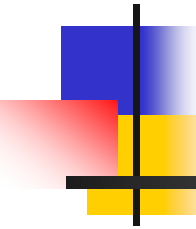


COMP1140: Database and Information Management

Lecture Note – Week 11

Dr Suhuai Luo

School of SEEC

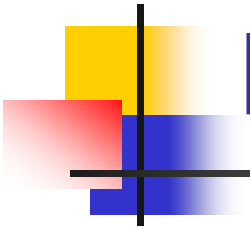
University of Newcastle

Note

- Pls take your time to fill SFC, Thanks. You can do it
  - Either follow the link from your email notification, or
  - Goto myuon.newcastle.edu.au , or
  - the homepage of UONLine (Blackboard)
- SQL test result will be released on BB from Tue 12pm. You can discuss with your instructor on your performance in this week lab only.
- Assignment 3 is due next week at the beginning of your lab session
  - Deliver hard copy to your marker; softcopy to BB as well
  - You are required to attend your lab session, the SQL part of the A3 will be marked on site – no show, no mark
  - Read the assignment specifications & the marking guide; make sure to implement normalised tables; meet constraints etc.
  - More Q & discussion at the end of this lecture
- Course review will be done in lecture next week. Pls come.

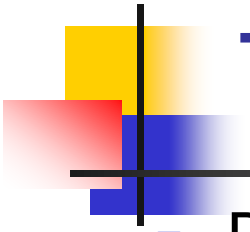
Note

- Pls take your time to fill SFC, Thanks. You can do it
  - Either follow the link from your email notification, or
  - Goto myuon.newcastle.edu.au , or
  - the homepage of UONLine (Blackboard)
- SQL test result will be released on BB from Tue 12pm. You can discuss with your instructor on details of your performance, in your lab this week only.
- Assignment 3 is due next week at the beginning of your lab session
  - Deliver hard copy to your marker; softcopy to BB as well
  - You are required to attend your lab session, the SQL part of the A3 will be marked on site – no show, no mark
  - Read the assignment specifications & the marking guide; make sure to implement normalised tables; meet constraints etc.
  - More Q & discussion at the end of this lecture
- Course review will be done in lecture next week. Pls come.



Last Lecture

- Views
- Transactions
- Triggers
- Stored Procedures



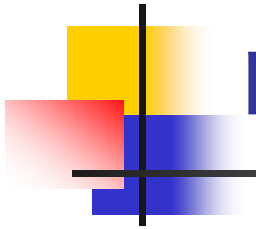
This week

- Discretionary Access Control
  - Schema
  - Privileges
  - Users and Roles
  - GRANT, DENY and REVOKE

  Ref. chapter 20.2, chapter 7, chapter 8
- Physical Database Design
  - Data Storage: Overview
  - Indexes: Overview
  - Query Execution Plans: Overview

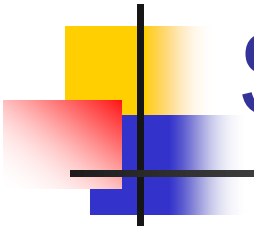
  Ref. chapter 18, appendix F

Database authorization mechanisms



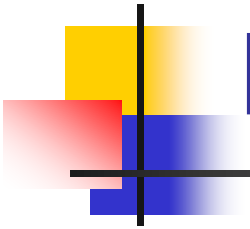
- Mandatory access control - is based on system-wide policies that cannot be changed by individual users
  - Each database object is assigned a certain classification level
  - Each subject is given a designed clearance level
  - A subject requires necessary clearance to read/write a DB object
- Discretionary Access Control - each user is given appropriate access rights/privileges on specific database objects
  - Typically users obtain certain privileges when they create an object and can pass some or all of these privileges to other users
  - SQL supports discretionary access control through statements GRANT & REVOKE

SQL: Naming database objects

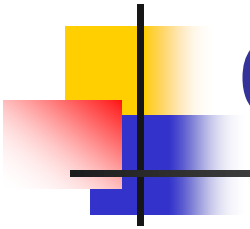


- Database objects are referenced via a hierarchy:
  - Server-Database-Schema-Object
- Each database object has a **fully qualified name** as follows:
  - *ServerName.DatabaseName.SchemaName.ObjectName*
  - The fully qualified name is unique for each database object
  - **ServerName**: Name of the database server
  - **Database**: Name of database within the server
  - **SchemaName**: Name of schema within the database
  - **ObjectName**: Name of the database object

Database Schema



- Relations and other database objects exist in an *environment*.
- Each environment contains one or more *catalogs*, and each catalog consists of set of schemas.
- Schema is named collection of related database objects.
- Objects in a schema can be tables, views, domains, collations, translations, and character sets. All have same owner



CREATE SCHEMA

- CREATE SCHEMA [Name |
  AUTHORIZATION CreatorId]

- DROP SCHEMA Name [RESTRICT | CASCADE]

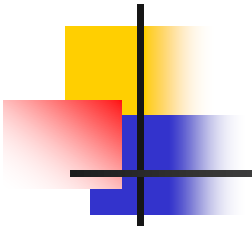
- With RESTRICT (default), schema must be empty or operation fails.
- With CASCADE, operation cascades to drop all objects associated with schema in the order defined when created. If any of these operations fail, DROP SCHEMA fails.

