Lecture 03: Structured Query Language SQL - DDL

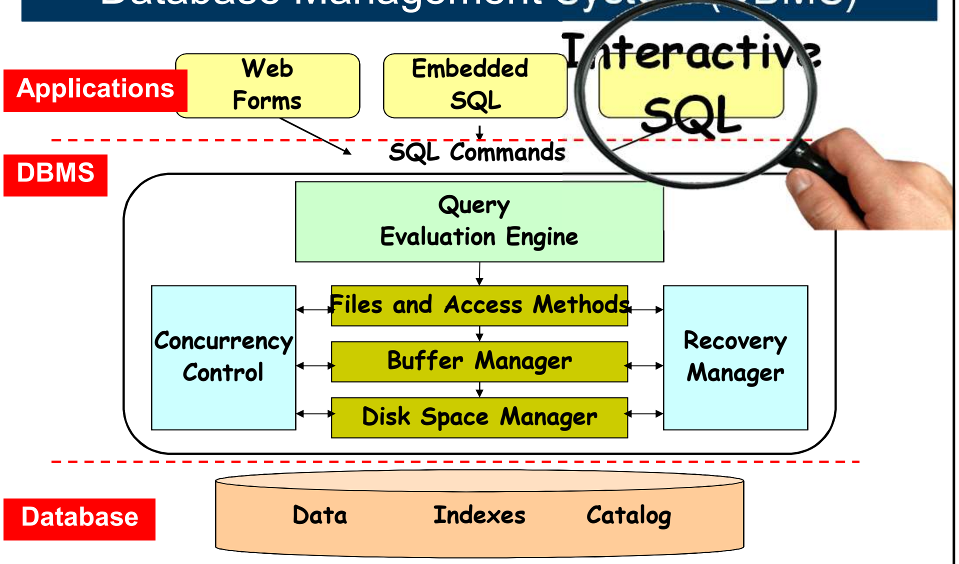
CS 1555: Database Management Systems

Constantinos Costa
http://db.cs.pitt.edu/courses/cs1555/current.term/

Jan 15, 2019, 16:00-17:15
University of Pittsburgh, Pittsburgh, PA

Lectures based: P. Chrysanthis & N. Farnan Lectures

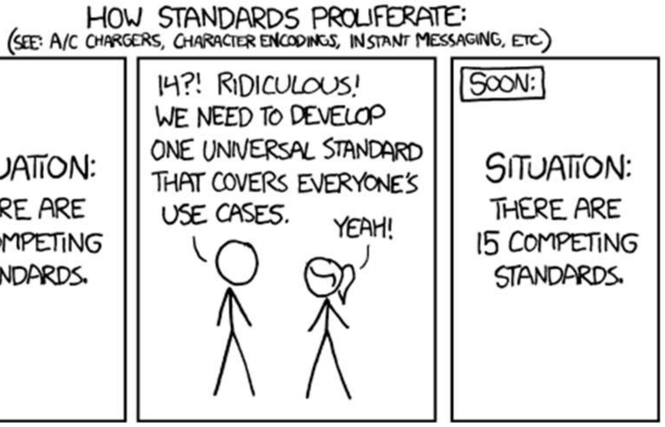
Database Management System (DBMS)



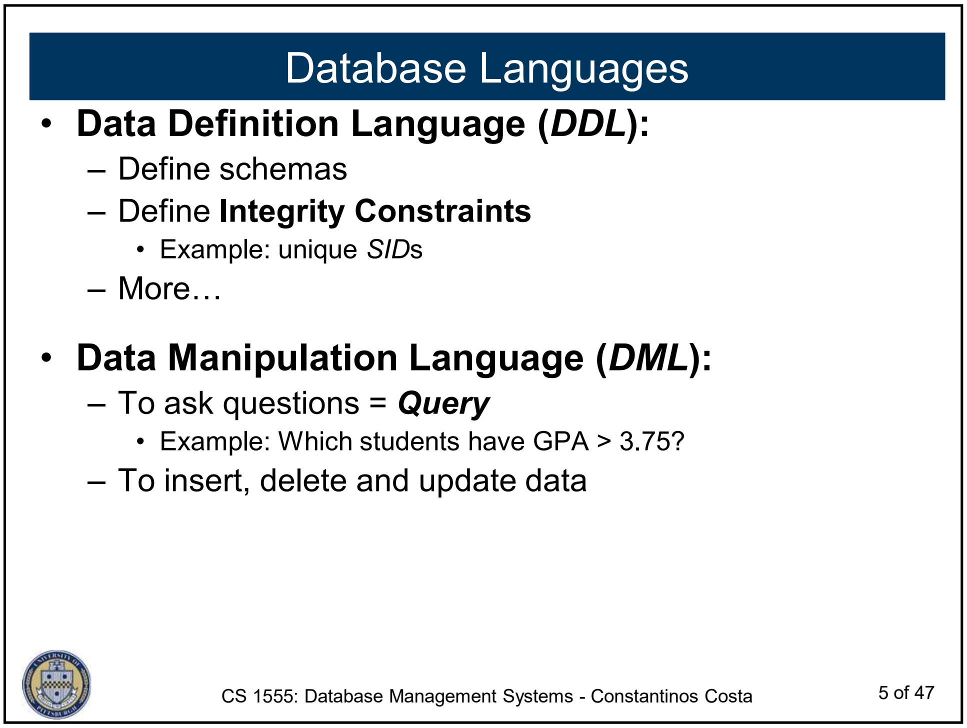
SQL

- SQL is the query language for the **System R** developed at IBM San Jose [Astraham, Gray, Linsday, Selinger,…]
- SQL is the de-facto standard on most RDBMS
- Most successful standardization effort
  - SQL (ANSI 1986)
  - SQL1 (ANSI 1989)
  - SQL2 or SQL92 (ANSI 1992)
  - SQL3 (ANSI 1999/2000/2003) -- Core and Packages
  - SQL 2008
  - SQL 2013

A word about Standards

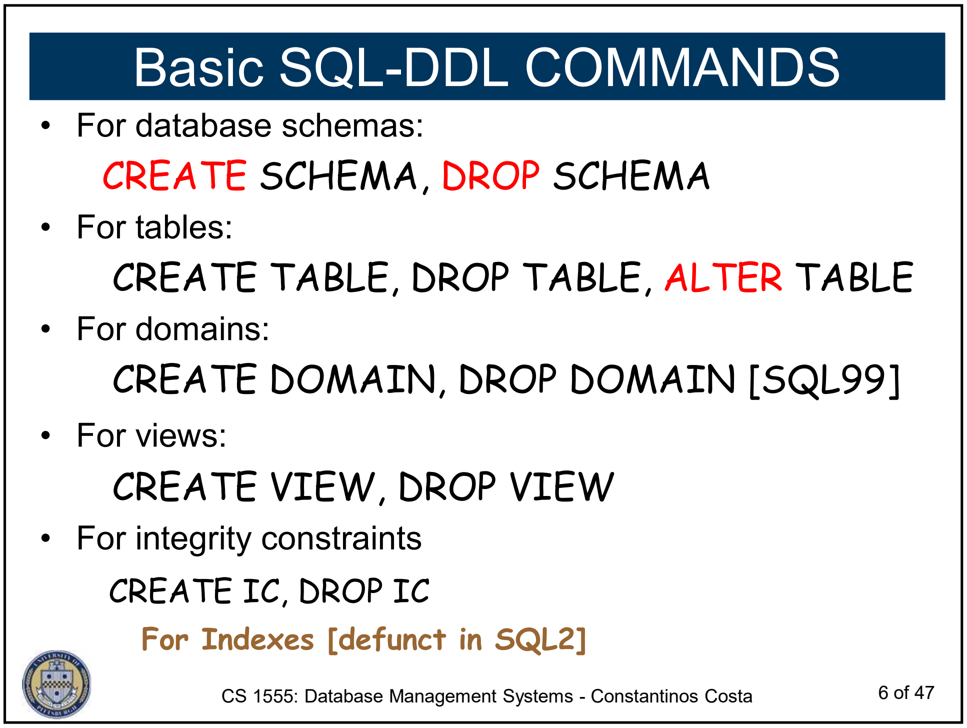

HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC.)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

http://xkcd.com/927/



Database Languages

- **Data Definition Language (*DDL*):**
  - Define schemas
  - Define **Integrity Constraints**
    - Example: unique *SID*s
  - More…
- **Data Manipulation Language (*DML*):**
  - To ask questions = ***Query***
    - Example: Which students have GPA > 3.75?
  - To insert, delete and update data



Basic SQL-DDL COMMANDS

- For database schemas:

```
CREATE SCHEMA, DROP SCHEMA
```

- For tables:

```
CREATE TABLE, DROP TABLE, ALTER TABLE
```

- For domains:

```
CREATE DOMAIN, DROP DOMAIN [SQL99]
```

- For views:

```
CREATE VIEW, DROP VIEW
```

- For integrity constraints

```
CREATE IC, DROP IC
```

For Indexes [defunct in SQL2]

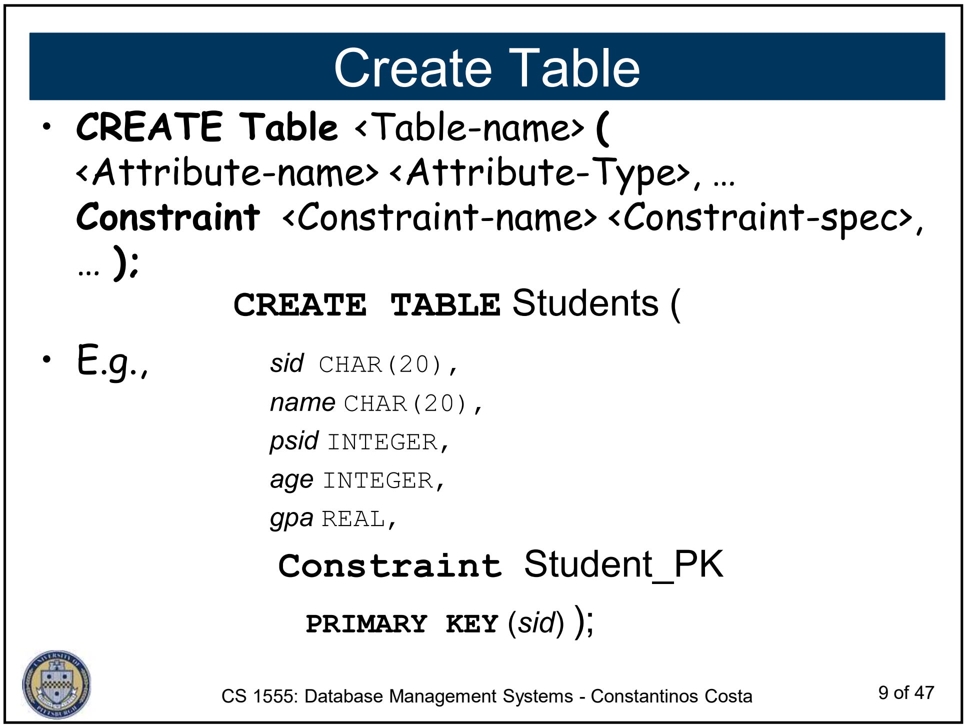
Database Schema

- **CREATE SCHEMA** <database-name>
  AUTHORIZATION <user-identifier>;
- E.g. **CREATE SCHEMA** micro_db

