SQL

Lec 3 SQL1.

1. comparison with null is unknown
   arithmetic with null is null
   e.g. X is null ⇒ X<10 is unknown, X+20 is null
2. false > unknown > true
   For U→U, F & U→F, T or U→T, T and U→U
3. x is distinct from y → x,y can be null

Lec 5 SQL2

1. Foreign key constraint
   ① must refer to primary key (could be same table)
   ② value either be primary key value or null
   ③ match full allows only both value or both null

2 Deferrable Constraints (1)

```
create table  Employees (
      eid           integer primary key,
      ename         varchar(100),
      managerId     integer,
      foreign key   (managerId) references Employees
                    deferrable initially deferred
);

insert into Employees values (1, 'Alice', null), (2, 'Bob', 1), (3, 'Carol', 2);

begin;
delete from Employees where eid = 2;
update Employees set managerId = 1 where eid = 3;
commit;
```

Deferrable Constraints (2)

```
create table  Employees (
      eid           integer primary key,
      ename         varchar(100),
      managerId     integer,
      constraint employees_fkey  foreign key (managerId)  references Employees
                    deferrable initially immediate
);

insert into Employees values (1, 'Alice', null), (2, 'Bob', 1), (3, 'Carol', 2);

begin;
set constraints employees_fkey deferred;
delete from Employees where eid = 2;
update Employees set managerId = 1 where eid = 3;
commit;
```

3. Union, intersect, except. 加上all preserve duplicates
4. Subquery
   exists (subq): T if subq non-empty, F if empty
   expr in (subq): T if expr match one of subq, F if not match
   expr op any (subq): T if true, F if not true or subq empty
   expr op all (subq): T if subq empty or true, F if any one false.

Lec 6 SQL3

1. Aggregate function
   min(A), max(A), avg(A), sum(A), count(A)
   count(*), avg/sum/count(distinct A)

2. Select, having, group by property
   for each A in select or having one of following conditions must hold ① A appear in group by
   ② A as an aggregate expression
   ③ primary key appear in group by

3. Conceptual Evaluation of Queries

select	**distinct** select-list
from	from-list
where	where-condition
group by	groupby-list
having	having-condition
order by	orderby-list
offset	offset-specification
limit	limit-specification

1. Compute the cross-product of the tables in **from-list**
2. Select the tuples in the cross-product that evaluate to *true* for the **where-condition**
3. Partition the selected tuples into groups using the **groupby-list**
4. Select the groups that evaluate to *true* for the **having-condition** condition
5. For each selected group, generate an output tuple by selecting/computing the attributes/expressions that appear in the **select-list**
6. Remove any duplicate output tuples (distinct).
7. Sort the output tuples based on the **orderby-list**
8. Remove the appropriate output tuples based on the **offset-specification** & **limit-specification**

4. common table expression
   with R1 as (Q1), R2 as(Q2), ... Rn as(Qn)
   select/insert ... statement; // Rx can use all prev Ri

Case

```
select name, case
        when marks >= 70 then 'A'
        when marks >= 60 then 'B'
        when marks >= 50 then 'C'
        else 'D'
end as grade
from   Scores;
```

```
case
 when condition1 then result1
 ...
 when conditionn then resultn
 else result0
end
```

```
case expression
 when value1 then result1
 ...
 when valuen then resultn
 else result0
end
```

Coalesce

Tests

name	first	second	third
Alice	pass	null	null
Bob	fail	pass	null
Carol	fail	fail	pass
Dave	fail	fail	fail
Eve	fail	fail	null

name	result
Alice	pass
Bob	pass
Carol	pass
Dave	fail
Eve	fail

```
select name,
       coalesce(third, second, first)as result
from Tests;
```

- Returns the first non-null value in its arguments
- Null is returned if all arguments are null

10

Nullif

Tests

name	result
Alice	absent
Bob	fail
Carol	pass
Dave	absent
Eve	pass

name	status
Alice	null
Bob	fail
Carol	pass
Dave	null
Eve	pass

```
select name,
       nullif(result,'absent') as status
from Tests;
```

- nullif(value1, value2)
- Returns null if value1 is equal to value2;
- otherwise returns value1

Like

Customers

cname	area
Homer	West
Lisa	South
Maggie	East
Moe	Central
Ralph	Central
Willie	North

cname
Maggie
Willie

```
select cname from Customers where
cname like '_ _ _%e';
```

- Underscore _ matches any single character.
- Percent % matches any sequence of 0 or more characters.

11

Universal Quantification

Find the names of all students who have enrolled in all the courses offered by CS department.

