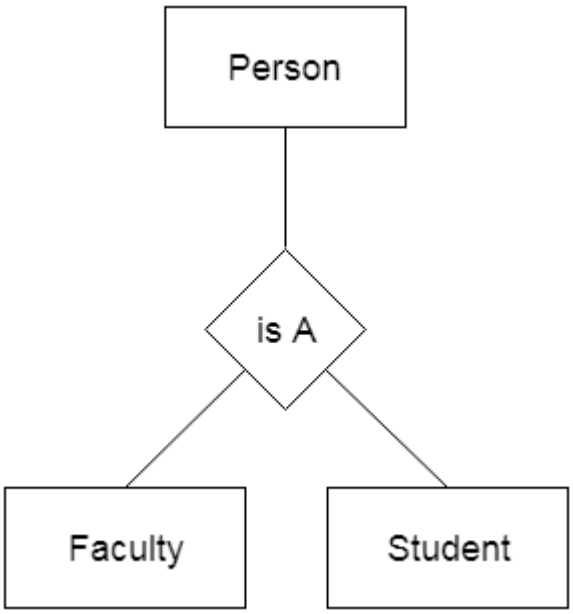
Generalization

- **Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.**
- **In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.**
- **Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.**
- **In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.**

For example, Faculty and Student entities can be generalized and create a higher level entity Person.

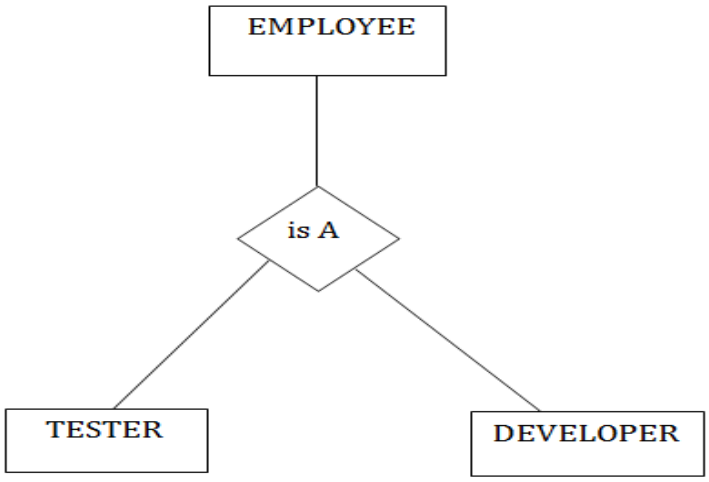


Specialization

- **Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.**

- **Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.**
- **Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.**

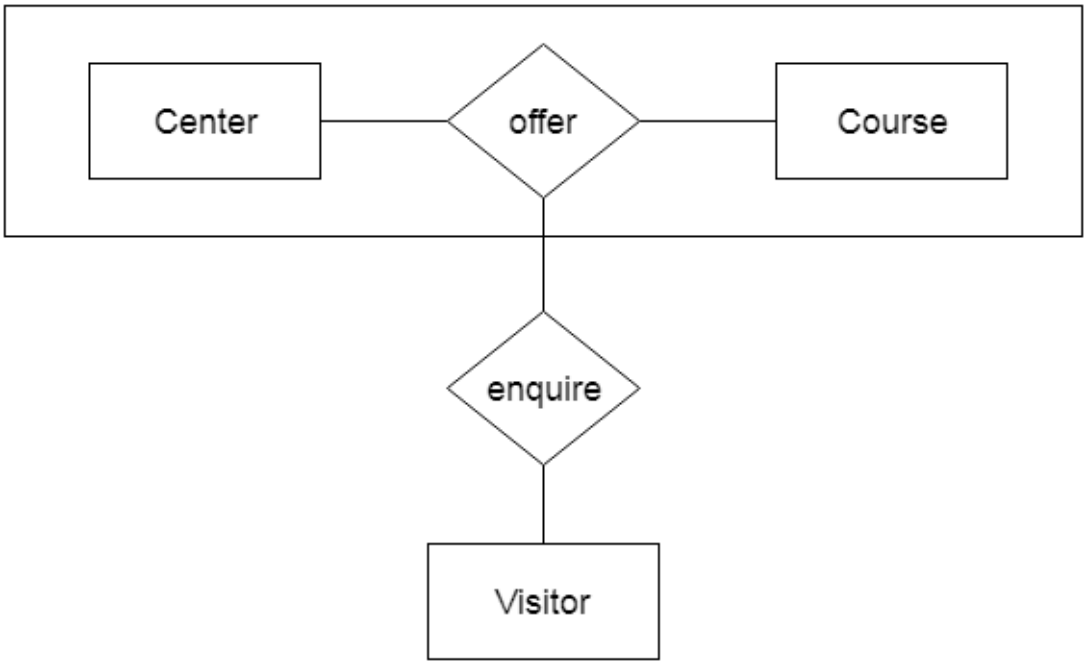
For example: In an Employee management system, EMPLOYEE entity can be specialized as a TESTER or DEVELOPER based on what role they play in the company.



Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

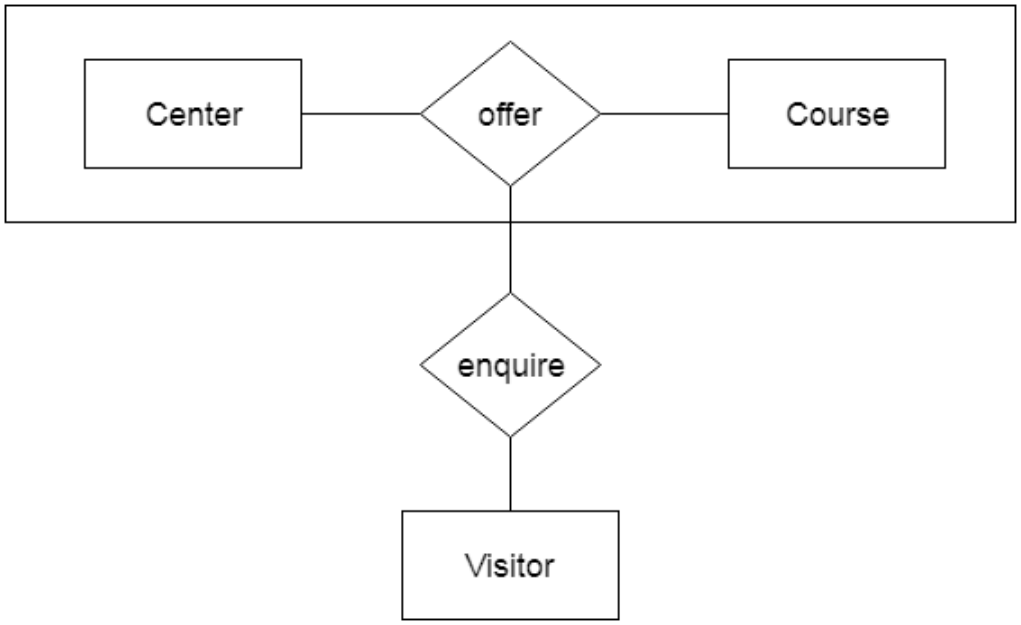
For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



Aggregation

In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

For example: Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



2.1 Relational Query Languages:

Relational algebra is used to break the user requests and instruct the DBMS to execute them. Relational Query language is used by the user to communicate with the database. They are generally on a higher level than any other programming language.

This is further divided into two types:

- Procedural Query Language
- Non-Procedural Language

Procedural Query Language

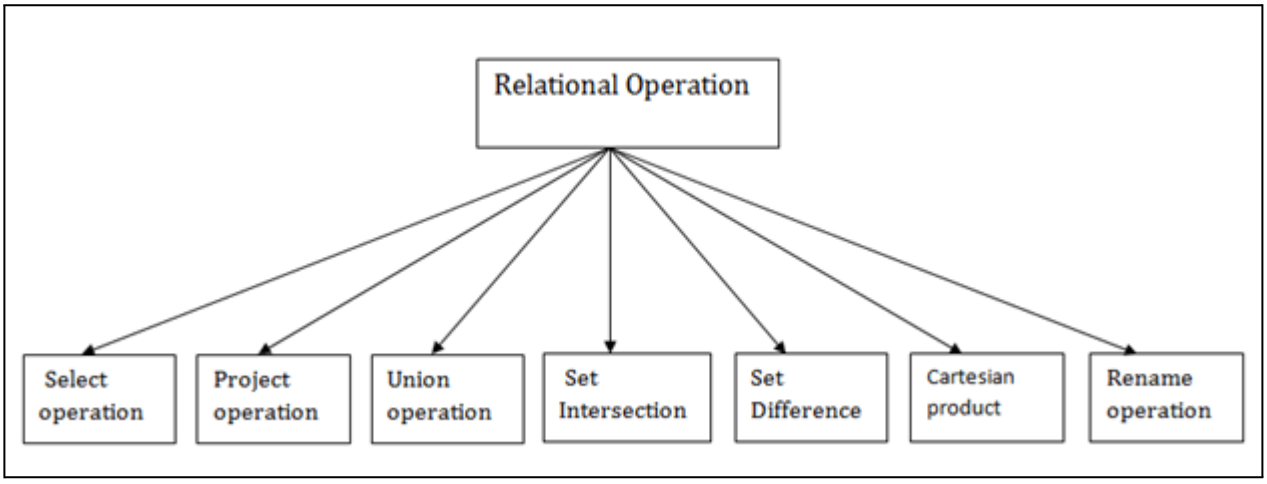
The user instructs the system to perform a set of operations on the database to determine the desired results.