CREATE SCHEMA (cont'd)

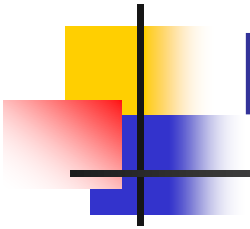
Example:

```
CREATE SCHEMA Sprockets
   AUTHORIZATION Annik
     CREATE TABLE NineProngs (source int,
   cost int, partnumber int)
     GRANT SELECT TO Mandar
     DENY SELECT TO Prasanna;
GO
```

Access Control - Authorization Identifiers and Ownership



- Authorization identifier is normal SQL identifier used to establish identity of a user.
- Usually has an associated password.
- Used to determine which objects user may reference and what operations may be performed on those objects.
- Each object created in SQL has an owner, as defined in AUTHORIZATION clause of schema to which object belongs.
- Owner is the only person who may know about the object.

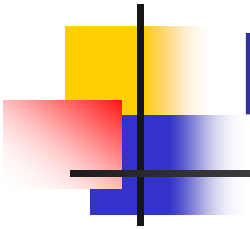


Privileges

- Are the actions that a user is permitted to carry out on a given base table or view
- Common DML privileges
  - **SELECT**: Retrieve data from table/view.
  - **INSERT**: Insert new rows into table/view.
  - **UPDATE**: Modify rows of data in a table/view.
  - **DELETE**: Delete rows of data from a table/view.
  - **REFERENCES**: Reference columns of named table in integrity constraints.
  - **USAGE:** use domains, collations, character sets

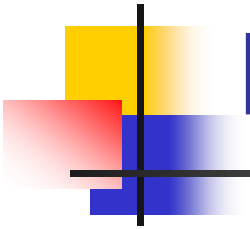
Privileges (contd.)

- Common DDL privileges
  - **ALTER**: Confers the ability to change the properties of object (ability to alter, create, or drop any securable that is contained within that scope).
    - For example, ALTER permission on a schema includes the ability to create, alter, and drop objects from the schema
  - **ALTER ANY**: Confers the ability to CREATE, ALTER, or DROP individual instances of the objects.
    - For example, ALTER ANY SCHEMA confers the ability to create, alter, or drop any schema in the database



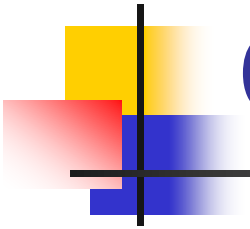
Privileges (contd.)

- Privileges may be granted at different levels
  - Server level
  - Database level
  - Schema level
  - Object level



Privileges (contd.)

- Examples of database-level privileges:
  - **BACKUP DATABASE**: Ability to backup a database
- Example of server-level privileges
  - **ALTER ANY DATABASE**: Ability to create, drop or configure any database



Granting Privileges

- SQL has the GRANT statement used for granting permissions:

- Format:

```
GRANT   {PrivilegeList | ALL PRIVILEGES}
ON      ObjectName
TO      {AuthorizationIdList | PUBLIC}
[WITH GRANT OPTION]
```

Granting Privileges (contd.)

- *PrivilegeList* consists of one or more of above privileges separated by commas.
- ALL PRIVILEGES grants all privileges to a user.
- PUBLIC allows access to be granted to all present and future authorized users.
- *ObjectName* can be a base table, view, domain, character set, collation or translation.
- WITH GRANT OPTION allows privileges to be passed on.

Example: GRANT

- Give John White full privileges to Staff table.

```
GRANT ALL PRIVILEGES
ON Staff
TO JohnWhite WITH GRANT OPTION;
```

- Give users Personnel and Director SELECT and UPDATE on column salary of Staff.

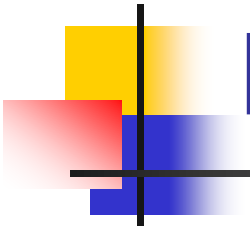
```
GRANT SELECT, UPDATE (salary)
ON Staff
TO Personnel, Director;
```

Example: GRANT Specific Privileges to PUBLIC

- Give all users SELECT on Branch table.

```
GRANT SELECT
ON Branch
TO PUBLIC;
```

