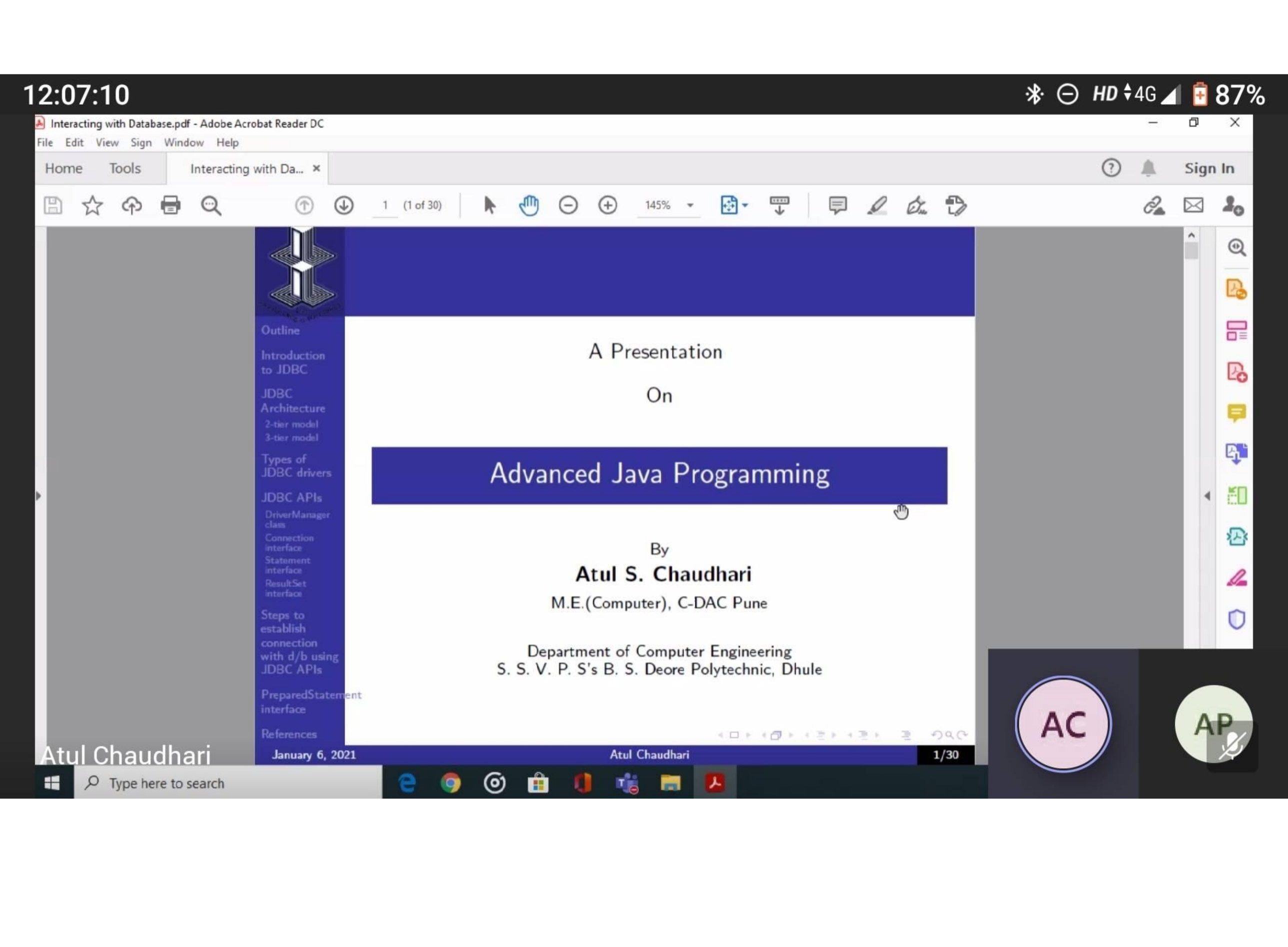
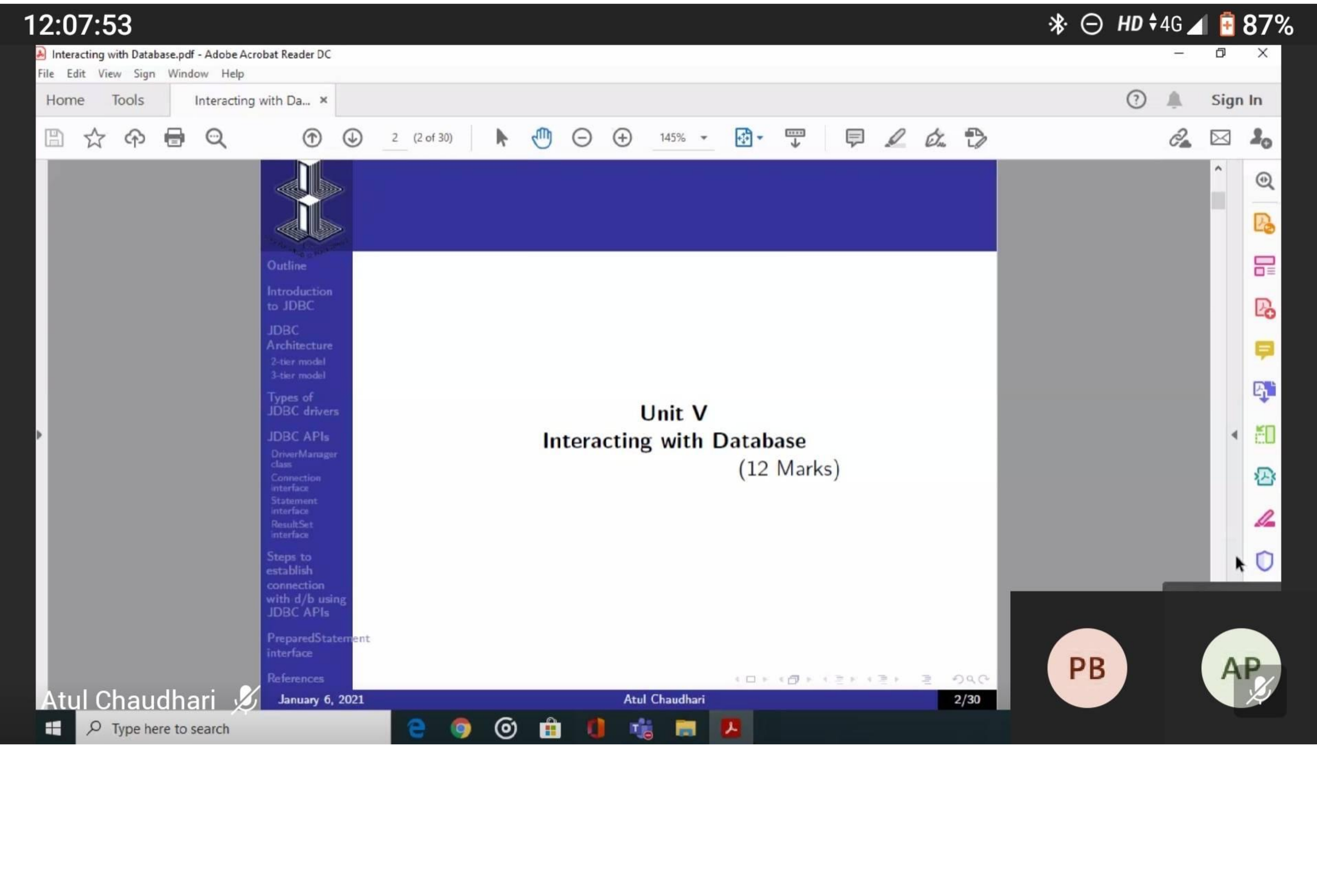
