

CS403 Short Notes Final term

Lec 23 to45

BY vuonlinehelp.blogspot.com

YOUTUBE LINK:

WEBSITE LINK:

[HTTPS://VUONLINEHELP.BLOGSPOT.COM/](https://vuonlinehelp.blogspot.com/)

IMPORTANT NOTE

You should never copy paste the same answer which we are providing. Please do make some changes in the solution file. If you appreciate our work, please share our website vuonlinehelp.blogspot.com with your friends.

آپ کو کبھی بھی وہی جواب کاپی پیسٹ نہیں کرنا چاہیے جو ہم فراہم کر رہے ہیں۔ براہ کرم حل فائل میں کچھ تبدیلیاں کریں۔ اگر آپ ہمارے کام کی تعریف کرتے ہیں تو براہ کرم ہماری ویب سائٹ vuonlinehelp.blogspot.com کو اپنے دوستوں کے ساتھ شیئر کریں

Lecture No. 23

Physical Record and Denormalization

Denormalization is a technique to move from higher to lower normal forms of database modeling in order to speed up database access. Denormalization process is applied for deriving a physical data model from a logical form. In logical data base design we group things logically related through same primary key. There is a valid reason for denormalization that is to enhance the performance. Denormalization must balance the need for good system response time with the need to maintain data, while avoiding the various anomalies or problems associated with denormalized table structures.

DE normalization Situation 1

Merge two ET into one with one to one relationship. Even if one of the ETs is optional, so joining can lead to wastage of storage, however if two accessed together very frequently then merging might be a wise decision.

DE normalization Situation 2

Many too many binary relationships mapped to three relations. Queries needing data from two participating ETs need joining of three relations that is expensive. The relation created against relationship is merged with one of the relations created against participating ETs Reduces one join.

- EMP(empId, eName, pjId, sal)
- PROJ(pjId, pjName)
- WORK(empId, pjId, dtHired, sal)
- EMP(empId, eName, pjId, sal)
- PROJ(empId, pjId, pjName, dtHired, sal)

DE normalization Situation 3

One too many situations when the ET on one side does not participate in any other relationship, then many side ET is appended with reference data rather than the FK.

- STUDENT and HOBBY relations

Partitioning

DE normalization leads to merging different relations, whereas partitioning splits same relation into two. Two types are possible horizontal and vertical.

Horizontal Partitioning

Table is split on the basis of rows. The idea is that it is more efficient to process a table with small number of rows rather than a large table. Also helps in the maintenance of tables, like security, authorization, backup Even can be placed on different disks to reduce disk contention.

Some of the horizontal partitioning types:

- Range partitioning
- Hash partitioning
- List partitioning

Range Partitioning:

In this type of partitioning range is imposed on any particular attribute. So in this way different partitions are made on the basis of those ranges with the help of select statement.

Hash Partitioning:

It is a type of horizontal partitioning. In this type particular algorithm is applied and DBMS knows that algorithm. So hash partitioning reduces the chances of unbalanced partitions to a large extent.

List Partitioning:

In this type of partitioning the values are specified for every partition. So there is a specified list for all the partitions. So there is no range involved in this rather there is a list of values.

Lecture No. 24

Vertical Partitioning

Same table is split into different physical records; depending on the nature of accesses. PK is repeated in all vertical partitions of a table to get the original table.

- STD (stdId, sName, sAdr, sPhone, cgpa, prName, school, mtMrks, mtSubs, clgName, intMarks, intSubs, dClg, bMarks, bSubs)
- STD (stdId, sName, sAdr, sPhone, cgpa, prName)
- STDACD (sId, school, mtMrks, mtSubs, clgName, intMarks, intSubs, dClg, bMarks, bSubs)

Replication

SQL Server Replication is a process of copying and distributing data and Database objects from one Database to another Database and synchronizing all the data between the Databases to maintain integrity and consistency of the data. The process can be set to synchronize data continuously in real-time or scheduled to run at the desired intervals. Mainly the SQL Server Replication is used to copy and reproduce data at the desired targets. Some of the SQL Server Replication techniques are listed below:

- One-Way
- One-to-One
- Many-to-One
- Bi-Directional

Clustering Files

Clustering means to place records from different tables to place in adjacent physical locations, called clusters. Increases efficiency since related records are placed close to each other. Clustering is also suitable to relatively static situations. Define cluster, define the key of the cluster, include the tables into the cluster while creating associating the key with it.

Other Related Issues

To be discussed later during storage concepts discussion

- File structures
- Indexes
- RAID

Summary of Physical DB Design

Database design is the process of transforming a logical data model into an actual physical database. Data definition language (DDL) skills to translate the physical design into actual database objects.

Structured Query Language (SQL)

SQL is an ANSI standard computer language for accessing and manipulating databases. SQL is standardized, and the current version is referred to as SQL-92. Any SQL-compliant database should conform to the standards of SQL at the time.

SQL use

SQL is used to communicate with a database. According to ANSI (American National Standards Institute), it is the standard language for relational database management systems. SQL statements are used to perform tasks such as update data on a database, or retrieve data from a database. Some common relational database management systems that use SQL are: Oracle, Sybase, Microsoft SQL Server, Access, Ingres, etc.

Benefits of Standard SQL

- Reduced training cost
- Application portability
- Application longevity
- Reduced dependence on a single vendor
- Cross-system communication

SQL is used for any type of interaction with the database through DBMS. It can be used for creating tables, insertion in the table and deletion as well and other operations also.

SQL

SQL is used for any type of interaction with the database through DBMS. Right from creating database, tables, views, users Insertion, updation, and deletion of data in the database.

SQL and DBMSs

Implementation of SQL always includes the flavor of the particular DBMS, like in names of data types. Proprietary SQLs exist, like T-SQL and PL-SQL.

MS SQL Server

The DBMS for our course is Microsoft's SQL Server 2000 desktop edition. Two main tools Query Analyzer and Enterprise Manager; both can be used. For SQL practice we will use QA.

Terminology of SS

Instance of SQL Server Instance contains different objects like databases, security, DTS etc. Then databases object of SS contains multiple databases, each database contains multiple objects like tables, views, users, roles etc.

Working in SS

After installing SS, we will create database using create command after this we will create tables and other objects within this database.