Non-Procedural Language

The user outlines the desired information without giving a specific procedure for attaining the information.

2.2 Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

Where:

σ is used for selection prediction.

r is used for relation.

p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relations can be used as relational operators like =, ≠, ≥, <, >, ≤.

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rests of the attributes are eliminated from the table.
- It is denoted by ∏.

Where: **A1**, **A2**, **A3** is used as an attribute name of relation **r**.

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

8. Rename Operation

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

2.3 Joins

SQL JOINS

INNER JOIN


```
SELECT *
FROM A
INNER JOIN B ON A.key = B.key
```

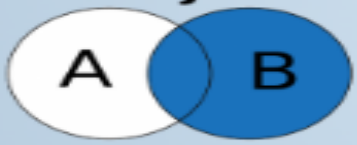
LEFT JOIN


```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
```

LEFT JOIN (sans l'intersection de B)


```
SELECT *
FROM A
LEFT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

RIGHT JOIN




```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
```

RIGHT JOIN (sans l'intersection de A)


```
SELECT *
FROM A
RIGHT JOIN B ON A.key = B.key
WHERE B.key IS NULL
```

FULL JOIN


```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
```

FULL JOIN (sans intersection)

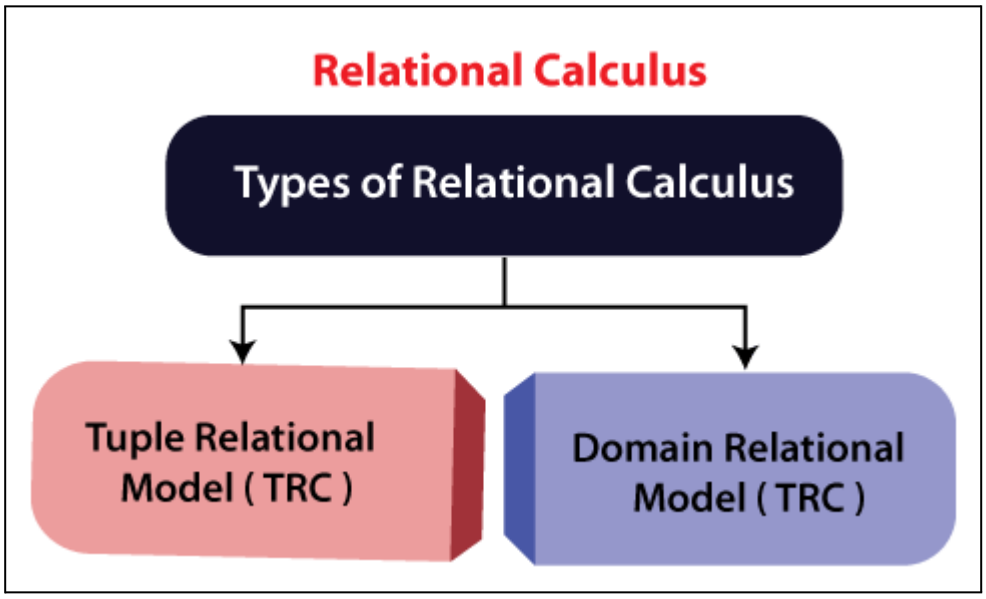

