01-03-design-intro-notes

# MODULE 01 NOTES

**INTRODUCTION TO**

**RELATIONAL DATABASE DESIGN**

**INTRODUCTION TO RELATIONAL DATABASE DESIGN**

These notes are not to be used in other courses, by other instructors, or in other Colleges without the permission of all relevant authorities. The notes are intended for Seneca College students enrolled in DBS201

**PREAMBLE**

.

These notes are meant as a supplement to the class discussions. The class discussions will allow you to further clarify the explanations. The design process in class may have a slightly different answer or may take a slightly different approach depending upon student class participation. The methodology used on the course currently to solve design problems will differ from the remaining set of notes. The ***more methods you see and use the better you will develop the skills needed to solve relational database*** design problems. That is what experience does.

In some of the sample problem cases there are solutions available. Since there is no organization to ask questions to directly, the students or instructor may suggest some other assumptions and a solution may change a bit to suit the different requirements. The design of the database should reflect the desires of the business or organization and not the assumptions of the designer. Please advise me (Ron Tarr) of any errors, or better ways of presenting the material.

If you miss classes you can stay up to date using these notes and other resources. There are lots and lots of resources available on the internet. The more sources you read the better. All sources will have a different way of explaining some ideas.

CAUTION: unfortunately, some of the examples available on the Internet are incorrect without further explanation. Some of the examples lack this further explanation or understanding of what the business wanted. The result being that you may misinterpret something that you have read. Please bring anything you don’t understand to class with you and we will try and solve the problem together.

Credit for some of the materials in this course should go to a number of Professors at Seneca College who wrote notes on design and added cases. Two that come to mind are Carol Terentiak and Patricia Belvedere.

**WHAT IS A DATABASE**

ASIDE: The file that holds the OH has additional data or another approach to the notes

A database is a collection of interrelated data, which can be shared by many users. When we refer to a database, we are referring to the stored data. Therefore **the database is the form the data is stored in.** Another data storage system is file-oriented. Word documents, pictures and programs you may have written are stored as files (txt, jpg, mp3, and doc). Each of the storage formats within the file will be different. So a database is just another data storage format.

To run an organization whether it is a large business, a school or even a small operation requires that information or data is stored. Minimum types of data stored would be sales and expenses or in the case of the College, incoming funds (tuition, grants) and outgoing expenses (rent, wages, utilities). For any reasonable size operation, again using the College for an example, there is a huge amount of data kept on employees (faculty, support, administrative, management, tutors, technicians etc), students (full-time, part-time, summer camps), subjects, programs, facilities, suppliers, utilities and a lot more. All of these “things” that an organizations wants to keep details on are known as entities (more explanation later).

It may not appear that in some cases there is a relationship between all the data stored in an organizations database. It may appear that there is no relationship between students and suppliers of cleaning supplies. There isn’t a direct relationship but there exists the relationship of incoming funds from students (tuitions) and outgoing funds (payments of invoices). These relationships exist through other data about banking or accounting.

A database then is about how **the data is stored and the relationships between the entities** or groups of stored data.

# WHAT IS A DBMS

To manage all the data and the relationships requires a lot of software. Software to keep track of the entities (tables), how they relate to each other, how to get data into and out of one or more tables, how to ensure the integrity or quality of the data, how to control access to the data to certain users, how to prevent data corruption, backup, recovery, and a LOT more. A DBMS (Database Management System) is the software that manages the database. This software provides the interface between the user and the data.

Examples of DBMS's would be

3 major players are: ORACLE DB2 Microsoft SQLServer

Other you might hear about:

SYBASE -- was a major player in 90’s but market share dropping

PEOPLESOFT, SAS -- have smaller market shares

MySQL -- often heard about, 2nd largest user base, but not a major corporate player,

-- seen a little more often on some Linux platforms in parts of corporations

-- was bought out a few years ago by Oracle

ACCESS -- Microsoft’s small personal or departmental size DBMS for PC’s

-- Microsoft pushes SQLServer in a corporate environment

There are others, but for large size operations Oracle, DB2 and SQLServer are the big ones.