For each student,
there should not be any CS course
that the student has not enrolled in.

```
select name from Students S
where not exists (
        select courseId
        from Courses C
        where dept = 'CS'
        and not exists (
                select 1
                from Enrolls E
                where E.cid = C.courseId
                and E.sid = S.studentId
        )
);
```

12

PL/SQL Lec 8

1. FOR i IN REVERSE 10..1 LOOP -- END LOOP;
2. Cursor
   curs CURSOR FOR (select * from Scores order by Mark);
   r RECORD;
   FETCH curs INTO r; RETURN NEXT;
   fetch PRIOR FROM : prior row
   FIRST FROM / LAST FROM
   ABSOLUTE 3 FROM: the 3-rd tuple

Lec 9 Trigger

1. Trigger can access TG_OP, TG_TABLE_NAME, OLD, NEW
2. Trigger function should be created b4 trigger

Row-Level Trigger	INSERT t_1	UPDATE t_1	DELETE t_1
BEFORE, RETURN t_2	INSERT t_2	UPDATE t_2	DELETE t_1
BEFORE, RETURN Null	No INSERT	No UPDATE	No DELETE
AFTER	INSERT t_1	UPDATE t_1	DELETE t_1

3. BEFORE INSERT ON Scores
   FOR EACH ROW EXECUTE FUNCTION for_elise_func();
   trigger at different levels : ROW / STATEMENT → should return null but no effect
   trigger timing : instead of only for row-level
4. CREATE constraint TRIGGER -- AFTER
   DEFERRABLE initially immediate
   for each row ... ;
   BEGIN; set constraint deffered; COMMIT;

FD & NF

Lec 10 FD

1. Armstrong's axioms
   ①. Reflexivity : ABC → AB (set → subset)
   ②. Augmentation: A→B then AC→BC
   ③. Transitivity : A→B and B→C then A→C
   additional rules
   ④. decomposition: A→BC then A→B, A→C
   ⑤. union : A→B and A→C then A→BC
2. closure
   if $\{X\}^+$ contains Y, then X→Y

Lec 11 BCNF

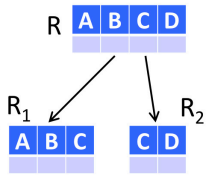
1. non-trivial FD : RHS is not a subset of LHS
   decomposed FD : RHS consists of only 1 attr
   how to find non-trivial & decomposed FD for a table?
   ①. for every subset, find its closure
   ②. for each closure, remove trivial attr on RHS
   ③. derive the non-trivial and decomposed FD
2. BCNF :
   every non-trivial and decomposed FD has a superkey as its LHS.
   how to check?
   ①. Find keys of R (by closure)
   ②. Find n-t and d FDs of R (by closure).
   ③. check LHS are all superkeys
3. BCNF, an easier way to check
   if exists subset $A=\{A_1, A_2 \cdots A_m\}$ such that
   $\{A\}^+$ contains more but not all, then R not in BCNF
4. BCNF decomposition
   ①. Find "more but not all" set X.
   ②. decompose R into R_1 and R_2
   R_1 contains all attr in X^+ $\{a,b,c\}$
   R_2 contains X and attr not in X^+ $\{a,d\}$
   ③. repeat ①, ② until no "more but not all"

Lec 12 3NF

1. 3NF Definition:
   iff. all n-t and d FDs satisfy
   either ①. LHS is a superkey
   or ②. RHS is a prime attr (i.e. appear in a key)
2. Good : preserve all FDs.
3. an easier way to check
   ①. Find "more but not all" subset A
   ②. if $\{A\}^+$ has an attr that not in A and not a prime
   if such A exists, R not in 3NF
4. 3NF Decomposition

3NF Decomposition Algorithm

- Given: A table R, and a set S of FDs
  - e.g., R(A, B, C, D)
    S = {A→BD, AB→C, C→D, BC→D}
- Step 1: Derive a minimal basis of S
  - e.g., a minimal basis of S is
    {A→B, A→C, C→D}
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
  - e.g., after combining A→B and A→C, we have {A→BC, C→D}
- Step 3: Create a table for each FD remained
  - R_1(A, B, C), R_2(C, D)
- Step 4: If none of the tables contains a key of the original table R, create a table that contains a key of R (any key would do)



5. minimal basis S'
   ①. every FD in S' can be derived from S, and vice versa
   ②. FD in S' is n-t and d
   ③. no FD in S' is redundant
   ④. for each FD, none of the attr on LHS is redundant

Algorithm for Minimal Basis

- Step 1: Transform the FDs, so that each right hand side contains only one attribute
- Step 2: Remove redundant attributes on the left hand side of each FD
- Step 3: Remove redundant FDs