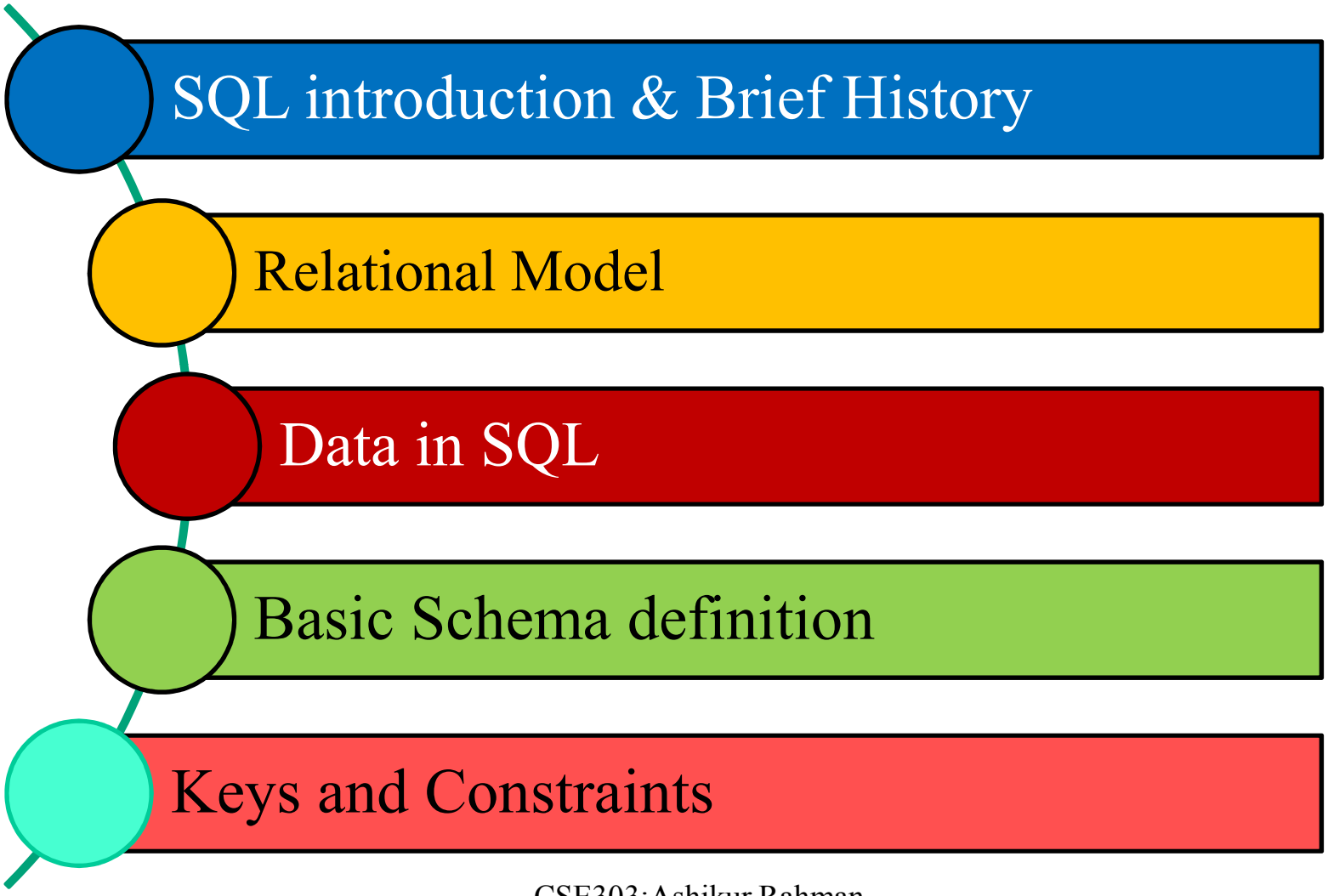
Database Systems
CSE 303

Lecture 02

2016

Structure Query Language (SQL)

Today's Outline (mainly from chapter 2)



SQL Introduction

- Standard language for querying and manipulating data

  Structured **Q**uery **L**anguage

- Original name was SEQUEL:
  Structured English QUEry Language

- Correct modern pronunciation is
  S-Q-L.

