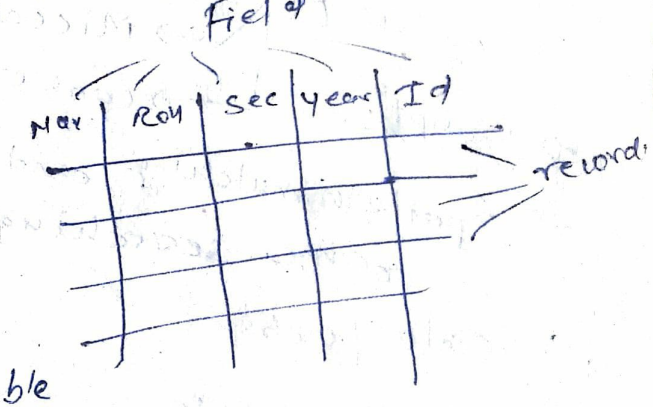
DBMS Lab 8/1/18

Rules for SQL : Structured Query Language

(1) statement always starts with a verb followed by no. of clauses.
(2) space is used to seperate the clause
(3) , is used to seperate parameter
; is used to end a statement.
(4) statement may split across lines but not within keyword.

Component of SQL

(i) Data Retrival Component : SELECT
(ii) Data Manuplation language (DML) :- INSERT, UPDATE, DELETE, MERGE
(iii) Data Definition Language (DDL) :- CREATE, ALTER, RENAME, DROP, TRUNCATE.
(iv) Transaction Control Language (TCL) :- COMMIT, ROLLBACK (undo), GRANT, REVOKE.
save



> desc emp (finding emp table)
> select * from emp;
> set linesize 120
> select * from emp;

c: user > Silicon > desktop > folder : itsmine

SQL > spool on
SQL > spool c:/users/Silicon/Desktop/itsmine/oissl.txt

SQL > ed change command
> / run command

sql : credentials
username - SCOTT
pwd tiger
Hoststring Silicon.

Index page format:

1. SQL Assignment. Basic SQL

class

Datatype

String char varchar2 alphanumeric data

number NUMBER
NUMBER (P, S)
P → total no of digit
S → digit after decimal.

Date DD-MON-YY
e.g. '05-JAN-18'.

15/01/18

SQL function

→ Single row fn
- → character
  - → translate
  - → SUBSTR
  - → lower
  - → upper — select lower(ename) from emp; → for name in lower case; upper for upper case
  - → initcap
  - → length
  - → LTRIM → select LTRIM('silicon', 's');
  - → RTRIM
  - → Lpad — select Lpad(ename, 10, '*') from emp;
  - → RPAD:
- → numeric
- → date
- → Miscellaneous
- → conversion

→ Group fn:

→ pattern matching and range searching.

→ scalar function

→ Aggregate function

Number fn: Round: select round(15.19, 1) from dual;

→ SQRT select sal, sqrt(sal) from emp;

→ Trunc select trunc(125.815, 1) from dual.

→ mod select mod(15,7) from dual;

↳ floor select floor(24.4) from dual;
↳ ceil ceil(24.4) from dual;
↳ power select ~~power(3,2)~~ power(3,2) from dual;

↳ Groupfunction: avg select avg(sal) sal from emp;
: min
: max
: count
: sum

---

pattern matching and range search.

↳ Between
select * from emp where sal between 1000 and 2000; (both inclusive.)
or sal ≥ 1000 and sal ≤ 2000;

↳ in() select * from emp where job in('salesman', 'clerk', 'manager')
↳ or job not in

---

like/Notlike

LIKE : select * from emp where like 'A%'; starting with A
'%A'; ending with A.

like 'H____';
↳ column name — there must be exactly five characters and first is an H.

%A%
_ _
= A%
__A__
%A_

↳ Group by: select sum(sal) from emp group by deptno.
: select deptno, sum(sal) from emp group by deptno

```
select sum(sal) from emp
group by deptno
having sum(sal) > 2000;
```

→ type is name so arranged in decreasing order as in dictionary.

order by.

```
select * from emp
order by sal desc;
```

↓ descending

propertyno	type	rooms	rent
PG16	FLAT	4	450
PL94	FLAT	4	400
PG36	"	3	375
PG4	"	3	350
PA14	House	6	650
PG21	HOUSE	5	600

lst ordered by type and in type, then ordered by rent.

eg; select propertyno, type, rooms, rent from property for rent order by type, rent desc;