Lecture - 25

Rules of the Format

SQL, at its simplest, is a basic language that allows you to "talk" to a database and extract useful information.

Data Definition Language (DDL) contains the commands used to create and destroy databases and database objects. After the database structure is defined with DDL, database administrators and users can utilize the Data Manipulation Language to insert, retrieve and modify the data contained within it.

Following are the rules for writing the commands in SQL:-

- Reserved words in capital
- User-defined identifiers in lowercase
- Valid identifiers
- Optionals in []
- Curly braces means required items
- | means choices
- [,.....n] means n items separated by comma

Example

- **SELECT [ALL|DISTINCT] { * | select_list } FROM { table | view [,...n] }**
- **Select * from std**

Data Types in SS

A data type is an attribute that specifies the type of data that the object can hold: integer data, character data, monetary data, date and time data, binary strings, and so on. SQL Server supplies a set of system data types that define all the types of data that can be used with SQL Server.

Integer

Bit (0 or 1)

Tinyint (0 through 255)

Smallint -32768 to 32767

Int -2^{31} to $2^{31}-1$ (10 digits)

Bigint -2^{63} to $2^{63}-1$ (19 digits)

Decimal or numeric

- Decimal
Fixed precision and scale numeric data from $-10^{38}+1$ through $10^{38}-1$.
- Numeric
Functionally equivalent to decimal.

Text: handles textual data

- char: by default 30 characters, max 8000
- varchar: variable length text, max 8000
- Text: variable length automatically
- nchar, nvarchar, ntext

Money: handles monetary data

- smallmoney: 6 digits, 4 decimal
- money: 15 digits, 4 decimal

Floating point

- Float
- Real

Date

- Smalldatetime
- datetime

SQL Commands Types

SQL commands are categorized in three broad classes.

- Data definition language (DDL)
- Data manipulation language (DML)
- Data control language (DCL)

Data Manipulation Language

- DML commands are used to create, alter or drop tables or other database objects
- DBMSs provide graphic interface for creating tables

Lecture No. 26

DDL

It deals with the structure of database. The DDL (Data Definition Language) allows specification of not only a set of relations, but also the following information for each relation:

The schema for each relation

- The domain of values associated with each attribute.
- Integrity constraints.
- The set of indices for each relation.
- Security and authorization information.
- Physical storage structure on disk.

Create

The first data management step in any database project is to create the database. The name of the table must be unique to the database. The name must begin with a letter and can be followed by any combination of alphanumeric characters. The name is allowed to contain the underscore character (_).

Create table command is used to:

- Create a table
- Define attributes of the table with data types

Define different constraints on attributes, like primary and foreign keys, check constraint, not null, default value etc.

Format of Create Table

CREATE TABLE

```
[ database_name.[ owner ] . | owner. ] table_name
( { < column_definition >
  | column_name AS computed_column_expression
  | < table_constraint >
}
```

```
)  
| [ { PRIMARY KEY | UNIQUE } [ ,...n ] ]  
)
```

Create Table Command

```
CREATE TABLE Program (  
    prName char(4),  
    totSem tinyint,  
    prCredits smallint)
```

Lecture No. 27

Alter Table Statement

Purpose is to make changes in the definition of a table already created through Create statement. Can add, drop attributes or constraints activate or deactivate constraints; the format is.

Syntax

```
ALTER TABLE table  
{ [ ALTER COLUMN column_name  
    { new_data_type [ ( precision [ , scale ] ) ]  
    [ NULL | NOT NULL ]  
  ]  
  | ADD  
    { [ < column_definition > ]  
    | column_name AS computed_column_expression  
    } [ ,...n ]  
  | DROP  
    { [ CONSTRAINT ] constraint_name  
    | COLUMN column } [ ,...n ]  
  | { CHECK | NOCHECK } CONSTRAINT  
    { ALL | constraint_name [ ,...n ] }  
}
```

Alter Table Command

```
ALTER TABLE Student  
    add constraint fk_st_pr  
    foreign key (prName) references  
        Program (prName)  
ALTER TABLE program  
    add constraint ck_totSem  
    check (totSem < 9)
```

Removing or Changing Attribute

```
ALTER TABLE student
  ALTER COLUMN stFName char(20)
Alter table student
  drop column curSem
Alter table student
  drop constraint ck_st_pr
```

Data Manipulation Language

The non-procedural nature of SQL is one of the principle characteristics of all 4GLs - Fourth Generation Languages - and contrasts with 3GLs. SQL is a non-procedural language that is, it allows the user to concentrate on specifying what data is required rather than concentrating on the how to get it. There are two types of DML.

- First is procedural in which: the user specifies what data is needed and how to get it.
- Second is nonprocedural in which the user only specifies what data is needed.

The DML component of SQL comprises of following basic statements:

- Insert To add new rows to tables.
- Select To retrieve rows from tables.
- Update to modify the rows of tables.

Insert Statement

To insert records into tables

```
INSERT [ INTO] table
{ [ ( column_list ) ]
  { VALUES
    ( { DEFAULT | NULL | expression } [ ,...n ] )
  }
}
| DEFAULT VALUES
```

The basic format of the INSERT...VALUES statement adds a record to a table using the columns you give it and the corresponding values you instruct it to add. You must follow three rules when inserting data into a table with the INSERT...VALUES statement: The values used must be the same data type as the fields they are being added to. The data's size must be within the column's size. For instance, you cannot add an 80- character string to a 40-character column. The data's location in the VALUES list must correspond to the location in the column list of the column it is being added to. (That is, the first value must be entered into the first column, the second value into the second column, and so on.)

Lecture No. 28

Important FOR QUIZ

Insert

The INSERT statement allows you to insert a single record or multiple records into a table. It has two formats:

- INSERT INTO table-1 [(column-list)] VALUES (value-list)
- INSERT INTO table-1 [(column-list)] (query-specification)

The first form inserts a single row into table-1 and explicitly specifies the column values for the row. The second form uses the result of query-specification to insert one or more rows into table-1. Both forms have an optional column-list specification. Only the columns listed will be assigned values. Unlisted columns are set to null, so unlisted columns must allow nulls. The default column list contains all columns in table-1 in the order they were declared in CREATE TABLE.

