Lecture 03

Benefits of Database Approach (in terms of Data):

1. Minimal Data Redundancy
2. Improved Consistency
3. Sharing of data
4. Improved data Integrity
5. Transaction support
6. Enforcement of Security
7. Ease of Application Development
8. Reduce Program Maintenance
9. Enforcement of standard
10. Balancing of conflicting requirement

Minimal Data Redundancy

Data redundancy is a condition created within a database or **data** storage technology in which the same piece of **data** is held in two separate places.

Data redundancy occurs in database systems which have a field that is repeated in two or more tables. When customer data is duplicated and attached with each product bought, then redundancy of data is a known source of inconsistency, since the entity "customer" might appear with different values for a given attribute.

Database ***normalization (will be covered in detail)*** prevents redundancy and makes the best possible usage of storage.

Improved Consistency

Consistency in database systems refers to the requirement that any given **database** transaction must change **affected data** only in allowed ways. Any data written to the **database** must be valid according to all defined rules, including constraints, cascades, triggers, and any combination thereof.

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7839	KING	PRESIDENT		17-NOV-81	5000		10
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10

DEPT

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

In above I can not update wrong ***department no*** in employee tables because of the rules enforced in these two tables. Let say if I update deptno = 11 in employee table Rules will not allow me to do to(see below).

```
SQL> update emp set deptno = 92 where empno=7924;
update emp set deptno = 92 where empno=7924
*
ERROR at line 1:
ORA-02291: integrity constraint (SCOTT.FK_DEPTNO) violated - parent key not found
```

Sharing of data: Each authorized user can access (respective part of data) from same database. New application can be developed against the same shared data.

Improved data Integrity:

Data integrity is the assurance of (via constraints) the accuracy and consistency of **data** over its entire life-cycle (after all DMLs has been performed on the data)

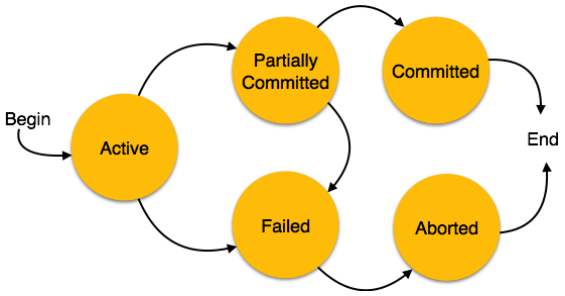
Data integrity is usually imposed during the database design phase through the use of standard procedures and rules. Data integrity can be maintained through the use of various error-checking methods and validation procedures.

An example of a data-integrity mechanism is the parent-and-child relationship of related records.

If a parent record owns one or more related child records all of the referential integrity processes are handled by the database itself ***which automatically ensures the accuracy and integrity of the data*** so that *no child record can exist without a parent(deptno was unable to update above because no parent was exist as a deptno in a parent table I.e. of department,* and that *no parent loses their child records(every foreign key record must have a primary defination)*. It also ensures that no parent record can be deleted **while the parent record owns any child records**(cant delete an entry from Dept table unless that department has been assigned to any of the employee). All of this is handled at the database level and ***does not require coding integrity checks into each applications.***

Transaction support:

Transaction supports maintain data consistency with the following concept:



Transaction best example is transferring money from *Bank Account A* to *Account B* would be consistent if let say following act happens:

A) **400 $ deducted from Bank Account A**
B) **Light fails/system fails / any failure in transaction**
C) **No updation Found in bank account B**

What if there would be no concept of transaction then *Account A* debtor of 400 $ for *account B* have to pay 800$ or more ☹

This problem has been resolved via diagram above (transaction support) that no updation will reflect the system unless transaction has been committed. In case of Failure-abortion all changes would be revert back (rollback)

Enforcement of Security

Database security refers to the collective measures used to protect and **secure** a **database** or **database** management software from illegitimate use and malicious threats and attacks. It is a broad term that includes a collection of processes, tools and methodologies that ensure **security** within a **database** environment.

Some of the ways database security is analyzed and implemented include:

- Restricting unauthorized access and use by implementing strong and multifactor access and data management controls
- Load/stress testing and capacity testing of a database to ensure it does not crash.
- Physical security of the database server and backup equipment from theft and natural disasters.
- Reviewing existing system for any known or unknown vulnerabilities and defining and implementing a road map/plan to mitigate them.

