Page No.
Date

Unit 3 -
Relational Database Design.

- Relational Database → collection of tables having unique names.

* CODD'S Rules -

- Rule 0 → Foundational Rule
  - any RDBMS → manage database → relational capabilities.
- Rule 1 → The Information Rule
  - Info in RDBMS → represent logically only storing values in table.
- Rule 2 → Guaranteed Access Rule
  - every item → RDBMS → logically accessible ↔ table name, primary key, column name
- Rule 3 → Systematic Treatment of Null values rule
  - never leave it empty instead give dummy value (or lead to difficulty during data cleaning)
  - supported in RDBMS → missing info & inapplicable info → syst way
- Rule 4 → Dynamic Online Catalog Based on Relational Model.
  - Database Dictionary → catalog (structure descrip. of Database & stored entries)
  - catalog → same rules as rest database.
    → same query language.

Page No.
Date

- Rule 5 → Comprehensive Data Sublanguage Rule.
  - RDBMS supports → multiple languages
  - At least one language → statements are expressible, per some well-defined syntax.
- Rule 6 → View Updating Rules
  - all views theoretically updated → updated by system.
- Rule 7 → Relational level oper.
  - high level insert, update & delete.
- Rule 8 → Physical Data Independance.
  - shouldn't affect logical layer & users.
- Rule 9 → Logical Data Independance.
  - appln prog & terminal activites → logically unimpaired.
- Rule 10 → Integrity Independence.
  - all integrity constraints → language will be allowing & not the appln program.

Page No.
Date

- Rule 11 – Distribution Independance
  - User → shouldn't know where data is actually from where distributed.
    - (should think) → data is centeralized - (one site only).
- Rule 12 – Non-subversion Rule

high level (multiple records at time) → low level (single record at time). → while converting these shouldn't be allowance of breaking integrity rules & constraints.

- Keys -
- Super Key (SK) –

→ set of one or more attributes within a table that can uniquely identify each record within table.

- Candidate Key –

TRU NOTE®

Keys

Candidate Key = Super Key − Primary Key.

Superkey (SK)	Candidate Key (CK)	Primary Key (PK)	Foreign Key (FK)	Alternate Key (AK)
set of one or more attributes within a table that uniquely identify each record within table.	• Is sub set of superset • single attribute or minimal comb. of attributes that uniquely identify each record in table.	• candidate key chosen by database designer to identify tuple in relation uniquely.	• single attribute or collection of attributes in one table refers to primary key of another table.	• Is candidate key not chosen by database designer to uniquely identify tuples.

• every candidate key is superkey but every superkey is not candidate Key.

Student (parent table)

Reg No	Roll No	Phone	Name	Marks
R101	001	1111111	AAA	88
R102	002	22222222	BBB	83
R103	003	3333333	CCC	93
R104	004	44444444	DDD	67

Super Key → Reg No, Roll No., Phone, Name.
Candidate Key ⇒ Reg No, Roll No & Phone.
Primary Key → Reg No
Alternate Key → Roll No & Phone.
Foreign Key → Reg No (child table).

Course (child table):

CourseID	Course Name	Reg No
C111	Comp Sci	R103
C112	Electrical	R101
C113	~~Mech~~	R104
C114	Civil.	R102

eg
Roll
001
002
003
004

Page No.
Date

* Constrains -

1) Domain Constraint

- defines domain or set of values for attribute
- data types
  - string
  - character
  - integer
  - time, date, currency, etc.

2) Key Constraint or NULL constraint.

- Key → identify record from table
- Primary key → identify record uniquely.
- All values of PK → unique & can't be NULL

eg →

RollNo	Name	Marks	Phone
001	AAA	88	111111
001	BBB	83	2222222
003	CCC	98	3333333
004	DDD	67	4444444

→ Duplicate values? Not allowed!!!

eg

RollNo	Name	Marks	Phone
001	AAA	88	111111
002	BBB	83	2222222
003	1234	98	33333333
004	DDD	67	44444444

→ name can't be numeric value must be string

RollNo	Name	Marks	Phone
001	AAA	88	111111
001	BBB	83	2222222
003	CCC	98	333333
	DDD	67	444444

↓ NULL is not allowed.

