### **Basic SQL**

Instructor: Wang-Chiew Tan

Reference: A First Course in Database Systems, 3<sup>rd</sup> edition, Chapter 2.3

9

### What is a Data Model?

- A data model is a mathematical formalism that consists of two parts:
  - A notation for describing data and mathematical objects for representing data.
  - 2. A set of operations for manipulating data.
- The relational data model
- What is the associated query language for the relational data model?

# Two different query languages

- Codd proposed two different query languages for the relational data model.
  - Relational Algebra
    - Queries are expressed as a sequence of operations on relations.
    - Procedural language.
  - Relational Calculus
    - · Queries are expressed as formulas of first-order logic.
    - Declarative language.
- Codd's theorem: The Relational Algebra query language has the same expressive power as the Relational Calculus query language.

11

# Procedural vs. Declarative Languages

- Procedural program:
  - The program is specified as a sequence of operations to obtain the desired the outcome. I.e., how the outcome is to be obtained.
  - E.g., Java, C, ...
- Declarative program:
  - The program specifies what is the expected outcome, and not how the outcome is to be obtained.
  - E.g., Scheme, Ocaml, ...

# An example

• In mathematical terms, the sequence F<sub>n</sub> of Fibonacci numbers is defined by the recurrence relation

$$\begin{aligned} \mathbf{F_n} &= \mathbf{F_{n-1}} + \mathbf{F_{n-2}} \\ \text{with seed values } \mathbf{F_0} &= \mathbf{0} \text{ and } \mathbf{F_1} = \mathbf{1}. \\ F(n) &= \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ F(n-1) + F(n-2) & n > 1 \end{cases} \end{aligned}$$

• 0,1,1,2,3,5,8,13,...

13

### Fibonacci numbers in C++

#### Fibonacci numbers in Ocaml

let rec fibonacci = function
| 0 -> 0
| 1 -> 1
| n when n > 1 -> fibonacci (n-1) + fibonacci (n-2)

 NOTE: it is possible to write a program in C++ in a "declarative" way similar to the above.

15

# SQL – Structured Query Language

- Is SQL a procedural or a declarative language?
- Keep this question in mind as we learn SQL.
  - We will revisit this question later.
- SQL pronounced as "sequel"
- Principle language used to describe and manipulate data stored in relational database systems.
- Neither relational algebra nor relational calculus proposed by Codd.
- Current standard: SQL-2011

### SQL DDL and SQL DML

- Two main aspects to SQL:
  - Data Definition Language (DDL)
    - Sublanguage of SQL used to create, delete, modify the definition of tables and views.
    - For declaring database schemas.
  - Data Manipulation Language (DML)
    - Sublanguage of SQL that allows users to insert, delete, and modify rows of tables and pose queries.
    - For asking questions about the database and modifying the database.

17

### Relations in SQL

- Three types of relations
  - 1. Stored relations (or tables)
    - These are tables that contain tuples and can be modified or queried.
  - 2. Views
    - Views are relations that are defined in terms of other relations but they are not stored. They are constructed only when needed.
  - 3. Temporary tables
    - These are (intermediate) tables that are constructed by the SQL execution engine during the processing of SQL queries and discarded when done.

# **Atomic Data Types**

- CHAR(n): fixed-length string of up to n characters.
- VARCHAR(n): also a string of up to n characters.
- BIT(n)
- BIT VARYING(n)
- BOOLEAN: true/false/unknown.
- INT or INTEGER
  - Analogous to int in C.
- SHORTINT
  - Analogous to short int in C.
- FLOAT or REAL
- DOUBLE PRECISION
  - Analogous to double in C.
- DATE and TIME
  - Character strings of a special form.

19

# Defining a table

#### **CREATE TABLE** Movies (

);

```
title CHAR(100),
year INT,
length INT,
genre CHAR(10),
studioName CHAR(30),
producerC# INT
```

title year length genre studioName producerC#

# Defining a table (cont'd)

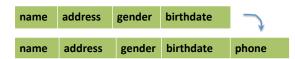
```
CREATE TABLE MovieStar (
name CHAR(30),
address VARCHAR(255),
gender CHAR(1),
birthdate DATE
);
```

name address gender birthdate

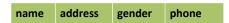
21

# **Modifying Relation Schemas**

- DROP TABLE Movies;
  - The entire table is removed from the database schema.
- ALTER TABLE MovieStar ADD phone CHAR(16);
  - Adds an attribute "phone" with type CHAR(16) to the table MovieStar.



