Concerning the assignment review, it's really easy. I had the feeling he just asks you little questions to make sure that it's really you who did the assignment.

e.g.

- What does UML mean?
- Is a FK allowed to be null? Yes
- What may cause a FK to be null? DELETE SET NULL
- What's a weak entity type? an entity that doesn't have its own primary key, uses a foreign key from a strong entity type as part of its primary key
- Is there an existence dependency between a WET and its owners? yes, a weak entity type is always existence dependent of its owner
- What's the shortcoming of representing the specialization through set types in the Codasyl model?

Part Bart Baessens:
- Q1 (10p): EER given, map to relational model. Describe (discuss examples) what cannot (semantically) be modeled in the EER or relational model and describe what loss of semantics you have by mapping it to a relational model. Also specify the NOT NULL constraints and primary/foreign keys where needed.

Q2 (10p): describe briefly (in bullets is enough):

1. impedance mismatch
   - Data structures of DBMS and data sublanguage may be different from data structures of host language
     1. Solutions
        1. DBMS and host language with comparable data structures
           1. e.g. object-oriented DBMS combined with Java
        2. Using middleware to map data structures from the DBMS to the host language and vice-versa
   - **Example:** Java combined with MySQL
2. Shared aggregation + example
   - Part object can (at the same time) belong to multiple composite objects
   - Maximum cardinality at composite side is undetermined
   - Part object can also occur without belonging to composite object
   - **Example:** Company with zero or multiple employees
3. BCNF + example
   - Boyce-Codd Normal Form:
     - i. Every relation in BCNF is also in 3 NF; however a relation in 3 NF is not necessarily in BCNF
     - ii. A functional dependency FD: X → Y is called **trivial** if Y is a subset of X
     - iii. A table is in Boyce-Codd normal form if and only if, for every

one of its non-trivial functional dependencies X → Y, X is a superkey—that is, X is either a candidate key or a superset thereof.

- Example:
  i. Supplier (SNR, SNAME, PRODNR, QUANTITY)
  ii. With Superkey {SNR, PRODNR} and superkey {SNAME, PRODNR}
  iii. SNR ® SNAME and SNAME ® SNR
  iv. *After BCNF*
     1. $\text{Supplier}_{\text{BCNF 1}}$ (<u>SNR,</u> SNAME) $R_{\text{BCNF 1}}$ (SNR, <u>SNAME</u>)
     2. $\text{Supplier}_{\text{BCNF 2}}$ (<u>SNR</u>, <u>PRODNR</u>, QUANTITY) $R_{\text{BCNF 2}}$ (<u>SNAME</u>, <u>PRODNR</u>, QUANTITY)

4. View materialization
   - A materialized view is a database object that contains the results of a query.
5. Outer Join + example
   - An outer join does not require each record in the two joined tables to have a matching record. Different outer joins: Left outer join (contains all records of left table), right outer join (all records of right table) and full outer join (all records of both tables)
   - **Example:**

     -
       ```
       SELECT *
       FROM employee LEFT OUTER JOIN department
       ON employee.DepartmentID = department.DepartmentID;
       ```

Employee.LastName	Employee.DepartmentID	Department.DepartmentName	Department.DepartmentID
Jones	33	Engineering	33
Rafferty	31	Sales	31
Robinson	34	Clerical	34
Smith	34	Clerical	34
John	NULL	NULL	NULL
Steinberg	33	Engineering	33

6. Procedural DML
   - DML statements specify **how** to navigate in the database to locate and modify the data
   - Usually starts by positioning on one specific record (data instance) and navigate from there onwards to other records
   - User defines optimizations, if any

- The user must know the details of the system to write good queries
- No query optimizer available
- Record-at-a-time DML

7. Query with NON EXISTS (+ give example (write query))
   - A query that gives a result back if it has no records from the requested state.
   - **<u>Example</u>**:
     i. Retrieve the names of employees who have no dependents
     ii.
```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE NOT EXISTS  (SELECT *
                   FROM DEPENDENT
                   WHERE SSN = ESSN);
```

8. View with check option
   - View: Views can be seen as *virtual* tables, without physical tuples. A view definition consists of a formula that determines which attributes from the *base tables* are to be shown upon invocation of the view. The view's content is generated upon this invocation.
   - If rows are inserted or updated through an updatable view, there is the chance that such row does not satisfy the view definition after the update, i.e. the row cannot be retrieved trough the view.
   - The WITH CHECK option allows to avoid such "unexpected" effects: UPDATE and INSERT statements are checked for conformity with the view definition.

9. 3NF
   - A functional dependency X ® Y in a relation schema R is a ***<u>transitive dependency</u>*** if there is a set of attributes Z that is neither a candidate key nor a subset of any key of R, and both X ® Z and Z ® Y hold.
   - A relation schema is in 3NF if it satisfies 2NF and no nonprime attribute of R is transitively dependent on the primary key.
   - **<u>Example:</u>**
     i. (<u>SSN</u>, ENAME, DNUMBER, DNAME, DMGRSSN)
     ii. After 3NF:
        1. (<u>SSN</u>, ENAME, DNUMBER)
        2. (<u>DNUMBER</u>, DNAME, DMGRSSN)

10. Functional independence
    - Function
      i. Interface (signature): name of the function and its arguments
      ii. Implementation (method): specifies how the function should be executed
    - Implementation (method) can change without impact on software applications
    - **<u>Example</u>**: Sorting “I don't care **how** it is done, just get it done”

11. EER categorization

- Categorization implies that objects are grouped into categories, usually for some specific purpose
- A category is a subclass that has several possible superclasses. Each superclass represents a different entity type. The category represents a collection of objects that is a subset of the *union* of the superclasses.
- Inheritance in the case of categorisation corresponds to an entity inheriting only the attributes and relationship types of that superclass it is a member of (selective inheritance).
- A categorisation can be *total* or *partial*.

Person Company
Arrow to acc. Arrow to acc.
Accountholder

12. SQL with EXISTS
    - A query that gives a result back if it has records from the requested state.
    - **Example**:
        i. Retrieve the names of employees who have dependents
        ii. **SELECT** FNAME, LNAME
        FROM EMPLOYEE
        WHERE EXISTS (**SELECT** *
        FROM DEPENDENT
        WHERE SSN = ESSN);
13. 2NF
    - A functional dependency X ® Y is a ***full functional dependency*** if removal of any attribute A from X means that the dependency does not hold anymore.
    - A functional dependency X ® Y is a ***partial dependency*** if some attribute A Î X can be removed from X and the dependency still holds.
    - An attribute is called a **prime attribute** if it is a member of some candidate key, otherwise, it is called a nonprime attribute.
    - A relation schema R is in 2NF if it satisfies 1NF and every nonprime attribute A in R is fully functional dependent on any key of R.
    - **Example:**
        i. (SSN, PNUMBER, PNAME, HOURS)
        ii. After 2NF
            1. (PNUMBER, PNAME)
            2. (SSN, PNUMBER, HOURS)
14. composite aggregation UML
    - Composite object composed using part objects
    - Part object can only belong to one composite
    - Maximum cardinality at composite side is 1
    - Part object is removed when composite is removed
    - **Example:** Bank with multiple accounts