  AUTHORIZATION panos;
- **DROP SCHEMA** <db-name> [**RESTRICT** | **CASCADE**];
  - Restrict: removes the schema if the db has no data
  - *Cascade*: removes everything, data and definitions
- E.g., **DROP SCHEMA** micro_db **RESTRICT**;

Schema and Catalog

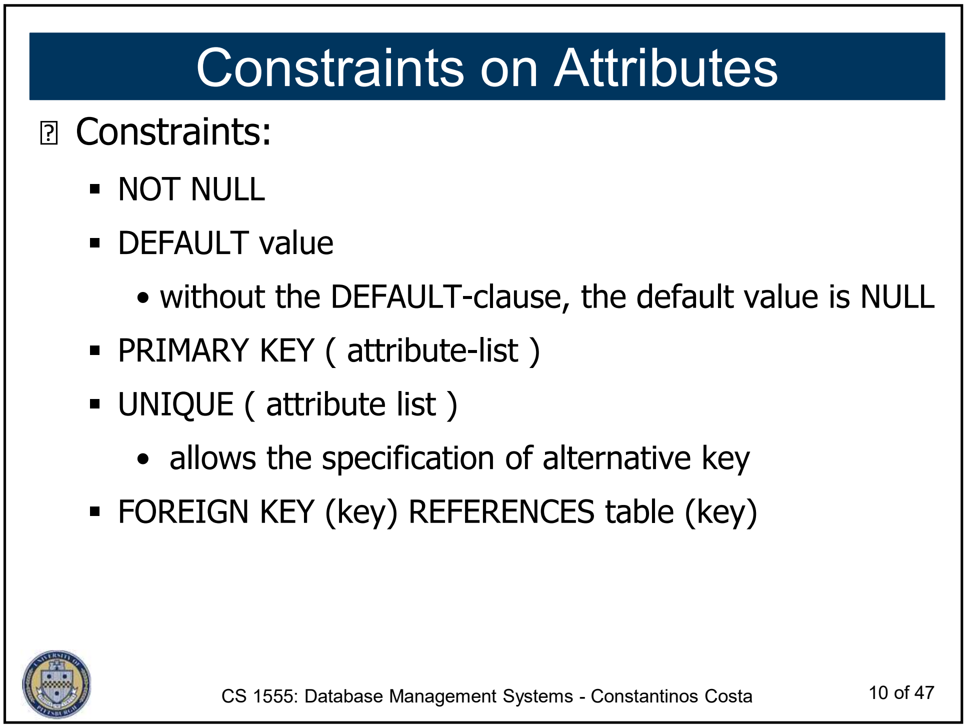
- **Schema** describes the data stored in the database
- **Catalog** contains the definitions of schemas
- INFORMATION_SCHEMA
  - Schemas and Base relations (tables) (tbl_name, creator, #of_tuples, tuple_length, #of_attributes…)
  - Attributes of Relations (columns) (tbl_name, atrb_name, type, format, order, key_no, ...)
  - Indexes
  - (tbl_name, index_name, key_attribute,...)
  - Authorizations
  - Integrity Constains
- Naming of tables: Schema_name.Table_name
- Query: \d+ table name; or using SELECT



Create Table

- **CREATE Table** <Table-name> **(**
  <Attribute-name> <Attribute-Type>, …
  Constraint <Constraint-name> <Constraint-spec>,
  … **);**
- E.g.,

```
CREATE TABLE Students (
    sid  CHAR(20),
    name CHAR(20),
    psid INTEGER,
    age INTEGER,
    gpa REAL,
    Constraint Student_PK
      PRIMARY KEY (sid) );
```



Constraints on Attributes

- Constraints:
  - NOT NULL
  - DEFAULT value
    - without the DEFAULT-clause, the default value is NULL
  - PRIMARY KEY (attribute-list)
  - UNIQUE (attribute list)
    - allows the specification of alternative key
  - FOREIGN KEY (key) REFERENCES table (key)

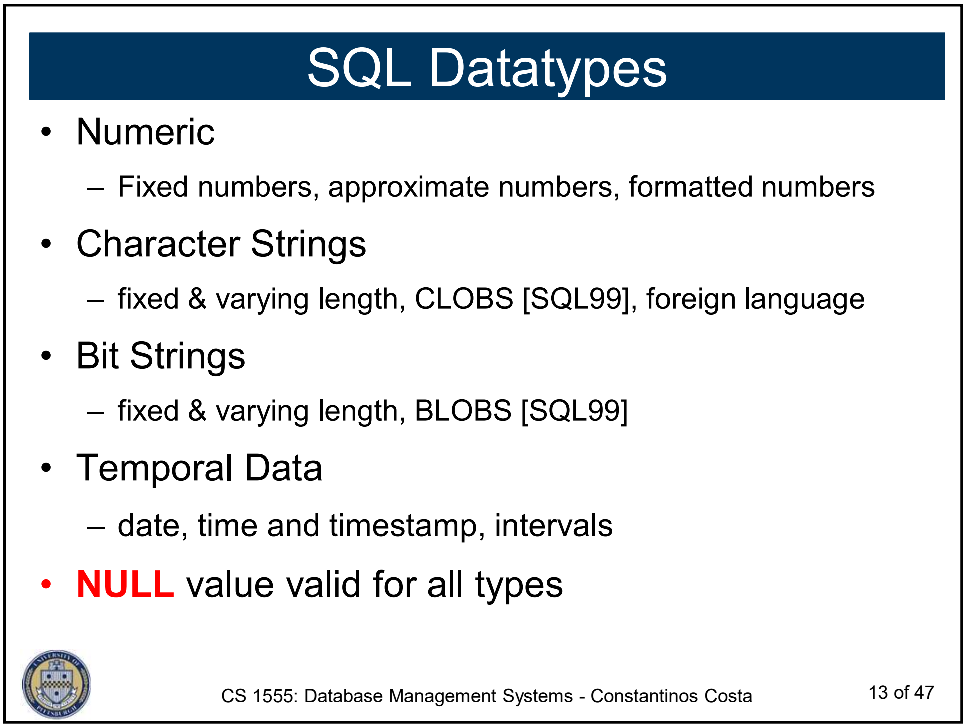
Creating and deleting schemas

- **CREATE SCHEMA** *name* **AUTHORIZATION** *user*;
- **DROP SCHEMA** *name* [**RESTRICT**|**CASCADE**];
  - ○ RESTRICT: removes schema if it doesn't contain any elements
  - ○ CASCADE: remove schema and everything it contains

○ A **schema (PostgresSQL)** is a named collection of tables.

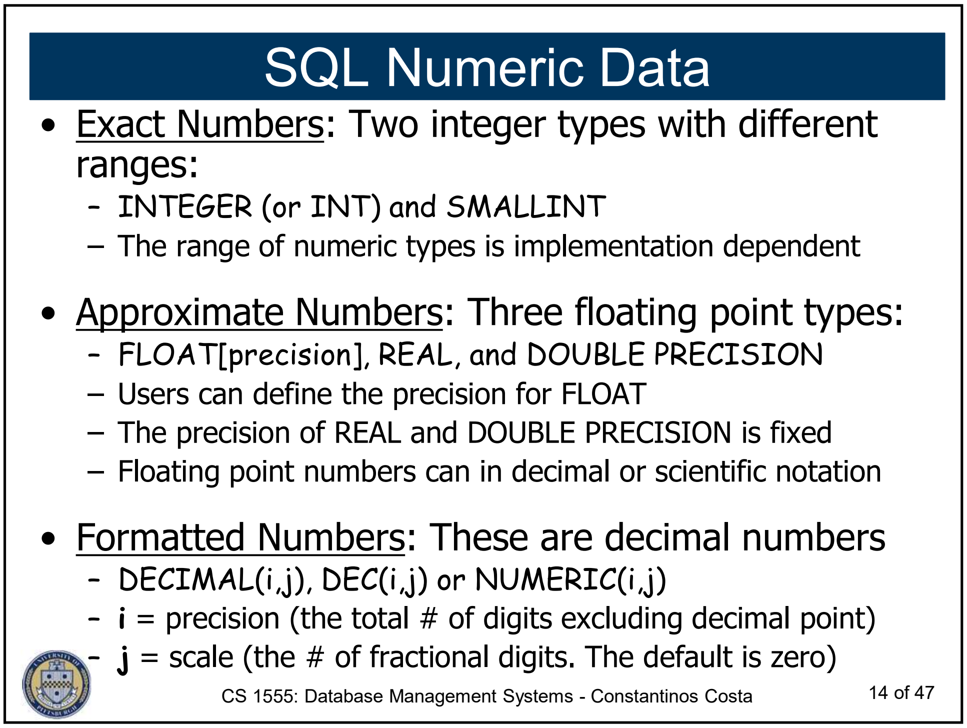
Create Table Schema

```
• CREATE TABLE STUDENT
   ( SID    INTEGER,
     Name   CHAR(20),
     PSID   INTEGER NOT NULL,  -- REQUIRED for AK
     AGE    INTEGER,
     GPA    REAL,
     Major  CHAR (10),
     CONSTRAINT STUDENT_PK
        PRIMARY KEY (SID),
     CONSTRAINT STUDENT_UN
        UNIQUE (PSID),
     CONSTRAINT STUDENT_FK
        FOREIGN KEY (Major) REFERENCES Department (DNO)
        ON UPDATE CASCADE ON DELETE NO ACTION
   );
```



