DB Programming

Database Systems

Presented by Rubi Boim

Agenda

- Project Details
- Basic Oracle Usage
- Little More Complex Oracle stuff..
- JDBC
- Coding Tips

Agenda

- Project Details
- Basic Oracle Usage
- Little More Complex Oracle stuff..
- JDBC
- Coding Tips

Oracle Data Types

There are 3 main groups of types:

- Character
- Numeric
- Date

- http://download.oracle.com/docs/cd/B28359_01/server.111/b28318/datatype.htm

Oracle Data Types – Character

- Character set is established when you create the database (UTF-8,ANSI..). So ignore.. ☺
- Char: Fixed length! Short value is padded
- Varchar2: variable-length
- Both types needs to define max length (4000 max. for more use BLOB)

Oracle Data Types – Numeric

- Implemented by the “Number” data type

- A Number has two properties:
  - precision: Total number of digits
  - scale: Number of digits after the point
  (up to 38 digits)

- For floating point (if you need..):
  BINARY_FLOAT, BINARY_DOUBLE

Oracle Data Types – Numeric

- Number Example

Input Data	Specified As	Stored As
7,456,123.89	`NUMBER`	7456123.89
7,456,123.89	`NUMBER(*,1)`	7456123.9
7,456,123.89	`NUMBER(9)`	7456124
7,456,123.89	`NUMBER(9,2)`	7456123.89
7,456,123.89	`NUMBER(9,1)`	7456123.9
7,456,123.89	`NUMBER(6)`	(not accepted, exceeds precision)
7,456,123.89	`NUMBER(7,-2)`	7456100

Oracle Data Types – Date

- Implemented by the “Date” data type
- Stores “dates” and “times”
- Default format is DD-MON-YY
- Use the TO_DATE function for any other format

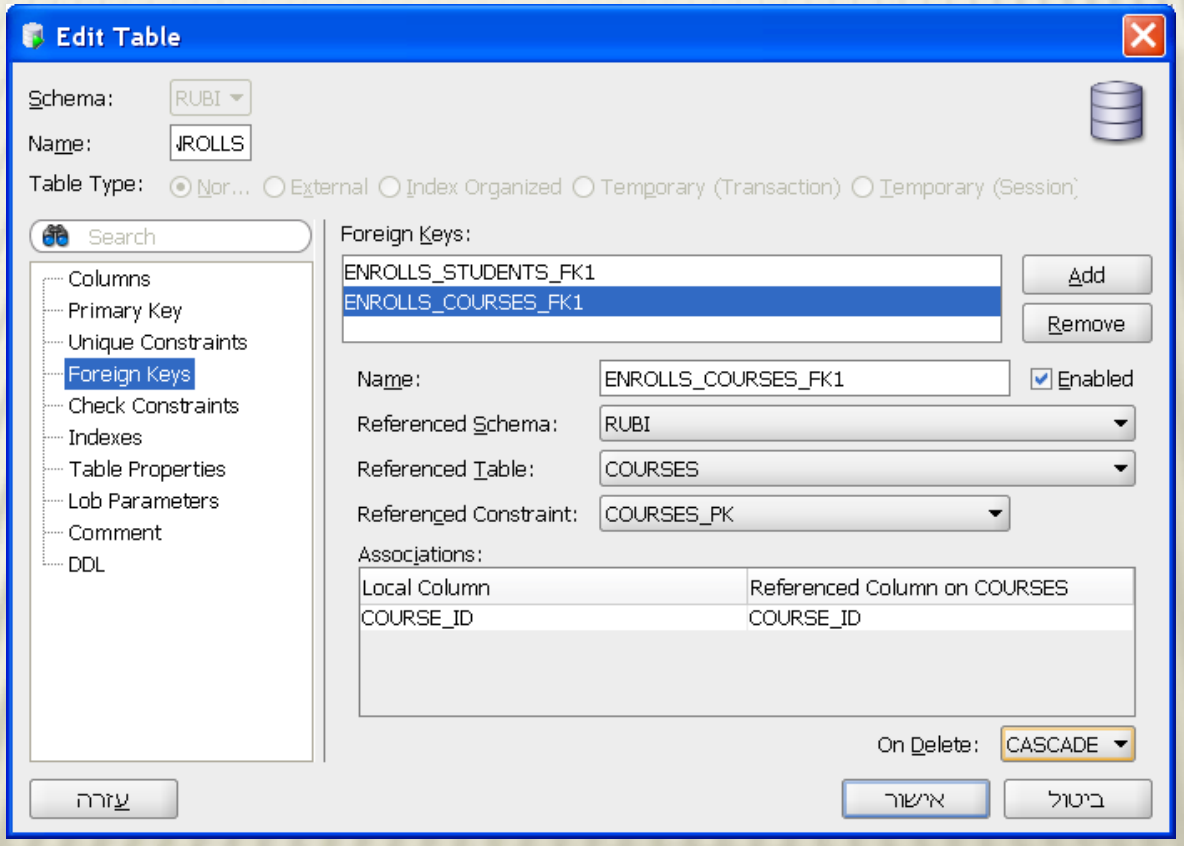
```
TO_DATE('13-AUG-66 12:56 A.M.','DD-MON-YY HH:MI A.M.')
```

Define Foreign keys

- Don’t forget to define the primary key on the other table..

- What happens when you delete the “key record” from the “primary table”?
  - Restrict
  - Cascade
  - Set null

Define Foreign keys



Basic oracle usage - Demo

- Demo..
  - create table (data types)
  - define primary key
  - define foreign keys (insert / delete data)