15. runtime DB processor
    - The runtime database processor is responsible for executing the code (generated by the query code generator), whether in compiled mode or interpreted mode, to produce the response to the query.
16. SQL with group by and having
    - Group By:
        i. The GROUP BY-clause allows to apply aggregate functions *to subgroups of tuples in a relation*, based on some attribute values.
        ii. The tuples that have the same value of some attribute(s), called the *grouping attribute(s)* are grouped. The function is applied to each such group independently.
        iii. The GROUP BY-clause specifies these grouping attributes, which should also appear in the SELECT-clause, so that the value resulting from applying each function to a group of tuples appears along with the value of the grouping attribute(s).
        iv. <u>Example:</u>
            1. For each department, retrieve the department number, the number of employees in the department, and their average salary
            2. **SELECT** DNO, **COUNT**(*), **AVG**(SALARY)
               FROM EMPLOYEE
               GROUP BY DNO;
    - Having:
        i. When a GROUP BY-clause is used, the HAVING-clause allows to select only groups that satisfy certain conditions.
        ii. <u>Example:</u>
            1. On each project on which more than two employees work, retrieve
               the project number, the project name, and the number of employees that work on the project
            2. **SELECT** PNUMBER, PNAME, **COUNT**(*)
               FROM PROJECT, WORKS_ON
               WHERE PNUMBER = PNO
               GROUP BY PNUMBER, PNAME
               HAVING COUNT(*) > 2;
17. Relationship type in the hierarchical model (+example)
    - The hierarchical model only allows 1:n relationship types (**Note that 1:n always refers to the maximum cardinalities!)**
    - Parent record (1 … 1) à Child record (0 … n)
18. Declarative DML
    - DML statements specify **what** data should be retrieved or what changes should be made
    - DBMS determines access path and navigational strategy (query optimizer)
    - Set-at-a-time DML

· One only needs to specify *which* data to retrieve

19. Generalisation in EER
    · Generalisation is the process of minimising the differences between entities by identifying common features.
    · This is the identification of a generalised superclass from the original subclasses. This is the process of identifying the common attributes and relationships.
    · For instance, taking:
        - car(<u>regno</u>,colour,make,model,numSeats)
        - motorbike(<u>regno</u>,colour,make,model,hasWindshield)
        - And forming:
        - vehicle(<u>regno</u>,colour,make,model,numSeats,hasWindshielf)
    · Generalisation (also called abstraction) is the reverse process of specialisation.
    · Generalisation corresponds to a *bottom-up* process of conceptual synthesis.
20. 4NF and example
    · *Multi-valued dependency*: XààY if each X value exactly determines a set of Y values, independently of the other attributes
    · Satisfies BCNF and for every one of its non-trivial multivalued dependencies XààY, X is a superkey
    (COURSE, INSTRUCTOR, TEXTBOOK) à (COURSE, TEXTBOOK) and (COURSE, INSTRUCTOR)
21. Correlated query
    · a correlated sub-query (also known as a synchronized subquery) is a sub-query (a query nested inside another query) that uses values from the outer query in it's where clause
    · à a condition in the WHERE-clause of a nested query references some attribute of a relation declared in the outer query à the inner query is evaluated once for each tuple or combination of tuples in the outer query
    · *Retrieve first name and last name of all employees with a higher wage than the manager of their department*

```
SELECT E1.FNAME, E1.LNAME
FROM EMPLOYEE E1
WHERE E1.SALARY >=
    (SELECT E2.SALARY
      FROM EMPLOYEE E2, DEPARTMENT D
       WHERE E1.DNO = D.DNUMBER
       AND D.MGRSSN = E2.SSN)
```

à Note: no E1 in the second query à for each tuple in E1 the inner query will be executed à correlated query

22. Question about PDBM: when mapping a EER model to CODASYL, the example in the slides got me confused. It even has 1:1 set types. But in Codasyl you can only model (0..1) to (0..n), right? And then by indicating loss in semantics you can point out that some things could not be modelled (such as all minimum one cardinalities)

Yeah I thought that was weird as well but it is my understanding that the cardinalities that are shown are the cardinalities that we WANT to model (i.e. the same cardinalities as the EER model), offcourse we can't model these because, as you said, CODASYL is 0...1 to 0...N only.
I agree, it's confusing and an odd choice. My impression is the same as Tom's.
Also in the slides, when he writes 1:1 (not on a model, but in the description), he actually means a relation (0..1)____________(0..1).

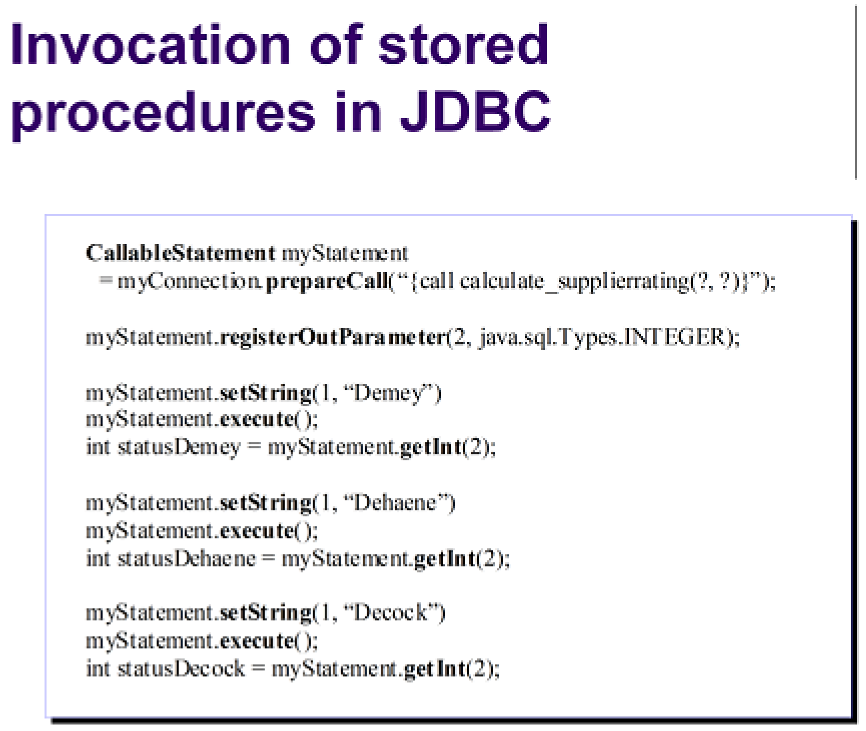
.

Prof. LEMAHIEU's part

Ch6

1- Q1 was +- like this: "Is it possible (in principle) for an application designer to use JDBC instead of API and to switch from an Oracle DB system to a DB2 system without having to rewrite the entire application?
-> I said something about API/JDBC being the mediator between application and database (drew the image with API driver etc... Don't know what chapter that is exactly) and that the drivers translate everything (referred to a printer driver, that the computer does not need to know the type of the printer, the driver translates everything).

2-A piece of Java code with a callable statement. Had to talk about What the code did and witch API and type it is etc.



Invocation of stored procedures in JDBC

```
CallableStatement myStatement
  = myConnection.prepareCall("{call calculate_supplierrating(?, ?)}");

myStatement.registerOutParameter(2, java.sql.Types.INTEGER);

myStatement.setString(1, "Demey")
myStatement.execute();
int statusDemey = myStatement.getInt(2);

myStatement.setString(1, "Dehaene")
myStatement.execute();
int statusDehaene = myStatement.getInt(2);

myStatement.setString(1, "Decock")
myStatement.execute();
int statusDecock = myStatement.getInt(2);
```

Here parameters represented with question marks too. In stored procedure parameters by default input parameters which means you use them for input for stored procedure. There are also parameters for input and output parameters. There is also output parameters for getting result from stored procedure.

So we have two parameters first supplier name, second calculation of supplierrating. The first parameter is input parameter, second parameter output parameter which will contain the result of thecalculation.

