**tso omvs sessions(3)**

Assignment 3 – Theory   
Spring 2018

***laverty@rmu.edu***

Contents

[Introduction 4](#_Toc504135178)

[Researching Your Answers 4](#_Toc504135179)

[A Little Advice Before you start 5](#_Toc504135180)

[Demonstrating Knowledge and Increased Penalties for Irrelevant Answers 6](#_Toc504135181)

[You Must Submit YOUR Answer in this Original Word Document to Blackboard 6](#_Toc504135182)

[You Must RENAME this Original Word Document to Include your LAST NAME 7](#_Toc504135183)

[NEVER submitted an Assignment as an Email Attachment 7](#_Toc504135184)

[ONLY Submit a FINAL Version of ALL Assignment 7](#_Toc504135185)

[Requests to Clear Previously Assignments for Re-Submission 7](#_Toc504135186)

[Submitting Late Assignments 7](#_Toc504135187)

[Academic Integrity and Plagiarism 8](#_Toc504135188)

[How to complete Content Questions 9](#_Toc504135189)

[1.0 Introduction to Database Design 10](#_Toc504135190)

[1.1 Conceptual design 10](#_Toc504135191)

[1.2 Logical design 11](#_Toc504135192)

[1.3 Bottom-up, Top-down, or Combination Design Approaches 11](#_Toc504135193)

[1.4 Overview of the Steps Recommended for Database Design 12](#_Toc504135194)

[1.5 How does one determine an entity or table? 13](#_Toc504135195)

[2.0 Normalizing a Database 14](#_Toc504135196)

[2.1 Rule 1 -Separate all repeating fields and store these repeating fields into a separate table. 14](#_Toc504135197)

[2.1.2 Primary Key and Non-Key Columns 15](#_Toc504135198)

[2.1.2 Composite Primary Keys and Atomic Primary Key 16](#_Toc504135199)

[2.1.3 Limitations of Composite Primary Key 16](#_Toc504135200)

[2.1.4 Natural Primary Keys 16](#_Toc504135201)

[2.1.5 Advantages of an Auto Increment Primary Key 17](#_Toc504135202)

[2.2 Rule 2 - All non-key fields should be functionally dependent upon the "entire" primary key. 17](#_Toc504135203)

[2.3 Rule 3 - Eliminate any non-key fields that can be created, calculated or derived by a combination of the other non-key fields, within the row or between rows. 19](#_Toc504135204)

[2.4 A Four-Table Normalized Design to Store a Customer's Invoice 20](#_Toc504135205)

[2.5 Bachman Diagrams 20](#_Toc504135206)

[2.6 Pros of a Normalized or De-normalized Database Design 21](#_Toc504135207)

[2.6.1 Pros and Cons of a Normalized Database Design 21](#_Toc504135208)

[2.6.2 Pros and Cons of De-normalized Database Design 21](#_Toc504135209)

[2.7 Comparing Transactional Database Storage Design to Data Warehouse Design (OLTP to OLAP) 22](#_Toc504135210)

[2.8 Descriptive vs. Predictive vs. Prescriptive Data Analytics 22](#_Toc504135211)

[2.9 Terrorist Attacks: A Comparison of Descriptive, Predictive and Prescriptive Analytics 23](#_Toc504135212)

[2.10 Jeff Jonas (IBM researcher) and Contextual Data Processing Models 25](#_Toc504135213)

[3.0 Index Analysis and Design 26](#_Toc504135214)

[3.1 Advantages of Database Indexes in Conceptual Design 26](#_Toc504135215)

[3.1 Disadvantages or Costs of Indexes 28](#_Toc504135216)

[3.3 Applying Index Design to a Customer Invoice Database 29](#_Toc504135217)

[3.3.1 Stock Number Index in the Invoice Item Table 29](#_Toc504135218)

[3.3.2 Invoice Date Index in the Invoice Header 29](#_Toc504135219)

[3.3.3 Purchase Order Number Index in the Invoice Header 29](#_Toc504135220)

[3.3.4 Customer Number Index in the Invoice Header Table 30](#_Toc504135221)

[4.0 Data Relationships 31](#_Toc504135222)

[4.1. One-To-One Relationships 31](#_Toc504135223)

[4.2 One-To-Many Relationships 33](#_Toc504135224)

[4.3 Many-To-Many Relationships 34](#_Toc504135225)

[4.3.1 Student – Major Many-to- Many Relationships 34](#_Toc504135226)

[4.3.2 Invoice – Inventory Many-to- Many Relationships 34](#_Toc504135227)

[4.4 Appling Relationship Analysis to the Customer Invoice Database 36](#_Toc504135228)

[5.0 Review Questions 38](#_Toc504135229)

[5.1 Questions - Functional and Data Analysis 38](#_Toc504135230)

[5.2 Questions - OLTP and OLAP 38](#_Toc504135231)

[5.3 Questions - Bottom–Up versus Top-Down Database Design Approaches 40](#_Toc504135232)

[5.4 Questions - Limitations Selecting a Primary Key 41](#_Toc504135233)

[5.5 Questions - Natural Key versus Auto-incrementing Key Design Considerations 42](#_Toc504135234)

[5.6 Questions - Data Relationships 43](#_Toc504135235)

[6.0 Applying Normalization and Indexes Analysis 46](#_Toc504135236)

[7.0 Entity Relationship Modeling (ERD Diagrams) 48](#_Toc504135237)

[7.1 Introduction to ERD: Nouns versus Verbs 48](#_Toc504135238)

[7.2 Common Errors Identifying Major Nouns and Entities 49](#_Toc504135239)

[7.3 ER Diagrams Usage 50](#_Toc504135240)

[7.3.1 Database ERD Diagrams 50](#_Toc504135241)

[7.3.1 Software Engineering ERD Diagrams 50](#_Toc504135242)

[7.4 ER Diagram Symbols and Notations 50](#_Toc504135243)

[7.4.1 Entity 51](#_Toc504135244)

[7.4.2 Weak Entity 51](#_Toc504135245)

[7.4.3 Attribute 51](#_Toc504135246)

[7.4.4 Multivalued Attribute 52](#_Toc504135247)

[7.4.5 Derived Attribute 52](#_Toc504135248)

[7.5 Relationship 53](#_Toc504135249)

[7.51 Recursive Relationship 53](#_Toc504135250)

[7.5.2 Multiplicity, Cardinality and Participation 54](#_Toc504135251)

[7.5.3 Degrees of Relationship (Cardinality) 54](#_Toc504135252)

[7.6 How to Draw ER Diagrams 54](#_Toc504135253)

[7.7 ER Diagram Best Practices 55](#_Toc504135254)

[7.8 Generalization Specialization, Aggregation and Composition in DBMS 56](#_Toc504135255)

[7.8.1 Generalization and Specialization in DBMS 56](#_Toc504135256)

[7.8.2 Aggregation in DBMS 58](#_Toc504135257)

[7.8.3 Composition in DBMS 59](#_Toc504135258)

[7.9 Questions - Applying ERD Concepts 60](#_Toc504135259)

[7.9.1 Examples and Description of UML Concepts 61](#_Toc504135260)

**Enter your Name Here 🡺**

# Introduction

## Researching Your Answers

Most requirements of this assignment will require you to research answers from your text book (**you must read the text book to get some answers**), from the Internet, from a video or any other reasonable source. Many Internet sources, video links, text book and Blackboard presentations are provided in this document to help you START your research.

Assignments are always a great place to read and reference your text book. May students assume that they should start by reading the text book. While this reading the text book is ALWAYS at great idea, the following assignment questions may help you focus on what is important in this course***. One strategy is to review an assignment question for important keywords, e.g., multitasking, Procedure Division, virtualization, etc. Then look for those key words in either 1) your text book "detailed" table of contents in the beginning of the book, or 2) in the index at the end of the book. Some text books have key word glossaries at the end of the chapter or end of the book.***

Every semester the Internet sources, e.g., a Google keyword search, or Internet video, e.g., a YouTube keyword search may be improved. Better students start with research sources provided in this assignment, and then search for improved or updated answers. ***While your objective may be to demonstrate your knowledge to EARN an excellent grade on this assignment, better students are always looking towards the future to impress internship and career recruiters for those interesting and high entry-level salaries.*** Employers don't pay you for a grade in any single class or assignment. Employers will pay you BIG MONEY for demonstrated knowledge or skills. Hopefully, this assignment will prepare you.

A single research source maybe very incomplete or the format or the content may not be appropriate for some required answers. Some students do not READ the text book or review the Blackboard presentations provided to you in Blackboard. Some answers are only available from Blackboard presentations or update documents. When assignment requirements may be only answered by viewing a YouTube video, you will be notified.

You may cut-and-paste answers whenever appropriate. You MUST synthesize your answers to include multiple sources. I would be impressed that you consult a Google image search and paste and appropriate image to supplement your answer.

While I permit cut-and-paste, I do expect you use your own words so that you:

1. Organize the answer

2. Demonstrate that you have read what you have cut-and-pasted

3. Use any means that clearly displays that you have gained knowledge.

## A Little Advice Before you start

There is NO requirement to read each reference link or view each video in detail. Some of the links will have overlapping content. Some links will provide more information than the question requirements, but employers consider these topics to be minimal knowledge of a RMU CIS graduate.

It is recommended to visit each reference link and overview the content. Then read each question and return to each reference link or video as needed. You may supplement your answers with content from your text book by using a question keyword and looking up in the glossary or index in the back of the book. PDF text books can be easily searched for keywords.

It is not required to read your text books before working on this assignment unless specified in a requirement. Text book contents are used to support quiz questions (which contain the answers), which are used on the tests. The reference links and videos are excellent resources.

The topics presented in the course assignments have been highly recommended by two or more employers who recruit RMU CIS students as minimum computer hardware, operating system, and application development knowledge. Employers expect that students should be able to present one or two sentences of the majority of keywords applicable to job requirements in a face-to-face interview.

On the other hand, each assignment is allocated 100 points out of a total of 1000 points. The number of questions or hands-on activity on each assignment varies. Assume that an average assignment has 50 requirement questions. This means that a requirement may be worth 2 points towards your final grade. The bottom line is that missing a few questions will have little effect of your assignment or final grade. Not completing an assignment will generally decrease your final grade by at least a letter grade. Do not waste time on the small problems.

## Demonstrating Knowledge and Increased Penalties for Irrelevant Answers

If you can DEMONSTRATE your knowledge of the topic for the requirement there will be no penalty for your answer. It is not the intention of this assignment to be "not picky". Parital credit will be awarded as appropriate.

If you cut-and-paste and pray that your instructor will not read your inappropriate and irrelevant answer, the question will be penalized by increasing the deduction points beyond the original requirement points. The instructor hates irrelevant cut-and-paste BS, or answers that appears that the student is guessing and hopes the instructor does not read the answer.

***The instructor reserves the right to increase the penalty for any submitted question or assignment that may be construed as "wasting the instructor's time".*** Therefore, a four-point requirement may be penalized as six-points (two additional point penalty for wasting the instructor's time). Sometimes blank answers will earn you’re a high assignment grade than BS answers. For example, a submitted 100-point assignment may be penalized as 125 minus points when your final grade is calculated for any assignment that should have never been submitted in the first place.

## You Must Submit YOUR Answer in this Original Word Document to Blackboard

***This Assignment Word Document will contain hidden markers that may be used to detect plagiarism and provide an audit trail of those who may have modified the Word document.***  Many students in my classes work very hard to complete and learn from their assignments. It is not fair to those students who have professionally demonstrated their knowledge to receive the same grade as those who have plagiarized their assignments

**You MUST answer ALL requirement in this Word document and ONLY THIS Word Document. You MAY NOT use or edit any other word processor, except any version of Microsoft Word.**

**Do not use GOOGLE DOCS or Open Office DOCX files at any time. If you use any other Word Processor you will be assigned ZERO credit.**

**If you do not have a copy of WORD**, you may use VMWARE VIEW (available from the RMU website) to access a virtual lab computer which contains any software needed for this course.

<http://www.rmu.edu/web/cms/departments-offices/administration-services/it/Pages/vmware-view.aspx>

NEVER STORE ANY DOCUMENTS ON THE DESKTOP OF VMWARE VIRTUAL COMPUTER. You will lose your document. It is preferable to store your documents on RMU Drive U: If necessary you can email the document to yourself.

## You Must RENAME this Original Word Document to Include your LAST NAME

**YOU MUST enter your name in the beginning of this document as provided and "Save As" this document using a new name that starts with your LAST NAME, assignment number and semester, e.g., Jones Assignment 1 Summer 2018.docx**

If you do not rename your document, your assignment will be penalized by 10%.

