apuntes-de-todas-las-lecciones-m...

Syrei

GESTIÓN Y ADMINISTRACIÓN DE BASES DE DATOS

3º Grado en Ingeniería Informática

Escuela Politécnica Superior (Jaén)
Universidad de Jaén

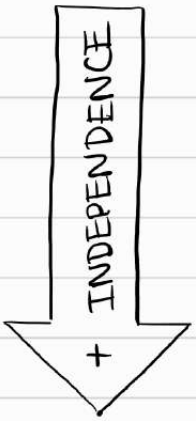
LESSON 1A

ADVANTAGES OF THE ARCHITEXTURE CLIENT / SERVER

1. Delegation of responsibilities
2. Independence of the physical situation of the data
3. Server exploits powerful characteristics of Hw+So
4. Possible to accomplish optimization and updating independently
5. Use of the network: concurrent access + decrease traffic

- the server solves queries

LAYERS OF ABSTRACTION (of a database)



- PHYSICAL: data storage (files)
- LOGICAL: these structures (tables)
- CONCEPTUAL: abstraction, design
- VIEW: how we see the data

TABLE DEFINITION

- Null atributes
- Unique (no duplicates)
- Check (condition)
- Primary key
- External key
- Foreign key (referential integrity)
- Triggers or sequences (identity integrity)

- SEQUENCES
  - Provides unique values to primary keys
  - Avoids programming sequences
  - Improves the concurrence

```
CREATE SEQUENCE <name>
  [INCREMENT BY <number>] [START WITH <number>]
  [MAXVALUE <number> | NOMAXVALUE]
  [MINVALUE <number> | NOMINVALUE]
  [CYCLE | NOCYCLE]
  [CACHE <number> | NOCACHE]
  [ORDER | NOORDER];
```

↳ cache: stores items in the main memory (the rest are in an external secondary disk)

- MANAGEMENT
  - Currval: current value
  - Nextval: next value

* when trying to use `CURRVAL` before any `NEXTVAL` (you don't obtain the first value) an `EXCEPTION` will happen

* `DUAL`: "fake" table, it is used to obtain the values of the current environment

1. EXISTEN TRES TIPOS DE ESCALA PARA MEDIR LA TEMPERATURA. ¿CUÁLES SON?

A) CELSIUS, FAHRENHEIT Y KELVIN

B) DORITOS FLAMIN' HOT, RUFFLES FLAMIN' HOT Y CHEETOS FLAMIN' HOT

2. SI PABLO TIENE 9 DORITOS FLAMIN' HOT Y MARÍA LE QUITA UNO, ¿QUIÉN SE PICA MÁS?

A) PABLO

B) LA LENGUA DE MARÍA

3. UN TREN SALE DE MADRID A 200 KM/H A LAS 9 AM, Y OTRO DE BARCELONA A 100 KM/H A LAS 10 AM... Y TÚ, ¿ESTÁS FLAMIN' HOT?

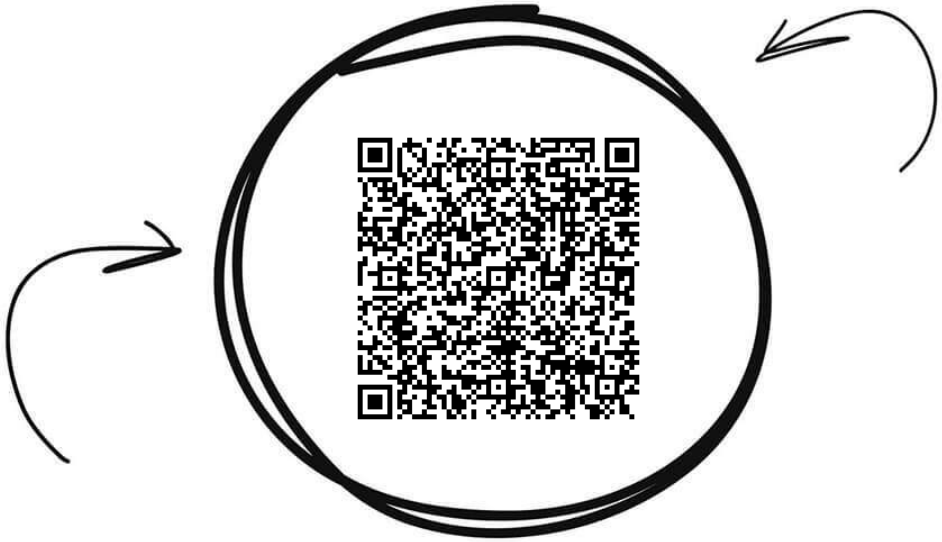
A) EL DE BARCELONA

B) SÍ, ESTOY FLAMIN' HOT

SI HAS RESPONDIDO B) A TODAS LAS PREGUNTAS, ENHORABUENA, ESTÁS FLAMIN' HOT

SI HAS RESPONDIDO A) EN CUALQUIERA DE ELLAS, VUELVE AL INICIO DEL TEST

GESTIÓN Y ADMINISTRACIÓN DE...



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

PROCEDURAL EXTENSIONS TO SQL: PL/SQL

Turning SQL into a traditional procedural language

(1) BLOCK OF PL/SQL

```
DECLARE
    [ constants, variables, cursors, exceptions ]
    BEGIN
        [ PL/SQL sentences ]
        [ EXCEPTION exception handling ]
    END;
/
```

- VARIABLE DECLARATION

```
<name> [ CONSTANT ] { base_type | anchor_type }
       [NOT NULL ]   ↳ domain (table or column)
       [ := expr ] → default value
```

- base_type: date, varchar, number, boolean...
- anchor_type:

  { table / cursor } %ROWTYPE or table.attribute %TYPE

(2) SELECT assign result to variables

```
SELECT (...) INTO <variables_list>
FROM (...);
    DECLARE
        <variable> TYPE or table.attribute %TYPE;
    BEGIN
        [normal select query: result MUST be a single value];
```

(3) CONTROL STRUCTURES

- LOOP

(a)
```
<tag> LOOP
        (sentences)
      END LOOP;
```

(b)
```
<tag> WHILE (condition) LOOP
        (sentences)
      END LOOP;
```

(c)
```
<tag> FOR (counter | variable)
      IN ([REVERSE] begin_value ... end_value |
            cursor | select_query) LOOP
         [sentences]
      END LOOP
```

- LOOP EXIT

```
EXIT [tag] [WHEN (condition)];
```

- CONDITIONAL

```
IF (condition) THEN (sentences)
[ ELSIF (condition) THEN (sentences)]
[ ELSE (sentences) ]
END IF;
```

(4) CURSORS

DEF.: A working area that stores the data returned by a 'select' and allows the access to it.

- CIRCLE OF LIFE
  1 - Cursor declaration
  2 - Variables declaration
  3 - Open
  4 - For i each row: variable_i ← row_i
  5 - Close

- CURSOR DECLARATION

```
CURSOR <name> [(parameter IN type ...)]
IS (select)
[FOR UPDATE OF (attribute)];
```

- OPEN CURSOR

The select query is executed when the cursor opens

```
OPEN <cursor> [(value ...)];
```

- FOR i EACH ROW

```
FETCH <cursor>
INTO {variable_list | variable};
```

- CLOSE CURSOR

```
CLOSE <cursor>;
```

STATE OF THE CURSOR	
	cursor_name%FOUND: Last fetch returned a row
	cursor_name%NOTFOUND: Last fetch didn't return a row
	cursor_name%ISOPEN
	cursor_name%ROWCOUNT

STORED PROCEDURES

Sequence of sentences associated with a name and that can be called.