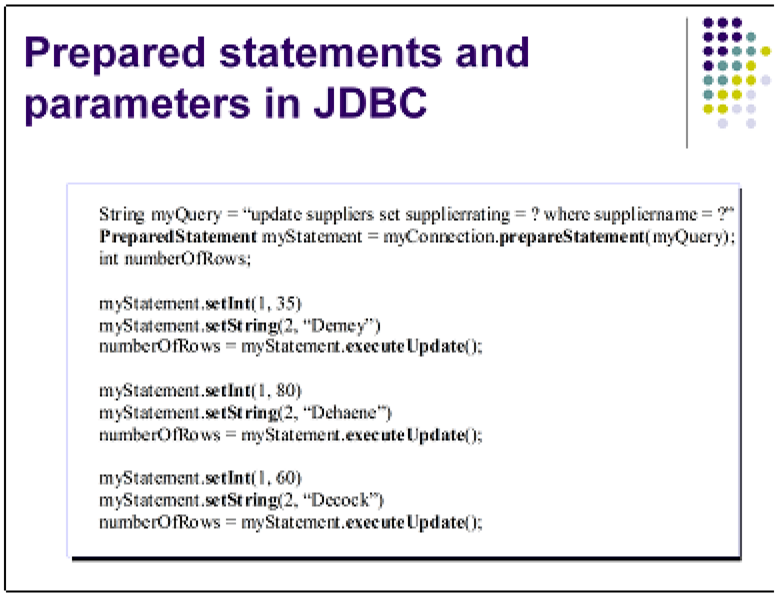
We created stored procedure calculate supplier rating, stored in catalog of database, now we can call it from jdbc by this syntax.

So the first parameter by default input parameter. We define explicitly the second parameter as output parameter.

We set input variable trough stored procedure by means of first parameter as input parameter. We read the result by output parameter. Then we can execute the same stored procedure multiple times. We set different input variables and get different output parameters.

So stored procedure is sql code in catalog which we call it.We only need to know parameters of the stored procedure and the name of the stored procedure. Only internals know the names of the columns and tables.

3- prepared statement, describe the concept where it's used, etc



Prepared statements and parameters in JDBC

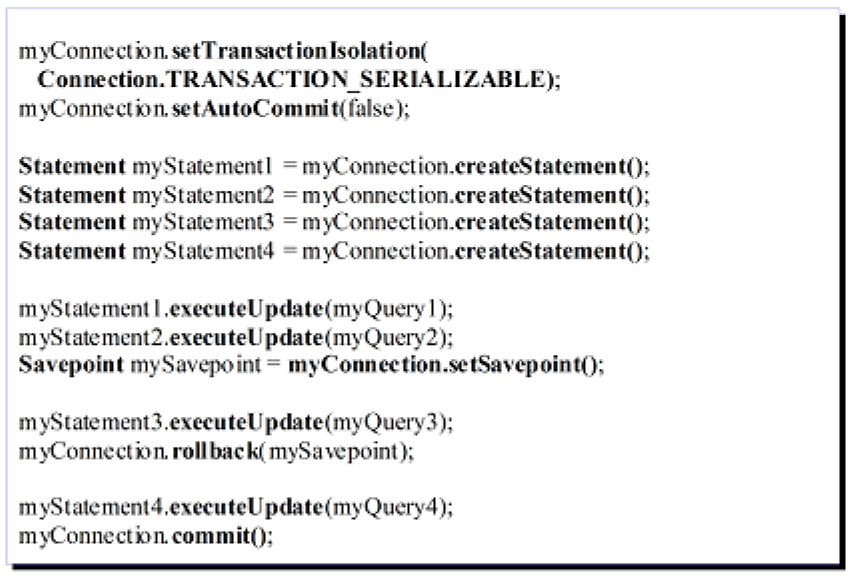
```
String myQuery = "update suppliers set supplierrating = ? where suppliername = ?"
PreparedStatement myStatement = myConnection.prepareStatement(myQuery);
int numberOfRows;

myStatement.setInt(1, 35)
myStatement.setString(2, "Demey")
numberOfRows = myStatement.executeUpdate();

myStatement.setInt(1, 80)
myStatement.setString(2, "Dehaene")
numberOfRows = myStatement.executeUpdate();

myStatement.setInt(1, 60)
myStatement.setString(2, "Decock")
numberOfRows = myStatement.executeUpdate();
```

So with prepared statement you bind once and execute multiple times.

First we define the query. So it's bond to the database, access path is conceived.

First execution set some number supplier name and number. With executeupdate we get an integer value showing how many rows effected with this update. And we execute second and third queries without bonding. If it was normal statement it would be bond and executed together.

4-(Ch6),(Ch7). he gave piece of code with isolated transaction schedule, asked is it a callable interactive or embedded, and which transaction will commit etc.



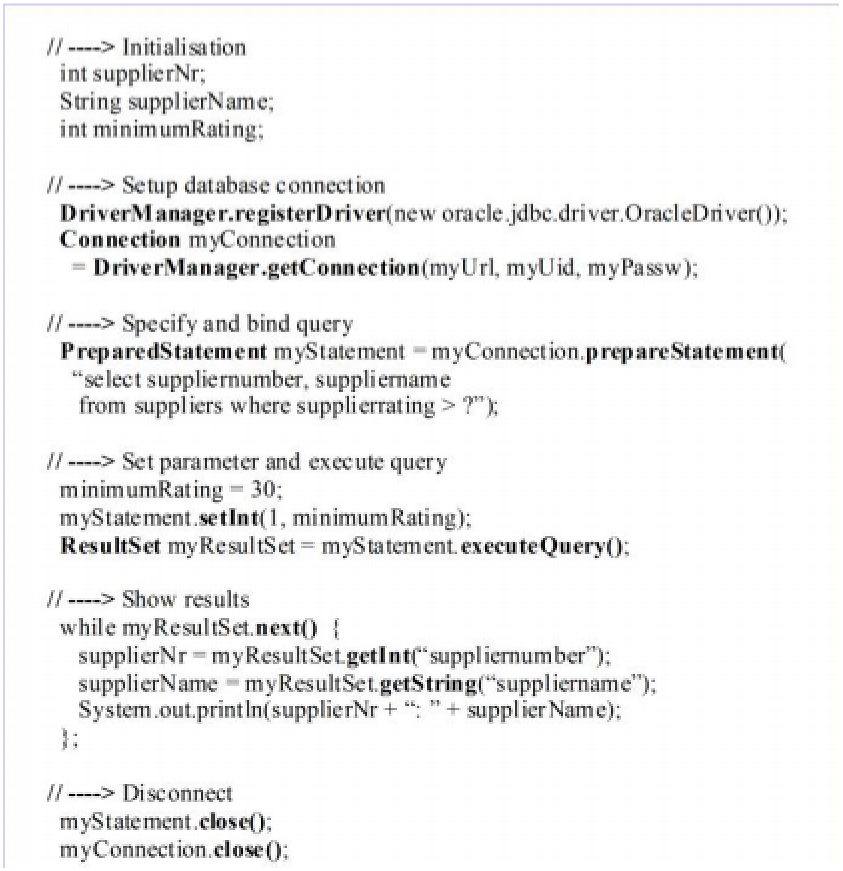
```
myConnection.setTransactionIsolation(
  Connection.TRANSACTION_SERIALIZABLE);
myConnection.setAutoCommit(false);

Statement myStatement1 = myConnection.createStatement();
Statement myStatement2 = myConnection.createStatement();
Statement myStatement3 = myConnection.createStatement();
Statement myStatement4 = myConnection.createStatement();

myStatement1.executeUpdate(myQuery1);
myStatement2.executeUpdate(myQuery2);
Savepoint mySavepoint = myConnection.setSavepoint();

myStatement3.executeUpdate(myQuery3);
myConnection.rollback(mySavepoint);

myStatement4.executeUpdate(myQuery4);
myConnection.commit();
```

setAutoCommit(false)-> not each individual sql statement treated as a transaction,but transaction spanning multiple statement

There are four update statements. Execute first two. We have savepoint there which means intermediate position where you can rollback to. Executed the third statement and rollbacked to the savepoint. Then execute the fourth statement and commit the statement. At the end update 1,2 and 4 are persisted in the database, update 3 is not.

5- Explain Prepared Statements in JDBC? Is there something similar in SQLJ?
For Prepared Statements, you can talk about how it is late binding but not as late as normal statements, so you only have to bind it once and can reuse it as needed. You can also compare it to Stored Procedures which are early binding. I think for SQLJ, you just say that there is no late binding available.