## NEVER submitted an Assignment as an Email Attachment

All assignments are to be submitted to the instructor by using the Assignment Link in the Blackboard system. Assignments submitted as an email attachment will NOT be graded. THE INSTRUCTOR NEVER ACCEPTS ANY ASSIGNMENT AS AN EMAIL ATTACHMENT FOR ANY REASON.

## ONLY Submit a FINAL Version of ALL Assignment

Never submit an incomplete assignment for grading. Only submit your final version of ALL assignment documents for grading. You can only submit an Assignment once.

## Requests to Clear Previously Assignments for Re-Submission

If you make an error submitting an assignment, you must contact the instructor to clear your previous assignment submission. If you made an error on any assignment, you may request that the previous assignment submission be cleared so that you may resubmit the assignment again. Please only submit a completed assignment.

## Submitting Late Assignments

While the assignments have a recommended due date, the instructor does not penalized your assignment grade if you are slightly late. Please do not send the instructor an email if you are going to submit your assignment late. The instructor is flexible and assumes you have a good excuse. After you are more than two weeks late the instructor does reserve to penalize the assignment or not accept the assignment if this late submission is unfair to other students enrolled in the course who had completed their assignments on time.

It has been the experience of the instructor that students who are excessively or consistently late, may ask a friend to provide them a copy of their assignment, which will violate the RMU Academic Integrity Policy. (Please carefully read the next section!) ***If a friend asks you for a copy of your assignment "to get an idea what the instructor wants", you are risking a zero assignment grade, an F final grade, or a RMU Academic warning or suspension.***

## Academic Integrity and Plagiarism

When an instructor has possession of an electronic document, it is easy to detect plagiarism. Microsoft Word provides a variety of FREE anti-plagiarizing tools. The content of your submitted Assignment WORD document will be COMPARED to each other student who has submitted this assignment in the current class or any previous class as time permits. ***The content of each student's assignment may NOT be copied from any other current or past student enrolled in this class. Each assignment is to be prepared by ONE student. Assignments are NOT a group-prepared assignment.***

Some students may attempt to SAVE AS another student's completed assignment and rename it using their name. Some students may attempt to Cut-and-Paste answers from one student's assignment document to another student's assignment document. As time permits, the forensic tools used to compare ALL student's assignments with other assignment will often detect anomalies which will provide absolute proof of plagiarism. ***On-ground tests may be used to compare the student’s knowledge to performance on assignments. All acts of plagiarism and forensic data will be submitted the RMU Academic Integrity Board to determine university-wide penalties, such as grade penalties, warnings, suspension, and change of a previous course grade for previous course students. All current and previous students involved in the plagiarism may be affected RMU Academic Integrity Board.***

***If a friend asks you for a copy of your assignment "to get an idea what the instructor wants", you are risking a zero assignment grade, an F final grade, or a RMU Academic warning or suspension. You are responsible to protect your assignment Word Document.***

***You, however, may discuss assignment requirements, provide research assistance, assist other students to debug programs or other hands-on-requirements, tutor students, or provide other advice that may assist the students in acquiring knowledge. But the actual preparation of an individual assignment must have been completely prepared by the student who submitted the assignment. Sections of the assignments may be copied from the internet as per the individual assignment's directions. Please contact the instructor if you need assistance interpreting this RMU Academic Integrity Policy. (Ref.16-1.)***

Many believe that if you a "stupid" enough provide another student, whom may compete with you for a future internship or career, a copy of your assignment, then you deserve the same penalty as the other student. If you are a "real" friend, tutor your friend.

***The instructor reserves the right to require face-to-face hands-on demonstrations or face-to-face tests to provide additional evidence to be submitted to the RMU Academic Integrity Board.***

## How to complete Content Questions

Review questions are also be provided at the end of the tutorial. The following is an example of a review question format. Since type the answer in provided grey or colored box.

1. What is the purpose of a partitioned data set? Answer:

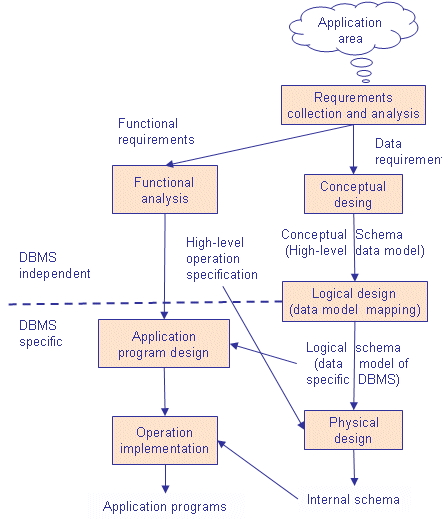
Type in the answer to the question into the grey or colored box.

**It is recommended that you use Table of Contents at the beginning of the tutorial to review and navigate to the concept presented in the review question. Students will find that using the document FIND tool or searching GOOGLE may also be valuable for researching the review question answer.**

# 1.0 Introduction to Database Design

The following reading is important to answer the questions in Requirement 1.

While a database may store vast amounts of data, they significantly differ from flat tables in the both the manner in which the data is organized and designed. While issues such as: referential and transactional integrity, concurrency, data recovery and data independence user interfaces are important features of databases, the most critical determinant of a data base's success is how we organize, store and access the data.

In a system analysis and design course, you are taught that any new project starts with a project request, which is important to success, continuity and operation of an organization. The project request is followed by a feasibility analysis which analyzes the economic (cost-benefit analysis), operational (implementation), technical, and schedule to determine if the project request should proceed. After project selection, the first step of a project is Requirement Collection and Analysis. Requirement Analysis emphasizes functional (what procedures are to be performed - the verbs) and data (what data and information is to be processed and stored - nouns) requirements of the project.

In an entry-level programming course, the Functional Requirements will be converted into program specifications. Normally the instructor will provide program specifications for each assignment and there would be no need for data and database design. The only program data requirements would be to list the variables and data types of variables.

This review will emphasize data or database design based on the data requirements, which may accessible to multiple or hundreds of programs for the given application area. Examples of application areas may include employee maintenance, payroll, customer maintenance, sales and invoicing, production scheduling, job costing, account receivable, etc.

In the real world one cannot separate functional analysis and design from database analysis and design. But, in education it is important to separate these concepts in programming and database courses so that we do not overwhelm students. Therefore, the following discussion will ignore functional design, application program design (specifications) and program coding, testing, and implementation.

## 1.1 Conceptual design

***Conceptual design* organizes the data requirements at a theoretical level which we can use for any type of database, data file or dataset.** We will use concepts like entities, attributes, domains, and constraints. Normalization, Bachman and ERD diagrams are important to conceptual design. But, while each of these concepts are important and common to all storage systems, they are theoretical and do not directly apply to real world databases. Since the concepts of entities, attributes, domains, and constraints are common, it does no matter the type of database, e.g., hierarchical, network, or relational, or what type of dataset, e.g., sequential, relative, index, etc., is physically used Normalization, Bachman and ERD diagrams will be the starting point.

## 1.2 Logical design

***Logical design* maps (converts) the conceptual terminology into the concepts used by the appropriate DBMS to be used to implement the database.** In conceptual design we express concepts, such as, entities, attributes, domains, and constraints, however, in a relational physical DBMS like Oracle we use implementation concepts such as tables, columns, data types and constraints. ***Therefore, an entity is a table, an attribute is a column, a domain is similar to a data type, a constraint can be a primary key, foreign key or check constraint, and so forth***. But, in a flat file system, an entity is a data file, an attribute is a data field, and so forth.

Physical design will use the same terminology and concepts of the logical design, except there are some adjustments that must be made, often due to the limitation of the DBMS or a flat file system. For example, not all DBMSs provide for referential integrity or foreign keys. Not all DBMSs provide check constraints. While these conceptual design requirements specify referential integrity and data validations, some limitations of DBMS will mean that the program application must enforce these conceptual design requirements through program code rather than the DBMS or flat file system. With a relational DBMS, physical design means that follow the SQL syntax and naming rules of the physical data base to be used. While Oracle, DB2, MySQL, and SQL Server use SQL, approximately 5-10% of the SQL statements are unique to the specific relational DBMS.

## 1.3 Bottom-up, Top-down, or Combination Design Approaches

There are three strategies that may be used start designing a data base Bottom-up, Top-down, or Combination. Since the results of data requirement analysis for a project may be real-world and practical, users, business and system analysts prefer using a bottom-up approach. ***The bottom-up approach starts by listing the detailed data requirements, and then attempts to organize the detailed data into entities, tables or data files. A bottom-up approach will start by identifying the columns or attributes that need to be stored, and then organize the columns into tables later.***

Business users and analyst are comfortable with the day-to-day data and procedural details of their job or proposed project. Users and business analysts will start with data requirements such as information to be displayed for an invoice, or information to be displayed for a report, other documents, web page design or so forth. But, these bottom-up details may distract the database designer from properly applying normalization, selecting primary keys, determining data relationships, and so forth. An employee's name may be used in hundreds of reports and documents, but the rules of normalization requires that the employee's name should only be stored once.

For example, consider the logical relationship between a student and course for which they are registered. A student has an address and a tuition balance. A course has an instructor, room number, date and time of the course. It is important for student financial services to store detailed data to send the student a bill and for a student to know the course location, time and name of the instructor. But, when we normalize by applying Codd's rules of normalization a database design, it is more important for the data base designer to understand that to understand the role of conceptual, logical and physically implementation fundamentals of data base design, before specifying the detail data required for a user or a business analysis.

***A top-down design approach will start first by identifying tables before specifying all data details of columns to be stored in a table.*** Most experience database designers will use a combination approach which emphasizes tables, but also includes consideration of some major columns. Identifying major columns is important to select primary keys, evaluating relationships between entities, and designing indexes.

## 1.4 Overview of the Steps Recommended for Database Design

1**. Table Analysis** - Identify and list the tableor entity candidates.

2. **Relationship Analysis** - Determine the ***relationship*** between the tables, e.g., one-to-one, one-to-many (most likely), and many-to-many. (This is a part of the normalization process.)

3. **Primary Key Analysis** -Identify the primary key column(s) (unique identifier) of a table row, e.g., potential primary key constraint. Most tables will have a primary key column already in common use. If not just use a, auto increment column. A table can only have one primary key. (This is a part of the normalization process.)

4**. Non-Key Column Analysis** - Start identifying a sample of non-key columns (other than the primary key). It is not recommended to identify all columns. Write a good, clear description of the data to be stored in the table to guide this process.

5. **Normalization** - Apply the normalization process.

6. **Referential Integrity Analysis -** Identify relationships which require referential integrity, or primary/foreign keys.

7. **Data Type Analysis** - Identify the appropriate data types of columns.

8. **Check Constraint Analysis** - Identify the columns which have limited value to be stored.

9. **Index Analysis.** Identify columns which are foreign keys, or will often be used in a WHERE, ORDER BY, or GROUP BY clauses.

## 1.5 How does one determine an entity or table?

If you inherit an existing data base or any data storage system, the selection of tables may be obvious. However, if you are designing a new database, identifying tables will require a common sense, proper English skills and a little experience. As stated previously, data is NOUNs and programs are VERBs. **Therefore, a table is a major noun and a column is a minor noun.**

Practically speaking, how do you determine a MAJOR NOUN when conducting a functional or data requirement analysis? Typically, this is a NOUN that is mention most often during an interview or requirement analysis. Using the example of the student and the course, you will see references such as "the student does this…", the student need this …", the student…, the student…, etc. The issue of student name may be mentioned once or twice, but the concept of a student will be mention over-and-over again. Think of the concept of a student, or student table as a storage bucket that will contain detailed information about a student.

In this course's assignments, simply start by reading a case or system description and circle or list your candidates on a piece of paper. Even though a business report or document may be major nouns and often repeated, rarely will these be candidates for a possible table or entity. Normally, the role of a business report or document is to combine tables and columns in a manner that will produce information.

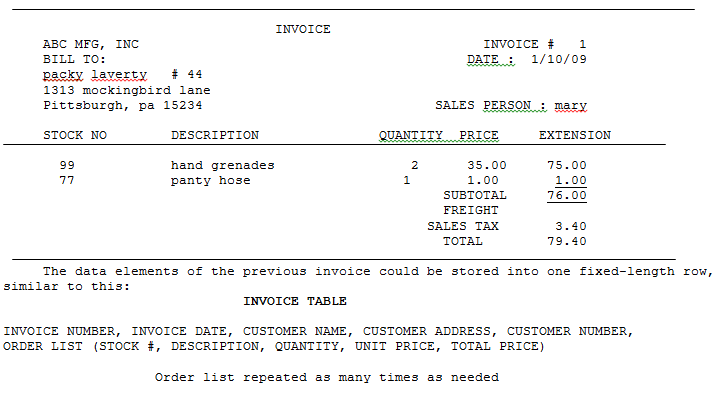
With experience, one will learn that there a few new and innovative database designs. Every design requirement in the course or the real world, Google will have a database design that will serve as a starting point to identify tables. In the real world, most systems have the same tables. What is different between system-to-system, or database-to-database, is the column names of a table. With a little creativity in using a Google search, you will always find table names appropriate to your requirements.