ALTER TABLE MovieStar DROP birthdate;



#### **Default Values**

```
CREATE TABLE MovieStar (
name CHAR(30),
address VARCHAR(255) DEFAULT "Hollywood",
gender CHAR(1),
birthdate DATE,
);
```

- When a new tuple is inserted and no value is specified for the attribute address, then the value for this attribute will default to "Hollywood".
- If no default value is declared explicitly, then the value of the attribute will default to NULL.
  - NULL is a special value in SQL to represent unknown values.

23

# More examples on default values

```
CREATE TABLE MovieStar (
   name
                   CHAR(30),
  address
                   VARCHAR(255)
                                   DEFAULT
                                               'Hollywood',
                                               '?',
  gender
                   CHAR(1)
                                   DEFAULT
   birthdate
                   DATE
                                               '0000-00-00',
                                   DEFAULT
);
```

ALTER TABLE MovieStar ADD phone CHAR(16) DEFAULT 'unlisted';

### Keys



- A key constraint (or a key in short) of a relation schema R is a subset K of attributes of R such that
  - 1. For every instance r of R, every two distinct tuples of r must differ in their values from K.
    - Contrapositive: if two tuples agree on their values from K, then they must be the same tuple.
  - 2. Minimal: no proper subset of K has the above property.
- A *superkey* is a set of attributes of R that includes a key of R.
  - Fact. All keys are superkeys but some superkeys are not keys.

25

### **Examples**

- Student(studentID, name, major, gender, avgGPA).
  - {studentID} is a key. It is also a superkey.
  - {studentID, name} is a superkey but not a key.
  - {studentID, name, major, gender, avgGPA} is a superkey but not a key.









- One key is chosen and define as the *primary key*, while the rest are *candidate keys*.
- Student(studentID, name, dob, major, gender, avgGPA)
  - {studentID}, {name, dob} are keys and also superkeys.
    - {studentID} is the primary key.
    - {name, dob} is the candidate key.
  - {name, dob, avgGPA} is a superkey.

# **Declaring keys**

- Two ways to declare an attribute or set of attributes to be a key in the CREATE TABLE statement that defines a stored relation.
- If a single attribute is a key, it can be declared in two ways:

```
CREATE TABLE MovieStar (
   name
               CHAR(30) PRIMARY KEY,
                                           CREATE TABLE MovieStar (
   address
               VARCHAR(255),
                                               name
                                                           CHAR(30),
  gender
               CHAR(1),
                                               address
                                                           VARCHAR(255),
                                               gender
                                                           CHAR(1),
   birthdate DATE
                                               birthdate
                                                           DATE,
);
                                               PRIMARY KEY (name)
                                           );
```

27

# Declaring keys (cont'd)

• If the key consists of multiple attributes, then it can be declared as follows:

```
CREATE TABLE Movies (
title CHAR(100),
year INT,
length INT,
genre CHAR(10),
studioName CHAR(30),
producerC# INT,
PRIMARY KEY (title,year)
);
```

### PRIMARY KEY vs. UNIQUE

• In the previous examples, the keyword "PRIMARY KEY" can be replaced by "UNIQUE".

```
CREATE TABLE MovieStar (
                                     CREATE TABLE MovieStar (
    name
                CHAR(30),
                                         name
                                                     CHAR(30),
    address
                VARCHAR(255),
                                         address
                                                     VARCHAR(255),
    gender
                CHAR(1),
                                         gender
                                                     CHAR(1),
    birthdate
                DATE,
                                         birthdate
                                                     DATE,
    PRIMARY KEY (name)
                                         UNIQUE (name)
);
                                     );
```

29

### PRIMARY KEY vs. UNIQUE (cont'd)

```
CREATE TABLE MovieStar (
                                     CREATE TABLE MovieStar (
                CHAR(30),
                                                     CHAR(30),
    name
                                         name
    address
                VARCHAR(255),
                                         address
                                                     VARCHAR(255),
    gender
                CHAR(1),
                                         gender
                                                     CHAR(1),
    birthdate
                DATE,
                                         birthdate
                                                     DATE,
    PRIMARY KEY (name)
                                         UNIQUE (name)
);
                                     );
```

- None of the tuples in the MovieStar relation have null name values.
- Tuples are uniquely identified by their name values.
- There can be at most one primary key for a table.
- Tuples in MovieStar relation can contain null name values.
- Tuples in MovieStar relation with non-null name values are uniquely identified by their name values.
- There can be multiple unique constraints for a table in addition to a primary key.