3) Integrity constraint.

- rules applied on database to ensure validity of data
  - entity integ.
    - ↓ primary key value can't be NULL
  - Referential integ.
    - ↓ whenever a foreign key is used it must refer a valid existing primary key in parent table.

4) Enterprise Constraints -

↳ semantic constraints.

- additional rules by users or admin.
- based on multiple tables

TRU NOTE®

Page No.
Date

Database Design –

1) Minimum Redundancy

disadv → wastage of storage space.
↳ Leads to Insert, Update & Update Anomalies.

	C1	C2	C3	
r1	X	Y	Z	→ Redundancy of data
r2	X	Y	Z	
r3	P	Q	R	~~wastage~~
r4	M	S	K	

2) Lesser Null values in Tuples.

disadv → wastage of storage space.
↳ difficult to interpret Null values
↳ Difficult to apply aggregate func^{n}.

Name	Age	Contact No
Jack	17	5433345
Harry	24	~~8907897~~
John	21	NVL

3) allow us to retrieve data easily.

Data redundancy & Update Anomalies.

Data redundancy → Cond^{n} in database.
same piece of data → two different places.

Problems caused by Redundancy –

Redundant Storage – info stored repedetly.
Update Anomalies – one copy update, rest aren't → anomaly
Insertion Anomalies – insertion of new record repeated info ⇨ relation schema
Deletion Anomalies – lose imp info → deletion of particular record.

Page No.
Date

* Normalization.

↳ technique to removes or reduce redundancy from a table.

reorganizing data in database → 2 basic req.

1) No redundancy of data. (all stored in one place)
2) Data dependencies are logical. (all related data items are stored together)

Need →

1) eliminate redundant data.
2) reduce chances of data errors.
3) database to → take less disk space.
4) Increase performance.
5) data integrity & consistency.

* 1NF (1st Normal form).

EF Codd
↳ Father of DBMS.

- Table shouldn't contain any multivalued attribute.

RollNo	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS

1NF →

Form (1)

RollNo.	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Onkar	C
3	Onkar	DBMS.

RollNo + Course → composite Primary Key.

Form 2

Roll No. (Primary Key)	Name	Course 1	Course 2.
1	Sai	C	C++
2	Harsh	Java	NULL
3	Onkar	C	DBMS.

Page No.
Date

Form 3.

Roll No	Name
1	Sai
2	Harsh
3	Onkar

Roll No.	Course.
1	C
1	C++
2	Java
3	C
3	DBMS.

Primary Key → Roll No, Course
foreign Key → Roll No.

* Second Normal Form (2^{nd} NF).

- should be in 1NF.
- shouldn't have partial functional dependancy.
- all non-prime attributes should be fully functional dependant on candidate key

CustomerID	StoreID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai

Page No.
Date

Candidate Key → Customer ID, StoreID.
Prime Attributes → Custome ID, Store ID.
Non Prime → Location.

Store ID	Location
1	Delhi
2	Banglore
3	Mumbai

Customer ID	store ID.
1	1
1	3
2	1
3	2
4	3

* Functional Dependancy -

Determinant (X) → (Y) dependant attribute.

X Determines Y
or
Y is determined by X.

✓ Sid → Sname (Valid) (Sid)

Sid	Sname
1	Ranjit
2	Ranjit.

diff person same name.

✓ Sid → Sname (valid)

Sid	Sname
1	Ranjit
1	Ranjit.

same person data redundancy.

✓ Sid → Sname. (Valid)

Sid	Sname
1	Ranjit
2.	Varun.

2 diff student.

