What is a master table?

Designing master tables is the first step in database design. Master tables hold basic information about a system. To understand master tables, we need to understand the concept of a transaction. A transaction is an activity performed by entities within the system. The permanent data is called MASTER data like Employee Information, department info. Day to day business data is called TRANSACTION data like sales, salaries, expenses. In SAP, Material, customer, and vendor etc. are called master data tables.

**Group By**

I understand the point of **GROUP BY x** but how does **GROUP BY x, y** work, and what does it mean?

Group By X means put all those with the same value for X in the one group.

Group By X, Y means put all those with the same values for both X and Y in the one group

**EXISTS vs. IN**

The EXISTS function searches for the presence of a single row meeting the stated criteria as opposed to the IN statement which looks for all occurrences.

TABLE1 - 1000 rows

TABLE2 - 1000 rows

(A)

SELECT t1.id

FROM table1 t1

WHERE t1.code IN (SELECT t2.code

FROM table2 t2);

(B)

SELECT t1.id

FROM table1 t1

WHERE EXISTS (SELECT '1'

FROM table2 t2

WHERE t2.code = t1.code)

For query A, all rows in TABLE2 will be read for every row in TABLE1. The effect will be 1,000,000 rows read from items.

In the case of query B, a maximum of 1 row from TABLE2 will be read for each row of TABLE1, thus reducing the processing overhead of the statement.

**Rule of thumb:**

If the majority of the filtering criteria are in the subquery then the IN variation may be more performer.

If the majority of the filtering criteria are in the top query then the EXISTS variation may be more performer.

I would suggest, you should try both variants and see which works the best.

**See the link as details:**

<https://asktom.oracle.com/pls/asktom/f?p=100:11:0::no::p11_question_id:442029737684>

**Example 02:**

**a)**

**SELECT DISTINCT** s.OFFICIALNUMBER **AS** O\_No , s.FULLNAME **AS Name**, r.RANK\_NAME **AS Rank**, bpu.NAME **AS** posting\_unit , s.SAILORID

**FROM** sailor s

**left join** bn\_rank r **on** r.RANK\_ID = s.RANKID

**LEFT JOIN** bn\_branch b **ON** s.BRANCHID = b.BRANCH\_ID

**LEFT JOIN** bn\_daogroup dao **ON** b.DAO\_GROUPID = dao.GROUP\_ID

**LEFT JOIN** bn\_posting\_unit bpu **ON** s.POSTINGUNITID = bpu.POSTING\_UNITID

**LEFT JOIN** transfer t **ON** s.SAILORID = t.SailorID

**LEFT JOIN** partii pa2 **on** pa2.PartIIID = s.FIRSTPARTID

**where** s.SAILORSTATUS=1

**AND** dao.GROUP\_ID **IN** (1,2,3,4,5,7)

**AND** p.ORG\_ID **IN** (63) **and not exists** (**select** 1 **FROM** sailor m **left join** assessment ass **on** ass.SailorID = m.SAILORID **where** AssessYear = **'2018'and** m.SAILORID = s.SAILORID )

**b)**

**select** m.**SAILORSTATUS**,m.**SAILORID**, m.**OFFICIALNUMBER**, m.**FULLNAME**, (**CASE WHEN** (p.**Name** != **''**) **THEN** concat(r.**RANK\_NAME**, **'('**, p.**Name**, **')'**) **ELSE** r.**RANK\_NAME END**) **AS** RANK\_NAME, p.**NAME** posting\_unit

**from**

(**select SAILORSTATUS**, **SAILORID**, **OFFICIALNUMBER**, **FULLNAME**, **RANKID**, **FIRSTPARTID**, **POSTINGUNITID**, **BRANCHID**, **AREAID**, **ZONEID**