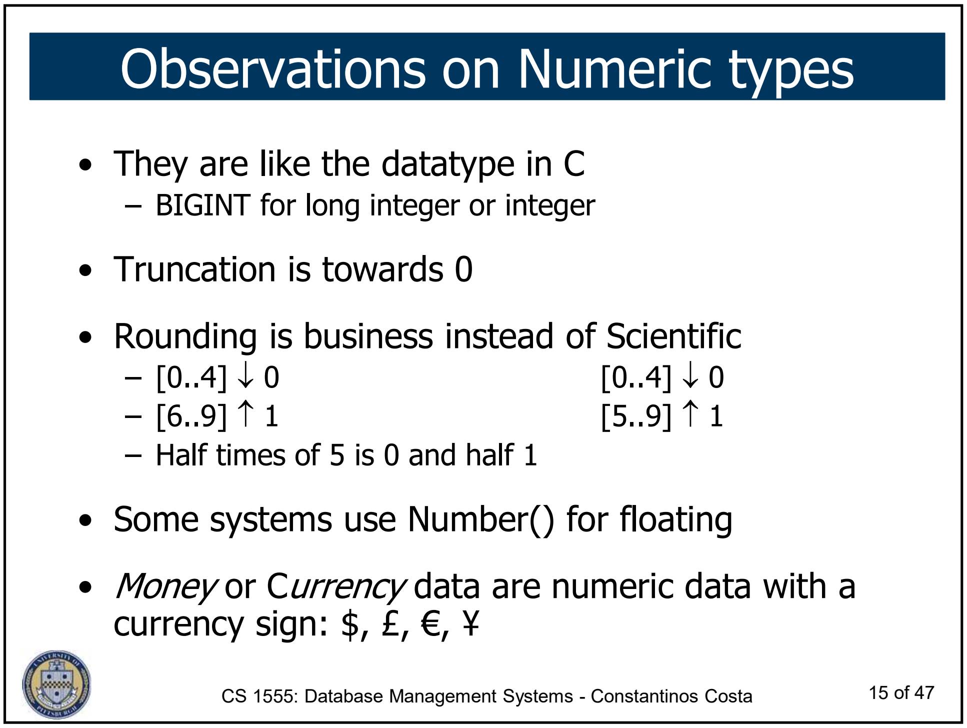
SQL Datatypes

- Numeric
  - Fixed numbers, approximate numbers, formatted numbers
- Character Strings
  - fixed & varying length, CLOBS [SQL99], foreign language
- Bit Strings
  - fixed & varying length, BLOBS [SQL99]
- Temporal Data
  - date, time and timestamp, intervals
- **NULL** value valid for all types



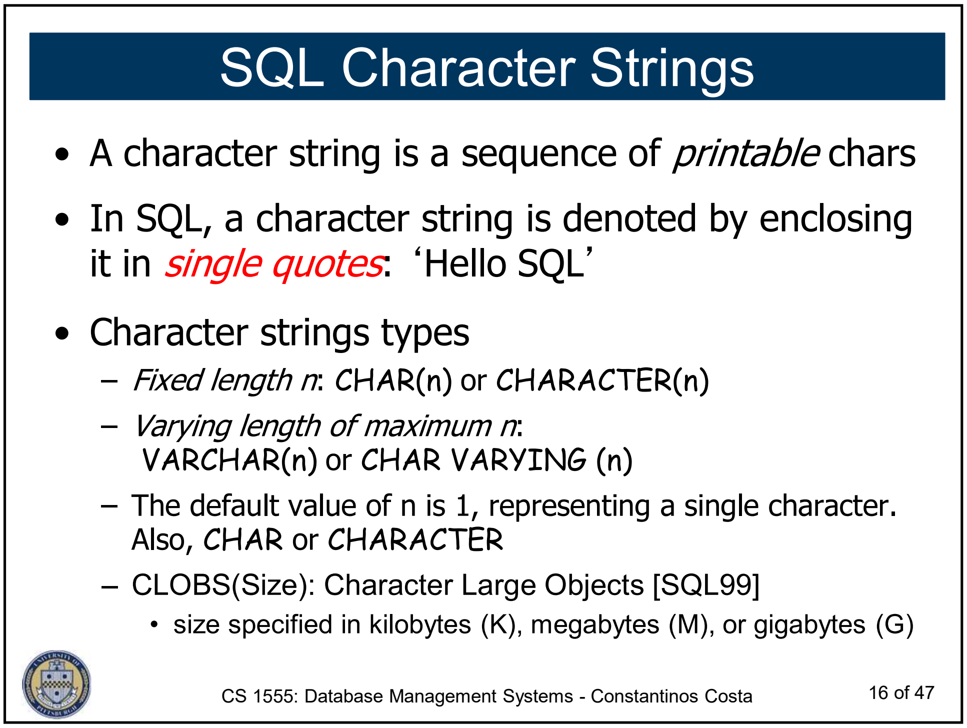
SQL Numeric Data

- Exact Numbers: Two integer types with different ranges:
  - INTEGER (or INT) and SMALLINT
  - The range of numeric types is implementation dependent
- Approximate Numbers: Three floating point types:
  - FLOAT[precision], REAL, and DOUBLE PRECISION
  - Users can define the precision for FLOAT
  - The precision of REAL and DOUBLE PRECISION is fixed
  - Floating point numbers can in decimal or scientific notation
- Formatted Numbers: These are decimal numbers
  - DECIMAL(i,j), DEC(i,j) or NUMERIC(i,j)
  - **i** = precision (the total # of digits excluding decimal point)
  - **j** = scale (the # of fractional digits. The default is zero)



Observations on Numeric types

- They are like the datatype in C
  - BIGINT for long integer or integer
- Truncation is towards 0
- Rounding is business instead of Scientific
  - [0..4] ↓ 0 [0..4] ↓ 0
  - [6..9] ↑ 1 [5..9] ↑ 1
  - Half times of 5 is 0 and half 1
- Some systems use Number() for floating
- *Money* or C*urrency* data are numeric data with a currency sign: $, £, €, ¥



SQL Character Strings

- A character string is a sequence of *printable* chars
- In SQL, a character string is denoted by enclosing it in *single quotes*: ‘Hello SQL’
- Character strings types
  - *Fixed length n*: CHAR(n) or CHARACTER(n)
  - *Varying length of maximum n*:
    VARCHAR(n) or CHAR VARYING (n)
  - The default value of n is 1, representing a single character. Also, CHAR or CHARACTER
  - CLOBS(Size): Character Large Objects [SQL99]
    - size specified in kilobytes (K), megabytes (M), or gigabytes (G)

SQL Character Strings

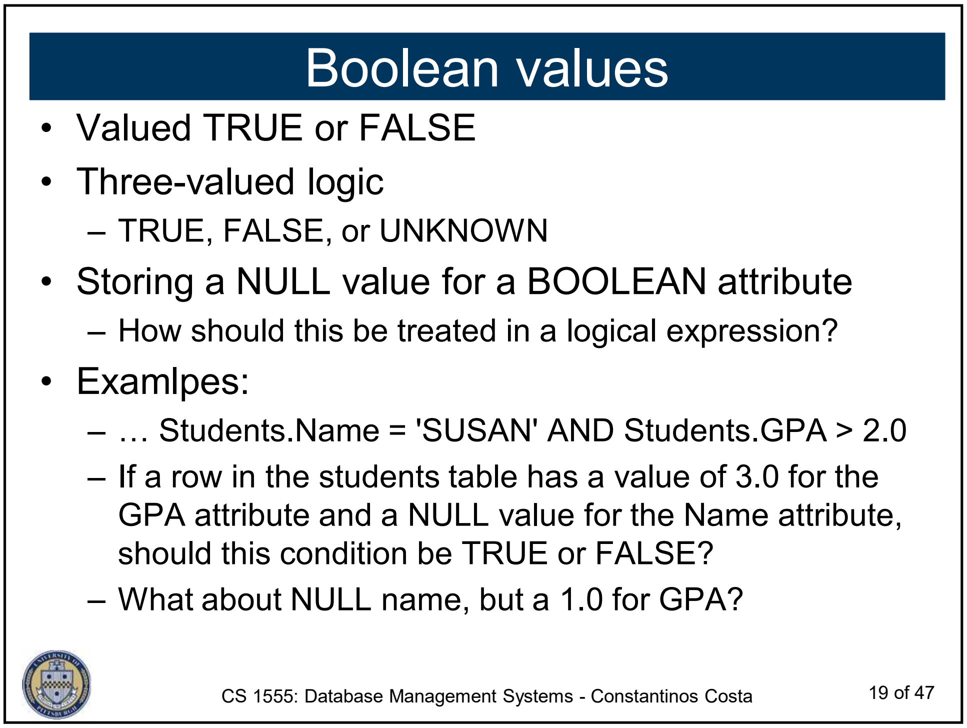
- Concatenation operator: ||
  - 'abc' || 'XYZ' results in 'abcXYZ'

- Foreign-language characters (ISO-defined chars):
  - NATIONAL CHAR(n)
  - NATIONAL VARCHAR(n)

SQL Bit Strings

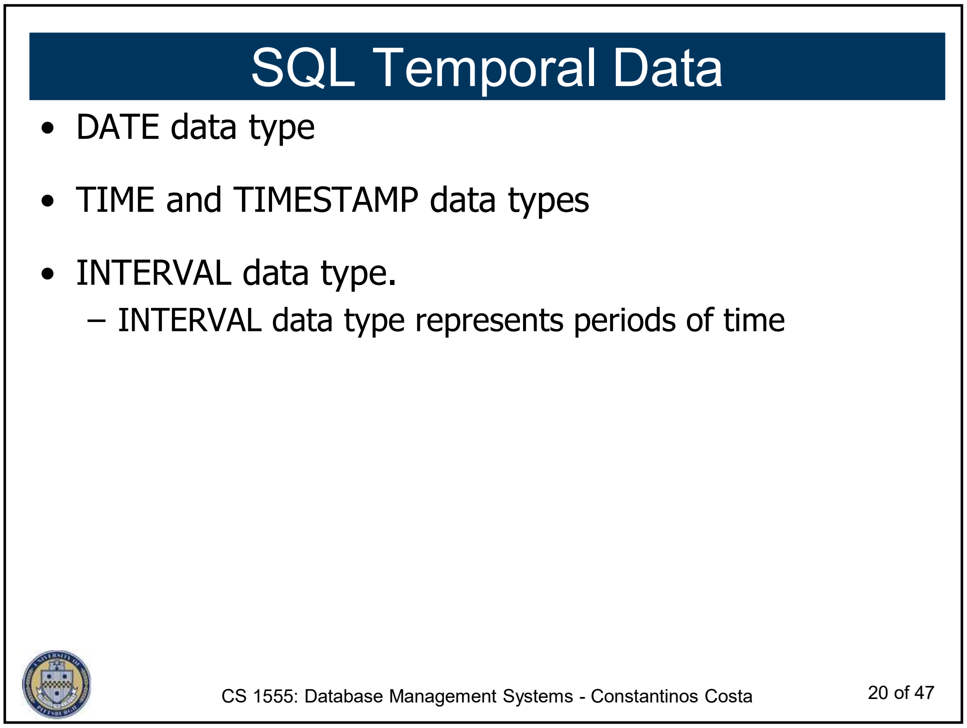
- Bit strings are sequences of binary digits, or bits
- In SQL, a bit string is denoted by enclosing it in *single quotes*: B’0101100110’
- Bit String types
  - *Fixed length n*: BIT(n)
  - *Varying length of maximum n*: VARBIT(n) or BIT VARYING (n)
- The default value for n is 1

