MGMT 58200

Management of Organizational Data

Database for Design Alternatives

Session 2 Group 10

Team Member: Li-Ci Chuang, Mu-Hua Hsu, Xue Han, Yu-Lin Tai

Background:

Design Alternatives is a local interior design business located in Lafayette, IN. The company is owned by Susan Benedict, a certified kitchen designer whose expertise is unsurpassed in the area. Founded in 1994, Design Alternatives started out offering solutions to kitchen and bathroom designs that match people's lifestyles but now extends its business to designing more areas in the house.

Mrs. Benedict has working experience of taking care of the seniors. As an interior designer, she found out that most housing facilities were not friendly enough to the elders and disabilities. Therefore, in addition to offering pleasing designs for normal demand, Design Alternatives is distinguished as the leader of *Aging in Place* nowadays. The company utilizes inclusive techniques that go far beyond the American Disabilities ACT. Using universal design principles and assistive technologies, Design Alternatives aims to help its customers create space with high livability, comfort, and creative style.


- **Current Status:**

Design Alternatives is a small local business. Besides Mrs. Benedict, the company has eight designers, half are lead designers and the other half are interns. The designers are responsible for customer consulting, design drawing according to the customers' requirements, and ordering the products, such as materials or furniture, from the vendors. The profit comes from

the price spread and the designing fees. The main cost is composed of two parts: purchase cost of products to vendors and designers' payroll. The payment to the vendors is based on orders, while the payment to the designers are calculated from their timesheets.

Like other traditional small business owners, Mrs. Benedict used to keep all the information in paper documents, which caused a dreadful mess in her office and resulted in difficulties in retrieving information. The only electronic data resources are the designers' timesheets she received by emails and the files exported from QuickBooks, an accounting software for small businesses. However, due to the complexity of this accounting system, it is still challenging to retrieve directly useful and organized information. For the payroll part, Mrs. Benedict still has to calculate the payments manually.

- **Description of the Original Dataset**

Right now, the company provides five datasets, some of which are paper documents, and some of which are excel files. The first one is called the Task dataset. It includes the task name, the related contacting customer information, and corresponding designers responsible for this task. However, this dataset is quite messy because the task name is irregular and it is combined by the customer name and some descriptions. Redundant issues are also common since several tasks might belong to one customer, but contact information is repeatedly recorded. We also received a Designer dataset that includes the designer's wage, position, and private information. These two above datasets are all related to a timesheet data set. That is, it tells us which designer works on which project at a specific time period. Vendor data set is a dataset containing all information of vendors that the company once cooperated with. The Checks dataset includes the money checks that the company has signed, including checks signed for the designers and the vendors.