# Graphical Portrayal of DBMS

USER – i.e. end-user, programmer, developer

**DIFFERENCES -- DATABASE vs STANDARD FILE SYSTEM**

OPERATING SYSTEM

STORED DATA

DBMS

PROGRAMS

**PLUS -- what does the DBMS supply**

1 **The description of the data is stored *once* as part of the database, independent of the application program using it.** (CREATE TABLE command defines the table, the columns, data type etc) For example the data STUDENT NAME may be described as character data of varying length to a maximum length of 60. SCORE may be a numeric field of maximum length 3. The DBMS stores that data and provides some checking to ensure that a value 7892 cannot be stored in SCORE because the length of the data exceeds 3 numbers as defined above. Further limitations can be put on the data and the DBMS does all the checking to ensure the integrity of the data.

In 3GL languages like C (orin the 90's COBOL) the description of the data is described in the application program (If you have taken C you define the data in each program with statements like *int variable-a, variable-b* or the *fscanf* function that defines the data type and length). Each program that accesses the data in a file must describe what the data looks like to the program. If there are 200 programs accessing student data then each has the description of the data in the program. Make a change to the data such as adding a field and the programs that access this data must also be changed. That is a lot of work in a business environment to make those types of changes. With a relational database only the programs that use the added field need to be changed. As another example, if there are restriction that a score must be between 345 and 789 then each program that allows scores to be changed or inserted must have the logic to restrict the values of score built into the program. The DBMSon the other hand will handle this logic for you once it is defined.

2 **The relationship between data are maintained by the DBMS rather than the programmer**

If there is a file of customer data (customer id, address, contact person, salesperson id etc) and there is a file of orders (order id, customer id that placed the order ... etc) then the application program must describe both file contents (see 1 above) and do the logic that relates the contents of the two files together through customer id. In relating the 2 files we can get the customer and address with all the order numbers the customer has. Actually to see all the orders and the details about the orders may require joining a lot more files such as order-line file and product file. Through the common field customer id, the DBMS can handle the connection and reduce the amount of programming.

3 **The DBMS comes with application development software, such as a 4GL to develop applications coupled to screen development software; this is in addition to the standard SQL.**

The data in the database can be accessed via the 4GL language SQL. SQL is an industry standard to access relational databases.

Suppose the user wanted a list of student names and the date of birth from a student file (student file contains: student id, name, address, phone, date of birth, gender, OHIP #, Social Insurance Number, and more). A program written in C would need to define the variables or fields, open the file storing the data and loop through each student selecting out the two fields required, displaying or writing a report. In SQL the code would look like the following.

SELECT NAME, DATEOFBIRTH 🡨 this line tells the DBMS what to select

FROM STUDENT; 🡨 this line tells the DBMS what table to find the data in

The DBMS knows where NAME and DATEOFBIRTH are located in the table. Remember the table has lot more data than those two pieces of information about each student. The data descriptions are held by the database. Notice that the programming is reduced. There is no error of defining the name size as 20 when it is 25. Again, the DBMS has all that information.

However, like all languages it has limitations. In simple form SQL was meant to retrieve and manipulate data. It was not meant for complicated data manipulation or fancy displays.

In order to do forms (window-like screens) and reports for the users or to do other data manipulations normally required by a business, the DBMS comes with additional software. When you purchase Microsoft Office it comes with word processing, spreadsheet and other software components. The DBMS also has several components and one of them is a 4GL language that will allow for rapid application development. In the case of Oracle the language is PL/SQL. There are screen and report designer software.

4 **Software is also provided to handle SECURITY, BACKUP, and RECOVERY**

5 **Programming Interface** that allows the use of existing programs written in languages like Java, COBOL, C, C++, RPG, Visual Basic to access the data in the database, by changing the I/O statements.

The existing programs will still work. As an aside, one of the reasons COBOL still exists even though the organization stores data in a database is that the instead of the programs accessing files the programs were modified slightly to retrieve the data from a database. The SQL would "read/retrieve" the data from the database insert it into the variables and the COBOL program would continue to manipulate the data the same way it always did. This required no significant change to the logic of the program. This programming interface is referred to as a pre-compiler.

6 **Data Dictionary.**

7 **Costs a lot more.** With all the additional software that handles a lot of what programmers used to do, there is a charge. Something like ACCESS is cheap but it won’t run a business of any size. For huge operations the DBMS software can run into the millions of dollars as well as huge amounts of yearly maintenance fees to support the software.