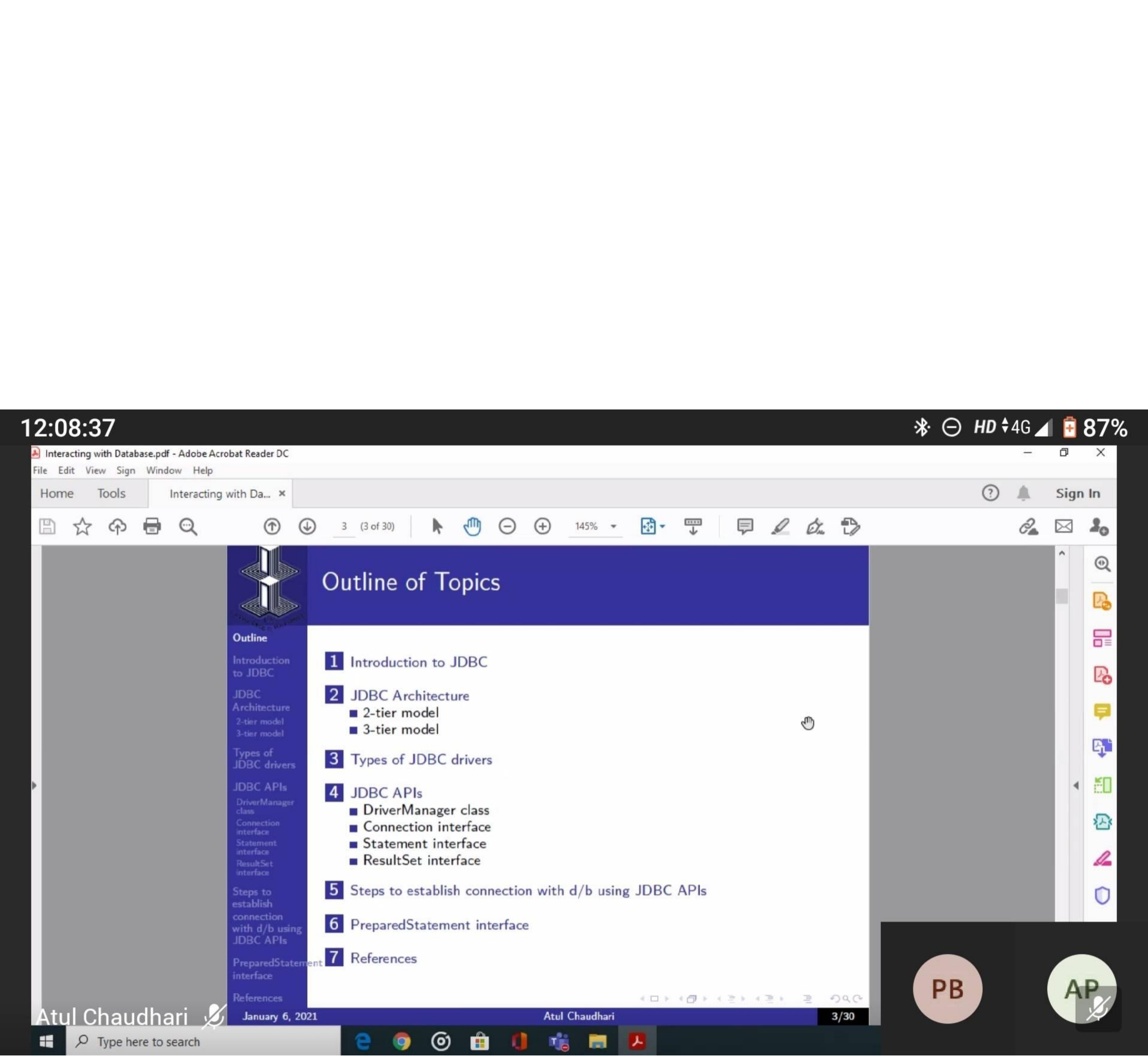
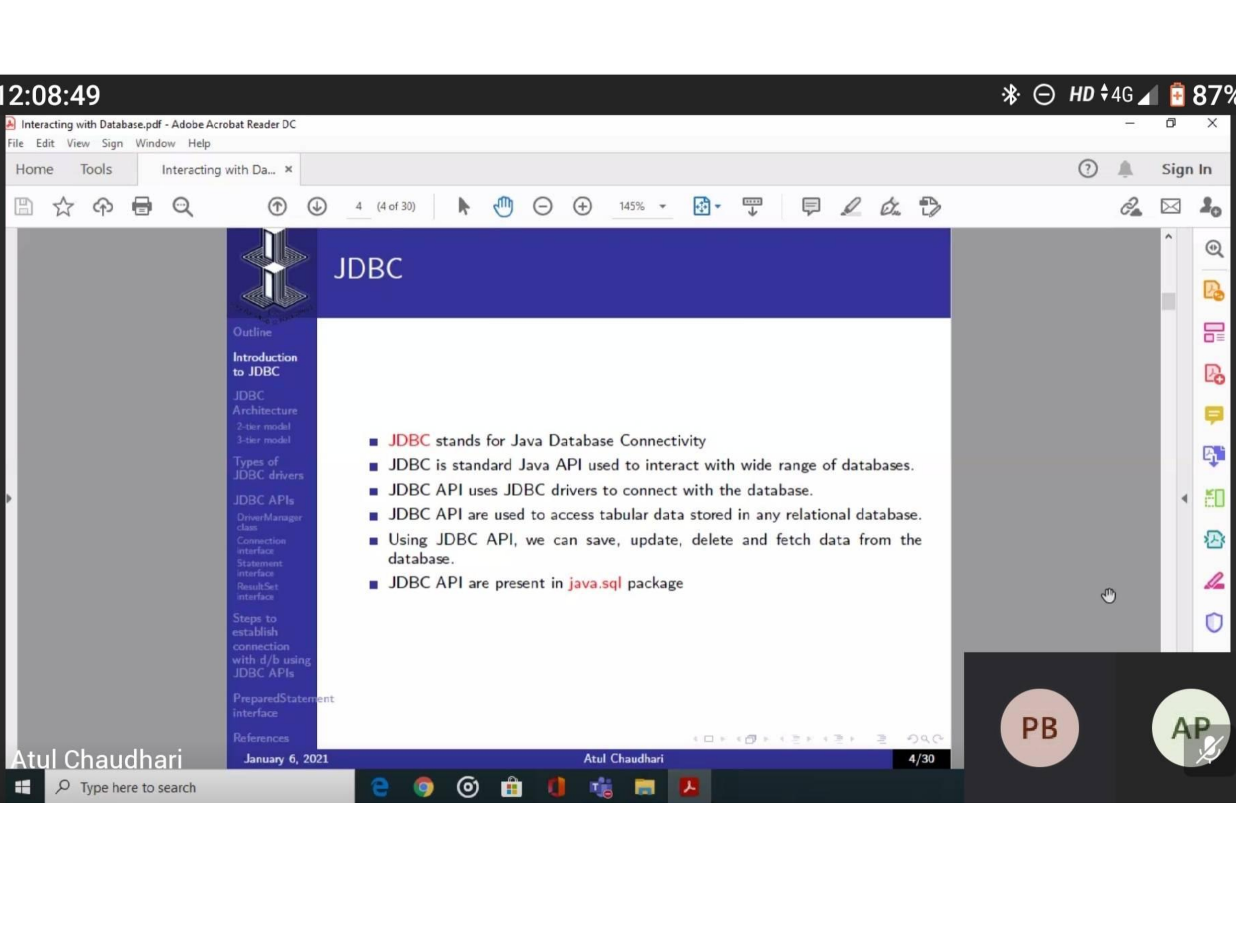