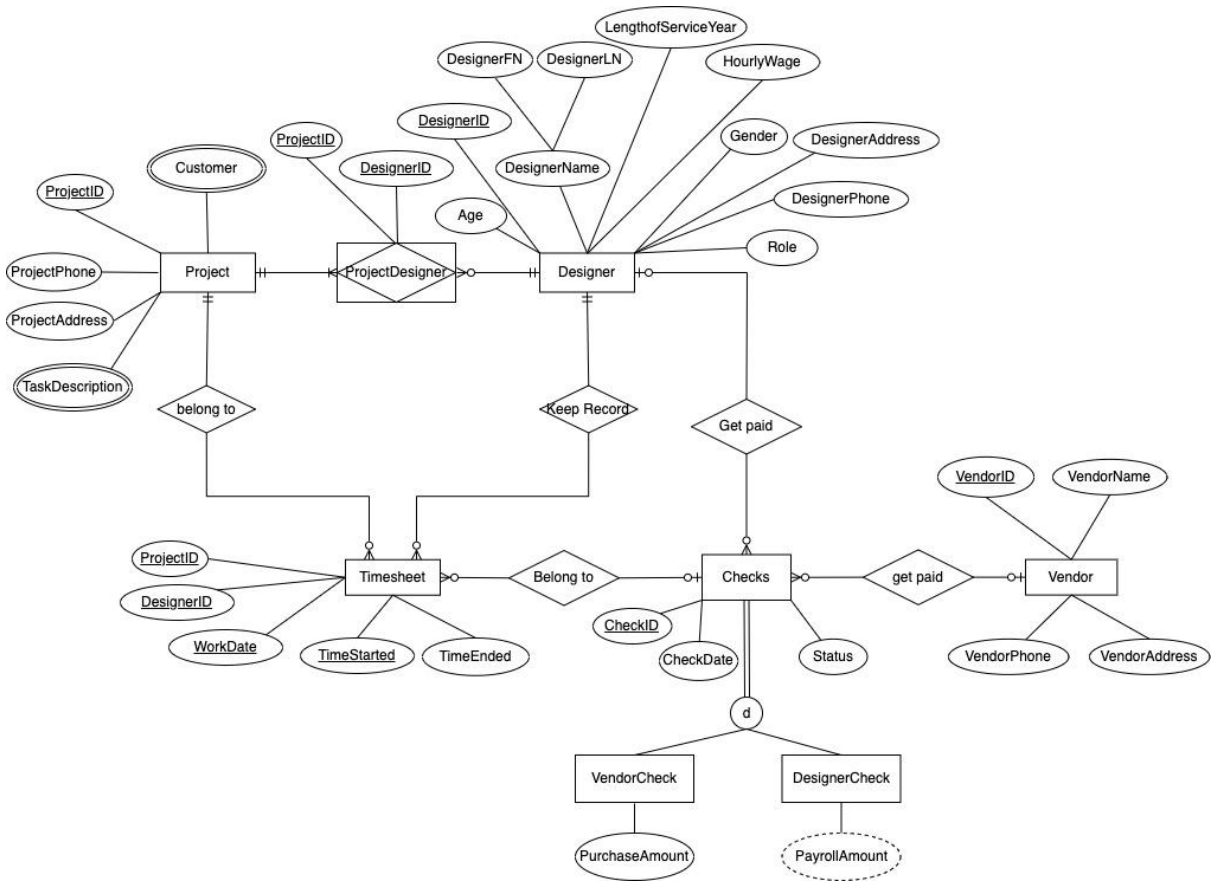
Introduction:

The business owner Mrs. Benedict was struggling with managing designer work and their work hours to the corresponding task. The lack of management results in the difficulty of keeping track with the project process and the hurdle of calculating the allocation of payroll cost based on each project. Furthermore, Mrs. Benedict currently has to calculate the payroll of the designers manually, timing up the hourly wage to every single time slot on the timesheet from different designers individually, and then sum up to import corresponding information to the accounting system for further payments.

In the post-Covid days, Mrs. Benedict also realized that people are less willing to pay for putting their house interior; therefore, she would like to make some suggestions about how to retain previous customers and keep track of both the labor and operating expenses.

By building up a database, the payroll process in Mrs. Benedict's business could be dramatically improved in several aspects. Firstly, calculations could be automatically made in the database, reducing the possibility of misstatement due to manual process. Furthermore, leveraging the power of the database, Mrs. Benedict could easily make queries to group data in different ways to follow up on either designers' working hours or project process at any point of time. Moreover, information could be stored in an organized way and easily retrieved to make additional decisions, such as budgeting the cost of a future project based on historical data and how much time and resources should be devoted in certain cases, collecting valuable information about the vendors and customers, and tracking operating expenses and labor performance.

Conceptual Data Modelling:



