CIS 3060 Web Site

Database Management Systems

Course Number CIS 3060

Course Name Database Management Systems

Course Description This course provides an introduction to the design, development, implementation, and manipulation of databases as well as covering the essentials of file processing. The student will create information level database designs from a set of user requirements and implement those designs employing a 4GL database tool. Finally, the student will be introduced to current topics in the database field, which may include Data Warehousing, Distributed Systems, Object-Oriented Systems, Business Intelligence, and Big Data Analytics.

Prerequisites CIS 2010 and CIS 2110

Course Objectives Upon completion of this course you will be skilled in:

- Informational level modeling using the DBDL and E-R modeling techniques.
- Normalization.
- SQL (Structured Query Language)
- Physical implementation using the relational database model.

In addition, you will have knowledge of the framework and content of the following:

- Vocabulary specific to the database area.
- DBMS functionality.
- Data Warehouse and Business Intelligence.
- Database administration.
- Distributed systems.
- Big Data Analytics

Instructor Dr. George Garman

Office AD 590N

Hours Monday & Wednesday 1:30 - 2:00 and 3:15 - 5:15

Joseph Martinez

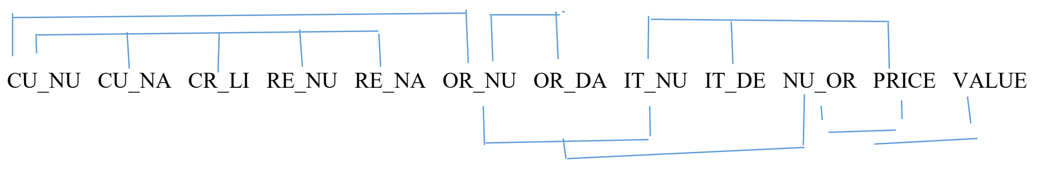
CIS-3060-002

Second Assignment

Normalizing Relationships (20 Points)

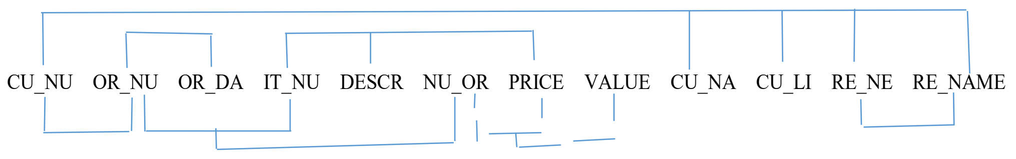
TAL

1a) Draw the Dependency Diagram for the report identifying the primary key of the relationship. Show all of the relevant dependencies.



1b) Is the relation in 1NF? If not, convert the relationship to 1NF and identify all keys. Draw the revised Dependency Diagram.

First Normal



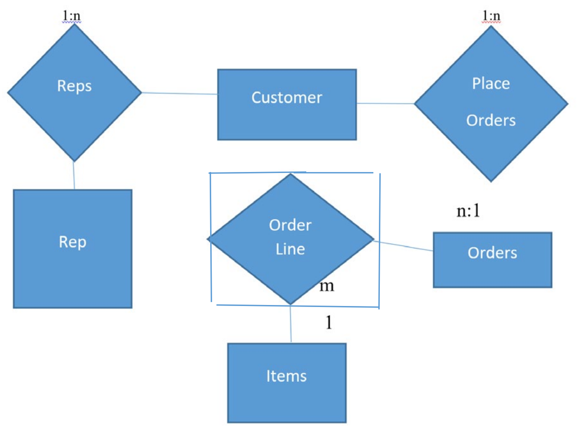
1c) Is the relation in 2NF? If not, convert the relationship to 2NF and identify all keys. Draw the revised Dependency Diagram.
No, it is not in 2NF

1d) Is the relation in 3NF? If not, convert the relationship to 3NF and identify all keys. Draw the revised Dependency Diagram.

	PK	FK	
ORDERS:	ORD_NUM	CUS_NUM	ORD_DATE
CUSTOMER:	CUS_NUM	REP_NUM	CUS_NAME
ITEMS:	ITEM_NUM	DESCRIPTION	PRICE
ORDER LINE:	ORD_NUM	ITEM_NUM	NUM_ORDERED
REPS:	REP_NUM	REP_NAME	

1e) Is the relation in 4NF? Yes

1f) Draw an (Chen) E-R model to represent the entities associated with the final TAL Distributors design. Include the following in the diagram.

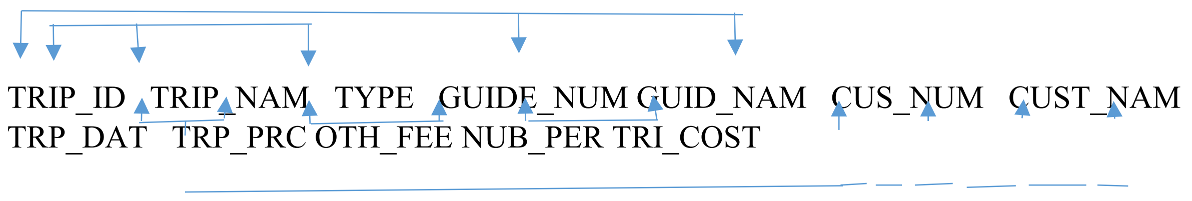


Derived attribute is NUM_ORD,   PRICE,   VALUE

Composite attribute: Not any

Colonial Adventure Tours

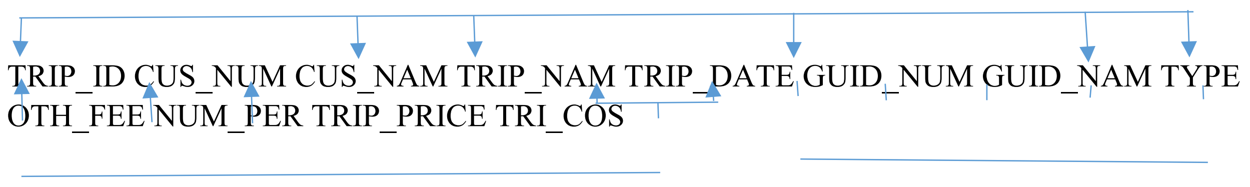
2a) Draw the Dependency Diagram for the report identifying the primary key of the relationship. Show all of the relevant dependencies.



TRIP_ID TRIP_NAM TYPE GUIDE_NUM GUID_NAM CUS_NUM CUST_NAM TRP_DAT TRP_PRC OTH_FEE NUB_PER TRI_COST

2b) Is the relation in 1NF? If not, convert the relationship to 1NF and identify all keys. Draw the revised Dependency Diagram.

