

# Carbon Accounting and Sustainable Designs in Product Lifecycle Management

**Prof. Deepu Philip**  
Department of Management Sciences

**Dr. Amandeep Singh Oberoi**  
Imagineering Laboratory

**Dr. Prabal Pratap Singh**  
Department of Management Sciences

Indian Institute of Technology, Kanpur

Week 11

Lecture 48

**Database Schema (Part-1)**

Hello everyone, welcome to the course on Carbon Accounting and Sustainable Designs in Product Lifecycle Management. I am Dr. Prabal Pratap Singh and we are co-teaching this course with Professor Deepu Philip and Dr. Amandeep Singh. In the course, till now we have covered various topics regarding carbon accounting databases and today we will try to understand what is a database schema.

- What are databases
- Major Carbon Accounting Database
- What is DBMS
- Evolution
- Advantages of DBMS
- Fundamental Terminologies
- ER diagram

## Database Schema

Carbon Accounting & Sustainable Designs in Product Lifecycle Management

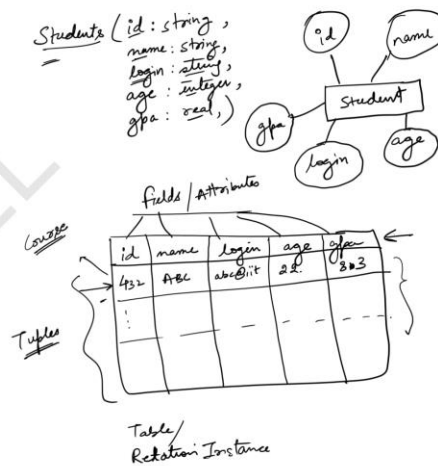
Dr. Prabal Pratap Singh  
Indian Institute of Technology Kanpur

So, until now we have covered what are databases. Then we talked about major carbon accounting databases in world and particularly in india as well we also talked about what is a database management system what is a DBMS.

Then we discussed about the evolution of these technologies and what are the advantages over simple files. We also looked upon the fundamental terminologies. That are useful and that are essential for managing and using these DBMS. And we also talked about what are ER diagrams, that is entity relationship diagrams. Now, today we will try to understand what are the database schema and how to translate these ER diagrams into a database schema, right.

## Relational Model

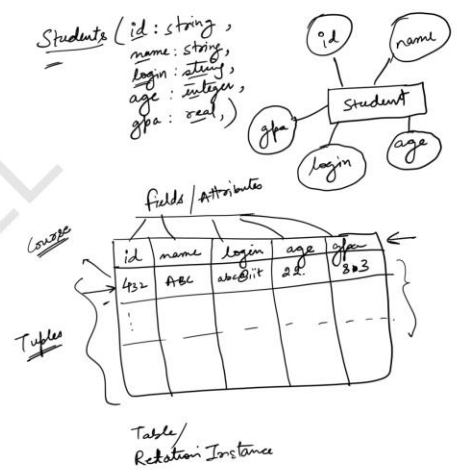
- Relational model is based on Relations  
↳ Connection
- Relational Schema → specify the Relation name  
Name of attributes  
Domain of each field.  
→ schema specifies Domain constraint
- Relation instance :→ set of tuples known as records.  
→ Each row is a tuple  
→ All rows will have same number of fields.
- Degree / Arity → Number of fields.
- Cardinality → Number of tuples/rows.



So, the outline for today is, that what is a relational model first of all. We have talked about it earlier as well, but we talked about it briefly and so we will see how data is represented in RDBMS. So, we already talked about the database design process, and we have completed various steps like requirement analysis, conceptual design, and now we also talked about the entity relationship diagram.

# Relational Model

- Relational model is based on Relations
  - ↳ Connection
- Relational Schema → specify the Relation name
  - Name of attributes
  - Domain of each field
- schema specifies Domain constraint
- Relation instance: → set of tuples known as records.
  - Each row is a tuple
  - All rows will have same number of fields.
- Degree/Arity → Number of fields.
- Cardinality → Number of tuples/rows.