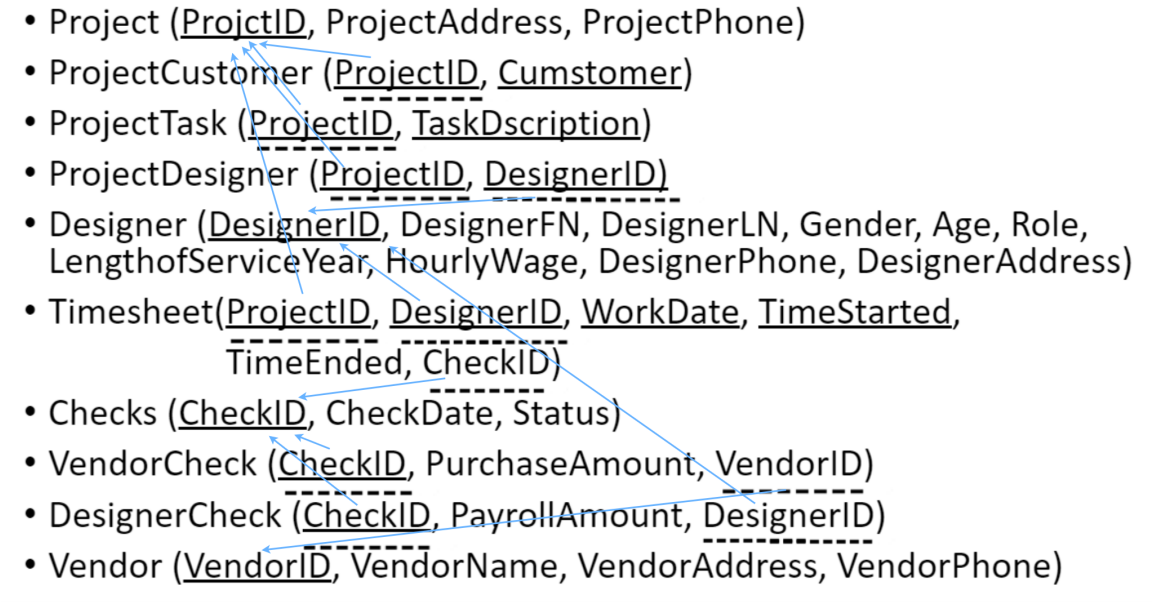
Business Requirements:

- Each designer can work on many projects or nothing at all, while each project could be conducted by one or multiple designers.
- One project may include zero or many timesheet records. But each timesheet record belongs to one and only one project.
- Every designer keeps zero to many timesheet records while each record corresponds to one and only one designer.
- Every designer gets no payment or gets paid with multiple checks. Every check is paid to one designer or not.

- Similarly, every vendor gets no payment or gets paid with multiple checks. Every check is paid to one vendor or not. Every check is paid to either a designer or a vendor.

- Every timesheet record is the basis of the payable amount of only one check, or not calculated yet. Each check paid to a designer is calculated on one or more timesheet records. But if it is paid to a vendor, then it has no relationship with the timesheet entity.

Relational Data Model (3NF)



Note: 0NF relational schema is included in the appendix.

Commentary on Data Model & Design Choices:

The main goal of our database design is to map the task and work hours in order to ease the difficulty of keeping track of the project process and the burden of calculating the allocation of payroll costs. Therefore, we have to relate the hourly wage, which is under the designer entity, as calculating bases and the task dataset, to the timesheet entity for checking what the employees actually work on. However, we found a serious issue when handling the task dataset. The task names in the dataset are combinations of several same and multiple customers with different task descriptions. Such a case could cause non-atomic issues. Besides, several tasks might belong to the same clients so that the contact information will be repeatedly stored, which causes data redundancy issues. Therefore, we decided to create a new dataset called Project, which is generated by grouping the tasks. Each project will include a unique project ID, the tasks belonging to this project, as well as the customers belonging to this project. To avoid non-atomic issues, we set task descriptions and customers as multiple attributes.

After converting the tasks dataset into a project entity, we can connect the Project, Designer, and Timesheet entity to each other. Since the Designer and Project have a many to many relationship, we added an associative entity between them.

The next objective of our project is to understand whether the payroll is paid as well as the expense composition. In order to do this, we can use Checks Entity as our data source. Therefore, we connected the Checks with the one who got paid, the Vendor or the Designer. Since there are two kinds of checks recording some of the different information, we decided to make the checks signed to the designer and the checks signed to vendors as the subclass (VendorCheck, CustomerCheck) to improve the database space efficiency.

Queries:

Query1

Natural language description of the query

We wanted to know task duration in hour units and total wage for each record of the timesheet.

