

Computer Science 3319 – Databases – Course Notes

Week 1

(Week 1 - Video Topic #1 - Introduction)

- Data = Known facts that can be recorded and have an implicit meaning.
- Database = A collection of related data.
- Mini-World = A part of the real world about which data is stored in a database. E.g. medical office, a retail business, a library are all mini worlds. A database is a model of a mini world.
- Database Management System (DBMS) = A software package/system used to create and maintain a computerized database.
 - *Why is a DBMS file BETTER than a flat file for storing data?*
 - Reduced data redundancy.
 - Able to isolate programs from data.
 - Provides persistent storage of data between execution of programs.
 - Multiple users can use and share data.
 - More abstract conceptually.
 - *Why is a DBMS file LESS CONVENIENT than a flat file for storing data?*
 - Flat Files are free.
 - DBMS requires trained staff.
 - DBMS has initial complexity.
- Database System = The DBMS software together with the data itself. Sometimes, the applications are also included.
- Flat File = Stores a database in a plain text file.

(Week 1 - Video Topic #2 – Who is Involved?)

- DBMS Roles:
 - Database Administrator (DBA):
 - Responsibility 1 – installs, updates, upgrades, and backs up the DBMS.
 - Responsibility 2 – monitors performance and security of the DBMS.
 - Database Designer:

- Responsibility 1 – collect all the data, and identify it and its relationships.
- Responsibility 2 – create a model of the data and structure it.
- System Analysis and Application Programmer:
 - Responsibility 1 – develop the specs for the system.
 - Responsibility 2 – develop software that implements the specs, and test it.
- End User:
 - Responsibility 1 – Access the database by generating reports.
 - Responsibility 2 – Access the database to update, maintain and query it.

(Week 1 - Video Topic #3 – Three Schema Architecture)

- Schema = describes a database but not the actual data itself.
- Instance = an occurrence of a data item described in the schema.
- State/Snapshot = the data in the database at a moment in time.
- 3 Schema Architectures:
 - Internal Level – physically stores the database (Ex hard disks).
 - Conceptual Level – create the columns and rows that create and store the database.
 - External Level – programs that access the data that is stored in the database, and takes only specific data and hides the rest.
- Logical Data Independence = ability to change the conceptual schema without affected the external schema or views (changing aspects that will only affect one program not all using the data).
- Physical Data Independence = ability to change the physical schema without changing the conceptual schema.
- You want both independences.




(Week 1 - Video Topic #4 – Modeling & Intro to ER Diagrams)


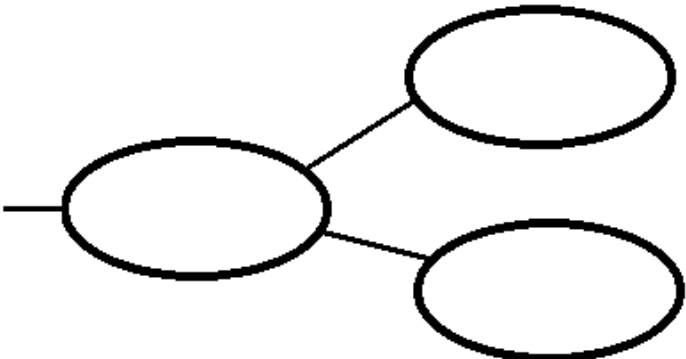

- Check what data reports are typically made for the type of database you are designing.
- Narrow your scope to minimize redundancy.
- Entity Relationship Model (ER Diagram) = a visualization of your data. Created by Peter Chen in 1976.
- Use:
 - Microsoft Visio.

- <https://www.draw.io/>
- <https://cloud.smartdraw.com/>

(Week 1 - Video Topic #5 – ER-D's – Creating Entities)

- Entity = a single “thing” that exists independently (Ex employee).
- Attribute = describes a “thing” (Ex salary).
- Domain/Value = value taken on by an attribute (Ex name).
- Multi-valued Attribute = attributes that can have multiple values (Ex degree, or grades).
- Single-valued Attribute = attributes that only have a single value (Ex age).
- Composite Attribute = attributes that can be broken down (Ex full name).
- Simple Attribute = attributes that are simple... (Ex 45).
- Derived Attribute = an attribute derived from another attribute (Ex age from birthday).
- Stored Attribute = an attribute stored at base (Ex birthdate).
- Null Values = a value that may not be known or applicable to all situations (Ex an apartment number when they live in a house).
- Key = a value that makes an entity unique.

ER Type:	How to draw it?
Entity Type	
Attribute	
Key Attribute	
Multi-Valued Attribute	

	
Composite Attribute	
Derived Attribute	

(Week 1 - Video Topic #6 – ER-D's – Representing Relationships)

- Relationship = a named grouping of entities.
- Relationship Set = an ordered list of entity sets.
- Relationship Type = among n entity types, E1, E2... En defines a set of associations among entities.
- Binary Relationship = between 2 entities. Ex. Student takes a course.
- Ternary Relationship = between 3 entities. Ex. A part is supplied by a supplier, for a project.
- Unary Relationship = between 1 entity and itself. Ex. An employee manages other employees.]
- Cardinality Ratio = the number of relationship instances that an entity can participate in. Typical examples include:
 - One – One (Ex.) = Employee manages a department. A department can only be managed by 1 employee (a manager.
 - Many – One (Ex.) = Employee works for department. One department can have many employees.
 - Many – Many (Ex.) = Employee works on project. Many projects can be worked on by many employees.