# PostgreSQL Meta Commands

- \|
  - List the databases.

After you connect to a database, you can perform the following:

- \c
  - List the tables of a database.
- \d table
  - List the columns of a table.

http://www.postgresqlforbeginners.com/2010/11/interacting-with-postgresql-psql.html

31

### SQL DDL and SQL DML

- Two main aspects to SQL:
  - Data Definition Language (DDL)
  - Sublanguage of SQL used to create, delete, modify the definition of tables and views.



- For declaring database schemas.
- Data Manipulation Language (DML)
  - Sublanguage of SQL that allows users to insert, delete, and modify rows of tables and pose queries.
  - For asking questions about the database and modifying the database.

Reference: A First Course in Database Systems, 3<sup>rd</sup> edition, Chapter 6.1

# Database schema for our running examples

• Let us assume we have the following database schema with five relation schemas.

Movies(title, year, length, genre, studioName, producerC#)
StarsIn(movieTitle, movieYear, starName)
MovieStar(name, address, gender, birthdate)
MovieExec(name, address, cert#, netWorth)
Studio(name, address, presC#)

33

# A simple query in SQL

• Find all movies produced by Disney Studios in 1990.

SELECT \*

**FROM** Movies

WHERE studioName = 'Disney' AND year = 1990;

# Basic form of a SQL query

Basic form:

```
SELECT [DISTINCT] c_1, c_2, ..., c_m Attribute names FROM R_1, R_2, ..., R_n Relation names [WHERE condition]
```

- We will focus on one relation R<sub>1</sub> for now.
- What is the semantics (or meaning) of such a query?

35

# A simple query in SQL

• Find all movies produced by Disney Studios in 1990.

SELECT \* The symbol "\*" is a shorthand for all attributes of relations in the FROM clause.

WHERE studioName = 'Disney' AND year = 1990;

Movies	Title	Year	Length	Genre	studioName	producerC#
	Pretty Woman	1990	119	true	Disney	999
	Monster's Inc.	1990	121	true	Dreamworks	223
	Jurassic Park	1998	145	NULL	Disney	675

# A simple query in SQL (cont'd)

• Find all movies produced by Disney Studios in 1990.

SELECT \*

**FROM Movies** 

WHERE studioName = 'Disney' AND year = 1990;

Result

Title	Year	Length	Genre	studioName	producerC#
Pretty Woman	1990	119	true	Disney	999

37

# An even simpler query in SQL

Find all movies.SELECT \*

**FROM Movies** 

Equivalent to: SELECT \* FROM Movies WHERE true

Movies

S	Title	Year	Length	Genre	studioName	producerC#
	Pretty Woman	1990	119	true	Disney	999
	Monster's Inc.	1990	121	true	Dreamworks	223
	Jurassic Park	1998	145	NULL	Disney	675

# An even simpler query in SQL

Find all movies.

SELECT \*

**FROM Movies** 

n	 	4

Title	Year	Length	Genre	studioName	producerC#
Pretty Woman	1990	119	true	Disney	999
Monster's Inc.	1990	121	true	Dreamworks	223
Jurassic Park	1998	145	NULL	Disney	675

39

# An even simpler query in SQL (cont'd)

Return the title and year of all movies.

SELECT title, year 🧸

**FROM Movies** 

A Projection Query:

• Only a subset of attributes from the relation(s) in the FROM clause is selected.

Result

Title	Year
Pretty Woman	1990
Monster's Inc.	1990
Jurassic Park	1998

# An even simpler query in SQL (cont'd)

Return the title and year of all movies produced by Disney Studios in 1990.
 SELECT title, year

**FROM Movies** 

WHERE studioName = 'Disney' AND year = 1990;

Result	Title	Year
	Pretty Woman	1990

41

# An even simpler query in SQL

Return the years of all movies with length less than 140.