The SQL code and the results

```
# Query1
# Calculate designers' task duration, total wage per record of timesheet
SELECT Timesheet.DesignerID, Designer.DesignerFN, Designer.DesignerLN, ProjectID,
WorkDate, TIME_TO_SEC(timediff(TimeEnded, TimeStarted))/3600 AS DurationHr,
Designer.HourlyWage, @wage := ROUND(TIME_TO_SEC(timediff(TimeEnded,
TimeStarted))/3600*HourlyWage, 2) AS Wage
FROM Timesheet LEFT JOIN Designer ON Timesheet.DesignerID =
Designer.DesignerID
ORDER BY WorkDate;
```

[Output]

DesignerID	DesignerFN	DesignerLN	ProjectID	WorkDate	DurationHr	HourlyWage	Wage
1	Katie	Switzer	375	2021-06-29	2.5833	17	43.92
1	Katie	Switzer	213	2021-06-29	0.5000	17	8.50
1	Katie	Switzer	213	2021-06-30	0.3333	17	5.67
1	Katie	Switzer	375	2021-06-30	3.5000	17	59.50
1	Katie	Switzer	375	2021-07-01	0.4167	17	7.08
8	Amanda	Anthony	453	2021-07-01	3.5000	10	35.00

Business problem intended to resolve

Because the information of the timesheet includes designer ID, TimeStarted and TimeEnded only, the client can not map to the total wage (which depends on the duration and different hourly wage) per record. We calculated the duration in hour units first, then mapped DesignerID to the specific HourlyWage, then can get the final wage. This solves the client problem that she always needs to do time unit conversion manually and repeated calculation for each employee.

Query2

Natural language description of the query

We want to know which employee has the most monthly payment in each role.

The SQL code and the results

```
SELECT *
FROM(SELECT Year(Timesheet.WorkDate) AS Year, Month(Timesheet.WorkDate) AS
Month, Designer.Role,
Designer.DesignerID, Designer.DesignerFN, Designer.DesignerLN,
SUM(ROUND(TIME_TO_SEC(timediff(TimeEnded, TimeStarted))/3600*HourlyWage,
2)) AS TotalWage
FROM Timesheet INNER JOIN Designer ON Timesheet.DesignerID =
Designer.DesignerID
GROUP BY Year, Month, Designer.DesignerID) AS T
WHERE TotalWage = (SELECT MAX(TotalWage) FROM Timesheet INNER JOIN
Designer ON Timesheet.DesignerID = Designer.DesignerID)
GROUP BY Year, Month, Role;
```

[Output]

Year	Month	Role	DesignerID	DesignerFN	DesignerLN	TotalWage
2021	6	Lead Designer	1	Katie	Switzer	117.59
2021	7	Intern	8	Amanda	Anthony	230.00
2021	7	Lead Designer	1	Katie	Switzer	79.33
2021	8	Intern	2	Abby	Mikel	245.00
2021	8	Lead Designer	5	Linda	Walamooth	502.50
2021	9	Intern	2	Abby	Mikel	265.00
2021	9	Lead Designer	5	Linda	Walamooth	326.25
2021	10	Intern	2	Abby	Mikel	50.00

Business problem intended to resolve

Once the client gets the salary information in the timesheet, she wants to know which employee she needs to pay the most in both Lead Designer and Intern role per month. Further, she can get the result that she wants to see to check the individual salary amount given the time period by putting ”Having” query additionally.

Query3

Natural language description of the query

List all the expected designer payment amounts in the timesheet which are not yet paid per month.

The SQL code and the results

```
SELECT Year(Timesheet.WorkDate) AS Year, Month(Timesheet.WorkDate) AS Month,
Designer.DesignerID, SUM(ROUND(TIME_TO_SEC(timediff(TimeEnded,
TimeStarted))/3600*HourlyWage, 2)) AS TotalWage
FROM Timesheet INNER JOIN Designer ON Timesheet.DesignerID =
Designer.DesignerID
WHERE CheckID IS NULL
GROUP BY Month, DesignerID
ORDER BY Month, DesignerID;
```

[Output]

	Year	Month	DesignerID	TotalWage
▶	2021	10	1	527.02
	2021	10	2	50.00
	2021	10	3	55.00
	2021	10	4	105.00
	2021	10	5	548.75

Business problem intended to resolve

The salary payment is scheduled to proceed by month, and one check is for each unique designer only. We filtered the taks records that are not paid yet in the Timesheet, so that the owner can summarize the total salary amount per designer to finish the payment.