Managing Privileges with Roles



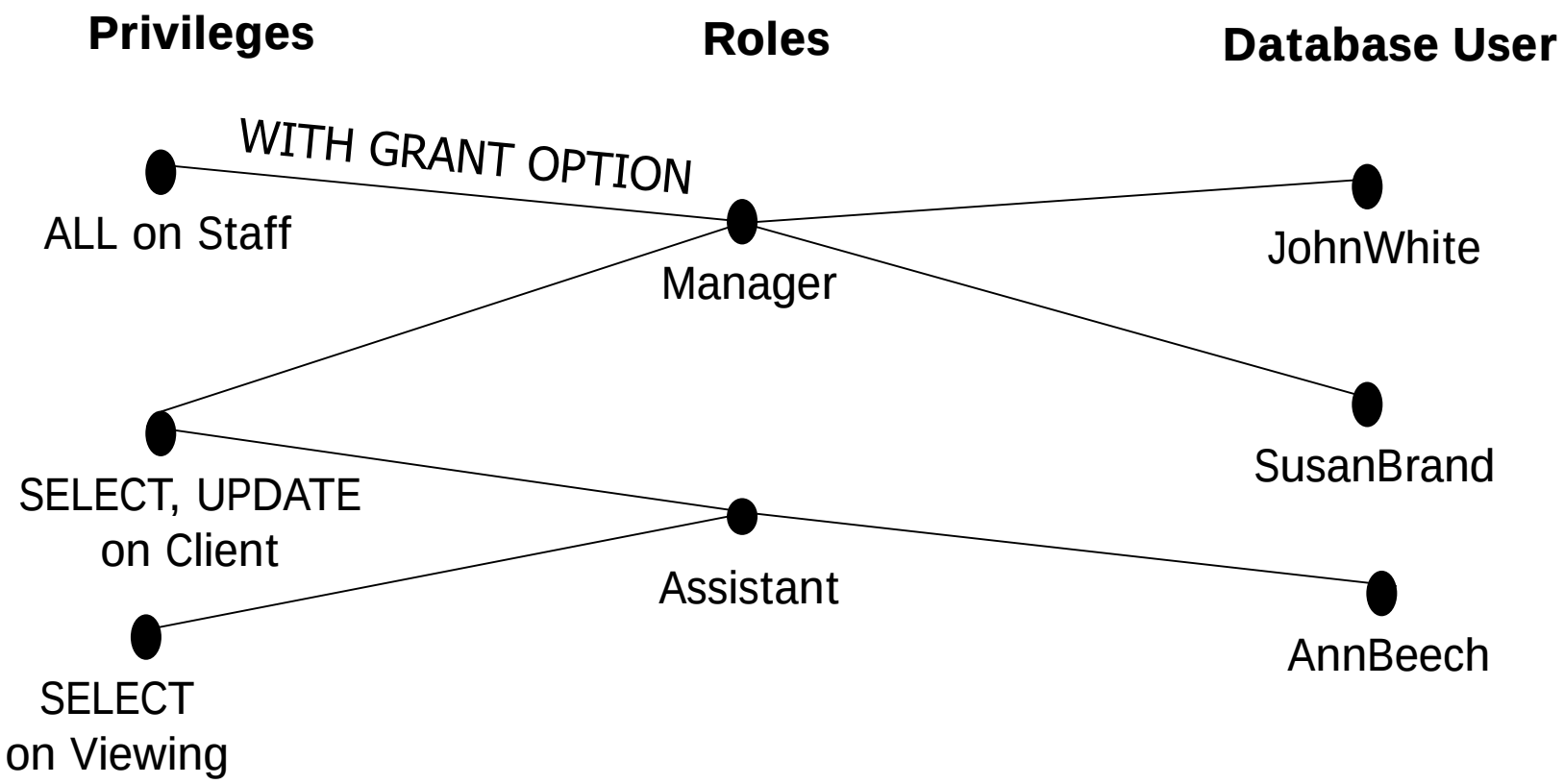
- It becomes quite arduous to manage a large number of users and permissions.
- *Roles* enable group permission and manage it easier.
- Example,
  - CREATE ROLE Manager;

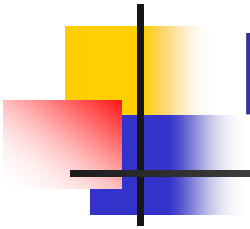
Roles (contd.)

- Now permissions allowed to *managers* can be provided to the Manager role.
  - GRANT ALL PRIVILEGES
    ON Staff
    TO Manager

- By adding database user “John White” as a member of the Manager role, JohnWhite will acquire all privileges of *Manager* role

Roles (contd.)


Privileges
Roles
Database User
WITH GRANT OPTION
ALL on Staff
Manager
JohnWhite
SELECT, UPDATE
on Client
SusanBrand
Assistant
SELECT
on Viewing
AnnBeech

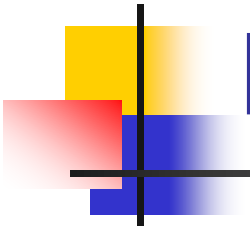


Roles (contd.)

- Managing permission would be convenient
  - Add a permission to all managers => Add a permission to role manager
  - A new manager is recruited. => Add new database user to role manager

  Etc.