NOTE: The database is only one component of the DBMS although the term database, when used, often means the DBMS. Conversely the term ORACLE is really the DBMS, but it is often used in an ORACLE environment to mean the database.

**PRACTICE WORKSHOP**

**Step 1 – Design a record structure**

Think of a traditional file processing system that would keep track of a student's data. For example, consider the basic data such as name, address, phone etc. needs to be kept. Also assume that you want to keep track of the student's course data, which includes the course code, course name, professor's name and office number, as well as the grade obtained in the course.

For this exercise we could use a spreadsheet as a way of organizing he data or it can be done on paper.

**Design a record structure that would store the data required.**

### **Step 2 - Adding another record structure to the structure just designed.**

Assume that the college hires students to work in the labs. Create another record structure that would store the student's data in an employee file with name, address, phone, salary, start-date etc.

**Design a record structure that would store the data required.**

**Step 3 – Add sample data**

Create test data for yourself and two of your friends. Each student will have data for 3 similar courses, from the same professor.

Create test data for the employees file for one of your friends, as they have been hired as a tutor.

**Step 4 – Identify redundant data**

Identify the redundant data within the test data you have created. What problems exist with storing and maintaining this data?

**BEGINNING OF A SOLUTION**

### PART 1

Developing a record layout for STUDENT might look like:

NAME

ADDRESS

PHONE

CRS-CODE

CRS-NAME

PROFNAME

PROFROOM

GRADE

Do the layout as columns of a spreadsheet and add sample data.

Sample record layout STUDENT

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Address | Phone | Crscode | Crsname | Profname | Profroom | Grade |
| Tarr | 1 Yonge | 905-222-1234 | CCS125 | Intro to C Programming | Walton | 2170 | A |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

Now add some extra data and the spreadsheet looks like this:

Adding some data

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Address | Phone | Crscode | Crsname | profname | profroom | Grade |
| Tarr | 1 Yonge | 905-222-1234 | CCS125 | Intro to C Programming | Walton | 2170 | A |
| Tarr | 1 Yonge | 905-222-1234 | ACC101 | Intro to Accounting | Fuzz | 2120 | A |
| Tarr | 1 Yonge | 905-222-1234 | SYS101 | Intro to Systems | Langer | 2555 | C |
| Anders | 2 Avenewt | 416-491-5050 | CCS125 | Intro to C Programming | Walton | 2170 | B |
| Anders | 2 Avenewt | 416-491-5050 | SYS101 | Intro to Systems | Langer | 2555 | C |
|  |  |  |  |  |  |  |  |

Developing the record layout for EMPLOYEES might look like:

NAME

ADDRESS

PHONE

SALARY

START-DATE

DATE-OF-BIRTH

Again add some sample data to the columns.

Sample record layout EMPLOYEES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Address | Phone | Dateofbirth | Startdate | Salary |
| Tarr | 1 Yonge | 905-222-1234 | 1980 Dec 20 | 2001 May 9 | 12.00 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Going back to the student spreadsheet below**, notice the amount of duplication is considerable**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Name | Address | Phone | Crscode | Crsname | profname | profroom | Grade |
| Tarr | 1 Yonge | 905-222-1234 | CCS125 | Intro to C Programming | Walton | 2170 | A |
| Tarr | 1 Yonge | 905-222-1234 | ACC101 | Intro to Accounting | Fuzz | 2120 | A |
| Tarr | 1 Yonge | 905-222-1234 | SYS101 | Intro to Systems | Langer | 2555 | C |
| Anders | 2 Avenewt | 416-491-5050 | CCS125 | Intro to C Programming | Walton | 2170 | B |
| Anders | 2 Avenewt | 416-491-5050 | SYS101 | Intro to Systems | Langer | 2555 | C |
|  |  |  |  |  |  |  |  |

Other factors

Employees have benefits, such as long term disability, life insurance, dental plan and pension plans. All of these companies providing the services need data about employees sent to them on a regular basis. For example the dental plan insurer wants name, address, date of birth, marital status for the employee and also the insurance number of the College and the insurance number of the employee.

Notice that many of the same pieces of data exist in the EMPLOYEES record. The same is again true to meet the requirements of the other insurers.

