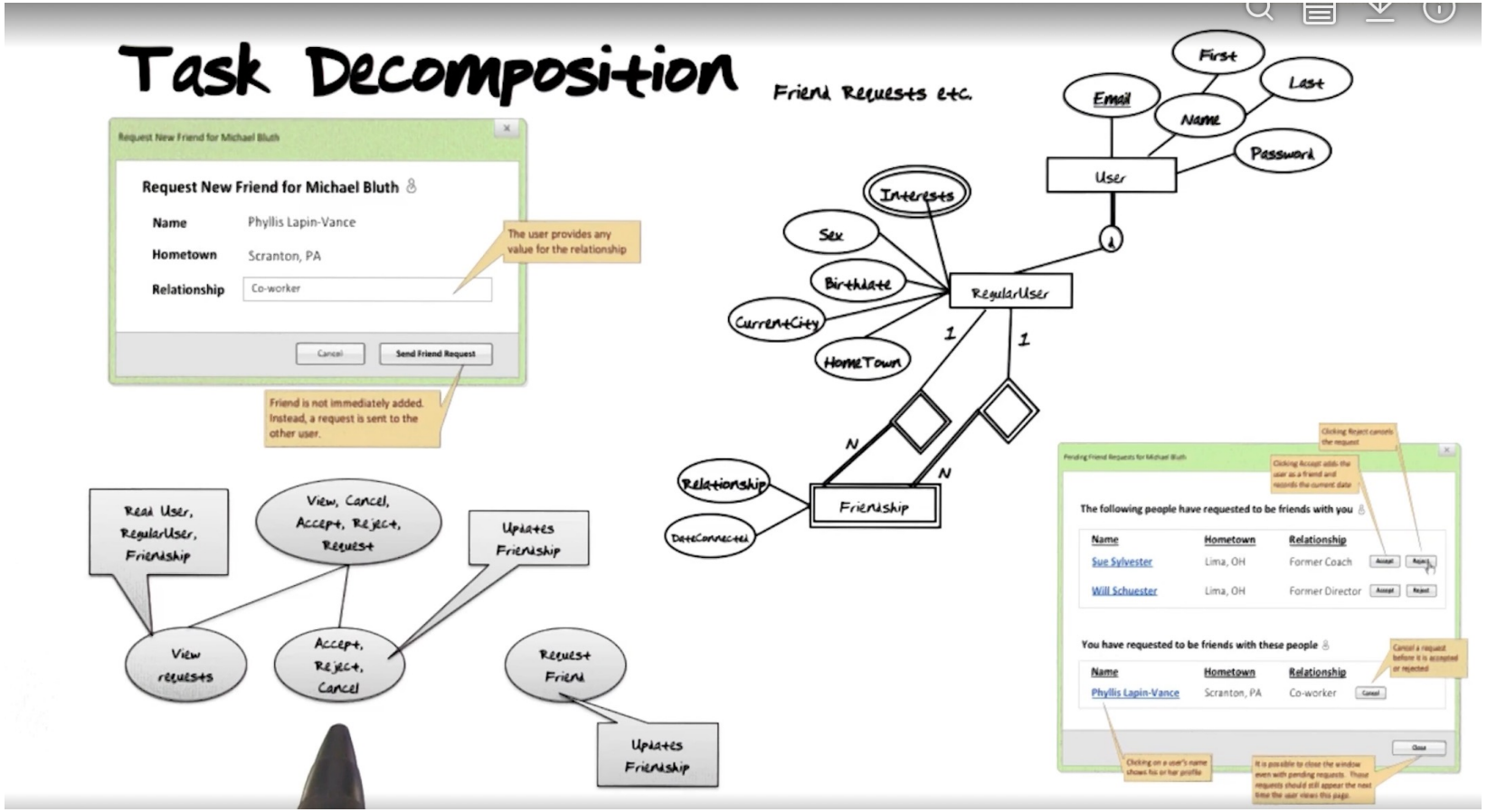
If *SAVE SCHOOL*: **Add School. View Profile**
If *SAVE JOB*: **Add Job. View Profile**