**from** sailor

**where SAILORSTATUS** = 1

**and SAILORID not in** (**select SailorID from** assessment **where AssessYear** = 2018))m

**left join** bn\_rank r **on** m.**RANKID** = r.**RANK\_ID**

**left join** bn\_branch br **on** br.**BRANCH\_ID** = m.**BRANCHID**

**LEFT JOIN** bn\_posting\_unit p **ON** m.**POSTINGUNITID** = p.**POSTING\_UNITID**

**left join** partii par **on** m.**FIRSTPARTID** = par.**PartIIID**

**LEFT JOIN** bn\_trade t **on** t.**TRADE\_ID** = par.**TradeID**

**LEFT JOIN** bn\_daogroup dao **ON** dao.**GROUP\_ID** = br.**DAO\_GROUPID**

**where** m.**SAILORSTATUS** = 1

In the above code, we are getting as follows. The **not exists** from a) and **not in** from b) are the same like this

**not** *exists* (**select** 1 **FROM** assessment asses **where AssessYear** = **'2018'and** asses.**SailorID** = s.**SAILORID**)

**SAILORID not in** (**select SailorID from** assessment **where AssessYear** = 2018)

Let’s take a look another example

**SELECT** r.**RANK\_CODE**, r.**RANK\_ID**, r.**RANK\_NAME**, b.**BRANCH\_ID**, b.**BRANCH\_NAME**

**FROM** bn\_branch b, bn\_rank r

**WHERE** b.**BRANCH\_ID** = r.**BRANCH\_ID**

**AND** r.**ACTIVE\_STATUS** = 1

**AND** b.**ACTIVE\_STATUS** = 1

**AND** *EXISTS*(**SELECT** 1

**FROM** bn\_equivalent\_rank er

**WHERE** er.**ACTIVE\_STATUS** = 1

**AND** er.**EQUIVALANT\_RANKID IN** (11,1,2)

**AND** er.**EQUIVALANT\_RANKID** = r.**EQUIVALANT\_RANKID**)

OR…

**select r.RANK\_CODE, r.RANK\_ID, r.RANK\_NAME, b.BRANCH\_ID, b.BRANCH\_NAME**

**from** bn\_branch b

**join** (**select RANK\_CODE**, **RANK\_ID**, **RANK\_NAME**, **BRANCH\_ID from** bn\_rank **where ACTIVE\_STATUS** = 1 **and EQUIVALANT\_RANKID in** (2,1)) r **on** r.**BRANCH\_ID** = b.**BRANCH\_ID**

**where** b.**ACTIVE\_STATUS** = 1

**group by** b.**branch\_id**

**OR…**

**select b.BRANCH\_ID, b.BRANCH\_CODE, b.BRANCH\_NAME**

**from** bn\_branch b

**where** *exists* (**SELECT** 1 **FROM** bn\_rank

**WHERE ACTIVE\_STATUS** = 1

**AND BRANCH\_ID** = b.**BRANCH\_ID**

**AND EQUIVALANT\_RANKID in** (2))

**order by BRANCH\_ID**

**Example 03:**

SELECT 1 FROM TABLE\_NAME means, "Return 1 from the table". It is pretty unremarkable on its own, so normally it will be used with WHERE and often EXISTS. **select 1** from table will return the constant 1 for every row of the table. It's useful when you want to cheaply determine if record matches your where clause and/or join.

**SELECT \* FROM TABLE1 T1 WHERE EXISTS (**

**SELECT 1 FROM TABLE2 T2 WHERE T1.ID= T2.ID**

**);**

Basically, the above will return everything from table 1 which has a corresponding ID from table 2

**SELECT \* FROM TABLE1 T1 WHERE ID IN (SELECT ID FROM TABLE2);**

Sometimes **SELECT 1 FROM TABLE\_NAME** will be used. This does not offer significant benefit over selecting an individual column, but, depending on implementation, it may offer substantial gains over doing a **SELECT \***

If you mean something like this

**SELECT \* FROM AnotherTable**

**WHERE EXISTS (SELECT 1 FROM table WHERE...)**

Then it's a myth that the 1 is better than

**SELECT \* FROM AnotherTable**

**WHERE EXISTS (SELECT \* FROM table WHERE...)**

[The 1 or \* in the EXISTS is ignored and you can write this as per Page 191 of the ANSI SQL 1992 Standard:](http://www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt)

**SELECT \* FROM AnotherTable**

**WHERE EXISTS (SELECT 1/0 FROM table WHERE...)**

**Example 04:**

Another Example of **not in**

**RankID IN (SELECT bn\_rank.RANK\_ID FROM bn\_rank WHERE bn\_rank.EQUIVALANT\_RANKID IN ($branchRank))**

**Avoid In-Line Queries**

In-line queries execute for each column for each row.