The VALUES Clause in the INSERT Statement provides a set of values to place in the columns of a new row. It has the following general format: VALUES (value-1 [, value-2]...)

Attribute Alias

This function attribute enables you to specify multiple aliases for functions. Where a function is defined in the current translation unit, the alias call is replaced by a call to the function, and the alias is emitted alongside the original name.

__attribute__((alias)) function attribute

The **DISTINCT** keyword is used to return only distinct (different) values. The SELECT statement returns information from table columns. But what if we only want to select distinct elements With SQL, all we need to do is to add a DISTINCT keyword to the **SELECT** statement.

Lecture No. 29

Skip

Lecture No. 30

LONG/important

ORDER BY Clause

The **ORDER BY** clause allows you to sort the records in your result set. The ORDER BY clause can only be used in SELECT statements. The **ORDER BY** clause sorts the result set based on the columns specified.

Functions in SQL

LONG

A **function** is a special type of command. Infact, functions are one-word command that return a single value. There are normally two types of functions. First is Built in, which are provided by any specific tool or language. Second is user defined, which are defined by the user.

Categories of Functions

These categories of functions are specific to SQL Server. Depending on the arguments and the return value, categorized

- Mathematical (ABS, ROUND, SIN, SQRT)
- String (LOWER, UPPER, SUBSTRING, LEN)
- Date (DATEDIFF, DATEPART, GETDATE())
- System (USER, DATALENGTH, HOST_NAME)
- Conversion (CAST, CONVERT)

Using Functions

```
SELECT upper(stName), lower(stFName), stAdres, len(convert(char, stAdres)),
```

```
FROM student
```

	(No column name)	(No column name)	stAdres	Adress Length
1	AMJAD	hussain	8-SD Lahore	11
2	AMJAD	rehan	I8 Ibd	6
3	TAHIRA EJAZ	nek muhammad	AJ Road	7
4	ARIF ZIA	zia khan	GM Rawalpindi	13
5	SUHAI DAR	nek muhammad	I-8 Islamabad	13
6	M. ALI	m. saad	JT Lahore	9
7	SADIA ZIA	NULL	NULL	NULL
8	SHOAIB ALI	rahmat ali	G-6 Islamabad	13

Aggregate Functions

Operate on a set of rows and return a single value, like, AVG, SUM, MAX, MIN, and STDEV. Attribute list cannot contain other attributes if an aggregate function is being used.

Aggregate Function Example

SELECT avg(cgpa) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student.

Output is.....

	Average CGPA	Maximum CGPA
1	2.9166666666666665	3.2

Group by Clause

The GROUP BY clause can be used in a SELECT statement to collect data across multiple records and group the results by one or more columns.

- SELECT stName, avg(cgpa) as 'Average CGPA', max(cgpa) as 'Maximum CGPA' from student
- SELECT prName, max(cgpa) as 'Max CGPA', min(cgpa) as 'Min CGPA'

FROM student GROUP BY prName

	prName	Max CGPA	Min CGPA
1	BCS	3.2	3.2
2	BIT	2.8	2.8
3	MBA	2.8	2.8
4	MCS	3.2	2.3

HAVING Clause

The HAVING clause is used in combination with the GROUP BY clause. We can restrict groups by using having clause groups satisfying having condition will be selected.

The syntax for the HAVING clause is:

```
SELECT select_list  
[ INTO new_table ]  
FROM table_source  
[ WHERE search_condition ]
```

[GROUP BY *group_by_expression*]
[HAVING *search_condition*]
[ORDER BY *order_expression* [ASC | DESC]]

Accessing Multiple Tables

LONG

Until now we have been accessing data through one table only. But there can be occasions where we have to access the data from different tables. So depending upon different requirements data can be accessed from different tables.

Following are the methods of accessing data from different tables:

- Cartesian Product
- Inner join
- Outer Join
- Full outer join
- Semi Join
- Natural Join

Cartesian product

A **Cartesian join** gives a Cartesian product. A Cartesian join is when you join every row of one table to every row of another table. You can also get one by joining every row of a table to every row of itself. No specific command is used just **Select** is used to join two tables.

Certain columns can be selected, same column name needs to be qualified. Similarly can be applied to more than one tables, and even can be applied on the same table

SELECT * from Student, class, program

Lecture No. 31

Accessing Multiple Tables Following are the methods of accessing data from different tables:

Inner Join

Only those rows from both tables are merged that have same value for the common attribute; equijoin implemented by different methods. Common attributes need not to have same name, but must have same domain. Applied generally between tables having referential integrity between them.

- `SELECT * FROM course INNER JOIN program`
- `ON`
- `course.prName = program.prName`
- `Select * FROM`
- `Course c inner join program p ON c.prName = p.prName`

Outer Join

Inner join plus the missing rows from one or more tables Left, Right and Full Outer Joins.

- Right Outer Join: Inner join plus rows from the non-matching rows from right table Left outer join performs the same thing but missing rows of the left side table.
- A Left Outer Join B = B Right Outer Join A Missing values are replaced with NULLs
- Full Outer Join: Inner join plus the non-matching rows from both tables

Semi Join

First inner join and then projected on the attributes of one table.

Advantage: tells the rows involved in join

No operator available as such, but can be implemented by select_list. `SELECT distinct p.prName, totsem, prCredits FROM program p inner JOIN course c.`

`ON p.prName = c.prName`

Self-Join

In self-join a table is joined with itself. This operation is used when a table contains the reference of itself through PK, that is, the PK and the FK are both contained in the same table supported by the referential integrity constraint.

Sub queries

Subquery is also called nested query and is one of the most powerful features of SQL. A nested query is a query that has another query embedded within it; the embedded query is called a subquery. subquery typically appears within the WHERE clause of a query. Subqueries can sometimes appear in the FROM clause or the HAVING clause. Here we have discussed only subqueries that appear in the WHERE clause.

- `SELECT * from student where cgpa >`
- `(select max(cgpa) from student where prName = 'BCS')`

That is it that we have to cover about DML in this course, and the golden rule.

A relational database stores all its data inside tables, and nothing more. All operations on data are done on the tables themselves or produce other tables as the result.