First Normal



TRIP_ID CUS_NUM CUS_NAM TRIP_NAM TRIP_DATE GUID_NUM GUID_NAM TYPE OTH_FEE NUM_PER TRIP_PRICE TRI_COS

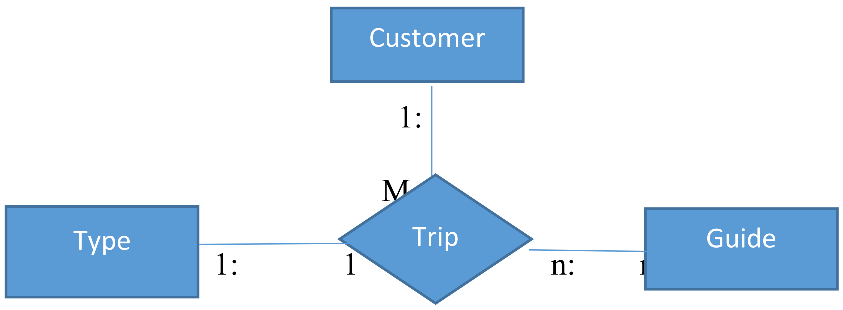
1c) Is the relation in 2NF? If not, convert the relationship to 2NF and all keys. Draw the revised Dependency Diagram. **No**, it is not in 2NF

2d) Is the relation in 3NF? If not, convert the relationship to 3NF and identify all keys. Draw the revised Dependency Diagram.

TRIP:	TRIP_ID	GUIDE_NUM	TRIP_DATE	CUS_NUM
CUSTOMER:	CUS_NUM	NUM_PERSONS	TRIP_PRICE	OTHER_FEES
GUIDE:	GUIDE_NUM	TRIP_ID	GUIDE_NAME	
PRICE:	TRIP_ID	CUST_NUM	TYPE	

2e) Is the relation in 4NF? Yes

Draw an (Chen) E-R model to represent the entities associated with the final TAL Distributors design. Include the following in the diagram.



```
-- Name


select *

from rep;

-- Question 4

Select        customer_name,

                     balance,

                     street,

                     city,

                     state

From           Customer;



Select        customer_name,

                     street,

                     city,

                     state

From           Customer;



Select        Balance,

Credit_Limit,

Credit_Limit - Balance Available

From          Customer;
```

```
Create table xyx

(x char(3), y char(7)     );



Select           To_Char(Balance,'$999,999.99'),

                           To_Char(Credit_Limit,'$999,999.99'),

                           To_Char(Credit_Limit - Balance,'$999,999.99') Available

From          Customer;



select        reservation_id,

                        num_persons,

                        trip_price,

                        other_fees,

                        num_persons * (trip_price + other_fees) "Cost"

from            Reservation;



select      sum(balance),

sum(credit_limit - balance),

max(balance),

min(balance),

avg(balance),

count(*)
```

```
from            customer;


select              customer_num, customer_name, balance

from                  customer

order by    balance desc ;


select rep_num,

sum(balance), sum(credit_limit), count(*), max(balance)

from customer

group by (rep_num)

having rep_num != '45'

order by (sum(balance)) desc;


select       *

from            customer

where      City = 'Fullton'

and                rep_num = 45

or                    balance > 1000;


select       *
```

```
from            customer
where
                          (rep_num = 45
or                      balance > 1000)
and                 City = 'Fullton';



select        customer_name
from            customer
where       credit_limit >= 5000
and                 credit_limit <= 10000;



select        customer_name
from            customer
where       credit_limit not between 5000 and 10000;



select        customer_name
from            customer
where       credit_limit != 10000;



select        customer_name
```

```
from            customer

where      credit_limit is not null;



select        customer_name

from            customer

where      customer_name like '_h%';



select rep_num,

sum(balance)

from customer

where balance > 2000

group by (rep_num)

having        rep_num != '45'

order by (sum(balance)) desc;



select customer_name

from      customer

where rep_num in ('15', '45');
```

Joseph Martinez

CIS-3060-002

Exam 1 Review

Chapter One-Database Systems

Data management is a discipline that focuses on the proper generation, storage, and retrieval of data. A **database** is a shared, integrated computer structure that stores a collection of the following:

• End-user data—that is, raw facts of interest to the end user.

• Metadata, or data about data, through which the end-user data are integrated and managed.

A database management system **(DBMS)** is a collection of programs that manages the database structure and controls access to the data stored in the database. **Data inconsistency** exists when different versions of the same data appear in different places. A **query** is a specific request issued to the DBMS for data manipulation—for example, to read or update the data. Simply put, a query is a question, and an **ad hoc query** is a spur-of-the-moment question. The DBMS sends back an answer (called the **query result set**) to the application.

Extensible Markup Language (XML) is a special language used to represent and manipulate data elements in a textual format. An **XML database** supports the storage and management of semi structured XML data. The term **NoSQL** (Not only SQL) is generally used to describe a new generation of database management systems that is not based on the traditional relational database model. NoSQL databases are designed to handle the unprecedented volume of data, variety of data types and structures, and velocity of data operations that are characteristic of these new business requirements.

Database design refers to the activities that focus on the design of the database structure that will be used to store and manage end-user data.

A file system exhibits **structural dependence**, which means that access to a file is dependent on its structure. **Structural independence** exists when you can change the file structure without affecting the application's ability to access the data. A **data anomaly** develops when not all of the required changes in the redundant data are made successfully. **Data redundancy** exists when the same data are stored unnecessarily at different places.

The DBMS uses the **data dictionary** to look up the required data component structures and relationships,
thus relieving you from having to code such complex relationships in each program.
A **query language** is a nonprocedural language—one that lets the user specify what must be done without having to specify how. **Structured Query Language (SQL)** is the de facto query language and data access standard supported by the majority of DBMS vendors.

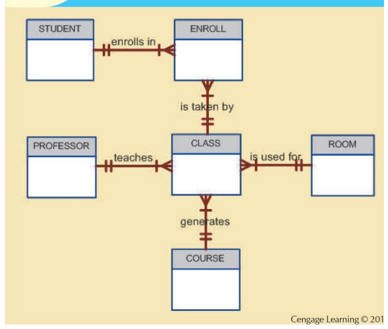
Chapter Two-Data Models

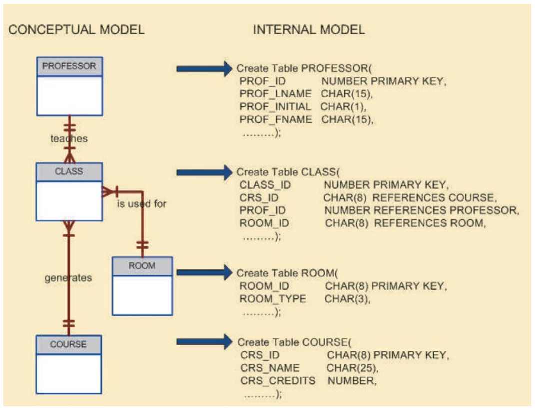
Data modeling, the first step in designing a database, refers to the process of creating a specific data model for a determined problem domain. A **data model** is a relatively simple representation, usually graphical, of more complex real-world data structures. An **entity** is a person, place, thing, or event about which data will be collected and stored. An **attribute** is a characteristic of an entity. For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone number, customer address, and customer credit limit. Attributes are the equivalent of fields in file systems.
A **relationship** describes an association among entities. **One-to-many (1:M or 1**..*) relationship. A painter creates many different paintings, but each is painted by only one painter. **Many-to-many (M:N** or *..*) relationship. An employee may learn many job skills, and each job skill may be learned by many employees. **One-to-one (1:1** or 1..1) relationship. A retail company's management structure may require that each of its stores be managed by a single employee. A **constraint** is a restriction placed on the data. Constraints are important because they help to ensure data integrity.