If *CANCEL*: Go to **View Profile** for current User};

BoldUnderline: Task definition. **Bold**: Task call. *Italics*: Button names



EXAMPLE 3: FRIEND REQUESTS



Login

Task Decomp

Lock Types: Read-only on RegularUser table
Number of Locks: Single
Enabling Conditions: None
Frequency: Around 200 logins per day
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

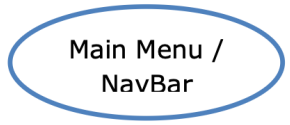
Abstract Code

- User enters *email* ('$Email'), *password* ('$Password') input fields.
- If data validation is successful for both *username* and *password* input fields, then:
  - When ***Enter*** button is clicked:
    - If User record is found but user.password != '$Password':
      - Go back to **Login** form, with error message.
    - Else:
      - Store login information as session variable '$UserID'.
      - Go to **View Profile** form.
- Else *email* and *password* input fields are invalid, display **Login** form, with error message.

Main Menu / Navigation Bar

Task Decomp

Lock Types: Lookup User Name and City, all are Read-only.
Number of Locks: Single
Enabling Conditions: Trigger by successful login.
Frequency: User Detail and Menu Options have the same frequency.
Consistency (ACID): not critical, order is not critical.
Subtasks: Mother Task is not needed. No decomposition needed.

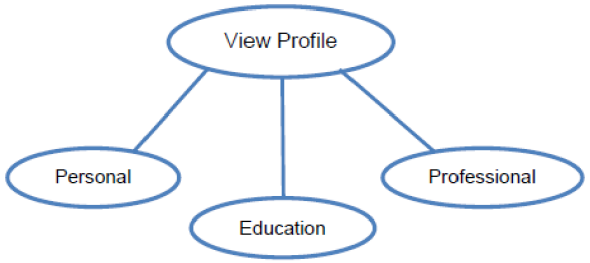


Abstract Code

- Show "***View Profile***", "***Edit Profile***", ***"View Friends"*** ", ***"Search for Friends"*** ", ***"View Requests"*** ", ***"View Status Updates***", and "***Log out***" tabs.
- Upon:
  - Click ***View Profile*** button- Jump to the **View Profile** task.
  - Click ***Edit Profile*** button- Jump to the **Edit Profile** task.
  - Click ***View Friends*** button- Jump to the **View Friends** task.
  - Click ***Search for Friends*** button- Jump to the **Search for Friends** task.
  - Click ***View Requests*** button- Jump to the **View Requests** task.
  - Click ***View Status Updates*** button- Jump to the **View *Status Updates*** task.
  - Click ***Log Out*** button- Invalidate login session and go back to the **Login** form.

View Profile

Task Decomp



Lock Types: 3 read-only lookups of Personal, Education, and Professional information for a RegularUser
Number of Locks: Several different schema constructs are needed
Enabling Conditions: All 3 are enabled by a user's login or a friend's lookup
Frequency: Low- All 3 have the same frequency
Consistency (ACID): is not critical, even if the profile is being edited by the user while a friend is looking at it.
Subtasks: All tasks must be done, but can be done in parallel. Mother task is required to coordinate subtasks. Order is not necessary.

Abstract Code

- User clicked on ***View Profile*** button from **Main Menu**:
- Run the **View Profile** task: query for information about the user and their profile where $UserID is the ID of the current user using the system from the HTTP Session/Cookie.
  - Find the current User using the User.email; Display users first and last name;
  - Find the current RegularUser using the User Email; Display RegularUser Sex, Birthdate, CurrentCity, Hometown.
  - Find and display the current user interests;
  - Find each School for the RegularUser:
    Display School name and Years Graduated;
    Find School Type;
    Display SchoolType Name;
  - For each Employer for the RegularUser:
    Display Employer Name and Job Titles;

  When ready, user selects next action from choices in **Main Menu**

…etc.