Application Programs

Programs written to perform different requirement posed by the users/organization are the application programs. Application programs can be developed in parallel or after the construction of database design.

General Activities in Application Programs

Following are the general activities, which are performed during the development of application programs:

- Data input programmes
- Editing
- Display
- Processing related to activities
- Reports

User Interface

LONG

The design of the user interface is therefore critical to the success or failure of a project. Get it right, and your users will forgive the occasional infelicity in implementation. Get it wrong, and it won't really matter how efficient your code is.

An effective user interface minimizes the time users require to learn and implement the system. Once the system is implemented, productivity gains are higher. Effective interfaces that closely match the users' expectations and work processes also minimize the need for external documentation.

Effective interface

An effective interface doesn't impose its requirements on users. It never forces users to play by its rules; it plays by the users' rules. An effective interface doesn't force users to learn a bunch of uninteresting stuff just to use it. And finally, it doesn't behave in unexpected ways. Following are the two types of user interfaces:

- Text based
- Graphical User Interface (GUI) most commonly called as Forms

Text Based User Interface

In text based user interface certain keyboard numbers are designated for any action but it is used very rarely nowadays.

- Adding a Record ----- 1

➤ Deleting a Record	----- 2
➤ Enrollment	----- 3
➤ Result Calculation	----- 4
➤ Exit	----- 5

Forms

Forms are now days used extensively in the application programs. Following are the different types of forms

➤ Browser based

These are web-based forms.

- They are developed in HTML, scripting language or Front Page.

➤ Non-Browser/Simple

- Visual Basic, Developer, MS Access

➤ Form tools support simple built-in functionality

- Graphical User Interface

User Friendly Interface

LONG

- Beginners (What and How)
- Intermediate (What)
- Experts (Both)

Beginners

Everyone is a beginner at some point. Beginners need to know what your system does before they start to learn how to use it. The best way to present this information is outside the main system itself. For simple systems, an introductory dialog box that describes the system can be sufficient. Online help isn't a good option for beginners. They might not know it exists or, if they do, how to use it.

Intermediate

Intermediate users know what the system does, but they often forget the details of how. Fortunately, the Microsoft Windows interface provides a lot of tools for helping these users. A well-designed menu system is one of the best tools for reminding intermediate users of the system capabilities. A quick scan of the available menu items will immediately remind them of the functions available and at the same time allow them to initiate the appropriate task.

Experts

Expert users know what to do and how to do it. They're primarily interested in doing things quickly. The more shortcuts you can build into your system, the happier you will make this group of users. In my experience, expert users tend to be keyboard oriented.

Tips for User Friendly Interface

LONG

Following are some of the important tips, which should be adhered for interfaces:

- It should be user friendly and user must not search for required buttons or text boxes.
- Interface should be designed in such a manner that user is in charge of the form.
- It should be designed in manner that user memory is always tested
- You should be consistent in your approach while designing interface.
- It should be processes based rather than the data structure based.

LECTURE NO. 33

Microsoft Access

User will take interest in the application if user interface is friendly. We then discussed different tools, which are used in the development of any application. In this lecture we will see the designing of input forms. An input form is an easy, effective, efficient way to enter data into a table. Input forms are especially useful when the person entering the data is not familiar with the inner workings of Microsoft Access and needs to have a guide in order to input data accurately into the appropriate fields. Microsoft Access provides several predefined forms and provides a forms wizard that walks you through the process of creating a form. One of these predefined forms will be used in the example below. You can also create your own customized forms by using Microsoft Access form design tools. Following things must be ensured for input forms:

- Forms should be user friendly
- Data integrity must be ensured, which means that database must represent the true picture of real system.
- Checks can be applied within the tables' definition or through input forms so first we will run MS Access and select new option from the file. Next it will ask the name of database as follows.

Lecture No. 34

Classification of Physical Storage Media

Classifications of Physical Storage Media Storage media are classified/ three characteristics according to following characteristics:

- Speed of access
- Cost per unit of data
- Reliability

We can also differentiate storage as either

- Volatile storage
- Non-volatile storage

Computer storage that is lost when the power is turned off is called as **volatile storage**. Computer storage that is not lost when the power is turned off is called as **non – volatile storage**, pronounced cache is a special high-speed storage mechanism. It can be either a reserved section of main memory or an independent high-speed storage device. **Two types of caching are commonly used in personal computers:**

- **Memory caching**
- **Disk caching**

A **memory cache**, sometimes called a **cache store** or **RAM**. Cache is a portion of memory made of high-speed static RAM (SRAM) instead of the slower and cheaper DRAM used for main memory. The Intel 80486 microprocessor, for example, contains an 8K memory cache, and the Pentium has a 16K cache. Such internal caches are **often called Level 1 (L1) caches**. Most modern PCs also come with external cache memory, **called Level 2 (L2) caches**. These caches sit between the CPU and the DRAM. Like L1 caches, L2 caches are composed of SRAM but they are much larger. Disk caching works under the same principle as memory caching, but instead of using high-speed SRAM, **a disk cache uses conventional main memory**.

Cache hit

When data is found in the cache, it is called a cache hit, and the effectiveness of a cache is judged by its hit rate. **Many cache systems use a technique known as smart caching, in which the system can recognize certain types of frequently used data.**

RAID – Redundant Array of Inexpensive Disks

Many disks that look as a single disk to OS but have better *performance* and better *reliability*. RAID has the property that the **data are distributed over the drives to allow parallel operations.**

RAID 0

- **Simple Striping**
- **Virtual single disk is divided up into strips of k sectors each.**

- RAID 0 writes consecutive stripes over the drives in round robin fashion.
- Controller will break any disk command into k commands (one for each of the disks) and will read/write in parallel.

RAID 1

- It duplicates all the disks.
- On write, every stripe is written twice. On a read, either copy can be used.
- *Fault tolerance* is excellent: if a drive crashes, the copy from the duplicated disk is simply used instead.

RAID-2

- RAID Level 2, which uses Hamming error correction codes, is intended for use with drives which do not have built-in error detection. All SCSI drives support built-in error detection, so this level is of little use when using SCSI drives.

RAID-3

- RAID Level 3 stripes data at a byte level across several drives, with parity stored on one drive.