So, the next part is how to transfer all this information into a relational model, since we are using an RDBMS in this course, right. Next is transformation. how to utilize this the data stored in a particular database and how to retrieve or modify or manage various user roles using the RDBMS.

So we will learn this declarative language today that is structured query language also known as SQL and people usually call it SQL. So this will be a very important information regarding any database management system that are using relational data type, right. Further we will look upon what are integrity constraints and how to store data that are legal as per the definition of DB. Initially, developers design how to store data in RDBMS and then when the user is storing that data, we should have particular constraints that will keep on checking whether the data the user is storing is correct or not, right. Finally, we will understand how to translate this ER diagram into a database schema, right.

So, let us start. First of all relational model. So, this is the fundamental concept of RDBMS. So, what happens is relation model is nothing but relational model is based on relations. Now, what is a relation in this DBMS?

It is nothing but connections between two different types of data or entities or relationships. So, further there is a relational schema. If you are using RDBMS and you

are trying to develop a schema, so that is a relational schema. It will specify the relation name or the name of attributes or the domain name. of each field or each column. Now this schema specifies the domain of the field or we can call it a domain constraint.

So, we briefly talked about this domain in earlier lecture as well. It is nothing but the data type that we are storing in a particular attribute. So, let us go back to a student database schema that we have showed earlier in a previous lecture. And it is a translated version from an ER diagram, from a very basic ER diagram. So, let us start from there and understand what are those schemas and how to translate it into actual database tables.

So we have shown a database schema for students and it was something like this students they will have a id and the data type of that id is string. Further they will have a name this will also be a string and they will have a login id it is also a string age will be a number. So it will be an integer and finally GPA that is the it would be a real number so all these things are the domain of these attributes now. This is the table name that we will design by translating it into a database table. And these are the attributes, right.

So, if you try to design an ER diagram for the student, then it will be something like this. Student is an entity. And it will have an ID. It will have a name. It will have a login.

It will have a age. And finally, a GPA. Right. Now, we have not defined which is the primary key or what is the unique identifying information yet. But we will get through that. So, this is a schema.

Now, from this, this is not how we are storing our data in a database table. So to store and translate this into a table we will have a table something like this in the database it will have some columns so it will have a id then name then login age and gpa, right. Now, these are the fields. So, these values these all are the fields. These are in the terms of ER diagram they are also known as attributes right.

Further, the next thing is we have some information about a particular student. So, let us say this is a student with an ID 432 and it has a name ABC with a login ABC at IIT with an age 22 and a GPA of 8.2. Now, this information, this is a particular row in a table and this is also known as a tuple in a database table. So, all this information is together known as a tuple in the database table and we can have multiple records in a table. So, all these things they are known as tuples.

So now this student, each student can have a course. So there could be a different course enrollment table where each student will be related to it. So these kinds of relations

between two entities are efficiently stored by these relational models. Now we have further few more terminologies like relational instance. So it is nothing but set of tuples which are known as records now you can call it a table database table or a relation instance both are the same thing.

So, instance can be a table and each row of the table is a tuple and all rows will have same number of fields. So, even if you design a particular table using pen and paper, you will first design its attributes, right. After that, you start filling the rows of the tables. So, that is why we are first trying to develop this relational schema. That is, what are the different kinds of attributes?

What are the data types of those attributes like this? So, we will first, the developers will first try to engage with the user who wants a new database management system. They will ask all the information in the requirement analysis. Then, during the conceptual design, they will try to capture all those realities that was captured in the requirement analysis and will try to further develop the entity relationship diagram. Those ER diagrams will be translated into these kinds of schemas that we have learned here.

And in these schemas, what happens is, we first try to develop these fields. Once the fields are finalized, then we start filling the data in the database table. So, in this database table, some developers also use other terms like degree or arity. So these are nothing but the number of fields of a particular relational instance. And we can also define cardinality as the number of tuples, rows.

So, if this table or this student table has 100 students, then cardinality of this table is 100. And if we are storing these 5 attributes only, then the degree or rarity of this table is 5, right.

