Tutorial 1: Introduction to advanced databases

Advanced DataBases (NDS/AI)

Exercise 1

Write true or false (T/F) next to each statement — Briefly justify your answer.

#	Statement	T/F	Justification
1	The schema changes often, the instance is stable.		
2	A primary key can be null.		
3	DDL is used for the structure, DML for the data.		
4	ACID concerns the reliability of transactions.		
5	The E/R model is identical to the relational model.		
6	In ANSI/SPARC, the external level refers to user views.		
7	MVCC increases the parallelism of reads.		
8	NoSQL databases always replace RDBMSs.		
9	An FK must reference an existing PK.		
10	An index changes the result of a SELECT.		

Exercise 2

Match each No. with each Letter.

No.	Abbreviation / Concept	Letter	Definition
1	DDL	M	Table of the relational model (set of tuples and attributes).
2	DML	B	User view (representations tailored to needs).
3	DCL	P	Descriptive property of an entity/relationship.
4	TCL	G	Software managing storage, integrity, access and optimization (DBMS).
5	Referential integrity	C	Manipulation queries: SELECT, INSERT, UPDATE, DELETE.
6	External level (ANSI/SPARC)	J	Structure definition: CREATE, ALTER, DROP.
7	Candidate key	D	Schema / Instance: stable structural plan vs data at a given moment.
8	PK (Primary Key)	H	Conceptual link between entities (Entity–Relationship model).
9	FK (Foreign Key)	A	Uniqueness constraint identifying each tuple.
10	Schema / Instance	N	Transaction control: COMMIT, ROLLBACK, SAVEPOINT.
11	ACID	E	Access commands: GRANT, REVOKE.
12	Entity	O	Mandatory reference to an existing key in another table.
13	Attribute	L	Set of columns that can serve as a primary key.
14	Relation	F	Reliability properties of transactions (ACID).

© 2026 Mr Laïdi FOUGHALI — www.al-moualime.com/adb — CC BY-NC-SA 4.0

15	Relationship (E/R model)	I	Constraint pointing to an existing key in another table.
16	DBMS	K	Real-world object/thing being modeled.

Exercise 3

Data integrity

Consider the schema of the following fictitious database, "Bank account management":

```
BRANCH(CODE, ADDRESS)
ACCOUNT(CODE, NUMBER, CLIENT, ADDRESS)
OPERATION(CODE, NUMBER, OP_TIMESTAMP, OPERATION, AMOUNT, BRANCH_BALANCE,
    CLIENT_BALANCE)
```

Sample instance (to give meaning to the schema)

CODE (BRANCH)	ADDRESS
2101	CENTRE VILLE SKIKDA
2512	CONSTANTINE CENTRE

CODE	NUMBER	CLIENT	ADDRESS
2512	2426136-90	CHIEKH NASS MLAH	4, rue Saint Jean, 25000 Constantine
2101	2425368-11	TAHAR OULD TAHAR	2, rue Allées, 21000 Skikda
2512	5252636-20	ALI BEN ALI	Route Bab El Oued n°5, 25200 Belvue

CODE	NUMBER	OP_TIMESTAMP	OPERATION	AMOUNT	BRANCH_BAL.	CLIENT_BAL.
2512	2426136-90	SYS_TS	DEPOSIT	15000.00	?	?
2101	2425368-11	SYS_TS	DEPOSIT	10000.00	?	?
2512	2426136-90	SYS_TS	WITHDRAWAL	5000.00	?	?
2512	5252636-20	SYS_TS	DEPOSIT	12000.00	?	?
2512	2426136-90	SYS_TS	WITHDRAWAL	12000.00	?	?
2512	5252636-20	SYS_TS	DEPOSIT	12000.00	?	?

(OP_TIMESTAMP takes the system Timestamp value)

We want to create this database; beforehand, we will devise the set of **structural constraints** that would guarantee the integrity of its data.

1) Domain constraints (`DOMAIN`)

Propose `DOMAIN`s (or equivalent types) for:

- **DCODE**: branch code in the format AABB, where AA is the wilaya (province) number (e.g. 21 for Skikda, 25 for Constantine) and BB ∈ [01..99].
  (Specify: pattern/check, NOT NULL if relevant, possible collation.)
- **DTEXT**: alphanumeric strings (≈ 60 characters max) for CLIENT / ADDRESS.
  (Specify: length, UPPER/TRIM if needed, NOT NULL if relevant.)
- **DNUMBER**: account number in the format CCCCCCC-CC (C = digit).
  (Specify: pattern/length, NOT NULL if relevant.)
- **DMONETARY**: amounts / balances (suitable numeric type).
  (Specify: bounds, CHECK (value ≥ 0), NOT NULL if relevant.)
- **DOPERATION**: controlled values 'DEPOSIT' or 'WITHDRAWAL'.
  (Specify: set check, case, NOT NULL.)
- **DTIMESTAMP**: system timestamp (type + appropriate DEFAULT).
  (Specify: NOT NULL if relevant; future date forbidden?)

2) Structural constraints (schema)

Declare and **justify**: PK, FK, UNIQUE, NOT NULL, CHECK, DEFAULT (and the referential actions `ON DELETE`/`ON UPDATE`).

- **BRANCH**: `CODE` PK of type DCODE; `ADDRESS` DTEXT, NOT NULL.
- **ACCOUNT**: PK = (CODE, NUMBER); `CODE` FK → BRANCH(CODE) (specify the referential action and justify it); `CLIENT`, `ADDRESS` DTEXT, NOT NULL; (optional) UNIQUE(NUMBER) if global uniqueness is assumed — to be justified.
- **OPERATION**: FK (CODE, NUMBER) → ACCOUNT(CODE, NUMBER); `CODE` FK → BRANCH(CODE) (branch consistency); `OP_TIMESTAMP` DTIMESTAMP + system DEFAULT; `OPERATION` DOPERATION (CHECK on values); `AMOUNT`, `BRANCH_BALANCE`, `CLIENT_BALANCE` DMONETARY, NOT NULL, CHECK (≥ 0).

3) Consistency CHECKs (tuple level) — OPERATION table

Propose two CHECKs (a specification is sufficient):

1. Date: future timestamp forbidden (and/or inconsistent format to be blocked).
2. Amount/Type: if `OPERATION = 'WITHDRAWAL'` then `AMOUNT > 0`.

Explain in one sentence why, for a nullable column, a CHECK must tolerate NULL (pattern of the form `VALUE IS NULL OR ...`).

4) Non-structural constraints ("business" logic)

List 3 to 5 relevant business rules that are not purely structural and explain how to enforce them (e.g. BEFORE/AFTER triggers, secure views, procedures). Possible examples: daily withdrawal limit per account; balance never negative after a withdrawal; branch consistency (`OPERATION.CODE` must equal the CODE of the referenced account); detection of duplicate "instantaneous" operations (same account + same rounded timestamp).

© 2026 Mr Laïdi FOUGHALI — www.al-moualime.com/adb — CC BY-NC-SA 4.0