RAID-4

- RAID Level 4 stripes data at a block level across several drives, with parity stored on one drive. The parity information allows recovery from the failure of any single drive.

RAID-5

- RAID Level 5 is similar to level 4, but distributes parity among the drives. However, the performance for reads tends to be considerably lower than a level 4 array. The cost per megabyte is the same as for level 4. The manner data records are stored and retrieved on physical devices. The technique used to find and retrieve store records are called access methods.

Sequential File Organization

Records are arranged on storage devices in some sequence based on the value of some field, called sequence field. Sequence field is often the key field that identifies the record.

Lecture No. 35

File Organizations

- The **File** is a collection of records. Using the primary key, we can access the records. The type and frequency of access can be determined by the type of file organization which was used for a given set of records.
- File organization is a logical relationship among various records. This method defines how file records are mapped onto disk blocks.
- File organization is used to describe the way in which the records are stored in terms of blocks, and the blocks are placed on the storage medium.
- The first approach to map the database to the file is to use the several files and store only one fixed length record in any given file. An alternative approach is to structure our files so that we can contain multiple lengths for records.
- Files of fixed length records are easier to implement than the files of variable length records.

Sequential files provide access only in a particular sequence. That does not suit many applications since it involves too much time. Some mechanism for direct access is required.

Direct Access File Organization:

For most users, the file system is the most visible aspect of an operating system. Finally, when multiple users have access to files, it may be desirable to control by whom and in what ways files may be accessed. This control is known as file protection. Following are the two types:

- Indexed Sequential
- Direct File Organization

Indexed sequential file:

As the name suggests, records are stored in a sequence (generally w.r.t. PK). But index is also maintained to provide direct access. Index is a special file, sort of a table that lists the key value for each record and the address of the record. Address may be the track no. In case of direct access requirement, index is accessed rather than the file itself. Location of the record is obtained from index and then the record is accessed from that location.

Dense Key Index

- An index that stores entry for each record.
- One entry for each record is wastage, since within the track search has to be sequential.

Nondense Index

- Rather than entry for every record, entry for the record with highest key value in the track is maintained.
- Index size reduces, search also gets more efficient.

Defining Keys:

An indexed sequential file must have at least one key. The first (primary) key is always numbered 0. An indexed sequential file can have up to 255 keys; however, for file-processing efficiency it is recommended that you define no more than 7 or 8 keys. (The time required to insert a new record or update an existing record is directly related to the number of keys defined; the retrieval time for an existing record, however, is unaffected by the number of keys.) You must define each key in the following terms:

- Position and size
- Data type
- Index number
- Options selected

Lecture No. 36

Hashing

A hash function is computed on some attribute of each record. The result of the function specifies in which block of the file the record should be placed. Hashing provides rapid, non-sequential, direct access to records. A key record field is used to calculate the record address by subjecting it to some calculation; a process called hashing.

Hashing involves computing the address of a data item by computing a function on the search key value. A hash function h is a function from the set of all search key values K to the set of all bucket addresses B .

Hash Functions

Independent of the number of search keys. We hope records are distributed uniformly among the buckets. The worst hash function maps all keys to the same bucket. The best hash function maps all keys to distinct addresses. Ideally, distribution of keys to addresses is uniform and random.

Hashed Access Characteristics

Following are the major characteristics:

- No indexes to search or maintain
- Very fast direct access
- Inefficient sequential access
- Use when direct access is needed, but sequential access is not.

Mapping functions

Map () SQL function is used to create a map column of Map Type on Data Frame. The input columns to the map function must be grouped as key-value pairs. e.g. (key1, value1, key2, value2 ...).

Note: All key columns must have the same data type, and can't be null and All value columns must have the same data type.

Handling the Collisions

Multiple keys map to the same integer, then elements with different keys may be stored in the same "slot" of the hash table. It is clear that when the hash function is used to locate a potential match, it will be necessary to compare the key of that element with the search key. But there may be more than one element, which should be stored in a single slot of the table. Various techniques are used to manage this problem:

- Chaining,
- Overflow areas,
- Re-hashing,
- Using neighboring slots (linear probing),
- Quadratic probing,
- Random probing,

Chaining:

One simple scheme is to chain all collisions in lists attached to the appropriate slot. The tradeoff is the same as with linked lists versus array implementations of collections: linked list overhead in space and, to a lesser extent, in time.

Overflow areas

In general, a data type overflow error is when the data type used to store data was not large enough to hold the data. Furthermore, some data types can only store numbers up to a certain size. An overflow error will be produced, for example, if a data type is a single byte and the data to be stored is greater than 256.

Re-hashing:

Re-hashing schemes use a second hashing operation when there is a collision. If there is a further collision, we re-hash until an empty "slot" in the table is found. The re-hashing function can either be a new function or a re-application of the original one. As long as the functions are applied to a key in the same order, then a sought key can always be located.

Linear probing:

The simplest approach to resolve a collision is linear probing. In this technique, if a value is already stored at a location generated by $h(k)$, it means collision occurred then we do a sequential search to find the empty location. Here the idea is to place a value in the next available position.

Clustering:

Linear probing is subject to a clustering phenomenon. Re-hashes from one location occupy a block of slots in the table which "grows" towards slots to which other keys hash. This exacerbates the collision problem and the number of re-hashed can become large.

Collision handling

A well-chosen hash function can avoid anomalies which are result of an excessive number of collisions, but does not eliminate collisions. We need some method for handling collisions when they occur. We'll consider the following techniques:

- Open addressing
- Chaining

Open addressing

The hash table is an array of (key, value) pairs. The basic idea is that when a (key, value) pair is inserted into the array, and a collision occurs, the entry is simply inserted at an alternative location in the array. Linear probing, double hashing, and rehashing are all different ways of choosing an alternative location. The simplest probing method is called linear probing.

Lecture No. 37

Index

Any subset of the fields of a relation can be the search key for an index on the relation. *Search key* is not the same as *key* (e.g. doesn't have to be unique ID). An index contains a collection of *data entries*, and supports efficient retrieval of all records with a given search key value *k*. Typically, index also contains auxiliary information that directs searches to the desired data entries. Can have multiple (different) indexes per file for example file sorted by *stid*, with a hash index on *cgpa* and a B+tree index on *sname*. Indexes on primary key and on attribute(s) in the unique constraint are automatically created. We can still create more.