Ex 1)



An Artist can paint many paintings, but a painting can only be painted by one artist normally.

(Week 1 - Video Topic #7 – ER-D's – Participation)

- Total Participation = requires the other entity to exist or be valid. (Shown with a double line).
- Partial Participation = does not require the other entity in order to exist or be valid. (Single line).

(Week 1 - Video Topic #8 – ER-D's – Weak Entities)

- Weak Entity:
 - Its key is shown with a dashed underline.
 - Its diamond and box are double outlined.
 - Cannot exist without total participation (double line) with its owner.
 - Its owner has partial participation (single line).

(Week 1 - Video Topic #9 – ER-D's – Semantic Guidelines)

- Use singular names for all entities.
- Try to have the relationships read left to right, and top to bottom.

(Week 1 - Video Topic #10 – ER-D's – Crows Feet Notation)

Zero or More



One or More

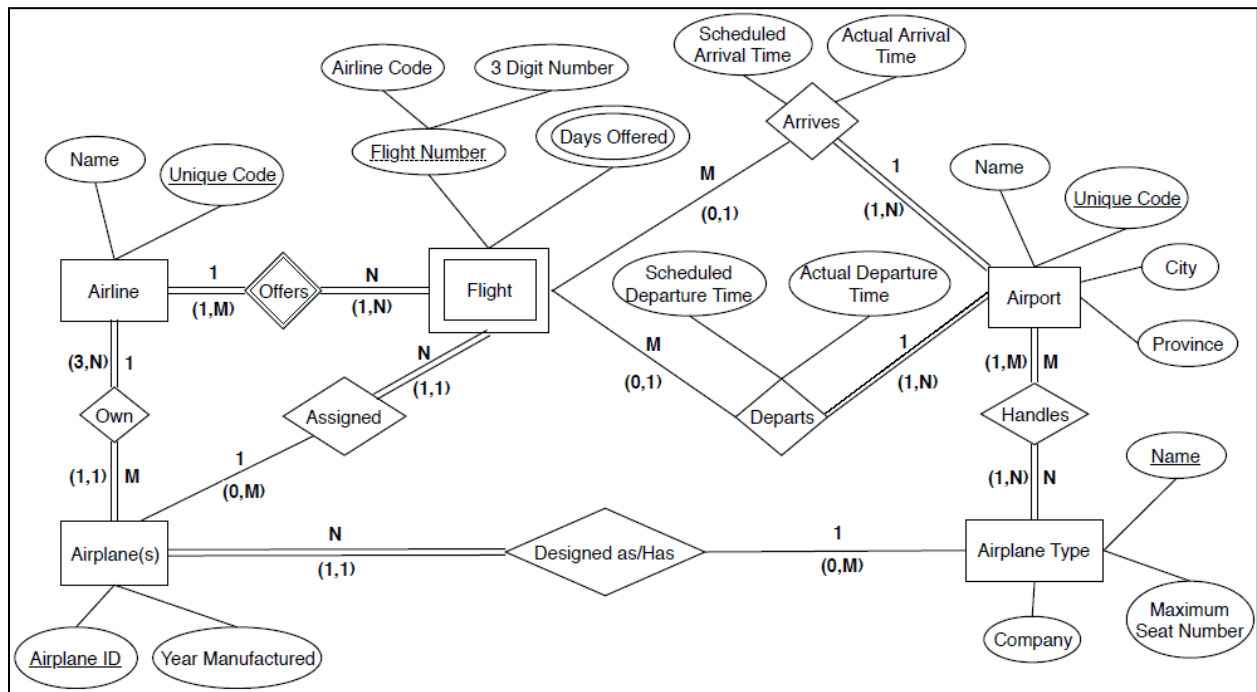


One and Only One



Zero or One





Week 2

(Week 2 - Video Topic #1 – Before the Relational Model/Database)

- Hierarchical Data Model = uses records, record types and parent-child relationships.
 - Uses pointers.
 - Introduced by Codd in 1970.

(Week 2 - Video Topic #2 – The Importance of Keys)

- The key makes a row unique, and identifiable.
- Super Key = any set of attributes that enforces the idea that no tuples are alike.
- Candidate Key = sometimes a table will have 2 possible key. In that case each key is a super key, and the candidate key is a “minimal super key”.
 - Ex. employee number and SSN, each of them is a candidate key.
- Primary Key = pick one candidate key to identify tuples in the relation (signify this key by underling it).
 - This could be a composite key (2 or more attributes combined together).
- Foreign Key = an attribute or set of attributes within one relation that matches the candidate key of some other (possibly the same) relation.

_____ (Week 2 - Video Topic #3 – Relational Database Terminology) _____

QUESTION: If we have:

- $D1 = \{\text{Simpson, Flanders, Smithers}\}$
- $D2 = \{\text{Homer, Ned}\}$
- $D3 = \{40, 30\}$

What would $D1 \times D2 \times D3$ give us:

$\{(\text{Simpson, Homer, 40}), (\text{Simpson, Homer, 30}),$
 $(\text{Flanders, Homer, 40}), (\text{Flanders, Homer, 30}),$
 $(\text{Smithers, Homer, 40}), (\text{Smithers, Homer, 30}),$
 $(\text{Simpson, Ned, 40}), (\text{Simpson, Ned, 30}),$
 $(\text{Flanders, Ned, 40}), (\text{Flanders, Ned, 30}),$
 $(\text{Smithers, Ned, 40}), (\text{Smithers, Ned, 30})\}$

_____ (Week 2 - Video Topic #4 – Properties of Relations) _____

- In a proper relational table, with defines unique keys and names, the order of the tuples does not matter. As they can be changed with view statements.
- In a set, no elements are repeated, making tuples unique.

