Manipulating NULL Values

- NULL values must be considered explicitly
  - IS NULL and IS NOT NULL
- NULL in a condition yields UNKNOWN
- SQL provides operators to test for specific conditions
  - IS FALSE and IS NOT FALSE
  - IS TRUE and IS NOT TRUE
  - IS UNKNOWN and IS NOT UNKNOWN
- Query: ?

```
SELECT SID, Name
FROM STUDENT AS S
WHERE ((S.Major= 'CS') and (S.Gender = 'F' )) IS NOT FALSE;
```


Truth Tables

X	NOT
TRUE	FALSE
UNK	UNK
FALSE	TRUE

OR	TRUE	UNK	FALSE
TRUE	TRUE	TRUE	TRUE
UNK	TRUE	UNK	UNK
FALSE	TRUE	UNK	FALSE

AND	TRUE	UNK	FALSE
TRUE	TRUE	UNK	FALSE
UNK	UNK	UNK	FALSE
FALSE	FALSE	FALSE	FALSE

SQL CASE Statement & NULL

- Implements if-then-else functionality
- Easy way to handle NULLs
- Simple expression on equality:

```
SELECT SID, CASE Major
            WHEN IS NULL THEN 'undecided'
            WHEN 'CS' THEN 'good choice'
            ELSE 'recruit'
            END AS Strategy
FROM STUDENT
WHERE CLASS = 'Sophomore';
```

- ELSE is Optional; all SIDs with unmatched Major are shown

SQL CASE Statement & NULL…

- Search, complex expression and beyond equality

```
SELECT SID, CASE
            WHEN Major = 'CS' THEN 'good choice'
            WHEN Major IS NULL AND QPA > 3.25 THEN 'go after'
            WHEN Major IS NULL AND QPA > 2.75 THEN 'recruit'
            ELSE 'ignore'
            END AS Strategy
FROM STUDENT
WHERE CLASS = 'Sophomore';
```

- Alias for CASE is optional
- The value of the THEN could be of any type

Basic *SQL*: Two Table Manipulation

```
SELECT    [DISTINCT] attribute-list | *
FROM      table1, table2
WHERE     join-condition & selection-condition
```

- *Cartesian product: table1* x *table2* if no *Joint-Condition*
- *Joint-Condition:* Similar to selection-condition
  - *Expression-table1* **op** *expression-table2*
  - **op** ∈ {<,<=,=,>,>=,<>}
  - combined using **AND**

Relational Operators in SQL

- STUDENT(SID, Name, Major)
- ENROLLS(CID, SID, Term, Grade)
- STUDENT X ENROLLS

```
SELECT  STUDENT.*, ENROLLS.*
FROM    STUDENT, ENROLLS;
```

- STUDENT $\bowtie_{SID=SID}$ ENROLLS

```
SELECT  S.*, E.*
FROM    STUDENT S, ENROLLS E
WHERE   S.SID  = E.SID;
```


Join Operator (SQL2)

- JOIN was introduced for specifying the join conditions in the FROM-clause:

  table1 **JOIN** table2 **ON** join-condition

- Example of Condition-Join:

```
SELECT  SID, S.Name, Term
FROM    (STUDENT S JOIN ENROLLS E ON S.SID=E.SID)
WHERE   E.CID = 1555;
```


Natural Join

- NATURAL JOIN (without ON-clause)

  table1 **NATURAL JOIN** table2

- Use renaming of attribute if there is a need, e.g.,

```
SELECT  *
FROM    (LIBRARIAN NATURAL JOIN SECTION AS
                                S(SNO, SName, Head))
WHERE   SName = 'Children';
```

- Natural join over some attributes: **USING** (attribute-list)

```
SELECT  SID, SName, Term
FROM    (STUDENT NATURAL JOIN ENROLLS USING (SID))
WHERE   ENROLLS.CID = 1555;
```