ODBC

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- ODBC stands for Open Database Connectivity
- Before JDBC, ODBC API was the database API to connect and execute the query with the database.
- ODBC API uses ODBC driver which is written in C language
- That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

12:09:18

Interacting with Database.pdf - Adobe Acrobat Reader DC

File Edit View Sign Window Help

Home Tools Interacting with Da... Sign In

6 (6 of 30) 145%

JDBC Architecture

Outline

Introduction to JDBC

JDBC Architecture

2-tier model

3-tier model

Types of JDBC drivers

JDBC APIs

DriverManager class

Connection interface

Statement interface

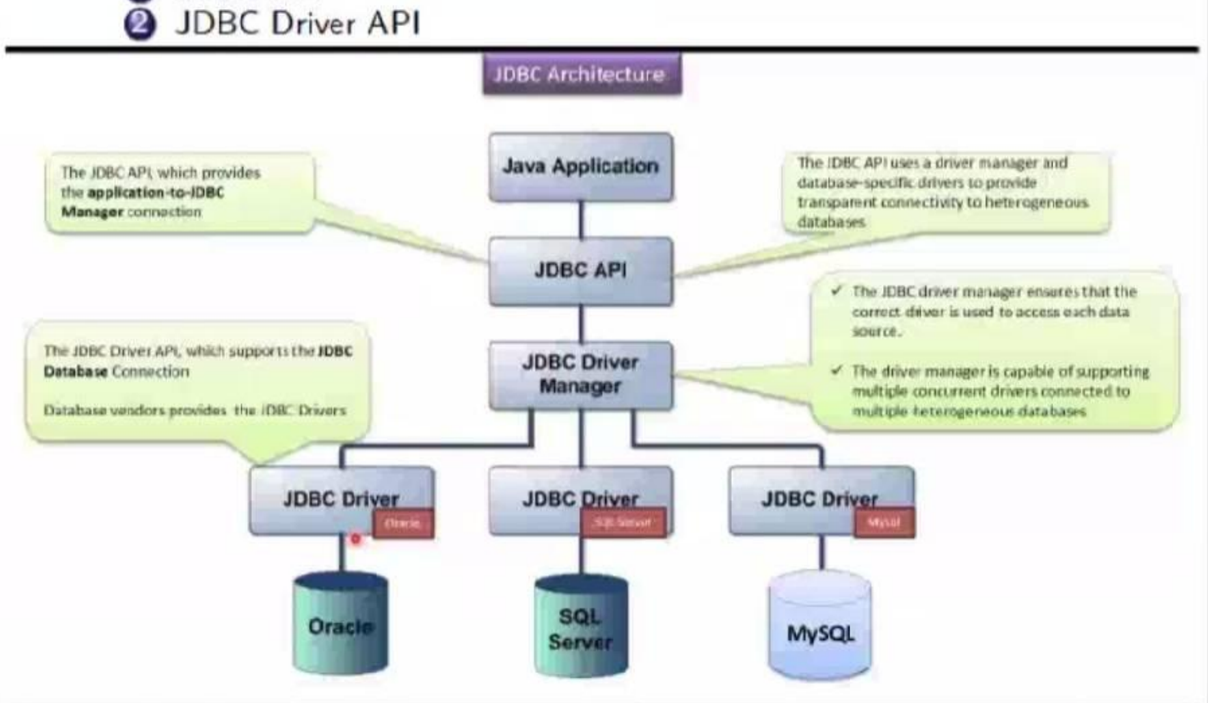
ResultSet interface

Steps to establish connection with d/b using JDBC APIs

PreparedStatement interface

References

- JDBC API supports both 2-tier and 3-tier processing models for d/b access.
- In general, JDBC Architecture consists of two layers -
  1. JDBC API
  2. JDBC Driver API