PHASE 1 TASK ASSIGNMENT

Username	Task
slu376	~~IFD/EER Diagram/Data Formats/Constraints~~/Task decomposition & Abstract Codes
dli698	IFD/ EER Diagram/Task decomposition& Abstract Codes
ychen3903	EER Diagram/Constraints
dcai63	IFD/EER Diagram

FRONT END & BACK END TOOKIT

- Front-end: React + CSS + AI?
- Back-end: Python + Flask
- Database: choose one from ~~PostgreSQL /~~ MySQL
- QA Testing: Postman
- Environment: Docker

FRONT END & BACK END ASSIGNMENT

Part	Username
Front-end	Slu376, ychen3903
Back-end	Slu376,dcai63
SQL Query	Dli698,dcai63, ychen3903
QA Testing	—

ASSUMPTIONS

- Unless otherwise specified as optional, all attributes are required!
- If given a list of potential values, choices should be limited to that list.
- If a set of values is listed with "and/or", combinations of those values are possible, while “or” indicates only a single value is possible.
- You should create normalized schemas.
- Minimize the use of NULL attributes whenever and wherever possible.
- Ensure that you store non-numeric data that appear as numbers (such as street numbers, phone numbers, postal codes, etc.) as strings and not numeric data type.
- Avoid “catch-all” forms with unnecessary inputs that the user would leave empty or NULL.
- Not need to be concerned about handling concurrent operations that could conflict and introduce inconsistencies in your database, so transactions and locking will not be required.
- Any actions listed as “performed by the database administrator” will not be performed in our application, but rather manually, via SQL prompt or similar interface.
- Do not create any extra functionality that is not mentioned in this specification (such as email notifications, etc.) or attempt to enhance your final product beyond what the specification requires. Adding unwanted functionality can and will impact your grade!

OBJECTIVE

Design a schema to support a consolidated view of the products offered and sold in all stores across the country.

Attention: You should create a normalized schema with as little redundancy as possible.

STORE

- Store
  - unique store number
  - store's phone number
  - store's street address (city name, state, store's zip code)

- Note: It is possible that multiple stores are located in the same city.

CITY

- City
  - City Name
  - State
  - Population of city

PRODUCT

- Product
  - Product ID (PID)
  - Product Name
  - Retail Price
  - Sale Date
  - Product Category

- Notes:
  - We will assume that all products are available and sold at all stores — that is, there is no need to specify that a certain product is only available at a certain store.
  - Each category has a name, which we assume to be unique. Every product must be in at least one category.

MANUFACTURER

- Manufacturer
  - Unique numeric id
  - Manufacturer Name

- Note:
  - It is possible that multiple products are made by the same manufacturer.

ONSALE

- OnSale
  - Sales Start Date
  - Sales End Date
  - Sales Duration [Constraints: no more than 25 days (i.e. less than or equal to 25 days)]
  - Product ID (PID)
  - Sale Price [constraints: lower than retail price]
  - Maximum discount [constraints: no more than 90% off the retail price]
  - Holiday Name

- Notes:
  - The retail price is in effect unless there is a sale.
  - If a product is on sale for multiple days in a row, then a record is stored in the data warehouse for each day of the sale. It is possible that the same product goes on sale multiple times (i.e., on several different days) with different sale prices on different sale days.
  - At the same time, if a product goes on sale on a certain day, it is on sale at the same price on this day in all stores—i.e., stores are not allowed to hold sales independently or have store-specific sale prices.
  - The data warehouse disallows sale prices that are higher than retail prices. [Constraints]
  - Even if a manufacturer does not specify a maximum discount, Peachtree Savings Club's general rule is that no product may be discounted more than 90% off the retail price.
  - Savings Club administration does not allow any product to be on sale for more than 25 days in any calendar month.