**Notice that every course taken by a student needs a new record of information**.

That means that all the personal data about the student is repeated, only the course related data is different. If the student takes 20 courses then the name, address etc is repeated 20 times. In the example above only 3 columns of student data is repeated, but there are many more columns of data for example gender, who to call in case of an accident, OSAP, tuition fees paid or outstanding that would be stored. When these other columns are considered the problems of **redundant** data become even worse.

Seneca over the years has had many millions of students (full-time, part-time, summer camps, industry courses). That means there is a lot of duplicate data being stored with this layout.

WHAT IF … instead of the layout above the record was extended so that there were columns for the student and columns for each course they took. Because of the width of the paper the following is a simplified look at that type of layout

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Name | Personal data in several columns | Course 1  In several columns | Course 2 | Course 3 | Course 4 | Course N |
| Tarr | Address etc | CCS125  Info | ACC101  Info | SYS101  Info | Etc | Course 20  Info |
| Anders | Address | CCS125  Info | SYS101  Info |  |  |  |

In each of the columns such as COURSE 1 above it would be 3 columns containing the course code, description and grade for each course taken by the student. That would mean that the personal data of Ron Tarr is recorded once and data about each course is recorded once only.

If Ron Tarr took 20 courses but Sharma Anders took 2 courses space would have to be left for the maximum number of courses a student can take. In this example Anders would have 18 course spaces empty. This is a large waste of storage space.

Look at the employee record layout below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Name | Address | Phone | Dateofbirth | Startdate | Salary |
| Tarr | 1 Yonge | 905-222-1234 | 1980 Dec 20 | 2001 May 9 | 12.00 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Notice that the employee data has a lot of the same fields as the student data. If a student can be an employee or an employee can also be a student, then data about the same person is duplicated.

The next section elaborates on the problems of traditional file systems.

## PROBLEMS WITH TRADITIONAL FILE SYSTEMS

(Some background reading)

Because of the nature of storing the data definitions within the programs, programs and files became tightly locked together. In order to make changes to an application system, programming modification was required. In a business operation it isn’t easy to just change a program. The business needs to approve any changes that someone wanted, the IT department needs to be contacted about the change, a programmer needs to be available to make the change (they are not sitting around waiting for someone to give them work) , the program gets changed, the program changes need to be tested as an individual program, then tested again within the system (system testing) before it can be put back into production. All changes had to be carefully logged. Even simple changes required a lot of time and care in order to avoid a business making errors in their day-to-day operations.

With a traditional application approach, each application kept its own data and tended to carry on processing tasks, independent of each other. This obviously led to duplication of data which in turn led to wasted maintenance time, disk space and processing time. In addition redundant data led to inconsistency among the duplicate data. For instance, suppose the organization stored the address in the Employee file. It may also have a file with the same name and address and other information, which is used to notify the insurance company of who is currently entitled to Long Term Disability or Life Insurance. The same may exist in another file for the provider of Dental Insurance. Another set of similar data may be retained to meet union member lists. If the employee also enrolls in a course at the College, much of the same data will be kept in the student files.

Suppose the employee was also lucky enough to have a product or service that the College would like to buy. Much of the same data is again stored in a file of supplier information. You can appreciate quite quickly the duplication of effort to simply gather and load the data for the first time, much of which is duplicate information. Now, imagine if that same individual sent in a change of address. Who do they send the change to?? Payroll?? Hopefully, they will update employee information plus all the other benefit companies that need the change. Well, that's a good choice, but that will only trigger the change in Payroll’s set of files. Chances are, the other files like student registration files will remain as is, leading to inconsistencies. Then the question appears some time later -- which of the many address locations contain the correct data???

Companies or organizations store a lot of data about their customers. Mining that data for information useful to improve or grow the information is difficult if the data is in a lot of different files and the consistency or accuracy of the data is in question.

ASIDE: DBMS’s certainly go a long way to handling ad hoc queries. There is also a tremendous growth today in Business Intelligence tools that can “mine” the existing data to answer more key business questions and to be able to analyze the data in many ways. See also the Data Warehousing course.

**THE DBMS ANSWER**

The obvious solution to the problems we have seen is to store the data only once. Having done that, access must be available to the users of the data, whether it is an application program or an end user on a terminal. Often a user friendly 4 GL is provided to allow ease of access.