January 6, 2021 Atul Chaudhari 6/30

Atul Chaudhari

PM

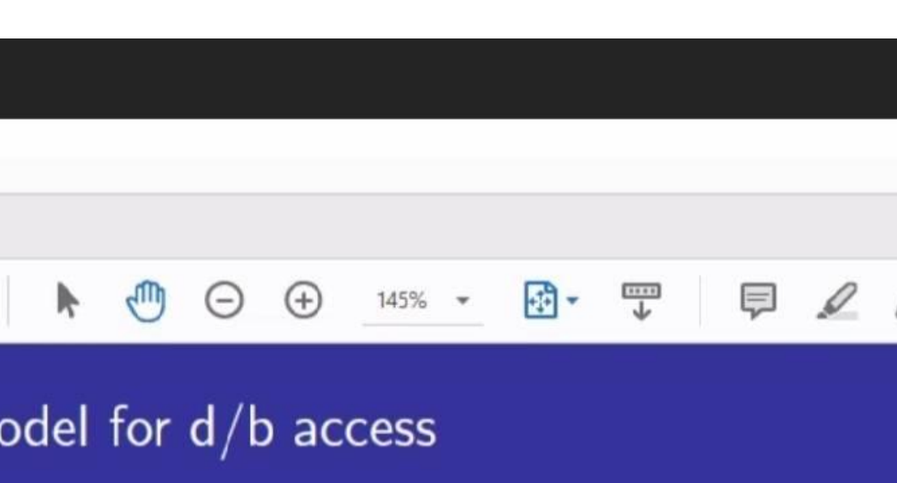
AP

Type here to search

2-tier model for d/b access

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

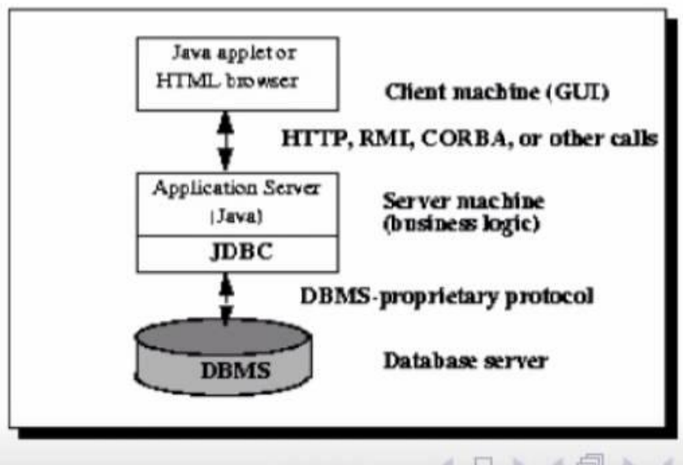
- In the two-tier model, a Java application talks directly to the d/b.
- In this model JDBC driver are used to communicate with the particular d/b
- A user's commands are delivered to the d/b and the results of those statements are sent back to the user.
- In this model d/b may be located on another machine to which the user is connected via a network.
- This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server.
- Because of tight coupling 2-tier application runs faster

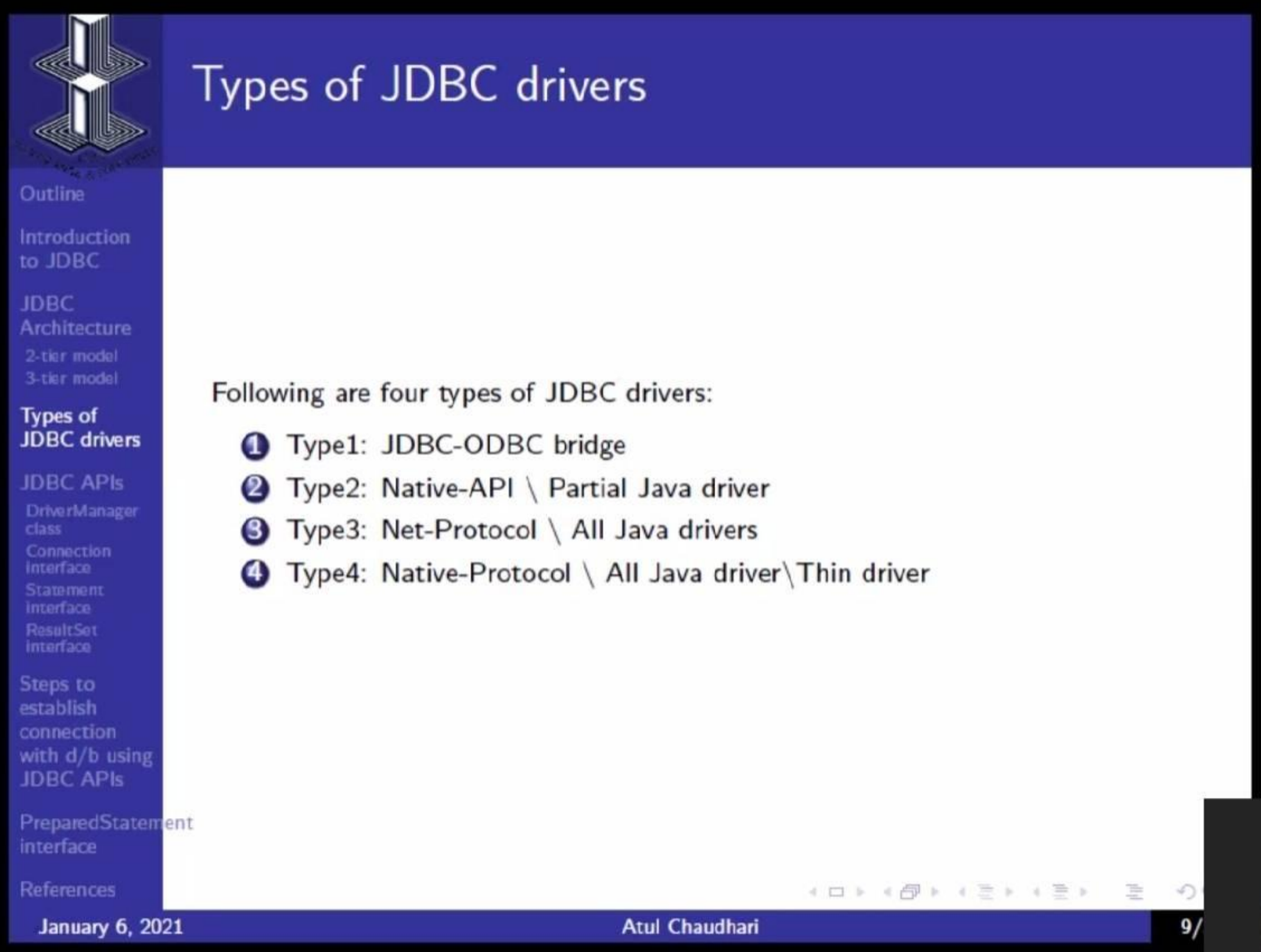


Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

3-tier model for d/b access

- In the 3-tier model, commands are sent to a "middle tier" of services, which then sends the commands to the d/b
- The d/b processes the commands and sends the results back to the middle tier, which then sends them to the user.
- It can connect to different d/bs without changing code
- In this model business logic is clearly separated from d/b
- There are three components of 3-tier model
  1. Client-tier
  2. Middle-tier
  3. Data Source Layer





Type1: JDBC-ODBC bridge

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- JDBC-ODBC Bridge is a Type1 JDBC driver.
- Type 1 JDBC driver converts JDBC operation into C language ODBC API, ODBC calls are then pass to appropriate ODBC driver.