The **relational** model represented a major breakthrough for both users and designers. To use an analogy, the relational model produced an “automatic transmission” database to replace the “standard
transmission” databases that preceded it. The **relational database model** is implemented through a very sophisticated relational database management system (RDBMS). A **relational diagram** is a representation of the relational database’s entities, the attributes within those entities, and the relationships between those entities.

Figure 2.3 shows the different types of relationships using three ER notations: the original **Chen notation**, the **Crow’s Foot notation**, and the newer **class diagram notation**, which is par

FIGURE 2.9 Conceptual model for Tiny College





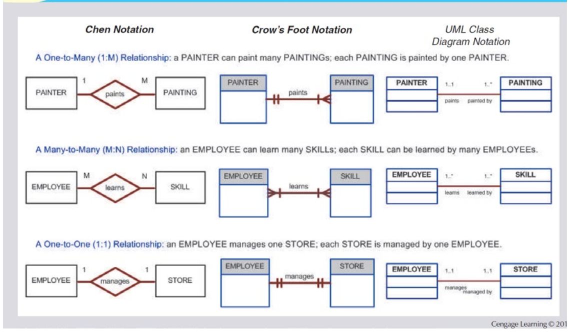
```
Create Table PROFESSOR(
 PROF_ID        NUMBER PRIMARY KEY,
 PROF_LNAME     CHAR(15),
 PROF_INITIAL   CHAR(1),
 PROF_FNAME     CHAR(15),
  .........);

Create Table CLASS(
 CLASS_ID       NUMBER PRIMARY KEY,
 CRS_ID         CHAR(8)  REFERENCES COURSE,
 PROF_ID        NUMBER REFERENCES PROFESSOR,
 ROOM_ID        CHAR(8)  REFERENCES ROOM,
  .........);

Create Table ROOM(
 ROOM_ID        CHAR(8) PRIMARY KEY,
 ROOM_TYPE      CHAR(3),
  .........);

Create Table COURSE(
 CRS_ID         CHAR(8) PRIMARY KEY,
 CRS_NAME       CHAR(25),
 CRS_CREDITS    NUMBER,
  .........);
```

FIGURE 2.3 The ER model notations



A One-to-Many (1:M) Relationship: a PAINTER can paint many PAINTINGs; each PAINTING is painted by one PAINTER.

A Many-to-Many (M:N) Relationship: an EMPLOYEE can learn many SKILLs; each SKILL can be learned by many EMPLOYEEs.

A One-to-One (1:1) Relationship: an EMPLOYEE manages one STORE; each STORE is managed by one EMPLOYEE.

Chapter Three-The Relational Database Model

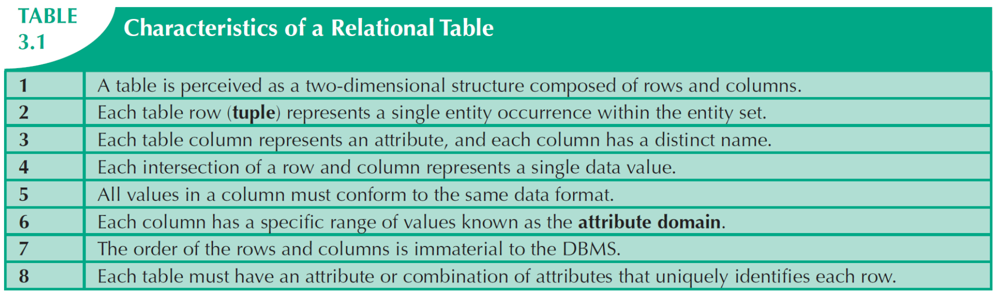
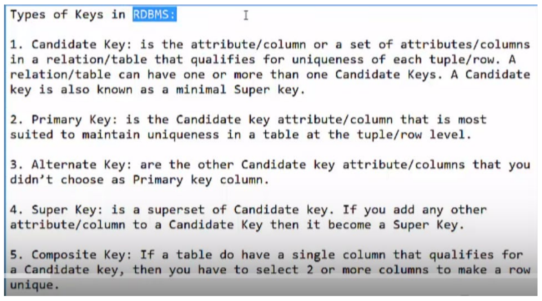


TABLE 3.1 **Characteristics of a Relational Table**

1	A table is perceived as a two-dimensional structure composed of rows and columns.
2	Each table row (**tuple**) represents a single entity occurrence within the entity set.
3	Each table column represents an attribute, and each column has a distinct name.
4	Each intersection of a row and column represents a single data value.
5	All values in a column must conform to the same data format.
6	Each column has a specific range of values known as the **attribute domain**.
7	The order of the rows and columns is immaterial to the DBMS.
8	Each table must have an attribute or combination of attributes that uniquely identifies each row.

The column's range of permissible values is known as its **domain**. Each table must have a primary key. In general terms, the **primary key (PK)** is an attribute or combination of attributes that uniquely identifies any given row.

The relationship is called **functional dependence**, which means that the value of one or more attributes determines the value of one or more other attributes. In this functional dependency, the attribute whose value determines another is called the **determinant** or the key. The attribute whose value is determined by the other attribute is called the **dependent**.


```
Types of Keys in RDBMS:

1. Candidate Key: is the attribute/column or a set of attributes/columns
in a relation/table that qualifies for uniqueness of each tuple/row. A
relation/table can have one or more than one Candidate Keys. A Candidate
key is also known as a minimal Super key.

2. Primary Key: is the Candidate key attribute/column that is most
suited to maintain uniqueness in a table at the tuple/row level.

3. Alternate Key: are the other Candidate key attribute/columns that you
didn't choose as Primary key column.

4. Super Key: is a superset of Candidate key. If you add any other
attribute/column to a Candidate Key then it become a Super Key.

5. Composite Key: If a table do have a single column that qualifies for
a Candidate key, then you have to select 2 or more columns to make a row
unique.
```

Types of Keys

A **candidate key is unique,** is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more candidate keys, but one candidate key is special, and it is called the **primary key**. This is usually the best among the candidate keys.

A **foreign key** (FK) is the primary key of one table that has been placed into another table to create a common attribute. Foreign keys are used to ensure **referential integrity**, the condition in which every reference to an entity instance by another entity instance is valid.

Finally, a **secondary key** is defined as a key that is used strictly for data retrieval purposes.