C.A:- 24/01/18

```
create table celebs (id INTEGER, name TEXT, age INTEGER);
```

id INTEGER PRIMARY KEY

o/p:

	celebs
id (PK)	INTEGER
name	TEXT
age	INTEGER

Add value

```
insert into celebs (id, name, age)
values (1, 'Justin', 19);
```

or

```
insert into celebs
values (1, 'Justin', 19)
```

op:

id	name	age
1	Justin	19

```
select name from celebs;
```

o/p Justin

Add another value

```
insert into celebs (id, name, age)
value (2, 'Taylor', 25);
```

o/p: ie the new table is (using: select * from celebs)

id	name	age
1	Justin	21
2	Taylor	25

```
update celebs
set twitter_handler = '@taylor'
where id = 2;
```

op:

id	name	age	twitter_handler
2	Taylor	25	@taylor

```
delete from celebs where twitter_handle is NULL;
select * from celebs;
```

create table awards(id INTEGER PRIMARY KEY, recipient TEXT is not NULL, award_name TEXT DEFAULT "GRAMMY");

DB sheare award

id	INTEGER
recipient	TEXT
award_name	TEXT

```
select distinct genere from movie
where year > 2014;
```

```
Select * from movies
where name like 'The%';
```

The avenger
The - -

```
select name, year, imdb_rating
from movies
order by imdb_rating desc;
```

Question. statement

1. Use a case statement to create a column called mood that is
   - fun if genere is romance
   - fun if genere is comedy
   - serious in all other case

```
SQL7 Selet name,
     case
       when genre is 'romance' then 'fun'
       when genre is 'comedy' then 'fun'
       when genre is not 'romance' or genre is
            not 'comedy'. then 'serious'
     END
     from movies;
```

O/P:

name	case when genre is 'romance' ... 'serious'
Avatar	
JurassicWorld	

note: every column name, command, function, datatypes... can be written in both smaller or upper case letters

25/01/18

Aggregate functions in SQL

i) count() : count the number of rows

~~sum()~~ takes attribute (name of column) as argument and returns no. of non-empty rows in that column.

DB schema (for example)

fake_apps

id	Integer
name	Text
category	text
downloads	Integer
price	Real

eg:
```
select count(*)
from fakeapps;
```
o/p: count(*)
200 // 200 no. of rows.

ii) sum() : Returns sum of all the values.

argument - name of column
return - sum of all the values in that column

eg; select sum(download)
from fake_apps;

o/p: sum(downloads)
3322760

iii) max()/min : Get the largest or smallest values.

argument: name of column
return: largest/smallest value in the column

eg; select min(downloads)
from fake_apps;

o/p: min(downloads)
1367.

iv) avg(): returns the average value.

argumen: name of column
return: average value for that column.

eg; select avg(price)
from fake_apps;

o/p: avg(price)
2.02365

v) round(): round the values in a column to the number of decimal places specified by the integer.

two arguments.
- a column name
- an integer

eg; select name, round(price, 0) from fake_apps;

o/p;

name	Round(price, 0)
siliconphase	0.0
Ponzolab	1.0

Clauses:

⇒ GROUP BY: Used with aggregate function in collaboration with the select to arrange identical data into groups.
* comes after any where, but before order by or limit.

e.g; select category, sum(downloads) from fake_apps group by category

o/p:

category	sum(downloads)
Book	160864
Business	178726
Catalogs	18618
⋮	

e.g; select category, price, avg(downloads) from fake_apps group by category, price;

o/p:

category	price	avg(downloads)
Books	0.0	11926.5
Books	0.99	27705.5
Books	1.99	21778.87
⋮		

⇒ HAVING: grouping data with some more conditions;

e.g; select price, second(avg(downloads)) from fake_apps group by price having count(name) > 9;

o/p:

price	second(avg(downloads))
0.0	1576.0
0.99	1597.0
⋮	

Note: select count(distinct propertyNo) as myCount from viewing where viewDate Between '1-May-01' AND '31-May-01';

sol:

myCount
2

good question from Record.

Assign-1;
Ages 1 2; 12, 21, 28, 29,
" 3; 4, 9, 13, 14, 16
" 4;

29/1/24

19

① add_months select add_months(sysdate, 4) from emp;

② last day select last_day(sysdate) from dual;

③ Next_day select next_day(sysdate, 'saturday') from dual;

④ round → select ~~round(to_date~~ round(to_date('01-JUL-04'), 'YYYY') from dual;

⑤ month_between

select months_between(hiredate, sysdate) from emp;

29-JAN-18

⇒ SP TH

twenty nine * 29th 1st

→ select to_char(hiredate, 'DDSP') from emp.

hiredate: any date or date variable

DDSP → DDSPTH

29th JAN — Eighteen

DDTH — MOM — 44 SP

---

select LTRIM(ename, substr(ename, 0, 1)) from emp;

substr(col, 1, 1)