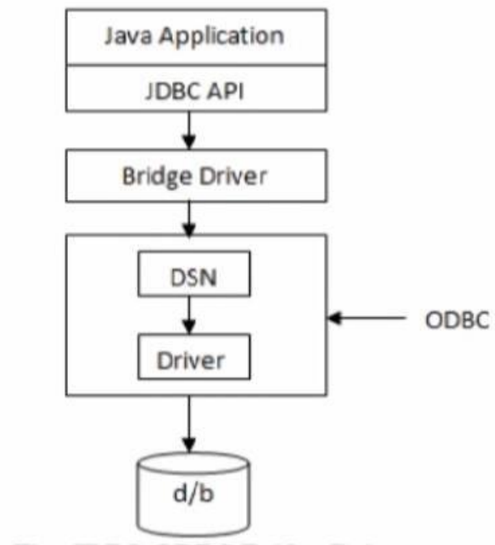


Fig: JDBC-ODBC Bridge Driver

Type2: Native-API \ Partial Java driver

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- The Native API driver uses the client-side libraries of the database.
- The driver converts JDBC method calls into native C/C++ API calls, which are unique to the database.
- These drivers are typically provided by the database vendors
- The Native driver needs to be installed on the each client machine.
- The Vendor client library needs to be installed on client machine

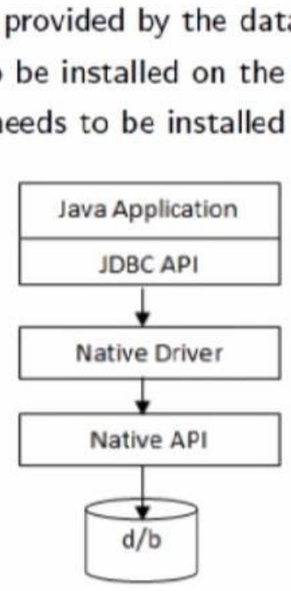


Fig: Native API / Partly Java Driver

Type3: Net-Protocol \ All Java drivers

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- These drivers can only be implemented with three tier architecture.
- The Type3 driver uses middleware (application server) that converts JDBC calls directly or indirectly into the vendor-specific database protocol.
- It is fully written in java.

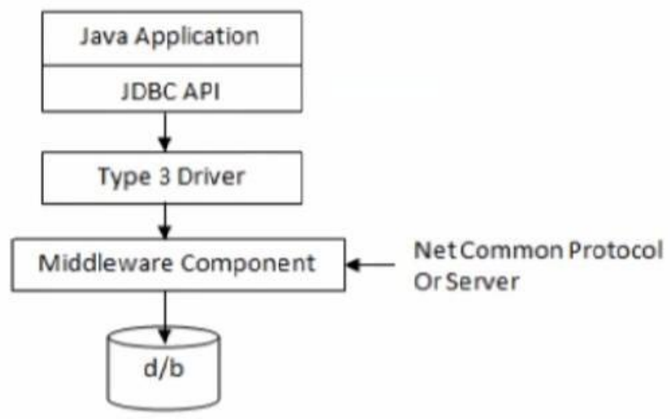


Fig: Net Protocol / All Java Drivers

Type4: Native-Protocol \ All Java driver\Thin driver

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- The thin driver converts JDBC calls directly into the vendor-specific database protocol.
- It is fully written in Java language.

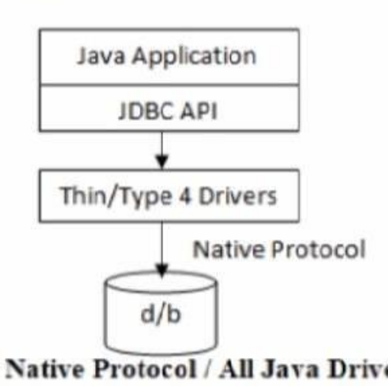
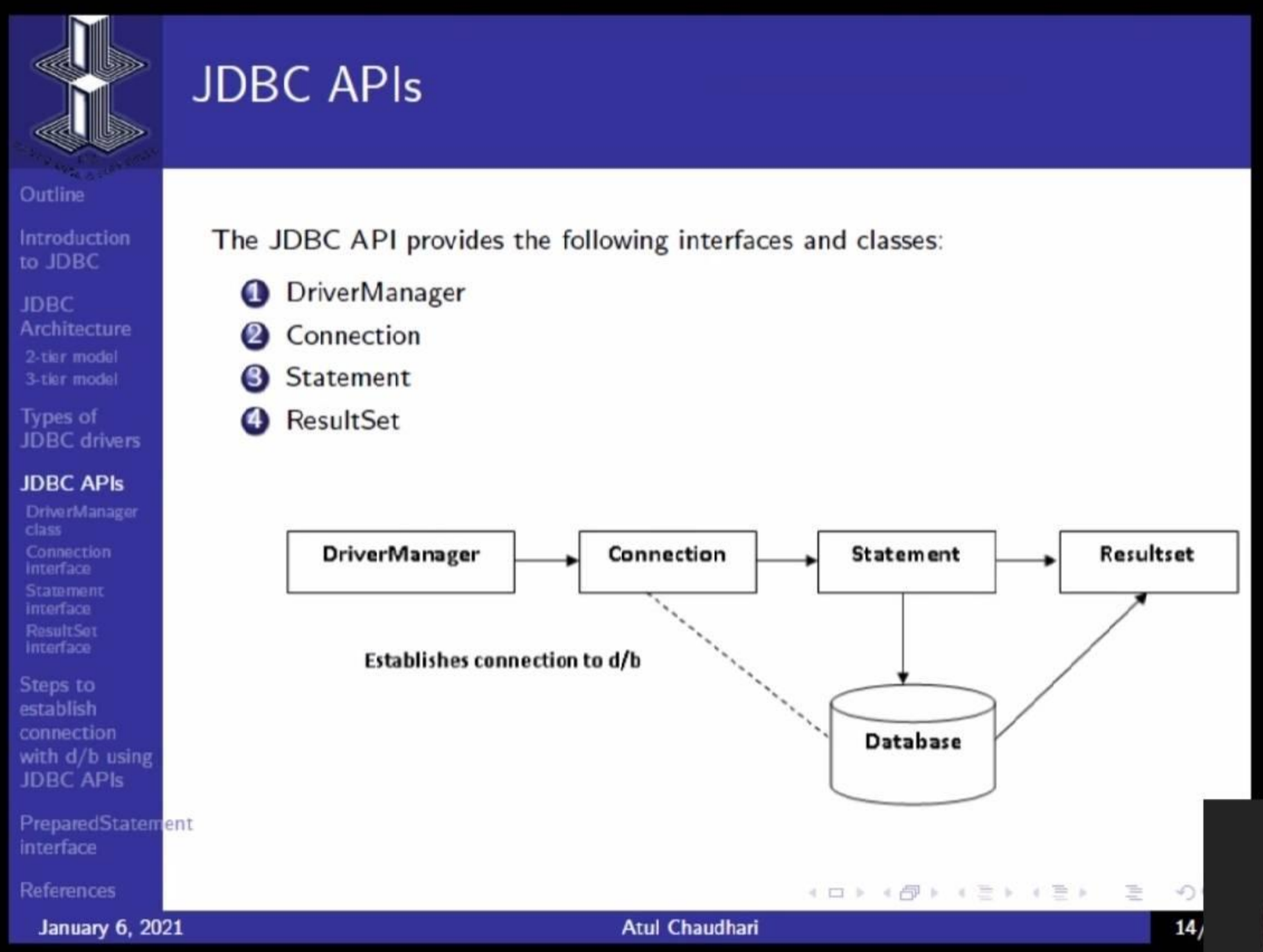