For example, if a main query has 100 columns, and brings 1000 rows, then each column query executes 1000 times. Altogether, it is 100 multiplied by 1000 times. This is not scalable and cannot perform well. Avoid using in-line sub queries whenever possible.

Avoid the following use of in-line queries. If this query returns only a few rows this approach may work satisfactorily; however, if the query returns 10000 rows, then each sub or inline query executes 10000 times and the query would likely result in Stuck threads.

**SQL CASE Statement**

The SQL CASE statement is the way of implementing if/then logic. The CASE statement is followed by at least one pair of WHEN and THEN statements and Every CASE statement must end with the END statement. The ELSE statement is optional in a CASE statement.

**For Example:**

**SELECT player\_name, year,**

**CASE WHEN year = 'SR' THEN 'Senior Player'**

**WHEN year = 'JR' THEN 'Junior Player'**

**ELSE NULL**

**END AS position**

**FROM benn.college\_football\_players**

The CASE statement checks each row to see if the conditional statement - year = 'SR' is true.

For any given row, if that conditional statement is true, the word “Senior Player” gets printed in the column that we have named **position**.

In any row for which the conditional statement is false, nothing happens in that row, leaving a null value in the position column.

At the same time, SQL is retrieving and displaying all the values in the player\_name and year columns.

**Example 01:**

Multiple OR operator implementing in a CASE statement like so

This can be either:

**CASE ebv.db\_no**

**WHEN 22978 THEN 'WECS 9500'**

**WHEN 23218 THEN 'WECS 9500'**

**WHEN 23219 THEN 'WECS 9500'**

**ELSE 'WECS 9520'**

**END as wecs\_system**

Or, this one:

**CASE**

**WHEN ebv.db\_no IN (22978, 23218, 23219) THEN 'WECS 9500'**

**ELSE 'WECS 9520'**

**END as wecs\_system**

**Example 02:**

We can use directly or operator in the case statement like this

**CASE**

**WHEN ebv.db\_no = 22978 OR**

**ebv.db\_no = 23218 OR**

**ebv.db\_no = 23219**

**THEN 'WECS 9500'**

**ELSE 'WECS 9520'**

**END as wecs\_system**

**Example 03:**

Simple switching statement using case

SELECT

\*

FROM

Test

WHERE

Account\_Location = (

CASE LocationType

WHEN 'location' THEN @locationID

ELSE Account\_Location

END

)

AND

Account\_Location\_Area = (

CASE LocationType

WHEN 'area' THEN @locationID

ELSE Account\_Location\_Area

END

)

Or, we can try like this

SELECT column1, column2

FROM viewWhatever

WHERE

(@locationType = 'location' AND account\_location = @locationID)

OR

(@locationType = 'area' AND xxx\_location\_area = @locationID)

OR

(@locationType = 'division' AND xxx\_location\_division = @locationID)

**Example 04:**

Case with union all statement

**SELECT** abc.**SAILORID**, abc.Name, abc.Relation, abc.RELATIONID, n.**NomineeID**,n.**Percentage**

**FROM**

(**select SAILORID**, **FATHERNAME Name**, **'Father'**Relation, 4 RelationID **from** sailor **where SAILORID** = **'5557'**

**union all**

**select SAILORID**, **MOTHERNAME Name**, **'Mother'**Relation, 1 RelationID **from** sailor **where SAILORID** = **'5557'**

**union all**

**select SailorID** SAILORID, **SpouseName Name**, **'Wife'**Relation, 5 RelationID **FROM** marriage **where SailorID** = **'5557'and MaritalStatus** = 1