✓ Sid → Sname. (Invalid) same sid

Sid	Sname
1	Ranjit
1	Varun

Page No.
Date

* Third Normal Form (3rd NF)

- should be in 2NF. (don't have partial functional dependency)
- doesn't have transitive dependancy.

functional dependancy → indirectly formed by two func" dependancies

Roll No	State	city
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

CK = {Roll No}
FD → Roll No → State.
PA = {Roll No} State → city.
NPA = {State, city}.

* BCNF → Boyce Codd Normal Form).

→ higher version of Third Normal Form.
→ deals with anomly that is not handled by 3NF.

3NF Table → no mutiple overlapping candidate keys
↓
BCNF.

Cond" → R → 3rd F
→ fun" dependancy → X → super key.
Y → prime attribute X → non prime.

NF	2 NF	3NF	BCNF	4th NF	5NF.
multivalued attribute ly. gle value	• In 1st NF. + • No partial dependency • only full dependency	In 2NF + No transitive dependency • No Non-Prime should determine non-prime	• In 3rd NF +LHS must be CK or SK	* In BCNF + No multivalued dependency X →→ Y	In 4th NF. + lossless decomposition.

3NF

→ Third Normal Form.

→ • In 2NF.
• fun^n dependency → X → Y.
• X → superkey
• Y → prime attribute.

→ 3NF → without sacrificing all dependencies

→ 3NF → without losing any info from old tables.

BCNF.

→ Boyce Codd Normal form.

→ BCNF → 3NF
X → Y
X ~~should be~~ superkey

→ Dependencies may not be preserved in BCNF.

→ lossless decomposition. hard to obtain in BCNF

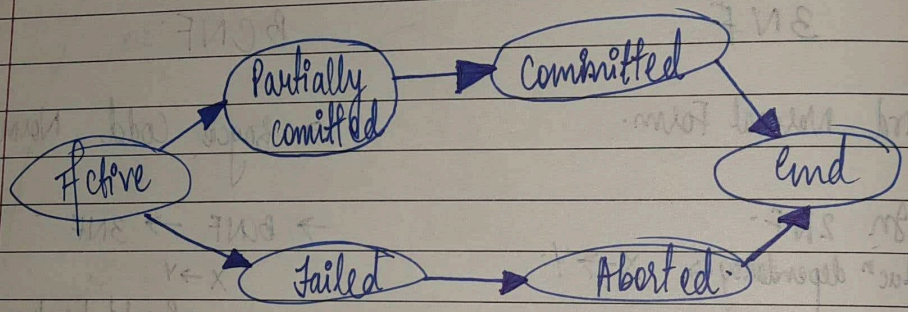
→ BCNF → may loose some information from old table.

Unit-4

- Database Transaction Management.

- Transaction - group of tasks from single logical unit.

- Transaction States -



1) Active - First state of transaction.
   eg → insert, update or delete → done here
   saving data → not done here.
   • transaction → stays in this state → executing.

2) Partially committed - final statement → executed.

3) Failed - checks made by database fails.
   no longer proceed further.

4) Aborted - failed to execute → database recovery system → previous consistent state
   if not → database to consistent state
   ↳ by aborting or rolling back transaction

5) Commited -

- transcn executes all operations → commited.
- last step of transaction → executed without fail.

*. ACID properties -

1) Atomicity → each transcn → single unit → fully completed or not at all completed
→ No transn → is left half completed
→ Database → either before transcn execution of after execu
→ not in executing state.
eg- withdrawal of money. (all five steps or none of step).

2) Consistency → database → consistent state → after transaction
eg- ATM withdrawal opern → balance update → consistent state

3) Isolation → more than one transaction → executed parallel
→ all step executed → its the only transcn in syst
→ No transcn → affect existence of other transc
eg → bank manager → checking acc balance → customer
→ balance befor withdrawal or after.