A **composite key** is a key that is composed of more than one attribute. An attribute that is a part of a key is called a **key attribute**. A **superkey** is a key that can uniquely identify any row in the table. In other words, a superkey functionally determines every attribute in the row. One specific type of superkey is called a candidate key. A candidate key is a minimal superkey—that is, a superkey without
any unnecessary attributes.
Entity integrity is the condition in which each row (entity instance) in the table has its own unique identity. To ensure entity integrity, the primary key has two requirements: (1) all of the values in the primary key must be unique, and(2) no key attribute in the primary key can contain a null. Null values are problematic in the relational model.

A **null** is the absence of any data value, and it is never allowed in any part of the primary key. From a theoretical perspective, it can be argued that a table that contains a null is not properly a relational table at all. From a practical perspective, however, some nulls cannot be reasonably avoided.For example, not all students have a middle initial. As a general rule, nulls should be avoided as much as reasonablypossible. In fact, an abundance of nulls is often a sign of a poor design. Also, nulls should be avoided in the database
because their meaning is not always identifiable. For example, a null could represent any of the following:
• An unknown attribute value
• A known, but missing, attribute value
• A “not applicable” condition

Relational Set Operators- These operators are **SQL commands.**

The relational operators have the property of **closure**; that is, the use of relational algebra operators on existing relations(tables) produces new relations. Numerous operators have been defined. Some operators are fundamental, while others are convenient but can be derived using the fundamental operators. In this section, the focus will be on the

SELECT (or RESTRICT), PROJECT, UNION, INTERSECT, DIFFERENCE, PRODUCT, JOIN, and DIVIDE operators.

SELECT is the command to show all rows in a table. It can be used to select only specific data from the table that meets certain criteria. This command is also referred to as the Restrict command.

PROJECT is the command that gives all values for certain attributes specified after the command. It shows a vertical view of the given table.

UNION combines all rows from two tables, excluding duplicate rows. To be used in the UNION, the tables must have the same attribute characteristics; in other words, the columns and domains must be compatible. When two or more tables share the same number of columns, and when their corresponding columns share the same or compatible domains, they are said to be **union-compatible**. (rare)

INTERSECT takes two tables and combines only the rows that appear in both tables. The tables must be union-compatible to be able to use the Intersect command or else it won't work. (rare)

DIFFERENCE in another SQL command that gets all rows in one table that are not found in the other table. Basically it subtracts one table from the other table to leave only the attributes that are not the same in both tables. For this command to work both tables must be union-compatible.

PRODUCT yields all possible pairs of rows from two tables—also known as the Cartesian product. Therefore, if one table has six rows and the other table has three rows, the PRODUCT yields a list composed of 6 x 3= 18 rows.

JOIN is the real power behind the relational database, allowing the use of independent tables linked by common attributes. JOIN takes two or more tables and combines them into one table. This can be used in combination with other commands to get specific information. There are several types of the Join command. The **Natural Join**, Equijion, Theta Join, **Left Outer Join** and **Right Outer Join**.

DIVIDE has specific requirements of the table. One of the tables can only have one column and the other table must have two columns only.

The **data dictionary** provides a detailed description of all tables in the database created by the user and designer. Thus, the data dictionary contains at least all of the attribute names and characteristics for each table in the system. In short, the data dictionary contains metadata—data about data. (pg.89)
The **system catalog** can be described as a detailed system data dictionary that describes all objects within the database, including data about table names, the table's creator and creation date, the number of columns in each table, the data type corresponding to each column, index filenames, index creators, authorized users, and access privileges. In effect, the system catalog automatically produces database documentation. As new tables are added to the database, that documentation also allows the RDBMS to check for and eliminate **homonyms and synonyms**.

the problems inherent in the many-to-many relationship can easily be avoided by creating a **composite**
entity (also referred to as a **bridge entity** or an **associative entity**). Because such a table is used to link the tables that were originally related in an M:N relationship, the composite entity structure includes—as foreign keys—*at least* the primary keys of the tables that are to be linked.

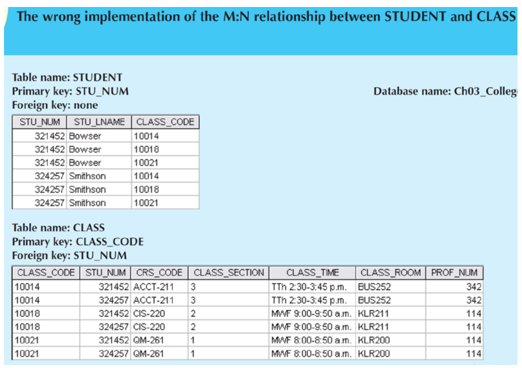

The wrong implementation of the M:N relationship between STUDENT and CLASS

Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch03_Colleg

STU_NUM	STU_LNAME	CLASS_CODE
321452	Bowser	10014
321452	Bowser	10018
321452	Bowser	10021
324257	Smithson	10014
324257	Smithson	10018
324257	Smithson	10021

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: STU_NUM

CLASS_CODE	STU_NUM	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	321452	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10014	324257	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	321452	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10018	324257	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	321452	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114
10021	324257	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114

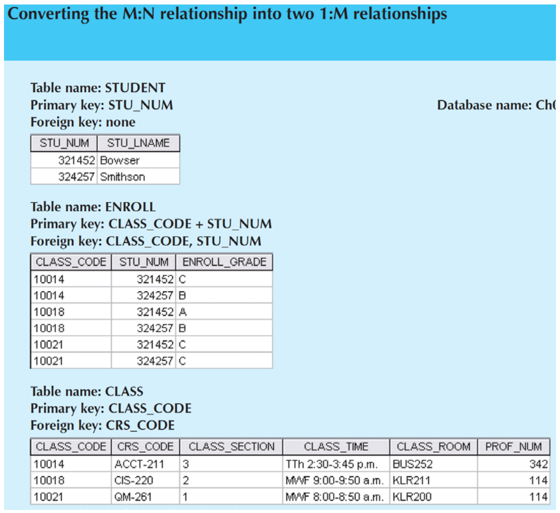

Converting the M:N relationship into two 1:M relationships

Table name: STUDENT
Primary key: STU_NUM
Foreign key: none

Database name: Ch0

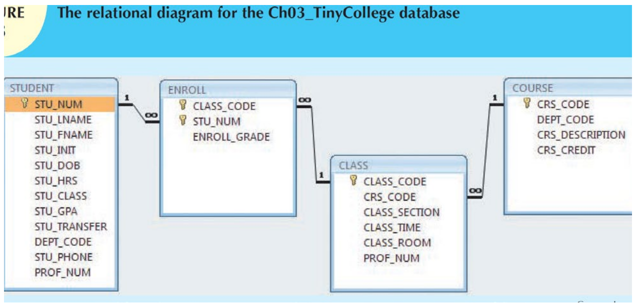
STU_NUM	STU_LNAME
321452	Bowser
324257	Smithson