**union all**

**select SailorID** SAILORID, **Name**, (**case when** (**Gender** = 1) **THEN 'Son'ELSE 'Daughter'END**)Relation,

(**case when** (**Gender** = 1) **THEN** 7 **ELSE** 6 **END**)RelationID

**from** children **where SailorID** = **'5557'**) abc

**left JOIN** nominee n **ON** abc.**SAILORID** = n.**SailorID AND** n.**RelationID** = abc.RELATIONID

**WHERE** abc.**SAILORID** = **'5559'ORDER BY** n.**Percentage**, abc.RELATIONID;

**Example 05:**

Case with select distinct statement

**select DBMS\_LOB.substr(REPLY\_DESC, 4000) REPLY\_DESC, FORUM\_ID, REPLY\_ID, IS\_STUDENT, CRE\_BY, CRE\_DT,**

**case when** r.IS\_STUDENT = **'Y'then**

**select distinct** FULL\_NAME\_EN

**from** STUDENT\_PERSONAL\_INFO

**where** STUDENT\_ID = r.CRE\_BY

**when** IS\_STUDENT = **'N'then**

**select FULL\_NAME**

**from** SA\_USERS

**where USER\_ID** = r.CRE\_BY

**end** avatar\_name

**from** UMS\_FORUM\_REPLY r

**where** FORUM\_ID = **'41'**

**order by** CRE\_DT **asc**

**Example 06:**

Case statement in update

**UPDATE** UM\_LAB\_EXP\_ASSIGN

**SET** LAB\_ID = **CASE WHEN** EXP\_ID = 25 **THEN** 37

**ELSE** LAB\_ID

**END**,

ACTIVE\_STATUS = **CASE WHEN** EXP\_ID = 25 **THEN 'Y'**

**ELSE** ACTIVE\_STATUS

**END**,

CRE\_BY = **CASE WHEN** EXP\_ID = 25 **THEN** 9

**ELSE** CRE\_BY

**END**,

ORG\_ID = **CASE WHEN** EXP\_ID = 25 **THEN** 1

**ELSE** ORG\_ID

**END**

**WHERE** EXP\_ID = 25

For more [link](http://www.dotnet-tricks.com/Tutorial/sqlserver/1MS1120313-Understanding-Case-Expression-in-SQL-Server-with-Example.html) or [link2](https://mode.com/sql-tutorial/sql-case/)

**Sub Query**

You can do the general idea at least 3 ways, using a LEFT JOIN, and also using NOT IN and NOT EXISTS.

**Via LEFT JOINS.**

SELECT student\_name

FROM table\_A a

LEFT JOIN table\_B b ON a.student\_id = b.student\_id

WHERE b.student\_id IS NULL

This gets all student information in table\_A, where the student is not in table\_B.

**and here it is via NOT EXISTS:**

SELECT student\_name

FROM table\_A a

WHERE NOT EXISTS (SELECT student\_id FROM table\_B b WHERE b.student\_id = a.student\_id)

and via NOT IN

SELECT student\_name

FROM table\_A a

WHERE a.student\_id NOT IN (SELECT student\_id FROM table\_B b)

**PHP Vs SQL**

PHP:

<?php foreach ($branch as $branchRow): ?>

<optgroup label ="<?php echo $branchRow->BRANCH\_NAME ?>">

<?php $rank = $this->db->query("SELECT RANK\_ID, RANK\_CODE, RANK\_NAME FROM bn\_rank WHERE ACTIVE\_STATUS = 1 AND BRANCH\_ID = $branchRow->BRANCH\_ID AND EQUIVALANT\_RANKID in ($eq\_rank)")->result(); ?>

<?php foreach ($rank as $row): ?>

<option value="<?php echo $row->RANK\_ID ?>"><?php echo "[" . $row->RANK\_CODE . "] " . $row->RANK\_NAME ?></option>

<?php endforeach; ?>

<?php endforeach; ?>

</optgroup>

SQL:

**select** b.**BRANCH\_ID**, b.**BRANCH\_CODE**, b.**BRANCH\_NAME**

**from** bn\_branch b

**where** *exists* (**SELECT** 1 **FROM** bn\_rank

**WHERE ACTIVE\_STATUS** = 1

**AND BRANCH\_ID** = b.**BRANCH\_ID**

**AND EQUIVALANT\_RANKID in** (1,11))

**order by BRANCH\_ID**