Query4

Natural language description of the query

We want to know which companies are Design Alternatives' top vendors.

The SQL code and the results

```
SELECT  Vendor.VendorName, sum(PurchaseAmount) AS TotalAmount,
count(VendorCheck.VendorID) AS Frequency FROM VendorCheck
LEFT JOIN Vendor ON Vendor.VendorID = VendorCheck.VendorID
GROUP BY VendorCheck.VendorID
HAVING TotalAmount >  500 or Frequency > 1
ORDER BY Frequency DESC, TotalAmount DESC;
```

[Output]

	VendorName	TotalAmount	Frequency
▶	Frank Anderson	1853.85	5
	Hicks Gas	1543.2299999999998	4
	Biggs	1351.45	3
	OnePointe Solutions	1290.7199999999998	3
	Baer Supply	1065.98	3
	Practicon	986.01	3
	Skyway Concession Co	956.15	3

Business problem intended to resolve

The relationship between companies and suppliers is important, but it is difficult to maintain good relationships with all of the suppliers. Using this query, our client can understand with which vendors it has the most frequent transactions as well as the corresponding total trading amount, in order to prioritize to improve relationships with these top vendors.

Query5

Natural language description of the query

(1)
Input new payroll check information into the database (insert a new row in table Checks and DesignerCheck, and update CheckID information in table Timesheet).
For example, on December 2nd, 2021, Mrs. Benedict wrote a check to Abby (DesignerID=2) for her work in October, 2021. And the CheckID is 5547.

(2)
Update check status when the check is cashed.

The SQL code and the results

(1)

```
INSERT INTO Checks VALUES (@newcheck := 5547, '2021/12/02', 'Outstanding');
INSERT INTO DesignerCheck VALUES (@newcheck, @payee :=2,(SELECT SUM
(ROUND (TIME_TO_SEC (timediff (TimeEnded, TimeStarted)) / 3600*HourlyWage, 2))
FROM Timesheet INNER JOIN Designer ON Timesheet.DesignerID =
Designer.DesignerID
      WHERE Year(Timesheet.WorkDate) = 2021 AND Month(Timesheet.WorkDate) = 10
AND Designer.DesignerID = @payee AND CheckID IS NULL));

UPDATE Timesheet SET CheckID = @newcheck
WHERE Year(WorkDate) = 2021 AND Month(WorkDate) = 10 AND DesignerID =
@payee AND CheckID IS NULL;
```

[Output]
Checks:

CheckID	CheckDate	Status
5547	2021/12/02	Outstanding

DesignerCheck:

CheckID	DesignerID	PayrollAmount
5547	2	50.00
5547	2	50.00

(2)

```
UPDATE Checks SET Status = 'Cashed' WHERE CheckID = @newcheck;
```

[Output]

CheckID	CheckDate	Status
5547	2021/12/02	Cashed

Business problem intended to resolve

Our client writes checks to her designers for every month's performance. She'd like to find an easy way to help her calculate how much she needs to pay and also keeps track of these payments in different tables synchronously.

Query6

Natural language description of the query
List the top 5 loyal customers through the frequency of projects.
The SQL code and the results
SELECT ProjectCustomer.Customer, COUNT(Timesheet.ProjectID) AS FrequencyofCustomer FROM ProjectCustomer INNER JOIN Timesheet ON Timesheet.ProjectID = ProjectCustomer.ProjectID GROUP BY ProjectCustomer.Customer ORDER BY FrequencyofCustomer DESC LIMIT 5; [Output]

Customer	FrequencyofCustomer
Painter	7
Rena Aiken	4
Braz	4
Alex Eros	4
Heidbreder	4

Business problem intended to resolve
The owner wants to know about customer retention, this can be captured by the number of projects which are requested by the same customer, that is the FrequencyofCustomer column in the result.