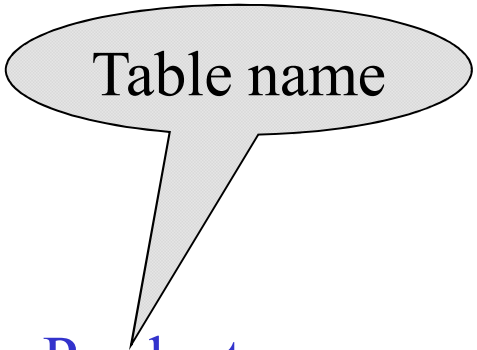
Chronology

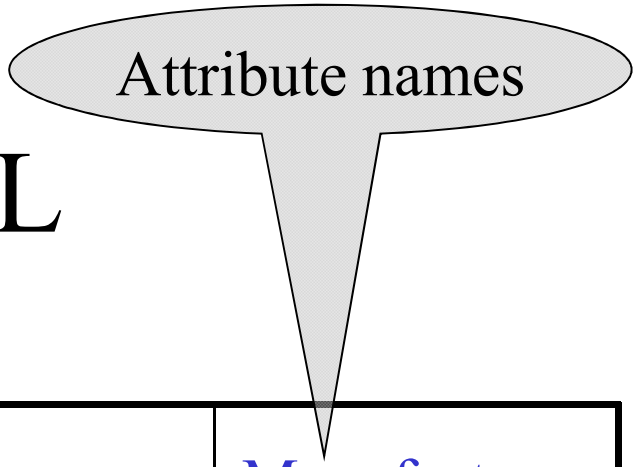
- 1970s IBM - first relational database System R, then DB2.
  Others include:
  Ingres Database - query language QUEL
  Digital - Relational Database Operator
  ISBL - relational algebra DML
  dBase family of products for PCs
- 1979 Oracle
- 1980 First standardisation efforts.
- 1984 ISO SQL standard - many flaws but universally adopted.
- 1992 Update to standard called SQL92 - The basic standard for any modern database
- 1999 Update to standard called SQL99 - Oracle database conforms to SQL99
- 2003 Current standard SQL2003 - Not many databases fully support this standard yet.

SQL

- Data Definition Language (DDL)
  - Create/alter/delete tables and their attributes
  - Today's lecture...
- Data Manipulation Language (DML)
  - Insert/delete/modify tuples in tables
  - Query one or more tables
  - Following lectures

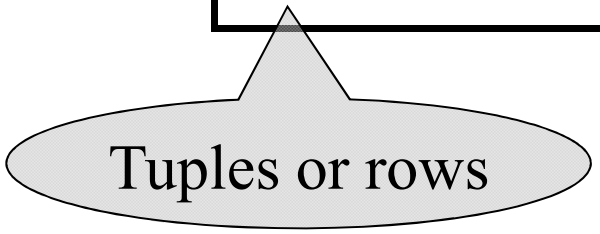
Tables in SQL





Product

PName	Price	Category	Manufacturer
Gizmo	$19.99	Gadgets	GizmoWorks
Powergizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi



Tables Explained

- A relation is a table.
  - (Envisioned by E.F. Codd 1970, IBM)
- The *schema* of a table is the table name and its attributes:

Product(PName, Price, Category, Manfacturer)

- *Database* = collection of relations.
- *Database schema* = set of all relation schemas in the database.

Tables Explained

- A tuple = a record/ a row in the table
  - Example:

    (SingleTouch, 149.99, Photography, Canon)
- A table = a set of tuples
  - Like a list…
  - …but it is unordered:
    no **first()**, no **next()**, no **last()**.