_____ (Week 2 - Video Topic #5 – Mapping Entities to the Relational Model (RM)) _____

- N/A - barely going to be on exam.

_____ (Week 2 - Video Topic #6 – Mapping 1-1, 1-M Relationships to an RM) _____

- N/A - barely going to be on exam.

_____ (Week 2 - Video Topic #7 – Mapping M-M Relationships to an RM) _____

- N/A - barely going to be on exam.

____ (Week 2 - Video Topic #8 – Mapping Weak and Multivalve Entities + 7 Mapping Rules) ____

- N/A - barely going to be on exam.

_____ (Week 2 - Video Topic #9 – Mapping an ER Diagram to a RM) _____

- N/A - barely going to be on exam.

_____ (Week 2 - Video Topic #10 – Referential, Integrity and Semantic Constraints) _____

- N/A - barely going to be on exam.

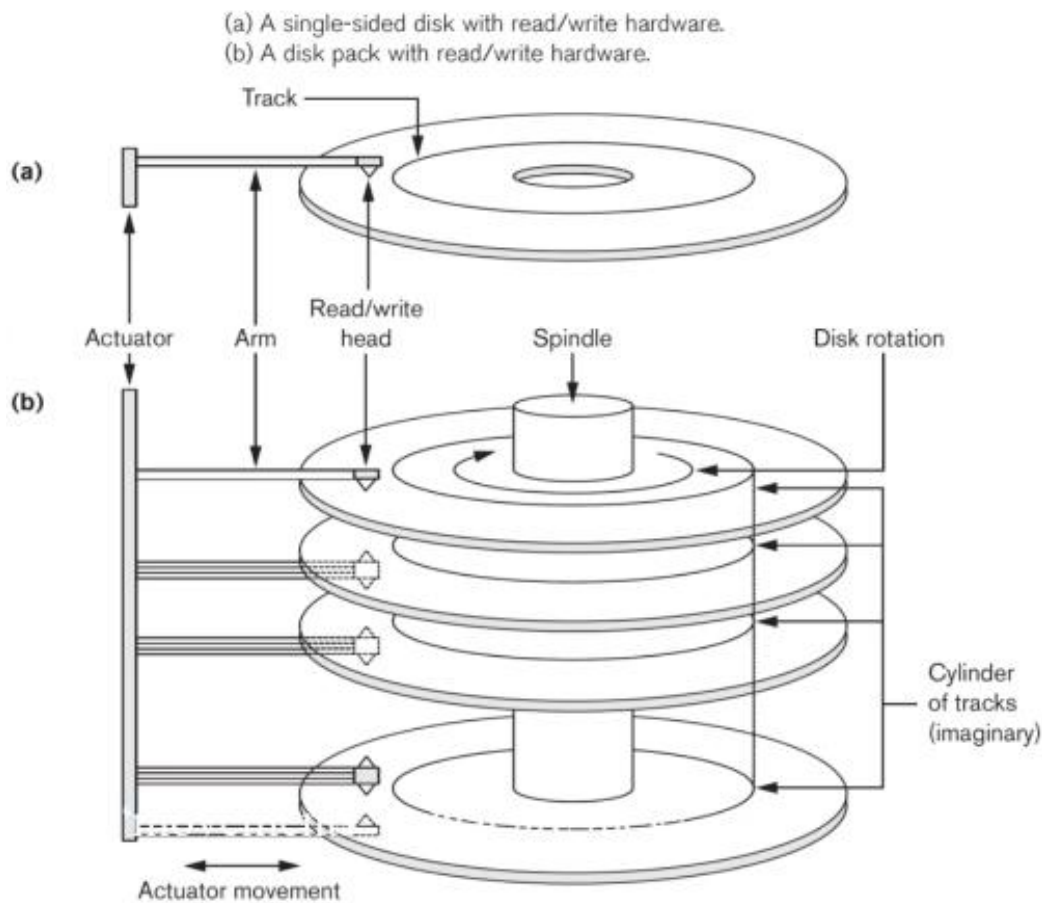
Week 3

_____ (Week 3 - Video Topic #1 – Intro to Secondary Storage) _____

- Primary Storage = CPU (main memory).
- Secondary Storage = magnetic disks (hard drives) and flash memory (SSD).
 - Cheaper than CPU's.
 - Less volatile.
 - Has more memory space.
- Tertiary Storage = optical disks (CD-ROM, DVD) and tapes.
- Primary File Organization = how the file records are physically placed on the disk (ordering of the primary attribute) and how the records can be accessed.
- Secondary Organization = how can we efficiently access attributes OTHER than the primary attribute?

_____ (Week 3 - Video Topic #2 – How Magnetic Disks Work) _____

- How a hard drive is physically built.



- The further away the sector is from the center of the disk, the more data chunks it can store.