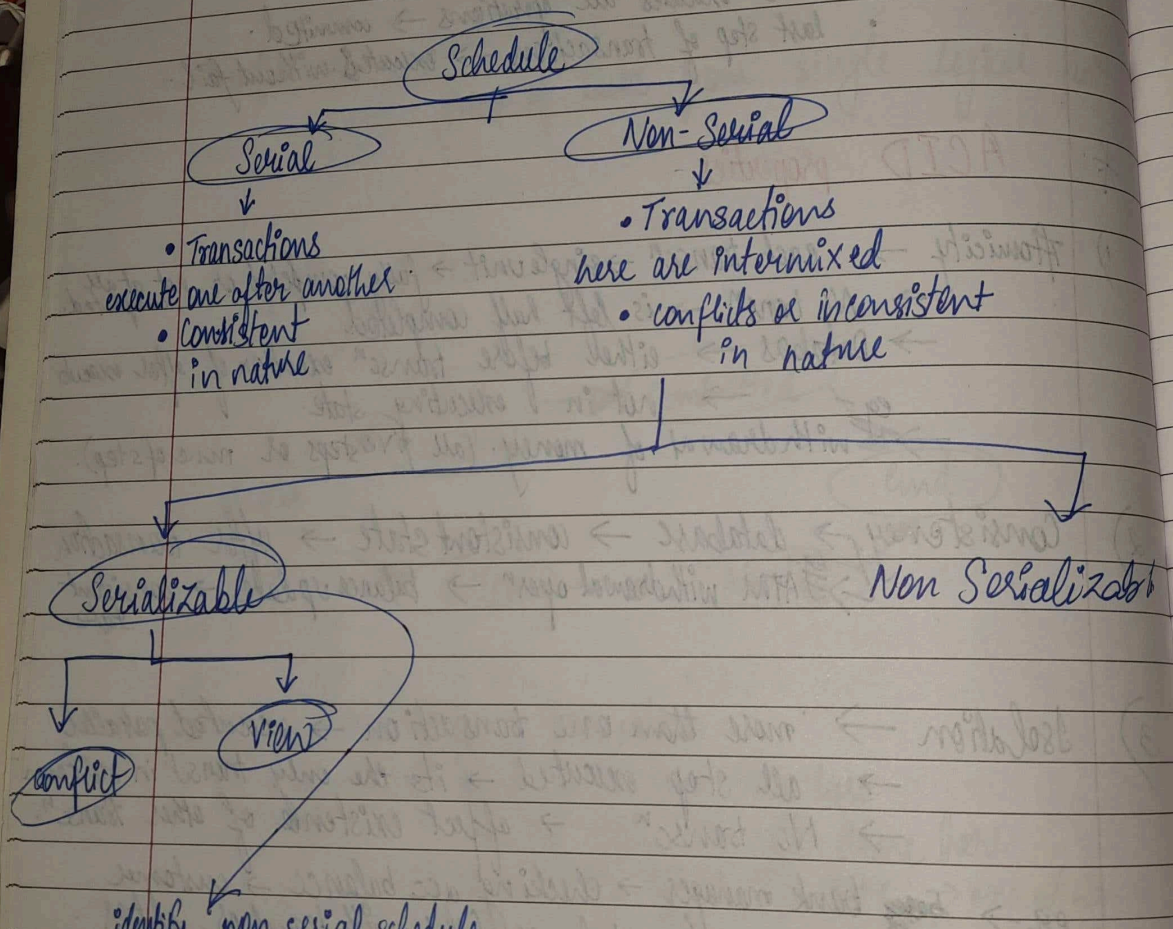
4) Durability - database strong enough → system faliure
- Insert / update → handle / commit to database
- faliure → recover → consistent state.
log of each transaction & its faliure

eg → atm withdrawal.

TRU NOTE®

Page No.
Date

Concept of Schedule



identify non serial schedule & find transaction equivalent to serial schedule.

Page No.
Date

* Serializability → helps to identify which non serial schedule & find transaction equivalent to serial schedule.
  - multiple transc^n → run concurrently
    → leads to inconsistency of data

  - conflict
    - should be diff transactions.
    - oper^n performed on same data items.
  - View
    - every view is not conflict serializability
    - complex to test view serializing.