Table name: ENROLL
Primary key: CLASS_CODE + STU_NUM
Foreign key: CLASS_CODE, STU_NUM

CLASS_CODE	STU_NUM	ENROLL_GRADE
10014	321452	C
10014	324257	B
10018	321452	A
10018	324257	B
10021	321452	C
10021	324257	C

Table name: CLASS
Primary key: CLASS_CODE
Foreign key: CRS_CODE

CLASS_CODE	CRS_CODE	CLASS_SECTION	CLASS_TIME	CLASS_ROOM	PROF_NUM
10014	ACCT-211	3	TTh 2:30-3:45 p.m.	BUS252	342
10018	CIS-220	2	MWF 9:00-9:50 a.m.	KLR211	114
10021	QM-261	1	MWF 8:00-8:50 a.m.	KLR200	114



Chapter Four-Entity Relationship (ER) Modeling

A **required attribute** is an attribute that must have a value; in other words, it cannot be left empty. An **optional attribute** is an attribute that does not require a value; therefore, it can be left empty. **composite identifier**, a primary key composed of more than one attribute. A **simple attribute** is attribute that cannot be subdivided. For example, age, sex,and marital status would be classified as simple attributes. A **single-valued attribute** is an attribute that can have only a single value. **Multivalued attributes** are attributes that can have many values. A **derived attribute** is an attribute whose value is calculated (derived) from other attributes. Entities that participate in a relationship are also known as **participants**, and each relationship is identified by a name that describes the relationship. **Cardinality** expresses the minimum and maximum number of entity occurrences associated with one occurrence of
the related entity. An entity is said to be **existence-dependent** if it can exist in the database only when it is associated with another related entity occurrence. If an entity can exist apart

from all of its related entities, then it is **existence-independent**, and it is referred to as a **strong entity** or **regular entity**. A **weak relationship**, also known as a **non-identifying relationship**, exists if the primary key of the related entity does not contain a primary key component of the parent entity. A **strong relationship**, also known as an **identifying relationship**, exists when the primary key of the related entity contains a primary key component of the parent entity.

Optional participation means that one entity occurrence does not require a corresponding entity occurrence in a particular relationship. **Mandatory participation** means that one entity occurrence *requires* a corresponding entity occurrence in a particular relationship. A **relationship degree** indicates the number of entities or participants associated with a relationship. A **unary relationship** exists when an association is maintained within a single entity. A **binary relationship** exists when two entities are associated. A **ternary relationship** exists when three entities are associated.

Chapter Six-Normalization of Database Tables

Normalization is a process for evaluating and correcting table structures to minimize data redundancies, thereby reducing the likelihood of data anomalies. **Denormalization** produces a lower normal form; that is, a 3NF will be converted to a 2NF through denormalization. A **partial dependency** exists when there is a functional dependence in which the determinant is only part of the primary key (remember the assumption that there is only one candidate key). A **transitive dependency** exists when there are functional dependencies such that X S Y, Y S Z, and X is the primary key. A **repeating group** derives its name from the fact that a group of multiple entries of the same type can exist for any *single* key attribute occurrence. A **determinant** is any attribute whose value determines other values within a row.
An **atomic attribute** is one that cannot be further subdivided. Such an attribute is said to display **atomicity**. **Granularity** refers to the level of detail represented by the values stored in a table's row.

CIS 3060 Final Study Guide Chapter 5, 10, 12, 13, 14

Chapter 5

Object Oriented is a database management system in which information is represented in the form of object, UML class diagrams are an object-oriented modeling language, and therefore do not support the notion of "primary or foreign keys" found mainly in the relational world. Rather, in the object-oriented world, objects inherit a unique object identifier at creation time.

Inheritance- In the object-oriented data model, the ability of an object to inherit the data structure and methods of the classes above it in the class hierarchy. See also *class hierarchy*.

Relational databases have a hierarchy

hierarchy- The organization of classes in a hierarchical tree in which each parent class is a *superclass* and each child class is a *subclass*. See also *inheritance*.

class: a means of grouping all the objects which share the same set of attributes and methods.

For example, you might identify multiple types of musical instruments: piano, violin, and guitar. Using the generalization approach, you could identify a "string instrument" entity **supertype** to hold the common characteristics of the multiple **subtypes.**

Avoid Nulls

subtype discriminator—The attribute in the supertype entity that determines to which entity subtype each supertype occurrence is related. Multiple Boolean= And / Or

Partial Completeness-can have a null-doesn't have to be in a subclass

Total Completeness- can't have nulls- True / False- at least one has to be true

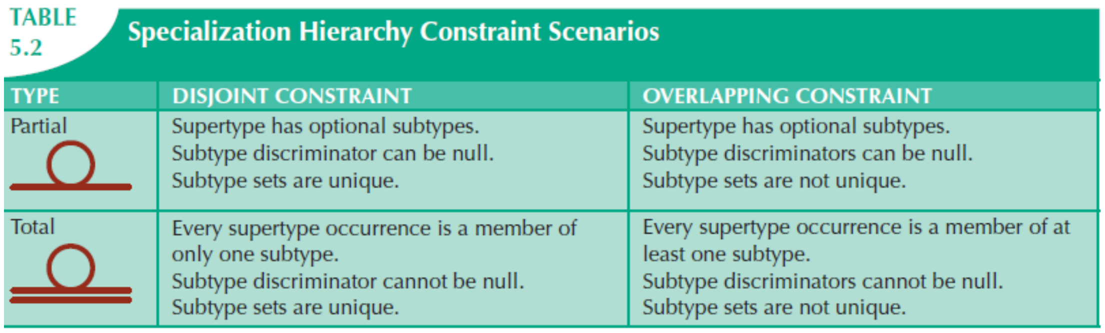


TABLE 5.2 Specialization Hierarchy Constraint Scenarios

TYPE	DISJOINT CONSTRAINT	OVERLAPPING CONSTRAINT
Partial	Supertype has optional subtypes. Subtype discriminator can be null. Subtype sets are unique.	Supertype has optional subtypes. Subtype discriminators can be null. Subtype sets are not unique.
Total	Every supertype occurrence is a member of only one subtype. Subtype discriminator cannot be null. Subtype sets are unique.	Every supertype occurrence is a member of at least one subtype. Subtype discriminators cannot be null. Subtype sets are not unique.

An **entity cluster** is a "virtual" entity type used to represent multiple entities and relationships in the ERD. An entity cluster is formed by combining multiple interrelated entities into a single, abstract entity object. An entity cluster is considered "virtual" or "abstract" in the sense that it is not actually an entity in the final ERD. Instead, it is a temporary entity used to represent multiple entities and relationships, with the purpose of simplifying the ERD and thus enhancing its readability.

• OFFERING, which groups the COURSE and CLASS entities and relationships

• LOCATION, which groups the ROOM and BUILDING entities and relationships

primary key (PK)—Unique, value, No nulls, Also, a candidate key selected as a unique entity identifier.   Desiable Primary key = Non Intellegent

Chapter 10

Transaction Management and Concurrency Control

Transactions-A request upon the database, or action/transactions don't always change/could just read.