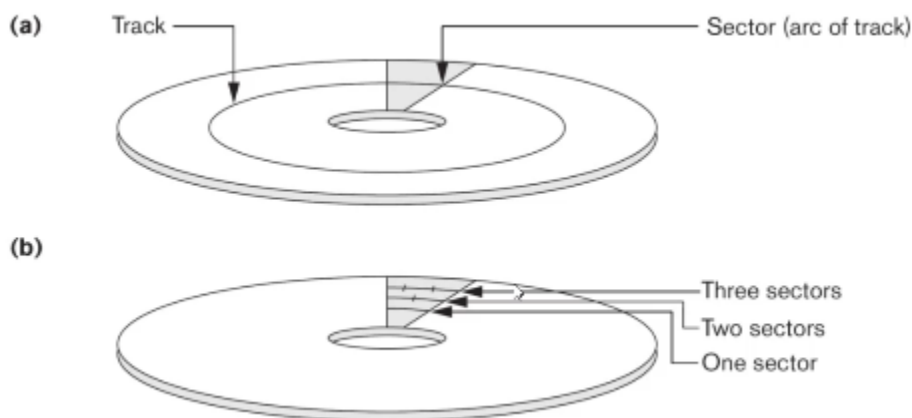


Figure 17.2
Different sector organizations on disk. (a) Sectors subtending a fixed angle. (b) Sectors maintaining a uniform recording density.

- Seek Time = time it takes to move the arm to the correct track.
- Latency Time = time it takes to move the arm to the correct sector on the track.
- Block Transfer = time it takes to move the data.

- Fixed Length = exact amount of bytes associated with each attribute.
- Variable Length = non exact amount of bytes, but instead uses separator characters.
- Block size = maximum number of bytes it can hold (related to the OS).
- Blocking Factor (BFR) = refers to the number of records per block.
- Spanned Record = refers to records that exceed the size of one or more blocks, and therefore span multiple blocks. Uses pointers to link them.
- Unspanned Record = Goes up to only 1 block.

_____ (Week 3 - Video Topic #3 – Adding, Modifying, & Deleting Hard Drive Records) _____

- You can:
 - Open, find, findnext, findinrange, read, insert, delete, modify, close, reorganize, and read_ordered.

_____ (Week 3 - Video Topic #4 – Heap Organization) _____

- Heap = a pile/ unordered file. Each time you get a new record you add it to the end.
 - Easy to add to, harder to modify and hard to delete and then reorder.
 - Good when:
 - Data is loaded in bulk.
 - Not a lot of data.
 - When data has an indexed key.
 - Bad when:
 - Searching, deleting, and sorting.

EXAMPLE

QUESTION: Find the average search time to find a record if you use a heap organization for the following scenario:

- $r = 100,000$ records stored on a disk with block size $B = 2048$ bytes.
- Each record (R) is a fixed size of $R = 500$ bytes.
- Blocking Factor (bfr) = $2048/500 = 4$ records per block (fill in the blank)
- # of blocks needed is $100,000/4 = 25,000$ blocks
- Linear Search: on average $b/2 = 25,000/2 = 12,500$ block accesses

(Week 3 - Video Topic #5 – Ordered Organization)

- Ordering by a specific key, primary or secondary key.
- *Advantage:*
 - Modifying is easy.
 - No sorting required.
 - Binary search by key is easy.
- *Disadvantage:*
 - Inserting and deleting requires complete resorting.
 - Binary search in general is hard.

EXAMPLE

QUESTION: Find the **WORST** case search time to find a record if you are search for a field that the records are sorted on by on the disk:

- $r = 100,000$ records stored on a disk with block size $B = 2048$ bytes.
- Each record (R) is a fixed size of $R = 500$ bytes.
- Blocking Factor (bfr) = $2048/500 = 4$ records per block (fill in the blank)
- # of blocks needed is $100,000/4 = 25,000$ blocks
- Binary Search: **Worst Case:** $\log_2 b = \log_2 25000 = 14.6 (15)$ block accesses

(Week 3 - Video Topic #6 – Hash Organization)

- N/A - barely going to be on exam.