* Concurrency Control
  → a mechanism that ensures that simultaneous execution of more than one transactions does not lead to any database inconsistency.

Concurrency Control → lock based protocol
deadlock handling
multiple granularity
timestamp based protocol
validation based protocol.

Need of concurrency → ensure isolation
resolve read - write or write-writ
To preserve consistency of databa

* Types of problem in concurrency -

1) Dirty Read / Uncommited Read Problem
2) Incorrect Summary
3) Lost Update
4) Unrepeatable read
5) Phantom Read.

1) Lost Update Problem -

problem occurs when two transactions that access the same database items operations interleaved → value of database incorrect

2) Dirty Read or Uncommited Read Problem -

transⁿ reads data immediately after write operation of previous transaction.

3) Non-repeatable read problem -
Inconsistent analysis problem.

particular transaction sees two different values for same row within its lifetime.

7-9
8-10
9-11
10-12.

Page No.
Date

4) Phantom read protocol -
special case of non repeatable read problem.

one transaction makes changes in database system & due to these changes another transaction can't read data item.

* Lock Based Protocol -
- isolation property in transactions.
- allow a transaction to access a data item → only if it currently holding a lock on item

→ exclusive use of locks on data item for current transaction.

→ shared lock (Read lock)
↳ read data item only
↳ denoted by lock-S.

→ exclusive lock (write lock)
↳ used for both read & write operations.
↳ denoted by lock-X

Rule of thumb
↳ Any number of transactions can hold shared lock on item.
↳ But exclusive lock can only have one transaction.

Page No.
Date

* Two phase locking protocol -

Growing phase (Locking phase) -
- transaction may obtain locks → doesn't release lock
- locks acquired → no locks released

Shrinking phase (unlocking phase) -
- acquired ~~[illegible]~~ lock → doesn't obtain any new lock.
- locked release → no locks acquired.

lock point → last lock posⁿ or first lock posⁿ.

Rule → 2PL → All locks operations precede all the unlock operation.

Adv → always ensures serializability

Disadv → May not be free from irrecoverability
not free from deadlocks
not free from starvation
not free from cascading rollback.

Types of 2PL → Strict 2PL → all exclusive mode locks be held until the transaction commit.
↓ (ie)
exclusive locks (unlocked)
↓
~~transaction~~ is commit

Rigorous 2PL
↓
locks are to be held until transⁿ comits
↓
transaction → serialized in order which they commit.

Page No.
Date

* Timestamp Ordering Protocol-

- order of transaction is decided in adv based on their timestamp.
- schedules are serialized acc to timestamps
- unique value assign to every transⁿ.
- Tells order (when they enter into system).

Read_TS(RTS) = Last (latest) transaction no. which performed read successfully

Write_TS (WTS) = Last (latest) transⁿ no. which performed write successfully.

10:00	10:10	10:15	TS(Tᵢ)
T_1	T_2	T_3	
100	200	300	
older		youngest	

Adv → Schedules are serializable.
→ No waiting for transaction & no deadlock situation.

Disadv → Schedules are not recoverable once transⁿ occurs.
→ same transⁿ may continously abort or restart.

* Deadlock -

Two trans^n depend on each other for something.

four cond^n for deadlock to occur.

- Mutual exclusion cond^n :-
  atleast one resource can't be used by multiple process at time
- Hold & wait cond^n -
  a process holding resource can request for add^n resources that are held by other process in system.
- No preemption cod^n - a resource → cant be forcibly taken by process
  ↑ only process can release resource held by it.
- Circular wait cond^n -
  one process is waiting for resource held by 2nd process & second is waiting for 3rd process & so on & last process waiting for 1st process → CHAIN.

Deadlock can be handled by

1) Deadlock prevention
   1) Wait & Die → if resource held by other
      ↳ DBMS checks timestamp of both trans^s & allows older to wait.
      timestamp
   2) Wound & Wait
      ↓
      older trans^n requests resource
      ↓
      which is held by younger
      ↳ then older forces younger to kill transc^n & release resources
      ↳ younger restarts trans^n with same timestamp
2) Deadlock detection
   ↓
   examines state of system
   invoked periodically
   to determine if it occured
   or no
) wait for graph. → Graph → circle or loop → deadlock present.