Domains

- Data types of attributes
  - Restriction: all attributes are of atomic type
    - that is elementary type i.e., integer, string
- Each attribute has an associated domain
  - i.e. the domain of Pname is string
  - The domain of price is real numbers

Creating Tables: SQL command

```
CREATE TABLE  Student (
        name            CHAR(50),
        id              NUMBER(10),
        gender          CHAR(1),
        address         CHAR(100),
        gpa             FLOAT
);
```

Common data types

- Characters:

  Syntax: CHAR[(maximum_size)]

  – Examples `kamal’
    - CHAR(40)
      – name CHAR(40)
    - CHAR
      – gender CHAR
  – Padded with blank at right

Common data types

- VARCHAR:

  Syntax: VARCHAR(maximum_size)

  – Variable length character string

  – Example

    - VARCHAR(40)
      - address VARCHAR(100)
    - VARCHAR → not permitted

  – Memory is allocated based on exact length of string.

Common data types

- CHAR vs. VARCHAR:
  - VARCHAR: no wastage of memory
  - CHAR: faster processing
  - Which one to use when?
    - If you want to store password?
    - If you want to store last name/first name etc...?

Common data types

- For numbers (like C)
  - INT or INTEGER
  - REAL
  - FLOAT

Common data types

- Fixed point decimal numbers:
  - DECIMAL[(precision , scale)] in SQL
  - NUMBER[(precision , scale)] in ORACLE
    - precision: how many digits in total
    - scale: how many digits after/before decimal
- In SQL Decimal(p,s)

Example

When you insert 7,456,123.89

(In this number 9 digits in total, 2 decimal digits)

Data Type		Value stored
- Number(9)	⇒	7456124
- Number(9,1)	⇒	7456123.9
- Number(9,2)	⇒	7456123.89
- Number(7,-2)	⇒	7456100
- Number(6)	⇒	Rejected

- Only NUMBER → any fraction

Common data types

- Date data type:

  Standard SQL

  YYYY-MM-DD

  Example:

  Date ‘1994-10-21’

Common data types

- Oracle provides a function to_date for creating an object of Date data type:

  General syntax

  TO_DATE(<string>, '<format>')

  Example:

  TO_DATE(‘1994-10-21’,’YYYY –MM-DD’)

  TO_DATE(‘1994-OCT-21’,’YYYY –MON-DD’)

  TO_DATE(‘1994-OCT-21’,’YYYY –MON-DD’)

Common data types

- Oracle's default date format is DD-MON-YY
  - If you use this format you don't have to use to_date function.

Common data types

- For outputing date you can use to_char function.
  - The general usage of TO_CHAR is:
    TO_CHAR(<date>, '<format>')

  where the <format> string can be formed from over 40 options. Some of the more popular ones are on the next slide

Common data types

MM	Numeric month (*e.g.*, 07)
MON	Abbreviated month name (*e.g.*, JUL)
MONTH	Full month name (*e.g.*, JULY)
DD	Day of month (*e.g.*, 24)
DY	Abbreviated name of day (*e.g.*, FRI)
YYYY	4-digit year (*e.g.*, 1998)
YY	Last 2 digits of the year (*e.g.*, 98)
RR	Like YY, but the two digits are ``rounded'' to a year in the range 1950 to 2049. Thus, 06 is considered 2006 instead of 1906

Common data types

- Two dates can be compared using = <> > etc.
- Two dates can only be subtracted NO addition, multiplication or division.

NULL data

Any value of any type can be NULL.

- Signals missing value.

Students(sid: *string*, name: *string*, email: *string*, gpa: *float*
dob*: date*)

- Value doesn't exist.
- Value exists but unknown.
- Value not applicable to this record.

NULL data

Any value of any type can be NULL.

- Signals missing value.