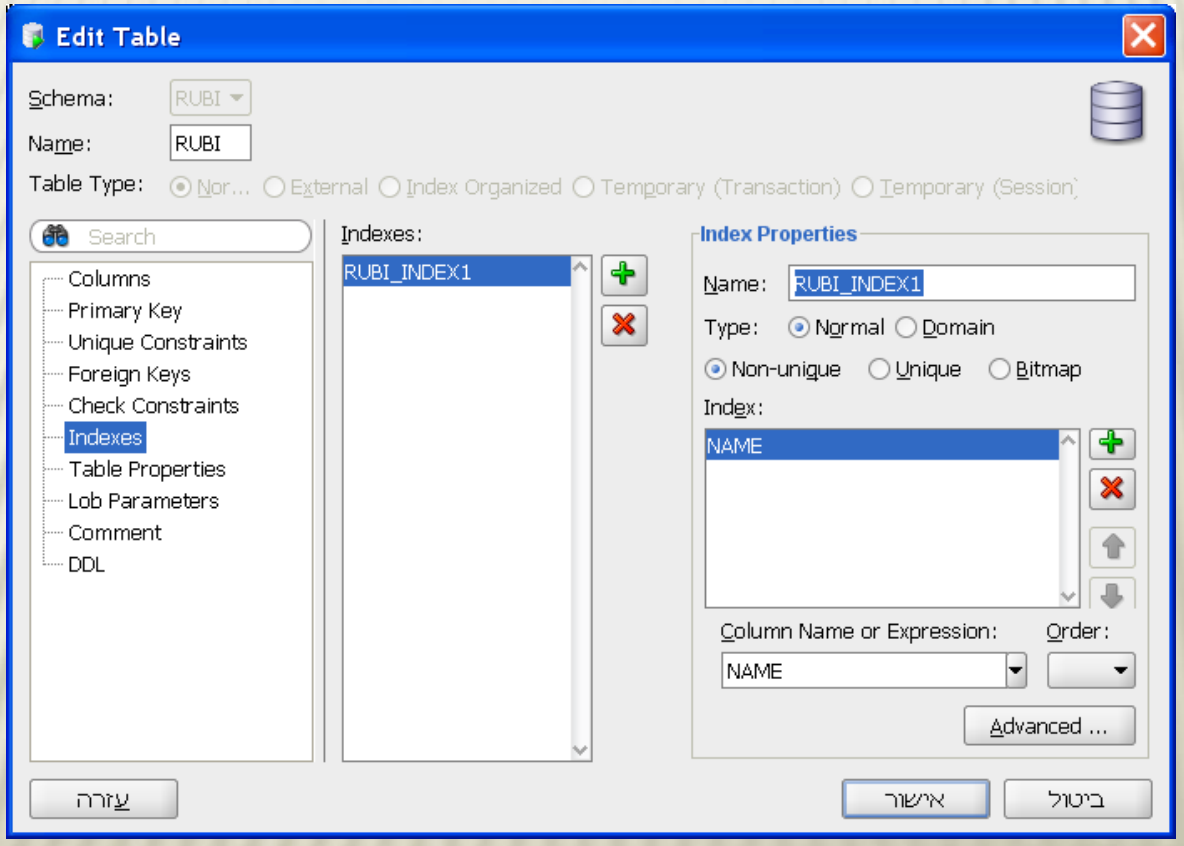
Agenda

- Project Details
- Basic Oracle Usage
- Little More Complex Oracle stuff..
- JDBC
- Coding Tips

Index

- Index improves the speed of operations on a table
- Can be created using one or more fields
- You will later learn more..
- But don’t forget, its important

Index - HowTo


Edit Table
Schema:
RUBI
Name:
RUBI
Table Type:
Nor...
External
Index Organized
Temporary (Transaction)
Temporary (Session)
Search
Columns
Primary Key
Unique Constraints
Foreign Keys
Check Constraints
Indexes
Table Properties
Lob Parameters
Comment
DDL
Indexes:
RUBI_INDEX1
Index Properties
Name:
RUBI_INDEX1
Type:
Normal
Domain
Non-unique
Unique
Bitmap
Index:
NAME
Column Name or Expression:
Order:
NAME
Advanced ...
עזרה
אישור
ביטול

“AutoNumber”

ID	NAME
1	Rubi
2	Tova
3	Itay
4	Dvir
…	…

- How do you know how to assign an ID??

“AutoNumber” – Algorithm?

Lock table

new_id = 1 + select max id from table

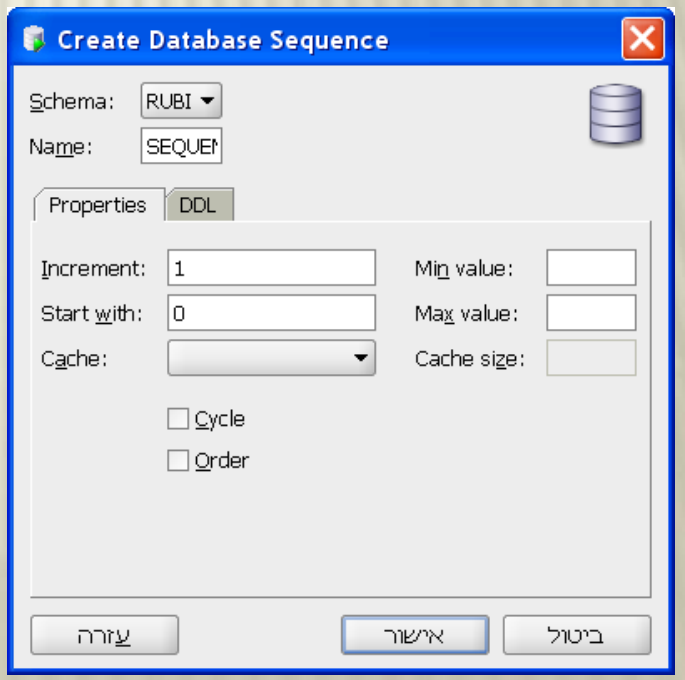
insert into table values(new_id, ”Rubi”);

Unlock table

Sequence

- Sequence - an object from which multiple users may generate unique integers

- NEXTVAL() - increments the sequence and returns the new value.



Sequence – Insert example

- If we defined the sequence as TEST_SEQ

```
INSERT INTO test
   values(TEST_SEQ.NEXTVAL, 'rubi')
```

- Usually, sequence is defined as table_name +”_SEQ”

Sequence – Can we do better?

