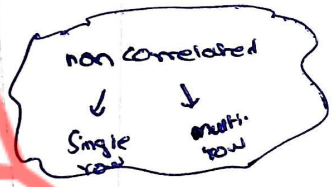
DBMS 2 Day 1 - Subqueries Sum up

- What? A select stmt within main Statement/Clause/Query
- Benefits/why? Simplify logic, debug & performance increases.
- pre-requisite? common key column b/w main query & subquery ; use `( )' parenthesis.
- Note: dynamic value return by the subquery used by the main query ; used in select, from, where, having, etc

- Classification of subqueries

  - based on structure
    - scalar subquery
    - row /single row subquery
    - column /single colm subquery
    - table subquery

  - based on dependence with Main query (i.e. if subquery runs based on i/p from main query)
    - Dependence subquery => run only once.
    - Correlated subquery => runs multiple times (based on i/p from main query)

- Classification of operators

  - based on no. of rows
    - Single row op. => for single row subquery
      - Comparison
        - e.g. =, >, <, etc
    - Multi row op. => for multiple row subquery
      - Comparison
        - e.g. any, all, in
  - based on Subquery
    - Scalar subquery => for it : =, >, <, etc
    - Column subquery => for it : <any/all> <comparison op.> ; in

      same for 'row' table subquery

→ Search condition (dynamic value from subquery used in main query)

- 1 Comparison test => =, >, <, etc
- 2 Membership test => 'in'
- 3 Existence test => exists
  - check record exists as such in subquery
- 4 Quantified test => <any/all> <comp. op.>
  - to validate multiple rows of the subquery.

→ Subquery in where clause
- with all above 4 search conditions.

→ Subquery in from clause
- → outer reference / derived table (with specific column)
- → is the most example of table subquery. * Imp.
- → used mainly with joins.

→ Subquery in with clause => once executed, & stored value, multiple times called in main query

→ Subquery in having clause
- → Comparison of grouping result with grouped result.

→ Nested subquery => subquery within subquery

1 order of execution: low level subquery to high.

2 order of our thinking to code: High (final result) to low level

- → no limit of no. of subqueries within.
- → used with comparison test, membership test
- → mostly not preferred practice.
- → loss of data (& column: common / not common) unlike joins.

Recursive CTE

```
With recursive <cte name> <column name> as
(
    Stmt1 (once only execute)
    Set operators (union / intersect / except)
    Stmt2 (used for recursive calling)
)
_____ ;
```

DBMS 2 Day 2 window function / OLAP / Analytical func. / Adv. agg. func.

- window function => [in Select Strmt]
  - Calculate for each row in given key column (mentioned in over clause) with it related set of rows.
  - it is an imaginary window.
  - two types: 1. Agg. window func. (e.g. sum, max etc)
    2. non Agg. window func. (e.g. rank, ntile)
- over ( ) clause
  - no specification i.e. over() => use whole data set / table.
    - e.g. sum(numbers) over()
  - 3 main sub-clauses in this over() clause
    - partitioned by => for fixed window. (1st in execution in over clause)
      - it skips the repetition ~~data~~ tup.
    - order by => 1st it sorts given column based in each partitioned then 2nd it uses sliding window
      - for agg. function calculation window
      - Not think of this Concept of sliding window in non-agg. window func. (rank, ntile, first_value, etc)
      - is most important for rank, ntile, 1st value, last value, nth value (mainly)
    - frame specification
      - range
      - row

Order by & frame specification are most depend on each other. least dependent in this 3 is partitioned (which can stand by on own itself)

- Advantage
  - reduces no. of self-joins.
  - ↑ performance as => Mapping rows is not there, unlike joins.
  - uses
    - row value, agg. value portrait in each row.
    - helpful in cummulative calculation.
    - More granular level results.

- window Agg. functions
  - for each row calculate the agg. value for given key column for a ordered set of rows / related set of rows (i.e. partitioned, orderby, frame specification) & => gives result in each row.
  - e.g. Sum, Avg., max, min, etc

- non-Ag. window fnctns
  - rank => Mainly based on orderby Clause in Over clause
    - sequential, for each partitioned ordered set of rows.
    - Based on Gaps & repitition Classified

rank	✓Gaps & ✓rep.
percent-rank	✓ Gaps & ✓rep.
row-number	no Gap & no rep.
dense rank	no Gap & ✓rep

- Ntile => Mainly based on order by clause in over clause.
  - to bucket the related rows into specified no. of buckets.
  - last bucket can have some more or less.
- first value, last value, nth value
  - Mainly based on order by clause in overclause.
  - return respective value from related set of rows based on given key column.
  - return null if no such value possibility
- lag & lead
  - also Mainly based on orderby clause in overclause.
  - But preferred to use the (main/sub -query) order by with CTE & then call the CTE. (as it may return in wrong if used order by clause in over clause)
  - To back/forth with specified value in given key column.
- Note
  - leave null as null (as may be misinterpreted if changed)
  - while pooling tables, if already column name exists, then donot use alias with that name.
  - no window function inside another window function.

for multiple times use later

- naming convention of window function => [after from clause] window ——— as <windowname>().