# ER Diagram to DB Schema

ER diagram → initial, high-level Database design

Basic Table creation command

```
CREATE TABLE Students (
  ID INT PRIMARY KEY,
  NAME VARCHAR (50),
  Age INT
);
```

|              | ER Diagrams           | DB Schema                           |
|--------------|-----------------------|-------------------------------------|
| Entity       | Real world object     | DB table / relation                 |
| Relationship | Relation b/w entities | Foreign keys                        |
| Attribute    | Entity properties     | Column/Fields of Relation Instance. |
| Primary key  | Uniquely identify     | Identify tuples uniquely.           |

Now, let us try to initiate our translation from an ER diagram to database schema. So, we learned that ER diagram is nothing but ER diagram is an initial and high level database design. So let us try to differentiate between ER diagrams and database schema.

So we can create this differentiation like. So, we can have an entity in a E-R diagram and this is about E-R diagram say this is. So entity as per the er diagrams is a real world object, right. Like we have seen different examples for like school management system it can be a student or the class of a student further in a employee database management system. It could be an employee entity set or the department entity set, right.

But when we are trying to translate an entity to a database schema, it will become a database table. Or we can say database relation. The other thing in ER diagrams was relationship. So, relationships are the relations between entities and while developing this schema. We will use foreign keys to capture this relationship between different entities.

So we will understand what is a foreign key what are different kinds of integrity constraints and then we will understand how to connect two different kinds of entities in an actual database management system. So, the next thing is attributes. So, if we have an entity or entity set, they will have their own particular characteristics. So, these are known as attributes. Now, these attributes during the ER diagrams, we define them as entity properties.

And in the DB schema, we can call them as columns or fields of relation instance. One more thing we can differentiate these two different topics are using primary keys so this is in during the er diagrams we uniquely identify. The entity is in an entity set, right. And similar thing happens in a DB schema as well. We use these primary keys as an integrity constraint and we identify tuples uniquely.

So, we have seen this student's example. So, we can write a actual SQL means the structured query language that is how we are trying to store this database schema in a particular database table on our computer using this following basic table creation command. So to store that student example, we can write create table name of the table that was students and we will open our braces and we can write what are the attributes so we have an id. And the type of id is int that is it will be an integer and it will have an integrity constraint that is a primary key the next attribute could be name it will be a string. So we can define the data type in the database table using where care for a string.

So where care and the length of the string that could be let's say 50 and finally let's say the third attribute is age and it will also be an integer. Finally, we can close this definition. So, this simple command will create a database table without any tuples. It will only create a relation. In the database management system, now there are different things that we don't know that what are the different kinds of keywords that we can use and how to first create a database.

Then how to use what is the actual syntax for defining different kinds of arguments in this table so all this syntax we let us first introduce ourselves to the syntax of these commands. So that we can after creating this schema, we can directly create the actual tables in our database management system, right.

# Structured Query Language

- Standardized programming language → manage } Relational  
manipulate } Databases.
- CREATE →  
MODIFY → ALTER  
SELECT → to retrieve  
INSERT → to store data  
UPDATE →  
DELETE →  
DROP
- follows ANSI/ISO standards → ensure basic syntax is same  
across RDBMS.
- SQL is a declarative language  
highly user-friendly.

So, let us switch to Structured Query Language and then we will again switch back to translating ER diagrams. So, SQL is an standardized language. So, standardized language, programming language.

So, if it is a programming language, then we should use it for a particular purpose, right. So, we use it to manage and manipulate the relational databases. Now, as we learned about the database management systems, the evolution of these DBMS. We have learned that there are different kinds of products available in the market like some are open source, some are closed source. We have talked about different kinds of DBs like MariaDB, PostgreSQL, MySQL, Oracle DB.

So, each of these DBMS systems, what they do is they store your data and try to provide you different kinds of user management roles. Now, once your data is stored or you are in a position that you want to retrieve your data and use that data for your organization. Then there are different kinds of systems that can be possible so the complete uh society of these database management systems try to standardize this process of querying the data from the databases for different purposes. So, that is why we have this SQL language that is completely standardized using ISO or NC standardizations. So, any kind of SQL will provide you these following standard tasks that can be performed with any version of SQL.