- Stored in the database, they make easier:
  1 - Development of applications
  2 - Integrity and safety
  3 - Execution time optimization
  4 - Memory optimization

(A) SYNTAX

- STORED PROCEDURE:

```
CREATE PROCEDURE <name>          → [parameter type]: input
    [ <parameter> {IN | OUT | IN OUT} <type> ... ]
    IS (...); → block of code (body)
```

- FUNCTION:

```
CREATE FUNCTION <name> [ <parameter> IN <type> ]
RETURN <type>
    IS (...);
```

(B) PACKAGES

Procedures and functions can be all together in packages:

- Modularity
- Data hiding
- Improves the execution: loaded in the memory
- Application design
- Shared code

```
CREATE PACKAGE <name> AS ['variables' and 'public cursors']
    heads of functions and procedures
END;
CREATE PACKAGE BODY <name> AS ['variables' and 'private cursos']
    implemetation of functions and procedures
    [ BEGIN ( ... sentences ... )
END;
```

(C) TRIGGERS

Stored code associated with a table: it is executed when a certain event (insert, update, delete) takes place.

- Useful for
  - Generate values of columns
  - Complex data constrains
  - Backups and datacopies
- They MUST NOT hide design errors.
- They MUST be simple (± 30 lines)
- They MUST NOT replace integrity constraints.

- Definition

```
CREATE TRIGGER <name_trigger>
    { BEFORE | AFTER | INSTEAD OF (attribute) } → when
query(s) ← { DELETE | INSERT | UPDATE OF (attribute) }
    ← [ OR { DELETE | INSERT | UPDATE OF (attribute) } ]...
ON  <table> → on which table
    [ FOR EACH ROW [ WHEN (condition) ] ]
      DECLARE                          ← type of trigger
          (variables),
      BEGIN   ← can include exceptions
          (sentences);
      END <name_trigger>;
```

access to old/new values
:old and :new
(without "i" when inside in WHEN clause)

TYPE OF TRIGGER

(1) SENTENCE TRIGGER: once per sentence

(2) ROW TRIGGER: once per row
- access the values of the rows
- modify the row new values: BEFORE

- IF SEVERAL TRIGGERS FOR THE SAME DML query.

ORDER:
1. BEFORE row
2. DML sentence
3. AFTER row

manipulation: insert, update, delete.

- REPLACEMENT TRIGGERS

```
CREATE TRIGGER <name_trigger>
INSTEAD OF <DML> ON TABLE
[FOR EACH ROW] (...)
```

- COMPOUND TRIGGERS

The same trigger is executed along several moments
- use: share data along an event.

```
CREATE TRIGGER <name_trigger>
  {FOR { DELETE | INSERT | UPDATE OF (attribute) }
  [OR { DELETE | INSERT | UPDATE OF (attribute) }]...
ON <table>
  [ FOR EACH ROW [WHEN (condition)]]
  COMPOUND TRIGGER IS
        (initial section)
        [variables] [subprogram] [event] ...
/
```

- MUTANT TABLE

The table that is being modified
- Row triggers CAN'T read/modify its row
  ↳ EXCEPTION: BEFORE INSERT affects 1 row
- SOLUTION 1: auxiliary table + new sentence release
  (1) BEFORE: NEW SENTENCE TRIGGER
    - Aux table ← data needed
  (2) ROW TRIGGER
    - Read data needed from aux table
    - Local variable ← data that will be modified
  (3) AFTER: (NEW) SENTENCE TRIGGER
    - Read data from aux table and realites the changes

- SOLUTION 2: compound trigger
  (1) BEFORE: EACH ROW (event)
    - Aux table ← data needed
  (2) BEFORE: STATEMENT (event)
    - Read data needed from aux table
    - Local variable ← data that will be modified
  (3) AFTER: EACH ROW (event)
    - Read data from aux table and realites the changes

EXCEPTIONS HANDLING

The errors in execution time throw exceptions

- System exception
- User defined exception

```
DECLARE ... <name> EXCEPTION;
```

DATA DICTIONARY

- USER_TABLES
- USER_CONSTRAINTS
- USER_CONS_COLUMS
- USER_SOURCE
- USER_VIEWS
- USER_TRIGGERS
- USER_OBJECTS
- USER_ERRORS

EXERCISES

1. Lets suppose that you are connected to a BBDD in Oracle from a Java client . Considering the following assumptions, what would you use: a view, a SQL sentence or a stored procedure?

A) The client query is very dynamic and variable, depending on large number of parameters
SQL sentence

B) It is a extremely complex need of information in order to obtain a only value
stored procedure: function

C) A query that returns a relation that is intended to be used as a part of other queries
stored producedure: view

D) A set of queries that are always executed sequentially
SQL sentences or stored procedure (but not a function nor a view)

4. The information independence implies:
   1. If the physical information model is modified then the logical information model must be accordingly modified
   2. The logical information model modification does not necessarily imply modifying the physical information model
   3. Any modification in a BBDD will necessarily imply the modification of the applications tha her gain access
   4. The information independence is established only between the physical and logical layer of the BBDD
5. With regard to the variables in PL/SQL:
   1. They can be primitive type and of anchor type — base/anchor type
   2. Those of anchor type can be of the same type as a column of a table
   3. Those of anchor type can store several lines of a table
   4. Those of anchor type can store a only line of a cursor
6. With regard to the stored procedures:
   1. It is a procedure that is executed in the DBMS
   2. It is a procedure that is stored in the DBMS but it is executed in the client that invokes it
   3. It is a procedure that is executed in the DBMS but it is stored in the client that invokes it
   4. It allows to share code among different sessions

- UPDATE X SET X.a = X.a + 1
- CREATE TRIGGER tr_name
  AFTER UPDATE
  [FOR EACH ROW]

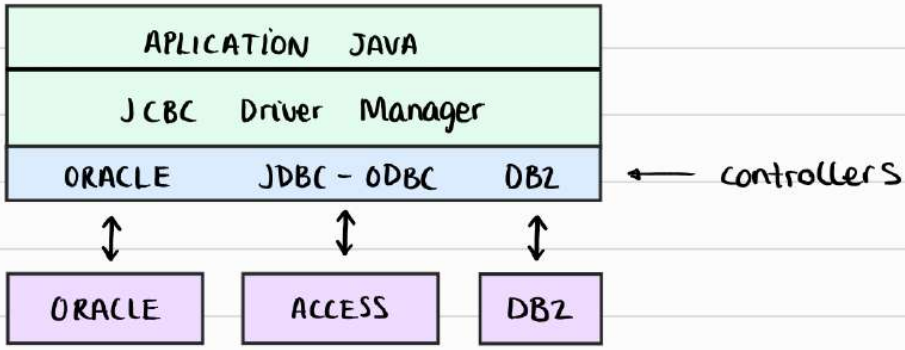
7. Let the SQL sentece: `UPDATE X SET X.a=X.a+1`. Let a trigger with the following header: "`CREATE TRIGGER TRG_UPD_X_A AFTER UPDATE FOR EACH ROW`":
   1. It will be executed once in any case
   2. It will be executed once per updated line of the table X
   3. It will not get access neither `:new` nor `:old` variables
   4. It will get reading access to the table X
8. Let the SQL sentece: `UPDATE X SET X.a=X.a+1`. Let a trigger with the following header: "`CREATE TRIGGER TRG_UPD_X_A AFTER UPDATE`":
   1. It will be executed once in any case
   2. It will be executed once per updated line of the table X
   3. It will not get access neither `:new` nor `:old` variables
   4. It will get reading access to the table X
9. A mutant table:
   1. It cannot be read by a row trigger — (no row triggers)
   2. It cannot be modified by a row trigger
   3. It can be read by a sentence trigger
   4. It is a radioactive table
10. The data dictionary:
    1. It is predefined by the DBMS, which keeps record of it
    2. A user is able to make read-only operations (partially)
    3. It describes, among other things, the logical model of information of the DDBB
    4. It describes, among other things, the physical model of information of the DDBB