- Why are we complicating things??
- Do all DBMS support sequences?
- If we change a sequence name, we need to update all our queries
- Can we separate it from the query?

```
INSERT INTO test(name) values('rubi')
```

Triggers

- A database trigger is procedural code that is automatically executed in response to certain events on a particular table

- Events:

BEFORE INSERT	AFTER INSERT
BEFORE UPDATE	AFTER UPDATE
BEFORE DELETE	AFTER DELETE

Triggers – Statement Level

- Occurs only once per Insert/Update/Delete

```
CREATE OR REPLACE TRIGGER  <trigger_name>
<BEFORE | AFTER> <ACTION>  ON  <table_name>
BEGIN
    <trigger_code>
END;
```

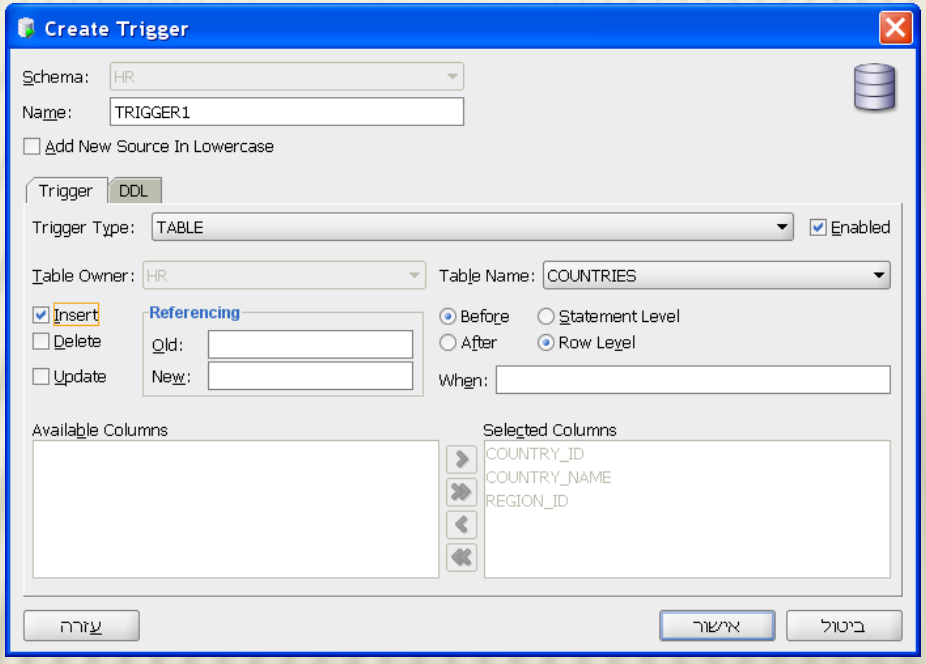
Triggers – Row Level

- Occurs for each row

```
CREATE OR REPLACE TRIGGER  <trigger_name>
<BEFORE | AFTER> <ACTION>  ON  <table_name>
FOR EACH ROW
    BEGIN
        <trigger_code>
    END;
```

Triggers – Row Level – Example

- You can not “just use the GUI” - you need to “code” the trigger”



- After you press “ok” you can edit the code

Triggers – Row Level – Example

- Use “NEW” to refer to the row
- dual – simply a scratch-pad

select max(12,54,2,75,142) from dual

```
CREATE bi_test
BEFORE INSERT  ON  test
FOR EACH ROW
    BEGIN
        SELECT      TEST_SEQ.NEXTVAL
        INTO        :NEW.id
        FROM        dual;
    END;
```

“Complex” Oracle Stuff

- Demo..
  - Create index
  - Create “Autonumber”:
    - Create Sequence
    - Create Trigger

Limit the Results

- What if your query returns 1,000,000 results?
- How to return the TOP *n* results
- How to return the results from *n* to *m*

Oracle's Rownum

- Works only on Oracle..
  (mysql has "Limit", sql-server has "Top")
- ROWNUM is a pseudocolumn (not "real")
- Each row is assigned with a number, starting with 1
- We can select just the ones we want..

Oracle's Rownum – NOT THAT SIMPLE!

- Its assigned BEFORE sorting or aggregation
- ROWNUM value is incremented only after it is assigned
- Read the previous lines 5 more times!

Oracle's Rownum – Example 1

```
SELECT      *
FROM        students
WHERE       ROWNUM > 1
```

- What NOT to do… ☺

Oracle's Rownum – Example 2

```
SELECT      *
FROM        students
WHERE       ROWNUM < 10
ORDER BY    students.name
```

- What NOT to do… ☺

Oracle's Rownum – Example 3

```
SELECT * FROM
(       SELECT          *
        FROM            students
        ORDER BY        students.name           )
WHERE   ROWNUM < 10
```

- This will work…

Oracle's Rownum – Example 4

```
SELECT * FROM
(       SELECT          *
        FROM            students
        ORDER BY        students.name           )
WHERE   ROWNUM >= 10    AND
        ROWNUM < 20
```

- What NOT to do… ☺

Oracle's Rownum – Example 5

```
SELECT * FROM
(       SELECT a.*, ROWNUM rnum FROM
        (
                SELECT      *
                FROM                students
                ORDER BY  students.name
        ) a
)
WHERE       rnum >= 10  AND
                rnum < 20
```

- Will work but we can do better (y)… ☺

Oracle's Rownum – Example 6

```
SELECT * FROM
(       SELECT a.*, ROWNUM rnum FROM
        (
                SELECT      *
                FROM                students
                ORDER BY  students.name
        ) a
        WHERE       ROWNUM < 20
)
WHERE       rnum >= 10
```

- That's the way… ☺

Oracle's Rownum – Final slide ☺

- There is a big difference between > and <
- If you are using "example 6", be sure the order by is unique (y?)
- btw, in MySQL its simply:

```
select * from students order by name limit 10,20
```