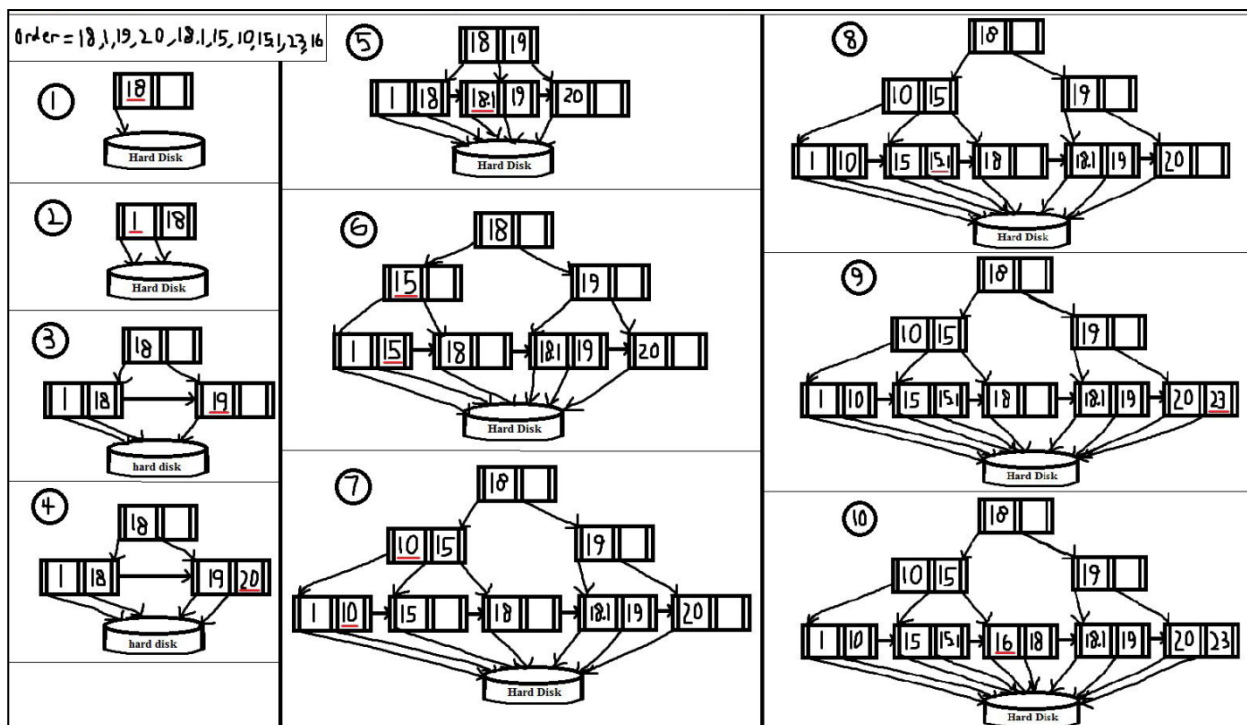
(Week 3 - Video Topic #7 – Introduction to Indices)

- N/A - barely going to be on exam.

(Week 3 - Video Topic #8 – Multilevel Indexes)

- N/A - barely going to be on exam.

(Week 3 - Video Topic #9 – B+ Tree Insertions)



(Week 3 - Video Topic #10 – Time Complexities)

- N/A - barely going to be on exam.

Week 4

(Week 4 - Video Topic #1 – Intro to Relational Algebra)

- Procedural = tells how to get the data.
 - Ex. Relational Algebra.
- Non -Procedural = tells what data is needed.
 - Ex. Relational Calculus.
- Relational Algebra = a theoretical language with operations that lets you perform one or more relations on a table to produce output.
 - Closed Language = if the output produced from a table operation, produces another table.
 - Basically if you produce a result that's of the same type as your data.

(Week 4 - Video Topic #2 – Projections & Selections)

- Unary = works on 1 relation only.

- $\sigma_{condition}(R)$ = Selection (Unary) – returns only the tuples from a relation that satisfy the specified condition (a row subset).
 - Ex. $\sigma_{(ProjectLocation = 'Toronto' \text{ or } ProjectLocation = 'London')}(Project)$
- $\pi_{attribute1, attribute2}(R)$ = Projection (Unary) – returns only the requested attributes (a column subset, with no duplicates) that satisfy the specified condition.
 - Ex. $\pi_{DeptName}(Department)$
- Cartesian Product = $S \times R$
- Union = $R \cup S$
 - A union of 2 tables returns a table with every row (no repetitions) from both tables.
 - Tables must have the same number of columns and same domain.
- Set Difference = $R - S$
 - Tables must have the same dimensions.
- You can use temporary tables to store info and union them after.
 - Ex. $Temp1(Field1) \leftarrow \pi_{Gender}(Employee)$
 - Ex. $Temp2(Field2) \leftarrow \pi_{Name}(Employee)$
 - $Result \leftarrow Temp1 \cup Temp2$

(Week 4 - Video Topic #3 – Union & Difference)

- You can use temporary tables to store info and union them after.
 - Ex. $Temp1(Field_Title1) \leftarrow \pi_{Gender}(Employee)$
 - Ex. $Temp2(Field_Title2) \leftarrow \pi_{Name}(Employee)$
 - $Result \leftarrow Temp1 \cup Temp2$

(Week 4 - Video Topic #4 – Cartesian Product)

- The product of 2 tables gives a table where every row in a new table matches each row in both tables.

Table 1				Table 2			
ID	FirstName	LastName	Age	ID	FirstName	LastName	Age
12	Homer	Smith	24	33	Marg	Jones	28
24	Gene	Simpson	13	24	Gene	Simpson	13
45	Walter	Reid	45				

Table 1 X Table 2							
ID	FirstName	LastName	Age	Table2.ID	Table2.FirstName	Table2.LastName	Table2.Age
12	Homer	Smith	24	33	Marg	Jones	28
24	Gene	Simpson	13	33	Marg	Jones	28
45	Walter	Reid	45	33	Marg	Jones	28
12	Homer	Smith	24	24	Gene	Simpson	13
24	Gene	Simpson	13	24	Gene	Simpson	13
45	Walter	Reid	45	24	Gene	Simpson	13

(Week 4 - Video Topic #5 – Joining Tables Using Inner Joins)

- **Join** \bowtie = a Cartesian Product with a Selection σ to find matches.

- Ex.

Table1				Table2				
A	B	C	D	A	E	C	F	D
7	Cow	Pink	22	6	Cow	Blue	Hat	33
8	Dog	Pink	33	6	Cow	Blue	Sock	44
9	Cow	Red	44	8	Cow	Pink	Shoe	44
				8	Cat	Pink	Hat	33

- **Natural Join** \bowtie = a join where you don't have to put the attributes. This joins the attributes that have the same name in both tables with equal values in both tables. (Or as close to equal). Does not show duplicates.

