Concept Of redundancy (Updation Anomalies)

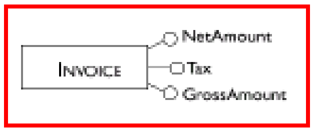
Hi! We are going to discuss one of the fascinating and important topics in a DBMS.

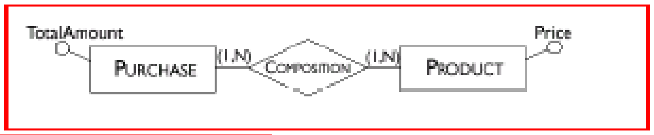
Analysis of Redundancies

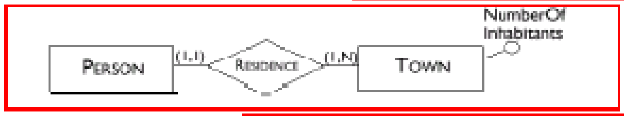
Before we go in to the detail of Normalization I would like to discuss with you the redundancies in the databases.

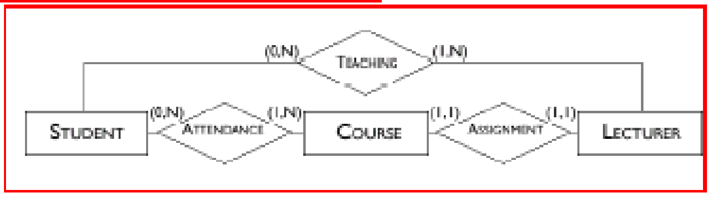
A redundancy in a conceptual schema corresponds to a piece of information that can be derived (that is, obtained through a series of retrieval operations) from other data in the database.

Examples of Redundancies









Deciding About Redundancies

The presence of a redundancy in a database may be decided upon the following factores

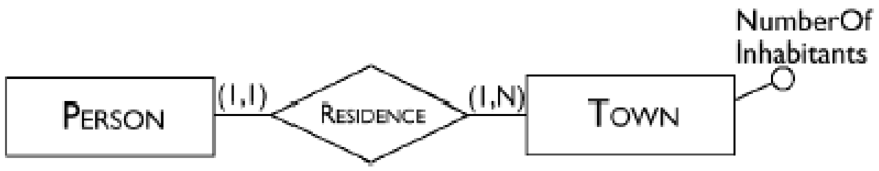
- **an advantage**: a reduction in the number of accesses necessary to
  Obtain the derived information;

- **a disadvantage**: because of larger storage requirements, (but, usually At negligible cost) **and the necessity to carry out additional operations in order to keep the derived data consistent**.

The decision to maintain or delete a redundancy is made by comparing the cost of operations that involve the redundant information and the storage needed, in the case of presence or absence of redundancy.

Cost Comparison: An Example

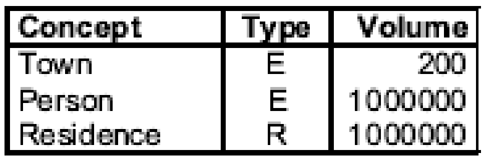
Now we will see the impact of redundancy with the help of an example.



In this schema the attribute `NumberOfInhabitants` *is redundant.*

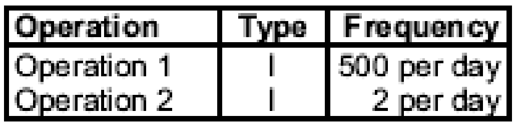
Load and Operations for the Example

Table of volumes



Concept	Type	Volume
Town	E	200
Person	E	1000000
Residence	R	1000000

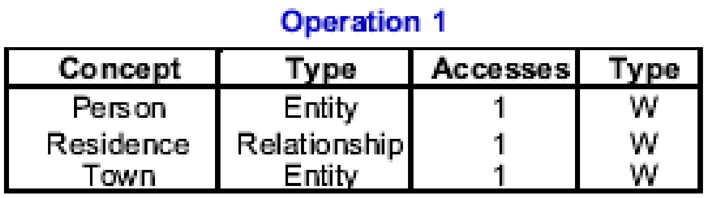
Table of operations



Operation	Type	Frequency
Operation 1	I	500 per day
Operation 2	I	2 per day

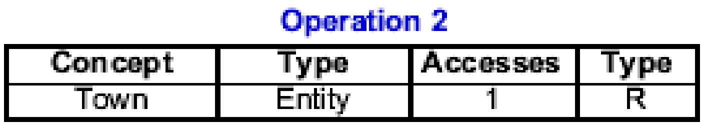
- ***Operation 1**: add a new person with the person's town of residence.*
- ***Operation 2**: print all the data of a town (including the number of inhabitants).*

Table of Accesses, with Redundancy



Operation 1

Concept	Type	Accesses	Type
Person	Entity	1	W
Residence	Relationship	1	W
Town	Entity	1	W



Operation 2

Concept	Type	Accesses	Type
Town	Entity	1	R

Issues related to Redundancies (Anomalies)

The time has come to reveal the actual facts why normalization is needed. We will look in to the matter in detail now.