Other Join Operations

- Outer Join operators:
  - LEFT OUTER JOIN or LEFT JOIN
  - RIGHT OUTER JOIN or RIGHT JOIN
  - FULL OUTER JOIN or FULL JOIN
  - NATURAL LEFT OUTER JOIN or NATURAL LEFT JOIN
  - NATURAL RIGHT OUTER JOIN or NATURAL RIGHT JOIN
  - NATURAL FULL OUTER JOIN or FULL JOIN
- CROSS JOIN: generates a cross product
- UNION JOIN: Outer Union operator

Outer Join Examples

- STUDENT(SID, Name, Class, Major)
  ENROLLS(CID, SID, Term, Grade)
- Q1:

```
SELECT *
FROM   (STUDENT S LEFT OUTER JOIN ENROLLS E
                                  ON S.SID=E.SID)
ORDER BY S.SID;
```

- Q2:

```
SELECT SID, S.Name, S. Major
FROM   STUDENT S NATURAL LEFT OUTER JOIN ENROLLS E
WHERE  E.Term IS NULL;
```


Outer Join Q1 Execution

Students

SID	Name	Class	Major
123	John	3	CS
124	Mary	3	CS
999	Newman	1	CS

Enroll

SID	CID	Term	Grade
123	CS1520	Fall 10	3.75
124	CS1520	Fall 10	4
123	CS1555	Fall 10	4
124	CS1555	Fall 10	NULL

Q1 RESULT

S.SID	S.Name	S.Class	S. Major	E.SID	E.CID	E.Term	E.Grade
123	John	3	CS	123	CS1520	Fall 10	3.75
123	John	3	CS	123	CS1555	Fall 10	4
124	Mary	3	CS	124	CS1520	Fall 10	4
124	Mary	3	CS	124	CS1555	Fall 10	NULL
999	Newman	1	CS	NULL	NULL	NULL	NULL

Outer Join Q2 Execution

```
SELECT SID, S.Name, S. Major
FROM   STUDENT S NATURAL LEFT OUTER JOIN ENROLLS E
WHERE  E.Term IS NULL;
```

Students

SID	Name	Class	Major
123	John	3	CS
124	Mary	3	CS
999	Newman	1	CS

Enroll

SID	CID	Term	Grade
123	CS1520	Fall 10	3.75
124	CS1520	Fall 10	4
123	CS1555	Fall 10	4
124	CS1555	Fall 10	NULL

Q2 RESULT

S.SID	S.Name	S. Major
999	Newman	CS

Set Operations

- SQL supports UNION, EXCEPT (difference), INTERSECT (not all vendors)
- UNION ALL retains duplicates
- Tables must be *union-compatible*

```
( SELECT SID
  FROM   STUDENT
  WHERE  Major = 'CS' )
UNION
( SELECT SID
  FROM   STUDENT
  WHERE  Major = 'Math' );
```

```
( SELECT SID
  FROM   STUDENT)
EXCEPT
( SELECT SID
  FROM   STUDENT
  WHERE  Major = 'Math' );
```


Merging Fields in Queries

- String Concatenation is denoted by two vertical bars (||)
  - || merges into a single string, one or more strings
- E.g., Display in a single value (one column) the name of all students in CS 2550 and their phone numbers

```
SELECT  Fname || ' ' || Lname AS Name, PhoneNumber
FROM    STUDENT NATURAL JOIN ENROLLS
WHERE   Dept || CourseNumber = 'CS2550';
```


Range Queries & Range Conditions

- BETWEEN, and its negation NOT BETWEEN, can be used with numeric, character and datetime datatypes
- Simplify the formulation of conjunction expressions
- E.g.,

```
SELECT *
FROM   LIBRARIAN
WHERE  (Salary >= 25000 AND Salary <= 35000);

SELECT *
FROM   LIBRARIAN
WHERE  (Salary BETWEEN 25000 AND 35000);
```


