



CSE460/560 DATA MODELS AND QUERY LANGUAGES

Structured Query Language (SQL) - Basic

Cheng-En Chuang

(Slides Adopted from Jan Chomicki and Ning Deng)



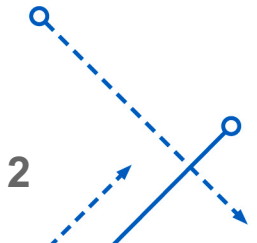
University at Buffalo

Department of Computer Science
and Engineering

School of Engineering and Applied Sciences

Outline

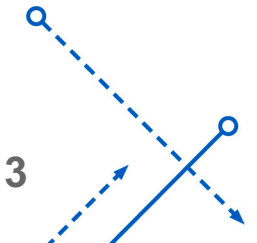
1. Introduction
2. DDL in SQL
3. DML in SQL
4. Basic SQL Queries





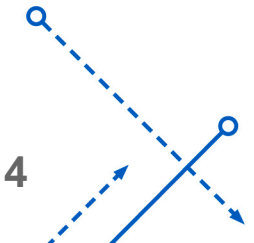
Outline

1. Introduction
2. DDL in SQL
3. DML in SQL
4. Basic SQL Queries



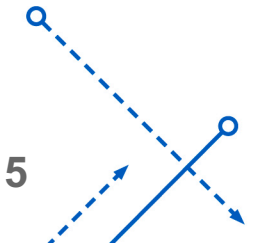
SQL

- Support
 - Virtually all relational DBMS
 - Vender-Specific Extensions
- Standardization
 - SQL2 (SQL-92)
 - SQL3 (SQL:1999)
 - SQL:2003 (revised SQL:1999)
 - SQL:2006
 - SQL:2011
 - SQL:2016



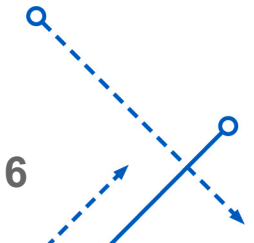
SQL Components

- Query language
- DDL
- DML
- Integrity constraints and views
- API's (ODBC, JDBC)
- Host language preprocessors (Embedded SQL)
- Support XML data and queries



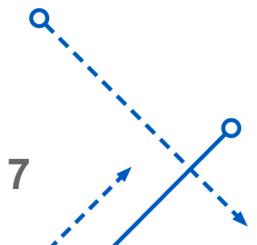
Outline

1. Introduction
- 2. DDL in SQL**
3. DML in SQL
4. Basic SQL Queries



Data Definition Language (DDL)

- Create relation schema
- Alter relation schema
- Drop relation schema

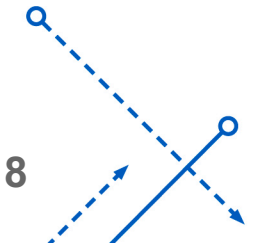


CREATE TABLE Statement

```
CREATE TABLE tbl-name(
    Attr1 Type1 LC1,
    Attr2 Type2 LC2,
    ...
    Attrn Typen LCn,
    GC1, GC2, ..., GCk
);
```

LC1 ... LCn: Local constraints
GC1 ... GCk: Global constraints

```
CREATE TABLE Student(
    SID VARCHAR(10),
    NAME VARCHAR(20) NOT NULL,
    DOB DATE NOT NULL,
    GENDER CHAR(1),
    PRIMARY KEY(SID)
);
```



ALTER TABLE Statement

Add a column:

```
ALTER TABLE reln_name ADD Attrk Typek LCk;
```

Drop a column:

```
ALTER TABLE reln_name DROP COLUMN Attrk;
```

Change column data type:

```
ALTER TABLE reln_name ALTER COLUMN Attrk Typek;
```

Change column local constraint:

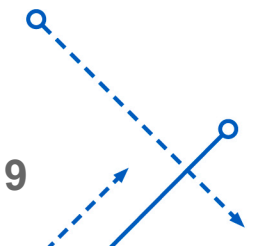
```
ALTER TABLE reln_name MODIFY COLUMN Attrk LC1;
```

Add a table constraint:

```
ALTER TABLE reln_name ADD CONSTRAINT Gck;
```

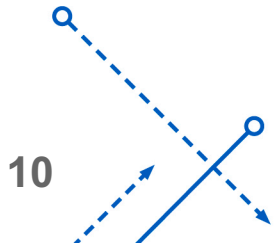
Drop table constraint:

```
ALTER TABLE reln_name DROP CONSTRAINT Gck;
```