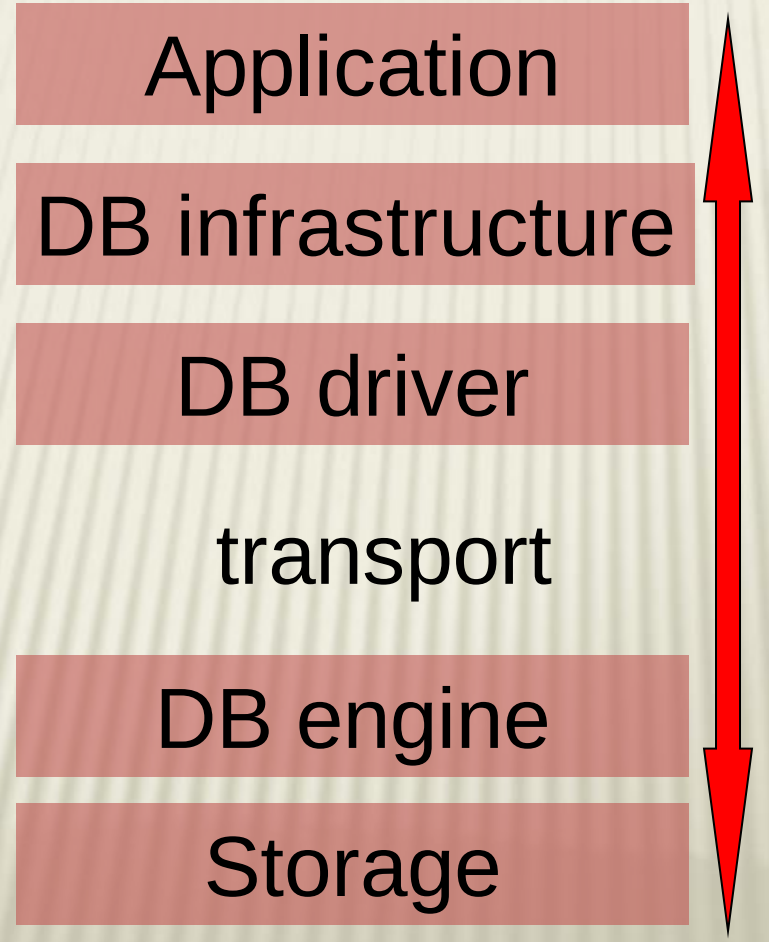
Little More Complex Oracle Stuff - Demo

- Demo..
  - create Sequence
  - create Trigger (for autonumber)
  - limiting the results

Agenda

- Project Details
- Basic Oracle Usage
- Little More Complex Oracle stuff..
- JDBC
- Coding Tips

During the last episode...



Concepts vs APIs

Concepts		APIs/Language
Connection		
Connection pooling		
Error Handling		ODBC
Fetching results	X	JDBC
Rowset		OCI/OCCI
Prepared statements		ADO.NET
Batch processing		

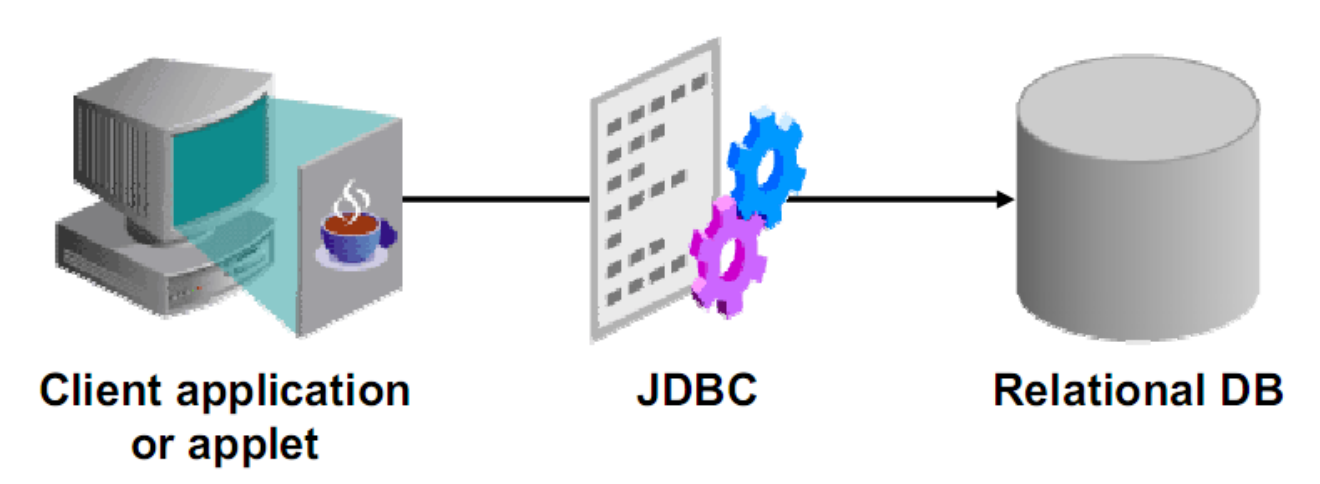
ODBC – Open Database Connectivity API

- Pros:
  - Cross platform and cross databases
  - Easy to use
- Cons:
  - Too low level