- Query expression => set of rules (search condition)
  - replace, instr, soundex, wild filtering, case format
- row value expression
  - row values with series of colun values.
  - expressed as
    - (1) (col name 1, 2, ...) (row value 1, 2, ...)
      - in insert stmt
    - (2) col name 1, 2, ... => in select stmt
- Cross tabulation
  - row to column & colum to rows
  - for summary & trend analysis
- joins v/s subquery
  - no complex logic in Main query
  - calculation much in main query
  - automatic filter.
  - filter rows (i.e. order of filtering is users wish)
- in operator => datatypes must match.
- not automatic default values assign in insert stmt (but there are symbol for it).

DBMS 2 Day 3 Sum up

Cume_dist

- Calculates cume_dist of a value w.r.t given values i.e. % of values ≤ the value.
- order by clause is very important in over clause

Orderby clause in overclause

- In general, very important & must for window functions without any arguments.
  - e.g. rank, cume_dist, etc.

Frame Specification

- order by clause in over clause is important as it executes after it. (but not must for all types)
- It helps to control the window Here by data.
- 1st unbounded preceding
  2nd Current row
  3rd unbounded following
- Two types
  - 'row'
    - deals with rows (not values)
    - helps to avoid default behavior of the Orderby Clause in over Clause which is skipping the repitition. Skip.
    - order by clause in over Clause is not mandatory.
  - range
    - deals with values (not rows)
    - order by clause in over Clause mandatory.

Considers all repitition in Calculation.

Syntax e.g.

over(row blw — and —>

over(order by range blw — and —>

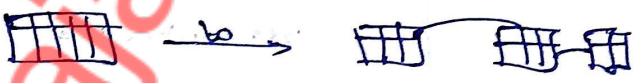
ACID Properties

txn → often is a group of stmts.

- Atomicity: one txn — either complete txn or not at all.
- Consistency: DB from one consistent state to another
  - one txn
  - with help of: 1 constraints, 2 cascade, 3 triggers, etc.
- Isolation: [for concurrent txns] DB state change like if happened sequentially
  - with help of isolation.
- Durability: once commit, then never loss even if system failure!

Normalization

- it is for RDBMS
- ↑ integrity, ↓ dependency, ↓ redundancy
  - check 2NF examples. (as split into diff tables)



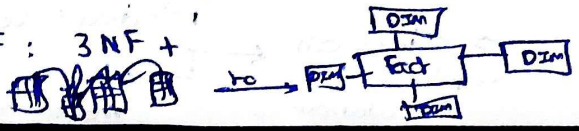
1NF: each column ~~value~~ each value is atomic/scalar (not e.g. 1,2)

2NF: 1NF + non-prime column are fully functionally dependent on primary key.

3NF: 2NF + non-prime with no transative dependy.

4NF: 3NF +

e.g. one cheque with many trans. (e.g. tax, Amt etc.)



i.e. Complex to Simple (fact-dim)

Cascade delete, update

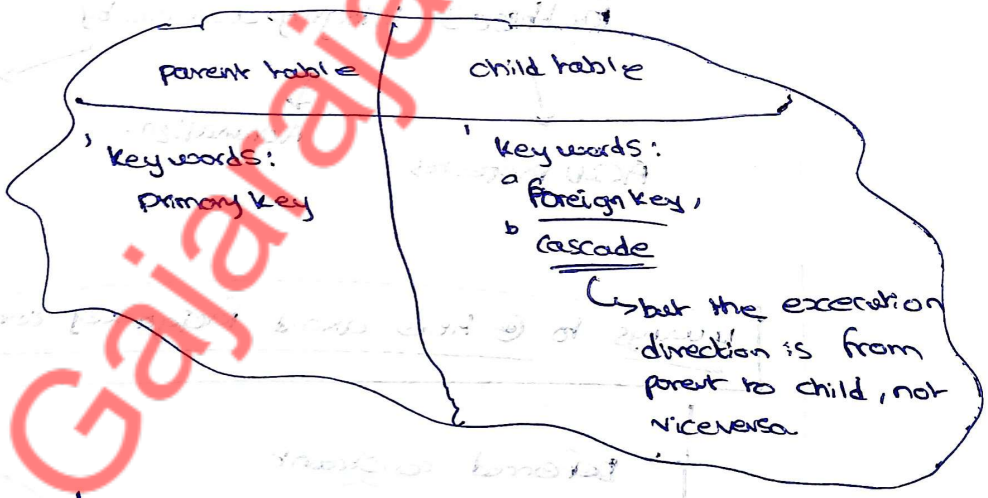
↳ Only for parent table to child table, not vice versa (in direction of execution)