LECCIÓN 1B

JDBC

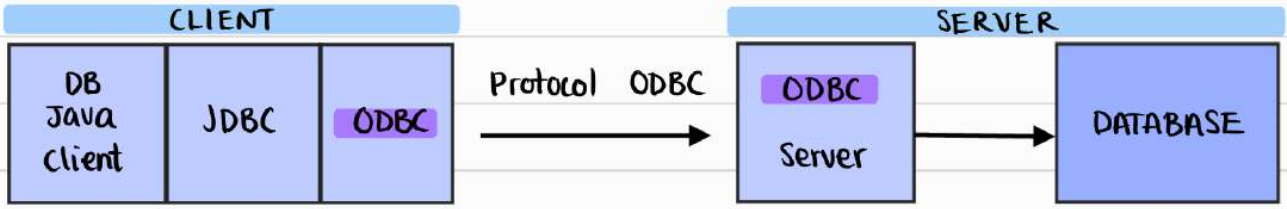
Provide transparency to the developer regarding the RDBMS

- Based on a Driver Manager: interface with every specific DB controller

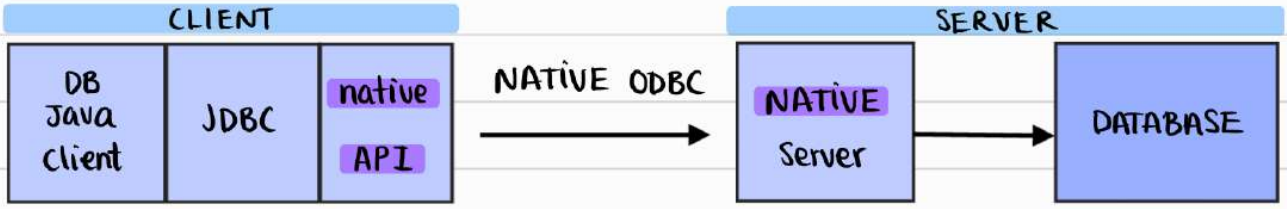


TYPES OF CONTROLLERS

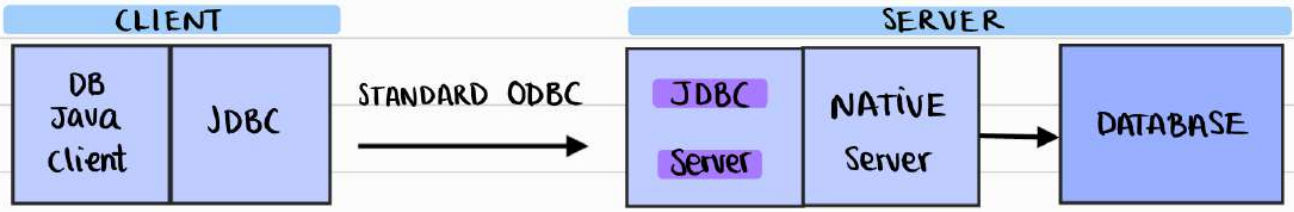
- CONTROL TYPE 1: Bridge JCBC - ODBC



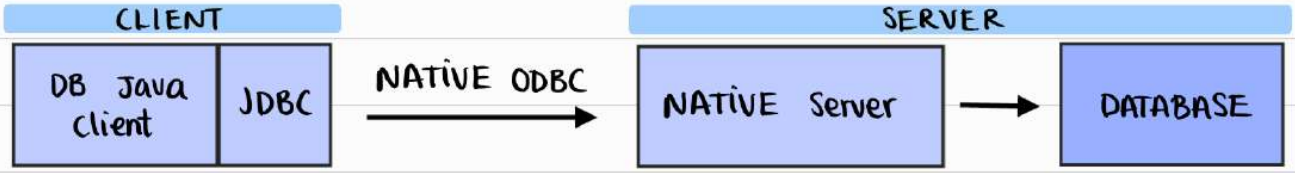
- CONTROL TYPE 2: Controller using native APIs



- CONTROL TYPE 3: Controller JDBC-NET pure Java



- CONTROL TYPE 4: 100% Pure Java



LIFE CYCLE OF A CONNECTION JCBC

1) Register a JDBC driver DriverManager
   Connection and communications with the DB are controlled by a DriverManage object.
   - Register a JDBC driver adapted for the DB target
   (vamos a hablar de este tema) :D

2) Establish connection Connection
3) Create judgements Statement
4) Execute judgements
5) Process the results ResultSet
6) Close connections

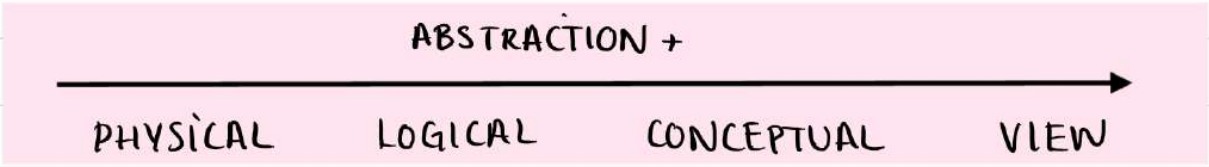
LESSON 2

TARGETS It is necessary to guarantee:

- Safety
- Privacy
- Availabity
- Integrity
- Independence
- Concurrent accesses
- Information redundancy
- Distributed data

TRADITIONAL ADMINISTRATION

1) Analyse and design of the DB.



2) Selection of a suitcase 'manage system' and related hardware.
   - Benchmarks
   - Safety tools
   - Data integrity and availability
   - Technical support
3) Tuning of DB performance: a DB is a dinamic artefact
   - Physical model: storage and accessibility of the data
   - Logical model: modification of the DS
4) Query optimization: + data ⟶ + queries ⟶ - performance
   - Query monitoring
   - Query rewriting
   - Modification / tuning of the physical model
5) Management of the safety, privacy and integrity of the info.
   - Safety: access to the DB
   - Privacy: protection of personal details
   - Integrity: model of reality
6) Design and implemention of safety copies policy.
   - Backup tools
   - Management of the redo log

LESSON 3

The administration of databases needs the knowledge of the structure of DBMS

- Structures of Memory } instance
- Processes
- Logical structure } database
- Physical structure

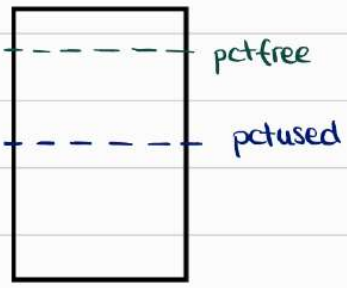
FILES

Software (installed in ORACLE_HOME) - Client
- Server

- **Physical structure: redo log files**
  - Data changes (DDL and DML)
  - The redo log files are organized in groups (at least 2)
    - Inside a group: member
  - Active group: redo log switch
- **Physical structure: data files**
  - Database: distributed logically in tablespaces
  - Tablespace: stored physically in 1+ data files
  - Data files: file of fixed size that stores the DB
- **Logical structure: segments**
  - Extent: contiguous set of data blocks
  - Storage block: unit of storage of the database
  - Free space: consecutive free extensions
  - Segments: set of extensions in a given tablespace
    - Every segment has a set of parameters that controls its storage growth:
      - Initial
      - Next
      - Minextents
      - Maxextents
      - Pctincrease
      - Pct free
      - Pctused
      - Tablespace
    - Types of segments: (a) Data (b) Indexes (c) Rollback (d) Bootstrap (e) Temporal data