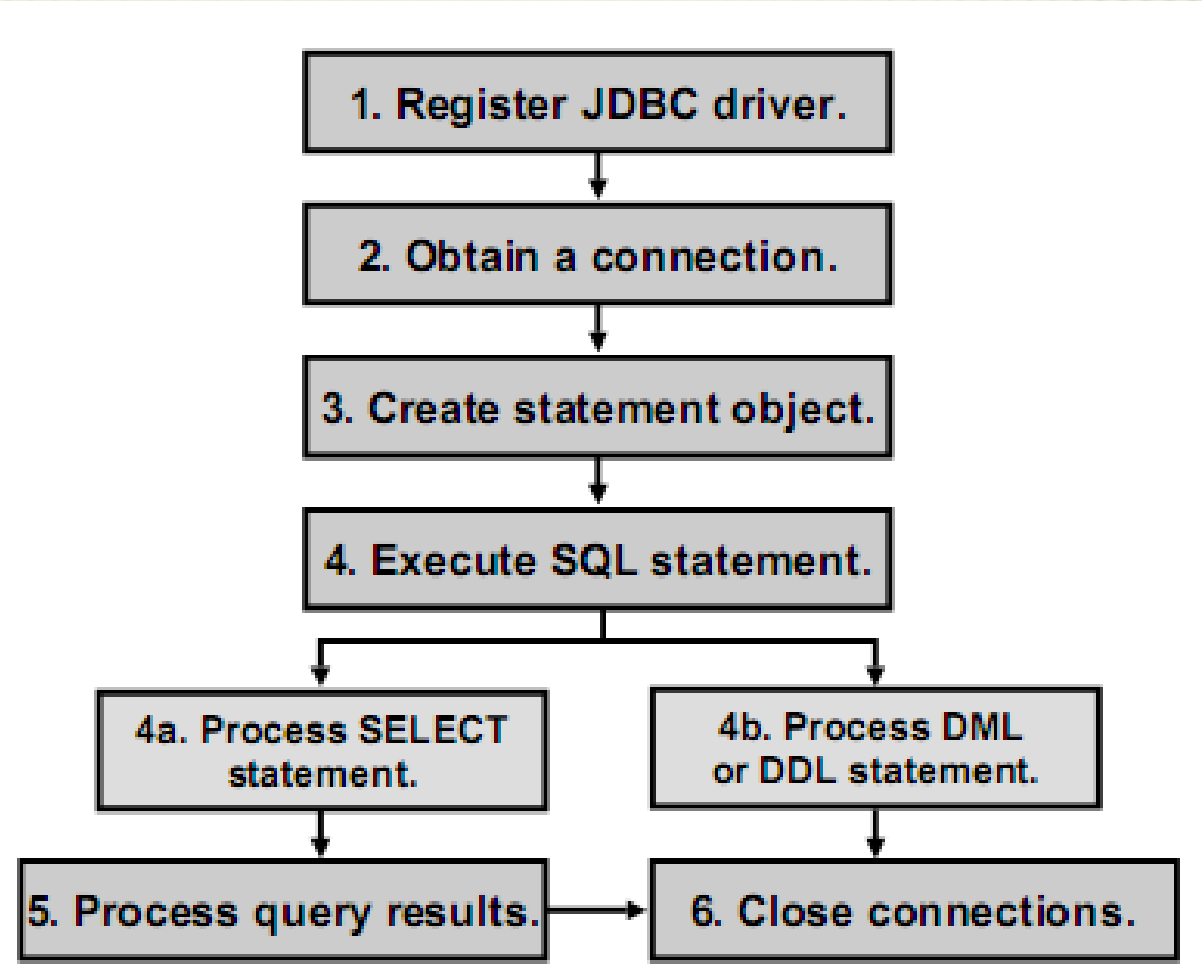
- We wont use it.. But its very very common

JDBC

- JDBC is a standard interface for connecting to relational databases from Java

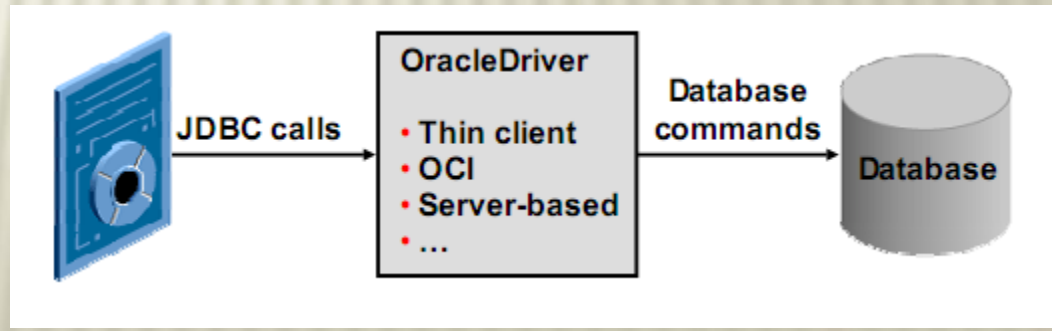


How to execute SQL using JDBC



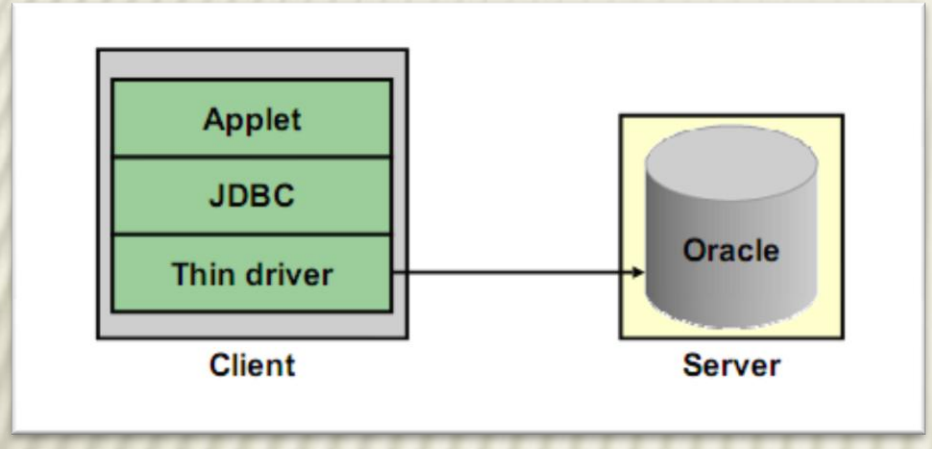
JDBC Oracle Driver

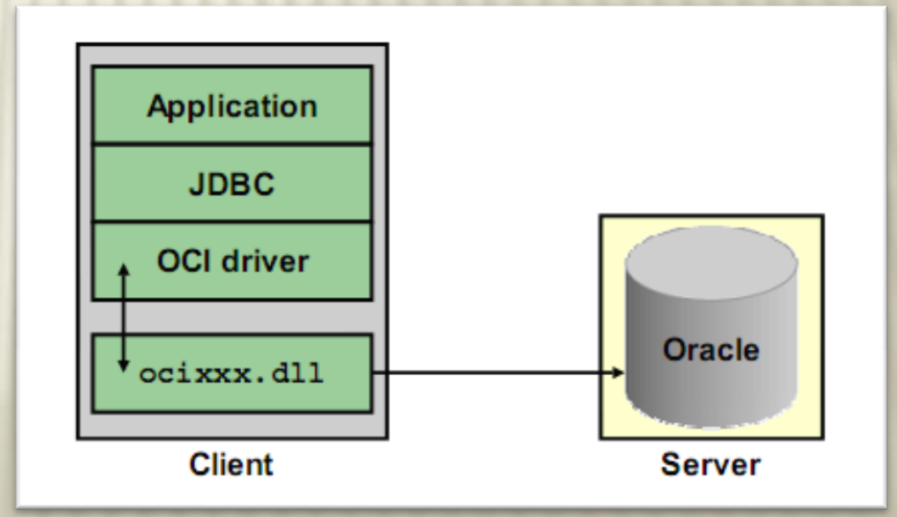
- Thin Client driver
  written in java
- OCI Driver
  written in java & c. must be installed
- ODBC Bridge
  (too general..)



