Introduction to the Relational Model

- The need for a DDL (Data Definition Language) to define the structure of entities and their relationships, and for a DML to specify manipulation of database contents was also established.
- These concepts, however, were presented in quite abstract terms, with no commitment to any particular data structure for entities or links nor to any particular function to manipulate data objects.
- There is no single method for organising data in a database, and many methods have in fact been proposed and used as the basis of commercial DBMS's. A method must fully specify:
    1. The rules according to which data are structured, and
    2. The associated operations permitted
- The first is typically expressed and encapsulated in a DDL, while the second, in an associated DML. Any such method is termed a Logical Data Model (often simply referred to as a Data Model). In short,

  Data Model = DDL + DML

  and may be seen as a technique for the formal description of data structure, usage constraints and allowable operations. The facilities available typically vary from one Data Model to another.

Integrity Constraint over Relations:

- Before one can start to implement the database tables, one must define the integrity constraints. Integrity means something like 'be right' and consistent. The data in a database must be right and in good condition.
- There are the domain integrity, the entity integrity, the referential integrity and the foreign key integrity constraints.

Domain Integrity:

- Domain integrity means the definition of a valid set of values for an attribute. You define
  - data type,
  - length or size
  - is null value allowed

- is the value unique or not

for an attribute.

- You may also define the default value, the range (values in between) and/or specific values for the attribute. Some DBMS allow you to define the output format and/ or input mask for the attribute.
- These definitions ensure that a specific attribute will have a right and proper value in the database.

Entity Integrity Constraint:

- The entity integrity constraint states that primary keys can't be null. There must be a proper value in the primary key field.
- This is because the primary key value is used to identify individual rows in a table. If there were null values for primary keys, it would mean that we could not identify those rows.
- On the other hand, there can be null values other than primary key fields. Null value means that one doesn't know the value for that field. Null value is different from zero value or space.

Querying relational data:

- The relational model designs data structures as relations (tables) with attributes (columns) and the relationship between those relations.
- The information about one entity of the real world is stored within one row of a table. However, the term *one entity of the real world* must be used with care. It may be that our intellect identifies a machine like a single airplane in this vein.
- Depending on the information requirements it may be sufficient to put all of the pieces as discrete entities including the relationship to the whole thing. If, for example, information about every single seat within the airplane is needed, a second table *seat* and some way of joining seats to airplanes will be required.
- This way of breaking up information about real entities into a complex data model depends highly on the information requirements of the business concept.
- Additionally there are some formal requirements, which are independent of any application: the resulting data model should conform to a so-called normal form.

- Normally these data models consist of a great number of tables and relationships between them. Such models will not predetermine their use by applications; they are strictly descriptive and will not restrict access to the data in any way.
- Depending on the information requirements, it may be sufficient to put all of the information into one row of a table airplane. But, in many cases it is necessary to break up the entity into its pieces and model as discrete entities including the relationship to the whole thing.
- Operations within databases must have the ability to act not only on single rows, but also on sets of rows. Relational algebra offers this possibility. Therefore languages based on relational algebra, e.g.: SQL, offer a powerful syntax to manipulate a great bunch of data within one single command.
- As operations within relational algebra may be replaced by different but logically equivalent operations, a language based on relational algebra should not predetermine how its syntax is mapped to operations (the execution plan).
- The language should describe what should be done and not how to do it. Note: This choice of operations does not concern the use or neglect of indices.
- As described before the relational model tends to break up objects into sub-objects. In this and in other cases it is often necessary to collect associated information from a bunch of tables into one information unit. How is this possible without links between participating tables and rows? The answer is: All joining is done based on the values which are actually stored in the attributes.
- The RDBMS must make its own decisions about how to reach all concerned rows: whether to read all potentially affected rows and ignore those which are irrelevant (full table scan) or, to use some kind of index and read only those which match the criteria.

Logical Database Design:

- A logical data model is a fully attributed data model that is independent of DBMS, technology, data storage or organizational constraints.
- It typically describes data requirements from the business point of view. While common data modelling techniques use a relational model notation, there is no requirement that resulting data implementations must be created using relational technologies.
- Common characteristics of a logical data model
    - Typically describes data requirements for a single project or major subject area.