The serious problem with using the relations is the problem of update anomalies. These can be classified in to

- *Insertion anomalies*
- *Deletion anomalies*
- *Modification anomalies*

Insertion Anomalies

An ***"insertion anomaly"*** is a failure to place information about a new database entry into all the places in the database where information about that new entry needs to be stored. In a properly normalized database, information about a new entry needs to be inserted into only one place in the database; in an inadequately normalized database, information about a new entry may need to be inserted into more than one place and, human fallibility being what it is, some of the needed additional insertions may be missed.

This can be differentiated in to two types based on the following example

Emp_Dept

EName	SSN	BDate	Address	DNumber	DName	DMGRSSN
Smith	123456789	1965-01-09	Kandivly	5	Research	333445555
Rajeev	333445555	1955-12-08	Vashi	5	Research	333445555
Greta	999887777	1968-07-19	Sion	4	Admin	987654321
Rajesh	987654321	1941-06-20	Dadar	4	Admin	987654321

First Instance: - To insert a new employee tuple in to Emp_Dept table, we must include either the attribute values for the department that the employee works for, or nulls (if the employee does not work for a department as yet). For example to insert a new tuple for an employee who works in department no 5, we must enter the attribute values of department number 5correctly so that they are *consistent,* with values for the department 5 in other tuples in emp_dept.

Second Instance: - It is difficult to insert a new department that has no employees as yet in the emp_dept relation. The only way to do this is to place null values in the attributes for the employee this causes a problem because SSN in the primary key of emp_dept table and each tuple is supposed to represent an employee entity- not a department entity.

Moreover, when the first employee is assigned to that department, we do not need this tuple with null values anymore.

<u>**Deletion Anomalies**</u>

A ***"deletion anomaly"*** is a failure to remove information about an existing database entry when it is time to remove that entry. In a properly normalized database, information about an old, to-be-gotten-rid-of entry needs to be deleted from only one place in the database; in an inadequately normalized database, information about that old entry may need to be deleted from more than one place, and, human fallibility being what it is, some of the needed additional deletions may be missed.

The problem of deletion anomaly is related to the second insertion anomaly situation which we have discussed earlier, if we delete from emp_dept an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.

Modification Anomalies

In Emp_Dept, if we change the value of one of the attribute of a particular department-say, the manager of department 5-we must update the tuples of all employees who work in that department; other wise, the database will become inconsistent. If we fail to update some tuples, the same department will be shown to have 2 different values for manager in different employee tuple which would be wrong.

All three kinds of anomalies are highly undesirable, since their occurrence constitutes corruption of the database. Properly normalized databases are much less susceptible to corruption than are unnormalized databases.

Update Anomalies --- Redundant information not only wastes storage but makes updates more difficult since, for example, changing the name of the instructor of CP302 would require that all tuples containing CP302 enrolment information be updated. If for some reason, all tuples are not updated, we might have a database that gives two names of instructor for subject CP302. This difficulty is called the *update anomaly*.

Insertional Anomalies -- Inability to represent certain information --- Let the primary key of the above relation be *(sno, cno)*. Any new tuple to be inserted in the relation must have a value for the primary key since existential integrity requires that a key may not be totally or partially NULL. However, if one wanted to insert the number and name of a new course in the database, it would not be possible until a student enrols in the course and we are able to insert values of *sno* and *cno*. Similarly information about a new student cannot be inserted in the database until the student enrols in a subject. These difficulties are called *insertion anomalies*.

Deletion Anomalies -- Loss of Useful Information --- In some instances, useful information may be lost when a tuple is deleted. For example, if we delete the tuple corresponding to student 85001 doing CP304, we will loose relevant information about course CP304 (viz. course name, instructor, office number) if the student 85001 was the only student enrolled in that course. Similarly deletion of course CP302 from the database may remove all information about the student named Jones. This is called *deletion anomalies*.

Review Questions

1. Explain insertion anomaly
2. Explain deletion and modification anomalies
3. what are the factors which decide the redundancy in a data base