Index Classification

Indexes are classified as under:

- Clustered vs. Un-clustered Indexes
- Single Key vs. Composite Indexes

- Tree-based, inverted files, pointers

Primary Indexes:

Consider a table with a Primary Key Attribute being used to store it as an ordered array. As we know, each BLOCK of memory will store a few records of this table. First need to know which block it is stored in. If we know the address of the block where the record is stored, searching for the record is very fast. Notice also that we can order the records using the Primary Key attribute values. Thus, if we just know the primary key attribute value of the first record in each block.

Secondary Indexes:

Users often need to access data on the basis of non-key or non-unique attribute; secondary key. Like student name, program name, students enrolled in a particular program. Records are stored on the basis of key attribute; three possibilities.

- Sequential search
- Sorted on the basis of name
- Sort for command execution

Index on secondary key is solution. Data remains in original order, however secondary key index maintains ordered sequence of records, and provides direct access. Three implementation approaches

- Inverted files or inversions
- Linked lists
- B+ Trees

Creating Index

```
CREATE [ UNIQUE ]
[ CLUSTERED | NONCLUSTERED ]
INDEX index_name
ON { table | view } ( column [ ASC | DESC ] [ ,...n ]
```

We will now see an example of creating index as under:

```
CREATE UNIQUE INDEX pr_prName
ON program(prName)
```

It can also be created on composite attributes. We will see it with an example.

CREATE UNIQUE INDEX
St_Name ON
Student(stName ASC, stFName DESC)

Properties of Indexes:

Following are the major properties of indexes:

- Indexes can be defined even when there is no data in the table
- Existing values are checked on execution of this command
- It support selections of form as under: field <operator> constant
- It supports equality selections as under: Either “tree” or “hash” indexes help here.
- It support Range selections (operator is one among <, >, <=, >=, BETWEEN)

Inverted Files

The attribute on which we want direct access, inverted file is created on that field. File is said to be inverted on that field More than one inversions can be created, even on all attributes. Inverted file contains the values of attribute in sorted order and the relative address of the record. DBMS searches appropriate record no and hands over to access method to retrieve the desired record.

Linked Lists

AKA Pointer Chain Needs an extra field in the physical record to store the link Relative addresses are used Head pointer needs to be stored Pointers for multiple fields can be created Circular pointers Double linked lists.

Tree Index

Index is maintained in hierarchical arrangements of nodes each node stores key values and pointers; nodes are connected through links. There is a special node called the root node each node has a parent node (but the root) and has one or more children (but the leaf nodes) Different types of trees exist.

B+ Trees

Number of pointers in a node is one more than the number of keys. Number of pointers called order of tree Distance of all leaf nodes from the root is same.

Lecture No. 38

Ordered Indices

In order to allow fast random access, an index structure may be used. A file may have several indices on different search keys. If the file containing the records is sequentially ordered, the index whose search key specifies the sequential order of the file is the primary index, or clustering index. Indices whose search key specifies an order different from the sequential order of the file are called the secondary indices, or nonclustering indices.

Clustered vs. unclustered:

If order of data records is the same as order of index data entries, then called *clustered index*. A file can be clustered on at most one search key. Cost of retrieving data records through index varies *greatly* based on whether index is clustered or not.

When to Use Indexes

- Useful on large tables
- Required on PK
- Useful for attributes appearing in where, order by and group by clauses
- Significant variety in values
- Check the limit of indexes with your DBMS
- Make your decision very much justifiable; indexes introduce overhead

Dense and Sparse Indices

There are two types of ordered indices:

- Dense Index
- Sparse Index

Dense Index:

An index record appears for every search key value in file. This record contains search key value and a pointer to the actual record.

Sparse Index:

Index records are created only for some of the records.

Multi-Level Indices

Use binary search on outer index. Scan index block found until correct index record found. Use index record as before - scan block pointed to for desired record. For very large files, additional levels of indexing may be required. Indices must be updated at all levels when insertions or deletions require it. Frequently, each level of index corresponds to a unit of physical storage (e.g. indices at the level of track, cylinder and disk).

Two-level sparse index

Index Update

Regardless of what form of index is used, every index must be updated whenever a record is either inserted into or deleted from the file.

- Deletion: Find (look up) the record. If the last record with a particular search key value, delete that search key value from index.
- Insertion: Find place to insert. Dense index: insert search key value if not present. Sparse index: no change unless new block is created. (In this case, the first search key value appearing in the new block is inserted into the index).

Indexes Using Composite Search Keys

The search key for an index can contain several fields; such keys are called composite search keys or concatenated keys. The hashed file organization supports only equality queries, since a hash function identifies the bucket containing desired records only if a value is specified for each field in the search key. A range query is one in which not all fields in the search key are bound to constants.

Lecture No. 39 and 40

Views

Views are generally used to focus, simplify, and customize the perception each user has of the database. Views can be used as security mechanisms by allowing users to access data through the view, without granting the users permissions to directly access the underlying base tables of the view.

Characteristics of Views

Not exactly the external views of 3-level schema architecture Major Types.

- Dynamic

- Materialized
- Partitioned

Dynamic Views

Data is not stored for the views Definition are stored in the schema Executed every time view is referred.

```
CREATE VIEW [ < database_name > . ] [ < owner > . ] view_name [ ( column [
,...n ] ) ]
[ WITH ENCRYPTION | SCHEMABINDING [ ,...n ] ]
AS
select_statement
[ WITH CHECK OPTION ]
```

Simple Views

As defined earlier simple views are created from tables and are used for creating secure manipulation over the tables or structures of the database. Views make the manipulations easier to perform on the database.

Complex Views

Complex views are by definition views of type which may comprise of many of elements, such as tables, views sequences and other similar objects of the database.

Deleting Views:

A view can be dropped using the DROP VIEW command, which is just like DROP TABLE.

Updates on Views

Updating a view is quite simple and is performed in the same way as we perform updates on any of the database relations. But this simplicity is limited to those views only which are created using a single relation. Those views which comprise of multiple relations the updation are hard to perform and needs additional care and precaution.