Page No.
Date

* Recovery Methods -

To bring back database into last consistent stage prior to occurrence of failure.

a) Shadow paging

- recovery scheme is consired to be made up of no. of fixed size disk pages
- If transactions execute serially
- maintain two page tables during transaction.
- improvement on shadow copy technique.
- fewer disk access than log based
- shadow page table & current page table.
- when transⁿ starts both are identical.
- shadow page table → never changed
  current page table → makes all changes.
- database befor failure → shadow directory

b) log based recovery

- most widely used structure for recording db modification is LOG.

Log → sequence of log records & maintain history of all activities in database.

operations → <start>
<update>
<commit>
<Abort>.

- log maintained → stable storage
- entries in log file → updating (before) physical database

transⁿ performs write
↳ log record should be write be created before DB is modified.

log based recovery technique

Deffered DB modification

- easy to implement only few operations needed.
  redo oper^n → sys failure (required)
- write oper^n → new value
- deffering execution of all write oper^n
  ↓
  transc^n partially commits
- changes of db not updated immediately
- write lock → held longer
- long transc^n → ↑more memory
  ↓
  log & local varibles could be high.

Immediate DB modification

- Redo & Undo → system failure (required)
- write oper^n → new & old value
- allows modification to be output of database transc^n → active.
- changes of db updation are under High concurrency.
- write locks → released after modification.
- manage with less space.

115- NO SQL Databases

Page No.
Date

* CAP theorem -

C A P → Partial tolerance.
↓ ↓
consistency Availability

• Brewer's theorem

Three components -

• Consistency → all reads receive the most recent write or an error.
• Availability → All reads contain data, but it might not be most recent
• Partial tolerance → sys continues to operate despite network failures.

Theorem states
↳ It is not possible to guarantee all three of desirable properties at same time in distributed system with data replication.

* Structured Data	Semi-Structured data	Unstructured data
• fixed & organized data	• combo of structured & unstructured data.	• not predefined or not organized data
• schema dependent & less flexible.	• more flexible than structured. ~~less than → unstructured.~~	• most flexible data
• SQL ⇄ access data present in schema	• tags & elements → access data	• only ~~textual~~ queries are possible
• Storage req for data → less	• storage req → data is significant.	• storage requirement ↳ data is huge.
• Eg → Phone Nos, Customer Name, Social Security Numbers	• eg → Server logs, Tweets organized by hashtags, emails sorted by inbox, sent or draft folders	• eg → emails & msg, image files, open ended survey answers

TRU NOTE®

Page No.
Date

BASE -

Basically **A**vailable, **S**oft State, **E**ventual Consistency

- Basically Available: system is guaranteed to be available in event of faliure
- Soft State: even without an input system state may change
- Eventual Consistency: system will be consistent over time.

ACID	BASE
• Atomicity consistency Isolation ~~Durability~~	•
• consistenty	• weak consistency.
• Availability → less imp	• Availability → most imp.
• Evolution difficult	• evolution easy
• posses expensive joins & relationship	• free from joins & relatn
• high maintenance cost	• low maintenance cost
• Vertical scaling	• Horizontal scaling.

Page No.
Date

SQL	NO-SQL
Relational DBMS	Non relational DBMS
Vertically scalable	horizontally scalable.
predefined schema	No schema or relaxed schema
SQL → query database	Unstructured query language
table based database	document based, graph or key value pair
emphasis on ACID properties	Flows, CAP theorem, Brewers
Schema → fixed or rigid	Schema → dynamic.
Pessimistic	optimistic.
Eg → MySQL, Oracle, PostgreSQL	Eg → MongoDB, Bigtable, Redis.

NO SQL → not only SQL → non tabular data.