- Logical structure: storage blocks

- PCTFREE: space of the block that is kept for future updates of the stored data in the block
- PCTUSED: minimal space of occupation of the block, under this percentage: empty enough to insert new rows



- The information dictionary

View	Object
USER_TABLESPACES	Tablespaces for a given user
DBA_DATA_FILES	Data files where tablespaces are stored
V$DATAFILE_HEADER	Head of the data files
V$LOGFILE, V$LOG	Redo log files
USER_SEGMENTS	Segments
USER_EXTENTS	Extensions
DBA_FREE_SPACE	Free space

LIFE CYCLE OF A CONNECTION TO THE DATABASE

1) Try to connect to the database
2) Corresponding Listener - captures the request
   - a Server process is created
3) The server process verifies in the dictionary the client credential
4) The program client sends a SQL statement to the server process
5) The server process interprets / creates an execution plan and verifies the client.
6) The server process verifies: Data blocks in the data buffer
   ↳ If not: data blocks are loaded fron the data files
7) Changes to the data:
   (1) copy previous information into rollback / space undo segment
   (2) writes the redo and undo entries to the redo log buffer
   (3)
8) Commit: server process writes the commit → redo log buffer
   waits until logw confirms the redo chages

9) The process dbwr (databasewriter) (from time to time) copies:
modified memory data block ⇝ data files
The process ckpt (c.t) synchronizes all the data

10) Redo file is full: logw closes current redo file
archiver creates a copy
logw opens a new one

11) The client disconnets: server process is finished.

EXERCISES

1. In Oracle it is possible to have:
   1. A only database and several instances → "multiple instances of the same database"
   2. Several databases and a only instance
   3. A database replicated in several machines → "standby database: replica, executed in other machine"
   4. The database and the instance in a only machine

2. In Oracle, the groups of files *redolog*:
   1. Among other things, they allow Oracle to continue a strategy of management of the data buffer of type "steal / no force"
   2. There is only a group of *redolog* files for database
   3. There is only a group of active *redolog* files for database → "there is always a only active group"
   4. There is only a group of on-line *redolog* files for database

   ↳ is not is this lesson

3. In Oracle, it is said that the groups of *redolog* files are multiplexed if and only if:
   1. There are several groups of on-line *redolog* files
   2. There is more than one member per group → multiplexed = multiple members
   3. The ARCHIVELOG mode is activated
   4. There are no offline files *redo log*

4. In Oracle, about the *tablespaces*:
   1. They are no part of any instance
   2. They are organized in segments, extensions and storage blocks
   3. They are managed by the *database writter* (dbwr) process → i guess so
   4. They are stored in one or more files → "stored physically in one/several data files"

   segment > extention > storage block → tablespace

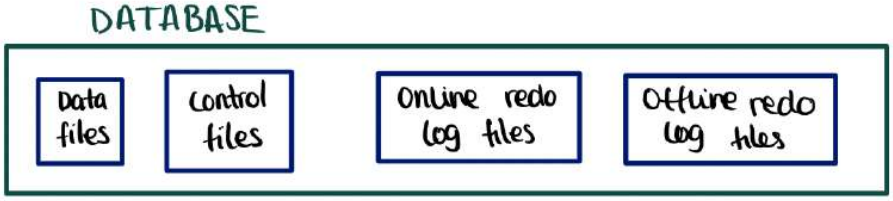
5. In Oracle, a table can be stored physically
   1. In a only segment
   2. In several extensions
   3. In several tablespaces
   4. In a only file

   → collection of extensions = segment

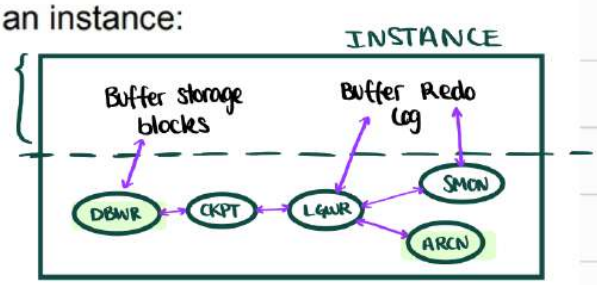
6. In Oracle, about the pctfree parameter:
   1. It must have a high value for tables which are expected to be rarely updated
   2. It is related to the management of the storage blocks
   3. It is relevant for operations of type DELETE → pctused
   4. It is relevant for operations of type UPDATE → free space saved for future updates

7. In Oracle, for every new session client, it is possible to setup the DBMS so that:
   1. A new server process is created → step (2): created by the listener
   2. A new instance is created
   3. A server process is shared with other clients
   4. The instance is shared with other clients

8. In the terminology of Oracle, some of the elements regarding with the database are:
   1. Data files
   2. Instance
   3. *Redo log* files
   4. Control files

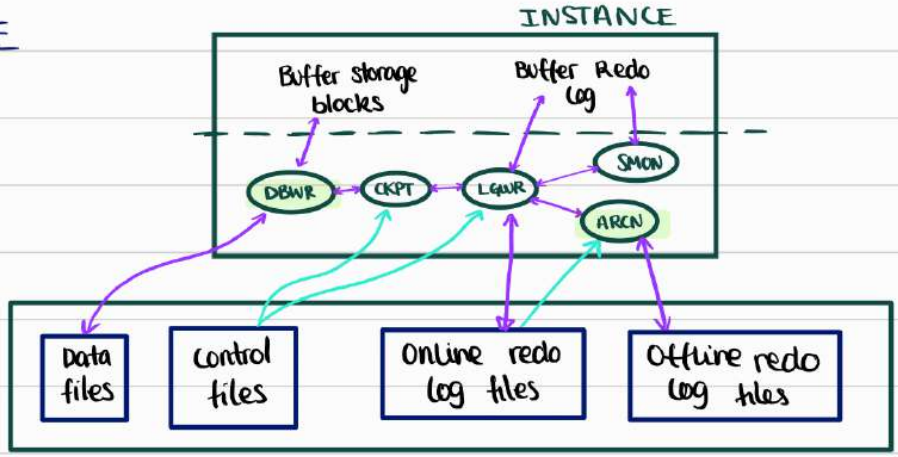


9. In Oracle, the following elements are a part of an instance:
   1. The system global memory *(SGA)*
   2. SQL*Plus
   3. The database writer process (*DBWR*)
   4. The archiver process (*ARCH*)



10. In Oracle, the following elements are a part of the global memory of the system
    1. The process global memory
    2. The buffer of the log
    3. The data buffer
    4. The *tablespace*

ORACLE INSTANCE



LESSON 4: PHYSICAL DESIGN OF DATABASES

PHYSICAL DATA DESIGN

Related to 'traditional administration of DB' task 3 and 4 (lesson 2)

- In contrast to the logical data model
  - MAXIMIZE THE PERFORMANCE: minimize queries executions.
  - DINAMIC DESIGN: evolves along the DB life
- ELEMENTS
  1. Indexes: rapid access to data
     - Improve queries, but worsen updates

     TYPES:
     - one or more rows per value
     - Concatenated
     - Functional
     - Index Organized Tables (IOT)

     USE:
     - ✗ when recovering most of the table
     - ✓ main keys (created by default)
     - ✓ external keys
  2. Partitions (clusters): physical allocation of logical data
     - PARTITIONED TABLES Distribution of the table among different segments.
       - Every segment can be stored in a different tablespace
       - PRINCIPE OF LOCALITY: improves SQL queries performance
       - Execution time of backups is decreased
       - USE: tables of great size (millions of rows)

     TYPES:
     1. Range partitioning
     2. Hash partitioning

     ```
     CREATE TABLE <table_name> (<attribute> NUMBER)
       PARTITION BY HASH (<attribute>) PARTITIONS (n);
     ```