Lecture No. 41

Materialized Views

A pre-computed table comprising aggregated or joined data from fact and possibly dimensions tables. Also known as summary or aggregate table. Views are virtual tables. In which query is executed every time .For complex queries involving large number of join rows and aggregate functions, so it is problematic. Its solution is materialized views also called indexed views created through clustered index.

An indexed view also automatically reflects modifications made to the data in the base tables after the index is created, the same way an index created on a base table does.

Materialized views are schema objects that can be used to summarize, compute, replicate, and distribute data. They are suitable in various computing environments such as data warehousing, decision support, and distributed or mobile computing:

- In data warehouses, materialized views are used to compute and store aggregated data such as sums and averages. Materialized views in these environments are typically referred to as summaries can be used for queries that include filter selections.
- Cost-based optimization can use materialized views to improve query performance by automatically recognizing when a materialized view can and should be used to satisfy a request.
- In distributed environments, materialized views are used to replicate data at distributed sites and synchronize updates done at several sites with conflict resolution methods.
- In mobile computing environments, materialized views are used to download a subset of data from central servers to mobile clients, with periodic refreshes from the central servers and propagation of updates by clients back to the central servers.

Transaction Management

A transaction can be defined as an indivisible unit of work comprised of several operations, all or none of which must be performed in order to preserve data integrity these two operations must be applied together or not at all. Thus, they constitute a transaction.

Properties of a transaction

All transactions share these properties: atomicity, consistency, isolation, and durability (represented by the acronym ACID).

- Atomicity:
This implies indivisibility; any indivisible operation (one which will either complete fully or not at all) is said to be atomic.
- Consistency:
A transaction must transition persistent data from one consistent state to another. If a failure occurs during processing, the data must be restored to the state it was in prior to the transaction.
- Isolation:
Transactions should not affect each other. A transaction in progress, not yet committed or rolled back (these terms are explained at the end of this section), must be isolated from other transactions. Although several transactions may run concurrently, it should appear to each that all the others completed before or after it; all such concurrent transactions must effectively end in sequential order.
- Durability:
Once a transaction has successfully committed, state changes committed by that transaction must be durable and persistent, despite any failures that occur afterwards.

A transaction can thus end in two ways: a commit, the successful execution of each step in the transaction, or a rollback, which guarantees that none of the steps are executed due to an error in one of those steps.

Transaction

Generally reflects the activities in the real world system one transaction in our exam system may be computing gpa of a student, Operations involved More complex transactions.

Consistent State of DB

Consistent state of the database

- Duplicated data does not conflict
- DB properly reflect the real-world system, i.e. follows the business rules

A transaction should transform database from one consistent state to another consistent state, may be inconsistent temp.

Transaction Termination

Two ways a transaction terminates or ends It may Commit or Abort If executed successfully, brings the database in a new consistent state, said to be committed. If otherwise, database has to be brought in prior consistent state such transaction is rolled back or undone A committed transaction cannot be undone; compensating transaction.

Transaction Boundaries

Means being and end of Tr Generally responsibility of programmer Otherwise approach varies from DBMS to DBMS.

Lecture No. 43

Database Updates

Updates are first made in the buffers in RAM, and then transferred to database. There may be some delay between the two.

Database Recovery

If a crash occurs during this delay then the database is in in consistent state. On restart DBMS has to identify it and recover DB into a consistent state.

Reasons of Failure

- Natural disaster
- Sabotage
- Carelessness
- Disk crash
- System software errors
- Generally, improper shut down of DBMS

Recovery Techniques

System crashes, buffers are lost but the disk copy is safe Need to determine the transactions that need to redone or undone.

Log File

Tool used for database recovery Transaction Record

- <T, starts>
- <T, commits> or <T, aborts>

Transaction Operations

Read (X) $X = X + 5$ Write (X) $Y = Y * 3$ Write (Y) Commit	➤ Operations of DBMS concern ➤ Concern of Recovery Manager ➤ So log file contains entries only for write operations
--	---

Concurrency Control

Concurrency control (CC) is the second part of the transaction management. The objective of the CC is to control the concurrent access of database by multiple users at the same time called the concurrent access.

Problems due to Concurrent Access

If not controlled properly, the concurrent access may introduce following three problems in database:

- Lost Update Problem
- Uncommitted Update Problem
- Inconsistent Analysis Problem

Lost Update Problem

This problem occurs when multiple users want to update same object at the same time.

TIME	TA	TB	BAL
t_1	Read (BAL)		1000
t_2		Read (BAL)	1000
t_3	BAL = BAL - 50		1000
t_4	Write (BAL)		950
t_5		BAL = BAL + 10	950
t_6		Write (BAL)	1010

It reflects a situation when a transaction updates an object and another transaction reads this updated value but later the first transaction is aborted.

TIME	TA	TB	BAL
t ₁	Read (BAL)		1000
t ₂	BAL=BAL+100		1000
t ₃	Write (BAL)		1100
t ₄		Read (BAL)	1100
t ₅		BAL=BAL*1.5	1100
t ₆		Write (BAL)	2650
t ₇	ROLLBACK		?

Inconsistent Analysis

This problem of concurrent access occurs when one transaction operates on many records, meanwhile another modifies some of them effecting result of first. This problem is expressed in the following table:

TIME	TA	TB
t ₁	Read (BAL_A) (5000)	
t ₂	INT = BAL_A * .05	
t ₃	TOT = TOT + INT	
t ₄		
t ₅		Read (BAL_A)
t ₆		BAL_A=BAL_A – 1000
t ₇		Write (BAL_A)
t ₈		Read (BAL_E) (5000)
t ₉		BAL_E = BAL_E + 1000
t ₁₀		Write (BAL_E)
t ₁₁	Read (BAL_E) (6000)	
t ₁₂	INT = BAS_E * .05	
t ₁₃	TOT = TOT + INT	

Serial Execution and history

Serial execution is an execution where transactions are executed in a sequential order, that is, one after another. A transaction may consist of many operations. Serial execution means that all the operations of one transaction are executed first, followed by all the operations of the next transaction and like that. A Schedule or History is a list of operations from one or more transactions. A schedule represents the order of execution of operations. The order of operations in a schedule should be the same as in the transaction. Schedule for a serial execution is called a serial schedule.