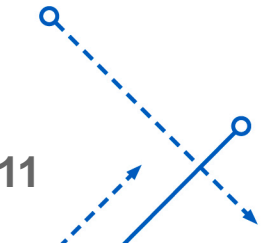
Outline

1. Introduction
2. DDL in SQL
- 3. DML in SQL**
4. Basic SQL Queries



Data Manipulation Language (DML)

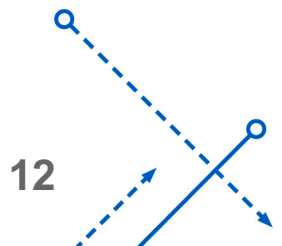
- Adding rows
- Deleting rows
- Update rows



INSERT INTO Statement

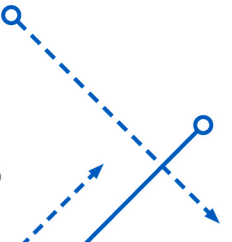
```
INSERT INTO reln_name (Attr1, Attr2, ...)  
VALUES (v1, v2, ...);
```

```
INSERT INTO reln_name  
VALUES (v1, v2, ...);
```



UPDATE Statement

```
UPDATE reln_name  
SET Attr1=v1, Attr2v2, ...  
WHERE C;
```



DELETE Statement

```
DELETE FROM reln_name  
WHERE C;
```

Outline

1. Introduction
2. DDL in SQL
3. DML in SQL
4. **Basic SQL Queries**



Basic SQL Query

- Basic Form
 - $SELECT A_1, \dots, A_n FROM R_1, \dots, R_k WHERE C;$
- Corresponding RA Expression
 - $\pi_{A_1, \dots, A_n}(\sigma_C(R_1 \times \dots \times R_k))$
- Example
 - $SELECT FirstName$
 $FROM Student$
 $WHERE Student.GPA > 3.5;$

Range Variables (Table Aliases)

- To refer to a relation more than once in the FROM clause
 - Range variables/aliases are used
- Example
 - Customer(cid, city)
 - Customers that are from the same city
 - ```
SELECT A.cid AS customer1, B.cid AS customer2, A.city
FROM Customer A, Customer B
WHERE A.cid <> B.cid
 AND A.city = B.city
```
- Corresponds to
- $$\rho_{customer1, customer2, city}(\pi_{cid, cid2, city} \left( Customer \bowtie_{cid \neq cid2 \wedge city = city2} \left( \rho_{cid2, city2}(Customer) \right) \right))$$



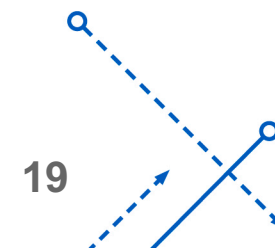
# Manipulating the Result

- SELECT \*: all columns are selected
- SELECT DISTINCT: duplicates are eliminated from the result
- ORDER BY  $A_1, \dots, A_m$ : the result is sorted according to  $A_1, \dots, A_m$
- $E \text{ AS } A$ 
  - Refer the result of  $E$  as  $A$



# Set Operations

- UNION
- INTERSECT
- EXCEPT



# UNION

- Computes the union of two compatible relations

- Example

- ```
SELECT S.Sid, S.Firstname
FROM Student S
WHERE S.Firstname='John'
UNION
SELECT S2.Sid, S2.Firstname
FROM Student S2
WHERE S2.Firstname='Mary';
```

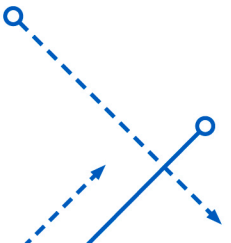
- Same as

- ```
SELECT S.Sid, S.Firstname
FROM Student S
WHERE S.Firstname='John'
OR S.Firatname='Mary';
```



# INTERSECT

- Computes the intersection of Two compatible relations
- Example
  - ```
SELECT S.Sid, S.Firstname  
FROM Student S  
WHERE S.Firstname ='John'  
INTERSECT  
SELECT S2.Sid, S2.Firstname  
FROM Student S2  
WHERE S2.GPA=4.0;
```



EXCEPT

- Takes input of two compatible relations
 - Computes the tuples in the first relation
 - But NOT in the second relation
- Example
 - ```
SELECT S.Firstname
FROM Student S
WHERE S.GPA < 3.5
EXCEPT
SELECT S2.Firstname
FROM Student S2
WHERE S2.Firstname='John';
```

# Recommended Reading

Database Systems: The Complete Book  
Chapter 6.1 – 6.5