FROM Movies
WHERE length < 140;

SELECT year

SELECT **DISTINCT** year FROM Movies

WHERE length < 140;

Result **Year**1990
1990

Result **Year** 1990

Multiset or bag semantics

Set semantics

# Bags (or multisets) vs. Sets

- From your basic set theory
- Every element in a set is distinct
  - E.g., {2,4,6} is a set but {2,4,6,2,2} is not a set.
- A bag (or multiset) may contain repeated elements.
  - E.g., {{2,4,6}} is a bag. So is {{2,4,6,2,2}}.
  - Equivalently written as {{ 2[3],4[1],6[1]}}.
- The order among elements in a set or bag is not important
  - E.g.,  $\{2,4,6\} = \{4,2,6\} = \{6,4,2\}$
  - $\{\{2,4,6,2,2\}\} = \{\{2,2,2,6,4\}\} = \{\{6,2,2,4,2\}\}.$

43

# Aliasing

- Allows you to rename the attributes of the result.
- Return the title and length of all movies as name and, resp., duration.
   SELECT title AS name, length AS duration
   FROM Movies

Result

name	duration
Pretty Woman	119
Monster's Inc.	121
Jurassic Park	145

# Aggregates in the SELECT clause

- Aggregates are allowed in the SELECT clause.
- Return the title and length of all movies as name and, resp., duration in hours

SELECT title AS name, length \* 0.016667 AS durationInHours FROM Movies

Result	name	durationInHours
	Pretty Woman	1.98
	Monster's Inc.	2.02
	Jurassic Park	2.42

45

### Constants in the result

- Constants can also added to the SELECT clause.
- Every tuple will have the same constant as specified in the SELECT clause.
   SELECT title AS name, length \* 0.016667 AS durationInHours, 'hrs.' as

inHours

**FROM Movies** 

Result	name	durationInHours	inHours
	Pretty Woman	1.98	hrs.
	Monster's Inc.	2.02	hrs.
	Jurassic Park	2.42	hrs.

# More on the conditions in the WHERE clause

- WHERE studioName = 'Disney' AND year = 1990;
- Comparison operators:
  - =, <>, <, >, <=, >=
  - Equal, not equal, less than, greater than, less than or equal, greater than or equal
  - E.g., WHERE year <= 1990
- Logical connectives:
  - AND, OR, NOT
  - E.g., WHERE NOT (studioName = `DISNEY' AND year <= 1990)
- Arithmetic expressions
  - +, -, \*, /, etc.
  - E.g.,

WHERE ((length\*0.01667) > 2 OR (length < 100)) AND year > 2000

47

# More on the conditions in the WHERE clause (cont'd)

- In general, the WHERE clause consists of a boolean combination of conditions using logical connectives AND, OR, NOT.
- Each condition is of the form

expression op expression

- expression is a column name, a constant, or an arithmetic or string expression
- op is a comparsion operator (=, <>, <, >, <=, >=, LIKE)

More on this later.

# String comparisons

- Strings are compared in lexicographical order.
  - Let  $a_1.a_2....a_n$  and  $b_1.b_2....b_m$  be two strings.
  - We have  $a_1.a_2....a_n < b_1.b_2....b_m$  if
    - 1.  $a_1.a_2....a_n$  is a proper prefix of  $b_1.b_2....b_m$ , or,
    - 2. For some 1 <= i < n, we have  $a_1=b_1$ ,  $a_2=b_2$ , ...,  $a_{i-1}=b_{i-1}$  and  $a_i < b_i$ .
- Examples:
  - 'Pretty' < 'Pretty Woman',</li>
  - 'butterfingers' < 'butterfly'</li>

49

# Pattern Matching in SQL

- · LIKE operator
  - s LIKE p, s NOT LIKE p
  - s is a string, p is a pattern
- '%' (stands for 0 or more arbitrary chars)
- '\_' (stands for exactly one arbitrary char)

SELECT \*

**FROM Movies** 

WHERE title LIKE 'Pretty%'

title LIKE "P\_etty%"

# Pattern Matching in SQL

- How do you match titles with an apostrophe?
  - E.g., 'Monster's Inc.'