Ease of Application Development : Database technology have save application development time by providing very efficient techniques that just call one back end function (procedures) and have reduced inline data dealing techniques from code.

Reduce Program Maintenance : We may take examples of dynamic website whose development time and cost is expensive but as far as maintenance is concern they are cheaper and easier than the static ones.

Enforcement of standards: Standards like *formats standards, convention standards, data Quality standards, documentation standards and Data accessing and updating standards.* Make the application uniform and reliable in terms of data.

Balancing of conflicting requirement:

DBA can choose the best file structure and access methods to get the optimal performance.DBA can structure the database to entertain to different users at a time who have two different requirements to benefit the whole organization. Let's say financial department user is accessing user database with a **view** of *employees* and their *salaries*. Whereas IT department is accessing database with **view** of *employees* and their *sorted order of their email id*

Data Independence

Data independence is the type of ***data*** transparency that matters for a *centralized* DBMS. It refers to the immunity of **user applications** to *changes made in the definition and organization of data*.In short data must have freedom to change its Physical representation and access techniques. for example if I have introduced new devices in the system then their data schemes may be change.

Data dependency of a *user application* requires changes in a code of application for every change in *definition and organization of data*.

Let's cover few concepts relevant to data independence.

Logical Vs Physical Fields(stored Field)

Logical	***Physical***
Field as seen by the application user	Field that is actually stored in database
Let say Age is viewed by users as **26 years** as **string** field	Age is actually stored as **26 as number(3)**

Representation of Numeric data, integer or floating point:

A numeric <data type> is defined by a descriptor that contains following pieces of information:

1. The <data type>'s name: either **INTEGER**, **SMALLINT**, **NUMERIC**, **DECIMAL**, **FLOAT**, **REAL** or **DOUBLE PRECISION**.
2. The <data type>'s precision.(total digits in a number)
3. The <data type>'s scale (Digits after decimal).

`123.45` (p=5,s=2)
`12.34` (p=4,s=2)
`12345` (p=5,s=0)
`123.4` (p=4,s=1)
`0` (p=0,s=0)

4. the <data type>'s Base Whether precision and scale are expressed in decimal or binary terms.
5. Mode define Whether number is complex or real.

```
complex<double> com_one;              // value 0 + 0i
complex<double> com_two(3.14);        // value 3.14 + 0i
complex<double> com_three(1.5, 3.14)  // value 1.5 + 3.14i
```

Representation of Character Data:

Varchar (20). Varchar is used for string datatype and 20 defines the length of characters as 20.

Data Materialization:

a) **Direct Materialization:** stored field without any changes materialized into logical field (seen by user)

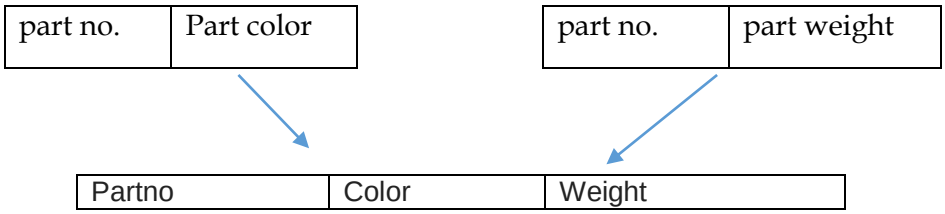
Example of Age as I have given in difference above.

b) **Indirect Materialization:** Some manipulation needed in stored field to create a logical field.

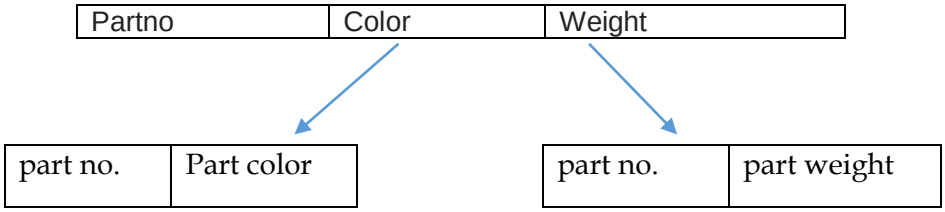
Let say for a product user is viewing **total price (not exist in DB)**which is a multiple of **Unit Price (exist in DB)**and **Quantity field(exist in DB).**

Structure of Stored records

Sometimes disperse data need to be consolidated for a view:



Sometimes Single piece of information needs a breakup for multidisciplinary information:



----------* End of Overview ***----------**

Note: from next lecture we will Start Database System Architecture