  - BLOBS (size): Binary Large Objects [SQL99]
  - size specified in kilobytes (K), megabytes (M), or gigabytes (G)



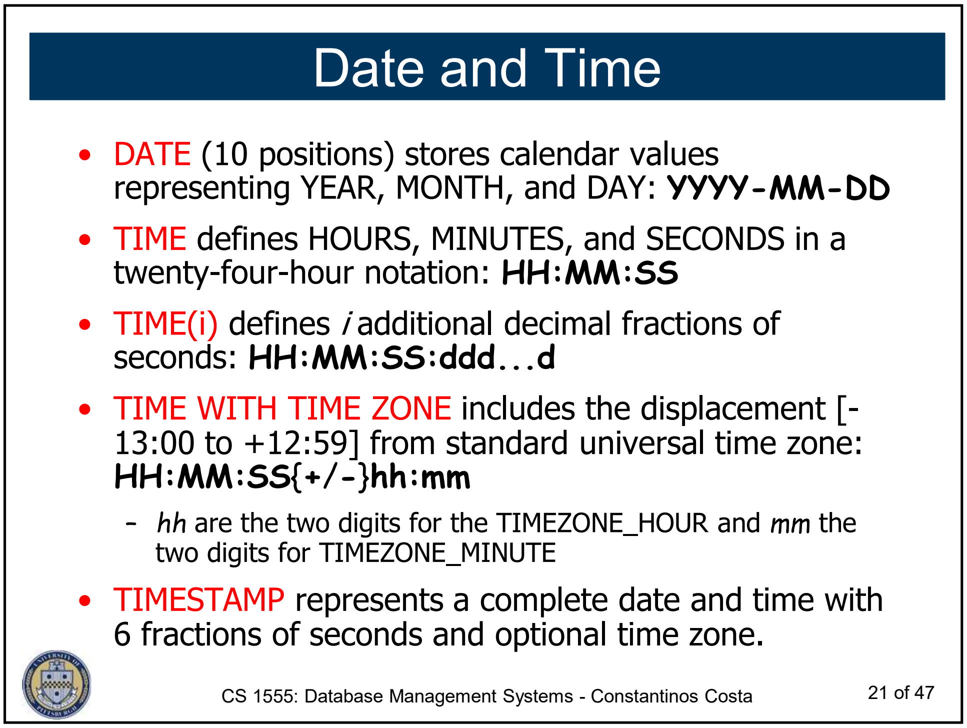
Boolean values

- Valued TRUE or FALSE
- Three-valued logic
  - TRUE, FALSE, or UNKNOWN
- Storing a NULL value for a BOOLEAN attribute
  - How should this be treated in a logical expression?
- Examlpes:
  - … Students.Name = 'SUSAN' AND Students.GPA > 2.0
  - If a row in the students table has a value of 3.0 for the GPA attribute and a NULL value for the Name attribute, should this condition be TRUE or FALSE?
  - What about NULL name, but a 1.0 for GPA?



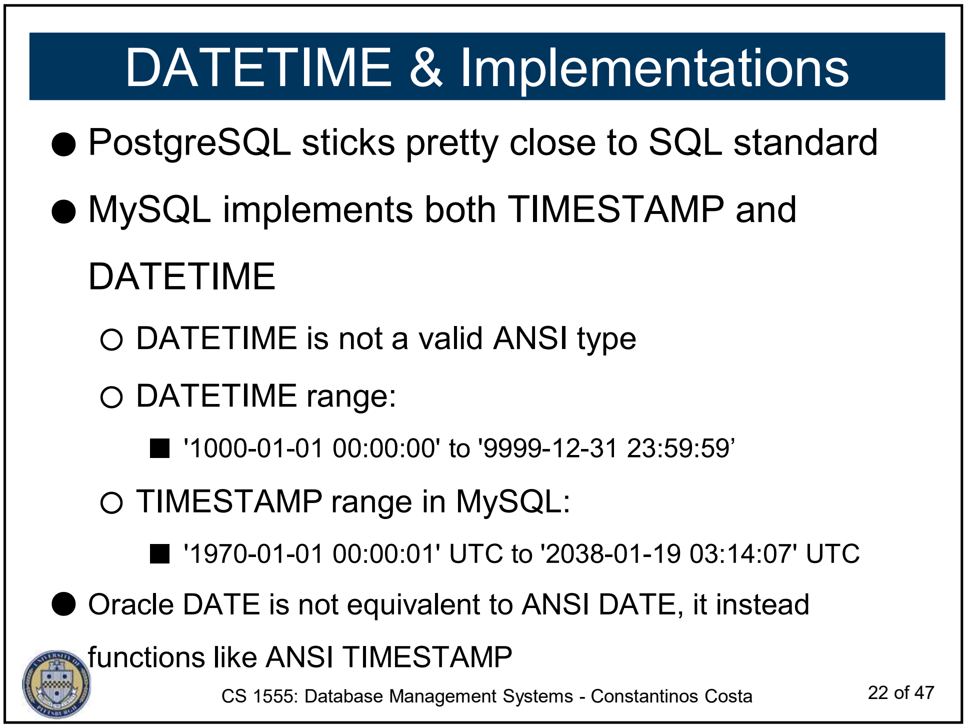
SQL Temporal Data

- DATE data type
- TIME and TIMESTAMP data types
- INTERVAL data type.
  - INTERVAL data type represents periods of time



Date and Time

- DATE (10 positions) stores calendar values representing YEAR, MONTH, and DAY: **YYYY-MM-DD**
- TIME defines HOURS, MINUTES, and SECONDS in a twenty-four-hour notation: **HH:MM:SS**
- TIME(i) defines *i* additional decimal fractions of seconds: **HH:MM:SS:ddd...d**
- TIME WITH TIME ZONE includes the displacement [-13:00 to +12:59] from standard universal time zone: **HH:MM:SS{+/-}hh:mm**
  - *hh* are the two digits for the TIMEZONE_HOUR and *mm* the two digits for TIMEZONE_MINUTE
- TIMESTAMP represents a complete date and time with 6 fractions of seconds and optional time zone.



DATETIME & Implementations

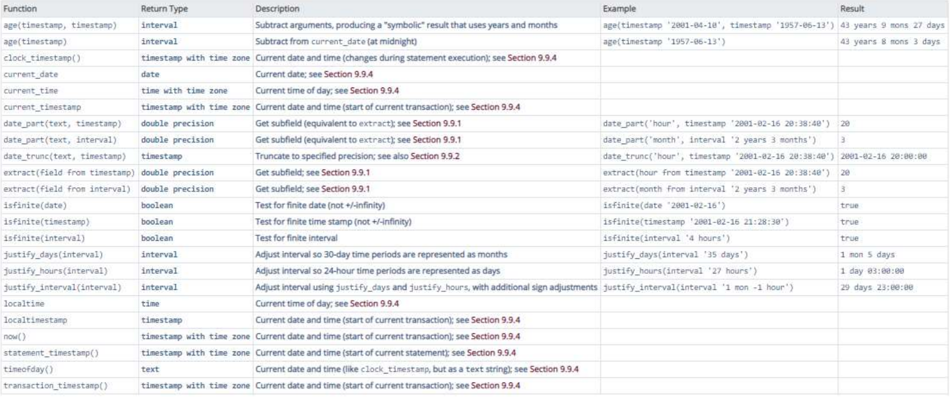
- PostgreSQL sticks pretty close to SQL standard
- MySQL implements both TIMESTAMP and DATETIME
  - DATETIME is not a valid ANSI type
  - DATETIME range:
    - '1000-01-01 00:00:00' to '9999-12-31 23:59:59’
  - TIMESTAMP range in MySQL:
    - '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC
- Oracle DATE is not equivalent to ANSI DATE, it instead functions like ANSI TIMESTAMP

Operations on Dates

- Represent periods of time and are used in operations on date and time data types
  - Datetime (+ or -) Interval = Datetime
  - Datetime - Datetime = Interval
  - Interval (* or /) Number = Interval
  - Interval (+ or -) Interval = Interval

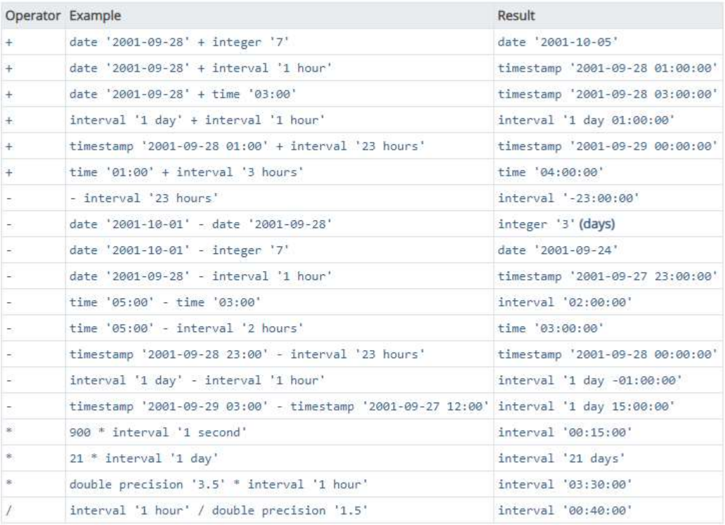
- Examples (ANSI SQL):
  - (CURRENT_DATE + INTERVAL '1' MONTH)
  - (CURRENT_DATE - INTERVAL ‘18' DAY)
  - (CURRENT_DATE – BirthDate)

Functions on Dates



Function	Return Type	Description	Example	Result
age(timestamp, timestamp)	interval	Subtract arguments, producing a "symbolic" result that uses years and months	age(timestamp '2001-04-10', timestamp '1957-06-13')	43 years 9 mons 27 days
age(timestamp)	interval	Subtract from current_date (at midnight)	age(timestamp '1957-06-13')	43 years 8 mons 3 days
clock_timestamp()	timestamp with time zone	Current date and time (changes during statement execution); see Section 9.9.4		
current_date	date	Current date; see Section 9.9.4		
current_time	time with time zone	Current time of day; see Section 9.9.4		
current_timestamp	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
date_part(text, timestamp)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('hour', timestamp '2001-02-16 20:38:40')	20
date_part(text, interval)	double precision	Get subfield (equivalent to extract); see Section 9.9.1	date_part('month', interval '2 years 3 months')	3
date_trunc(text, timestamp)	timestamp	Truncate to specified precision; see also Section 9.9.2	date_trunc('hour', timestamp '2001-02-16 20:38:40')	2001-02-16 20:00:00
extract(field from timestamp)	double precision	Get subfield; see Section 9.9.1	extract(hour from timestamp '2001-02-16 20:38:40')	20
extract(field from interval)	double precision	Get subfield; see Section 9.9.1	extract(month from interval '2 years 3 months')	3
isfinite(date)	boolean	Test for finite date (not +/-infinity)	isfinite(date '2001-02-16')	true
isfinite(timestamp)	boolean	Test for finite time stamp (not +/-infinity)	isfinite(timestamp '2001-02-16 21:28:30')	true
isfinite(interval)	boolean	Test for finite interval	isfinite(interval '4 hours')	true
justify_days(interval)	interval	Adjust interval so 30-day time periods are represented as months	justify_days(interval '35 days')	1 mon 5 days
justify_hours(interval)	interval	Adjust interval so 24-hour time periods are represented as days	justify_hours(interval '27 hours')	1 day 03:00:00
justify_interval(interval)	interval	Adjust interval using justify_days and justify_hours, with additional sign adjustments	justify_interval(interval '1 mon -1 hour')	29 days 23:00:00
localtime	time	Current time of day; see Section 9.9.4		
localtimestamp	timestamp	Current date and time (start of current transaction); see Section 9.9.4		
now()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		
statement_timestamp()	timestamp with time zone	Current date and time (start of current statement); see Section 9.9.4		
timeofday()	text	Current date and time (like clock_timestamp, but as a text string); see Section 9.9.4		
transaction_timestamp()	timestamp with time zone	Current date and time (start of current transaction); see Section 9.9.4		