Today's database management software has become very user-friendly, e.g., MySQL, SQL Server ACCESS, ORACLE, and DB2. An Access book or reference provides excellent examples on how easy it is to manipulate the data. But rarely will these database references provide user-friendly guidance to practically design a database to solve business's problems, other than three sample databases. It seems that Microsoft of other Database Providers want you to believe that the infamous "stork" gave you the database designs. Of course we know better "all database designs are born in the cabbage patch!" Database references and text books may teach you how to easily create and use a database, but provides little information to design a database.

The following presents three major discussions of data base design: 1) Normalization, 2) Index Analysis, and 3) Data relationships.

# 2.0 Normalizing a Database

Normalization is a technique that is used when designing and redesigning a database. The normalization process identifies 1) redundant data that might exist in a logical data structure, 2) determines unique keys needed to access data items, and 3) helps establish necessary relationships between data entities. Consider the following invoice to be stored in a database:



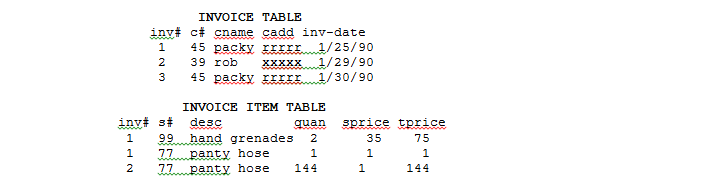
## 2.1 Rule 1 -Separate all repeating fields and store these repeating fields into a separate table.

**The first step in normalizing a database is to separate all repeating fields and store these repeating fields into a separate table**.

The INVOICE HEADER TABLE store one-time information of an invoice and the INVOICE ITEM TABLE will contain one row, which repeats for every line of the invoice.

This design is superior to using one fixed-length row used to store ten (10) invoice line items. For example, if an invoice only stores only one invoice item, then using a fixed-length row will waste storage space for the remaining nine invoice unused items. At the other extreme, a user would be unable to store an invoice with more than ten items. Separating repeating fields into a separate table permits you to store as many invoice line item rows as you need.

Notice that the invoice number is stored and repeated in both of the INVOICE HEADER and the INVOICE ITEM TABLE. **This duplication of conceptually identical columns or fields is frequently called a table interrelationship**. If we know the invoice number to access one table, we can you that same invoice number to access the information stored in the second table. Therefore, information stored in the invoice header and invoice item tables are related by the invoice number. (More discussion concerning table interrelationships will follow.) Below is a data base design illustrating the application of the first normalizing rule. Sample data has been provided.



### 2.1.2 Primary Key and Non-Key Columns

Before we discuss the second step used to normalize a database, we should clarify the concepts of a primary key and non-key columns. **A primary key column or field is used to uniquely identify a given entity.** The invoice number can be used to uniquely identify the invoice header. There is only one invoice # 1 and that invoice represented a transaction with customer packy of the specified date. Therefore, the column or field invoice number would be a good choice for the primary key for the invoice header row.

Consider the choice of a primary key for the invoice item row. Would STOCK # be a good primary key for the invoice item row? No! Consider the Non-Key columns of the INVOPICE ITEM TABLE. Non-key columns are any other columns besides the candidate primary key column STOCK#. Therefore, the Non-Key columns of the INVOICE ITEM TABLE are s# (Stock Number), desc(Description), quan (Quantity Sold), sp (List Price) and tprice (Total Price (quan times sp)).

The choice of the STOCK # column may uniquely identify the non-key fields such as description and list price of an inventory item, but not quantity sold on an invoice. Customer number 45 purchased 1 panty hose on invoice 1 and Customer number 39 purchased a gross of panty hose on invoice 2. Furthermore, if the stock # was to be unique this would mean that stock #77 could be stored only once in the invoice-item table. Therefore, we could only sell panty hose only once for any future invoice.

On the other hand, is the choice of the INVOICE NUMBER a good primary key for the invoice-item table? Again the answer is no. **This would mean that only one line item could be stored per invoice. As you can see invoice # 1 has two line items: hand grenades and panty hose.**

### 2.1.2 Composite Primary Keys and Atomic Primary Key

Now let's propose a composite key field, such as: INVOICE NUMBER + STOCK # to uniquely identify the row. The use of a one field to identify a row is called an **atomic key**. The use of multiple fields to describe a row is called a **composite key**.

As you can see, the key "invoice number + stock #" is a good choice in the above example, but it does create other limitations. For example, you can only sell panty hose once on an invoice. Considering the real life problems such as quantity discounts, panty hose may appear twice on the same invoice at two different prices.

### 2.1.3 Limitations of Composite Primary Key

Using the existing data fields as a composite key may not provide the answer to uniquely identifying a row. Sometimes we might have to make up our own key field. In my applications I have identified an invoice item row with the composite key invoice number + invoice line number. In other situations assigning an arbitrary unique reference number may be the only alternative.

Some databases do not require a primary key. But basic principles of internal control state that you should always design an audit trail. And to do this you should always design a unique identifier.

### 2.1.4 Natural Primary Keys

**A natural key is a unique identifier that is has been used or is commonly used to identify a specific item or event.** Examples include Social Security numbers, stock numbers, customer ids, and so forth. Some database designers complain that natural key may use more storage space than an integer auto-increment inter key, but this issue may be is overblown. If there's a good, useable natural key available, like a stock number, then use it! You are going to I'm going to store the natural key data in the table anyway.

On the other hand, if there is no good, obvious natural key, don't try to cobble together a crazy one. I've seen people try to make a natural key by concatenating together the first 6 letters of the customer's first name, his year of birth, and his zip code, and then pray that that will be unique. That sort of silliness is just making trouble for yourself. Often people end up taking on a sequence number to insure it's unique anyway, and at that point, why bother? Why not just use the sequence number by itself as the key?

### 2.1.5 Advantages of an Auto Increment Primary Key

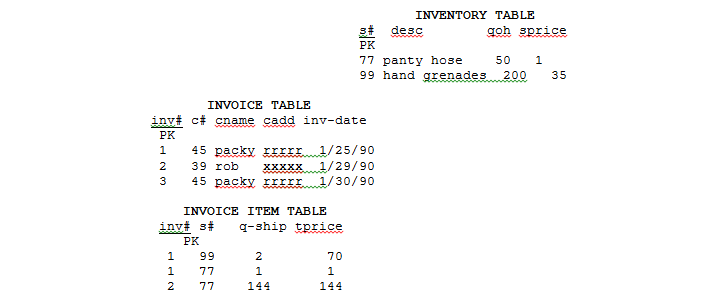
An auto-incremented numeric primary key is often used in transaction or journal type tables. An auto-incremented invoice number, journal number, reference number, confirmation number, etc. saves data entry time, guarantees uniqueness, but also provides for data entry sequence order that can be used to identify missing rows. For example, consider the invoice number sequence of 105, 106, 108, and 109. What happened to invoice number 107? You should never delete an invoice row. You may mark the invoice with a special status field, or store a separate invoice return row, or use an invoice return or cancel table

## 2.2 Rule 2 - All non-key fields should be functionally dependent upon the "entire" primary key.

**The second normalizing rule states that all non-key fields should be functionally dependent upon the "entire" primary key.**

Once we have selected an appropriate primary key for a given row we should determine if the non-key fields (description, quantity, price and extended price) are dependent upon the entire primary key. For example, stock # 77 identifies a panty hose. But does invoice # 1 (the invoice number is a part of the composite primary key) requires this invoice line item to be a panty hose? The answer is obviously no. Look for yourself. You can see the potential for repeating the description field "panty hose" many times since panty hose appears on multiple invoices, i.e., invoice #1 and # 2.

Therefore, description is not functionally dependent on both the invoice number and the stock number. Information involving the stock item should be placed in an inventory table like the one below. The primary key of the inventory table will be the stock number. Review the new design.



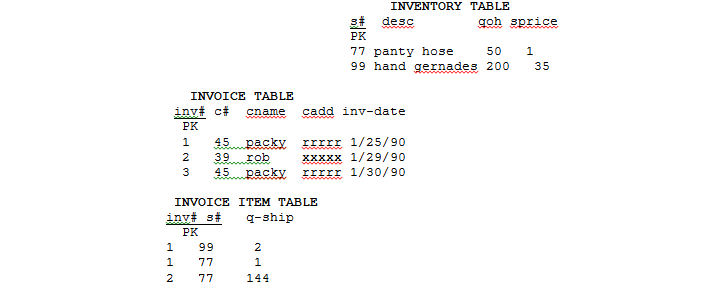
After reviewing the above database design, there are two other issues to be considered. First, notice that quantity column had to be clearly named. Database design doesn't permit you the flexibility of being ambiguous when naming columns fields when they pertain to two or more different concepts. In the inventory table, the data element "quantity" represents the quantity on hand. On the other hand, in the invoice item table the data element "quantity" represents the quantity shipped. Merely naming each field "quantity" would clearly confuse the two different meanings of the word quantity.

The second issue involves the use of the list price. In this design I assumed that the list price was the same for all customers and that the list price did not change. If we had assumed that the list price would change or that each customer would be charged a different price we should have stored the price as the shipped price in the invoice item row. We will use this approach in later example.

## 2.3 Rule 3 - Eliminate any non-key fields that can be created, calculated or derived by a combination of the other non-key fields, within the row or between rows.

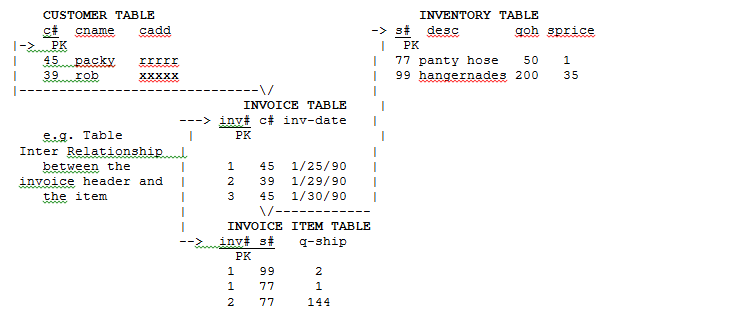
**The third step in normalizing a database is to eliminate any non-key fields that can be created, calculated or derived by a combination of the other non-key fields, within the row or between rows**.

For example: if you know the quantity to be ordered from the invoice item table and the list price of the item from the inventory table, you could calculate the total price of the invoice line item. Likewise, a total invoice price stored in the invoice header table would be redundant because we can quickly add the total price from each repeating invoice items. The fact that the calculation requires access to two different tables does not change the rule. Review the new design.



The previous example is still not completely normalized. Can you determine which of the three rules were violated? What should be the new design? The answer follows.

## 2.4 A Four-Table Normalized Design to Store a Customer's Invoice



## 2.5 Bachman Diagrams

The above diagram is called a Bachman diagram. The purpose of a Bachman diagram is to document a database design. A Bachman diagram will:

1. Display the major tables and a few columns of a database

2. Describe the inter-table relationships between the tables. These inter-table relationships depict logical design paths that programmers will use to access the data base rows in order to share data. In this example, the interrelationship will produce an invoice. The table interrelationship is a generic design concept. How they are implemented is dependent upon the database model that you are using. In a network data base table interrelationships are called sets. In a hierarchical data base they may be implemented as external or logical databases. In a relational database they be called foreign keys or even organized into a view. In a relational data base, these interrelationship paths are used by the SQL JOIN commands

3. Depicts the columns which are used as primary and indexes.

A Bachman diagram is not concerned with the non-key data. Though the following example will frequently display non-key fields to help the student understand the design in relationship of a real world output.