So, this could be create or modify or select insert update or delete. So these are the most basic tasks that you want to perform. You will try to either create a database or a database table. So you will use this keyword. Once you have a database or database table and you want to modify it, you can use alter command.

So to create, you have the same command with the same name. If you have a database stored and you want to retrieve the data, then you will use the select command to retrieve the the data. If you want to store more data then you can use this insert command to store data if you want to update some tables then you can use this update command and you can also delete the data from the database table. Or you can completely drop the database or the database tables. So you can use in that situation you can use drop command there are different kinds of commands but these are the most basic commands that every SQL language will provide.

So all these SQL will follow NC or ISO certifications. And why do we need these standards? So that we can ensure basic syntax is same across RDBMS. Now, there are different kinds of programming languages and since SQL is also a programming language. So SQL is a declarative language.

Declarative languages are highly user friendly. And why they are user friendly? Because you just declare what you want and the backend implementation of the commands that you write is completely taken away from the user. So you don't need to understand how they are doing it or implementing it behind the database. But you just need to define whether you want to create a database or you want to modify it using their standard syntax.


And then the command will operate and final process will happen inside the database, right. So that is why the different kinds of RDBMS use this declarative language, SQL.

# Structured Query Language

→ Create Database  
`CREATE DATABASE dbname;`      Create Database TestDB ;

→ Drop Database  
`DROP DATABASE dbname;`

→ Create Table  
`CREATE TABLE tablename (  
    cd_name datatype,  
    cd_name2 datatype,  
    :  
);`



So let us now understand one by one each command and what is the syntax of that command. So the first thing is to create a database. So, we can use.

Usually, we write in uppercase using these RDBMS consoles and we will see how we write on the actual computer as well. But first, before writing, we should understand what is the syntax so that we do not make syntactical errors. So to create a particular database, you will first write create in uppercase and then the database means you want to create a database and finally provide the name of the database. So it could be anything like create database TestDB. So this command will create a database with the name TestDB and the RDBMS will have the particular storage allotted for this TestDB.

Further, you can write in upper case or lower case. Since it is not mandatory to write in uppercase but usually developers follow this syntax. So that because some kinds of database ask you to only write in uppercases otherwise the query will fail or the command will fail. The next is the drop database or you want to delete a database completely so you can use this command. So this will completely delete this DB name, the database with the name DB name from the RDBMS.

The next command could be how to create a table. We can also call it a relation. So to create a relation, we can write create table and provide a table name. Then start the braces. You can continue in the same line, but usually consoles are not that big.

So we will try to create a new line here and we will provide the column name. And the data type of this column name we can have multiple column names so we can write call name to and the data type associated with it. So similarly, we can have multiple column names and then once we have defined all the columns or the fields of this table, we can close this braces and we need to provide a semicolon after this. So this is how you can create a table inside your database.

## Structured Query Language

→ Drop Database Tables  
`DROP TABLE tablename;` } completely remove the DB table with fields and tuples

→ Truncate Table  
`TRUNCATE TABLE table_name;` } only remove the records from the instance

→ ALTER Table

- `ALTER TABLE tablename ADD column_name datatype;`
- `ALTER TABLE tablename DROP column_name datatype;`
- `ALTER TABLE tablename RENAME COLUMN Old_name to New_name;`
- `ALTER TABLE tablename ALTER COLUMN column_name datatype;`

→ CONSTRAINT CREATION → `CREATE TABLE table_name (  
 column1 datatype CONSTRAINT;  
 column2 datatype CONSTRAINT;  
 ...  
 );`

Now analogous to how we created a database and then have a command to drop that database, we also have a command to drop the database tables.

So we can drop database tables as well. So it will have drop tables. And provide the table name right now this command will completely remove the database table with fields and tuples. But usually sometimes we want to remove all the content or all the tuples or the records stored in that database. So we have the functionality of doing that as well.

So let's say you have 100 rows in a particular table. You can either remove each row one by one or you can use this command to truncate a table. You can write truncate table and the table name. So, this command will only remove the records from the instance. But the database table still remains in the database, right.