- Table1 \bowtie Table2

- Ex. Table1 \bowtie Table2

A	E	C	F	D	B
8	Cat	Pink	Hat	33	Dog

- **Equi Join** \bowtie = a join where you specify the attributes to join the tables by. Show's duplicates.

- Table1 $\bowtie_{(attribute1=attribute2)}$ Table2

- Ex. Table1 $\bowtie_{(C=C)}$ Table2

Table1.A	B	Table1.C	Table1.D	Table2.A	E	Table2.C	F	Table2.D
7	Cow	Pink	22	8	Cow	Pink	Shoe	44
7	Cow	Pink	22	8	Cat	Pink	Hat	33
8	Dog	Pink	33	8	Cow	Pink	Shoe	44
8	Dog	Pink	33	8	Cat	Pink	Hat	33

- Ex. Table1 $\bowtie_{(D<D)}$ Table1

A	B	C	D	Table1.A	Table1.B	Table1.C	Table1.D
7	Cow	Pink	22	8	Dog	Pink	33
7	Cow	Pink	22	9	Cow	Red	44
8	Dog	Pink	33	9	Cow	Red	44

(Week 4 - Video Topic #6 – Using Outer Joins)





- **Full Outer Join**  = similar to join, except it includes all the rows from both tables.
 - Even if they don't have a matching value of the column you're joining, if there is no match it put's NULL's in the appropriate columns.

Table1				Table2		
A	B	C	D	B	F	G
12	Red	Can	24	Yellow	Cat	22
24	Red	USA	33	Red	Cat	33
45	Blue	Mex	33	Green	Dog	44
				Red	Dog	24
				Orange	Bird	33

Table1  Table2						
A	Table1.B	C	D	Table2.B	F	G
12	Red	Can	24	Red	Dog	24
24	Red	USA	33	Red	Cat	33
24	Red	USA	33	Orange	Bird	33
45	Blue	Mex	33	Red	Cat	33
45	Blue	Mex	33	Orange	Bird	33
Null	Null	Null	Null	Yellow	Cat	22
Null	Null	Null	Null	Green	Dog	44

- **Left Outer Join**  = a join in which tuples from the 1st table that do not have matching values with the 2nd table, will still appear in the resulting table with NULLS.
- **Right Outer Join**  = a join in which tuples from the 2nd table that do not have matching values with the 1st table, will still appear in the resulting table with NULLS.

(Week 4 - Video Topic #7 – Intersection)

- **Intersection** \cap = creates a new table from the given tables that includes only identical rows from both (no repeats).

(Week 4 - Video Topic #8 – Division)

- **Division** ($R \div S$) = returns a new table that contains only the columns in R, that were not in S, and only the rows from the remaining columns in R matching every single row in S.
 - R MUST have more columns than S.
 - The columns of S must be a subset of the columns of R.

- Ex. Write the relational algebra to find the project names of any projects that also have all the employees working on them that work on the project named "Acct6".

- $$\text{Acct6} \leftarrow \pi_{\text{ProjectNumber}}(\sigma_{\text{ProjectName}=\text{"Acct6"}}(\text{Project}))$$

$$\text{Acct6_Emp} \leftarrow \pi_{\text{EmpSSNum}}(\text{Works_On} \bowtie \text{Acct6})$$

$$\text{ProjNums} \leftarrow \pi_{\text{EmpSSNum}, \text{ProjectNumber}}(\text{Works_On})$$

$$\text{AllAcct6} \leftarrow \text{ProjNums} \div \text{Acct6_Emp}$$

$$\text{Result} \leftarrow \pi_{\text{ProjectName}}(\text{AllAcct6} \bowtie \text{Project})$$

Table AA:				Table BB:	
A	B	C	D	B	D
dog	2	77	pink	1	yellow
dog	3	77	yellow	2	pink
cat	2	88	pink	Table CC:	
pig	1	77	yellow		
pig	5	99	red		
cat	1	88	yellow		
		A	C		
		cat	88		
		owl	66		
owl	1	66	yellow		
owl	2	77	pink		
owl	2	66	pink		

(Week 4 - Video Topic #9 – Relational Algebra Review)

Table 6.1

Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R .	$\sigma_{\langle \text{selection condition} \rangle}(R)$
PROJECT	Produces a new relation with only some of the attributes of R , and removes duplicate tuples.	$\pi_{\langle \text{attribute list} \rangle}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\langle \text{join condition} \rangle} R_2$, OR $R_1 \bowtie_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\langle \text{join condition} \rangle} R_2$, OR $R_1 *_{(\langle \text{join attributes 1} \rangle), (\langle \text{join attributes 2} \rangle)} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2 ; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2 ; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2 .	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

Table Name	Table Layout
EMPLOYEE	FName Minit LName <u>SSN</u> BDate Address Sex
DEPARTMENT	DName <u>DNumber</u> MGRSSN * MGRStartDate
DEPTLOCS	<u>DNumber *</u> <u>DLocation</u>
WORKSON	<u>ESSN *</u> <u>PNO *</u> Hours
PROJECT	PName <u>PNumber</u> PLocation DNum*
DEPENDENT	<u>ESSN *</u> <u>DependentName</u> Sex BDate Relationship