Need
- Handling big data
- storing modelling structured (semi & unstructured data)
- efficient execution of db.
- Scales better & designed with web applns.
- easy scalability

Features
- Relaxed schema / free schema
- Multiple SQL db → executed
- process → unstructured & semi-structured data
- higher scalibility.
- cost effective
- data in form of key value pair, wide columns & graphs.

Types.

* NOSQL Databases.

1) Key Value Store
- simple NOSQL db
- handle → lots of data & heavy load.
- key-value storage → key is unique
  value → JSON, strings, obj, Binary.

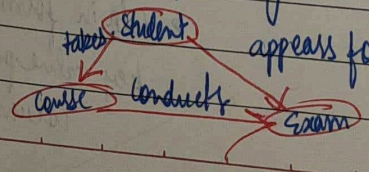
- Shopping Cart Contents
- DynamoDB
- Riak
- Redis

{ Customers:
[:1 → id, id (keys)
{"id":, "name": "Ankita"}, → name
{"id":2, "name": "Kavita"} → name
]
}.

2) Document Store
- use of key-value pair of store & retrieve data
- form → stored data → ~~XML~~ JSON
- most natural among all
- flexibility & ability to query on any field.

3) Graph Based
- used in appln where relationships among data element is important aspect
- connection b/w elements
  ↳ links or relationship
- connections → first class element of db.

Graph → Node → entities itself → people, student
Edge → relationship among entities.



4) Wide column store.
similar to
- traditional relational database
- no. of columns are not fixed.
- column db can quickly aggregate value of given column.

Page No.
Date

* CRUD operations -

Create Read Update Delete operations.

Create database.

use Database_name

create collection.

db.collection_name.insert({key1:value1, key2:value2});

OR

db.createCollection(name, options);

display collections.

show collections

Insert documents.

db.collection_name.insert({key, value})

Multiple doc.

db.student_details.insert(allstudents);

sorting (ascending order)

db.collection_name.find().sort({field_name:1})

Drop database

dropDatabase();

Drop collections.

db.collection_name.drop()

Delete docs

db.collection_name.remove(delete_criteria)

delete one doc.

db.collection_name.remove(delete_criteria, justOne)

remove all docs.

db.collection_name.remove({})

Update docus

db.collection_name.update(criteria, update_data)

TRU NOTE®

Page No.
Date

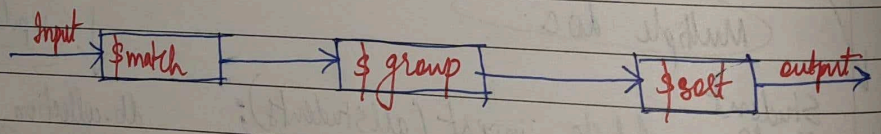
* Indexing →
special datastructure that store small part of collection's data in such a way → use it in querying.

Index store → value of index fields outside table.

- Index creation → db.<collection>.createIndex({KEY:1}).
- Find Index → db.<collection>.getIndexes().
- Drop Index → db.<collection>.dropIndex(IndexName).
- compound Index → db.Student_details.createIndex({name:1,age:

* Aggregation –
process data that results in the computed results.

aggregation framework
input → one or several collection
pipelines → perform successive transfo on data & result obtain



- $match() stage → filter documents that need to work with & which performed
- $group() stage → does aggregation job
- $sort() stage → sort resulting documents the way we require (ascending or descending) –

Page No.
Date

* Map Reduce -

- data processing model → perform opesn on large datasets & produce aggregate results.
- large volume of data.

Syntax →

```
> db.collection.mapReduce(
function() { emit(key, value); }  ← map fucn
function(key, values) { return reduce function(), }  ← reduce fucn.
   out: collection,  ← collection is created in which result of mapReduce is stored.
   query: document,
   sort: document,
   limit: number,
   }
}
```

- map fucn →
  emit fucn → two parameters key & value key.
  key → make groups
  second paramter → perform avg(), sum() calculated
- reduce fucn → aggregate funcn like avg() & sum()
- out → specify collection name where result is stored
- query → we will pass query to filter resultset
- sort → specifies optional sort criteria.
- limit → specifies optional maximum number of documents to be returned.