Intervals…



Operator	Example	Result
+	date '2001-09-28' + integer '7'	date '2001-10-05'
+	date '2001-09-28' + interval '1 hour'	timestamp '2001-09-28 01:00:00'
+	date '2001-09-28' + time '03:00'	timestamp '2001-09-28 03:00:00'
+	interval '1 day' + interval '1 hour'	interval '1 day 01:00:00'
+	timestamp '2001-09-28 01:00' + interval '23 hours'	timestamp '2001-09-29 00:00:00'
+	time '01:00' + interval '3 hours'	time '04:00:00'
-	- interval '23 hours'	interval '-23:00:00'
-	date '2001-10-01' - date '2001-09-28'	integer '3' (days)
-	date '2001-10-01' - integer '7'	date '2001-09-24'
-	date '2001-09-28' - interval '1 hour'	timestamp '2001-09-27 23:00:00'
-	time '05:00' - time '03:00'	interval '02:00:00'
-	time '05:00' - interval '2 hours'	time '03:00:00'
-	timestamp '2001-09-28 23:00' - interval '23 hours'	timestamp '2001-09-28 00:00:00'
-	interval '1 day' - interval '1 hour'	interval '1 day -01:00:00'
-	timestamp '2001-09-29 03:00' - timestamp '2001-09-27 12:00'	interval '1 day 15:00:00'
*	900 * interval '1 second'	interval '00:15:00'
*	21 * interval '1 day'	interval '21 days'
*	double precision '3.5' * interval '1 hour'	interval '03:30:00'
/	interval '1 hour' / double precision '1.5'	interval '00:40:00'

Quick Example.. Student Table

SID	Name	PSID	Age	GPA
546007	Jones	689065	18	3.4
546100	Smith	987452	18	3.2
546500	Smith	342875	19	3.8

```
CREATE TABLE Student
(  sid CHAR(20),
   name CHAR(20),
   psid INTEGER,
   age INTEGER,
   gpa REAL,
   Constraint Student_PK
     PRIMARY KEY (sid) );
```

```
CREATE TABLE Student
( sid CHAR(20)
    Constraint Student_PK
    PRIMARY KEY ,
  name CHAR(20),
  psid INTEGER,
  age INTEGER,
  gpa REAL );
```

Table Schema Storing Option

- **CREATE SCHEMA** CS1555;
- **CREATE TABLE** STUDENT

```
        (   SID    INTEGER,
            Name CHAR(20),
            PSID INTEGER NOT NULL,
            AGE  INTEGER,
       GPA  REAL,
      CONSTRAINT STUDENT_PK
         PRIMARY KEY (SID),
      CONSTRAINT STUDENT_AK
         UNIQUE (PSID));
        TABLESPACE                             --In postgres
        IS TABLESPACE {tablespace | users};    -- In Oracle
        ON {filegroup | DEFAULT};              -- In SQLServer
```

Table Schema (MySQL)

o **CREATE TABLE** STUDENT

```
      (  SID    INTEGER,
           Name CHAR(20),
         PSID  INTEGER NOT NULL,
         AGE   INTEGER,
         GPA   REAL,
         CONSTRAINT STUDENT_PK
            PRIMARY KEY (SID),
         CONSTRAINT STUDENT_AK
            UNIQUE (PSID) );
Engine = INNODB;       -- Required in MySQL  to support FK
Options: ARCHIVE, CSV, HEAP, Memory, myisam, ndbcluster
```

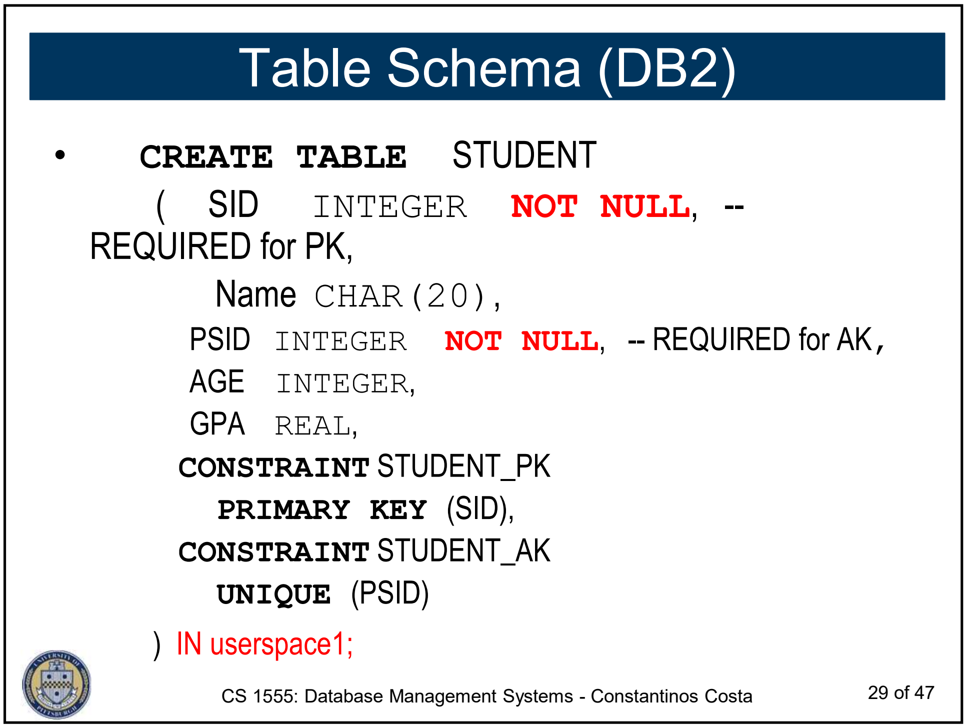
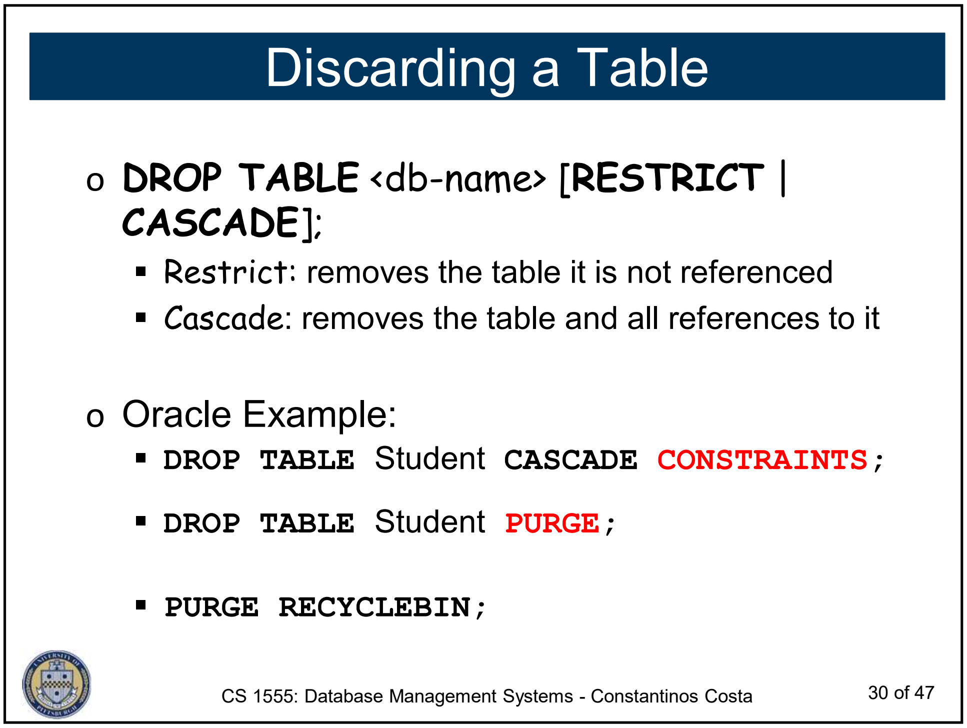


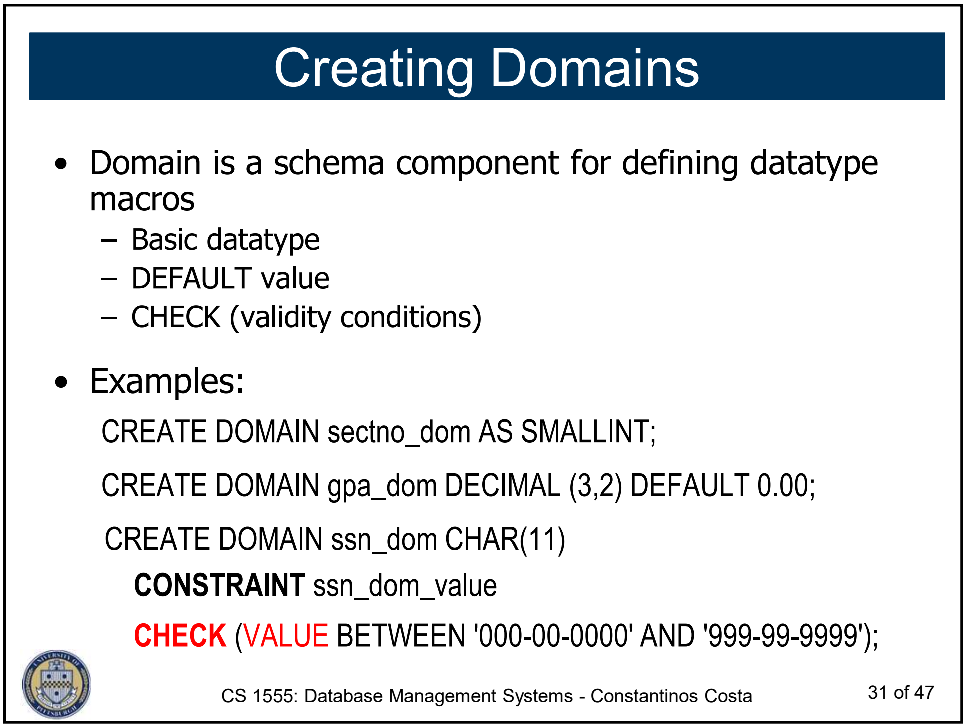
Table Schema (DB2)

```
•     CREATE TABLE   STUDENT
        (   SID    INTEGER   NOT NULL,  --
   REQUIRED for PK,
              Name CHAR(20),
             PSID  INTEGER   NOT NULL,  -- REQUIRED for AK,
             AGE   INTEGER,
             GPA   REAL,
            CONSTRAINT STUDENT_PK
              PRIMARY KEY  (SID),
            CONSTRAINT STUDENT_AK
              UNIQUE  (PSID)
        )  IN userspace1;
```