Consistent state-all integrity states are validated (must be logical)

A data base request is any SQL single statement

Logical transaction may be broken down into parts

Transaction properties have four main properties: atomicity, consistency, isolation, and durability. **Atomicity** means that all parts of the transaction must be executed, 4 steps; otherwise, the transaction is aborted. **Consistency** means one consistent state to another, and **isolation** means that data used by one transaction cannot be accessed by another transaction until the first one is completed. **Durability** means that changes made by a transaction cannot be rolled back (undone). In addition, transaction schedules have the property of **serializability**—the result of the concurrent execution of transactions is the same as that of the transactions being executed in serial order. (the same seats cannot be resold).

ACIDS Test-

Glossary only(below)

Transaction log—A feature used by the DBMS to keep track of all transaction operations that update the database. The information stored in this log is used by the DBMS for recovery purposes. Protects the data.

Replication- mirrored image

Pessimistic locking (Brutal Lock)—More than one transaction trying to manipulate the same data. (each person locks the other one out).

Shared lock (***exclusive** lock)* (nano-second speed**)** —A lock that is issued when a transaction requests permission to read data from a database and no exclusive locks are held on the data by another transaction. A shared lock allows other read-only transactions to access the database.

Deadlock- two locks locking each other out.

Mutual exclusive rule—only one transaction can get an exclusive lock at a time.

Lock granularity (Lock anyone or everyone out)—The level of lock use. Locking can take place at the following levels: database, table, page, row, and field (attribute). Lock something if you have to work on it.

Page-level lock (data bases are stored on system by page level —In this type of lock, the database management system locks an entire diskpage, or section of a disk. A diskpage can contain data for one or more rows and from one or more tables.

Row-level lock—A less restrictive database lock in which the DBMS allows concurrent transactions to access different rows of the same table, even when the rows are on the same page.

Binary lock—A lock that has only two states: *locked* (1) and *unlocked* (0). If a data item is locked by a transaction, no other transaction can use that data item.

ROLLBACK—A SQL command that restores the database table contents to the condition that existed after the last COMMIT statement.

Database recovery management- Update log first

Fragmentation – data base is kept in more than one node.

Vertical fragmentation—In distributed database design, the process that breaks a table into a subset of **columns** from the original table. Fragments must share a common primary key.

Horizontal fragmentation—The distributed database design process that breaks a table into subsets of unique **rows**.

Factors that lead from **Centralized to Distributed** systems- redundancy, security, more computing power, different needs and applications of each region.

Chapter 12

Transparencies

Distribution transparency- allows a distributed database to be treated as a single logical database.

Fragmentation Transparencies - data are fragmented on multiple sites w/o user knowledge.

Replication Transparencies- copies are made w/o user knowing.

Transaction Transparency- update data at more than one network site. Ensures that the transaction will be either entirely completed or aborted, thus maintaining database integrity.

Failure transparency- ensures that the system will continue to operate in the event of a node or network
failure. Functions that were lost because of the failure will be picked up by another network node.

Fault tolerance transparency- no particular node is needed for full operation.

Fail over transparency- no master site.

Fail save transparencies- protects people, property, or data from injury, damage, intrusion, or disclosure.

Performance transparency, which allows the system to perform as if it were a centralized DBMS. The

system will not suffer any performance degradation due to its use on a network or because of the network's platform differences. Performance transparency also ensures that the system will find the most cost-effective path to access remote data. The system should be able to "scale out" in a transparent manner, or increase performance capacity by adding more transaction or data-processing nodes, without affecting the overall performance of the system.

Heterogeneity transparency, which allows the integration of several different local DBMSs (relational, network, and hierarchical) under a common, or global, schema. The DDBMS is responsible for translating the data requests from the global schema to the local DBMS schema.

Dis /Advantages of Distributed System
Disadvantage:
technological disadvantage / protect all systems
Security / who may or may not authorize
Lack of standards / different cost

Advantage:
Faster data processing
Growth facilitation
Improved communications

Distributed Request / Ask for something
Distributed Processing / Analysis of data

Big Data - Java base

Big file, binary files, jpg. Left in natural form

Volume- How much? Scale up, (upgrading) scale out (add nodes)

Velocity- Speed of stream processing (inputs) filtering mechanism

Variety- (structured stars) predetermined form

Variability- determine meaning in data

Intelligence- the ability to understand language

Sentiment analysis- attempt to read passages, take that data and make sense of it

Veracity- trustworthiness of data

Visualization- Present graphically, good characteristics

Polygot Persistance- are going to use a variety of ways to store and retrieve data

Hadoop- high volume of data, lots of storage

Streaming access / intolerant

3 Types Nodes

Client- receive, execute info

Main – Has all of data on specific things

Data Node- gets the data from node

Block Reports- In the Hadoop Distributed File System (HDFS), a report sent every six hours by the data node to the name node informing the name node which blocks are on that data node.

Map Reducer- Mapping out products, Reducer infinite
An open-source application programming interface (API) that provides fast data analytics services; one
of the main Big Data technologies that allows organizations to process massive data stores.

Map Reducer- request or query
Hadoop- storage
Flume Scoop- extracts from data base to Hadoop
Pig- scripting language to get results from Map Reduce

Hide- data warehouse producing
Injestion- get data in
Plume- extracts data

Heterogeneous distributed database system (fully heterogeneous DDBMS)—A system that integrates different types of database management systems (hierarchical, network, and relational) over a network. It supports different database management systems that may even **support different data models** running under different computer systems, such as mainframes, minicomputers, and
microcomputers.

Joseph Martinez

CIS 3060-002

First Assignment

Data and Relationships (20 Points)

CustomerE	
C_ID	**C_Name**
C10	Abby
C20	Betty
C30	Charley
C40	David
C50	Ed

CustomerW	
C_ID	**C_Name**
C5	Frank
C20	Betty
C35	Glen
C40	David
C70	Henry

a. Are the two tables union compatible? Explain.

Yes, they are union compatible because UNION combines all rows from two tables, excluding duplicate rows. To be used in the UNION, the tables must have the same attribute characteristics; in other words, the columns and domains must be compatible. When two or more tables share the same number of columns, and when their corresponding columns share the same or compatible domains, they are said to be union-compatible.

b. Write the contents of the result of a UNION between the two tables.

C ID	C Name
C5	Frank
C10	Abby
C20	Betty
C30	Charley
C35	Glen
C40	David
C50	Ed
C70	Henry

c. Write the contents of the result of an INTERSECTION between the two tables.

C ID	C Name
C20	Betty
C40	David

d. Write the contents of the result of a DIFFERENCE between CustomerE and CustomerW.

C ID	C Name
C10	Abby
C30	Charley
C50	Ed

e. Write the contents of the result of a DIFFERENCE between CustomerW and CustomerE.