SELECT \*

**FROM Movies** 

WHERE title LIKE 'Monster's Inc.'

Wrong!

WHERE title LIKE 'Monsters''s Inc.'

WHERE title LIKE """ // matches "

51

### **Date and Time**

• See Chapter 6.1.6.

# SQL's three-valued logic

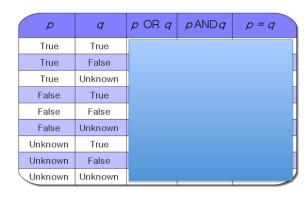
Students

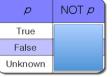
studentID	name	major	gender	avgGPA
112	Ann	Computer Science	F	NULL
327	Bob	NULL	M	3.90
835	Carl	Physics	М	4.00

- What is the result of the comparison "major = 'Computer Science' "AND "avgGPA > 3"?
- Three-valued logic TRUE, FALSE, UNKNOWN, where NULL is synonymous to UNKNOWN.
- If major is NULL, then "major='Computer Science'" evaluates to UNKNOWN.
- If avgGPA is NULL, then "avgGPA > 3" evaluates to UNKNOWN.

53

# SQL's three-valued logic (cont'd)





# SQL's three-valued logic (cont'd)

P	q	p OR q	p AND q	p = q
True	True	True	True	True
True	False	True	False	False
True	Unknown	True	Unknown	Unknown
False	True	True	False	False
False	False	False	False	True
False	Unknown	Unknown	False	Unknown
Unknown	True	True	Unknown	Unknown
Unknown	False	Unknown	False	Unknown
Unknown	Unknown	Unknown	Unknown	Unknown

p	NOT p
True	False
False	True
Unknown	Unknown

55

#### SELECT \*

**FROM Movies** 

WHERE major = 'Computer Science' AND avgGPA > 3.0;

- If the condition evaluates to UNKNOWN, then the tuple will not be returned.
- Both Ann and Bob will not be returned. In fact, none of the tuples will be returned.

# Ordering the Output

SELECT [DISTINCT] st of attributes> FROM R<sub>1</sub>, R<sub>2</sub>, ..., R<sub>n</sub> [WHERE condition] [ORDER BY <list of attributes>]

- ORDER BY presents the result in a sorted order.
- By default, the result will be ordered in ascending order. Otherwise, we state:
  - ORDER BY < list of attributes > DESC

57

#### **ORDER BY**

SELECT \*
FROM Movies
WHERE studioName = 'Disney' AND year = 1990
ORDER BY length, title;

- The result will list movies that satisfy the condition in the WHERE clause in ascending order of length, followed by title.
  - Shortest length movies will be listed first.
  - Among all movies with the same length, the movies will be sorted in ascending order of title.
- ORDER BY length, title DESC;
  - Longest length movies will be listed first.
  - Among all movies with the same length, the movies will be sorted in ascending order of title.

# Meaning of an SQL query with one relation in the FROM clause

```
SELECT [DISTINCT] c<sub>1</sub>, c<sub>2</sub>, ..., c<sub>m</sub>
FROM R<sub>1</sub>
[WHERE condition]
[ORDER BY < list of attributes > ] [DESC]
```

- · Let Result denote an empty collection.
- For every tuple t from R<sub>1</sub>,
  - if t satisfies condition (i.e., condition evaluates to true), then add the tuple that consists of c<sub>1</sub>, c<sub>2</sub>, ..., c<sub>m</sub> components of t into Result.
- If DISTINCT is stated in the SELECT clause, remove duplicates in Result.
- If ORDER BY < list of attributes> exists, order the tuples in Result according to ORDER BY clause.
- · Return Result.

59

### **Summary**

- Declarative vs. Procedural
- Data Definition Language (DDL)
  - CREATE TABLE, DROP TABLE, ALTER TABLE, PRIMARY KEY, UNIQUE, DEFAULT
- Data Manipulation Language (DML)
  - Basic SQL
    - SELECT...FROM...WHERE, SELECT DISTINCT...FROM...WHERE
    - NULL values in SQL