SQL Notes

- Went through major standardizations which led to its wide acceptance
  - SQL-86 (SQL)
    - Queries, basic definitions & manipulation
  - SQL-89
    - Referential integrity
  - SQL-92
    - Revised & Expanded
  - SQL-99
    - Archive rules & triggers, some recursive operations (!), object-oriented features

REVIEW: Examples (8): A new kind of query

EMP(EMPNO, NAME, DNO, JOB, MGR, SAL, COMMISSION)
DEPT(DNO, DNAME, LOC)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3.

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3.

REVIEW: Examples (8): continued (exists, ∃)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3. Answer: D1 and D2

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	
EMP(EMPNO, … ,	DNO,	JOB,	. . .)
500	D1	plumber	
900	D2	electrician	

$\pi_{DNO}(\pi_{JOB}(\sigma_{DNO = D3}(EMP)) * EMP)$

What about department D3?

REVIEW: Examples (8): continued (for all, ∀)

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

EMP(EMPNO, … ,	DNO,	JOB,	. . .)
100	D3	electrician	
200	D3	plumber	
300	D3	electrician	
400	D1	electrician	
500	D1	plumber	
600	D1	carpenter	
700	D2	electrician	
800	D2	carpenter	
900	D2	electrician	

SQL: Examples (8): continued (exists, ∃)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3. Answer: D1 and D2

```
select  DNO
from    DEPT
where exists
        (  select    *
           from      EMP  ED3
           where     ED3.DNO = 'D3'
            and  exists
                 (  select    *
                    from      EMP  EY
                    where     EY.JOB = ED3.JOB
                     and      EY.DNO = DEPT.DNO))
```

The order of the two "selects" does not matter.

Still gets D3.

SQL: Examples (8): continued (exists, ∃)

Q11. Find the numbers of those departments that have employees who can do ***some*** job that is done by an employee in department D3. Answer: D1 and D2

```
select  DNO
from    DEPT
where exists
          ( select  *
            from    EMP ED3
            where   ED3.DNO = 'D3'
             and  exists
                  (select *
                   from        EMP  Y
                   where       EY.JOB = ED3.JOB
                    and        EY.DNO = DEPT.DNO) )
      and  DNO <> 'D3'
```

Eliminate department D3.

SQL: Examples (8): continued (for all, ∀)

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

```
select  DNO
from    DEPT
where for all
        (  select   *
           from     EMP  ED3
           where    ED3.DNO = 'D3'
            and  exists
                    (  select   *
                       from     EMP  EY
                       where    EY.JOB = ED3.JOB
                        and     EY.DNO = DEPT.DNO))
```

However no for all exists in SQL.

The trick from logic.

For all X there exists a Y
such that P(X,Y) is true.
is equivalent to
There does not exist an X for which there does not exist a Y
such that P(X,Y) is true.

∀ X ∃ Y such that P(X,Y)
is equivalent to
~∃ X ~∃ Y such that P(X,Y)

SQL: Examples (8): continued (for all, ∀)

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

```
select  DNO
from    DEPT
where not exists
        (  select   *
           from     EMP  ED3
           where    ED3.DNO = 'D3'
            and  not exists
                    (  select   *
                       from     EMP  EY
                       where    EY.JOB = ED3.JOB
                        and     EY.DNO = DEPT.DNO))
```

However no for all exists in SQL.
Use two not exists.

SQL: Examples (8): continued (for all, ∀)

Q12. Find the numbers of those departments that have employees who can do ***all*** the jobs that are done by an employee in department D3. Answer: D1, but not D2

```
select  DNO
from    DEPT
where not exists                                  Eliminate department D3.
        ( select    *
          from      EMP  ED3
          where     ED3.DNO = 'D3'
            and  not exists
                  ( select    *
                    from      EMP  EY
                    where     EY.JOB = ED3.JOB
                      and     EY.DNO = DEPT.DNO) )
      and  DNO <> 'D3'
```


SQL: for all (∀) using count Function First Attempt:

```
select  DNO
from    DEPT D
where ( select    count(distinct JOB)          D3JOBS: Jobs done by employees in department D3.
        from      EMP  ED3
        where     ED3.DNO = 'D3')
          =
      ( select    count(distinct JOB)          DJOBS: Jobs done by employees in department D.
        from      EMP  EY
        where     EY.DNO = D.DNO)
```


SQL: for all (∀) using count Function Second Attempt:

```
select  DNO
from    DEPT D
where ( select    count(distinct JOB)          D3JOBS: Jobs done by employees in department D3.
        from      EMP  ED3
        where     ED3.DNO = 'D3')
          =
      ( select    count(distinct EY.JOB)       DJOBS: Jobs done by employees in department D that are also done by employees in department D3.
        from      EMP  EY, EMP ED3
        where     EY.DNO = D.DNO
          and     EY.JOB = ED3.JOB
          and     ED3.DNO = 'D3')
```

Why does this approach work?

The where clause is applied to a specific department D.

```
select  DNO
from    DEPT D
where ( select  count(distinct JOB)
        from    EMP  ED3
        where   ED3.DNO = 'D3')
          =
      ( select  count(distinct EY.JOB)
        from    EMP  EY, EMP ED3
        where   EY.DNO = D.DNO
          and   EY.JOB = ED3.JOB
          and    ED3.DNO = 'D3')
```

D3JOBS: Jobs done by employees in department D3.

DJOBS: Jobs done by employees in department D that are also done by employees in department D3.

Works if DJOBS $\subseteq$ D3JOBS.

- Why is this a general rule?
- Why is the rule satisfied in this case?