     3. List partitioning

     ```
     CREATE TABLE <table_name> (list of attributes)
       PARTITION BY LIST (<attribute>) (list of partitions);
     ```

4 Composite partitioning: subpartitions

5 Manual partitioning

```
CREATE TABLE <table_name> ( attributes );
    PARTITION BY SYSTEM ( partitions );
INSERT INTO 'PARTITION' (<atribute>) VALUE ( values );
```

6 Interval partitioning

```
CREATE TABLE <table_name> ( list of attributes )
    PARTITION BY RANGE (<attribute>) INTERVAL (...)
    ( PARTITION <code> VALUES LESS/MORE THAN (value));
```

- CLUSTERS

  Groups of tables stored together because they share columns that used together:
  similar rows are stored together → - access time to disc

3 Materialited: join operations improvement: view + data

```
CREATE MATERIALITED VIEW <view_name> [TABLESPACE <name_ts>]
    [BUILD {INMEDIATE | DEFERRED}]
    [REFRESH {ON COMMIT | ON DEMAND | } {COMPLETE | FAST | FORCED} |
            [START WITH <date>] NEXT <date>}
    [{ENABLE | DISABLE} QUERY REWRITE] AS (select ...);
```

- BUILD: 'when' the first load must be done
- REFRESH: 'how' must it be updated (it = materialited view)
- QUERY REWRITE: implicitly uses it in the othe queries execution

- IN THE DATA DICTIONARY:

table	Object
DBA_INDEXES ALL_INDEXES USER_INDEXES	indexes of the database
DBA_IND_COLUMNS ALL_INDEXES USER_INDEXES	Indexed columns in the database
DBA_CLUSTERS ALL_CLUSTERS USER_CLUSTERS	All the clusters Accessible Clusters for every user Property clusters of every user
DBA_CLU_COLUMNS ALL_CLU_COLUMNS USER_CLU_COLUMNS	Column clusters
DBA_PART_TABLES ALL_PART_TABLES USER_PART_TABLES	All the partitioned tables Accessible Partitions for every user Partitions property of every user
USER_MVIEWS	materialized view of every user

`select *`
`from user_objects`
`where object_type='MATERIALIZED VIEW'`
	materialized view of every user

pierdo espacio ⟷

Necesito concentración

ali ali oooh esto con 1 coin me lo quito yo...

WUOLAH

TUNNING THE PHYSICAL DATA DESIGN

Logs of the system: main log is the 'events log'

- STATPACK: obtain snapshots of the DB
- AUTOMATIC WORKLOAD REPOSITORY (AWR): alternative statpack
  - Provides a detailed analysis of the DB performance
- AUTOMATED DATABASE DIAGNOSIS MONITOR (ADDM)
  - System analysis: possible problems, causes and recommendations
  - Internally uses AWR
  - DBMS_ADVISOR is part of it.
- TKPROF: reports - descriptions of the queries

EXERCISES

1. Write a physical data model suitable for tuning the following logical data model and assumptions. Justify your answer

```
create table stu;
cod number (5,0) primary key,
dni varchar2 (9) unique not null,
name varchar2 (20) not null,
surnames varchar2 (40) not null,
e-mail varchar2 (20),
mobile number (9),
postal_address varchar2 (50) not null,
population varchar2 (50) not null
);

CREATE table subject (
 cod number (5,0) primary key,
 name varchar2 (40) not null,
 qualifications varchar2 (40) not null,
  ct number(2,1) not null check (ct>=3 AND ct<=7.5),
  cp number(2,1) not null check (cp>=3 AND cp<=7.5),
 constraint ck_subject (check (ct+cp <=9))
  );

CREATE table stusub(
 cod number (7,0) primary key,
 cod_stu number not null references to,
 cod_sub number not null references subject,
 date dates,
 constraint ck_stusub unique (cod_al, cod_as, it dates)
);
```

```
create table dpt (
 cod number (3,0) primary key,
 name varchar2 (40) unique not null
);
create table professor (
 cod number (4,0) primary key,
 dni varchar2 (9) unique not null,
 name varchar2 (20) not null,
 surnames varchar2 (40) not null,
 e-mail varchar2 (20),
 phone number (9),
 cod_dpt number references dpt
);


create table knowledge_area (
 cod number (4,0) primary key,
 name varchar2 (20) not null,
 cod_dpt number (3,0) references dpt
);

create table prosub (
 cod_pro number (4,0) references professor,
 cod_sub number (5,0) references subject,
 constraint pk_prosub primary key (cod_pro, cod_sub)
);
```

a. The whole credits of the subject are usually used as part of queries

- Index: add a calculated attributed 'credits'
  Create an index associated with the credits

b. The stusub table is able to have several millions of rows but it is usual to query the subjects of a only course

- Partition → useful in huge tables
  → PRINCIPLE OF LOCALITY: improves queries performace

c. The most frequent way to access the `knowledge_area` table is by joining such a table and the `dpt` table.

- Materialized view → improves join operations
  - Also: index in the external key and cluster

d. Every three months, the students office of the University analyses the age, origin and genre of the students and the grades and subjects that they are registered. For example: How many students pass "Management and administration of databases" attending to the city? What are the most popular grades keeping in mind the gender? What grades are reported to achieve most of the fails? etc.

To do what the exercise is saying: table combination.

- Materialized view: join improvement
  - Also: index to the external key

e. Supposing a normal use of the database: what other improvements would you propose?

Generate indexes for every external key.

2. On the physical design of the database:
   a) It is a direct consequence of the logical design of data
   b) Its main aim is to improve the integrity of the data
   c) Its main aim is to improve the performance of the database
   d) Any modification of the physical design of the database bears the modification of the logical data model

3. If it is created an index associated with a column of a certain table
   a) It will improve the access time to this table attending to the index-linked column
   b) It will improve the insertions of new lines in this table
   c) It implies that the column will not be able to store void values
   d) It implies that the column necessary is a primary or foreign key

4. An index-organized table…
   a) stores the whole row in the primary key index
   b) is that whose key is index-linked
   c) optimizes the access to tables that are defined with prime attributes only.
   d) is that whose attributes are prime

5. Regarding clusters defined in the physical design of the database:
   a) The rows of a set of tables that share one or more columns are stored physically together
   b) It optimizes natural join queries attending to the attributes used to define the cluster
   c) It is a type of physical organization of tables where the whole row is stored in the primary key index structure.
   d) A table is possible to belong to several clusters simultaneously

6. Regarding partitions defined in the physical design of the database, it is possible to affirm that:
   a) The rows of a set of tables that share one or more columns are stored physically together → clusters
   b) All partitions of the same table are stored in the same tablespace → segments in different tbs
   c) It distributes a table among several segments. Every segment stores only some of the columns of the table → several physical devices
   d) Its main use is the optimization of updates of data → any queries

LESSON 5: DATABASE SECURITY

OBJECTIVE: control who and how access to the stored data

REQUIREMENTS

- Who can gain access to the DB
- What data can a certain user gain access to
- Access to certain values of a specific data
- What operations can be realized on the data
- Allow statistical queries but no concreta information
- Transferable privileges allowed or not
- Restricted accesses: depends on when and where

MECHANISMS

1) USER IDENTIFICATION

Identify the user in order to know what privileges are granted

- Username + password
- Cards or keys
- Security questions
- Biometric identification

• Determination of the privileges:

- Discretionary Access Control (DAC): user - privilege list
- Mandatory Access Control (MAC): user - priority level

2) PRIVILEGES MANAGEMENT

PRIVILEGE: right to execute certain queries or other user's objects

- Can be assigned to an user or a role
- Can be revoked: all the others that come from the revoked one, must be revoked too.