6-You get a piece of java code as in the course:
--what does this do
--what type of api call level or embedded
--late of early binding
--discuss performance, errors, …



```
// ----> Initialisation
  int supplierNr;
  String supplierName;
  int minimumRating;

// ----> Setup database connection
  DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
  Connection myConnection
    = DriverManager.getConnection(myUrl, myUid, myPassw);

// ----> Specify and bind query
  PreparedStatement myStatement = myConnection.prepareStatement(
    "select suppliernumber, suppliername
     from suppliers where supplierrating > ?");

// ----> Set parameter and execute query
  minimumRating = 30;
  myStatement.setInt(1, minimumRating);
  ResultSet myResultSet = myStatement.executeQuery();

// ----> Show results
  while myResultSet.next()  {
    supplierNr = myResultSet.getInt("suppliernumber");
    supplierName = myResultSet.getString("suppliername");
    System.out.println(supplierNr + ": " + supplierName);
  };

// ----> Disconnect
  myStatement.close();
  myConnection.close();
```

Initialisation-> we define some attributes in java; some java variables. Here we define three variables with integer and string values.

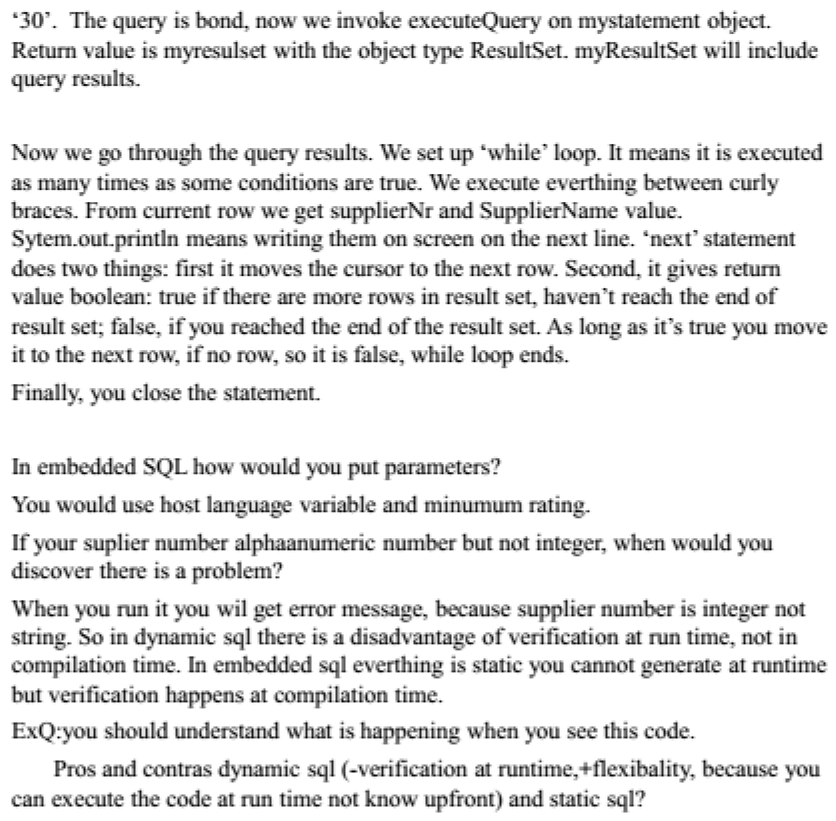
Connection-> we invoke method registerdriver on drivermanager to create a new oracle driver object.

getconnection method is invoked on driver manager with three parameters(myurl, myUid, myPassw) and this method invocation yields object, called myConnection which is of the type connection.

Specify and bind query-> on myconnection object we invoke prepareStatement. Return value is mystatement of the type preparedStatement

For preparedstatement we give parameter of the type string; sql statement. We assigned sql statement to our statement object and it is prepared; it is bond to the database system. So when the preparedstatement executed these string parameters passed to the database system and it is interpreted and bond by database system. At that time it is translated to acces path and something can be executed by database sytem. Bonding happens at runtime, If you compile this with just java compiler it will interprete it just string like 'Hello World'. But when it is executed it is passed to database, hopefully with good query, syntax, access rights it's bond to the database.

Set parameter and execute query-> Here we give value to the '?'. We define minumumRating. Then we invoke setint method on Mystatement object. '1' means the first '?' is replaced with minimumRating. So minumumRating was



'30'. The query is bond, now we invoke executeQuery on mystatement object. Return value is myresulset with the object type ResultSet. myResultSet will include query results.

Now we go through the query results. We set up 'while' loop. It means it is executed as many times as some conditions are true. We execute everthing between curly braces. From current row we get supplierNr and SupplierName value. Sytem.out.println means writing them on screen on the next line. 'next' statement does two things: first it moves the cursor to the next row. Second, it gives return value boolean: true if there are more rows in result set, haven't reach the end of result set; false, if you reached the end of the result set. As long as it's true you move it to the next row, if no row, so it is false, while loop ends.

Finally, you close the statement.

In embedded SQL how would you put parameters?

You would use host language variable and minumum rating.

If your suplier number alphaanumeric number but not integer, when would you discover there is a problem?

When you run it you wil get error message, because supplier number is integer not string. So in dynamic sql there is a disadvantage of verification at run time, not in compilation time. In embedded sql everthing is static you cannot generate at runtime but verification happens at compilation time.

ExQ:you should understand what is happening when you see this code.

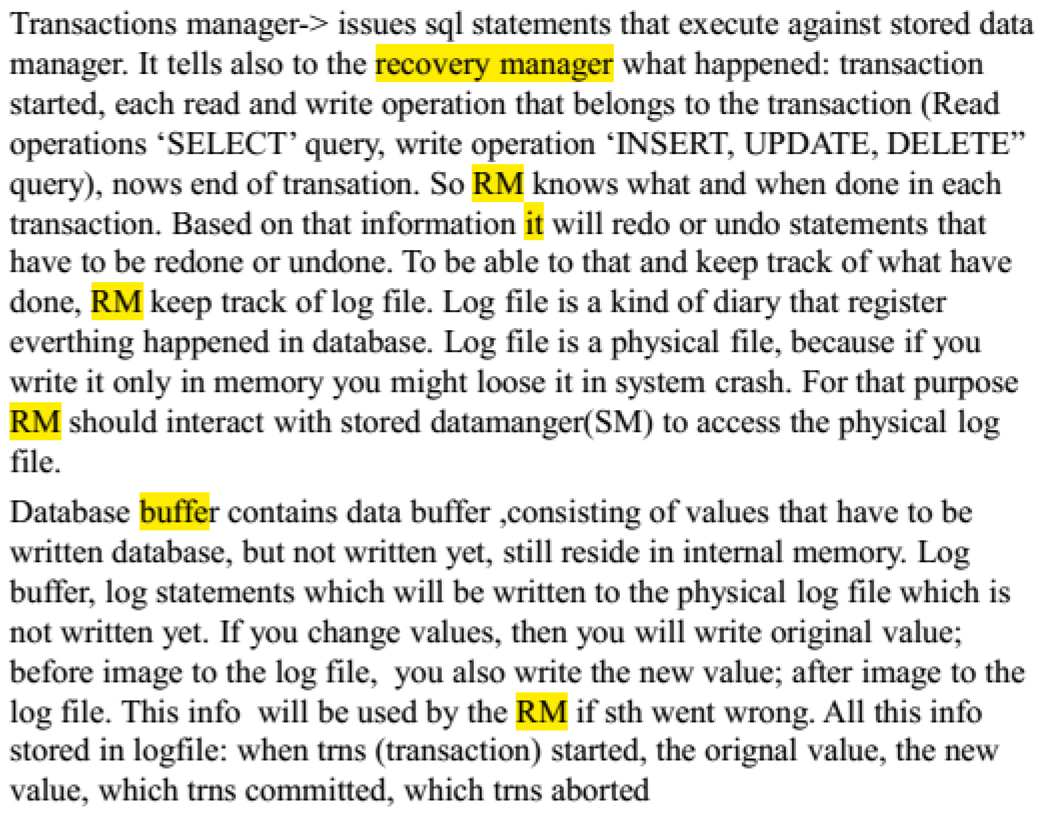
Pros and contras dynamic sql (-verification at runtime,+flexibality, because you can execute the code at run time not know upfront) and static sql?