- Drop a role with DROP ROLE statement



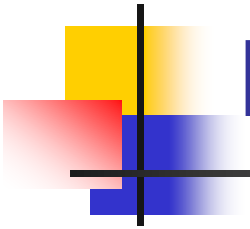
DENY

- Denies a permission to a principal.
- Prevents that principal from inheriting the privilege through its role memberships
- Basic Syntax:
  - DENY { ALL [PRIVILEGES] | *permission*}
    ON ObjectName
    TO AuthorizationIdList

Example: DENY

- Deny SELECT privilege on Viewings table to JohnWhite (even inheriting the privilege through role membership)

```
DENY SELECT
ON Viewings
TO JohnWhite
```



REVOKE

- **REVOKE** removes a previously granted or denied permission

```
REVOKE [GRANT OPTION FOR]
  {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
    [RESTRICT | CASCADE]
```

- ALL PRIVILEGES refers to all privileges granted to a user by the user who is revoking the privileges.

Example: REVOKE Specific Privileges

Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT
ON Branch
FROM PUBLIC;
```

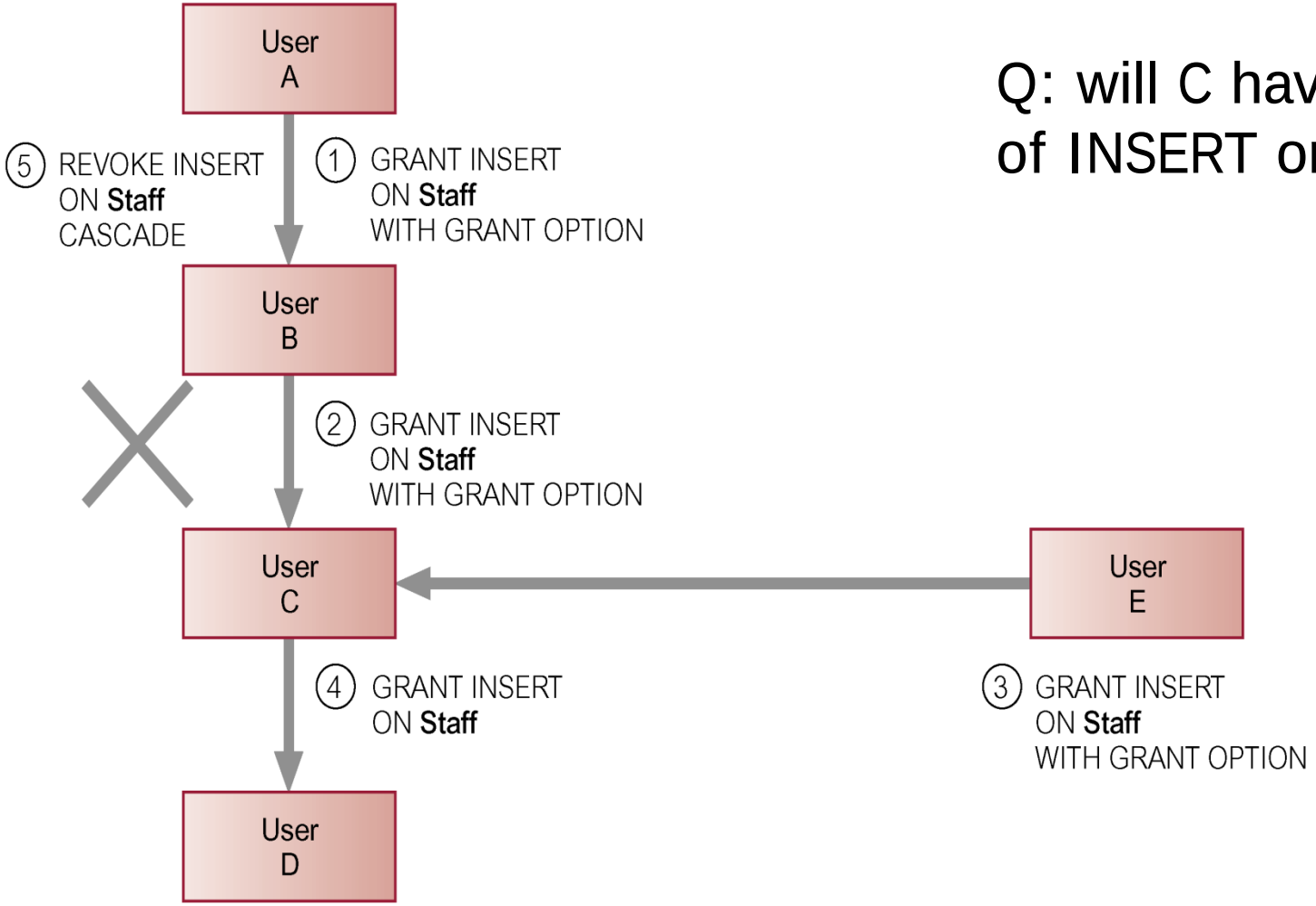
Revoke all privileges given to Director on Staff table.