1: starting

1: length of substr

5/2/18

Network and sharing center
↳ Local Area connection
↳ properties
↳ ipv4
↳ gateway address
192.168.21.254

12/2/18

emp e empm

ename empno mgr ename empno mgr

```
select e.ename, e.empno, m.ename, m.empno
from emp e, empm
where e.mgr = m.empno;
```

```
Select job, avg(sal) from emp groupby job
having avg(sal) = ( Select max(avg(sal)) from emp
                    group by job )
```

job avg(sal)

P 5000

SQL alias(as) are used to give a table (column) or a column in a table, a temporary name.

eg; select ename as name from emp

op.
name
Ragho

eg select ename as [name of employee] from emp

op.
name of employee
Raghow.

because ename as name of employee ⇒ gives us an error. Does not provide space.

so use [ ]

---

decode (tablename/exp, conditional value, Result, ..., default value)

Repetition allowed

temporary table name.

Transaction

- It is a unit of program exec^n that accesses and possibly updates the data item
- It is initiated by user program which is
- Transaction must see consistent DB
- During trans^n exec^n the db may be temporarily inconsistent
- when a trans^n complete successfully, the db must be consistent
- After a trans^n commits, the changes it has made to db persist, even if ther are system failures.
- Two main issues to deal with
  - i) various kind of failures (eg, ~~syst failure~~, h/w failures, syst crashes)
  - ii) concurrent exec^n of multiple trans^n

To preserve the integrity of DB, system must ensure the following set of properties called ACID prope

- Atomicity
- Consistency
- Durability
- Isolation

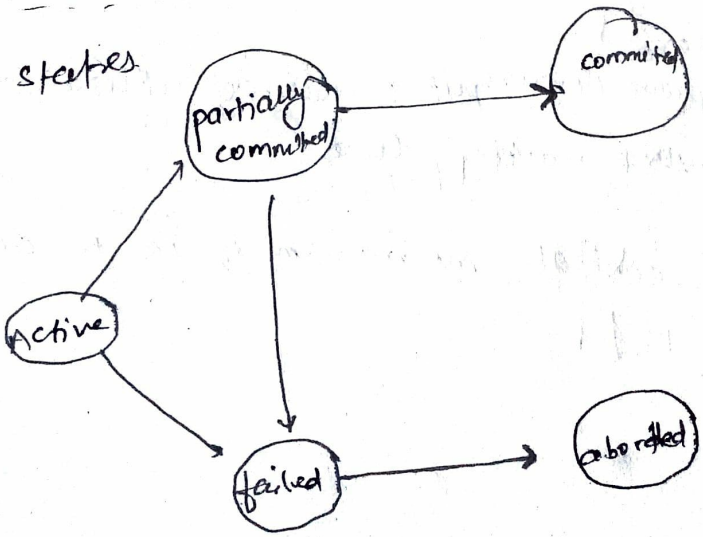
read (x): Transfer the data item from DB to local buffer
write (x): Transfer the data item from local buffer to the DB.

Requirement of ACID properties

e.g., steps:
1) read (A)
2) A := A - 50
3) write (A)
4) read (B)
5) B := B + 50
6) write (B)

→ if trans^n does not complete its exec^n successfully, then the trans^n is aborted
- ↑ when the changes has been undone by aborted trans^n, it is called trans^n rolled back
→ ↓ successfully commited trans^n is called commited.

Trans^n states



Implementation of Atomicity and durability

- the recovery-management component of the DBMS system, implements the supports of Atomicity & durability

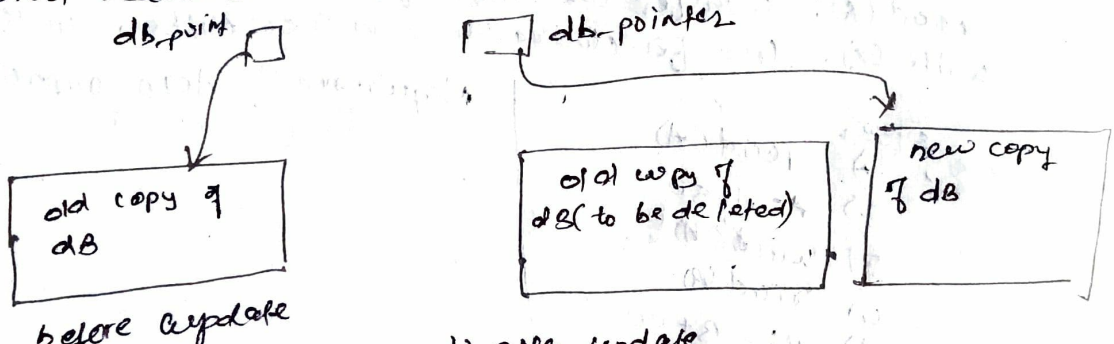
Shadow-DB scheme

- Assume only one trans^n is executing
- a pointer called db_pointer is pointing to current consistent copy of DB.
- All updates are made to shadow DB and db_pointer is made to point the shadow DB only when trans^n reaches to partially commited state and all the updated pages are flushed into disks. have been
- in case trans^n fails, old consistent copy of DB pointed by db_pointer can be used and shadow DB can be deleted