7-(Example Q) Define and explain the concept of "SQL binding". Discuss for JDBC and SQLJ how and when this binding is accomplished.

One of the difference between SQLJ and JDBC is the binding. SQL binding is translating code to an executable; doing typechecking, validation of syntax etc, checking wether table and column names exist, generating access path where database solve the query.
In JDBC there is call level interface. It has mostly late binding, because SQL code is provided as parameter to the method call. SQL query will only be bond to the databse at runtime which was treated as string during compilation.
In SQLJ there is embedded SQL. So it is embedded in host language and precompiled. Long before runtime it is bond to the database
+ of SQLJ>you can make typechecking earlier so you don't have to it on runtime.
Are any solutions in JDBC that allow to do binding more efficiently?
There is prepared statement which means you do the binding only once and you can execute the query multiple times. It is more efficient then binding it at each execution.

Ch7

1- was +- like this: Explain in detail what the role of the recovery manager in the database system is. And how is this linked to the buffer.

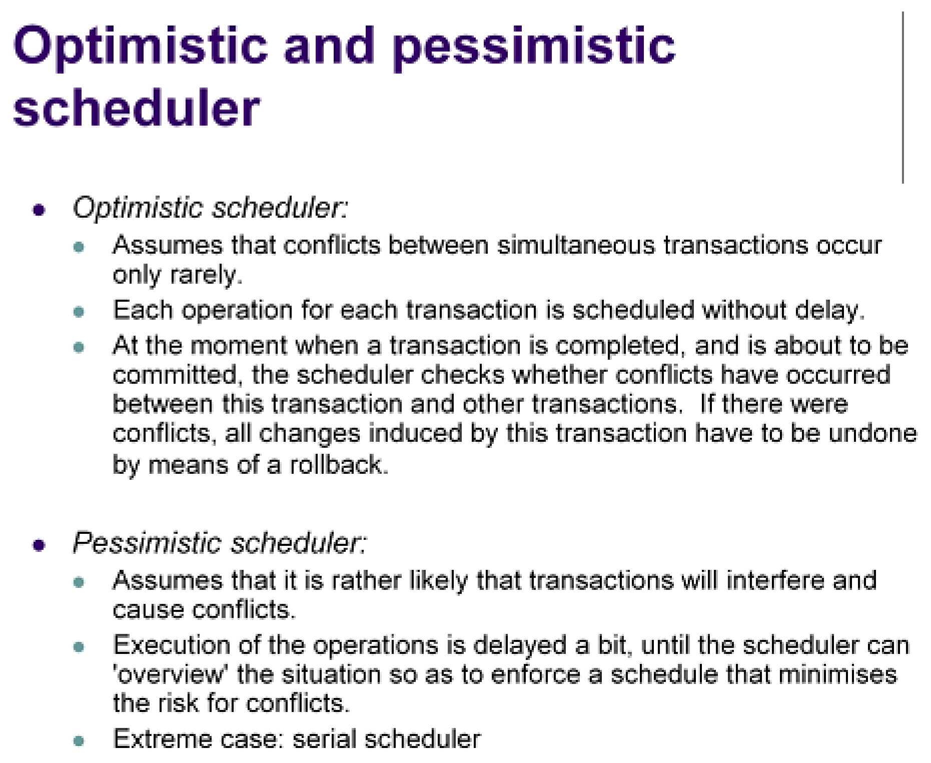


Transactions manager-> issues sql statements that execute against stored data manager. It tells also to the recovery manager what happened: transaction started, each read and write operation that belongs to the transaction (Read operations 'SELECT' query, write operation 'INSERT, UPDATE, DELETE" query), nows end of transation. So RM knows what and when done in each transaction. Based on that information it will redo or undo statements that have to be redone or undone. To be able to that and keep track of what have done, RM keep track of log file. Log file is a kind of diary that register everthing happened in database. Log file is a physical file, because if you write it only in memory you might loose it in system crash. For that purpose RM should interact with stored datamanger(SM) to access the physical log file.

Database buffer contains data buffer ,consisting of values that have to be written database, but not written yet, still reside in internal memory. Log buffer, log statements which will be written to the physical log file which is not written yet. If you change values, then you will write original value; before image to the log file, you also write the new value; after image to the log file. This info will be used by the RM if sth went wrong. All this info stored in logfile: when trns (transaction) started, the orignal value, the new value, which trns committed, which trns aborted

2-What is serializable? How can we enforce it? How does it relate to optimistic and pessimistic concurrencies?

If non-serial schedules are equivalent to a serial schedule, they are serialisable. For serial schedules, all transactions are processed consecutively.
Scheduler can check for each non-serial schedule whether it is serialisable. Alternatively, we can apply locking protocols which guarantees serialisability.



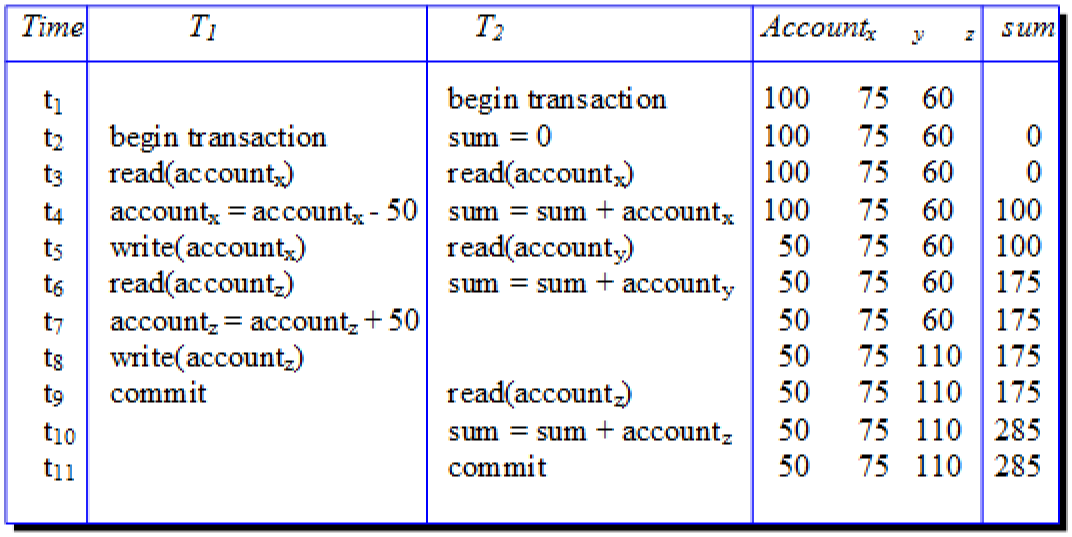
Optimistic and pessimistic scheduler

- *Optimistic scheduler:*
  - Assumes that conflicts between simultaneous transactions occur only rarely.
  - Each operation for each transaction is scheduled without delay.
  - At the moment when a transaction is completed, and is about to be committed, the scheduler checks whether conflicts have occurred between this transaction and other transactions. If there were conflicts, all changes induced by this transaction have to be undone by means of a rollback.
- *Pessimistic scheduler:*
  - Assumes that it is rather likely that transactions will interfere and cause conflicts.
  - Execution of the operations is delayed a bit, until the scheduler can 'overview' the situation so as to enforce a schedule that minimises the risk for conflicts.
  - Extreme case: serial scheduler

3- Given: Table with 2 transactions ('dirty read problem table)
Q: Is this serializable? Why (not)?