JDBC Oracle Driver

- Thin vs OCI

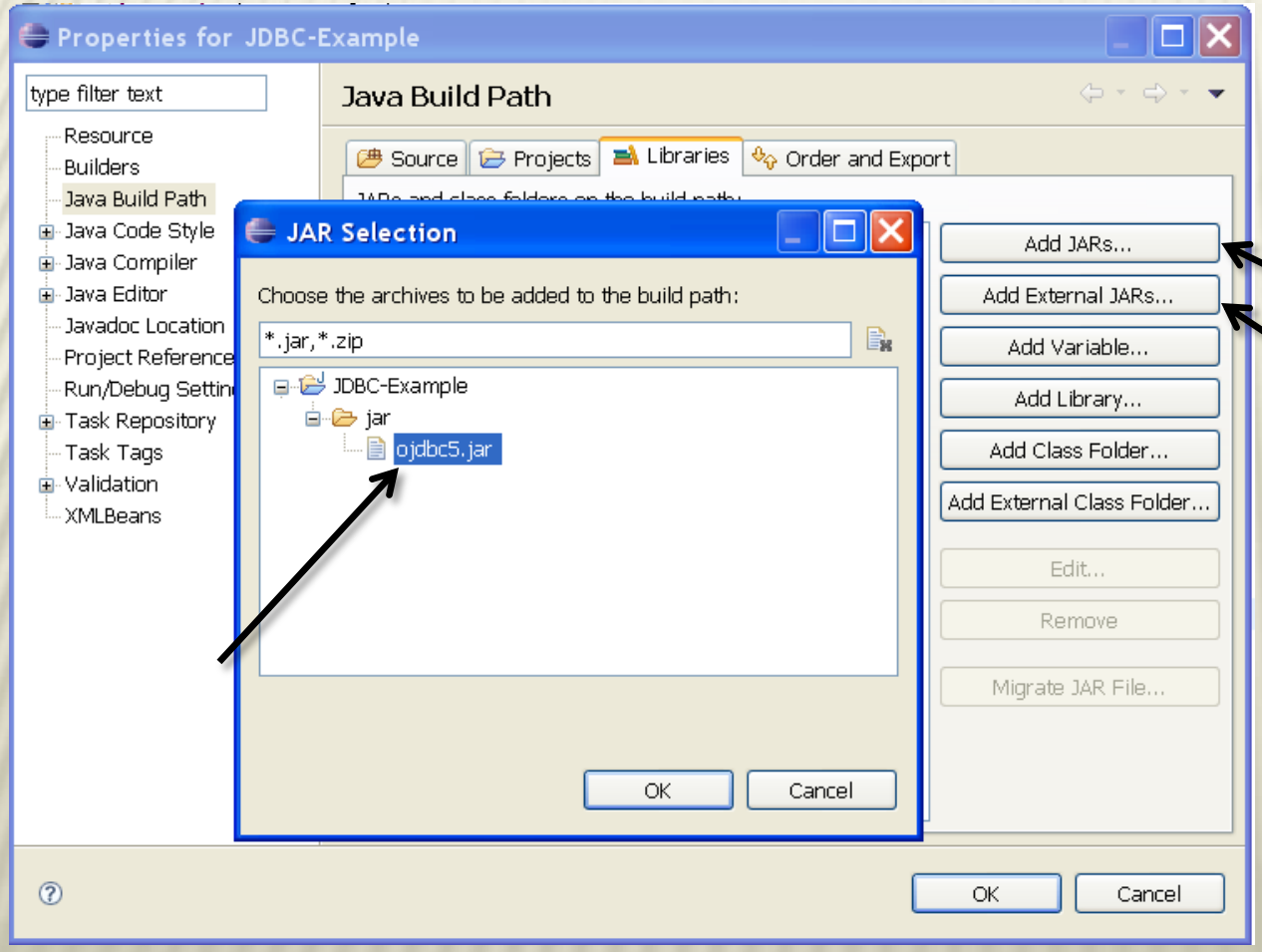




Preparing the Environment 1

- Download Oracle's JDBC driver: http://www.oracle.com/technology/software/tech/java/sqlj_jdbc/htdocs/jdbc_112010.html
- Can also be found at the course page
- Setup Eclipse:
  - add the jar "ojdbc6.jar" to the project

Preparing the Environment 2



If you copy the jar file to the project directory, press “add JAR”. Otherwise, “Add external JAR”

Preparing the Environment 3

- import java.sql.* (JDBC API)
- Register the driver in the code:
  Class.forName("oracle.jdbc.OracleDriver");

Opening a Connection

- Connection class - java.sql.Connection
- use the DriverManager with JDBC URL

```
conn = DriverManager.getConnection(
   "jdbc:oracle:thin:@localhost:1521:XE",
   “username",
   “password");
```

Opening a Connection

- Demo..

Creating a Statement

- Created from the connection object

```
Statement stmt = conn.createStatement();
```

Using a Statement

Three different methods:

- executeQuery(String) for SELECT statements returns ResultSet
- executeUpdate(String) for DML/DDL returns int
- execute(String) for any SQL statement returns boolean

executeQuery & ResultSet

ResultSet:

- Maintain a curser to its current row
- Provides methods for retrieving values:
  getInt(), getDate(), getString()..

- Fields can be identify by name or order:
  getXXX("Name")
  getXXX(2)

executeQuery & ResultSet

- Initially the cursor is positioned before the first row

```
stmt  =     conn.createStatement();
rs    =     stmt.executeQuery(
                    "SELECT * FROM employees");
while (rs.next() == true)
   System.out.println(rs.getString(“field”));
```

- Demo..

executeUpdate

- Again, via the statement
- Execute DDL or DML
- Returns Int for DML, 0 for DDL

executeUpdate

```
stmt=conn.createStatement();
result=stmt.executeUpdate(
                "DELETE FROM demo");
```

- Demo..

execute

- Executes any command for the DB
- Returns boolean (success/failure)
- Not sure you'll need it..