The DBMS controls the access to the data. With this approach application programs and/or users no longer can read or write directly with the data files but must go through the DBMS.

The DBMS will determine the necessary relationships to bring the data to the user as specified.

The DBMS also serves to enhance the user's access to the data by letting the user specify "WHAT DATA IS REQUIRED" rather than "HOW DOES THE DATA LOOK AND HOW DO WE GET IT". The **data definitions are stored within the database** and so the user is not concerned with the storage criteria, only the data name. (in C programming and other 3GL type languages the data or structure is defined in the program) The DBMS controls the access to the data, thus relieving the user from this task.

The SQL statement to retrieve customer id, name and address from the database may look as follows:

SELECT CUSTOMER\_ID, CUSTOMER\_NAME, ADDRESS

FROM CUSTOMER\_TABLE;

The SQL statement does not concern itself with the width of customer id or whether the datatype is character or numeric. It does not care if address starts at position 30 or 200 in the data for a customer. All of those concerns are handled by the DBMS freeing the programmer to decide what data is required.

To study the DBMS ANSWER to data processing problems means understanding the advantages that are associated with using a DBMS. Also, a study of database design is imperative. To simply take old application designs and transfer them to a database environment means few benefits would be realized.

Actually good design is good design. The problem with transferring older systems over is that *good design was often compromised in older systems*. For example opening, accessing and closing multiple files to put data together required tremendous processing. So in the past design compromises may have been made in order to gain efficiencies. As machines became faster some of these design compromises could have been eliminated, if the systems were built today. Also every time a language is decided upon to build applications, the language itself has both benefits and limitations.

ASIDE: If we choose the base 10 numbering system then the result of dividing one (1) by three (3) is approximately 0.3. If we were then to multiply the decimal result of 0.3 by 3 we should arrive back at a value of 1 but instead we get 0.9. The result is a 10% difference from the original value of 1. No amount of decimal places (0.33333333) when multiplied back by 3 will give 1. There is an inherent flaw in our numbering system. In other words each numbering system has some limitations, as does each programming language chosen.

Below are advantages of a database approach to data processing problems. See if you can explain why each is listed and how each benefits an organization.

## 1 REDUCED DATA REDUNDANCY

Data is stored in one place.

This is not really an advantage of database systems, as good design should account for this redundancy. Part of the apparent bad design of old systems had to do with efficiency, ease of opening multiple files, memory capacity etc. Today systems will often be de-normalized or designs changed to increase overall efficiency.

## 2 CONSISTENCY OF DATA

Again because it is stored in one place

## 3 SHARING OF DATA

Stored in one place -- means must share data

Due to sharing -- more people "can" have access to the data needed

- Not a case of some areas having the latest data

- All data users have the same state of the data

Sharing often adds to the business operations, as more knowledge is possible.

## 4 REDUCED PROGRAM DEVELOPMENT

No definitions required for each application

No re-work with most applications

- Easy to change field widths

- Easier to drop columns or add columns without effecting many applications

4GL for faster programming

Development tools - Screen designers

- State what you want NOT how to program it

## 5 REDUCED PROGRAM MAINTENANCE

Expand a field size. Do it once in data dictionary and all applications using the filed will have the expanded version. You have less to worry about. Fields on forms will scroll if the data is larger than the window.

Change field width on a screen. Simply click and stretch the field similar to stretching the open windows on the PC.

Change an application. You must test the application to ensure reliability, but you do not have to do module or system testing. This results in tremendous time savings.

Get it wrong. It is easier to re-write it correctly.

## 6 ENFORCEMENT OF STANDARDS

One name is used for all applications. It is called CITY and not capital city in one application, towns, villages, city, cities or locations in other applications

Validation stored in the data dictionary. All applications using the field have the same validation. A field called CHOICE has defined values of Y or N then all applications have the same validation rules built in. There is no need to code it in EVERY application.

Standards require one person to enforce the rules – the DBA.

## 7 DATA INDEPENDENCE

Data does not 'belong' to a program. Programs do not need to define data to use it. If you move the order or location of data in a flat file system, such as id, name, address becomes name, id, address then all programs accessing the data need to change their definitions of the data they are looking at.