Not serialasible. t1 reads intermediate result before rollback.

4- (Example Q) Examine the following transaction schedule. Is it serialisable ? Why (not) ?



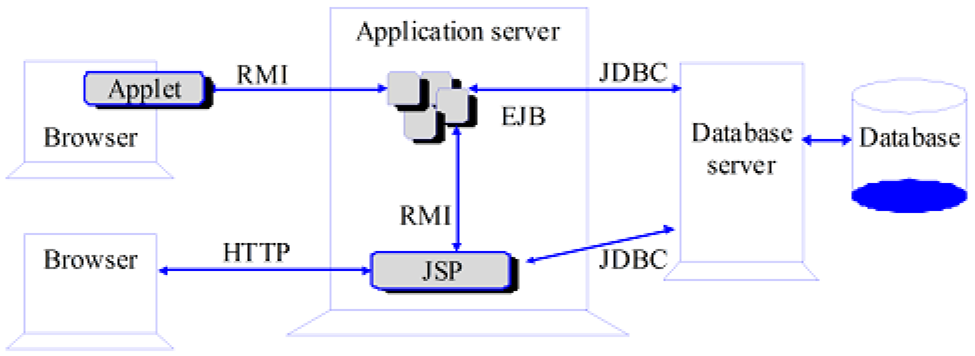
Time	T_1	T_2	$Account_x$	y	z	*sum*
t_1		begin transaction	100	75	60	
t_2	begin transaction	sum = 0	100	75	60	0
t_3	read($account_x$)	read($account_x$)	100	75	60	0
t_4	$account_x = account_x - 50$	$sum = sum + account_x$	100	75	60	100
t_5	write($account_x$)	read($account_y$)	50	75	60	100
t_6	read($account_z$)	$sum = sum + account_y$	50	75	60	175
t_7	$account_z = account_z + 50$		50	75	60	175
t_8	write($account_z$)		50	75	110	175
t_9	commit	read($account_z$)	50	75	110	175
t_{10}		$sum = sum + account_z$	50	75	110	285
t_{11}		commit	50	75	110	285

It is not serialisable. At least you have to explain the difference between serial and serialisable transactions. The problem is here with timing. An update is overwritten

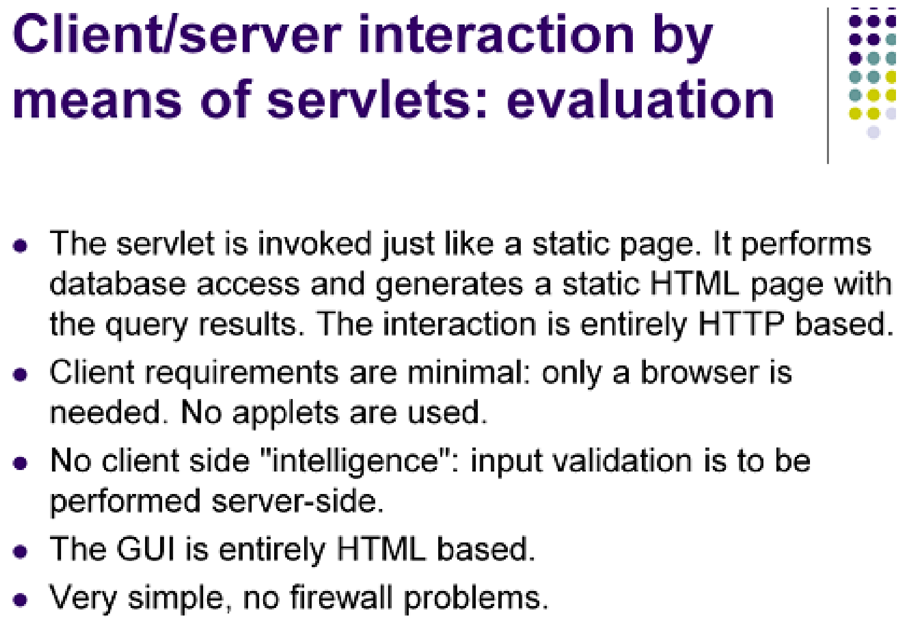
Ch8

1- We had a picture of the "Client/server interaction by means of a distributed object architecture" (see chap. 8). The question was "How could you adapt these architecture for browsers which can't use applets while keeping a distributed objects architecture".

There were two possibilities. The first one with server side scripts (JSP + RMI or ASP + WCF) as it's given in the slides. On the other hand, it's also conceivable through servlets (even if it's not really represented in the course).

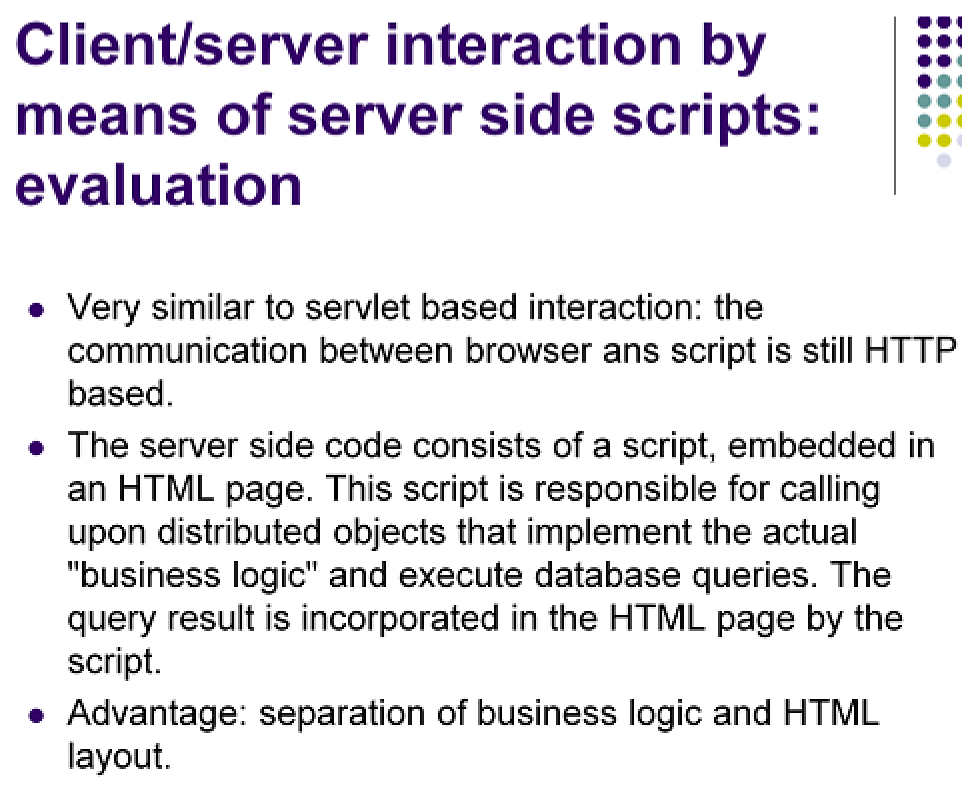


Afterwards he wanted me to give the advantage of using either server scripts or servlets. I didn't know the answer because in the course it's only written that they are quite similar. The answer is something like "separation between content and layout". I must admit I don't really know what it means.



Client/server interaction by means of servlets: evaluation

- The servlet is invoked just like a static page. It performs database access and generates a static HTML page with the query results. The interaction is entirely HTTP based.
- Client requirements are minimal: only a browser is needed. No applets are used.
- No client side "intelligence": input validation is to be performed server-side.
- The GUI is entirely HTML based.
- Very simple, no firewall problems.