## 2.6 Pros of a Normalized or De-normalized Database Design

Pros and Cons of a Normalized database design.- http://www.ovaistariq.net/199/databases-normalization-or-denormalization-which-is-the-better-technique/#.WRcXYNy1vcs

Advantages & Disadvantages of Normalization - <http://sqlserverwithbi.blogspot.com/2012/10/advantages-disadvantages-of.html>

Advantages & Disadvantages of Normalizing a Database - https://www.techwalla.com/articles/advantages-disadvantages-of-normalizing-a-database

### 2.6.1 Pros and Cons of a Normalized Database Design

1. Smaller database: By eliminating duplicate data, you will be able to reduce the overall size of the database.

2. Free from update and delete anomalies since duplicates are not allowed

2. Better transaction performance for inserts and updates are very fast because the data to be updated is located at a single table, but slower performance for complicated Select queries.

3. Allows tables to have less columns and allows you to fit more records (rows) per virtual data page of data or buffer.

4. Fewer indexes per table mean faster maintenance tasks such as index rebuilds.

5. Only join tables that you need, but to provide meaningful and complex query information this may more overhead. By spreading out your data into more tables, you increase the need to join tables.

6. Tables contain codes instead of real data: Repeated data is stored as odes codes rather than meaningful data, e.g., employee class "M1" instead of "management Level 1 Class". . Therefore, there is always a need to go to the lookup table for the value.

Although most successful databases are normalized to some degree, there is one substantial drawback of a normalized database: reduced database performance for complicate Select Queries. The acceptance of reduced performance requires the knowledge that when a query or transaction request is sent to the database, there are factors involved, such as CPU usage, memory usage, and input/output (I/O). To make a long story short, a normalized database requires much more CPU, memory, and I/O to process transactions and database queries than does a de-normalized database. This is especially true of SQL Joins and complex queries with require mathematical calculations involving multiple columns of one or more tables.

### 2.6.2 Pros and Cons of De-normalized Database Design

1. The data is present in the same table so there is no need for any joins, hence the selects are very fast, but slower performance for Inserts and Updates.

2. A single table with all the required data allows much more efficient index usage. If the columns are indexed properly, then results can be filtered and sorted by utilizing the same index. While in the case of a normalized table, since the data would be spread out in different tables, this would not be possible.

3. Because the data is duplicated, the updates and inserts become complex and costly, which increase the probability of update and insertion data anomalies.

4. More indexes require more overhead and index storage space

De-normalization is the process of taking a normalized database and modifying table structures to allow controlled redundancy for increased database performance. Attempting to improve performance is the only reason to ever de-normalize a database.

Data redundancy is increased in a de-normalized database, which can improve performance but requires more extraneous efforts to keep track of related data. Insertion and deletion anomalies will increase

Application coding renders more complications, because the data has been spread across various tables and may be more difficult to locate. In addition, referential integrity is more of a chore; related data has been divided among a number of tables. There is a happy medium in both normalization and de-normalization, but both require a thorough knowledge of the actual data and the specific business requirements of the pertinent company.

## 2.7 Comparing Transactional Database Storage Design to Data Warehouse Design (OLTP to OLAP)

OLTP stands for On Line Transaction Processing and is a data modeling approach typically used to facilitate and manage usual business applications. Most of applications you see and use are OLTP based. **Normalized database designs are very common in OLTP databases**

OLAP stands for On Line Analytic Processing and is an approach to answer multi-dimensional queries which are commonly used in data analytics and data mining applications. **De-normalize database designs are very common in OLAP data warehouse and data mart designs. The most popular design data warehouse design model is called the Star Schema**

Transaction databases, data warehouses and data marts are all based on an underlying relational database like Oracle and DB2. The major difference is the how they are designed, e.g., normalized or de-normalized. Transactional databases require processing speed and storage efficiency. Data warehouse and data marts databases are designed to answer management questions.

Given these different design requirements, many organizations support both OLTP and OLAP databases. OLTP databases are designed for real-time (immediate) processing. The normalized data stored in a OLTP database is periodically copied and transformed into a de-normalized design and loaded into a LOAP databases. This process is called ETL, e.g., extraction, transformation ad loaded. **A batch process may perform an ETL on a daily or hourly schedule.** Or database triggers, may automatically perform the ETL process on a transaction by transaction basis rather than using periodic batch processing.

The problem on this design is that two, differently designed relational databases are maintained. This requires substantially greater storage and administration requirements. In 2015, IBM designed the new z.13 mainframe processor and instruction set that permit real time predictive data analytics and data mining "in-place" with limited ETL requirements. This means that it is more efficient to store data in one relational database, than maintaining two normalized and de-normalized databases and using ETL. This major advance in technology is expected to change this process for large 500 corporations over the next decade.

## 2.8 Descriptive vs. Predictive vs. Prescriptive Data Analytics

The majority of raw data, particularly big data, doesn't offer a lot of value in its unprocessed state. Of course, by applying the right set of tools, we can pull powerful insights from this stockpile of bits. With data in hand, you can begin doing analytics. But where do you begin? And which type of analytics is most appropriate for your big data environment?

**Descriptive analytics** is "the simplest class of analytics," that allows you to condense big data into smaller, more useful nuggets of information. The purpose of descriptive analytics is to summarize. 80% of business analytics -- most notably social network analytics -- are descriptive. For example, number of posts, mentions, fans, followers, page views, kudos, +1s, check-ins, pins, etc. There are literally thousands of these metrics -- it's pointless to list them -- but they are all just simple event counters" Descriptive analytics may have limited value. Limitations include, 1) one does not access to key data,20 overly summarized, 2) the data is old, 3) not timely, 4) includes duplicates or evil twins from multiple sources, 5) may have inaccurate sources, and 6) does not include appropriate time-space or data origin identifier, that may be required to make good decisions. Most descriptive analytics are generated by batch processing and are more often used to search for marketing, product or customer opportunities.

**Predictive analytics** is the next step up in data reduction. It utilizes a variety of statistical, modeling, data mining, and machine learning techniques to study recent and historical data, thereby allowing analysts to make predictions about the future. The purpose of predictive analytics is NOT to tell you what will happen in the future, Predictive analytics is an emerging technology and most often using the same data sources as descriptive analytics. However, these sources are often inadequate. Therefore, a problem of predictive analytics is that you basically take data that you have to predict data you don't have,

**Prescriptive analytics** is a type of predictive analytics. It's basically when we need to prescribe an action, so the business decision-maker can take this information and act.

## 2.9 Terrorist Attacks: A Comparison of Descriptive, Predictive and Prescriptive Analytics

One outcome of the data analysis of 9-11 was the discovery that traditional government agency OLTP data warehouses were overly summarized and did not include data that could be used to provide insight to understand the cause of the U.S government failures to prevent the attack before 9-11. These OLTP warehouses were designed for descriptive analytics and were not robust enough to historical predictive insights.

Since 9-11 the design and data collection used for U.S. Government Terrorist data has been improved by adding more data sources and increasing the granularity of data to predict a future terrorist attacks. While the results of this data warehouse redesign continues to be improved to predicted WHY something has happen, predictive analysis is limited to prevent a terrorist attack. As stated previously has stated, Prescriptive Analytics requires a data warehouse that may warn that the terrorist attack may occur as new real time data is added to the data warehouse.

The recent revelations by Edward Snowden has made us aware that the U.S. Government may have access to our phone conversations. Storing actual phone calls may provide detail data to predict a future attack. However, at best an extremely number of phone numbers conversation contents may be analyzed for key words which will require further analysis. However, the results of San Bernardino and Orlando investigations proved that these terrorist's phone conversations were not monitored in real time and at best these phone conversations were reviewed to provide forensic information only after the terrorist's attack to predict the cause of attacks.

Dig Data means Big Data. The more predictive the data may be, the Bigger the Big Data will be. More importantly, Big Data implies greater processing time. For example, it takes NASA researchers on average of 1.34 years of mainframe processing time to predict the collision of two asteroids from data collected over several years from the Hubble Telescope. One would wonder, how many CPU hours would take to analyze my daughter's cell phone conversations over the last three years. The reality is no one is analyzing your phone call in real time. Do something really, really bad, the NSA and FBI will spend their limited processing budget to investigate you.

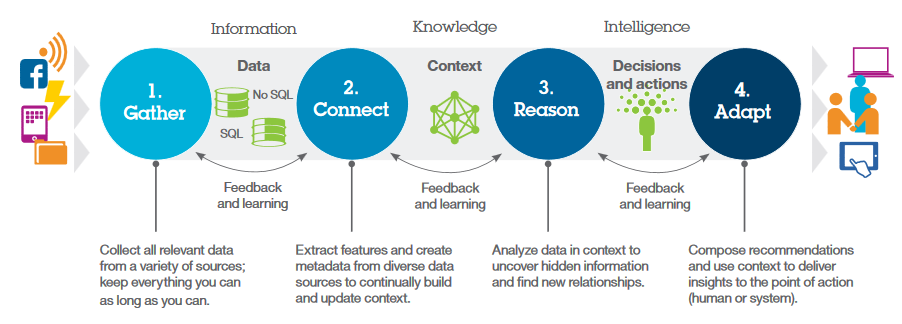
In its 2012 Digital Universe study, IDC projected that the digital universe will reach 40 zettabytes (ZB) by 2020. To put this number into perspective, consider that 40 ZB is equal to 57 times the amount of all the grains of sand on all the beaches on earth. Another visual: If you saved all 40 ZB onto today’s Blu-ray Discs, the weight of those discs would be equivalent to 424 Nimitz-class aircraft carriers.

The problem is that there is too much data to determine the meaning of data in order to make real time decisions to take actions. There are NO DATA WAREHOUSE AVAILABLE OR ADEQUATE CPU PROCESSING TIME that are available to use Prescriptive Analysis and establish rules to make a decision.

## 2.10 Jeff Jonas (IBM researcher) and Contextual Data Processing Models

Jeff Jonas (IBM researcher) has designed a Contextual Computing Schema based on the IBM z/13 Platform Processing Model (2016) that is designed for National Defense. If you want to learn more about this area I highly recommend that consult the following sources. I met Jeff Jonas in June 2016 and his insights are extremely interesting and hilarious.

* **Video - Big Data, New Physics, and Geospatial Super-Food - https://www.youtube.com/watch?v=8qubbhcEPJI**
* **Big Data. New Physics. - http://www.slideshare.net/mitforum/jeff-jonas-big-data-new-physics**
* **Empowering governments through contextual computing -http://www-935.ibm.com/services/multimedia/GBE03597USEN.pdf**



Popular data design models include relational, hierarchical, network and star. These designs do not have any context or meaning without substantial processing at a later time.

Jonas is developing a Prescriptive Data Model to support real time data analytics based on how we make decisions to put together a picture puzzle. Assume that you are provide a 200 piece puzzle, but you do not know what the finished puzzle looks like. Each piece of puzzle represents small container of information that relates to a given entity or finished puzzle. In previous discussions and entity was a person, transaction, place, time or event. As the piece of data is received it is individually analyzed to assign or relate to one or more entities. As Jonas indicates we simply provide context or meaning to the raw data.

As we put together a puzzle, it is easy to connect a puzzle pieces at the beginning of the assembly. As we receive more data, increasing processing or assembly time is required to complete the puzzle. However, as the image of the puzzle becomes obvious the amount of processing or assembly time dramatically decreases and we have enough context of the puzzle image and to make prescriptive decisions.

Compare this model to the Wheel of Fortune TV Show game. At some point the contestants can buy a vowel and accurately guess the phrase because they have enough contextual information to make a decision. Likewise, Jonas Contextual Data Model will permitted future data scientists to predict and take actions to prevent terrorist's attacks. As future data becomes available the model supports minimal processing power and timely requirements which may make reduce the cost of the preventing terror attacks comparable to "buying a vowel".

# 3.0 Index Analysis and Design

## 3.1 Advantages of Database Indexes in Conceptual Design

The use of a database index is optional in conceptual and physical design. The main reasons for using a database index are improved performance of queries and reduced response time. Some of us prefer a high-performance sports car. But, indexes are similar to a high-performance sport car, and indexes do have their disadvantages and higher cost.

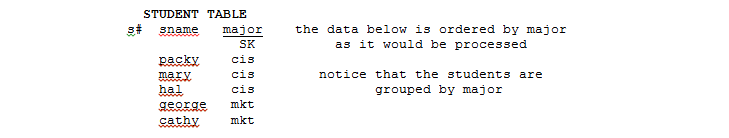
The following list four design objectives which indicate that creating database indexes, e.g., CREATE INDEX, maybe more beneficial than the increase of costs associated with the index.