Fig: Native Protocol / All Java Drivers

12:10:57

HD 4G 86%



AC

AP

Atul Chaudhari

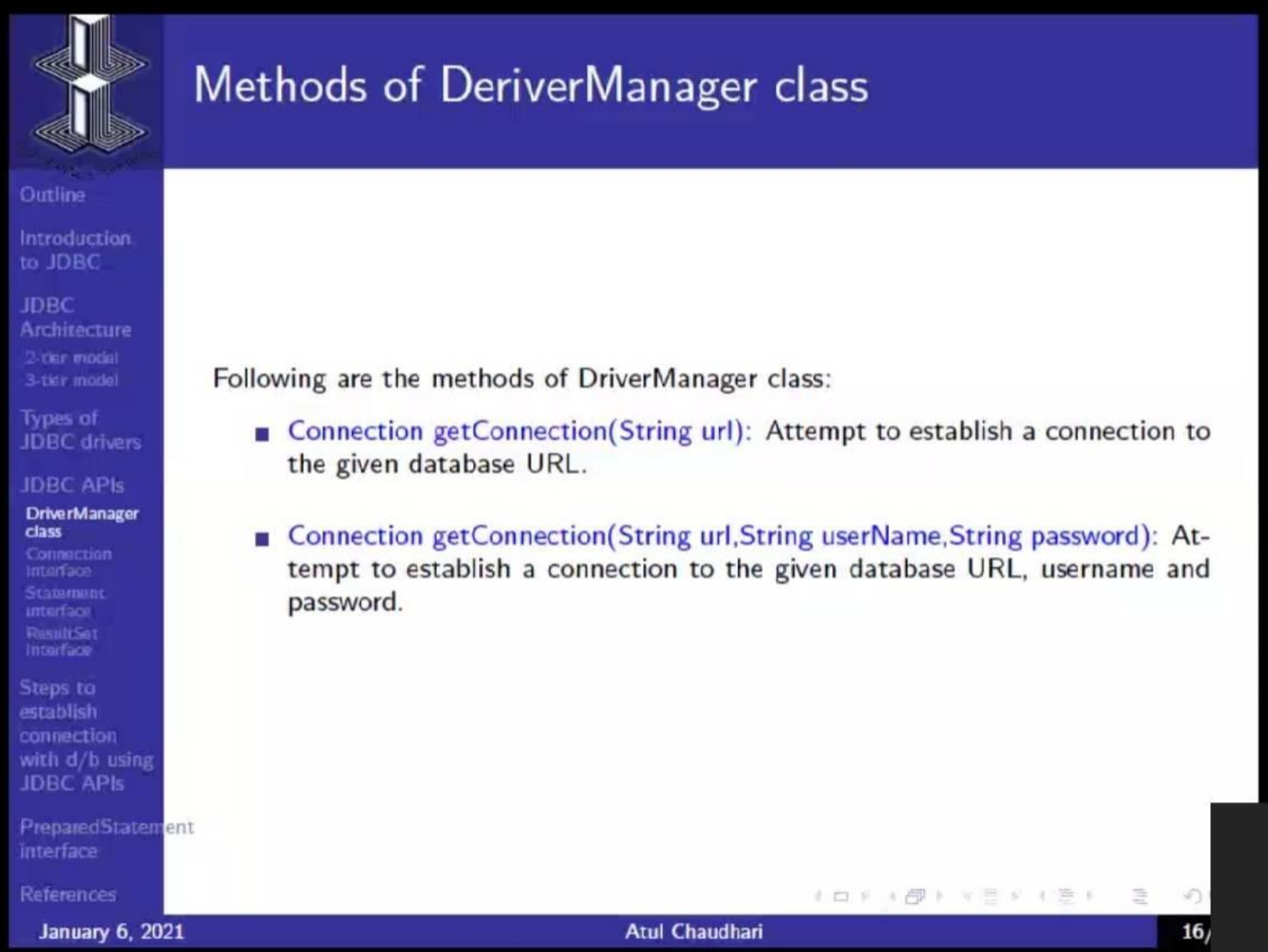
Type here to search

Scanned by TapScanner

DriverManager class

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

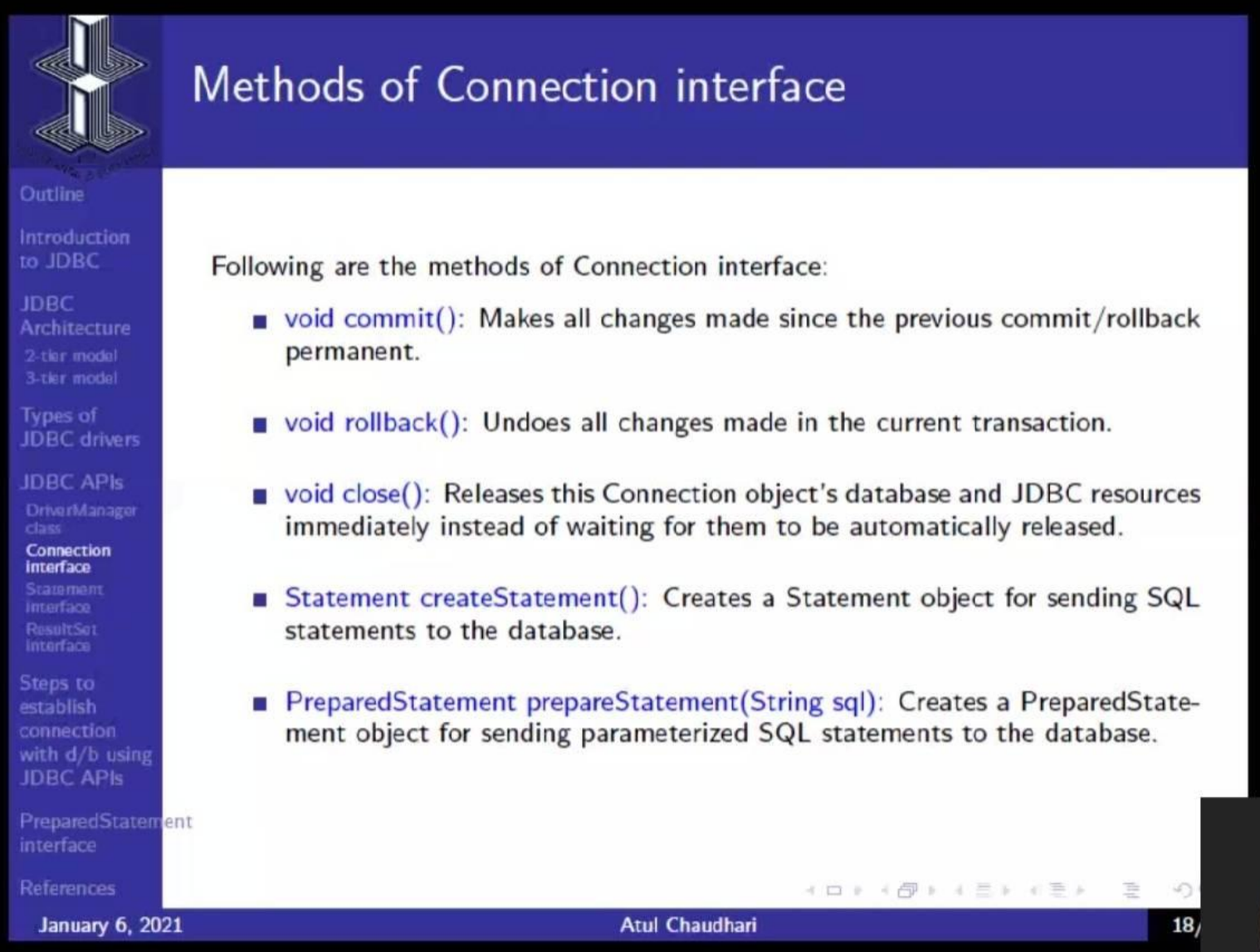
- DriverManager is a class in the Java Software Development Kit (J2SDK).
- DriverManager manages the set of JDBC drivers that are available for an application to use.
- Applications can use multiple JDBC drivers concurrently if necessary.
- By passing a URL for a specific JDBC driver to the DriverManager, the application informs the DriverManager about which type of JDBC connection should be returned to the application.
- DriverManager must be made aware of the available JDBC drivers so it can hand out connections.
- By making a call to the Class.forName(String classname) method, it loads a class into the running JVM based on its string name that is passed into the method.
- DriverManager object creates Connection object.