```
REVOKE ALL PRIVILEGES
ON Staff
FROM Director;
```

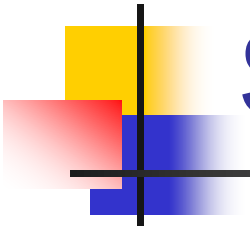
REVOKE (contd.)

- GRANT OPTION FOR allows privileges passed on via WITH GRANT OPTION of GRANT(i.e., the right to grant to 3rd party) to be revoked separately from the privileges themselves.
- REVOKE fails if it results in an abandoned object, such as an abandoned view, unless the CASCADE keyword has been specified.
- Privileges granted to this user by other users are not affected.

REVOKE (contd.)

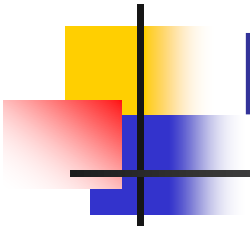


Q: will C have privilege of INSERT on staff?



Summary

- Database object hierarchy
  - Server-Database-Schema-Object
- Permissions
  - DML: INSERT, UPDATE, DELETE,...
  - DDL: ALTER, ALTER ANY,...
  - Permission at different levels:
    - Server level: ALTER ANY DATABASE,...
    - Database level: BACKUP DATABASE,...
- Roles and Users
- GRANT, DENY and REVOKE



Physical Database Design

- Data Storage: Overview
- Indexes: Overview
- Query Execution Plans: Overview

Database Design Process revisit

- Database design process consists of the following main steps:
  - Requirements Analysis
  - Conceptual Database Design
  - Logical Database Design
  - Physical Database Design

Physical Database Design

- Physical Database Design pertains to developing an implementation for the database targeting a specified DBMS

- At this stage, the database designer decides on implementation being well-aware of the features, capabilities and limitations of the ***target DBMS***.

Physical Database Design (contd.)

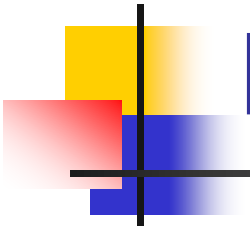
- This process includes:
  1. Translating logical data model for target DBMS (design base relations, constraints, derived data)
  2. Design file organisations and indexes focusing on performance