Closing Connections

- Important! So don’t forget..

- ResultSet.close()
- Statement.close()
- Connection.close()

Transactions

- By default, connection are autocommit
- Can be disabled by:
  conn.setAutoCommit(false)
- Commit a transaction: conn.commit()
- Rollback a transaction: conn.rollback()

Transactions – When to use?

- In general, in any logic operation that involves more than one call: insert/update/remove into several tables
- Inconsistent data is unacceptable!
- Don’t forget to use!

PreparedStatement

- Prevents reparsing of SQL statements
- Used for statements executed more than once
- Saves time
- Nicer code

PreparedStatement - how

- Specify a variable by “?”

```
PreparedStatement pstmt = conn.prepareStatement(
        "INSERT INTO demo(fname, lname) VALUES(?, ?)");
```

- Supply values for the variables:

  pstmt.setXXX(index, value)

- Execute the statement

  pstmt.executeUpdate();

PreparedStatement - example

```
PreparedStatement pstmt = conn.prepareStatement(
        "INSERT INTO demo(fname, lname) VALUES(?, ?)");

pstmt.setString(1, "Rubi");
pstmt.setString(2, "Boim”);
pstmt.executeUpdate();

pstmt.setString(1, “Tova");
pstmt.setString(2, “Milo”);
pstmt.executeUpdate();
```

- Demo..

Batch PreparedStatement

- PreparedStatement can be slow for long calls
- Batch together all the calls!
- I.E. instead of 50,000 calls, do one call with 50,000 parameters
- Improves performance dramatically!

Batch PreparedStatement - how

- Instead of pstmt.executeUpdate()
  do pstmt.addBatch()
- After all statement are added to the batch:
  int[] = pstmt.executeBatch()
- TIP: don't batch too much together
- Demo..

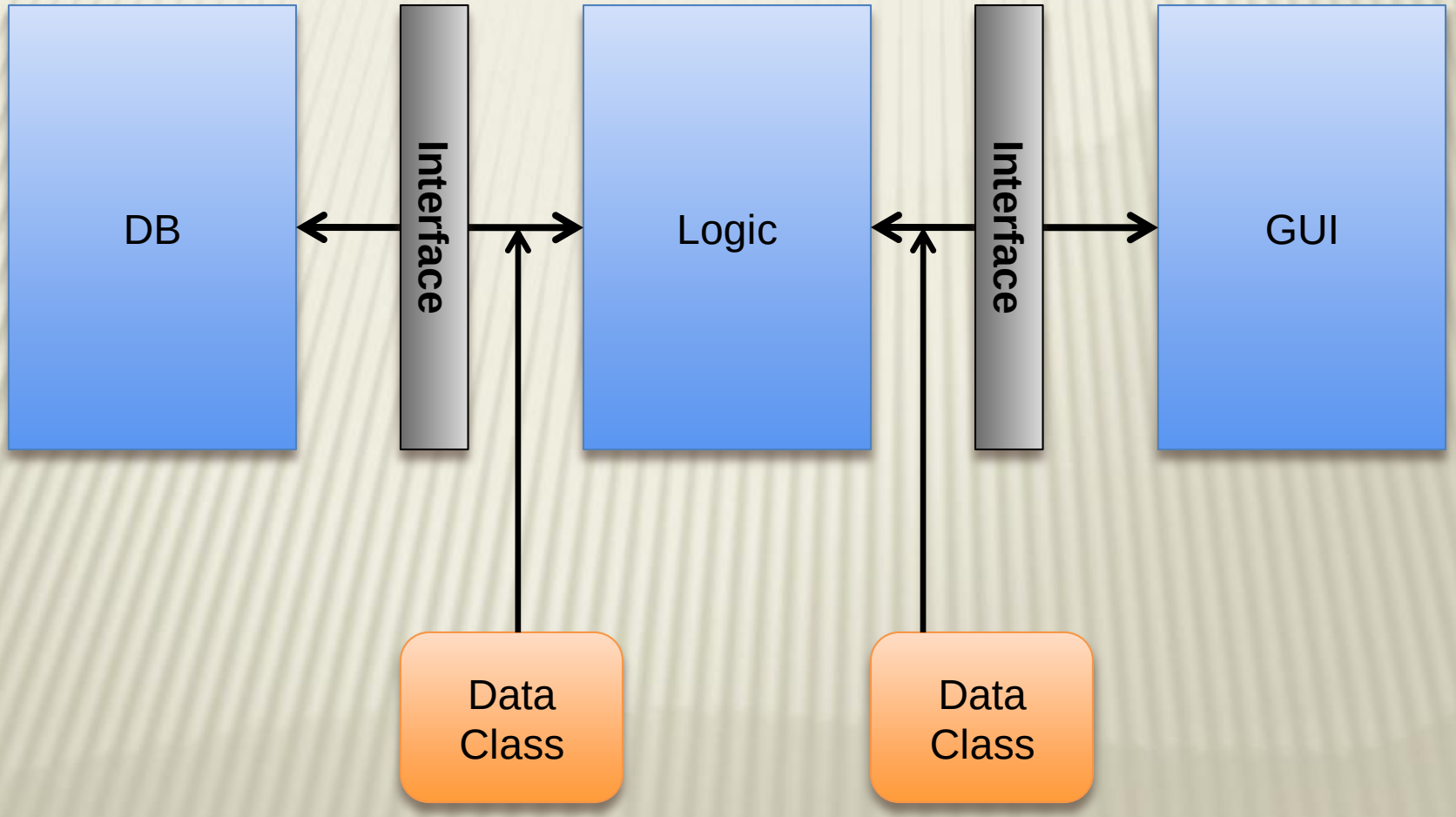
Agenda

- Project Details
- Basic Oracle Usage
- Little More Complex Oracle stuff..
- JDBC
- Coding Tips

Layering

- Separate the GUI!
- Separate the DB!
- Use classes to describe entities
- Use interfaces!

Layering



Reuse & Encapsulation

- Identify main processes
- Abstract implementation
- Reuse..
- NO COPY PASTE CODE

Don’t create too many functions

- Search for movies:
  searchMovieByName()
  searchMovieByDate()
  ..
- It’s the same query! just different “where” → manipulate the “where” in the function:
  SearchMovie(searchOptions?)