Interleaved Schedule

We would prefer concurrent execution, multiple users accessing database at the same time. An interleaved schedule consists of operations from different transactions.

Schedule

Expand SQL Server Agent, expand Jobs, right-click the job you want to schedule, and click Properties. Select the Schedules page, and then click New. In the Name box, type a name for the new schedule. Clear the Enabled check box if you do not want the schedule to take effect immediately following its creation.

$TA = \{Read(X), Write(X), commit\}$

$TB = \{Read(Y), Read(X), Write(Y), commit\}$

$S_1 = \{R_A(X), W_A(X), C_A, R_B(Y), R_B(X), W_B(Y), C_B\}$

$S_2 = \{R_B(Y), R_B(X), W_B(Y), C_B, R_A(X), W_A(X), C_A\}$

$S_3 = \{R_B(Y), R_B(X), R_A(X), W_B(Y), C_B, W_A(X), C_A\}$

$S_4 = \{R_A(X), R_B(Y), R_B(X), W_A(X), C_A, W_B(Y), C_B\}$

Interleaved Schedule

Interleaved schedule provides concurrent execution BUT Three problems of concurrent access may be encountered.

- Two transactions
 - Writing different variables
 - Reading same variable
 - Accessing same variable and one of them writing
- Conflicting operations
- Conflicting transactions

Serializability

An interleaved schedule is serializable if the final state of the schedule is equivalent to a serial schedule OR A schedule where conflicting operations are in a serial order.

Two major approaches

- Locking

➤ Timestamping

Locking

An object is locked before it performs any operation Two types of operations, two types of locks Shared and exclusive.

Transaction A

Transaction B

	Read	Write
Read	<i>Yes</i>	<i>No</i>
Write	<i>No</i>	<i>No</i>

Lecture - 45

Locking Idea

Before performing an operation a transaction applies for an S/E lock on an object If object free or lock compatible then lock granted otherwise transaction will wait.

Deadlock

Deadlock is a situation when two transactions are waiting for each other to release a lock. The transactions involved in deadlock keep on waiting unless deadlock is over.

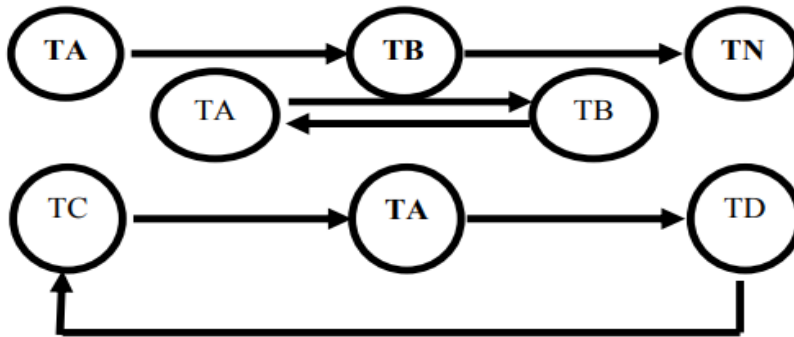
Deadlock Handling

Following are some of the approaches for the deadlock handling:

- Deadlock prevention
- Deadlock detection and resolution
- Prevention is not always possible
- Deadlock is detected by wait-for graph

Wait – for Graph:

It is used for the detection of deadlock. It consists of nodes and links. The nodes represent transaction, whereas arrowhead represents that a transaction has locked a particular data item. We will see it following example:

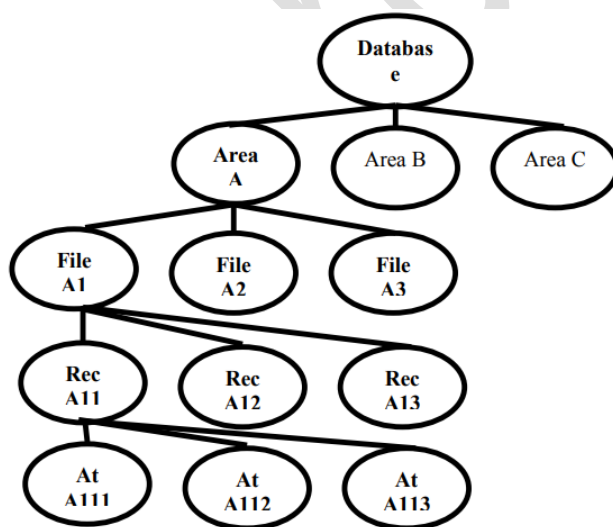


Two-Phase Locking

2PL ensures serializability but generates deadlock. The locks are granted and released in two phases, the growing and shrinking phase. A transaction must acquire a lock before doing any operation. Locks are only granted in growing phase, once a transaction enters in shrinking phase no locks can acquire.

Levels of Locking

Lock can be applied on attribute, record, file, group of files or even entire database. Granularity of locks: Finer the granularity more concurrency but more overhead.



Granularity of Locks

When a lock at lower level is applied, compatibility is checked upward. How will a lock at higher level know about a lock at lower level Individual checking is not feasible.

Intension Locks

When a lock is applied on a node, an intension shared (IS) or intension exclusive (IE) lock is applied on all nodes till the root. An intension lock at a node indicates a lock at the lower level. While applying for a lock on a node, compatibility with the intension lock is also checked Share intension lock (SIX). Conflicts with locks conflicting with S or IX; only one SIX.

EXTENDED COMP MATRIX					
	S	IS	X	IX	SIX
S	Y	Y	N	N	N
IS	Y	Y	N	Y	Y
X	N	N	N	N	N
IX	N	Y	N	Y	N
SIX	N	Y	N	N	N

Time stamping

Two-phase locking is not the only approach to enforcing database consistency. Another method used in some DMBS is timestamping. With timestamping, there are no locks to prevent transactions seeing uncommitted changes, and all physical updates are deferred to commit time.

- Locking synchronizes the interleaved execution of a set of transactions in such a way that it is equivalent to some serial execution of those transactions.
- Timestamping synchronizes that interleaved execution in such a way that it is equivalent to a particular serial order - the order of the timestamps.

Problems of Timestamping

- When a transaction wants to read an item that has been updated by a younger transaction.
- A transaction wants to write an item that has been read or written by a younger transaction.

Haroon Nadir