  3. Design user views and security measures

*Steps 1 and 3 has been covered to a certain extent in previous lectures/tutes/pracs

** Today, we'll introduce step 2

Design File Organizations and Indexes

- Determine and implement the optimal file organizations to store the base relations and the indexes that are required to achieve acceptable **performance**;
- Must understand the typical *workload* that database must support.
- Workload includes:
  - Most frequent/important queries and updates
  - Performance requirement for them

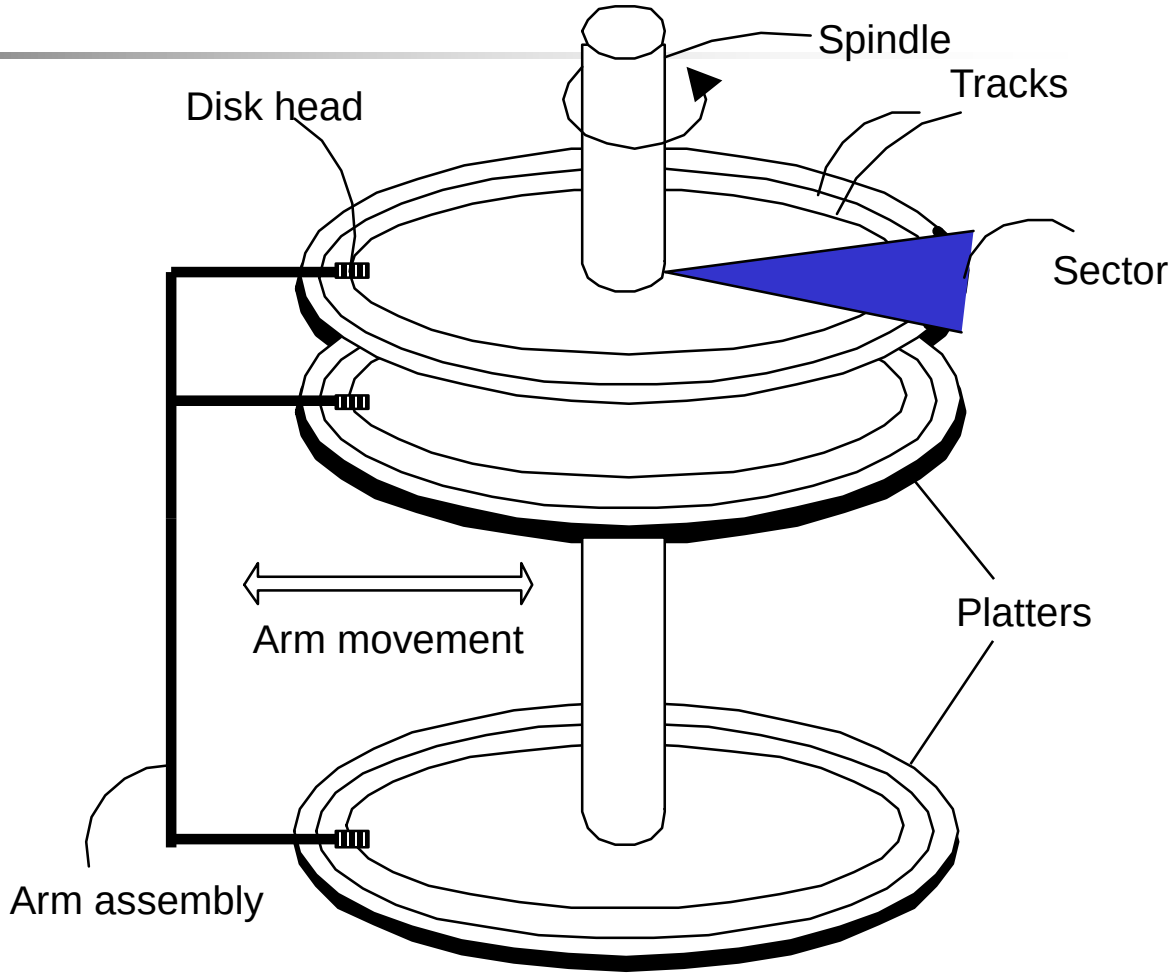


Data Storage: Disks and Files

- How is data stored in databases?
  - Secondary storage (typically on disks)
- Data are brought into memory from disk prior to db operations
- In terms of performance, most significant is the time pertaining to disk read and write (disk I/Os)
- Most DBMSs focus on reducing the time and access to the disk by implementing varied techniques

Disks

- Disk has platters
- Platters contains many tracks
- Each track is divided into smaller sector where data is stored
- Number of sectors make a block which is read/written from disk to memory (read) and vice-versa (write)

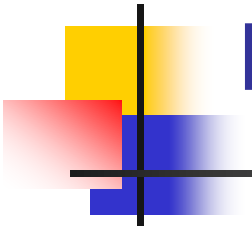


Accessing a Disk Page

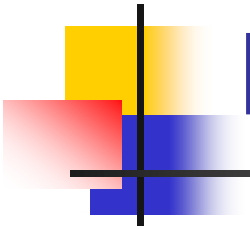
- Time to access (read/write) a disk block:
  - *seek time* (moving arms to position disk head on track)
  - *rotational delay* (waiting for block to rotate under head)
  - *transfer time* (actually moving data to/from disk surface)

Disk I/O takes significantly more time than a memory I/O

Why Not Store Everything in Main Memory?



- *Costs too much*.
- *Main memory is volatile*. We want data to be saved between runs. (Obviously!)
- Typical storage hierarchy:
  - Main memory (RAM) for currently used data (primary storage).
  - Disk for the main database (secondary storage).
  - Disk for archiving older versions of the data (tertiary storage).

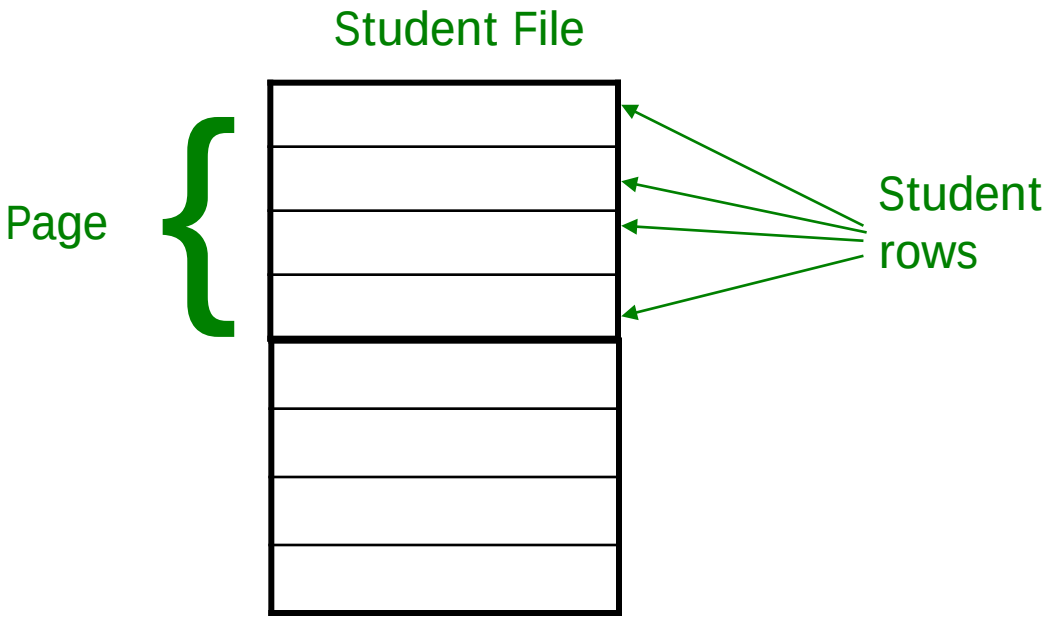


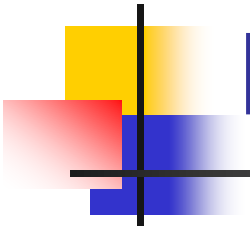
File Organization

- DBMS store tables as logical files
- Each *file* contains rows of the table
- Physically, a file contains a number of *pages*
- Size of page is same as size of block.

File Organization (contd.)

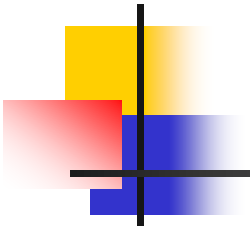
Example





File Organization

- Types of file organizations:
  - **Sequential file**: Order rows in the table according to certain attribute(s) values.
  - **Heap file**: Store rows in no particular order
  - **Hashed file**: Store rows based on a hash attribute(s) values

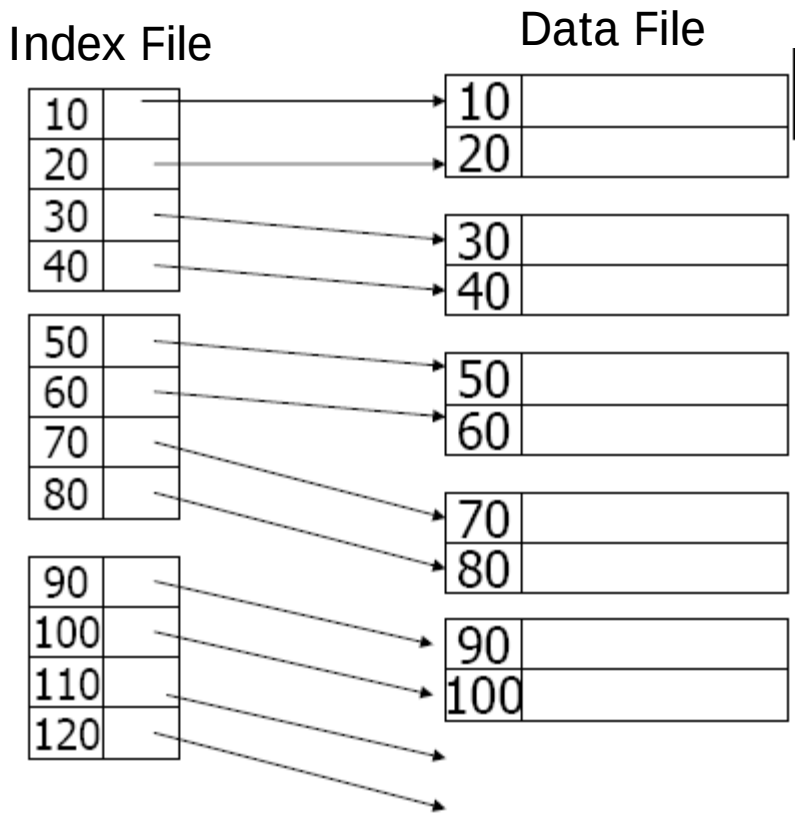


Indexes

- Indexes are auxiliary files that speed up selections on the rows of a table.
- Any subset of the fields of a relation can be the search key for an index on the relation.

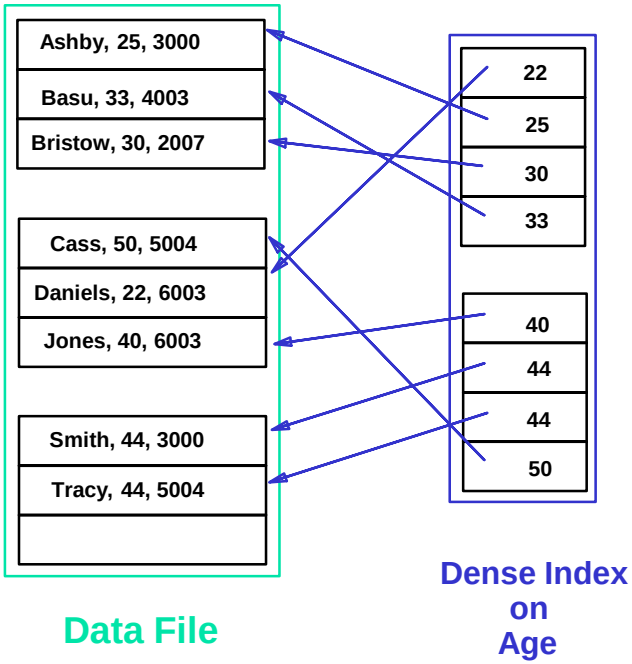
Indexes (contd.)

- Index file is smaller than the data file!
- Index file is always sorted or hashed!
- Searching the index using a search key field is usually efficient than searching a data file

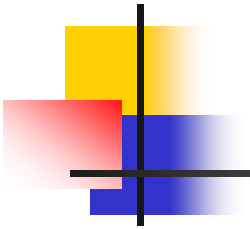


Indexes (contd.)

- Example



Indexes (contd.)

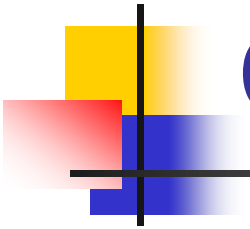


- Different types of data structures exist to build indexes which allow very efficient access to rows in a table

- Example of data structures:
  - B+ Trees
  - Hash
  - Bitmap

  etc.

  * Discussing these in detail is beyond the scope of this course

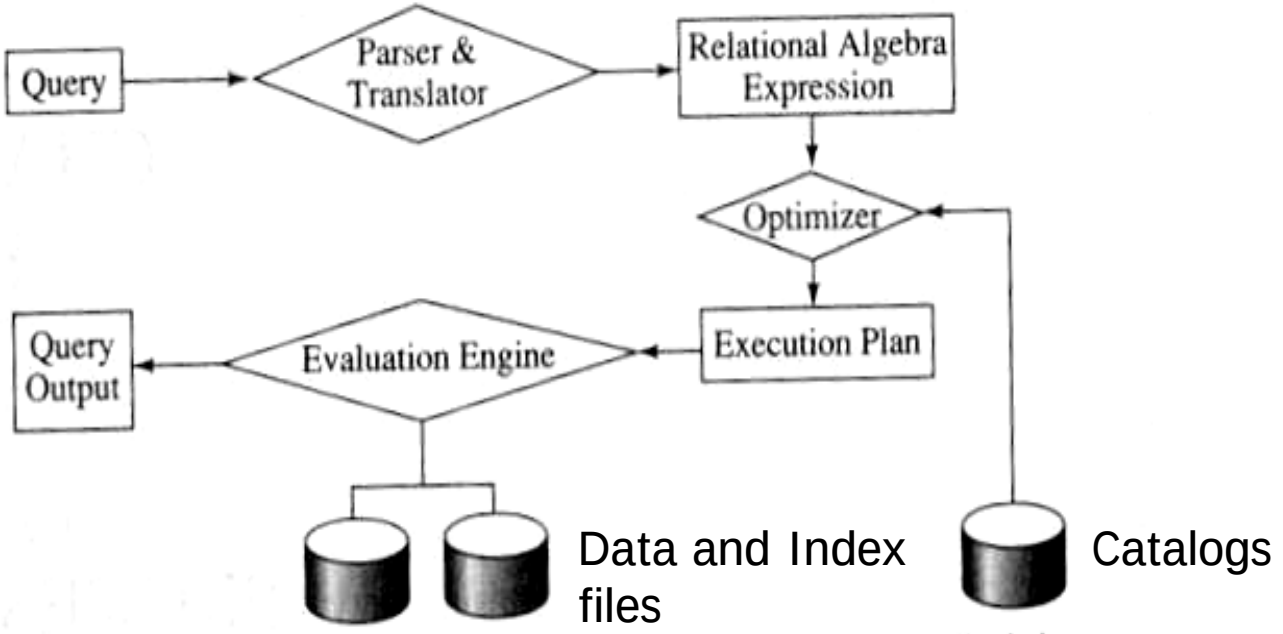


Query Execution Plans

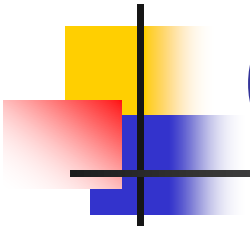