C ID	C Name
C5	Frank
C35	Glen
C70	Henry

2. Answer the following questions based upon the two tables shown below.

MOVIE			
M_ID	***Title***	***Rating***	***D_ID***
M10	A Funny Movie	G	D20
M20	A Tear Jerker	PG	D10
M30	A Violent Movie	R	D20
M40	A Mad Slasher Movie	R	D30
M50	A Romantic Movie	PG13	D10
M60	A Chaotic Movie	PG13	

Director	
D_ID	***D_Name***
D10	Andy
D20	Bob
D30	Chris
D40	Don

a. Write the results of the PRODUCT of MOVIE and DIRECTOR.

M ID	*Title*	*Rating*	*D ID*	*D ID*	*D Name*
M10	A Funny Movie	G	D20	D10	Andy
M10	A Funny Movie	G	D20	D20	Bob
M10	A Funny Movie	G	D20	D30	Chris
M10	A Funny Movie	G	D20	D40	Don
M20	A Tear Jerker	PG	D10	D10	Andy
M20	A Tear Jerker	PG	D10	D20	Bob
M20	A Tear Jerker	PG	D10	D30	Chris
M20	A Tear Jerker	PG	D10	D40	Don
M30	A Violent Movie	R	D20	D10	Andy
M30	A Violent Movie	R	D20	D20	Bob
M30	A Violent Movie	R	D20	D30	Chris
M30	A Violent Movie	R	D20	D40	Don
M40	A Mad Slasher Movie	R	D30	D10	Andy
M40	Mad Slasher Movie	R	D30	D20	Bob
M40	Mad Slasher Movie	R	D30	D30	Chris
M40	Mad Slasher Movie	R	D30	D40	Don
M50	A Romantic Movie	PG13	D10	D10	Andy
M50	A Romantic Movie	PG13	D10	D20	Bob
M50	A Romantic Movie	PG13	D10	D30	Chris
M50	A Romantic Movie	PG13	D10	D40	Don
M60	A Chaotic Movie	PG13		D10	Andy
M60	A Chaotic Movie	PG13		D20	Bob
M60	A Chaotic Movie	PG13		D30	Chris

M60	A Chaotic Movie	PG13		D40	Don

b. Write the results of SELECT ONLY A RATING OF R from MOVIE.

Movie		
M ID	**Title**	**Rating**
A	A Violent Movie	R
M40	A Slasher Movie	R

c. Write the contents of PROJECT TITLE AND RATING OF MOVIE.

Title	**Rating**
A Funny Movie	G
A Tear Jerker	PG
A Violent Movie	R
A Slasher Movie	R
Romantic Movie	PG13
Chaotic Movie	PG13

d. Write the results of the NATURAL JOIN between MOVIE and DIRECTOR

Movie			**Director**	
M ID	**Title**	**Rating**	**D ID**	**D Name**
M10	A Funny Movie	G	D20	Bob
M20	A Tear Jerker	PG	D10	Andy
M30	A Violent Movie	R	D20	Bob
M40	A Slasher Movie	R	D30	Chris
M50	Romantic Movie	PG13	D10	Andy

e. Write the results of the LEFT OUTER JOIN between MOVIE and DIRECTOR.

Movie			**Director**	
M ID	**Title**	**Rating**	**D ID**	**D Name**
M10	A Funny Movie	G	D20	Bob
M20	A Tear Jerker	PG	D10	Andy
M30	A Violent Movie	R	D20	Bob
M40	A Slasher Movie	R	D30	Chris
M50	Romantic Movie	PG13	D10	Andy
M60	Chaotic Movie	PG13		

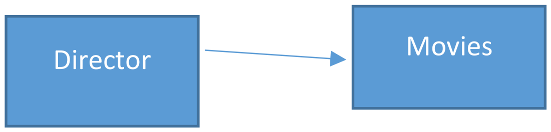
f. Write the results of the RIGHT OUTER JOIN between MOVIE and DIRECTOR.

Movie			**Director**	
M ID	**Title**	**Rating**	**D ID**	**D Name**
M10	A Funny Movie	G	D20	Bob

M20	A Tear Jerker	PG	D10	Andy
M30	A Violent Movie	R	D20	Bob
M40	A Slasher Movie	R	D30	Chris
M50	Romantic Movie	PG13	D10	Andy
			D40	Don

g. Draw the Entity Relationship diagram for this database

One to Many



Makes many

h. Write the Relational Schema for this database.

Movie (**<u>M ID, </u>**Title, Rating)

Director (**<u>D ID</u>**, D Name)

3. On page 110 of your Coronel and Morris textbook, there is a database called StoreCo. Use this database to answer questions 1 through 9. When asked to provide an explanation, a simple explanation is much better than a thesis.

1. For each table, identify the primary key and the foreign key(s). If a table does not have a foreign key, write *None*.

Table name: **Employee**
EMP_CODE is the primary key, STORE_CODE is the foreign key
Table name: **Store**
STORE_CODE is the primary key, and has two foreign keys; EMP_CODE and REGION_CODE.
Table name: **Region**
REGION_CODE is the primary key, No foreign key

2. Do the tables exhibit entity integrity? Answer yes or no, and then explain your answer.

Yes, because the property of a relational table that guarantees each entity has a unique value in a primary key and that the key has no null values.

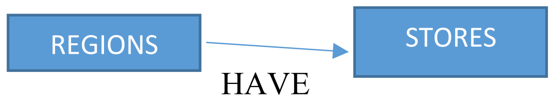
3. Do the tables exhibit referential integrity? Answer yes or no, and then explain your answer. Write *NA* (Not Applicable) if the table does not have a foreign key.

Yes, Table's do, because Foreign keys are used to ensure referential integrity, the condition in which every reference to an entity instance by another entity instance is valid. In other words, every foreign key entry must either be null or a valid value in the primary key of the related table.

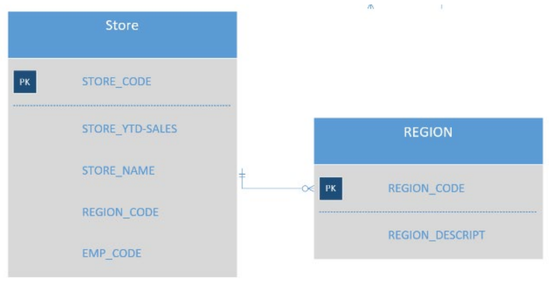
4. Describe the type(s) of relationship(s) between STORE and REGION.

The way that they are associated is STORE'S foreign key is REGION'S KEY, the 1:M relationship is the relational modeling ideal. Therefore, this relationship type should be the norm in any relational database design.