Client/server interaction by means of server side scripts: evaluation

- Very similar to servlet based interaction: the communication between browser ans script is still HTTP based.
- The server side code consists of a script, embedded in an HTML page. This script is responsible for calling upon distributed objects that implement the actual "business logic" and execute database queries. The query result is incorporated in the HTML page by the script.
- Advantage: separation of business logic and HTML layout.

2-(Example Q) Given: the water quality measurement database (discussed in a previous question). Suppose the Flemish government wants to make the database content available to two types of users: first, to the general public (such that people know where they can swim safely) and second, to some researchers that want to conduct detailed analyses of the evolution in water quality over time. Both types of users only have “read” access: they cannot update the database. Explain thoroughly how each of the following technologies can contribute to the interaction between user and database. Indicate also (and explain why !) whether they can be useful for only the general public, only the researchers or both. If you make any additional assumptions, then state them explicitly.

- JDBC
- RMI
- Server-side scripting

This is question about how you access the data. For general public you don’t need a complex business logic so server-side scripting would be enough. You will be need JDBC also in the server to access the database, not in web browser.
For researcher RMI might be used to get detailed answers to the queries with a nice interface, so they can look into all the data from all the angles. Plain HTML wouldn’t be enough, so it would be better to have an applet that directly call some server side

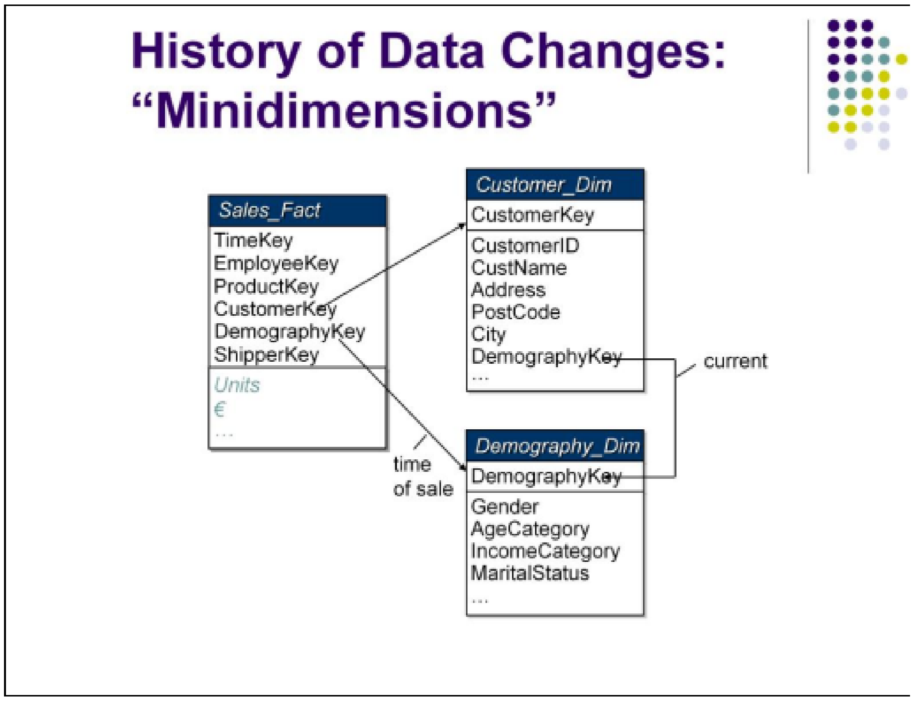
functionality through RMI. That’s in turn will access the database.
You can also say: I don’t want to bother researchers bother with firewall problem, so provide them rich internet application which doesn’t provide RMI

Ch9

1- describe denormalisation when working with DWH's. Give advantages, disadvantages, example.

When I talked about performance reasons (joins are most intensive query's to process), he asked if this issue is the same when working with operational databases. Here I referred to the big amount of simple query's vs a small amount of complex query's in DWH. I also mentioned snowflake schema and used the time dimension example.

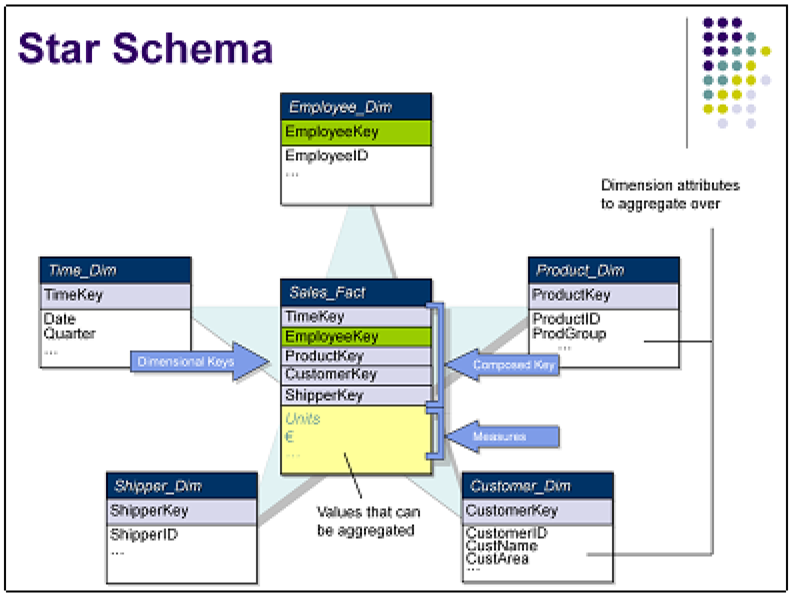
2- Explain what mini dimension is. How is this related to slowly changing dimensions. Give an example.



History of Data Changes: "Minidimensions"

Minidimensions-> you can divide set of attributes which likely changes to a subdimension. Current version of subdimension refered to the dimensiontable with a key. The old version is referred to the fact table.

3-Make a Star datawarehouse for the police for burglaries. And then also dealing with a changing dimension table.

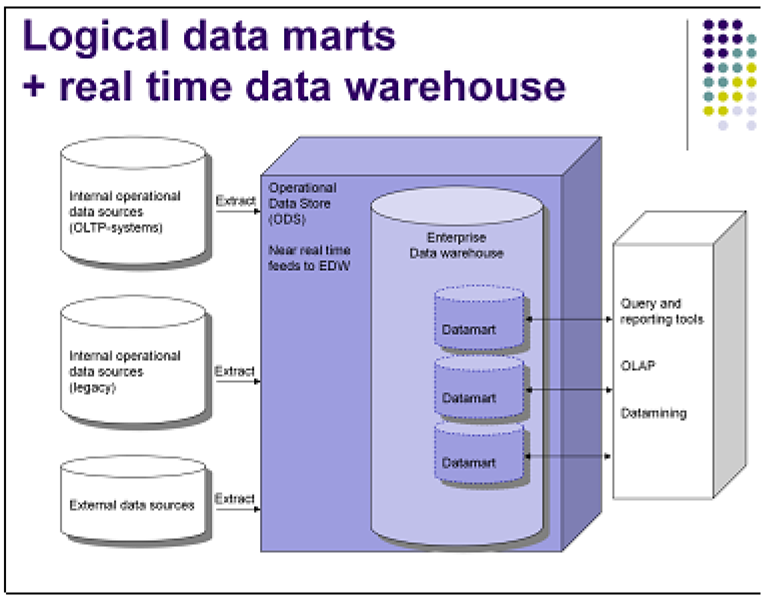


Primary key of fact table (composed key)-> combination of all the surrogate key of dimension tables

Surrogate key-> primary key of dimension tables which only serve to relate to the fact table

Why surrogate key, not original primary key?->original key is usually large, so it will take lots of space

4- data warehouse architecture with dependent data marts and operational data store, advantages + disadvantages, explain how it works



Logical data marts + real time data warehouse

Internal operational data sources (OLTP-systems)
Internal operational data sources (legacy)
External data sources
Extract
Extract
Extract
Operational Data Store (ODS)
Near real time feeds to EDW
Enterprise Data warehouse
Datamart
Datamart
Datamart
Query and reporting tools
OLAP
Datamining

Real time DW-> EDW is integrated with ODS. EDW is just a subset or additonal set of data along with ODS.