- May be integrated with other logical data models via a repository of shared entities.
- Typically contains 100-1000 entities, although these numbers are highly variable depending on the scope of the data model.
- Contains relationships between entities that address cardinality and nullability (optionality) of the relationships.
- Designed and developed to be independent of DBMS, data storage locations or technologies. In fact, it may address digital and non-digital concepts.
- Data attributes will typically have data types with precisions and lengths assigned.
- Data attributes will have nullability (optionality) assigned.

Introduction to Views:

- Logical database design is the process of deciding how to arrange the attributes of the entities in a given business environment into database structures, such as the tables of a relational database.
- The goal of logical database design is to create well-structured tables that properly reflect the company's business environment.
- The tables will be able to store data about the company's entities in a non-redundant manner and foreign keys will be placed in the tables so that all the relationships among the entities will be supported.
- Physical database design is the process of modifying the logical database design to improve performance.

Destroying / Altering Tables and Views:

- You already know that the table is the central RDBMS object. In a perfect world, you would never have to change anything in a table's structure after it was created. In real life though you need to modify table definitions quite often.
- Changing business rules are usually the main reason; incorrect initial design is also not uncommon, especially in test environments.
- The whole idea behind the ALTER TABLE statement is to make the table definition changes as fast and as painless as possible. The DROP TABLE statement is used to remove a table (both its data and definition) from the database.

Alter Table Statement:

- The SQL **ALTER TABLE** command is used to add, delete or modify columns in an existing table.
- You would also use ALTER TABLE command to add and drop various constraints on an existing table.

Syntax:

The basic syntax of **ALTER TABLE** to add a new column in an existing table is as follows:

```
ALTER TABLE table_name
ADD column_name datatype;
```

The basic syntax of ALTER TABLE to **DROP COLUMN** in an existing table is as follows:

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

The basic syntax of ALTER TABLE to change the **DATA TYPE** of a column in a table is as follows:

```
ALTER TABLE table_name
MODIFY COLUMN column_name datatype;
```

The basic syntax of ALTER TABLE to add a **NOT NULL** constraint to a column in a table is as follows:

```
ALTER TABLE table_name
MODIFY column_name datatype NOT NULL;
```

The basic syntax of ALTER TABLE to **ADD UNIQUE CONSTRAINT** to a table is as follows:

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint UNIQUE(column1, column2...);
```

The basic syntax of ALTER TABLE to **ADD CHECK CONSTRAINT** to a table is as follows:

```
ALTER TABLE table_name
ADD CONSTRAINT MyUniqueConstraint CHECK (CONDITION);
```

The basic syntax of ALTER TABLE to **ADD PRIMARY KEY** constraint to a table is as follows:

```
ALTER TABLE table_name
```

```
ADD CONSTRAINT MyPrimaryKey PRIMARY KEY (column1, column2...);
```

The basic syntax of ALTER TABLE to **DROP CONSTRAINT** from a table is as follows:

```
ALTER TABLE table_name
DROP CONSTRAINT MyUniqueConstraint;
```

If you're using MySQL, the code is as follows:

```
ALTER TABLE table_name
DROP INDEX MyUniqueConstraint;
```

The basic syntax of ALTER TABLE to **DROP PRIMARY KEY** constraint from a table is as follows:

```
ALTER TABLE table_name
DROP CONSTRAINT MyPrimaryKey;
```

If you're using MySQL, the code is as follows:

```
ALTER TABLE table_name
DROP PRIMARY KEY;
```

Relational Algebra:

- A query language is a language in which user requests information from the database. It can be categorized as either procedural or nonprocedural.
- In a procedural language, the user instructs the system to do a sequence of operations on database to compute the desired result.
- In nonprocedural language the user describes the desired information without giving a specific procedure for obtaining that information.
- The relational algebra is a procedural query language. It consists of a set of operations that take one or two relations as input and produces a new relation as output.
- **Fundamental Operations:**
    - SELECT
    - PROJECT
    - UNION
    - SET DIFFERENCE
    - CARTESIAN PRODUCT
    - RENAME

- Select and project operations are unary operations as they operate on a single relation. Union, Set Difference, Cartesian product and rename operations are binary operations as they operate on pairs of relations.
- **Other Operations:**
  - SET INTERSECTION
  - NATURAL JOIN
  - DIVISION
  - ASSIGNMENT
- The select operation is to identify a set of tuples, which is a part of a relation, and to extract only these tuples out. The select operation selects tuples that satisfy a given predicate or condition.
  - It is a unary operation defined on a single relation.
  - It is denoted as σ.

Relational Calculus:

- Comes in two flavours: Tuple relational calculus (TRC) and Domain relational calculus (DRC).
- Calculus has variables, constants, comparison ops, logical connectives and quantifiers.
  - TRC: Variables range over (i.e., get bound to) tuples.
  - DRC: Variables range over domain elements (= field values).
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called formulas.
- Answer tuple is essentially an assignment of constants to variables that make the formula evaluate to true.

Tuple Relational Calculus:

- A logical language with variables ranging over tuples:

  { T | Cond }

  Return all tuples T that satisfy the condition Cond.
- { T | R(T) }: returns all tuples T such that T is a tuple in relation R.
- { T.name | FACULTY (T) AND T.DeptId = 'CS' }. - returns the values of name field of all faculty tuples with the value 'CS' in their department id field.
  - The variable T is said to be free since it is not bound by a quantifier (for all, exists).

- The result of this statement is a relation (or a set of tuples) that correspond to all possible ways to satisfy this statement.
- Find all possible instances of T that make this statement true.

Domain Relational Calculus:

- We now present two relational "calculi" that we will compare to RA. First, what is the difference between "algebra" and a "calculus"?
- The usual story is that the algebra RA is operational – a RA expression describes a step-by-step procedure for computing a result relation.
- A calculus, on the other hand, is declarative – it is essentially a language of predicates which describe whether a tuple value (or a collection of scalar values) is contained in the result. Thus, in principle, a database management system has more freedom implementing a calculus based query language than an algebraic one.
- RA is a side-effect-free expression language, which in the programming languages community probably would not be considered operational. So the distinction is slightly dubious.
- Nevertheless, we present the first of our two calculi, the Domain Relational Calculus (DRC). Informally, this is a language of predicates whose variables range over scalar values in the data domain D. A query expression can construct a tuple and ask (with a predicate) whether it is present in a given relation in the database instance. A DRC expression has the following form:

$$E ::= \left\{ \langle A_1 : x_1, \ldots A_n : x_n \rangle \mid F \right\}$$
$$\text{where } FV(F) \subseteq \{x_1, \ldots, x_n\}$$

Expressive Power of Algebra and calculus:

- What queries can one express in SQL? Perhaps more importantly, one would like to know what queries cannot be expressed in SQL – after all, it is the inability to express certain properties that motivates language designers to add new features (at least one hopes that this is the case).
- This seems to be a rather basic question that database theoreticians should have produced an answer to by the beginning of the 3rd millennium.
- After all, we've been studying the expressive power of query languages for some 20 years now (and in fact more than that, if you count earlier papers by logicians on the

expressiveness of first-order logic), and SQL is the de-facto standard of the commercial database world – so there surely must be an answer somewhere in the literature.

- When one thinks of the limitations of SQL, its inability to express reachability queries comes to mind, as it is well documented in the literature (in fact, in many database books written for very different audiences, e.g. [1, 5, 7, 25]). Let us consider a simple example: suppose that R(Src, Dest) is a relation with flight information: Src stands for source and Dest for destination.
- To find pairs of cities (A, B) such that it is possible to y from A to B with one stop, one would use a self-join:

```
SELECT R1.Src, R2.Dest
FROM   R AS R1, R AS R2
WHERE R1.Dest = R2.Src
```

What if we want pairs of cities such that one makes two stops on the way? Then we do a more complicated self-join:

```
SELECT R1.Src, R3.Dest
FROM R AS R1, R AS R2, R AS R3
WHERE R1.Dest = R2.Src AND R2.Dest = R3.Src
```