- Not so easy on some parameters..
  searchMovieByActors()
  searchMovieByActorsAndDate() any ideas?

Configuration

- Your program will have several (many) variables:
  - server address
  - textfile location
  - number of connections/threads
  - ….
- Do not “hard code” them
- *.ini file, easy GUI, ….

Schema

- Well, you should be expert by now.. ☺
- Primary Key - ALWAYS integer!
- Use indexes to speed up (but not on every field)

Testing

- Obvious not?
- Try installing / running your program on different computers
- Connection drops
- Validate user input (date, number, special chars..)
- Your program should never fall!!

Good questions...

- Managing Database Connections
- Managing Security
- Managing Threads
- Error handling

How to insert with AutoNumber

- Assuming you created a trigger similar to the one showed before..
- Specify the exact fields in the “Insert” (I.E. neglect the “triggered” ones)

ID	NAME
1	Yonni
2	Tova
3	Dvir

INSERT INTO test(name) VALUES(‘Rubi’);

Retrieving the AutoNumber Generated

- When calling “executeUpdate”, you can specify which fields you can “get back”
- After executing, use getGeneratedKeys() to retrieve a resultset with the returned fields

```
stmt.executeUpdate("INSERT INTO demo(fname, lname)
                                VALUES('Rubi','Boim')",
                            new String[]{"ID"});
rs=stmt.getGeneratedKeys();
rs.next();
id=rs.getInt(1);
```

Retrieving the AutoNumber Generated

- Demo.. (I.E. there is an example code ☺)

How to insert Strings

- In an SQL Query, strings are surrounded by ‘
- But what if we want to insert the char ‘?

INSERT INTO test VALUES(‘It’s a test’);

- Simply add another ‘

INSERT INTO test VALUES(‘It’’s a test’);

How to insert Dates

- Read the “to_date” manual..
  http://www.techonthenet.com/oracle/functions/to_date.php

```
stmt.executeUpdate(
   "INSERT INTO demo(fname, lname, mydate)
   VALUES('Rubi',
                'Boim',
                to_date('13/12/2008', 'dd/mm/yyyy'))");
```

Important Tip for Manipulating Data

- What if tomorrow your switch to MySQL??
- Create your own functions for adjusting types (not just dates)

```
String fixDate(String old_date)
{ return “to_date(‘” + old_date + “'), 'dd/mm/yyyy')”}

stmt.executeUpdate(
"INSERT INTO demo(fname, lname, mydate)
    VALUES('Rubi', 'Boim',” + fixDate('13/12/2008‘) + ”)”);
```

Connection Pooling

- Opening a connection is “expensive”
- Multi tasks requires multi connections
- You should open a connection only when you need it (I.E. when a task asks for connection and there is no one available)
- When the task is done, do not close the connection but returns it to the “manager” for future use

Connection Pooling – example

- Example of what might it look..

```
MyConn conn = cManager.poolConn();
conn.getJDBCConn.executeQuery(..);

conn.returnConnection();        OR
cManager.returnConn(conn)
```

- Implement it your own way, but be sure to use "synchronized"

Thread Pooling

- If you build your application correctly, the GUI should be separate from the “program”
- Same concept as the Connection Pooling
- More about it when we talk about the GUI

Coding tips

- The following next slides are EXAMPLES for what NOT-TO-DO in the project. Basically they are based on last years submissions, which were altered to express important points.

Database Design

ID	Number
CD_NAME	NVARCHAR(50)
ARTIST_NAME	NVARCHAR(50)
GENRE	NVARCHAR(50)
YEAR	DATE

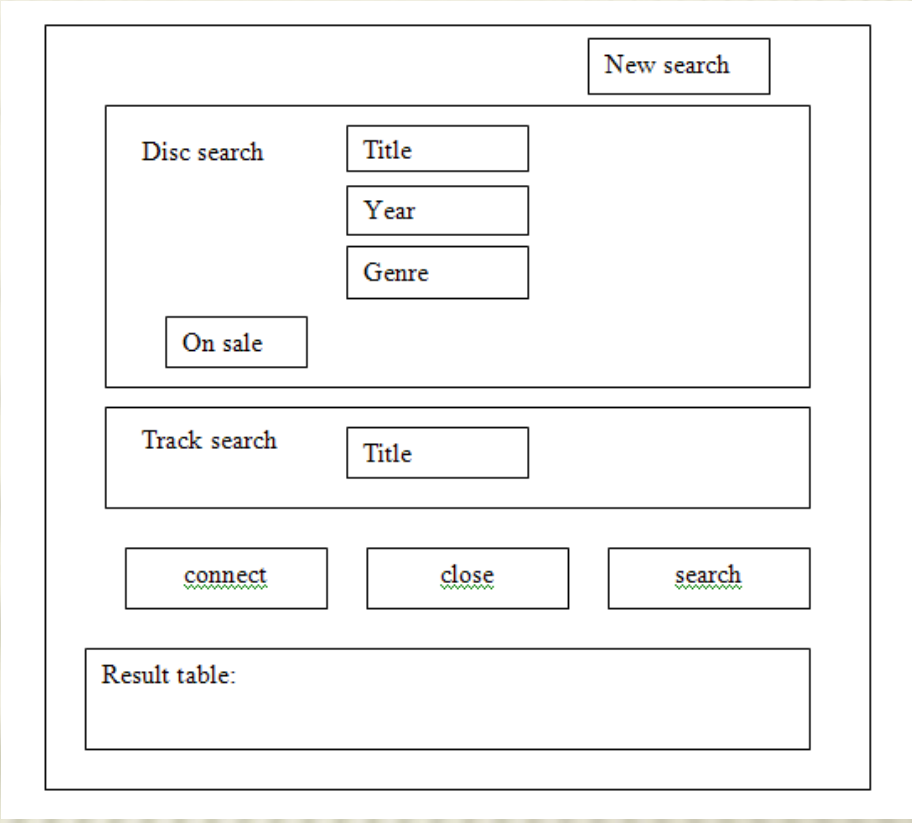
- Don’t forget to normalize the DB
- Use the right types

Usability

- “non-refreshed” windows
- “Hangs” windows

- “Please wait…Your query may take a couple of minutes…”

Usability II



Thank you