SALES

- Sales
  - Date
  - Product ID (PID)
  - Store
  - Quantity Sold
  - Listed Price

- Notes:
  - We will assume that our company does not need to deal with any sales tax.
  - Also, the data warehouse is not required to store which products were purchased together during a single sales transaction.

MEMBERSHIP

- Membership
  - Member ID
  - Signup Date
  - Signup Store
  - Membership Type [Domains: “Basic”, “Gold”, “Platinum” or “Diamond”]
- Notes:
  - Memberships can only be purchased in-store; online or mail signups are not supported.
  - Peachtree Savings Club currently offers four levels of membership: “Basic”, “Gold”, “Platinum” or “Diamond”.
  - There is a possibility, however, that the executive team will add a few more levels. Therefore, the DBA should have an option to update the list of possible levels, although this update will not be performed through our application: the DBA will use a manual SQL command to implement it.

DATA REQUIREMENTS

Savings Club's data analytics group is currently working on extracting sample data from their point-of-sale system for us to test our data warehouse functionality. However, to avoid revealing confidential information, the data security team has directed them to use old data limited to certain categories and has refused to allow any newer data to be used. Retrieving the data from tape backup and sanitizing it will take at least two to three months before it can be made available to us. Therefore, we will need to ensure that by that time our schema design matches the data as described here so that any transformations prior to loading the data into the database are kept to a minimum.

Phase 2

Phase 2	Design: • (revised) EER diagram • EER to Relational mapping. (25%) • SQL Create Table statements (25%) • Task designs w/abstract code that refers to EER replaced w/SQL that refers the relations (50%)	5, 6, 7, 9	Methodology III: Design Methodology III: Design [SQL]

Sample Documents:

- Project Spec Doc

USER INTERFACE

1. All reports will be accessible from a “dashboard” UI that must be developed.
2. On this main menu, the following statistics should be displayed along with any buttons/links to reports or functionalities:
    - Count of stores, manufacturers, products, and memberships sold
3. We must provide an interface for holidays to be maintained by the user. This interface must allow for viewing and adding holiday information (but not editing or deleting holidays) directly within the user interface.
4. Our UI must allow for updating the population of any city in the data warehouse, should a city’s population change.

REPORT 1 – MANUFACTURER'S PRODUCT REPORT

Generate Query for the following views:

- Manufacturer id
- Manufacturer Name
- Total Number of Products offered by the manufacturer
- Average retail price of all the manufacturers' products
- Minimum retail price
- Maximum retail price
- Retail price range (i.e., Maximum retail price – Minimum retail price)

- Sort by: Average retail price in descending order
- Show only the top 100 manufacturers based on average price

Note: In this report, only retail prices are used in calculations. Actual transactions at discounted prices are not considered.

REPORT 1 – MANUFACTURER'S PRODUCT REPORT (DRILL-DOWN)

- Drill-down details for each manufacturer, the following will be included in header:
  - Manufacturer id
  - Manufacturer name
  - Maximum discount
  - Product id
  - Product Name
  - Product Category
  - Retail Price
- Sort by: Retail Price in descending order, Product id in ascending order
- Note: If a product has multiple categories, it must not show up as multiple rows on the report, but as a single row with multiple categories concatenated together in alphabetical order, separated by forward slashes.

REPORT 2 – CATEGORY REPORT

Generate Query for the following views:

- Category Name
- Total number of products in that category
- Total number of unique manufacturers offering products in that category
- Average Retail Price
- Total revenue for all products sold in that category

- **Sort by: Category Name in ascending order**

Note: The revenue should certainly consider actual selling price, i.e., if the product was sold at retail price or at discounted price in this case).