3) ASSIGMENT AND CONTROL OF THE RESOURCES OF THE SYSTEM

DISC SPACE: main or temporary storge - disc quotas

RESOURCES LIMITS: basic in big multiuser systems (expensive)

- CPU time
- E/S operations
- Connection time
- Concurrent sessions number
- Idle time allowed in a session

4) AUDITS

Maintain a special file where the system registers automatically any interaction of the users with the data stored in the database.

- WHAT TO AUDIT
  - User
  - Date and hour
  - Terminal
  - Location
  - operation
  - Modified objects
- TYPES: sentences, privileges, scheme and fine grain

SECURITY IN SQL: ORACLE

- TYPES OF PRIVILEGES

(1) SYSTEM: things allowed in the data base

```
GRANT system_priv(s) TO {user, | role, |PUBLIC}
[IDENTIFIED BY password] [WITH ADMIN OPTION];
REVOKE privileged...
FROM {user | role | PUBLIC} ...;
```

(2) OBJECT: SQL sentences allowed for a user scheme

```
GRANT privileged [(column [column]...)]...
ON objected
TO {user | role | PUBLIC}...
WITH GRANT OPTION];
```

- REVOKE
  - A privilege is revoked for everyone
  - System ones are NOT eliminated in cascade
  - Object ones are able to be eliminated in cascade
    - They can ONLY be revoked by who granted them

```
REVOKE privileged...
ON objected
FROM {user | role | PUBLIC} ...
[CASCADE CONSTRAINT];
```

- **ROLES : privilege management**

  Group of privileges that are named and can be assign.
  - It is possible to grant an user any role
  - Can have a password
  - The name in the DB is unique
  - Is NOT part of any scheme
  - Modified: privileges modified accordinly
  - It is possible to (de)activate a role for a/some users

```
CREATE ROLE nombre_role [IDENTIFIED BY <CONTRASEÑA> | NOT IDENTIFIED];
```

- **PROFILE: assigment and control of resources of the system**

  Named set where the limits of use for resources are specified
  - Storage and quotas:

```
CRETE USER user
 IDENTIFIED {BY password | EXTERNALLY}
 [DEFAULT TABLESPACE tablespace]
 [TEMPORARY TABLESPACE tablespace]
 [QUOTA {número_entero [K|M] | UNLIMITED} ON tablespace]...
 [PROFILE profile];
```

VIEW

Virtual table that is built by using data from a number of tables or views.

- USE
  - Show only certain attributes
  - Show only certain rows
  - Not allowed access to the real table

View	Object
DBA_PROFILES	Profiles in the database
USER_RESOURCE_LIMITS	Parameters of resources assigned to the user
DBA_ROLES	Existing roles in the database
DBA_USERS ALL_USERS USER_USERS	Users of the database Visible users for the current user It describes the current user
DBA_ROLE_PRIVS	Roles granted to users
DBA_TAB_PRIVS	Permissions on objects of the database
DBA_SYS_PRIVS	Privileges of the system to users and roles
SYSTEM_PRIVILEGE_MAP	List of all the system privileges

EXERCISES

7. The Oracle privileges management :
   1. It stores the moment when a privilege was granted to a user
   2. It is possible that several users grant the same privilege to the same user
   3. If a user A revokes a privilege P to user B, then necessary user B loses the privilege P and, in turn, all those users granted by B with the privilege P
   4. A user can be granted with any number of privileges

8. In Oracle, the privileges type object:
   1. are necessary to create a table
   2. can be revoked exclusively by the same user who transmitted it
   3. it is not possible to eliminate them in cascade
   4. can be assigned to a role

9. In Oracle, the privileges of type system:
   1. are necessary to create a table
   2. can be revoked exclusively by the same user who transmitted it
   3. it is not possible to eliminate them in cascade
   4. can be assigned to a role

10. A Oracle user is able to receive:
    1. several roles and profiles
    2. several roles and a only profile
    3. an only one role and an only profile
    4. an only one role and several profiles

4 any roles // 1 profile

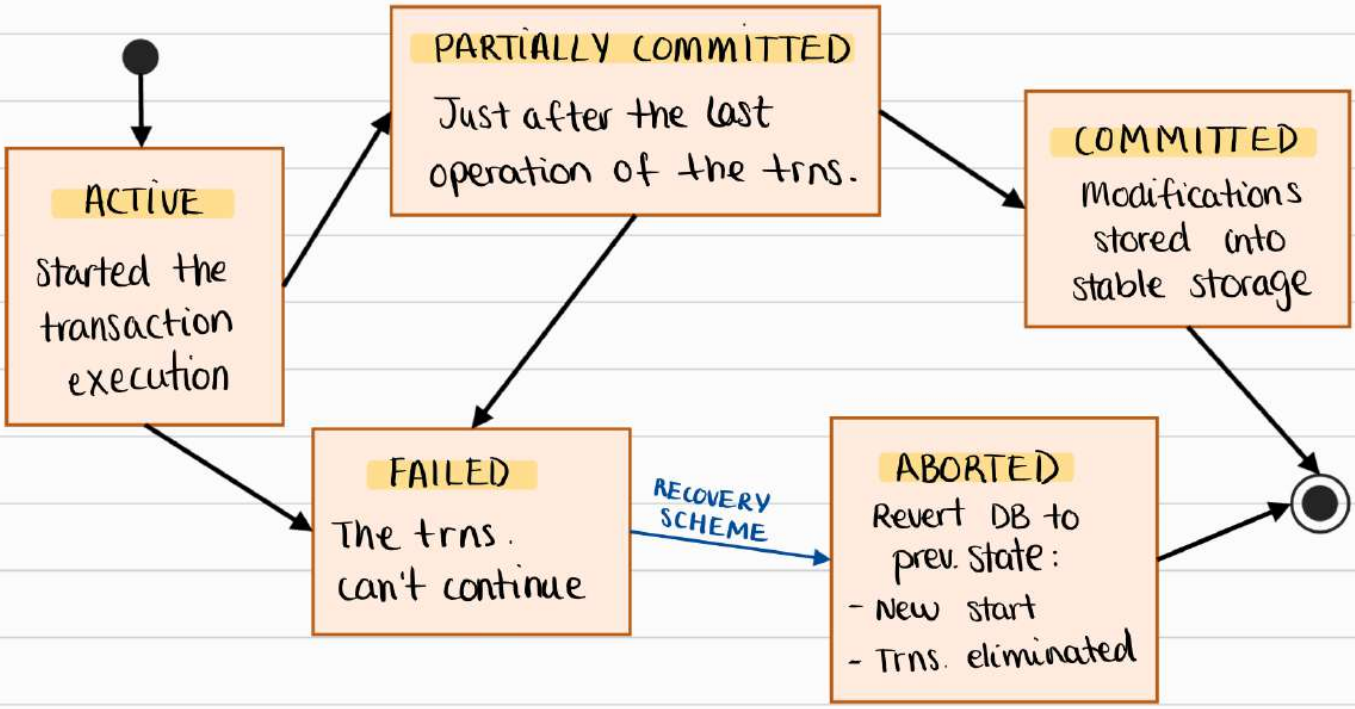
Y

LESSON 6: TRANSACTION MANAGEMENT

TRANSACTION

- Unit program that gains access to the DB and maintains consistency
- Logical unit of database processing
- Logical model of a change of the state in the DB domain model

- **TRANSACTION STATES**



- COMMIT: start transaction
- ROLLBACK: undo transaction
  - It is about 'data' and not its 'definition'
    (example: it does NOT erase table → DROP TABLE, the data inserted in the table is undone, but not the table)

• ACID properties	Atomicity	All transactions be "all or nothing"
	Consistency	Move between valid states
	Isolation	Concurrent execute
	Durability	Once a transaction is committed it will remain

OPERATIONS OF STORAGE

Data transfer between disc and application is carried out by using locks

CONCURRENCE

Several user transactions read/modify the same elements stored in the DB

- Increase productivity
- Increase use of the CPU
- Reduce response time of transactions

• CONFLICTS

Two operations are in conflict if

1. Belong to 2 different transactions
2. Gain access to the same element
3. At least one is 'write'

- Lost update
- Dirty reading
- Incorrect summary
- Not repeatable reading

• SCHEDULE: A schedule P of n concurrent trns. T1, T2... is a sequence of operations done by such transactions. Restrictions:

$\forall$ Transaction$_{(i)} \in P$: Ti operations are included in P in the same order.