Unit 6. Advances in DB.

Active Databases

- consists of triggers
- situation & action rules are embeded
- react automatically
- Trigger → unique technique.
- Oracle, DB2, Microsoft SQL server (allow use of triggers)

Deductive Databases

- deduction base on rules & facts
- concept of logic programming
- Prolog → language

1) Facts → same way as rel^n are specified in relational db

2) Rules → specify virtual relations that are not stored but can be formed by facts.

ECA → Action: → sequence of SQL statements: database trans^n or external progm. executed on occurence event.

event → db update oper^n performed explicitly on db

Condition → Rule action should be executed.

action → not specified → action → execute on occurance of event

Page No.
Date

* Main Memory -

- data resides permanently on main physical memory.
- backup copy of such database → maintained on disk.
- access to main memory is faster than disk access.
  transaction → fast
- Backup copy → pmp → main memory fails.
  ↳ entire db is lost.
- eg → CSQC, TimesTen.

XML

- extensible Markup Language.
- similar to HTML
- HTML → repres^n of data.
  XML → transport & store data

Uses → • display meta content
• exchange data b/w appl^n.
• data from db & use din more appl^n.

Features - • Transport / store data
• define & use own tag
• decides → how to display data
• permits data → create tags
• searching sorting, rendering or manipulating XML.

Adv → • Human Readable
• Edit XML docs → simple text editor
• Language neutral
• Tree structure → arranged systematically
• Independant of OS.

JSON

- Javascript object Notation
- store & retreive data.
- text based open format
- extended from Javascript tagging

Features -

- text based, lightweighted data interchange format
- language independant
- read & write
- machine to parse & generate
- C, C++, Java, JS, Perl, Python

Structure - (2)

1) collection of name/value pairs
obj, record, struct, dictionary
2) An ordered list of values.
array, vector, list or sequence

JSON obj -

- holds key value pair.
- key & value seperated by colon.

syntax - {string : value, ~}

eg → "Age" : 38.

Page No.
Date

OODBMS

- object oriented DBMS
- supports creating & modeling data as objects.
- Handles larges & complex data than RDBMS
- Data handling - store data as well as methods to use it.
- object - oriented
- Inheritance & encapsulation
  ↳ reduce data redundancy
- eg → object Database, objectivity/DB, object Store, Cache & ZODB

DBMS

DBMS

- store everything as a relation b/w them entities.
- simples data.
- Data handling - RDBMS stores only data.
- table - oriented.
- Normalization
  ↳ eliminate data ~~redundancy~~.
- eg → MSSQL, MySQL & Oracle

Geometric Data

- represent in db
  ↓
  normalized fucn.
- deals with mathematical opern
- info on storing geometric constructs in db-
  represented
  1) Line → coordinates of its endpoints
  2) Curve → approx a curve by partioning into a storage.
     - list → vertices in order
     - each seg → seprate tuple
       ↓
       indentifier of curve.

closed polygons →

- listed with vertices in order
  starting vertex is same as ending vertex.
- represent boundary edges as seprate tuples
- triangulation → divide polygon into triangles
  use identifier for each triangle.

Geographic Data

- data → spatial in nature
- collection of info
  ↓
  describe object & thing with relation to space.
- eg → map or satellite img

Appln

1) Web based road map service
   ↳ vehicle navigation
2) Vehicle navigation system
   ↳ store info roads & service
3) GPS (Global positioning System)
   ↳ info ~~broadcast~~ broadcast from GPS satellites.
   ↓
   accurate locn.

Types of Geographic Data

1) Raster Data
   ↓
   bit/map/pixel maps in 2D or more
   eg → satellite img of cloud cover
2) Vector Data
   ↓
   constructed from basic obj such as points & so,
   ↓
   region/lakes
   ↓
   polygons