```
SELECT *
FROM A
FULL JOIN B ON A.key = B.key
WHERE A.key IS NULL
OR B.key IS NULL
```

2.4 Relational Calculus:

Before understanding Relational calculus in DBMS, we need to understand **Procedural Language** and **Declarative Language**.

1. **Procedural Language** - Those Languages which clearly define how to get the required results from the Database are called Procedural Language. **Relational algebra** is a Procedural Language.
2. **Declarative Language** - Those Language that only cares about What to get from the database without getting into how to get the results are called Declarative Language. **Relational Calculus** is a Declarative Language.

So Relational Calculus is a Declarative Language that uses Predicate Logic or First-Order Logic to determine the results from Database.
Types of Relational Calculus in DBMS:



Relational Calculus is of Two Types:

1. Tuple Relational Calculus (TRC)
2. Domain Relational Calculus (DRC)

1. Tuple Relational Calculus (TRC)

It is a non-procedural query language which is based on finding a number of tuple variables also known as range variables for which the predicate holds true. It describes the desired information without giving a specific procedure for obtaining that information. The tuple relational calculus is specified to select the tuples in a relation. In TRC, the filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.
Notation:

{T | P (T)} or {T | Condition (T)}

Where
T is the resulting tuples
P(T) is the condition used to fetch T.
For example:
{ T.name | Author(T) AND T.article = 'database' }

Output: This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.
TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

2. Domain Relational Calculus (DRC)

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes. Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ¬ (not). It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable. The QBE or Query by example is a query language related to domain relational calculus.
Notation: { a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where
a1,a2 are attributes
P stands for formula built by inner attributes

For example:
{< article, page, subject > | ∈ javatpoint ∧ subject = 'database'}
Output: This query will yield the article, page, and subject from the relational java point, where the subject is a database.

2.5 Normalization:

A large database defined as a single relation may result in data duplication. This repetition of data may result in:

- Making relations very large.
- It isn't easy to maintain and update data as it would involve searching many records in relation.
- Wastage and poor utilization of disk space and resources.
- The likelihood of errors and inconsistencies increases.

So to handle these problems, we should analyze and decompose the relations with redundant data into smaller, simpler, and well-structured relations that are satisfying desirable properties. Normalization is a process of decomposing the relations into relations with fewer attributes.

Decomposition of a Relation-

The process of breaking up or dividing a single relation into two or more sub relations is called as decomposition of a relation.

Types of Decomposition-

Decomposition of a relation can be completed in the following two ways-

S.NO	Lossy Compression	Lossless Compression
1	Lossy compression is the method which eliminate the data which is not noticeable.	While Lossless Compression does not eliminate the data which is not noticeable.
2	In Lossy compression, A file does not restore or rebuilt in its original form.	While in Lossless Compression, A file can be restored in its original form.
3	In Lossy compression, Data's quality is compromised.	But Lossless Compression does not compromise the data's quality.
4	Lossy compression reduces the size of data.	But Lossless Compression does not reduce the size of data.
5	Algorithms used in Lossy compression are: Transform coding, Discrete Cosine Transform, Discrete Wavelet Transform, fractal compression etc.	Algorithms used in **Lossless** compression are: Run Length Encoding, Lempel-Ziv-Welch, Huffman Coding, Arithmetic encoding etc.
6	Lossy compression is used in Images, audio, video.	Lossless Compression is used in Text, images, sound.
7	Lossy compression has more data-holding capacity.	Lossless Compression has less data-holding capacity than Lossy compression technique.
8	Lossy compression is also termed as irreversible compression.	Lossless Compression is also termed as reversible compression.

Key Differences Between Normalization and Denormalization :

- Normalization is the technique of dividing the data into multiple tables to reduce

data redundancy and inconsistency and to achieve data integrity. On the other hand, De-normalization is the technique of combining the data into a single table

to make data retrieval faster.

- Normalization is used in OLTP system, which emphasizes on making the insert,

delete and update anomalies faster. As against, Denormalization is used in OLAP system, which emphasizes on making the search and analysis faster.

- Data integrity is maintained in normalization process while in de-normalization

data integrity harder to retain.

• Redundant data is eliminated when normalization is performed whereas de-normalization increases the redundant data.

• Normalization increases the number of tables and joins. In contrast, de-normalization reduces the number of tables and join.

• Disk space is wasted in de-normalization because same data is stored in different
places. On the contrary, disk space is optimized in a normalized table.

2.6 functional dependencies:

In relational database management, functional dependency is a concept that specifies the relationship between two sets of attributes where one attribute determines the value of another attribute. It is denoted as X → Y, where the attribute set on the left side of the arrow, X is called Determinant, and Y is called the Dependent. Functional dependencies are used to mathematically express relations among database entities and are very important to understand advanced concepts in Relational Database System

Types of Functional Dependencies in DBMS

1. Trivial functional dependency
2. Non-Trivial functional dependency
3. Multivalued functional dependency
4. Transitive functional dependency

1. Trivial Functional Dependency

In Trivial Functional Dependency, a dependent is always a subset of the determinant. i.e. If X → Y and Y is the subset of X, then it is called trivial functional dependency

Example:

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18

Here, {roll_no, name} → name is a trivial functional dependency, since the dependent name is a subset of determinant set {roll_no, name}. Similarly, roll_no → roll_no is also an example of trivial functional dependency.

2. Non-trivial Functional Dependency

In Non-trivial functional dependency, the dependent is strictly not a subset of the determinant. i.e. If X → Y and Y is not a subset of X, then it is called Non-trivial functional dependency.

Example:

roll_no	name	age
42	abc	17

43	pqr	18
44	xyz	18

Here, roll_no → name is a non-trivial functional dependency, since the dependent name is not a subset of determinant roll_no. Similarly, {roll_no, name} → age is also a non-trivial functional dependency, since age is not a subset of {roll_no, name}

3. Multivalued Functional Dependency

In Multivalued functional dependency, entities of the dependent set are not dependent on each other. i.e. If a → {b, c} and there exists no functional dependency between b and c, then it is called a multivalued functional dependency. For example,

roll_no	name	age
42	abc	17
43	pqr	18
44	xyz	18
45	abc	19

Here, roll_no → {name, age} is a multivalued functional dependency, since the dependents name & age are not dependent on each other(i.e. name → age or age → name doesn't exist !)

4. Transitive Functional Dependency

In transitive functional dependency, dependent is indirectly dependent on determinant. i.e. If a → b & b → c, then according to axiom of transitivity, a → c. This is a transitive functional dependency.

For example,

enrol_no	name	dept	building_no
42	abc	CO	4
43	pqr	EC	2
44	xyz	IT	1
45	abc	EC	2

Here, enrol_no → dept and dept → building_no. Hence, according to the axiom of transitivity, enrol_no → building_no is a valid functional dependency. This is an indirect functional dependency, hence called Transitive functional dependency.

5. Fully Functional Dependency

In full functional dependency an attribute or a set of attributes uniquely determines another attribute or set of attributes. If a relation R has attributes X, Y, Z with the dependencies X->Y and X->Z which states that those dependencies are fully functional.

6. Partial Functional Dependency

In partial functional dependency a non key attribute depends on a part of the composite key, rather than the whole key. If a relation R has attributes X, Y, Z where X and Y are the composite key and Z is non key attribute. Then X->Z is a partial functional dependency in RBDMS.

Advantages of Functional Dependencies

Functional dependencies having numerous applications in the field of database management system. Here are some applications listed below:

1. Data Normalization

Data normalization is the process of organizing data in a database in order to minimize redundancy and increase data integrity. Functional dependencies play an important part in data normalization. With the help of functional dependencies we are able to identify the primary key, candidate key in a table which in turns helps in normalization.

2. Query Optimization

With the help of functional dependencies we are able to decide the connectivity between the tables and the necessary attributes need to be projected to retrieve the required data from the tables. This helps in query optimization and improves performance.

3. Consistency of Data

Functional dependencies ensures the consistency of the data by removing any redundancies or inconsistencies that may exist in the data. Functional dependency ensures that the changes made in one attribute do not affect inconsistency in another set of attributes thus it maintains the consistency of the data in database.

4. Data Quality Improvement

Functional dependencies ensure that the data in the database is accurate, complete and updated. This helps to improve the overall quality of the data, as well as it eliminates errors and inaccuracies that might occur during data analysis and decision making, thus functional dependency helps in improving the quality of data in the database.

Normal Forms:

Starting with normal form	Convert to normal form	Abbreviation	Use the rule
Un-normalized data	First	1NF	remove repeating groups of data within a single tuple into new tuples with only one data value for each attribute
First	Second	2NF	Extract non-key partial dependencies (items not fully functionally dependent on the key) into a separate relation
Second	Third	3NF	remove transitive dependencies into a separate relation
Third	Boyce-Codd	BCNF	remove overlapping keys by identifying a single primary key and holding other values in a separate relation
Third Or Boyce-Codd	Fourth	4NF	remove multi-valued dependencies by extracting independent data into a separate relation
Fourth	Fifth	5NF	remove join dependencies caused by interdependent data

****