Discarding a Table

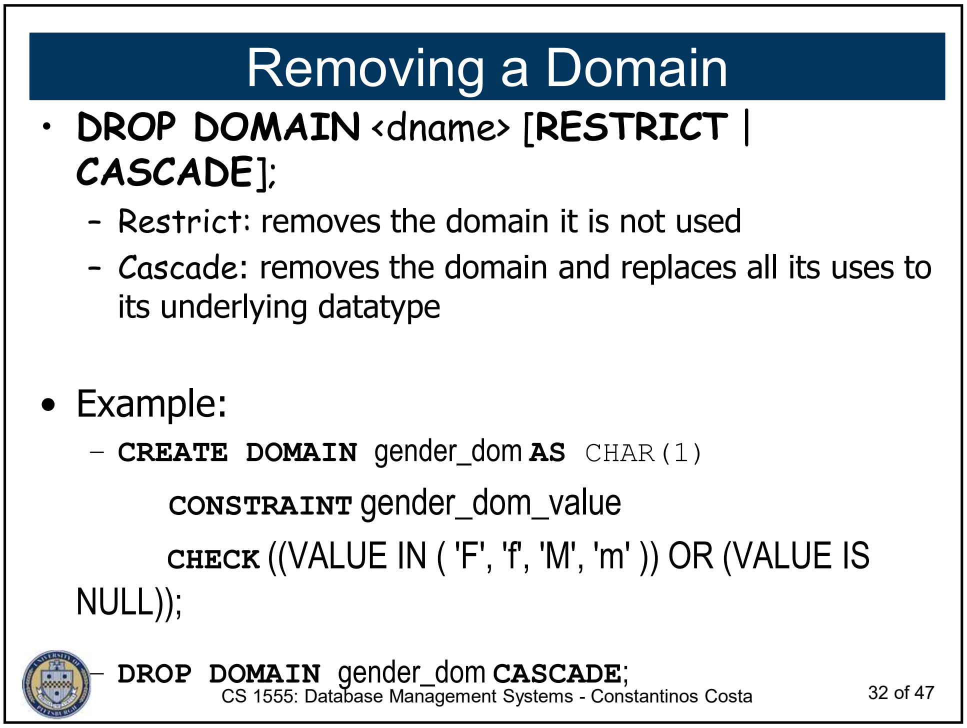
- **DROP TABLE** <db-name> [**RESTRICT** | **CASCADE**];
  - Restrict: removes the table it is not referenced
  - *Cascade*: removes the table and all references to it
- Oracle Example:
  - `DROP TABLE Student CASCADE CONSTRAINTS;`
  - `DROP TABLE Student PURGE;`
  - `PURGE RECYCLEBIN;`



Creating Domains

- Domain is a schema component for defining datatype macros
  - Basic datatype
  - DEFAULT value
  - CHECK (validity conditions)
- Examples:

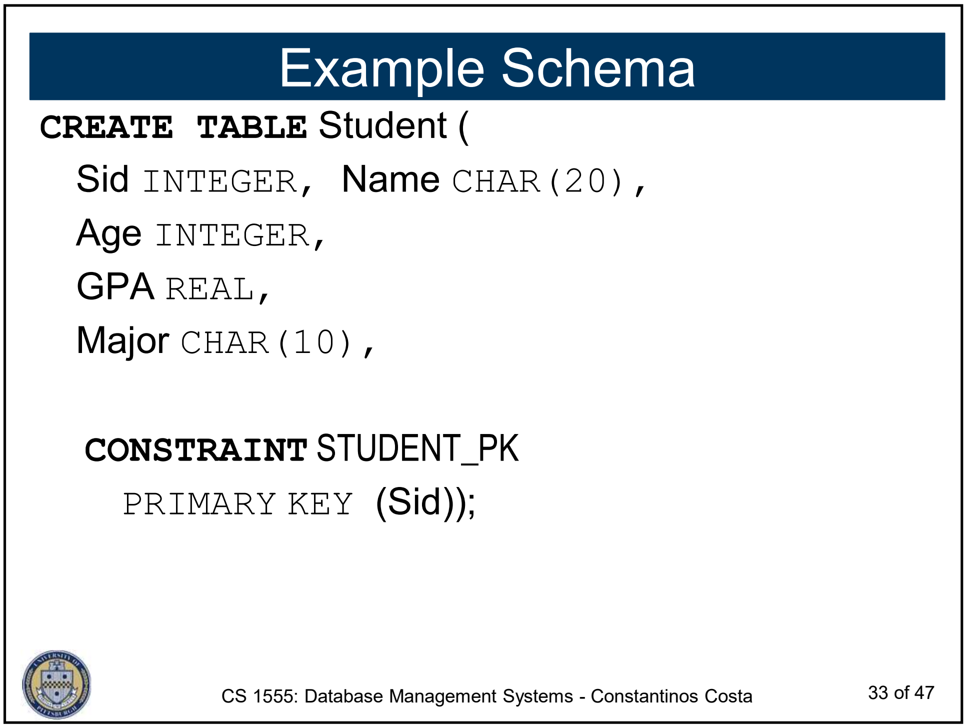
```
CREATE DOMAIN sectno_dom AS SMALLINT;

CREATE DOMAIN gpa_dom DECIMAL (3,2) DEFAULT 0.00;

CREATE DOMAIN ssn_dom CHAR(11)
   CONSTRAINT ssn_dom_value
   CHECK (VALUE BETWEEN '000-00-0000' AND '999-99-9999');
```



Removing a Domain

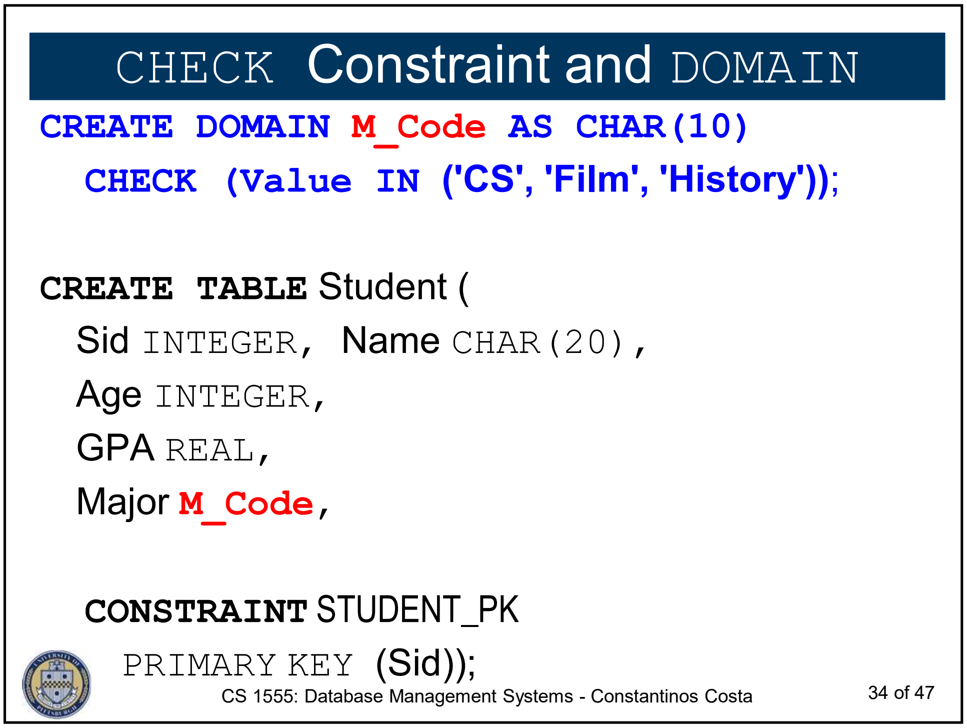
- **DROP DOMAIN** <dname> [**RESTRICT** | **CASCADE**];
  - Restrict: removes the domain it is not used
  - *Cascade*: removes the domain and replaces all its uses to its underlying datatype
- Example:

```
– CREATE DOMAIN gender_dom AS CHAR(1)
      CONSTRAINT gender_dom_value
      CHECK ((VALUE IN ( 'F', 'f', 'M', 'm' )) OR (VALUE IS
NULL));

– DROP DOMAIN gender_dom CASCADE;
```



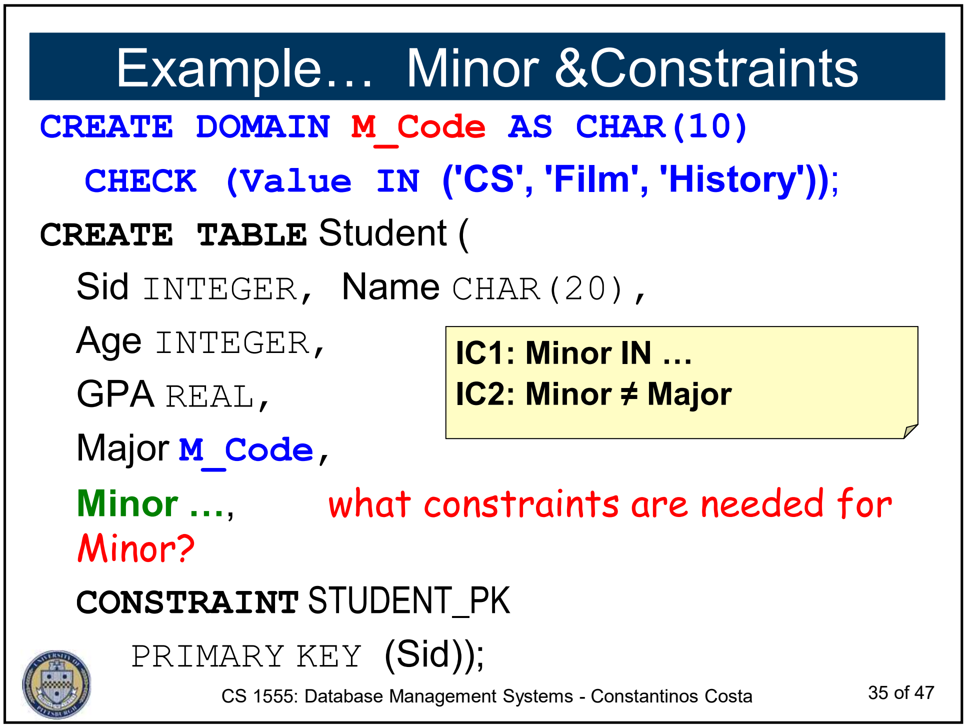
Example Schema

```
CREATE  TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major CHAR(10),

    CONSTRAINT STUDENT_PK
       PRIMARY KEY  (Sid));
```