5. Create the ERD to show the relationship between STORE and REGION.



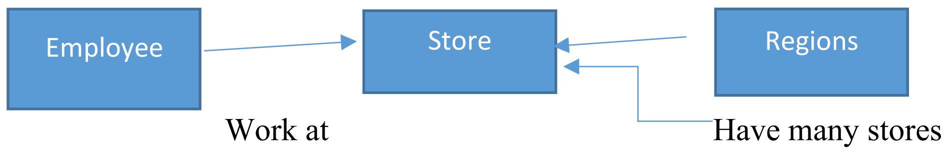
6. Create the relational diagram to show the relationship between STORE and REGION.



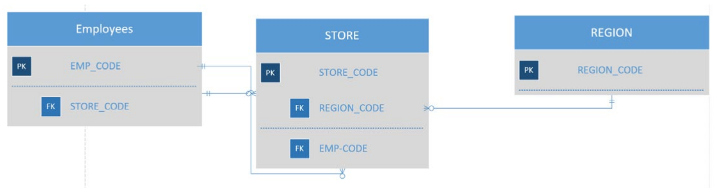
7. Describe the type(s) of relationship(s) between EMPLOYEE and STORE. (*Hint*: Each store employs many employees, one of whom manages the store.)

1:M, Store employees one manager, M:1, many employees work at one store, M:N, Regions have many stores.

8. Create the ERD to show the relationships among EMPLOYEE, STORE, and REGION.



9. Create the relational diagram to show the relationships among EMPLOYEE, STORE, and REGION.



4. On page 112 of your Coronel and Morris textbook, there is a database called TransCo. Use

this database to answer questions 17 through 23. Again, when asked to provide an explanation, a simple explanation is much better than a thesis.
17. For each table, identify the primary key and the foreign key(s). If a table does not have a foreign key, write *None*.

Table Name: **TRUCK**, Primary Key: TRUCK_NUM
BASE_CODE & TYPE_CODE: Foreign Keys
Table Name: **Base**, Primary Key: BASE_CODE, Foreign Key: None
Table Name: **Type**, Primary Key: TYPE_CODE, Foreign Key: None

18. Do the tables exhibit entity integrity? Answer yes or no, and then explain your answer.

Yes, because the property of a relational table that guarantees each entity has a unique value in a primary key and that the key has no null values.

9. Do the tables exhibit referential integrity? Answer yes or no, and then explain your answer. Write *NA* (Not Applicable) if the table does not have a foreign key.

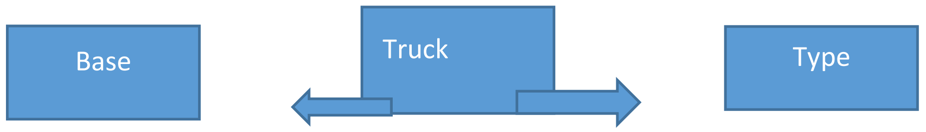
Table Name: **TRUCK**, Yes, because Foreign keys are used to ensure referential integrity,
Table Name: **BASE**, *NA*
Table Name: **TYPE**, *NA*

20. Identify the TRUCK table's candidate key(s).
Table Name: **BASE**, BASE_CODE:
Table Name: TYPE, TYPE_CODE:

21. For each table, identify a superkey and a secondary key.

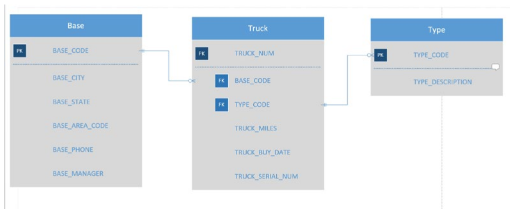
Table Name: **TRUCK**, *NA*
Table Name: **BASE**, BASE_CODE: Superkey, BASE_AREA_CODE: Secondary Key
Table Name: **TYPE**, TYPE_CODE: Superkey, TYPE_DESCRIPTION: Secondary Key

22. Create the ERD for this database.



Truck has a base & and a type

23. Create the relational diagram for this database.


Base
PK
BASE_CODE
BASE_CITY
BASE_STATE
BASE_AREA_CODE
BASE_PHONE
BASE_MANAGER
Truck
PK
TRUCK_NUM
FK
BASE_CODE
FK
TYPE_CODE
TRUCK_MILES
TRUCK_BUY_DATE
TRUCK_SERIAL_NUM
Type
PK
TYPE_CODE
TYPE_DESCRIPTION

3060 Final Review

Specialization Hierarchal

Master Level   R (A B C) Disc Not key attributes

R1 (A D E)   R2 (A F G) Only primary key comes down

Conjoin (here or there)

Overlapping T/F T/F Boolean   Overlapping must have subtype discriminator

subtype discriminator—The attribute in the supertype entity that determines to which entity subtype each supertype occurrence is related. Multiple Boolean= And / Or
Partial Completeness-can have a null-doesn't have to be in a subclass
Total Completeness- can't have nulls- True / False- at least one has to be true

Desirable properties of PK / 1 column

Design Trap

Chapter 10

Transaction Management and Concurrency Control

Transactions-A request upon the database, or action/transactions don't always change/could just read.

Transaction properties have four main properties: atomicity, consistency, isolation, and durability. **Atomicity** means that all parts of the transaction must be executed, 4 steps; otherwise, the transaction is aborted. **Consistency** means one consistent state to another, and **isolation** means that data used by one transaction cannot be accessed by another transaction until the first one is completed. **Durability** means that changes made by a transaction cannot be rolled back (undone). In addition, transaction schedules have the property of **serializability**—the result of the concurrent execution of transactions is the same as that of the transactions being executed in serial order. (the same seats cannot be resold).

ACIDS Test-

Logical transaction- the user views the data

Physical transaction- put into data base, system person

Graphic data base / Google maps

Data Analytics

Explanatory Analytics / what's the relation?

Predictive Analytics / how much to order?

Flow chart / data prep, repository

Prognosis Phases / forcast

Big Data - Java base

Big file, binary files, jpg. Left in natural form

Volume- How much? Scale up, (upgrading) scale out (add nodes)

Velocity- Speed of stream processing (inputs) filtering mechanism

Variety- (structured stars) predetermined form

Variability- determine meaning in data

Intelligence- the ability to understand language

Sentiment analysis- attempt to read passages, take that data and make sense of it

Veracity- trustworthiness of data

Visualization- Present graphically, good characteristics

Polygot Persistance- are going to use a variety of ways to store and retrieve data

Hadoop- high volume of data, lots of storage

Streaming access / intolerant

3 Types Nodes

Client- receive, execute info

Main – Has all of data on specific things

Data Node- gets the data from node

Block Reports- In the Hadoop Distributed File System (HDFS), a report sent every six hours by the data node to the name node informing the name node which blocks are on that data node.

Map Reducer- Mapping out products, Reducer infinite
An open-source application programming interface (API) that provides fast data analytics services; one
of the main Big Data technologies that allows organizations to process massive data stores.

Map Reducer- request or query
Hadoop- storage
Flume Scoop- extracts from data base to Hadoop
Pig- scripting language to get results from Map Reduce
Hide- data warehouse producing
Injestion- get data in
Plume- extracts data