**The first objective is to provide an alternate access to the row**. For example: How many people remember their customer number. It would be desirable to look up a customer by either their customer number or by their name. The same issue can be used to make the description in the inventory table an index. ***Columns that frequently used to search for a row by using the WHERE clause are a good candidate for an index.***

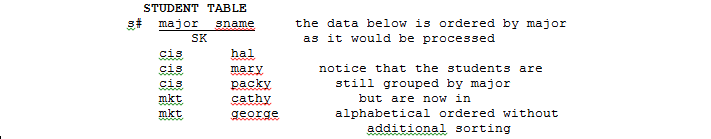
In order to make inventory description searches effective in the real world we would have to control the manner in which the users assign a description. For example, if we knew that a customer's name began with "LAV" it would be easy to search and read the next row until we found the correct row. However, if the current description system began with "4 X 10" in a hardware store this prefix could be used for lumber, bolts, pipes and so forth. Therefore, the description system would have to be better organized to begin with "Lumber, 4 X 10."

**The second objective for using an index is to provide an alternative dynamic order to the table**. As we add or delete rows to the customer table we can quickly retrieve a customer name order listing of the customer table, online without sorting. The same argument could be applied to the description index. ***Columns that frequently used to order selected rows by using the ORDER BY are a good candidate for an index.***

**The third objective for using an index is to group like data together**. Let's change the invoice example to a student table that would contain the fields below. In this example, the major would be designed as an index. Rarely would we look up a student by major. But there will be times that the chairman of the CIS department would want a quick listing of the activity of all their majors. Perhaps the CIS chairman is planning next semester's course offerings. They could access the data by the "major" index to produce their report. ***Columns that frequently used to provide categorical analysis of selected rows by using the GROUP BY are a good candidate for an index.***



There is a design problem illustrated by the previous example. While the data is grouped by major there is nothing that ensures that within a major the student's names would be alphabetically ordered. Using a composite index would correct this problem. In most databases the composite key fields do not have to be contiguous (side by side) data fields in the rows. However, Contiguous columns for this example, e.g., CREATE INDEX MYINDEX ON STUDENT(MAJOR, SNAME). A Composite Index normally indicates that the combination of columns will improved categorical analysis of queries



**The four objective for using an index is to require uniqueness (no duplicates) for column(s) other than the column(s) selected for the Primary Key.** .

Any table may have only ONE primary key. For example, an EMPLOYEE table may use the EMPLOYEE-NUMBER as a primary key. The EMPLOYEE-NUMBER is often displayed in reports rather than the employee's social security number. By designing a UNQUE INDEX of the social security number column, no one will be able to store duplicate social security numbers.

## 3.1 Disadvantages or Costs of Indexes

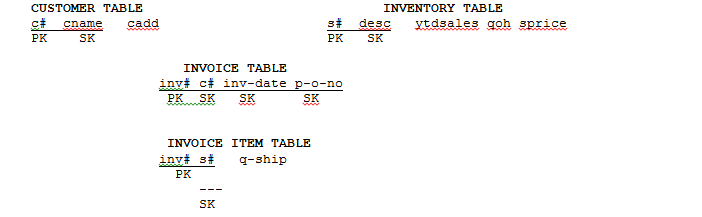
While indexes seem extremely useful they do have their cost. **Every index creates a separate data structure, and hence uses more disk space.**  In addition, each index slows the performance for each INSERT, UPDATE, or DELETE operation. Not only does the base table has to be updated, but also each index structure needs to be updated as well.

On one side companies excessively use the COBOL SORT verb and utility sorts. Sorting a table may be a partial alternative for an index. With this processing strategy the order of the data will be only as current as the last sort. Many of the online, real time powers are missing. On the other side, other users declare every column as an index (they say you never know when you might need it).

**In the final analysis, designing index should take in consideration the increased costs of storage and performance costs of the index, as compared to the information value, the response time and the frequency of use of that index.**

## 3.3 Applying Index Design to a Customer Invoice Database

Below is a Bachman diagram with additional indexes. Following is a discussion of each index and its information value



### 3.3.1 Stock Number Index in the Invoice Item Table

Designing an index for a stock number stored in the invoice item table will group all the sales together of a given stock number. An alternative to this design strategy is to store YTD sales in the inventory row. That would be acceptable but would not provide us with as much valuable information. Using table interrelationships one can determine not only how many panty hose were sold but also:

1. Who purchased panty hose? The invoice number of the invoice item table can be used to retrieve information stored the invoice header table. How much quantity, i.e., q-ship, did the customer buy and on which date did that customer purchased it, i.e., inv-date? The customer number, i.e., c#, stored in the invoice header table will provide a link to a customer's information?

2. On which date, i.e., inv-date, were items shipped for any customer? Is there a seasonal pattern of sales for this stock number?

### 3.3.2 Invoice Date Index in the Invoice Header

Designing an index for the invoice date, i.e., inv-date, would order invoices by date for sales to any customer. This, as the accountants would say "would give appropriate cut off for a given month's sales." In other words, producing a report in the by the order of the invoice date key would be produce a sales journal.

### 3.3.3 Purchase Order Number Index in the Invoice Header

Designing an index for the purchase order number would permit one to match, look up or cross-reference any of our invoices that satisfied a particular customer's purchase order number. Customers don't care about our invoice numbers. They only care if their purchase orders have been satisfied.

### 3.3.4 Customer Number Index in the Invoice Header Table

Designing an index for the customer number, i.e., c#, in the invoice header table would all a customer's invoices together. Without this key one would have to sequentially search all invoices (maybe thousands) a few invoices for a given customer. This index design strategy will provide detail information concerning which stock numbers and quantities the customer purchased and the last price that they were charged for a given item.

Using the invoice date one can analyze sales for a given customer by the date by which they purchase the given item. Below is a table that contains sample data and illustrates the three different methods of ordering or grouping the invoice header. Notice that when the invoice header is ordered by customer number there is no guarantee the invoice dates will be ordered, e.g., invoice numbers 52 and 52.

To ensure that invoices are ordered by date, when first group by customer then the index must be designed as a composite key: c# + inv-date. This improved key will be illustrated in a later discussion.

|  |  |  |
| --- | --- | --- |
| Invoice Header  ordered by Invoice Number | Invoice Header  ordered by Invoice Date | Invoice Header  Ordered by Customer Number |
| 50 99 1/3/98 xyz  51 77 1/3/98 fff  52 99 1/5/98 fgf  53 99 1/4/98 dfd  54 77 1/9/98 sdd  55 99 1/8/98 sds | 50 99 1/3/96 xyz  51 77 1/3/98 fff  53 99 1/4/98 dfd  52 99 1/5/98 fgf  54 77 1/9/98 sdd  55 99 1/8/98 sds | 51 77 1/3/98 fff  54 77 1/9/98 sdd  50 99 1/3/98 xyz  52 99 1/5/98 fgf  53 99 1/4/98 dfd  55 99 1/8/98 sds |

# 4.0 Data Relationships

A series of interrelated tables that is designed to minimize duplicated data is one the most frequently used definitions (though inadequate in my opinion) to describe a database. The manner in which these tables are interrelated will be described as:

**\* ONE TO ONE RELATIONSHIPS (1:1)**

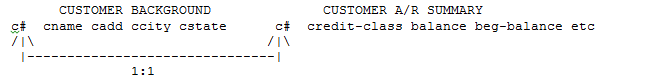
**\* ONE TO MANY RELATIONSHIPS (1:N)**

**\* MANY TO MANY RELATIONSHIPS (N:M)**

## 4.1. One-To-One Relationships

A 1:1 relationship is characterized by storing one row in a table that is functionally related to one row in another separate table. Let's begin this discussion with a bad example of a 1:1 relationship that frequently is used in industry to cover up original design mistakes.

Given the following two tables



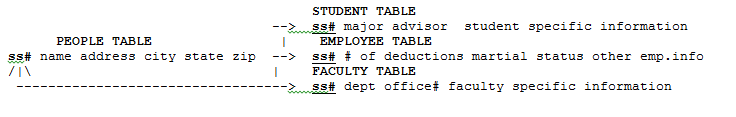
For every customer background row there is one customer a/r summary row. If all the non-key fields are functionally dependent upon the customer number why should we separate the information into two different tables? This may require more seek time to process the information and it may increase the number of file handles required by the operating system.

Many companies will begin designing a row like the customer background row and find later that they may need to increase the row length. Instead of expanding the length of the original row they may choose to store the new information in a second table and design the key of the new table as the same primary key of the original table. Beware of any two tables or tables that have the same primary key. Serious consideration should be given to combining these tables.

There may be other design issues that may still overrule my objections to this 1:1 design that may be more important. In some systems that perform row locking and security by tables, there may be reasons to segregate the information into two different tables. For example, you may wish all employees to view customer background information, while only designated employees could view the summarized a/r table. However, DB2 for example, can use a view to limit access to certain data fields.

Turning to an appropriate application of a 1:1 relationship let's begin with the following scenario. A student at a college may have his/her name, address, city, state, zip code stored redundantly in several different tables. This information may be stored in the prospective student table, the new registration table, the current student table, the employee table, the faculty table and the alumni table (assuming you survive this course). While it is already obvious that disk space may be wasted due to redundant data, we need to consider the procedural implications of this redundancy.

"Don't you ever move or get married and change your name or address if a database is designed as described above." Each duplicated table must be updated. Frequently, one table is updated by one program, but the other tables will updated by a different program. For example, the human resource program may update the employee table, but not the student table for a change of address. If that employee is also a student they may never get their grades. Therefore, let's give you a good 1:1 solution.



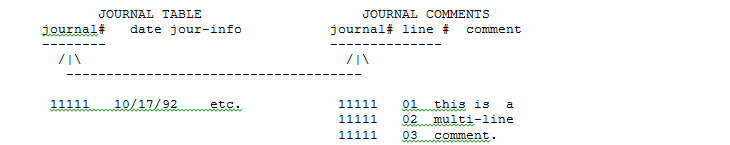
The people background information will only be stored in one table, the people table. The people table will contain common information that is not related to a particular specific status of an individual. Other specific information related to a particular status, e.g., faculty member, student, etc., will be stored in individual information tables specifically designed to store student, employee or faculty information. If person's address changes, then only the people table will be updated. If the person is a faculty member then a row will be stored in the faculty table. If the person is both a faculty member and a student, then a row will be stored in both tables. If you want to process student, you would first process all the rows in the student table, and then retrieve any necessary background information from the people table.

If you want to know if Mary Smith is a faculty member, all you have to do is inquire the faculty table. If you find a row (successful inquiry), then Mary is a faculty member. If you don't find a row (row not found), then Mary is not a faculty member. Notice that the "row not found" condition is an important tool in this design.

Some may argue that some type of people-type field should be used. For example, a people-type of "F" is a faculty member; an "S" is a student and so forth. In some situations, this coding scheme might be useful. But, how would design a code for a person who is both a student and a faculty member. What about coding a faculty member, alumni and student? How do we add a new people-type, such as children of faculty members? Notice that the combination of people-types and the flexibility of adding new types make the "people-type" field difficult to maintain.

***This example of a valid use of a one-to-one table relationship is often called Generalization and Specialization. One table, e.g., PEOPLE, is labeled the "Generalized table" when common information for all specialized information is stored. The Specialization tables, e.g., STUDENT, EMPLOYEE, and FACULTY, are related to the generalization tables through a common primary key column and will be combined with a SQL JOIN operation.***

Consider another good example of a 1:1 relationship. Frequently accountants store journal entries into some type of journal table. On occasion they want to make additional comments concerning the entry. If every journal entry is design to include a field for comments, then the rows without comments (generally 60-90% of the total) will be wasting space. Given this requirement consider the following design.



## 4.2 One-To-Many Relationships

These are probably the most popular design relationships in data base design. The first normalizing rule cited above applies this concept by separating repeating items, i.e., many, into a different table. We have already discussed several 1:N relationships. Consider the following examples.

* An invoice header has many line items
* A customer has many invoices
* A stock item appears in many line items of the invoices
* A class has many students
* A student takes many classes
* A recipe has many ingredients
* A journal transaction may have many debits and credits
* A car model may have many components
* A course may have many prerequisites

Valid One-to-One and Many-to-Many Relationship designs are rare. The most frequent relationship applied in database design is the One-to-One relationship.

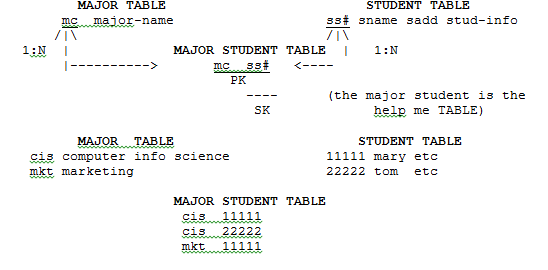
## 4.3 Many-To-Many Relationships

In a previous example, this paper proposed designing the major field as an index in order to group students by major. This design had one major shortcoming: a student could not have more than one major. The major-student relationship is, therefore, a many-to-many relationship.