Salary SuperSSN * DNO *

1. Get the first and last name of all the employees.

$\pi_{FName, LName}(EMPLOYEE)$

2. Get all the employees with the first name of "Homer"

$\sigma_{FName = "Homer"}(EMPLOYEE)$

3. Get the first and last name of all employees who work for department 5.

$\pi_{FName, LName}(\sigma_{PNO = 5}(EMPLOYEE))$

4. Get the first and last names of all employees who work for the "Research" Department.

$\pi_{FName, LName}(\sigma_{DNO = DNumber}(\pi_{DNumber}(\sigma_{DName = "Research"}(DEPARTMENT))))$

5. Find the last name and salary and department name of every employee who manages a department

$\pi_{LName, Salary, DName}(\pi_{ESSN = MGRSSN}(\pi_{DName, MGRSSN}(DEPARTMENT)))$

6. Find the first and last name and hours of all employees who work more than 40 hours.

$\pi_{FName, LName, Hours}(\pi_{ESSN}(\sigma_{Hours > 40}(WORKSON)))$

7. Find the first name of any employee who has a birthday the same date as their dependent.

$\pi_{FName}(\pi_{ESSN = SSN \ \& \ BDate = BDate}(DEPENDENT \times EMPLOYEE))$

Week 5

_____ (Week 5 - Video Topic #1 – #5 Virtual Machine Set up & SQL) _____

- N/A

_____ (Week 5 - Video Topic #6 – Intro to MySQL) _____

- To get into mySQL \rightarrow mysql -u root -ppassword
- SHOW DATABASES;
- DROP DATABASE IF EXISTS "database name";
- CREATE DATABASE "database name";

- USE "database name";
- SHOW TABLES
- DROP TABLE "table name";
- CREATE TABLE "table name" (Attribute1 non-null_type, Attribute2 type, Attribute3 type, PRIMARY KEY (Attribute1/2/3);
- SQL is a DDL and DML.
- DDL = data definition language.
- DML = data management language.

_____ (**Week 5 - Video Topic #7** – Inserting, Update & Deleting Rows) _____

- INSERT INTO TableName (Attr1Name, Attr2Name, Attr3Name) VALUES (Value1, Value2, Value3);
- or
- INSERT INTO TableName VALUES (Value1, Value2, Value3);
- UPDATE TableName SET AttrName = 'Value' WHERE Condition;
- Note: if you use single quotes they have to be between double quotes.
- DELETE FROM TableName WHERE Condition;

_____ (**Week 5 - Video Topic #8** – Using Select and Creative Views) _____

_____ (**Week 5 - Video Topic #9** – Implementing Our Case Study) _____

_____ (**Week 5 - Video Topic #10** – Some Final SQL Examples) _____

1. Show the team name of all the teams:

```
SELECT teamname FROM teams;
```

2. Show the city of where all games were played, with no repeats.

```
SELECT DISTINCT gamecity FROM games;
```

3. Show all the data in the officials' table. but show them in order of first name.

```
SELECT * FROM official ORDER BY firstname ASC;
```

4. Show the First and Last Name of all officials from Ottawa or New York.

```
SELECT firstname, lastname FROM official WHERE officcity =  
'Ottawa' OR officcity = 'New York';
```

5. List the last name of all head officials.

```
IN (SELECT headoff FROM game);  
SELECT lastname FROM official WHERE officialid
```

5. List the first and last name and official's city of any official who comes from a city with an "on" in the city name.

```
FROM official WHERE  
SELECT firstname, lastname, officcity  
FROM official WHERE officcity LIKE '%on%';
```

Week 6

(Week 6 - Video Topic #1 – Intro to Relational Calculus)

- In relational calculus we write one declarative statement that states what we want returned, rather than how.
- Nonprocedural Language = all done in one statement.
- In Tuple Relational Calculus:
 - $\{t \mid \text{COND}(t)\}$
 - Where t is a tuple variable and $\text{COND}(t)$ is a Boolean expression involving t . $\text{COND}(t)$ must be true for t in order to return that tuple.
 - Ex: you want to find the first name and last name of employees over 45.
 - $\{t, \text{Fname}, t, \text{Lname} \mid \text{EMPLOYEE}(t) \text{ and } t, \text{Age} > 45\}$

(Week 6 - Video Topic #2 – Joins in Relational Calculus)

- x
- x

QUESTION: Retrieve the birth date and address of the employee whose name is 'Jon Mortensen' assuming this is one of your tables (relations):

Employee					
<u>FName</u>	<u>Minit</u>	<u>Lname</u>	<u>SSN</u>	<u>BDate</u>	
Address	Sex	Salary	<u>SuperSSN</u>	<u>DNO</u>	

ANSWER: {t.Bdate, t.Address |
EMPLOYEE(t) AND t.Fname="Jon" and
t.Lname="Mortensen"}