Connection interface

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- Connection is an interface.
- It is a session between an application and a database.
- Connection interface helps by providing the different methods for transactions and error handling.
- Connection object creates Statement object.



Statement interface and its Methods

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- Statement is an interface.
- It is used for executing a static SQL statement on database.

Following are the methods of Statement interface:

- ResultSet executeQuery(String qry): It is used to execute given SQL statement, which returns single ResultSet object.

  Generally it is used to execute SELECT query on database

- int executeUpdate(String qry): It is used to execute DML and DDL statements on database

  In case of DML (INSERT, UPDATE, DELETE) statements it returns number of rows affected and

  In case of DDL (CREATE) statement it returns 0

- boolean execute(String qry)): Executes the given SQL statement, which may return multiple results.

ResultSet interface

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

- ResultSet is an interface.
- Resultset object contains result of SQL queries.
- Resultset object maintains cursor positioning to its current row of data.
- Initially the cursor is positioning before first row.
- ResultSet object provides corresponding getType() method for each data type to retrieve column value.
- Each method can be used with column name or by its ordinal position.
- Syntax:

  type rs.getType(String columnName)

  type rs.getType(int columnPosition)

  Here,
  type: varies with SQL datatype.
  rs: is object of ResultSet.
  columnName: name of column in a table.
  columnPosition: position of column in a table.

Methods of ResultSet

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

Following are methods of resultset object to retrieve column value:

SQL Datatype	JDBC/Java Datatype	getType( ) Method
VARCHAR	java.lang.String	getString( )
CHAR	java.lang.String	getString( )
BIT	boolean	getBoolean( )
NUMERIC	java.math.BigDecimal	getBigDecimal( )
TINYINT	byte	getByte( )
SMALLINT	short	getShort( )
INTEGER	int	getInt( )
BIGINT	long	getLong( )
REAL	float	getFloat( )
FLOAT	float	getFloat( )
DOUBLE	double	getDouble( )
DATE	java.sql.Date	getDate( )
TIME	java.sql.Time	getTime( )
CLOB	java.sql.Clob	
BLOB	java.sql.Blob	