### 4.3.1 Student – Major Many-to- Many Relationships

While a Many-to-Many relationship is conceptually valid, a relational database cannot physically implement this type of relationship. Therefore, the way to design a N:M relationship is to break it down into two 1:N relationships through the use of a "**help me**" table as described by Professor Bachman. Some database designers will call this third table an **"intersection**" table.



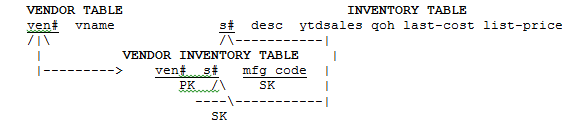
Now a student can have as many majors as they desire. You can process the major student table by using the primary key to get a list of all the students in a major in "student number order." Or you can use the index, which groups the majors for a given student, to list all the majors for a given student.

### 4.3.2 Invoice – Inventory Many-to- Many Relationships

Consider the following second example of an N:M relationship from our invoicing case. Isn't the invoice item an N:M relationship between the inventory table and the invoice header table? An invoice contains many different stock items and each stock item may be sold on many different invoices.

**Vendor – Inventory Many-to- Many Relationships**

We can also introduce another N:M relationship between the inventory table and the vendor table. A vendor supplies many inventory items. A given inventory item may be supplied by many vendors. Let's consider the following design:



Using the composite primary key of the vendor inventory table, i.e., ven# + s#, a report can be prepared that will provide a list of stock items supplied by a given vendor. The index, i.e., s#, can be accessed to produce a report that will provide a list of vendor numbers who supplies a given stock item. Notice that this discussion of the primary and indexes for the vendor inventory table is similar to that for the major student table. Most "help me" designs follow this pattern. **Therefore, a "help me" (intersection) table will contain two or primary keys of other related parent tables, e.g., the inventory and vendor table**. Later, when one implements this database design, one will want to declare these primary keys as foreign keys that reference the primary keys of the related table.

|  |  |
| --- | --- |
| VENDOR INVENTORY TABLE  Order by the composite key ven# + s# | VENDOR INVENTORY TABLE  Order by the key s# |
| Ven# S# mfg-code  10 23 brown model 1  10 667 screw driver  10 876 bolt  15 55 paper  15 667 big screw driver  15 765 miller lite  18 667 different screw driver  18 765 ic lite | Ven# S# mfg-code  10 23 brown model 1  15 55 paper  10 667 screw driver  15 667 big screw driver  18 667 different screw driver  15 765 miller lite  18 765 ic lite  10 876 bolts |

Notice that that the help-me (intersection) table illustrated in the above example can store other data than the primary keys of the related tables. Storing the mfg-code for a given inventory is a sound business practice. Users can cross reference our stock number to a vendor's mfg-code listed on a vendor's shipping slip that arrives in our company's receiving room. The receiving department can inquire and inventory item by mfg-code and assign our own stock number. Since two vendors may use the same mfg-code, the mfg-code is stored in the inventory vendor table and not the inventory table. The mfg-code is dependent upon the combination of the vendor and the stock number, not the stock number by itself.

## 4.4 Appling Relationship Analysis to the Customer Invoice Database

The following example integrates a little inventory control system and a sales person table into a previous design. The FIFO table recognizes the fact that inventory items may have layers of different cost. Ten units may have a cost of $5.00 a piece, while 6 other units may a different cost of $5.10. Hence, an inventory item may have many costs (sound familiar?). The date (in a format of YYMMDDD order to provide chronological order) is combined with the stock number to make a composite key. However, the Date/Time data types of modern databases automatically store data in such a way that is automatically ordered correctly. This FIFO primary key will ensure that the first layers that were the first to be placed in stock were also the first in the order for a given stock number.

Both the cost and the price are stored in the invoice item table so that we could use these fields at a later date to provide markup analysis; even if the current cost and list price changes. The concept of a sales person table was added so that we could analyze salesperson performance.



The previous Bachman diagram illustrates some new indexes (secondary keys) and additional tables (entities), which you can use to increase performance and response time and use inter-table relationships to join with other tables to provide real-time information to support decision making. The following is a list of decisions which are common in many retail organizations. While these decisions are possible without a properly indexed, normalized database, the previous design can support both OLTP and OLAP requirements

* Determine which salesperson is getting the most net gross profit markup using the SP# index in the INVOICE table
* Determine which salesperson is selling the most of a given inventory item and yielding the most markup using the SP# index in the INVOICE table and a join to the INVORICE item TABLE
* Determine how a given vendor's product line is performing using the V#-S# index and joining the INVOICE ITEM Table
* Determine the seasonality of sales for a given stock number using the inv-date index for performance.
* Determine the seasonality of sales a markup for a given customer using the C#+INVDATE index and a join to the INVOICE ITEM table

# 5.0 Review Questions

**Video - Database normalization Part 52 -** [**https://www.youtube.com/watch?v=NScuEk7CSNo**](https://www.youtube.com/watch?v=NScuEk7CSNo)

**Video - Normalization -** [**https://www.youtube.com/watch?v=IwswNIZ\_PSg**](https://www.youtube.com/watch?v=IwswNIZ_PSg)

**Video - Basics of RDBMS: Normalization - https://www.youtube.com/watch?v=tCabZRVXv2I**

***“Normalization” - Chapter 14 - Database Systems: A Practical Approach to Design, Implementation, and Management.***

***“Advanced Normalization” - Chapter 15 - Database Systems: A Practical Approach to Design, Implementation, and Management.***

A Process for Data Requirements Analysis -

<http://www.knowledge-integrity.com/Assets/DataRequirementsAnalysisProcess>

Analysis of Data Requirements Older database but discussion is great - <https://msdn.microsoft.com/en-us/library/6wty0xyc%28v=vs.71%29.aspx>

Functional requirements - <https://en.wikipedia.org/wiki/Functional_requirement>

Functional Requirements and Use Cases- <http://www.bredemeyer.com/pdf_files/functreq.pdf>

Use case - <https://en.wikipedia.org/wiki/Use_case>

After a proposed database design project has been approved, a detailed business requirement analysis is frequently conducted. **Two major phases are of interest at this point is functional analysis and data analysis.** While other detailed requirement analyses, such as, security, operational, design, interaction, operational, infrastructure analysis are conducted, these types of analyses will be ignored for now.

## 5.1 Questions - Functional and Data Analysis

1. List activities that are often performed during data requirements analysis that are necessary to prepare for database design. Answer =>

Identify and list the table

Establish relation between tables

Identify the primary key

Identify the non key which will be other than primary key.

Apply normalization process

Establish appropriate data type columns.

Apply constraint in those columns which have limited value to be stored.

Index analysis

Summer 2018

2. List activities that are often performed during function requirements analysis that are necessary to prepare for application program specification. Answer =>

Summer 2018

## 5.2 Questions - OLTP and OLAP

4. What are the functions of OLTP? Answer =>

Online transaction processing is a set of systems which supports high-transaction-oriented applications. OLTP. It is used main in those fields where large number of client transactions proccesd like bank, airlines and retailers.

Summer 2018

5. How does database normalization improve the performance of an OLTP-designed database, e.g., advantages and disadvantages? Answer =>

Normalization reduces the data redundancy in a database which helps in faster transactions because update, insert and deletes are performed only once on normalized database. The need to join table’s increases as data spreads in multiple tables and this makes the task tedious.

6. What are the functions of OLAP? Answer =>

7. How does database normalization affect the use of an OLAP-designed database, e.g., advantages and disadvantages? Answer =>

8. Most modern databases are physically implemented using the relational model, e.g., Oracle, DB2, etc. OLAP-designed relational databases based a de-normalized star schema is more popularly known as \_\_\_\_\_\_\_\_\_\_\_\_\_? Answer =>

data warehouse.

9. Data Analytics are the processes to provide information from normalized or de-normalized data sources to make decisions. Describe and provide an example of each major category of data analytics using the following table.

|  |  |
| --- | --- |
| **Categories of Analytics** | **Description and Example** |
| **Descriptive Analytics** |  |
| **Predictive Analytics** |  |
| **Prescriptive Analytics** |  |

10. Provide an example that describes the limitation for each of the major category of data analytics to provide information to make quality and timely decisions using the following table.

|  |  |
| --- | --- |
| **Categories of Analytics** | **Example of the Limitations to Provide Timely and Quality Information to Make Decisions** |
| **Descriptive Analytics** |  |
| **Predictive Analytics** |  |
| **Prescriptive Analytics** |  |

11.The NASA has captured and stored space images of objects and planets over time from the Hubble telescope located in space. Describe the phone conversations. Explain the limitations of using data analytics to use this object and planet data to forecast asteroid collisions with earth. Answer =>

12. Jeff Jonas has introduced the applying Context Computing to improve the data analytic decision making process. How would one apply Jonas Context Computing using his puzzle analogy to prevent terrorist's attacks? Answer =>

## 5.3 Questions - Bottom–Up versus Top-Down Database Design Approaches

13. Assume that you are given the requirement to design a database to store flight scheduling information to the expedia.com data base. List and describe the steps you would use to design this database requirement using a Top-Down database design approach. (The answers are in this assignment document.) Answer =>

14. Assume that you are given the requirement to design a database to store flight scheduling information to the expedia.com data base. List and describe the steps you would use to design this database requirement using a Bottom-Up database design approach. (The answers are in this assignment document.) Answer =>

15. The Top-Down database design approach recommends that one identifies MAJOR NOUNS during data requirement analysis. List an easy practical way to start this approach. (The answers are in this assignment document.) Answer =>

The great primary-key debate - http://www.techrepublic.com/article/the-great-primary-key-debate/

How to Choose a Primary Key - http://sqlmag.com/database-administration/sql-design-how-choose-primary-key

Choosing a Primary Key - <http://databases.about.com/od/specificproducts/a/primarykey.htm>

## 5.4 Questions - Limitations Selecting a Primary Key

16. A primary key uniquely identifies each record within a table, but that’s only half the story. Why shouldn't one select and design the vendors manufacture code as their primary key for an inventory table as compared to an internal stock number? Answer =>

Summer 2018

17. Why shouldn't one select and design use the customer's purchase order number for our customer order or invoice tables as compared to an internal number or identifier? Answer =>

Summer 2018

Universal Product Code - https://en.wikipedia.org/wiki/Universal\_Product\_Code

18. The Universal Product Code used various barcode standards is designed to be unique across all industries, vendors and products. But, if you try to use UPC Codes in a food market there may be several limitations of using UPC codes as a primary key. List some practical limitations. Answer =>

19. Due to the limitations of using UPC Codes (Barcodes) as a primary key, there are often functional requirements that we want to look up (Scan) a UPC barcode and then apply our internal stock number. How can you design your database table that that will provide quick data retrieval by UPC barcode? Answer =>

Summer 2018

20. The Social Security Number is designed to be unique. But, if you try to use the Social Security Number in a hospital there may be several limitations of using Social Security Number codes as a primary key. List some practical limitations. Answer =>

Summer 2018

## 5.5 Questions - Natural Key versus Auto-incrementing Key Design Considerations

21. List the advantages of using a natural primary key as compared to using an auto-incremented primary key. Answer =>

22. List the advantages of using an auto-incremented primary key as compared to using a natural primary key. Answer =>

Summer 2018

## 5.6 Questions - Data Relationships

23. Assume that you are given the requirement to design a database to store flight scheduling information to the expedia.com data base. A sample of table names include, but no limited to List table names (major nouns) examples that illustrate each the following type of relationships. A starting example is provided.

|  |  |
| --- | --- |
| **Types of Data Relations** | **Example of Relationship and Description** |
| **One-to-One Relationships** | 1. Airplane Inventory - Airplane Extra Notes (An airplane may or may not have Extra Descriptive notes 1:1)  2.  3.  4. |
| **One-to-Many Relationships** | 1. Airplane Types and Model Description - Airplane Inventory ( A given airplane model and type may have many different airplanes in inventory (1:N)  2.  3.  4. |
| **Many-to-Many Relationships** | 1. Airline Flights - Destinations (A given airline flight may have many different destinations and stops and destination may be served by many different airline flights N:M)  2.  3.  4. |

24. In relationship to database design explain the concepts of Generalization and Specialization. (The answers are in this assignment document if you read it.) Answer =>

25. While a Many-to-Many relationships exist in the real world, one cannot implement (physical design) a Many-To-Many relationship in relational databases. The solution is to create a Many-to-Many design is to create a help-me (Bachman) or intersection table. Describe the concept of designing a help-me or intersection data. (The answers are in this assignment document.) Answer =>