Query7

Natural language description of the query
We want to know the time spent and the cost group by designers of project No.375
The SQL code and the results
SELECT Timesheet.DesignerID, Designer.HourlyWage, sum(TIME_TO_SEC(timediff(Timesheet.TimeEnded, Timesheet.TimeStarted))/3600) AS DurationHr, HourlyWage*sum(TIME_TO_SEC(timediff(Timesheet.TimeEnded, Timesheet.TimeStarted))/3600) AS LaborCost FROM Timesheet INNER JOIN Designer ON Designer.DesignerID = Timesheet.DesignerID WHERE ProjectID = 375 GROUP BY Timesheet.DesignerID;

[Output]

	DesignerID	HourlyWage	DurationHr	LaborCost
▶	1	17	8.2500	140.2500
	7	20	2.5833	51.6667

Business problem intended to resolve

By selecting a specific Project ID, the owner of the company could easily understand how much time each designer works on the project and the corresponding cost that the owner has to bill to the customer. Our client could efficiently keep track of designers' work and also perform budgeting to new cases.

Query8

Natural language description of the query

We would like to understand the proportion of the project from the local customers.

The SQL code and the results

```
SELECT
count(CASE WHEN ProjectAddress LIKE "%Lafayette%" then 1 else null end)/count(*)
AS LocalProjectRatio
FROM Project;
```

[Output]

	LocalProjectRatio
▶	0.5545

Business problem intended to resolve

Design Alternatives is a small business locally founded in Lafayette, so the clients are most likely from this area. As the business grew, Design Alternatives now has many more clients from outside Lafayette. Using the query, the company could have a picture of its market strategy.

Query9

Natural language description of the query

Identify which project has the longest duration and how many designers work for this project

The SQL code and the results

```
SELECT ProjectID, (Max(WorkDate) - min(WorkDate) + 1) AS ProjectDays,
count(DesignerID) AS NumberofDesigner FROM Timesheet
GROUP BY ProjectID
```

ORDER BY ProjectDays DESC,count(DesignerID) DESC;
[Output]

ProjectID	ProjectDays	NumberofDesigner
453	328	4
411	313	3
538	311	2
494	310	2
261	300	2
324	294	3
213	293	4

Business problem intended to resolve

It is necessary to track the time span (days) for each project. The company could keep an eye on the project which has a longer period, because customer satisfaction might be affected if the project didn't close in an expected time. Based on the result, the company could also add or cut the number of designers for that project.

Query10

Natural language description of the query

We want to know what amount of money from the check hasn't been cashed.

The SQL code and the results

```
SELECT (COALESCE(SUM(PurchaseAmount),0)+
COALESCE(SUM(PayrollAmount),0)) AS TotalOutstanding FROM Checks
LEFT JOIN VendorCheck on VendorCheck.CheckID = Checks.CheckID
LEFT JOIN DesignerCheck on DesignerCheck.CheckID = Checks.CheckID
WHERE Status = "Outstanding";
```

[Output]

TotalOutstanding
1561.6399999999999

Business problem intended to resolve

Design Alternatives pays some of the designers and vendors by checks. Since it is important for a small company to keep track with its cash flow, the use of the query could help calculate the total amount of outstanding check, giving the owner a clearer idea about the correct amount of money that she could manage.

Apendix:

Relational Schema 0NF

- Project (ProjectID, TaskDescription, ProjectAddress, ProjectPhone, Customer)
- ProjectDesigner (ProjectID, DesignerID)
- Designer (DesignerID, DesignerFN, DesignerLN, Gender, Age, Role, LengthofServiceYear, HourlyWage, DesignerPhone, DesignerAddress)
- Timesheet(ProjectID, DesignerID, WorkDate, TimeStarted, TimeEnded, CheckID)
- Checks (CheckID, CheckDate, Status)
- VendorCheck (CheckID, PurchaseAmount, VendorID)
- DesignerCheck (CheckID, PayrollAmount, DesignerID)
- Vendor (VendorID, VendorName, VendorAddress, VendorPhone)