CHECK Constraint and DOMAIN

```
CREATE  DOMAIN M_Code AS CHAR(10)
    CHECK  (Value IN ('CS', 'Film', 'History'));

CREATE  TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major M_Code,

    CONSTRAINT STUDENT_PK
       PRIMARY KEY  (Sid));
```

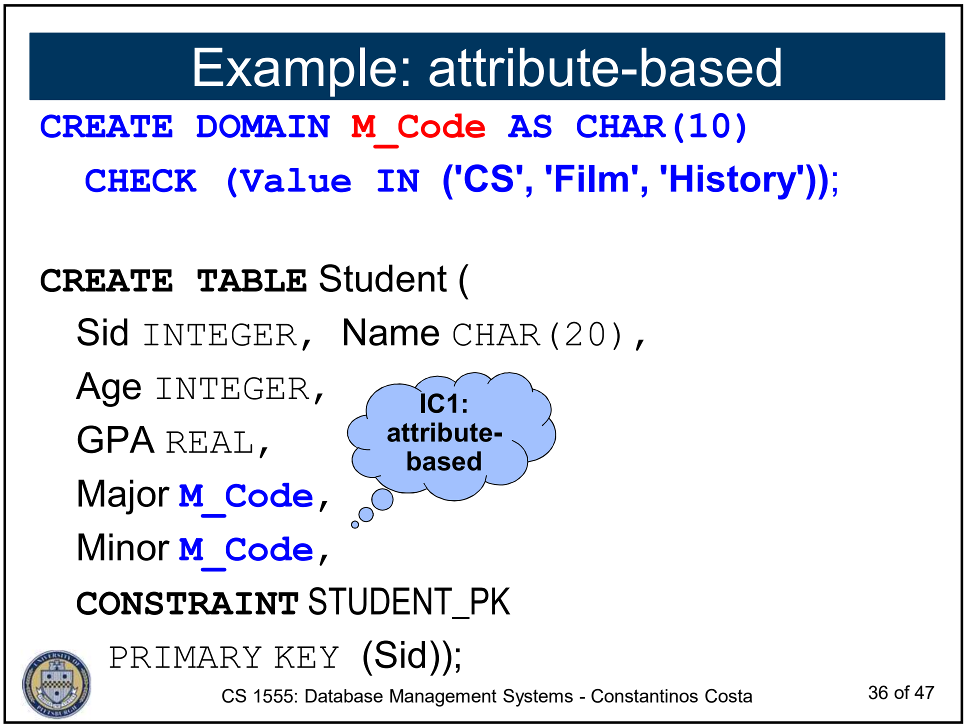


Example… Minor &Constraints

```
CREATE DOMAIN M_Code AS CHAR(10)
  CHECK (Value IN ('CS', 'Film', 'History'));
CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major M_Code,
   Minor …,
   CONSTRAINT STUDENT_PK
     PRIMARY KEY (Sid));
```

IC1: Minor IN …
IC2: Minor ≠ Major

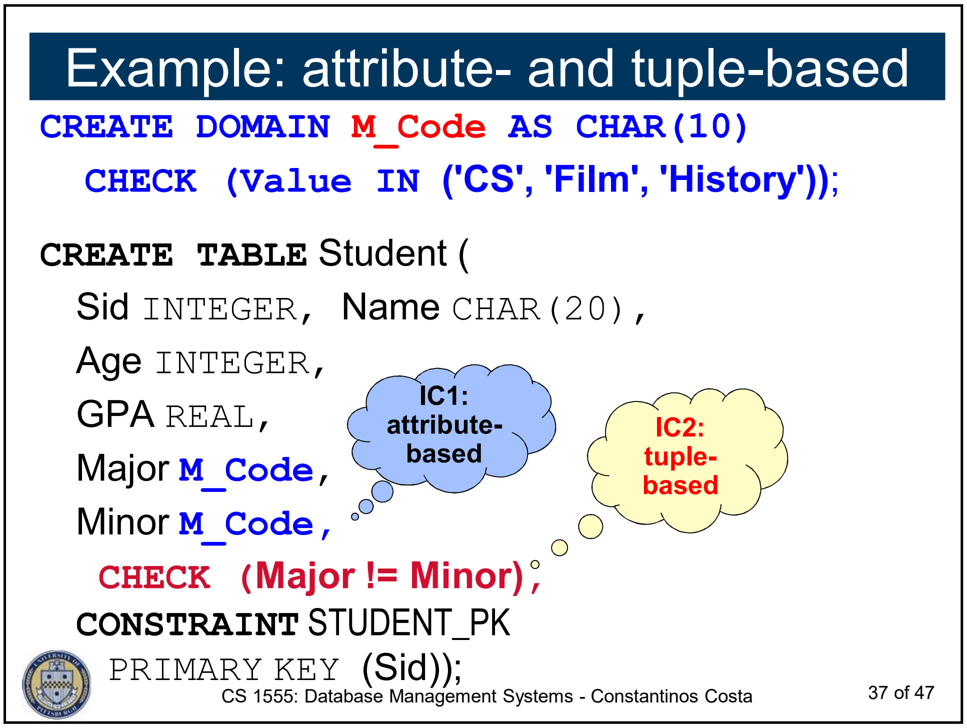
what constraints are needed for Minor?



Example: attribute-based

```
CREATE DOMAIN M_Code AS CHAR(10)
  CHECK (Value IN ('CS', 'Film', 'History'));

CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major M_Code,
   Minor M_Code,
   CONSTRAINT STUDENT_PK
    PRIMARY KEY (Sid));
```

IC1: attribute-based

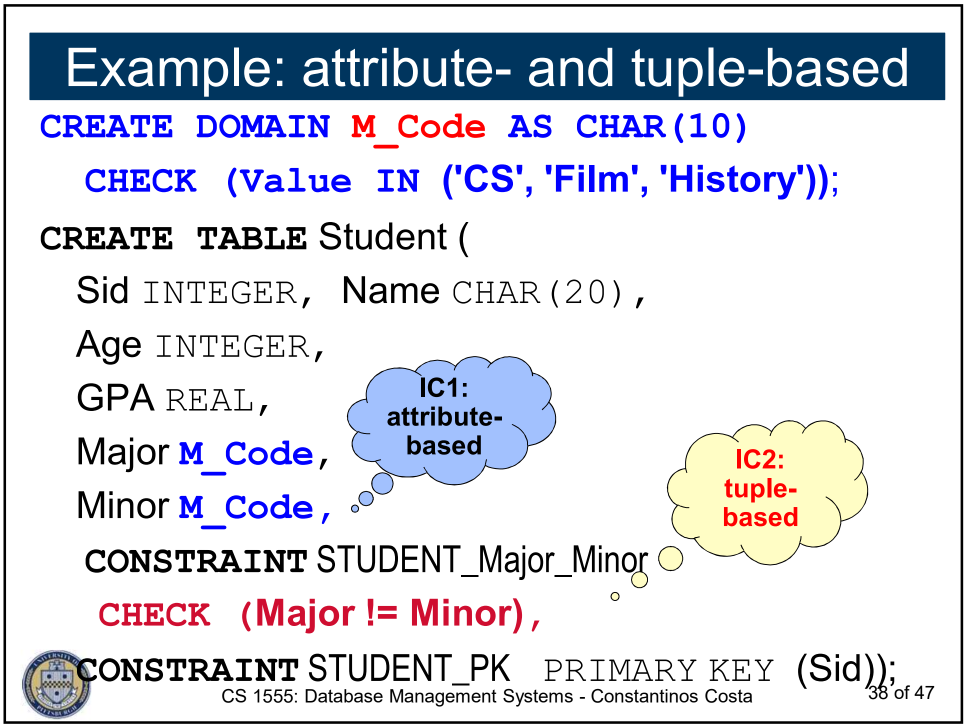


Example: attribute- and tuple-based

```
CREATE DOMAIN M_Code AS CHAR(10)
   CHECK (Value IN ('CS', 'Film', 'History'));

CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major M_Code,
   Minor M_Code,
    CHECK (Major != Minor),
   CONSTRAINT STUDENT_PK
    PRIMARY KEY (Sid));
```

IC1: attribute-based

IC2: tuple-based



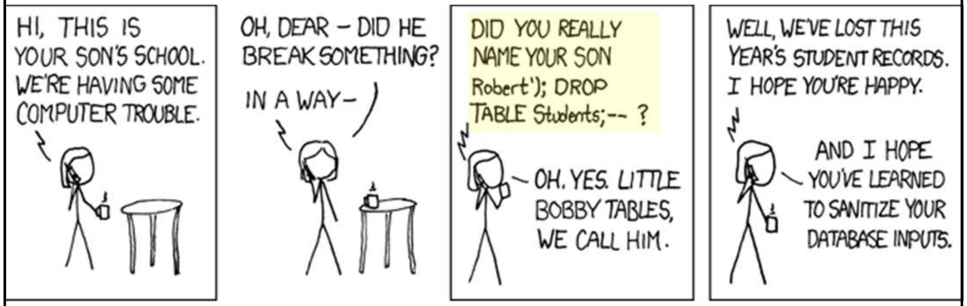
Example: attribute- and tuple-based

```
CREATE DOMAIN M_Code AS CHAR(10)
   CHECK (Value IN ('CS', 'Film', 'History'));
CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major M_Code,
   Minor M_Code,
    CONSTRAINT STUDENT_Major_Minor
    CHECK (Major != Minor),
   CONSTRAINT STUDENT_PK  PRIMARY KEY (Sid));
```