- X

_____ (Week 6 - Video Topic #3 – Division in Relational Calculus) _____

- X
- X
- X

_____ (Week 6 - Video Topic #4 – Domains in Relational Calculus) _____

- X
- X
- X

_____ (Week 6 - Video Topic #5 – Queries in Relational Calculus) _____

- X
- X
- X

_____ (Week 6 - Video Topic #6 – Intro to EER Diagrams) _____

- X
- X
- X

_____ (**Week 6** - Video Topic **#7** – Mapping an EER Diagram to Relational Tables) _____

- x
- x
- x

Week 7

_____ (**Week 7** - Video Topic **#1** – Intro to HTML) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#2** – Intro to CSS) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#3** – Intro to Java) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#4** – Intro to PHP) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#5** – Creating a Database & Making a PHP Webpage) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#6** – Connecting to MySQL & Error Checks) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#7** – Passing Data to a Webpage & Joins) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#8** – Adding a Table Row with PHP) _____

- x
- x
- x

_____ (**Week 7** - Video Topic **#9** – Adding a Table Column & Image Upload/Display) _____

- x
- x
- x

_____ (Week 7 - Video Topic #10 – How to use PHP Global Variables) _____

- x
- x
- x

Week 8

_____ (Week 8 - Video Topic #1 – Triggers) _____

- Triggers = a set of SQL statements that execute in response to certain events occurring in a table.
- Constraints = impose a limitation execute in response to certain events occurring in a table.

Sample MySQL Trigger

```
delimiter //
CREATE TRIGGER upd_check BEFORE UPDATE ON account
FOR EACH ROW
BEGIN
    IF NEW.amount < 0 THEN
        SET NEW.amount = 0;
    ELSEIF NEW.amount > 100 THEN
        SET NEW.amount = 100;
    END IF;
END;
delimiter ;
```

Parts of the trigger

- 1 TriggerName (upd_check)
- 2 Trigger activation time (BEFORE)
- 3 Triggering event (UPDATE)
- 4 Triggering table name (account)
- 5 Attribute in table (NEW.amount)
- 6 Granularity (FOR EACH ROW)
- 7 Trigger condition (IF NEW.amount < 0 THEN)
- 8 Trigger body (BEGIN ... END;)
- 9 In MySQL need to change the delimiter temporarily.

QUESTION: Assume we have the tables:

- STUDENT(Student_Num, LastName, ...Age)
- UNIV_STATS(Num_of_Students, Total_Age, Average_Age, ...)

Write a trigger that will keep the UNIV_STATS table accurate:

```
CREATE TRIGGER keeptotalscorrectup  
AFTER INSERT  
ON student  
FOR EACH ROW  
EXECUTE PROCEDURE addtount
```

```
CREATE TRIGGER keeptotalscorrectdown  
AFTER DELETE  
ON student  
FOR EACH ROW  
EXECUTE PROCEDURE minusfromcount()
```

(Week 8 - Video Topic #2 – Stored Procedures)

- Stored Procedure = a precompiled application program that executes in response to a single MySQL CALL statement (a function).
- *Advantages:*
 - Faster.
 - Reduces traffic.
 - Reusable.
 - Secure.
- *Disadvantages:*
 - Over usage can slow down an application.
 - Cannot be too complex.
 - Hard to debug.
 - Database migration becomes difficult.

```
01 DELIMITER //  
02  
03 CREATE PROCEDURE `proc_WHILE` (IN param1 INT)  
04 BEGIN  
05     DECLARE variable1, variable2 INT;  
06     SET variable1 = 0;  
07  
08     WHILE variable1 < param1 DO  
09         INSERT INTO table1 VALUES (param1);  
10         SELECT COUNT(*) INTO variable2 FROM table1;  
11         SET variable1 = variable1 + 1;  
12     END WHILE;  
13 END //
```

(Week 8 - Video Topic #3 – Intro to System Tables/Catalogs 1)

- (Information_schema) and (performance_schema) are automatically made and only readable tables.
- Information schema table holds all the databases, a row for each.
- System/Catalog tables allow you to concatenate all the tables and see how many rows each table has.

(**Week 8** - Video Topic #4 – System Catalogs 2)

- x
- x
- x

(**Week 8** - Video Topic #5 – Intro to Data Security)

- x
- x
- x

(**Week 8** - Video Topic #6 – Discretionary Control)

- x
- x
- x

(**Week 8** - Video Topic #7 – Role Based & Mandatory Access Control)

- x
- x
- x

(**Week 8** - Video Topic #8 – Statistical Database Security)

- x
- x
- x

Week 9

_____ (**Week 9** - Video Topic #1 – Intro to Normalization) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #2 – Intro to CSS) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #3 – Intro to Java) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #4 – Intro to PHP) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #5 – Creating a Database & Making a PHP Webpage) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #6 – Connecting to MySQL & Error Checks) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #7 – Passing Data to a Webpage & Joins) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #8 – Adding a Table Row with PHP) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #9 – Adding a Table Column & Image Upload/Display) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #10 – How to use PHP Global Variables) _____

- x

- x

- x

Week 10

_____ (**Week 10** - Video Topic #1 – Intro to Transactions) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #2 – Intro to CSS) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #3 – Intro to Java) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #4 – Intro to PHP) _____

- x

- x

- x

_____ (**Week 7** - Video Topic #5 – Creating a Database & Making a PHP Webpage) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #6 – Connecting to MySQL & Error Checks) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #7 – Passing Data to a Webpage & Joins) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #8 – Adding a Table Row with PHP) _____

- x
- x
- x

_____ (**Week 7** - Video Topic #9 – Adding a Table Column & Image Upload/Display) _____

- x
- x
- x

_____ (**Week 7 - Video Topic #10** – How to use PHP Global Variables) _____

- x
- x
- x