Add a column to a table and all selects (as part of a stored program) that were previously written are not affected

EXAMPLE SQL that already exists asks for this data to be displayed

SELECT C.CITY, C.STATE, C.TEAM\_NAME, S.RIVERS

FROM CITY C, STATE S

WHERE ...

If we add an extra column such as TEAMNAME to the STATE table, since the SQL above does not use that column it isn’t necessary to rewrite the SQL.

## 8 INFORMATION PROTECTION

Create Views, will allow only a restricted access to the data. You can write forms and reports or applications against the view and the data it presents.

In 3GL environment once the file is opened for access, then all data is available to be read. The privilege to open a file allows access to the entire data.

## 9 FASTER USER REQUEST TURNAROUND

Some things are simply easier to do with a database. Suppose a manager wanted a quick report to appear on the screen listing all students in the CPA program. In SQL this request might be answered by this query.

SELECT FIRSTNAME, LASTNAME

FROM STUDENT

WHERE PROGRAMNAME = ‘CPA’;

This is a lot shorter than C can write it.

There is less handling and less maintenance, which means less cost.

**IF DBMS DOES ALL THIS -- WHAT WILL YOU DO**??

**DESIGN**

**VALIDATE LOGIC IN DATA DICTIONARY**

**SECURITY**

**ANSWER USER REQUESTS**

**DEVELOP SCREENS**

... And lots more

Do not worry, you are not out of a job, the job just continues to change.

**DISADVANTAGES TO A DBMS**

Like programming languages that have advantages over others, there are also inherent limitations or disadvantages. C has advantages over COBOL in some areas and COBOL has advantages over C in other areas.

The DBMS is an operating environment and as such the functionality and capabilities of the software is complicated, similar to an operating system environment. That means in a large corporation there will be specialists who manage the DBMS. To take advantage of power of relational databases and the DBMS software requires good design, good implementation of the design, efficient set up of the entire operations, security, backup forms and reports developers and a lot more. The more complex the system the greater the potential exists for major problems if a poor decision is made.

There are other disadvantages, but it can be seen from the popularity of relational databases as a way to store information whether it is accessed within the organization or over the web proves the advantages far outweigh the disadvantages.

ASIDE: Some issues like the space devoted to the DBMS depend upon the environment. If the exposure to DBMS is through a product like ACCESS, then the operation tends to be small and the DBMS size would be relatively small.

ASIDE:

There were databases back in the 1960’s that are still used today. This is very much like COBOL it isn’t going away.

Object Oriented Databases (OODBMS) began in 1987, although relational databases were introduced heavily from the late 80’s through the 90’s.

No discussion is made of the Hierarchical or Network databases that proceeded relational databases.

## HOW TO FIX UP THE EXERCISE

(May be skipped as it will be done better in later notes)

**The following is overly *simplistic*.**

**LOOK FOR 🡪ALL RELATED DATA BEING REPEATED**

**AND 🡪GROUP IN A SEPARATE GROUPING**

**GIVE A NAME TO THE GROUPING**

**A PULL OUT COURSE DATA**

COURSE [CRS-CODE, CRS-NAME, PROF-NAME, ROOM]

### B PULL OUT ADDRESS INFORMATION

PERSON [NAME, ADDRESS, PHONE]

### C EMPLOYEE IS ALREADY IN A SEPARATE GROUPING

### D ASK HOW DO WE CONNECT DATA

COURSE -- to --> STUDENT

Via CRS-NAME or CRS-CODE

Therefore STUDENT -- must contain CRS-CODE

STUDENT – to --> PERSON

Via NAME or STUD-ID

Therefore PERSON and STUDENT must contain NAME

EMPLOYEE – to --> PERSON

Via NAME or EMP\_ID

Therefore EMPLOYEE and PERSON must have NAME

Where does GRADE go??

### E RESULTING LAYOUT

PERSON [NAME, ADDRESS, PHONE]

COURSE [CRS-CODE, CRS-DESC, PROF-NAME, ROOM]

EMPLOYEE [NAME, SALARY, START-DATE]

STUDENT [NAME, CRS-CODE, GRADE]

\*\*\* UNDERLINES are the connections between each table

The next modules cover the rules that help you make good design