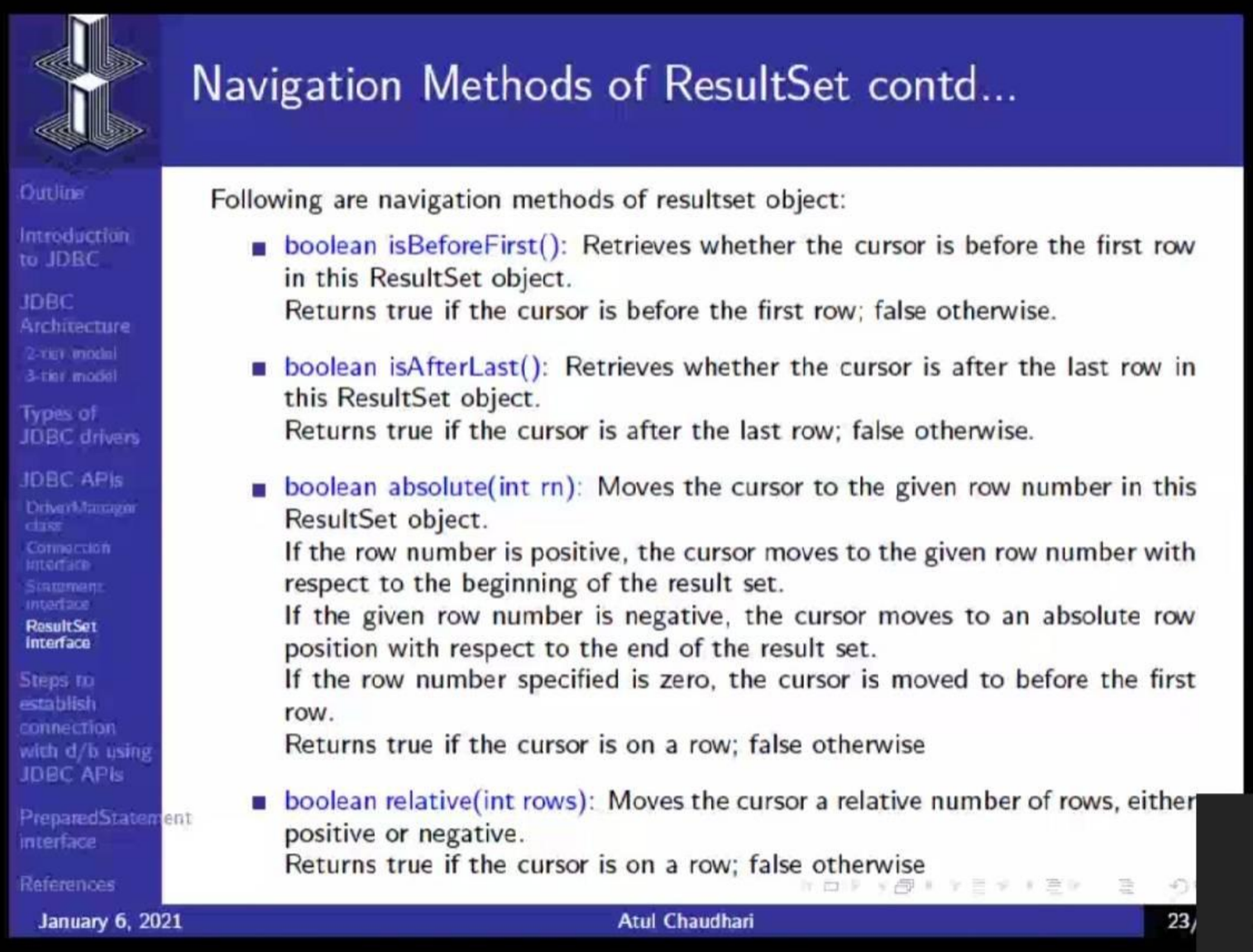
Bluetooth

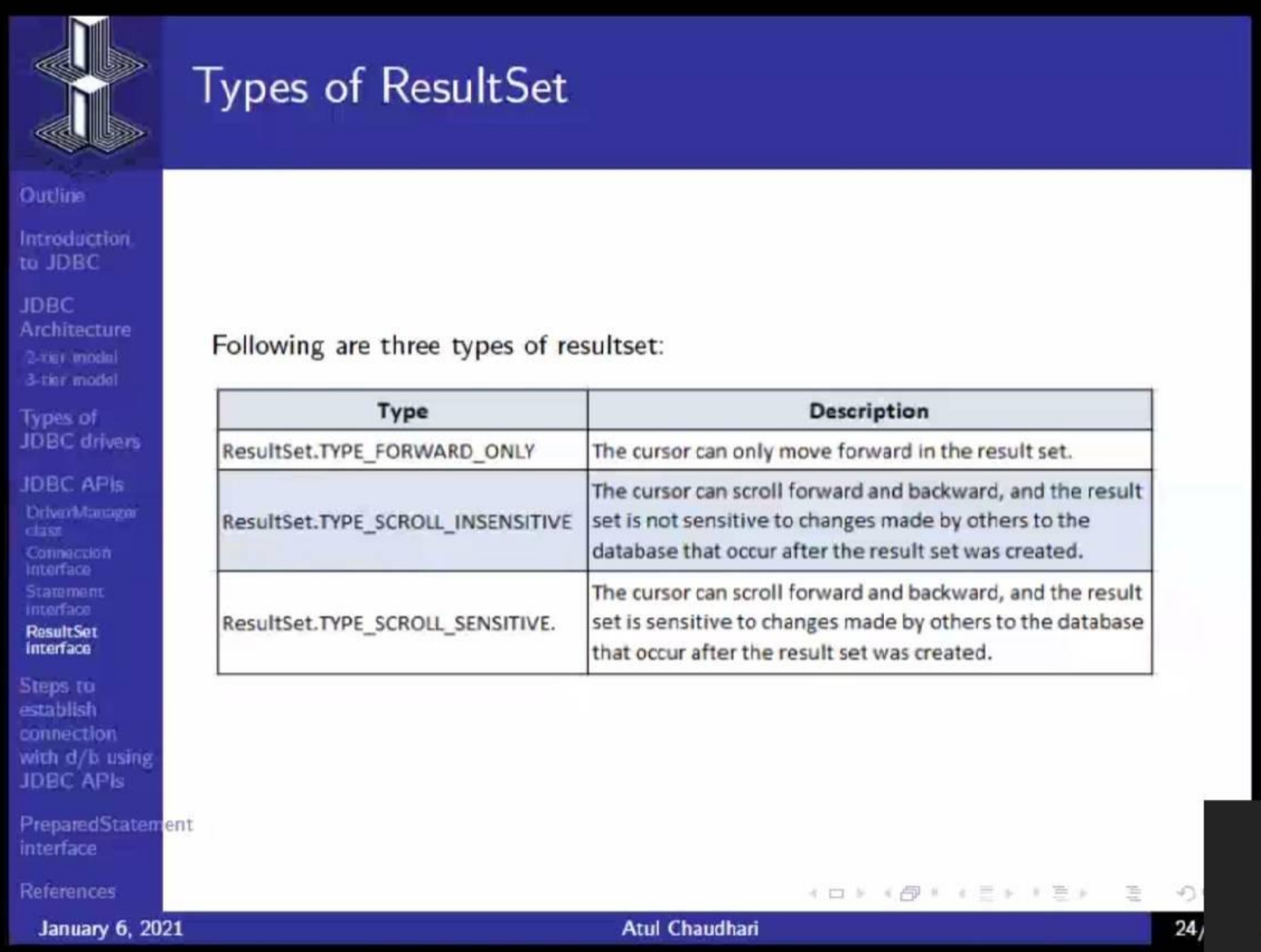
Navigation Methods of ResultSet

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

Following are navigation methods of resultset object:

- boolean first(): Moves cursor to the first record in ResultSet. If first record is available then it returns true otherwise false.
- boolean last(): Moves cursor to the last record in ResultSet. If last record is available then it returns true otherwise false.
- boolean next(): Moves cursor to the next record in ResultSet. If next record is available then it returns true otherwise false.
- boolean previous(): Moves cursor to the previous record in ResultSet. If previous record is available then it returns true otherwise false.
- void beforeFirst(): Moves the cursor to the front of this ResultSet object, just before the first row.
- void afterLast(): Moves the cursor to the end of this ResultSet object, just after the last row.
- boolean isFirst(): Retrieves whether the cursor is on the first row of this ResultSet object.
  Returns true if the cursor is on the first row; false otherwise.
- boolean isLast(): Retrieves whether the cursor is on the last row of this ResultSet object.





Concurrency of ResultSet

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

Following are possible concurrency of resultset:

Concurrency	Description
ResultSet.CONCUR_READ_ONLY	Creates a read-only result set. This is the default
ResultSet.CONCUR_UPDATABLE	Creates an updateable result set.

Steps to establish connection with d/b using JDBC APIs

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatement interface
References

1. import java.sql package
2. Load and register the JDBC driver
3. Define Connection URL
4. Establish a connection
5. Create a statement object to perform a query
6. Execute the Queries
7. Process the result
8. Close the Connection

PreparedStatement interface

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatem interface
References

- The PreparedStatement interface is a subinterface of Statement.
- It is used to execute parameterized query.

  Example: "insert into stud values(?, ?, ?)";

- It is used to execute pre-compiled SQL statement.
- It provides different types of setter methods to set the input parameters for the query.
- The prepareStatement() method of Connection interface is used to return the object of PreparedStatement.

Syntax:

PreparedStatement prepareStatement(String query)

Methods of PreparedStatement interface

Outline
Introduction to JDBC
JDBC Architecture
2-tier model
3-tier model
Types of JDBC drivers
JDBC APIs
DriverManager class
Connection interface
Statement interface
ResultSet interface
Steps to establish connection with d/b using JDBC APIs
PreparedStatem interface
References

Following are some important methods of PreparedStatement interface:

Method	Description
void setInt(int paramIndex, int value)	sets the integer value to the given parameter index.
void setLong(int paramIndex, long value)	sets the long value to the given parameter index.
void setFloat(int paramIndex, float value)	sets the float value to the given parameter index.
void setDouble(int paramIndex, double value)	sets the double value to the given parameter index.
void setString(int paramIndex, String value)	sets the String value to the given parameter index.
int executeUpdate()	executes the query. It is used for create, drop, insert, update, delete
ResultSet executeQuery()	executes the select query. It returns an instance of ResultSet.