- Every DBMS goes through a process called **query processing** prior to returning results when an SQL statement is given.

Steps in query processing

- The steps involved in query processing include...
  - Parsing and translation
  - Optimization
  - Evaluation

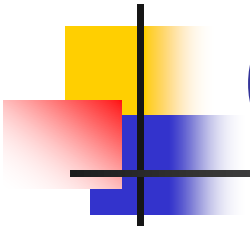


Steps in query processing (contd.)

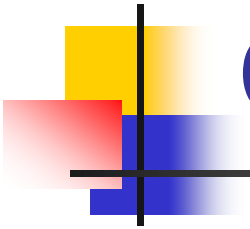


- Step 1: Parsing and translation
  - An SQL statement is verified for correct syntax and semantics
  - The SQL statement is converted to a extended relational algebra expression

Steps in query processing (contd.)



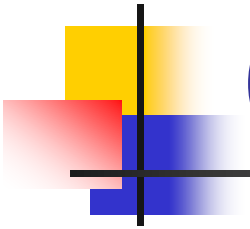
- Step 2: Optimization
  - An efficient query execution strategy is devised (based on existing indexes, file organizations etc.)



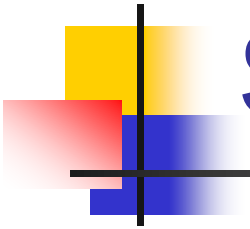
Optimization

- Heart of performance in a DBMSs
- Involves complex algorithms and techniques
- A simple example:
  - Asking a balance of a particular account in a million page Account table, with appropriate indexes and a optimized plan, returns results within milliseconds while an hour in an unoptimized plan.

Steps in query processing (contd.)

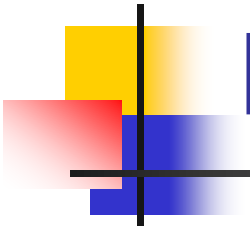


- Step 3: Evaluation
  - The selected query execution plan is evaluated to obtain the results



Summary

- Physical Database Design pertaining to performance requires:
  - Analysis of the typical workload
  - An understanding of capabilities and limitations of target DBMS
  - A good understanding of storage structures, files organization, indexes and query execution plans



Lab this week

- Review and practice on Discretionary Access Control, and Physical Database Design

Issues and discussions on Assignment 3

- A3 specification
- A3 marking guide