Students(sid: *string*, name: *string*, email: *string*, gpa: *float*
dob*: date*)

- Value doesn't exist. (for example email of a student)
- Value exists but unknown. (for example dob needs verification)
- Value not applicable to this record. (for example, L1 T1 student)

Creating table

```
CREATE TABLE  <tableName> (
        <list of attributes and types>
);
```

Creating table

```
CREATE TABLE  <tableName> (
        attribute1 type1,
        attribute2 type2,
        ….
        ….
);
```

Creating table

```
CREATE TABLE  Product (
        maker  VARCHAR(10),
        model  NUMBER(10),
        type     VARCHAR(10),
);
```

Case doesn’t matter

- Case insensitive:
  - Same: SELECT Select select
  - Same: Product product
  - Different: ‘Seattle’ ‘seattle’

- Constants:
  - ‘abc’ - yes
  - “abc” - no

Convention to follow

```
CREATE TABLE  Product (
        maker  VARCHAR(10),
        model  NUMBER(10),
        type     VARCHAR(10),
);
```

- SQL Commands: all caps
  - Example: *CREATE TABLE*
- Table name : Start with upper case , any subsequent word upper case
  - Example: *Product, MovieStar*
- Attributes: Start with lower case , any subsequent word upper case or separate by underscore.
  - Example: maker, producer, certNo (or cert_no)
- Data types
  - Example: CHAR, VARCHAR

Keys

Key = minimal set of attributes acting as a unique tuple identifier.

– Consider the universe of all potential relation data, not just what the table currently contains.

p_name	price	manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	39.99	GizmoWorks
Widget	19.99	WidgetsRUs
HyperWidget	203.99	Hyper

Product

What is the key here?

A. dept
B. dept, number
✓ C. dept, number, section
D. dept, number, section, instructor

Course

dept	number	section	instructor
MATH	101	A	Jones
MATH	101	B	Smith
MATH	102	A	Williams
PHYS	101	A	Baker
PHYS	102	B	Baker

Primary keys

Every table should select one *key* as *primary key*.

- Each tuple must have distinct non-NULL values of primary key attributes.
- Adding a new record with duplicate or NULL primary key will fail.

p_name	price	manufacturer
Gizmo	19.99	GizmoWorks
Powergizmo	39.99	GizmoWorks
Widget	19.99	WidgetsRUs
HyperWidget	203.99	Hyper

Product

Product (p_name: string, price: float, manufacturer: string)

Class Exercise

Movie relation

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

MovieStar relation

name	*address*	*gender*	*birthdate*
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

Movie relation

title	*year*	*length*	*genre*
Gone With the Wind	1939	231	drama
Star Wars	1977	124	sciFi
Wayne's World	1992	95	comedy

Movie(title: *string,* year:*int,* length*: int,* genre: *string*)

```
CREATE TABLE Movie(
        title    VARCHAR(50),
        year     INTEGER,
        length INTEGER,
        genre  CHAR(10),
        PRIMARY KEY (title, year)
)
```

MovieStar relation

name	*address*	*gender*	*birthdate*
Carrie Fisher	123 Maple St., Hollywood	F	9/9/99
Mark Hamill	456 Oak Rd., Brentwood	M	8/8/88
Harrison Ford	789 Palm Dr., Beverly Hills	M	7/7/77

MovieStar(name: *string,* address:*string,* gender*: char,* birthdate: *date*)

```
CREATE TABLE MovieStar(
        name        VARCHAR(50),
        address     VARCHAR(50),
        gender      CHAR(1),
        birthdate   DATE,
        PRIMARY KEY (name)
);
```

UNIQUE attribute

- Each tuple must have distinct values of UNIQUE attributes except null. More than one tuple may have null values in unique attribute(s)
- Adding a new record with duplicate in unique attribute will fail.

Students(sid: *string*, name: *string*, email: *string*, gpa: *float*
dob*: date,* username*: string*)

UNIQUE attribute (example)

Students(sid: *string*, name: *string*, email: *string*, gpa: *float*
dob*: date,* username*: string*)

```
CREATE TABLE Students(
        sid             CHAR(10),
        name            VARCHAR(50),
        email           VARCHAR(50),
        gpa             FLOAT,
        dob             DATE,
        username        CHAR(5) UNIQUE,
        PRIMARY KEY (sid)
);
```