- Assume disks do not fail

→ useful for text editor, but not for DBMS but extremely insufficient for large DB because single trans^n can cause copying of entire DB

→ Do esn of handle concurrent exec^n.



a) before update

b) after update

Concurrent exec^n

- multiple . . . .
- advantage
  - improve throughput & resource utilisation
  - reduce waiting time

concurrency control - mechanism of DBMS system to achieve isolation.

Shedule – A seq. of inst^{ns} of trans^{n} that specify the chronological order ~~of inst^{n}~~ in which inst^{ns} of current trans^{n} is executed.
- A shedule of a trans^{n} must contain all the inst^{ns} of the trans^{n}
- A shedule must preserve the order of inst^{ns}

Conflict equivalent:
If a shedule S can be transformed in a shedule (S') with series of swaps of non-conflicting instruction. Then the shedule S and S' called conflict equivalent.

conflict serializable:
If a shedule (S) is a conflict equivalent to a serial shedule then it is called conflict equivalent.

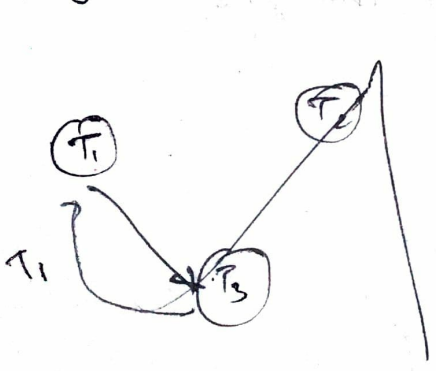
conflicting instructions
Let I_i I_j are consecutive instruction of Transaction T_i and T_j
swapping is possible if

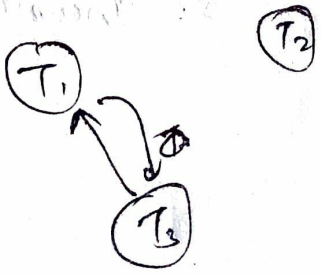
I_i = ~~read~~ $r(Q)$
I_j = $R(Q)$
} order does not matter

I_i = $R(Q)$
I_j = $w(Q)$
} order matter

I_i = $w(Q)$
I_j = $R(Q)$
} order matter

I_i = $w(Q)$
I_j = $w(Q)$
} order matter





```
SELECT SUBSTR("SQL Tutorial", 5, 3) AS ExtractString;
```

The SUBSTR() function extracts a substring from a string (starting at any position).

Note: The position of the first character in the string is 1.

1	2	3	4	5	6	7	8	9	10	11	12
S	Q	L		T	u	t	o	r	i	a	l

Note: The position of the last character in the string is -1.

-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
S	Q	L		T	u	t	o	r	i	a	l

Note: The SUBSTR() and MID() functions equals to the SUBSTRING() function.

Syntax

```
SUBSTR(string, start, length)
```

OR:

```
SUBSTR(string FROM start FOR length)
```

Q. Query the list of *CITY* names ending with vowels (a, e, i, o, u) from **STATION**. Your result *cannot* contain duplicates.

Ans -
select distinct city from station where city like '%a' or city like '%e' or city like '%i' or city like '%o' or city like '%u';

select distinct city from station where substr(city, length(city), 1) in ('a', 'e', 'i', 'o', 'u');
select distinct city from station where substr(city, -1, 1) in ('a', 'e', 'i', 'o', 'u');

upper(strng), lower(strng)

Q. Query the list of *CITY* names from **STATION** that either do not start with vowels or do not end with vowels. Your result cannot contain duplicates.

Ans -
select distinct city from station where upper(substr(city, 1, 1)) not in ('A', 'E', 'I', 'O', 'U') or substr(city, -1, 1) not in ('a', 'e', 'i', 'o', 'u');

Q. explain the follwing query
select name from students where marks > 75 order by substr(name, -3, 3), id;

Query: select ceil(avg(salary) - avg(replace(salary, 0, ''))) from employees;
Query: SELECT CAST(CEILING((AVG(CAST(Salary AS Float)) - AVG(CAST(REPLACE(Salary, 0, '')AS Float)))) AS INT)
FROM EMPLOYEES;

Q. https://www.hackerrank.com/challenges/earnings-of-employees/problem?isFullScreen=true