Partial Queries - Pattern Matching

- **LIKE / NOT LIKE** support comparisons with partial strings
  - A percent sign `%' indicates a match with an arbitrary number of characters including spaces
    - Note that '*' is not valid
  - An underscore sign '_' matches a single arbitrary character
- Retrieve all students with Pitt phone extension

```
SELECT  Name
FROM    STUDENT
WHERE   Phone  LIKE  '412.62%';
```

Pattern Matching…

- Retrieve all students with *local* phone numbers (any area code) which start with 6 and whose third digit is 3.

```
SELECT  Name
FROM    STUDENT
WHERE   Phone LIKE '___6_3%';
```

- Escape defines the escape character that causes SQL to interpret a wildcard char (%) as itself in a string:

```
SELECT  VideoName
FROM    RENTALS
WHERE   Discount LIKE '10&%' ESCAPE '&';
```


Regular Expressions

- **SIMILAR TO / NOT SIMILAR TO** support complex pattern matches a given string
- Retrieve all students with Pitt phone extension

```
SELECT  Name
FROM    STUDENT
WHERE   Phone SIMILAR TO '412.6(2|4)%';
```

- **https://www.postgresql.org/docs/9.3/functions-matching.html**

Metacharacters from POSIX

Meta-characters	Meaning	Examples
\	the match character is a special character, a literal, or a backreference	\n matches newline, \\ matches \, \(matches (
^	Matches the position at the start of the string	^A matches A if A is the 1st char in the string
$	Matches the position at the end of the string	$B matches B if B is the last char in the string
*	Matches the preceding character zero or more times	ba*rk matches brk, bark, baark, etc.
+	Matches the preceding character one or more times	ba+rk matches bark, baark, etc., but not brk.
?	Matches the preceding character zero or one time	ba?rk matches brk and bark only

Metacharacters from POSIX

Meta-characters	Meaning	Examples
{n}	Matches a character exactly n times, where n is an integer	hob{2}it matches hobbit
{n,m}	Matches a character at least n times and at most m times, where n and m are both integers	hob{2,3}it matches hobbit and hobbbit only
.	Matches any single character except null	hob.it matches hobait, hobbit, etc.
(pattern)	A subexpression that matches the specified pattern	anatom(ylies) matches anatomy and anatomies
xly	Matches x or y, where x and y are one or more characters	warlpeace matches war or peace
[abc] or [a-z]	Matches any of the enclosed characters or the specified range	[ab]bc matches abc and bbc; [a-c]bc matches abc, bbc, and cbc

Challenging yet Common Query

- Assume ENROLL(SID, CID, score)
- Find the ranking of students in CS2550 according to their scores. Your results should consider the case of tie.
  - E.g., output when 007 & 009 both received the highest score.

SID	Rank
007	1
009	1
003	3
005	4

Nested Queries & Set Comparisons

- One of the most powerful features of SQL
- Two definitions of a set:
  - Explicit: list the members of the set within ( )
    E.g., (1, 2, 3) and ('Science', 'Art', 'Children')
  - Implicit: define it as a subquery (nested query) whose output table (set) can appear in any clause in place of a table
    - can be used at the FROM-clause
    - can be used at the WHERE-clause by the selection condition of the *outer* SELECT statement

Scalar Subquery

- Scalar subquery: It is an inner query whose output is a single column and a single row
- This can be used in any expression in which a single value may appear
- E.g., Q1:

```
SELECT SID, Name
FROM   STUDENT
WHERE  SID = ( SELECT F.SID
               FROM   STAFF AS F
               WHERE  F.SSN = '132-32-2222' );
```


Another Example of Scalar Subquery

Q2: ?

```
SELECT S.SID, (SELECT DISTINCT MAX(Grade)
               FROM ENROLL E
               WHERE E.SID=S.SID) AS LG
FROM STUDENT S
WHERE S.Major = 'CS' ;
```