Logical datamarts->datamarts are not separate physical entities anymore but view onn EDW.

pros of logical D-> they are always in sync with DW

Contras->all users will use EDW,so if there is heavy user department, they might be burden on other users.

5-what is a purpose of surrogate key in DW.

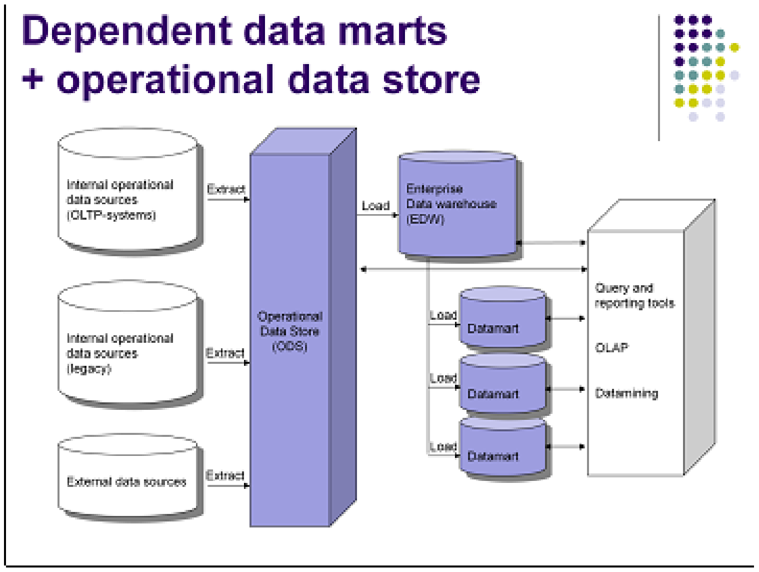
It is key of dimension table which replace original primary key of original relation, eg. EmployeKey instead of EmployeeID. Original primary key is usually too long, so using surrogate key we save some space.

The other reason is capturing slowly changing dimensions. Fex, when a customer moves, in operational database we change the address. But in Datewarehous we should add as another address. Using another surrogate key, we add the same customer id and but the new address. So we will have the same customerid with different surrogate keys. If we didn't use it, we couldn't capture this change.

6- Discuss all possible ways data marts can be integrated with a data warehouse

But the second question from Lemahieu is more to discuss the way of the ARCHITECTURE where DataMarts & Enterprise Datawarehouses are integrated... There are 3 architectures shown in picuteres of chapter 9... unfortunately I noticed this after the exam.

Datamart is loaded from staging area. Here staging area has only flat files, seperated with commas. It is used for formatting and transforming data, not for analysis.



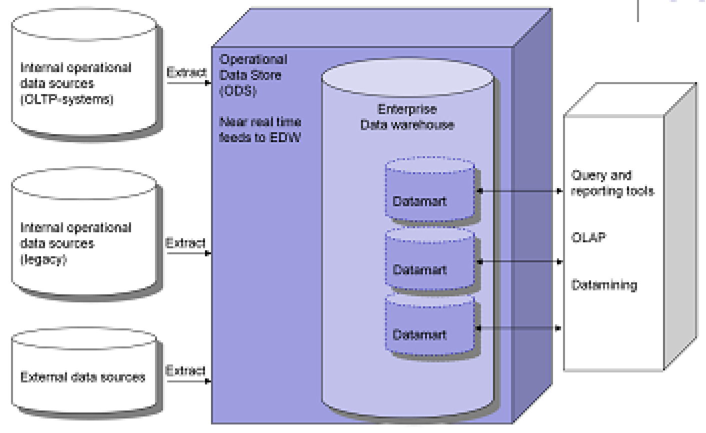
Dependent data marts + operational data store

Internal operational data sources (OLTP-systems)
Internal operational data sources (legacy)
External data sources
Extract
Extract
Extract
Operational Data Store (ODS)
Load
Enterprise Data warehouse (EDW)
Load
Datamart
Load
Datamart
Load
Datamart
Query and reporting tools
OLAP
Datamining

Once my enterprise wide DW operational,then I first feed it to enterprise DW and Datamarts get its data only from DW. So it is edditional step but you are sure that data is consistent in DW and datamarts.

Staging area is now called ODS which is not flat files but a database which can be queried.It contains all the detailed data from operational data.

Logical data marts
+ real time data warehouse


Internal operational data sources (OLTP-systems)
Internal operational data sources (legacy)
External data sources
Extract
Extract
Extract
Operational Data Store (ODS)
Near real time feeds to EDW
Enterprise Data warehouse
Datamart
Datamart
Datamart
Query and reporting tools
OLAP
Datamining

Real time DW-> EDW is integrated with ODS. EDW is just a subset or additonal set of data along with ODS.

Logical datamarts->datamarts are not separate physical entities anymore but view onn EDW.

pros of logical D-> they are always in sync with DW

Contras->all users will use EDW,so if there is heavy user department, they might be burden on other users.

7-explain in detail snowflake schema and compare to star schema

Used for data warehouse design, star schema has fact table and dimension tables. There are transactions, events in dimension table, which is usually denormalized, eg. employee_dim. Instead of the primary key of the original relation, here a surrogate key is used. Because operational primary keys is too long. It relates to the fact table.

Fact table includes all the keys of dimension tables and some measurements, eg. unit(kg). The primary key of fact table is collection of all the surrogate keys of dimensions.

In Snowflake schema is dimension tables are more normalized. For example, marital status is another dimension tables which refers to employee dimension table.

8- Explain the following query and especially the role of “GROUP BY ROLL UP” and “CASE GROUPING”.

```
SELECT CASE GROUPING(COUNTRY)
WHEN 1 THEN ‘ALL’ ELSE COUNTRY END AS COUNTRY,
CASE GROUPING(PROVINCE)
WHEN 1 THEN ‘ALL’ ELSE PROVINCE END AS PROVINCE,
CASE GROUPING(CITY)
WHEN 1 THEN ‘ALL’ ELSE CITY END AS CITY,
COUNT (*) AS NUMBER, AVG(SALARY) AS
AVERAGE-SALARY
FROM SALARY-DISTRIBUTION
GROUP BY ROLLUP(COUNTRY, PROVINCE, CITY);
```

Could someone expain to me te difference between GROUP BY ROLLUP and GROUP BY ROLLUP with CASE GROUPING? I don't really understand the CASE part. If possible, could someone as well explain CUBE?
For example: case grouping (city) returns a value of 1 if the city values are aggregated. So the group by roll up first groups the city values, afterwards the province values. When this grouping of the province values happens, 'abstraction' is made of the city values (they are aggregated) so normally the cells for the city values return a NULL. When this is the case the case grouping will return a 1, otherwise it 'll return a 0 (when the cells still display a value). I already forgot the cube though... but it's quite similar.

If CASE GROUPING isn’t used with GROUP BY ROLLUP, the result will include null values for attributes which didn’t aggregated. These null might be confused with other null values which stands for values unknown or irrelevant. With the help of CASE GROUPING, In the result we see “ALL” if particular attribute is aggregated, we get the name of particular attribute if particular attribute didn’t used for aggregation.

GROUP BY ROLLUP is used for attributes from same dimensions, eg. country, province, city, whereas GROUP BY CUBE is used for different dimensions, eg. gender, birthofdate

9-(Example Q) Explain the concept of 'federated databases'. Give a concrete (!) example of a situation where a federated database is preferable over a data warehouse. Give also an example where a data warehouse is preferable.

Management has a dashboard application where they can see the current situation of the company. In that situation, giving up-to-date information, federated database could be preferable.
Data warehouse would be preferable, if historical data needed. Fex, compare current

Christmas sales with last ten year’s sale.