↳ So, supports only cascade delte, update (related to parent table); not insert related to child table)

foreign key constraint

1. not allow delete, update in parent table value if the child already has that value
2. not allow insert value in child table if no such value in parent table

↳ Cascade works on top of foreign key constraint



↳ Syntax:

1. on update cascade/restrict
2. on delete cascade/restrict

↳ Cascade delete

↳ 1st delete the record from parent then all records related to it in the child table.

!!

Similarly cascade update, but it updates.

Boilerplate code

↳ pure syntax w/o logic, thinking.

↳ 1st write it then think logic.

e.g. with recursive <cteName> as (select * from t1 union select * from <cteName>) select * from <cteName>;

Data integrity

- maintain accuracy, reliability, relevant.
- data lineage.
- benefits → reduce cost w.r.t
  - Storage
  - decision mkg.
- Three type
  - row level integrity (achieved by P_key constraint)
    - primary key
    - no duplicate & null
    - entity integrity (related to it)
  - Column level integrity (achieved by column constraints)
    - column metadata ensured (eg. type, size, allowed values, etc)
  - Referential integrity (achieved by foreign key constraint)
    - Parent–child rln.

for these 3 integrity achieve by

- ACID properties
- Atomicity.
- referential integrity

4 ways to @ times avoid foreign key constraints

- Deferred constraint
- Set foreign key check=0
- remove the foreign key constraint
- Staging

Be careful.

Day4 DBMS2 Sump up

AutoCommit

- default is set 1 => All DML queries are permanent (In, up, delete)
- if 0 => then DML queries are temporary

Transcation => ensure atomicity (by commit / rollback) at end.

- Start Transaction ; → Start Txn
  - Then there is no autocommit : we need to commit / rollback explicity
- commit / rollback ;
  - commit → Saves permanently the temporary data
  - rollback → delete permanently the temporary data.
  - These two only ends the txn
    - => after this, txn complete & autoCommit is in previous state.

- Savepoint <savepoint_name> ; → it creates the savepoint to which we can roll back.
  - Once rollback to savepoint x then all DML queries executed x below it lost permanently. (even other savepoints like x+1, x+2, ----)

Note:
- It is for DML queries
- whereas DDL queries, commit if executed all things happened earlier & including it

- rollback to savepoint <savepoint_name> ;
- release savepoint <savepoint_name> ;
  - it just removes the savepoint (so not rollbackable to that removed savepoint) from set of savepoint. Note: It not deletes the DML queries executed

Isolation

- This is for concurrent txns.
- default InnoDB (where txns execute in mysql) is repeatable read.
- Set transcation isolation level _____ .
  - ⇓ only
  - then, start txn.
- 4 levels

(easy)
1. read uncommited => reads uncommited data
2. read Committed => reads commited data
3. repeatable read => reads repeatedly the same data. (but update/write with current data, not previous)
   - i.e. Non-rep. read => read different data.

(Strict)
4. Serializible => sequentially one after another execution of txns.

- Dirty read => read uncommitted data
- Non-rep. read => read diff. data
- phatom phenomena => sudden appear & disappear of data.

vadhe by level 2, 3, 4
vadhe by 3, 4
vadhe by 4

Views

- is Stored Select Stmt with Complex/Simple logic. [not stored data]
  - ⇓
  - So dynamic reflect only commited data.
- H/w check with diff. types of views
- Insertable/updateable/deletable view => i.e. View, in default, allow to run DML query for the base table. (Views to table)

↳ Materialized Views => Stores not select queries like views, but stores the actual data.

↳ refresed @ times [fast, complete]
- fast → only changes update.
- complete → truncate & populate

views → privacy

↳ Types

- Simple → one table with (simple logic)
- Complex → 1 or more table, with join, subquery, with clause, conditioned filtering
- Horizontal View → all columns of base table
- vertical view → selective columns of base table.
- Group view → Always not updatable view for agg. data
  → view with dynamic reflect of Group by clause in select Stmt., show only agg.
- join view → view with joins / subqueries.

↳ View Insertable / updatable / deletable :

data in		condition	change in View	change in table
Table	View	met / not met		
X	X	met / not met	X	X
✓	✓	not met	X	✓
✓	✓	met	✓	✓

↳ Views (Hlw) (xup) run DML query for the base table is controlled / restricted with help of with Check Constraint in where clause.