1. STRATEGIES BASED ON THE RECOVERABILITY

- NOT RECOVERABLE: commit data read ← not committed transactions
- RECOVERABLE: data read ← not committed transactions
- AVOIDS DELETE ON CASCADE: data read ↚ not committed transactions
- STRICT: data read/write ↚ not committed transactions

2. STRATEGIES BASED ON SERIALIZABILITY

Target of a protocol of control of concurrence

SCHEDULES

- CORRECT: leaves de DB in a consistent state
- SERIAL: one transaction at a time - is trivial and inefficient
- SERIALIZABLE: (non-serial) equivalent result to serial
- EQUIVALENT: keeps the order of every operation
  - CONFLICT EQUIVALECT: keeps the order of conflict operations

A schedule is CONFLICT-SERIALIZABLE if it is conflict equivalent to a serial

- OBJECTIVE: obtain a serializable equivalent schedule

DETECTION OF CONFLICT SERIALIZABILITY

- PRECENCE GRAPH or GRAPH OF SERIALIZABILITY
  - Guided graph: $G = (N, A)$ - N: set of nodes
    - A: set of guided edges

ALGORITHM

1) Create a nodo for every transaction $T_i \in P$
2) Create an arc $T_i \rightarrow T_j$ if:
   a) T_j reads after T_i writes (same element)
   b) T_j writes after T_i reads (same element)
   c) T_j writes after T_i writes (same element)

- ARC $T_i \rightarrow T_j$: T_i must appear before T_j to prevent conflicts

CYCLES:

- If P's graph contains a cycle → is not conflict serializable
- If P's graph does NOT contain cycles → is serializable

APPROACHES OF CONCURRENCY CONTROL

A DBMS cannot anticipate how there will be inserted the transactions' operations

1. LOCKING METHODS

It does NOT guarantee a serializable schedule

- SHARED LOCK: reading permissions `read_lock()`
  - Several transactions gain access to an element simultaneously
- EXCLUSIVE LOCK: reading and writing permissions `write_lock()`
  - One the first transaction to lock it gains access to an element
  - The new value will be visible for other trns. when the lock finishes

RULES OF USE

1. [read_lock or write_lock] before [read]
2. [write_lock] before [write]
3. [unlock] after completing [read or write]
4. While an element is locked: no need other locks
5. No [unlock] unless the was a [lock]

- TWO-PHASE METHOD (2PL):

1) EXPANDING PHASE: acquire locking, cannot release it
2) SHRINKING PHASE: release locking, cannot acquire it

- This method guarantees serializability but → deadlock, indefinite postponement

A - CONSERVATIVE or STATIC: lock all elements in the beginning
B - STRICT: don't release any exclusive locking until finishing
C - RIGOROUS: don't release any locking until finishing

2. DEADLOCK

Two or more transactions are waiting for each other to unlock elements

- APPROACHES

1 DEADLOCK TEMPORIZATIONS: wait = predefined period of time
- If the locking is not granted in this time → 'end of temporization'
  - ↳ the transaction is aborted

2 DEADLOCK PREVENTION: when Ti "produces" a conflict

A - ALGORITHM WAIT-DIE

- If $(M(Ti) < M(Tj))$ then Ti waits
- Else $(M(Ti) > M(Tj))$ then Ti dies

B - ALGORITHM WOUND-WAIT

- If $(M(Ti) < M(Tj))$ then Ti dies
- Else $(M(Ti) > M(Tj))$ then Ti waits

3 DEADLOCK DETENTION:

Periodic cross-check of the system state: check mortal locking

WAIT-FOR GRAPH: shows dependences between transactions

1) Node for every transaction
2) Arc for every locking conflict $Ti \rightarrow Tj$
   - when [unlock] the arc is erased

- It there is a cycle in the graph → deadlock detected

- THE PROBLEM OF THE INDEFINITE POSTPONEMENT

The concurrence-control protocol never selects to a transaction that is waiting for a locking

ALGORITHMS OF PREVENTION
- Strategy FIFO
- Priority increase in the waiting

ORACLE CONCURRENCE

- ISOLATION LEVELS IN SQL-92

1 SERIALIZABLE / READ ONLY: visible data = transaction beginning

- When trying to execute a DML on a row that was modified in a non-committed transaction → the DML operation fails
- READ ONLY is a subtype of SERIALIZABLE

2 READ COMMITTED (detect): visible data = query beginning

- When trying to READ-COMMIT tries to execute DML on a locked data → wait

- SUITABILITY OF SERIALIZABLE TRANSACTIONS
  - Expected data will not be modified
  - Necessity for reading consistency
  - Short transaction
  - CONS - out-of-time data consumption
    - Exception in time of execution
- LOCKINGS
  - Automatic management
    - Exclusive and shared lockings: lock but not modify
    - Table or row(s) block
    - Release at the end of the transaction
  - Manual management
    - Overlaps with the automatic locking
    - LOCK TABLE

EXERCISES

1. Why is concurrence necessary? what would happen if a DBMS did not provide this?

   - To assure the simultaneous transactions execute properly
   - Two simultaneous transactions could modify or obtain dirty data

2. What ACID property is related to concurrent systems?

   Isolation

3. Is always the best option to get serializable transactions? why?

   In theory, it's best to obtain serializable trns but in reality this requires an extra cost.

4. Give an example so that each one of the SQL-92 isolation levels is good enough, but not the previous level.

Nivel de aislamiento	Lectura sucia	Lectura no repetible	Lectura fantasma (*)
READ UNCOMMITED	Sí	Sí	Sí
READ COMMITED	No	Sí	Sí
REPEATABLE READ	No	No	Sí
SERIALIZABLE	No	No	No

Dirty reading: non-committed

1) READ UNCOMMITTED: no concurrency or trns. without shared data
2) READ COMMITTED: the data must be updated
3) REPEATABLE READ: finite tables
4) SERIALIZABLE: bancary transaction

5. Using the UNIVERSITY scheme, write two transactions T1 and T2 by using PL/SQL and simulate an execution plan so that they happen a:
   1. Lost update
   2. Dirty reads
   3. Not repetable reads
   4. Phanton reads

1. LOST UPDATE: 2 updates of the same data → 1st is 'lost'

T1:
```
UPDATE students
   SET age = age + 1;
COMMIT;
```

T2:
```
UPDATE students
   SET age = age - 1;
COMMIT;
```

2. DIRTY READ: 1 read → uncommitted modification

T1:
```
UPDATE students
   SET age = age + 1;
ROLLBACK;
```

T2:
```
SELECT *
  FROM students
```

✗ rollback

3. NOT REPETABLE READS: 2 reads → different results

T1:
```
SELECT age FROM students
```
```
SELECT age FROM students
```
• select

T2:
```
UPDATE students
   SET age = age + 1;
```

4. PHANTOM READS: 2 selects → 2nd gets more data

T1:
```
SELECT * FROM students
```
```
SELECT * FROM students
```
• select

T2:
```
INSERT INTO students;
COMMIT;
```

7. How does ORACLE guarantee that there will never be phanton reads?

```
T1                      T2
READ(X, xi);
xi := xi-N;
WRITE(X, xi);
                        READ(X, xj);
                        xj := xj+M;
                        WRITE(X, xj);
READ(Y, yi);
```

It must guarantee that the data is committed for reading.