Other thing you may want to do is how to modify the data in your database table. You know how to create a table. You know how to drop a table. You know how to remove the content of the table. Now how to modify that table.

So for that we use alter commands and its different variations alter table. So let's say you want to add a new column after designing your complete schema you found that during the requirement analysis one column has been missed. So now you want to add a new column. So you can what you can do is you can write alter table the name of the table that is table name and provide this add keyword and write the new column name that you want to add column name with its data type, okay. Similarly you may want to drop a column that is pre-existing in the table so you can write alter table table name drop column name data type.

So whatever the column name you will provide here it will get removed from the database and to run this command successfully. This column name should be pre-existing in your table if it is not then the sql system will throw an error the next command is you may want to have a column but change its name. So you want to rename its name. You can use alter table table name rename column old name to new name. So if you just fill the existing name of the column and provide a new name, this command will change it to the new name for that particular column or the field.

Or you can also alter the data type. So you can alter table name, alter column name and the data type. So in this all these commands I have followed a particular design that whatever is written in upper cases that are the keywords and whatever is written in the lower cases that is something that the user needs to provide while writing these commands. You can write this table name in upper case as well but to create to show what are the different things you are using in a particular command as a declaration. I am using this syntax right now.

So the next thing is constraint. So we can also provide different kinds of constraints. That is how different relations are connected with each other. So constraint creation. Let's say you are defining a table and to define a table we usually write create table name of the table and define the column types the details of the columns like column 1 with this particular data type.

And we can specify the constraint on this column. So, there are different kinds of constraints. We will learn it in the upcoming slides. But this is the syntax of how you are trying to define a particular constraint on a particular column 1. Let us say on column 2,

you have its own associated data type and you have a different constraint. On this column, constraint 2. So this way you can define the constraints on a particular field of a particular table.

## Structured Query Language

→ INSERT → Fill data into Tables.

```

INSERT into Table-name (
    column1, column2 ... column n
)
VALUES (value1, value2, ... value n)
  
```

→ SELECT query

```

SELECT expression
FROM table-name
WHERE condition;
  
```

SELECT \* FROM STUDENTS S WHERE S.age > 15;

The next thing you can do is if you have a blank table you have created a database. And now you have also initialized a table in that database. You need to fill some data into it, right. So to do that we have this insert query.

And what it does is fill data into tables. So, the syntax for this query is provide this keyword insert into tables. The name of the table in which you want to fill the data and initialize your braces then define where the data is like column 1 column 2 column 1 right. And then the actual values, so value 1, value 2. So what this complete command will do is it will first find the table name if it is existing then it will check whether there are these numbers of columns that is from 1 to n.

And then it will fill value 1 in column 1, value 2 in column 2 and value n in column n respectively, right. So with for filling let us go back to our table which we initially discussed. So, if you want to fill this row in this database table, then what you need to do is you will try to write insert into with the name of the table. Whatever the name of this

table is, then provide these column names and then provide the corresponding values. As per the syntax, which we have just discussed.

So, after successfully executing that command, you will have a particular row filled in your database table, right. The next query could be select. So if you have pre-filled data in your database table and you want to retrieve it to see what are the different contents of that database table. So you can use this select query. It will try to show you whatever the contents present.

So the syntax is select expression from the name of the table where condition. Now there are three different keywords select from and where. So select will try to retrieve the data from will specify from which table. So here you are providing the name of the table. And where is a filter which will try to provide any type of condition that a user want to implement on this query.

So if you are writing, let's say, a basic query is select star or asterisk from students. This is the name of the table. Let's say in your database, we have a table with the name students where let's say condition is student could be taken as an alias with S. So, where S dot age is greater than 15. So, what this query will do is it will first using this expression star or asterisk is a special expression. It will select all the rows from where from this student table and this student table is now known as an alias for this command.

And this alias s we are providing a further condition on this query that we only need those rows from the table where the age of the student is greater than 15, right. So all those rows that have age greater than 15 from the table student will be reported on the console when the user runs this query.

Thank you.