REPORT 3 – ACTUAL VERSUS PREDICTED REVENUE FOR SPEAKER UNITS

For each product in Speaker category, generate Query views for the following items:

- Product id
- Product Name
- Retail Price
- Total number of Units sold
- Total number of Units sold at a discount (i.e., during sale days)
- Total number of Units sold at retail price
- Actual revenue
- Predicted revenue had the product never been put on sale (based on 75% volume selling at retail price)
- The difference between actual revenue and the predicted revenue (i.e., diff = actual revenue – predicted revenue)

- Sort by: predicted revenue differences in descending order
- Show only predicted revenue differences greater than $5000 (positive or negative)

- Note:
  - If the difference is a positive number, it means that the discounts worked in favor of Peachtree Savings Club, because the predicted revenue is less than the actual revenue collected.
  - If it is a negative number, it indicates that the company would have been better off not offering the discounts.

REPORT 4 – STORE REVENUE BY YEAR BY STATE (DROP-DOWN BOX)

Generate Query views for the following items:

- States (drop-down box)
- Store Number
- Store Address
- Store zip code
- City Name
- Sales Year
- Total revenue
- Revenue for products sold at retail price
- Revenue for products sold at discounted price

- Sort by: Sales year in descending order, Revenue in descending order

REPORT 5 – AIR CONDITIONERS ON GROUNDHOG DAY?

Generate Query views for the following items:

- Year
- Total number of items sold that year in the air conditioning category
- Average number of units sold per day
- Total number of units sold on Groundhog Day (February 2) of the year

- Sort by: Year in ascending order

Note: Assume a year is exactly 365 days, no matter how many days of the year are represented in the database.

REPORT 6 – STATE WITH HIGHEST VOLUME FOR EACH CATEGORY (DROP-DOWN BOX)

Selection in drop-down box:

- Year
- Month

Generate Query views for the following items:

- Product Category Name
- States that sell the greatest numbers of units for each category
- Total number of units that were sold in all stores in that state

- Sort by: Product Category Name in ascending order

- Note: Note that each category will only be listed once unless two or more states are tied for selling the highest number of units in that category.

REPORT 7 – REVENUE BY POPULATION

Generate Query views for the following items:

- Year
- Average revenue for small cities
- Average revenue for medium cities
- Average revenue for large cities
- Average revenue for extra large cities

- **Sort by: Year in ascending order**

- Note:
  - The categories for city size are:
    - Small (population <3,700,000),
    - Medium (population >=3,700,000 and <6,700,000),
    - Large (population >=6,700,000 and <9,000,000)
    - Extra Large (population >=9,000,000)

REPORT 8 – MEMBERSHIP TRENDS

Generate Query views for the following parts:

1. Total number of memberships sold by year
    - Year
    - Total number of memberships sold for that year
- Sort by: Year in descending order

REPORT 8 – MEMBERSHIP TRENDS

2. Top 25 cities that sold the most memberships (drill-down by year)
    - Year (drill-down)
    - City Name, State
    - Total number of memberships sold for that city
- Sort by: Number of memberships in descending order, alphabetical order of city names in ascending order

3. Bottom 25 cities that sold the least memberships (drill-down by year)
    - Year (drill-down)
    - City Name, State
    - Total number of memberships sold for that city
- Sort by: Number of memberships in ascending order, alphabetical order of city names in ascending order
- Note:
    - Cities which have sold 250 or more memberships should have their total highlighted with a green background.
    - Cities which have sold 30 or fewer memberships should have their total highlighted with a red background.
    - Since same city names can be used in different states, be sure to include the state the city is in. The header of this report should include the selected year.

REPORT 8 – MEMBERSHIP TRENDS

4. City to store level drill-down report
    - Store number
    - Store street address
    - Store zip code
    - City
    - State
    - Total number of membership sold during that year in this store

- Note: Cities must have more than one store.

Phase 3

Phase 3	Implementation: • Evaluation of completed functionality demonstrated by your team, incorporating inserts, updates, deletes, and queries • Specific criteria will be provided later this semester	10, 11, 14, 15.1, 15.2, 16, 17	Methodology IV: Implementation