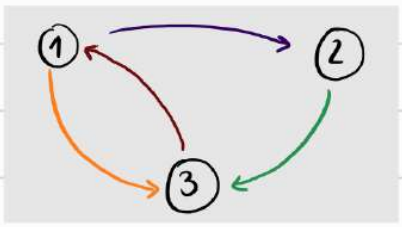
8. A system that guarantees the serializable isolation level, is it posible to happen lost updates?

```
T1                      T2
READ(X, xi);
xi := xi-N;
                        READ(X, xj);
                        xj := xj+M;
WRITE(X, xi);
READ(Y, yi);
                        WRITE(X, xj);
yi := yi+N;
WRITE(Y, yi);
```

No, because 'serializable' assures that the schedule is equivalent to a serial one: 1 transaction then the other

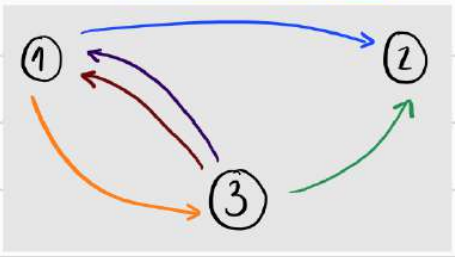
9. Which of the following schedule is serializable? Demonstrate it using graphs of precedences. For those that are serializables: What would be the equivalent serie schedule?

a. $r_1(X); r_3(X); w_1(X); r_2(X); w_3(X);$



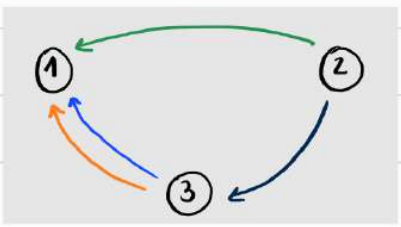
Not serializable

b. $r_1(X); r_3(X); w_3(X); w_1(X); r_2(X);$



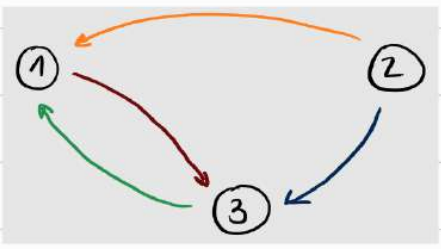
Not serializable

c. $r_3(X); r_2(X); w_3(X); r_1(X); w_1(X);$



Serializable

d. $r_3(X); r_2(X); r_1(X); w_3(X); w_1(X);$



Not serializable

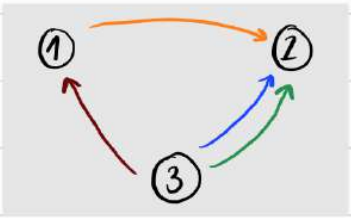
10. T1, T2 and T3 are transactions. Which of the following schedules is serializable? Demonstrate it using graphs of precedences. For those that are serializables: What would be the equivalent serie schedule?

T1: $r_1(X)$; $r_1(Z)$; $w_1(X)$;

T2: $r_2(Z)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;

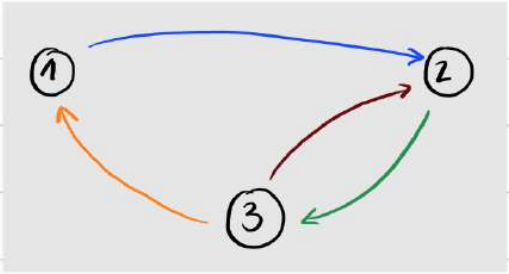
T3: $r_3(X)$; $r_3(Y)$; $w_3(Y)$;

a. $r_1(X)$; $r_2(Z)$; $r_1(Z)$;$r_3(X)$;$r_3(Y)$; $w_1(X)$; $w_3(Y)$; $r_2(Y)$; $w_2(Z)$; $w_2(Y)$;



→ Serializable

b. $r_1(X)$; $r_2(Z)$; $r_3(X)$; $r_1(Z)$; $r_2(Y)$; $r_3(Y)$; $w_1(X)$; $w_2(Z)$; $w_3(Y)$; $w_2(Y)$;



→ Not serializable

11. Create a schedule for exercise 10 by following the 2PL strict protocol

a transaction doesn't release a exclusive locking element until it finishes

T1	T2	T3
READ_LOCK (X)		
READ (X)		
READ_LOCK (Z)		
READ (Z)		
	READ_LOCK (Z)	
	READ (Z)	
		READ_LOCK (X)
		READ (X)
WRITE_LOCK (X)		
WAIT-DIE: WAIT()		
	READ_LOCK (Y)	
	READ (Y)	
		READ_LOCK (Y)
		READ (Y)
		WRITE_LOCK (Y)
		ABORT(): UNLOCK (X, Y)
WRITE_LOCK (X)		
WRITE (X)		
UNLOCK (X, Z)		
	WRITE_LOCK (Z)	
	WRITE (Z)	
		READ_LOCK (X)
		READ (X)
	WRITE_LOCK (Y)	
	WRITE (Y)	
	UNLOCK (Y, Z)	
		(rest of T3)

12. Given the following code SQL: what will return the sentence *select* of the row 8 (transaction T2)

T1	T2
1 create table to (x1 number, x2 number);	
2 insert into to values (1, 10); x1=1, x2=10	
3 commit;	
4	
5	set transaction isolation level serializable;
6 update x2=20 where x1=1; x1=1, x2=20	x2=10 select x2 from to where x1=1;
7 commit;	
8	x2=20 select x2 from to where x1=1;

a. 10
b. 20
c. an exception will happen in T1
d. an exception will happen in T2

13. Given two transactions T1 and T2 and the below execution plan, we can affirm

T1	T2
1 READ(X, xi);	
2 x_i := x_i-N;	
3 WRITE(X, xi);	
4	READ(X, xj);
5	x_j := x_j+M;
6	WRITE(X, xj);
7 READ(Y, yi);	
(an exception is generated in T1)	

a. At a level of isolation READ UNCOMMITED a dirty reads would happen
b. At a level of isolation READ UNCOMMITED a phanton reads would happen
c. At a level of isolation READ COMMITED a dirty reads would happen
d. At a level of isolation READ COMMITED a phanton reads would happen

14. In Oracle, a serializable transaction:

a. It can generate an exception if it tries to modify a data that has been modified in another transaction T2, after beginning T1
b. It can read not updated data
c. It is the minimal level of necessary isolation to guarantee that there will not be dirty reads
d. It is the minimal level of necessary isolation to guarantee that there will not be phantom reads

5. (1 puntos) Tomando como punto de partida la planificación P1 del ejercicio anterior, escribe una planificación no serie que use B2F riguroso con las transacciones T1, T2 y T3 del ejercicio anterior. Si aplicando P1 llegas a una situación de interbloqueo, aplica el algoritmo esperar-morir y continúa con la planificación hasta que concluyan las tres transacciones.

T1: ~~$r_1(X)$~~; $r_1(Z)$; $w_1(X)$;
T2: ~~$r_2(Z)$~~; ~~$r_2(Y)$~~; $w_2(Z)$; $w_2(Y)$;
T3: $r_3(X)$; $r_3(Y)$; $w_3(Y)$;

Rigorous: doesn't release any locking until the end
· wait-die