26. Describe the minimum columns that should be stored in a help-me or intersection table? (The answers are in this assignment document.) Answer =>

27. Why is a foreign keys often used in a help-me or intersection table? Answer =>

28. Given the previous example of a Many-to-Many Relationships in which an airline flight may have many different destinations and stops and destination may be served by many different airline flights list and example the specific minimum column names. Answer =>

29. List the reasons or advantages for creating an index or secondary key. (The answers are in this assignment document.) Answer =>

30. List the disadvantages for creating an index or secondary key. . (The answers are in this assignment document.) Answer =>

31. Check the appropriate column whether an INDEX will improve, decrease or have no effect on performance on performance or response time when each of the following SQL statements or clauses are executed.

|  |  |  |  |
| --- | --- | --- | --- |
| **SQL Statement/Clause** | **Improve Performance or Response Time** | **Decrease Performance or Response Time** | **No Effect on Performance or Response Time** |
| **SELECT** |  |  |  |
| **CREATE TABLE** |  |  |  |
| **WHERE** |  |  |  |
| **HAVING** |  |  |  |
| **GROUP BY** |  |  |  |
| **ORDER BY** |  |  |  |
| **UPDATE** |  |  |  |
| **DELETE** |  |  |  |

32 You create the following composite keys (index) which have multiple columns. When you use the SELECT statement either the GROUP BY or ORDER BY clauses how will the index example how will the data be grouped (categorical analysis) and sorted? (The answers are in this assignment document if you read it.)

|  |  |
| --- | --- |
| **Sample CREATE INDEX statement illustrating the concept of a composite index** | **Explanation of the Grouping (Categories) and Sorting** |
| **CREATE INDEX ABC ON VENDOR\_INVENTORY(VENDOR\_ID, STOCKNO)** | The data will be first sorted and grouped by VENDOR\_ID, then within each VENDOR-ID group the data will be stored by STOCKNO |
| **CREATE INDEX ABC ON INVOICE(CUSTOMER\_ID, DATE\_PURCHASED)** |  |
| **CREATE INDEX ABC ON MAJOR\_STUDENT(MAJOR\_CODE, STUDENT\_ID)** |  |
| **CREATE INDEX ABC ON INVOICE\_ITEM(INVOICE\_NUMBER, STOCKNO)** |  |

# 6.0 Applying Normalization and Indexes Analysis

The previous Bachman diagram illustrates some new indexes (secondary keys) and additional tables (entities), which you can use to increase performance and response time and use inter-table relationships to join with other tables to provide real-time information to support decision making. The following is a list of decisions which are common in many retail organizations. While these decisions are possible without a properly indexed, normalized database, the previous design can support both OLTP and OLAP requirements



List the table names, index names and table, and query order to provide information for the following reports using the following table.

Step 1. Find the Table Names that contains columns that will provide the required information.

Step 2. Start with the first query (main table) and list the index that will increase performance of listing the report in user-friendly order, e.g., listed in order by sale person number.

Step 3. To provide information, more than one table often needs to be joined or combined. Tables are joined or combined using a common field of common name. Which indexes in which table will improve the performance of matching the columns for the join?

Step 4. List the logical order of the join or steps simply using common sense as if you were preparing this report manually. What would you do first? What information would get second? And so forth.

|  |  |  |
| --- | --- | --- |
| **Required Report or Query Information** | **Tables to be used to find this information** | **Column Indexes and Table Name, and Query Order** |
| **Sales Person Markup Report.**  **List the markup (profit) for each salesperson from their sales.** | SALE PERSON, INVOICE,  INVOICE ITEM | 1) SP# in Salesperson to order report,  2) SP# in INVOICE to group by all invoices by SP#,  3) INV# + S# in INVOICE ITEM to retrieve all invoices associated with sales person number.  4) Perform math on Quan-Ship \* (Price-Ship – Cost-Ship\_ |
| **Sales Person Inventory Sales Details**  **List inventory items and units sold for each sales person's sales.** |  |  |
| **Vendor Item Performance Report**  **List of inventory items sold and the mark-up (profit) for each item by vendor.** |  |  |
| **Inventory Sales Seasonality Report**  **List the total quantity and total markup (profit) for each inventory item sold for ALL customers by month.** |  |  |
| **Customer Inventory Sales Seasonality Report**  **List the total quantity and total markup (profit) for each inventory item sold for each individual customer by month.** |  |  |

# 7.0 Entity Relationship Modeling (ERD Diagrams)

ER Model - Basic Concepts (TutorialsPoint) - <https://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm>

ERD Representation (TutorialsPoint) - <https://www.tutorialspoint.com/dbms/er_diagram_representation.htm>

Generalization, Specialization and Aggregation (TutorialsPoint) - <https://www.tutorialspoint.com/dbms/dbms_generalization_aggregation.htm>

Entity Relationship Diagram Tutorial – https://www.lucidchart.com/pages/er-diagrams

Entity Relationship Diagram Tutorial - <https://www.lucidchart.com/pages/entity-relationship-diagram-tutorial>

Entity Relationship Diagram - <https://www.smartdraw.com/entity-relationship-diagram/> Summer 2018

**Video - Entity-Relationship Diagrams - https://www.youtube.com/watch?v=c0\_9Y8QAstg**

**Video - Introduction to Entity Relationship Modeling - https://www.youtube.com/watch?v=vmFFEok91GU**

**Video - Cardinality - https://www.youtube.com/watch?v=alc5OWoLDkY**

**Video - ERD Entity Relationship Diagram Cardinality Relationships Part 1 -** [**https://www.youtube.com/watch?v=u2QqjofJvGo**](https://www.youtube.com/watch?v=u2QqjofJvGo)

**Video ERD Entity Relationship Diagram Cardinality Relationships Part 2 -** [**https://www.youtube.com/watch?v=U3108SqTktI**](https://www.youtube.com/watch?v=U3108SqTktI)

**Video - Entity Relationship Modeling to Relational Modeling**

[**http://www.youtube.com/watch?v=qRqzmAwmJwY&feature=channel**](http://www.youtube.com/watch?v=qRqzmAwmJwY&feature=channel)

**Video - Data Modeling - ERD Diagrams**

[**http://www.youtube.com/watch?v=WSNqcYqByFk&feature=channel**](http://www.youtube.com/watch?v=WSNqcYqByFk&feature=channel)

## 7.1 Introduction to ERD: Nouns versus Verbs

This is no best order of preparing the ERD diagram and Database Design Discussion. After reviewing the taped interview and notes from my clients several times (or re-reading your case several times), I start out by listing major nouns on a sheet of paper. Remember that database design starts by defining the relationship between major collections of information (nouns) that needs to be stored for future uses.

While "verbs" are very important in the future, we must be careful not to rush too far ahead. Application programmers are very concerned with verbs. For now, we need to plan to efficiently store that data that will be needed later to perform a given task.

What is a "major" noun? I wish I had a formula to describe this analytical process to you. I start be reviewing the taped interviews and notes from my clients and count the number of time a noun is mentioned. For example, if in a university application if the noun "Student Transcript" is mentioned over thirty times, there is a good chance we found a candidate for a table.

But, what a about the student's grade, credits for a course, semester, and course and section number? These are nouns. How many times are these nouns mentioned? This is very important information that needs to be stored. Later on we will realize that these nouns are simply details of a data to be stored in a student transcript. a non-key filed.

At this point you do not even have to understand the meaning of the Major Noun. The users are the content experts. Eventually, we must formalize the meaning of that major noun but for now it is simple: "They keep talking about it so it probably has to be stored."

ER diagrams constitute a very useful framework for creating and manipulating databases. **First**, ER diagrams are easy to understand and do not require a person to undergo extensive training to be able to work with it efficiently and accurately. This means that designers can use ER diagrams to easily communicate with developers, customers, and end users, regardless of their IT proficiency. **Second**, ER diagrams are readily translatable into relational tables which can be used to quickly build databases. In addition, ER diagrams can directly be used by database developers as the blueprint for implementing data in specific software applications. Lastly, ER diagrams may be applied in other contexts such as describing the different relationships and operations within an organization.

## 7.2 Common Errors Identifying Major Nouns and Entities

Your ability to distinguish between major and minor nouns will improve with practice and experience. Three mistakes that are frequently made by novice database designers will seriously harm your long term credibility.

1) Some users designers initially repeat-and-repeat the same major noun over and over again during discussions of their initial requirement analysis. If a user repeats a major noun over 20 times and the data base designer failed to consider these repetitive nouns in their preliminary design, the database designer would at the minimum be viewed as simply not listening. Users make mistakes of "not" telling you about important and non-important nouns - errors of omission. But, if a user talks and talks and talks about it - you better figure out how to handle it.

2) Users and organization have a legacy of terminology that they expect you to learn. For example, do not substitute the descriptive noun "Student Grades" for "Student Transcript" simply because you believe the new term is more descriptive or appropriate. If an organization has successfully used the terminology for years, never, never show your arrogance of changing it without politely presenting your case at a later time.

3) Never introduce a major noun or concept in your preliminary designs that was not presented in your research of the data base requirements. For example, if the requirement analysis of a Student Transcript system indicates that you are NOT to consider Student Accounts - then don't bring up student accounts. If the design is limited - do not exceed those limits.

Perhaps your experience has provided you a basis for recommend improvements. Never let those improvements exceed the scope of the project. Never propose improvements until the users are comfortable with your knowledge of the existing system. Do not open Pandora's Box. Even the most limited project has pitfalls that you cannot anticipate at early design point. Do not go off on Tangents. Summer 2018

## 7.3 ER Diagrams Usage

What are the uses of ER diagrams? Where are they used? Although they can be used to model almost any system they are primarily used in the following areas.

ER Models in Database Design

### 7.3.1 Database ERD Diagrams

They are widely used to design relational databases. The entities in the ER schema become tables, attributes and converted the database schema. Since they can be used to visualize database tables and their relationships it’s commonly used for database troubleshooting as well.

ER diagrams in software engineering. **A collection of entities and relationships is called a database**.

### 7.3.1 Software Engineering ERD Diagrams

Entity relationship diagrams are used in software engineering during the planning stages of the software project. They help to identify different system elements and their relationships with each other. It is often used as the basis for data flow diagrams or DFD’s as they are commonly known.

## 7.4 ER Diagram Symbols and Notations

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/ER-Diagram-Elements.jpeg)

There are three basic elements in an ER Diagram: entity, attribute, relationship. There are more elements which are based on the main elements. They are weak entity, multi valued attribute, derived attribute, weak relationship, and recursive relationship. Cardinality and ordinality are two other notations used in ER diagrams to further define relationships.

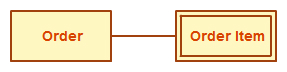
### 7.4.1 Entity

An entity can be a person, place, event, or object that is relevant to a given system. For example, a school system may include students, teachers, major courses, subjects, fees, and other items. In a relational database design an entity is analogous to a table. In flat file design an entity is analogous to a file or dataset. Entities are represented in ER diagrams by a rectangle and named using singular nouns.

**Entity Occurrence**. Each row in a table represents a specific, single occurrence of the entity. For example, if customer is an entity, a customer table represents the idea of customer; in it, each row represents one specific customer, such as Sue Smith. Keep in mind that entities become tables, attributes become columns, and **entity occurrences become rows.**

### 7.4.2 Weak Entity

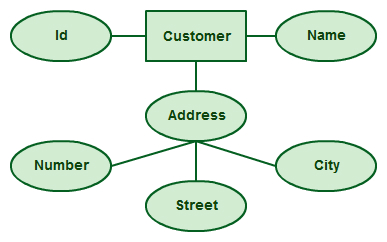
A weak entity is an entity that depends on the existence of another entity. In more technical terms it can be defined as an entity that cannot be identified by its own attributes. It uses a **foreign key** combined with its attributed to form the primary key. An entity like order item is a good example for this. The order item will be meaningless without an order so it depends on the existence of the order.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/weak-entity-ER-diagrams.jpeg)

Flat files do not support foreign keys. Hence it relies of application program logic to enforce this constraint

### 7.4.3 Attribute

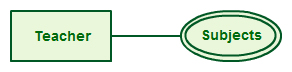
An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute. In database design an attribute is analogous to a table column. In flat files an attribute is analogous to a filed name. For example, the attribute Inventory Item Name is an attribute of the entity Inventory Item. An entity can have as many attributes as necessary. Meanwhile, attributes can also have their own specific attributes. For example, the attribute “customer address” can have the attributes number, street, city, and state. These are called composite attributes. Note that some top level ER diagrams do not show attributes for the sake of simplicity. In those that do, however, attributes are represented by oval shapes.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Attributes-ER-Diagrams.jpeg)

An attribute can have its own sub- attributes (composite attribute), e.g., Address is compose of Number, Street and City

### 7.4.4 Multivalued Attribute

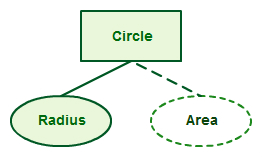
If an attribute can have more than one value it is called a multi valued attribute. It is important to note that this is different to an attribute having its own attributes. For example, a teacher entity can teach or have multiple subject values.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Multivalued-Attribute-ER-Diagrams.jpeg)

In relation database design a Teacher table may contain three attributes (columns) to store each subject. Another design alternative would be to create a second table, e.g., Subjects, to design a one-to-many relationship

### 7.4.5 Derived Attribute

An attribute based on another attribute. This is found rarely in ER diagrams, since the rules of normalization does not store calculated fields. For example, for a circle, the area can be derived from the radius.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Derived-Attribute-ER-Diagrams.jpeg)

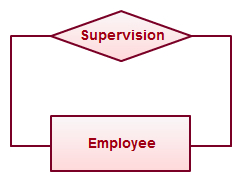
## 7.5 Relationship

A relationship describes how entities interact. For example, the entity “Carpenter” may be related to the entity “table” by the relationship “builds” or “makes”. Relationships are represented by diamond shapes and are labeled using verbs. The diamond symbol is often used to describe a relationship. Some ERD designers may omit the relationship diamond symbol, but will use the cardinality and optionality values

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Relationship-ER-diagrams.jpeg)

### 7.51 Recursive Relationship

If the same entity participates more than once in a relationship it is known as a recursive relationship. In the below example an employee can be a supervisor and be supervised, so there is a recursive relationship. For example, a supervisor is an employee, but most employees have a supervisor, which is simply another employee See examples of SQL Self joins

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Recursive-Relationship-ER-Diagrams.jpeg)

### 7.5.2 Multiplicity, Cardinality and Participation

**Multiplicity** describes the minimum and maximum of a relationship between two tables To summarize the a relationship between tables Multiplicity = Cardinality + Participation. Consider the multiplicity relationship between a teacher table (the parent table) and a course table

* **Cardinality**: Denotes the **maximum number** of possible relationship occurrences between tables in which a certain entity can participate in (in simple terms: at most). Design characteristics will answer the two questions of cardinality are 1) What is the maximum number of a course a teacher may teach, and 2) What are the maximum number of teachers that may teach a class.
* **Participation**: Denotes the minimum number of possible relationship occurrences between tables Denotes if all or only some entity occurrences participate in a relationship (in simple terms: at least). Design characteristics will answer two questions of participation are 1) What is the minimum number of a courses a teacher may teach (Must a teacher teach a class?), and 2) What are the minimum number of teachers that may teach a class (Must a class have a teacher?).

The following examples defines relationships between entities by placing the relationship in the context of minimum and maximum numbers In an email system, for example, one account can have multiple contacts. The relationship, in this case, follows a “one to many” model. There are a number of notations used to present cardinality in ER diagrams. Chen, UML, Crow’s foot, Bachman are some of the popular notations. For example one email account may have no contacts or may have many contacts. The following example uses UML to show cardinality. Many ERD design will omit the Has Relations symbol.

[](https://d3n817fwly711g.cloudfront.net/blog/wp-content/uploads/2012/03/Cardinality-ER-Diagrams.jpeg)

### 7.5.3 Degrees of Relationship (Cardinality)

The degree of relationship (also known as cardinality) is the number of occurrences in one entity which are associated (or linked) to the number of occurrences in another.

## 7.6 How to Draw ER Diagrams

Below points show how to go about creating an ER diagram.

1. **Identify all the entities** in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
2. **Identify relationships** between entities. Connect them using a line and add a diamond in the middle describing the relationship.
3. **Add attributes** for entities. Give meaningful attribute names so they can be understood easily.

## 7.7 ER Diagram Best Practices

1. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Terms that are simple and familiar always beats vague, technical-sounding words. In naming entities, remember to use singular nouns. However, adjectives may be used to distinguish entities belonging to the same class (part-time employee and full-time employee, for example). Meanwhile attribute names must be meaningful, unique, system-independent, and easily understandable.
2. Remove vague, redundant or unnecessary relationships between entities.
3. Never connect a relationship to another relationship.
4. Make effective use of colors. You can use colors to classify similar entities or to highlight key areas in your diagrams.

## 7.8 Generalization Specialization, Aggregation and Composition in DBMS

Generalization Specialization and Aggregation in DBMS are abstraction mechanisms used to model information. The abstraction is the mechanism used to hide the superfluous details of a set of objects. For example, vehicle is a abstraction, that includes the types car, jeep and bus.

So, the two abstraction mechanisms used to model information :

* Generalization (Specialization is the reverse process of Generalization)
* Aggregation

### 7.8.1 Generalization and Specialization in DBMS

Generalization is an abstracting process of viewing sets of objects as a single general class by concentrating on the general characteristics of the constituent sets while suppressing or ignoring their differences.

In simple terms, Generalization is a process of extracting common characteristics from two or more classes and combining them into a generalized superclass. So, it is a bottom up approach as two or lower level entities are combined to form a higher level entity.

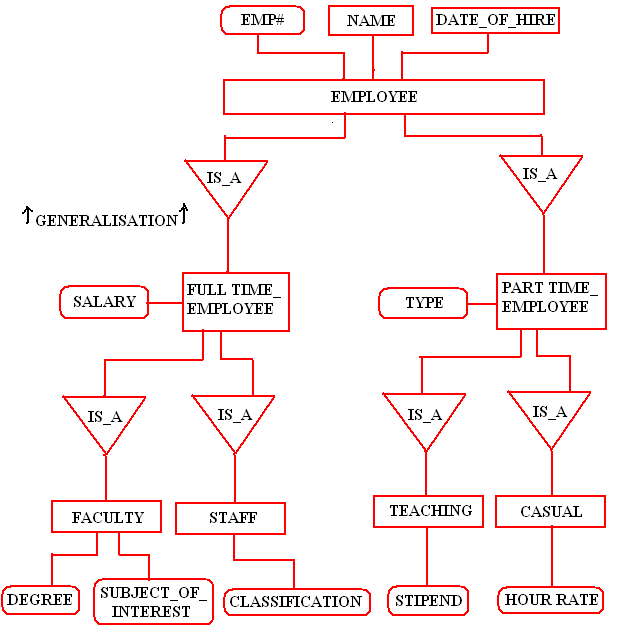
The common characteristics means here attributes or methods. Generalization is represented by a triangle with a line.

generalisation notation-Generalization Specialization and Aggregation in DBMS

Generalization/Specialization is an IS\_A relationship

**Example of Generalization :**

**For example,**a Part\_Time Employee and Full\_Time Employee entity sets can be generalized as EMPLOYEE entity sets;  
The Full\_Time\_Employee is a generalization of the entity set Faculty and Staff;  
The Part\_Time\_Employee is a generalization of the entity set Teaching and Casual.

[](http://www.edugrabs.com/wp-content/uploads/2015/07/GENERALISATION-ERD.bmp)

The design would normally be implemented using five tables. The Employee table will store attributes that are common to all employees, e.g., employee number, name and date of hire. The Full time-Employee table will store attributes which are common to all full time employees, e.g., salary. The Faculty and Staff tables will store information to all Faculty or Staff members.

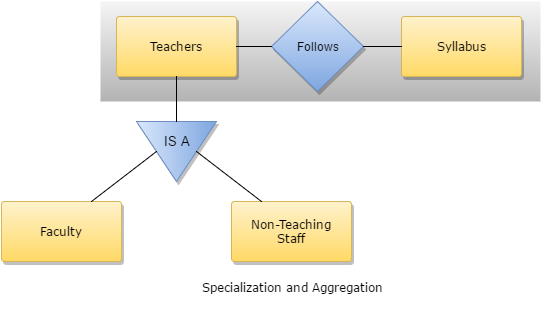
### 7.8.2 Aggregation in DBMS

Aggregation is the process of compiling information on an object, thereby abstracting a higher level object. So a n entity Person is derived by aggregating the characteristics name, house\_no., city and social security number(SSN).

A relationship represents a connection between two entity types that are conceptually at the same level. Sometimes you may want to model a **‘has-a’, ‘is-a’ or ‘is-part-of’ relationship**, in which one entity represents a larger entity (the ‘whole’) that will consist of smaller entities (the ‘parts’). This special kind of relationship is termed as an aggregation.

Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As shown in the image, a Professor 'has a' class to teach or has a course syllabus, e.g., the course syllabus is owned by the teacher..

An example of an aggregation is the ‘Teacher’ entity following the ‘syllabus’ entity act as a single entity in the relationship. In simple words, **aggregation is a process where the relation between two entities is treated as a single entity.**

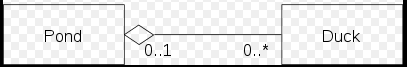


The aggregation symbol is also is graphically represented as a hollow diamond shape on the containing table or class with a single line that connects it to the contained table or class.



https://upload.wikimedia.org/wikipedia/commons/thumb/2/2a/KP-UML-Aggregation-20060420.svg/381px-KP-UML-Aggregation-20060420.svg.png

A professor may have one or more classes, but a class must have a professor



A Pond has zero or more Ducks, and a Duck has at most one Pond (at a time).

### 7.8.3 Composition in DBMS

The UML graphical representation of a composition relationship shows composition as a filled diamond shape on the containing class end of the lines that connect contained tables or cass(es) to the containing table or class.





A Car has exactly one Carburetor, and a Carburetor has at most one Car (Carburetors may exist as separate parts, detached from a specific car).

## 7.9 Questions - Applying ERD Concepts

1. Provide a user friendly description of the following concepts of Entity Relationship Modeling and Diagrams using the following tables. Also provide or describe a specific example of the concept using the final Bachman database design.



### 7.9.1 Examples and Description of UML Concepts

Each of the following ERD Concepts are clearly describe in Chapters 12 and 13 of your text book. However, you will ALSO need to apply these concepts to final Bachman database design.

|  |  |  |
| --- | --- | --- |
| **ERD Concept** | **Description** | **Example of the UML concept** |
| **UML Uniformed Markup Language** |  |  |
| **Entities or Entity Types** |  |  |
| **Entity Occurrence** |  |  |
| **Relationship Type** |  |  |
| **Degree of a Relationship Type** |  |  |
| **Attributes** |  |  |
| **Simple Attributes** |  |  |
| **Composite Attributes** |  |  |
| **Single-value attributes** |  |  |
| **Multiple-value attributes** |  |  |
| **Derived Attribute** |  |  |
| **Strong Entity Type** |  |  |
| **Weak Entity Type** |  |  |
| **Multiplicity** |  |  |
| **Cardinality** |  |  |
| **Participation** |  |  |

2. List the ERD symbol(S) used to graph the following concept using the following table

|  |  |
| --- | --- |
| **ERD Concept** | **ERD Graphing Symbol** |
| **Entities or Entity Types** |  |
| **Relationships** |  |
| **Attributes** |  |
| **Composite Attributes** |  |
| **Single-value attributes** |  |
| **Multiple Value attribute** |  |
| **Multiplicity, Cardinality and Participation** |  |
| **Aggregation** |  |
| **Composition** |  |

3. Assume that you are analyzing the relationship between storing flight reservation data for Expedia.com for different Airlines, e.g., United, American, WOW, etc. and From-TO Routes, e.g., PG to Las Vegas, NY to LA, etc. **List two specific design questions that you must consider to analyze the cardinality of this relationship.** Answer =>

4. Assume that you are analyzing the relationship between storing flight reservation data for Expedia.com for different Airlines, e.g., United, American, WOW, etc. and From-TO Routes, e.g., PG to Las Vegas, NY to LA, etc. **List two specific design questions that you must consider to analyze the participation of this relationship.** Answer =>

5. Assume that you are designing a database that will store information for micro beer brewery for each micro and its ingredient for each beer. **Describe and apply one of the following ERD relationships which will be most appropriate, e.g., Generalization, and Specialization, Aggregation or Composition.**  Answer =>

6. Assume that you are designing a database that will store investment information for a stock broker for the various types of stock, bonds and other investment alternatives **Describe and apply one of the following ERD relationships which will be most appropriate, e.g., Generalization, and Specialization, Aggregation or Composition**  Answer =>

7. A collection of entities and relationships is called a \_\_\_\_? Answer =>