IC1: attribute-based

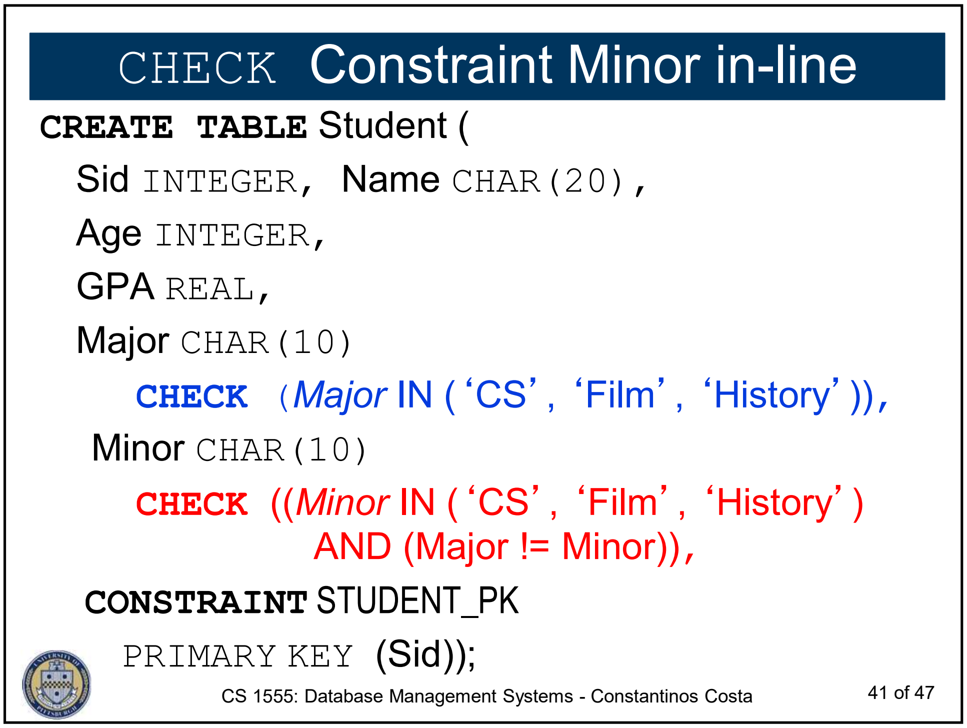
IC2: tuple-based

Be careful with your database inputs ☺


HI, THIS IS YOUR SON'S SCHOOL. WE'RE HAVING SOME COMPUTER TROUBLE.
OH, DEAR – DID HE BREAK SOMETHING?
IN A WAY–
DID YOU REALLY NAME YOUR SON Robert'); DROP TABLE Students;-- ?
OH. YES. LITTLE BOBBY TABLES, WE CALL HIM.
WELL, WE'VE LOST THIS YEAR'S STUDENT RECORDS. I HOPE YOU'RE HAPPY.
AND I HOPE YOU'VE LEARNED TO SANITIZE YOUR DATABASE INPUTS.

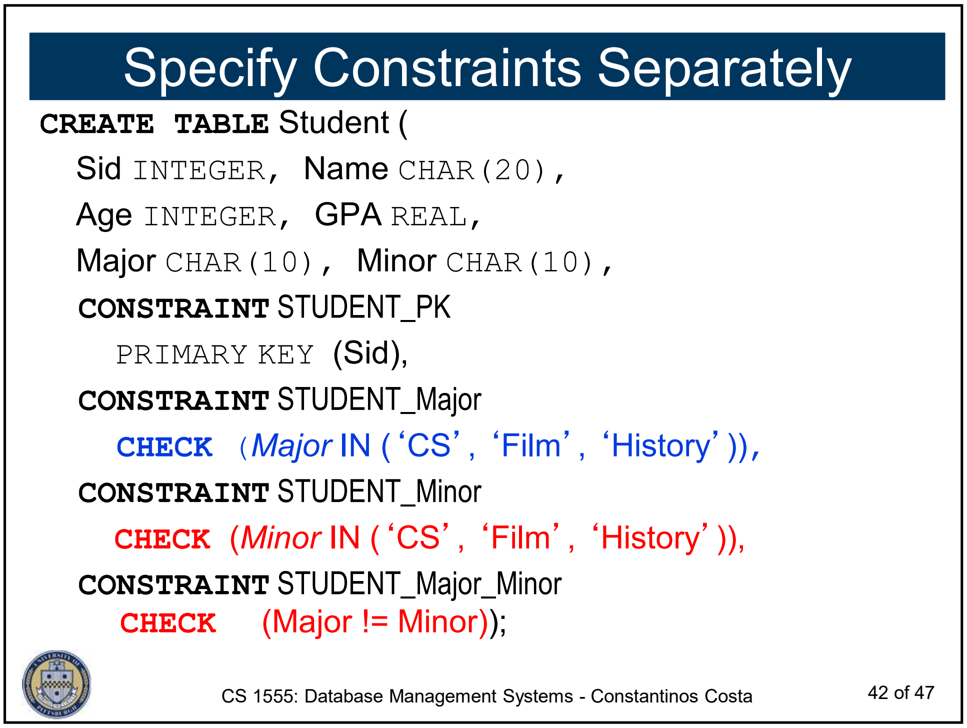
CHECK Constraint Major in-line

```
CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major CHAR(10)
        CHECK  (Major IN (‘CS’, ‘Film’, ‘History’)),

    CONSTRAINT STUDENT_PK
      PRIMARY KEY  (Sid));
```



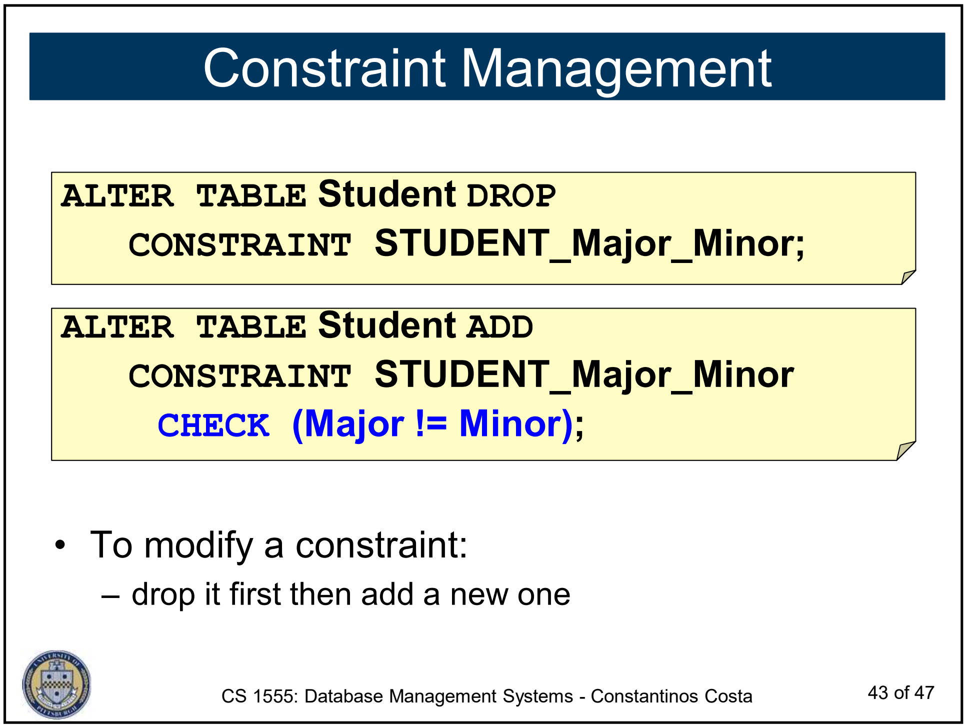
CHECK Constraint Minor in-line

```
CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,
   GPA REAL,
   Major CHAR(10)
        CHECK  (Major IN (‘CS’, ‘Film’, ‘History’)),
    Minor CHAR(10)
        CHECK  ((Minor IN (‘CS’, ‘Film’, ‘History’)
                     AND (Major != Minor)),
    CONSTRAINT STUDENT_PK
       PRIMARY KEY  (Sid));
```



Specify Constraints Separately

```
CREATE TABLE Student (
   Sid INTEGER,  Name CHAR(20),
   Age INTEGER,  GPA REAL,
   Major CHAR(10),  Minor CHAR(10),
   CONSTRAINT STUDENT_PK
      PRIMARY KEY  (Sid),
   CONSTRAINT STUDENT_Major
      CHECK  (Major IN (‘CS’, ‘Film’, ‘History’)),
   CONSTRAINT STUDENT_Minor
      CHECK  (Minor IN (‘CS’, ‘Film’, ‘History’)),
   CONSTRAINT STUDENT_Major_Minor
      CHECK    (Major != Minor));
```



Constraint Management

```
ALTER TABLE Student DROP
    CONSTRAINT STUDENT_Major_Minor;
```

```
ALTER TABLE Student ADD
    CONSTRAINT STUDENT_Major_Minor
      CHECK (Major != Minor);
```

- To modify a constraint:
  - drop it first then add a new one

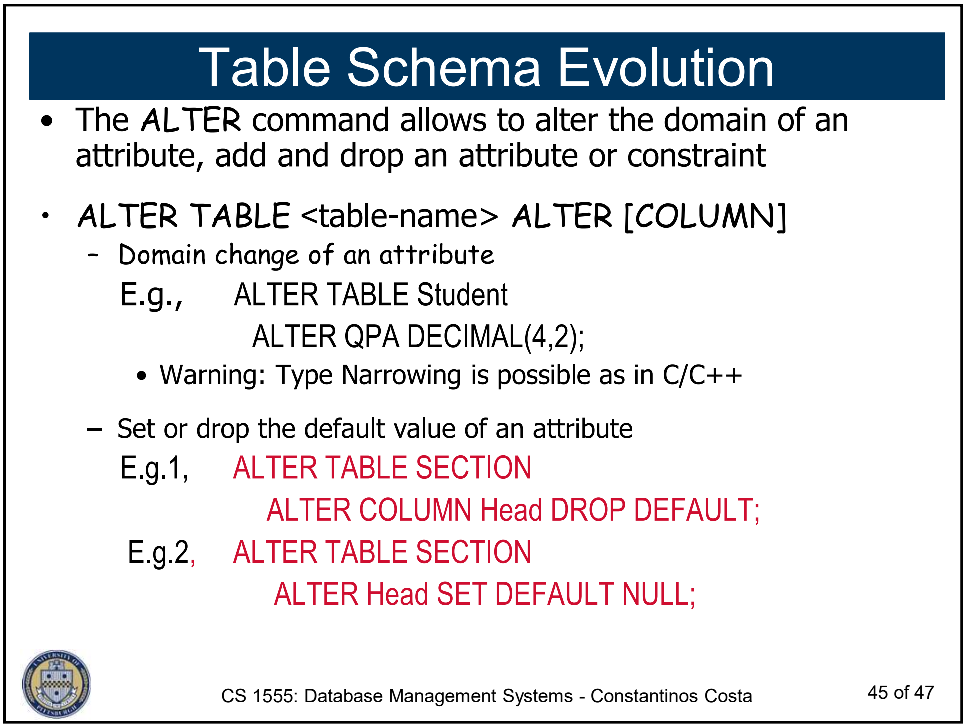


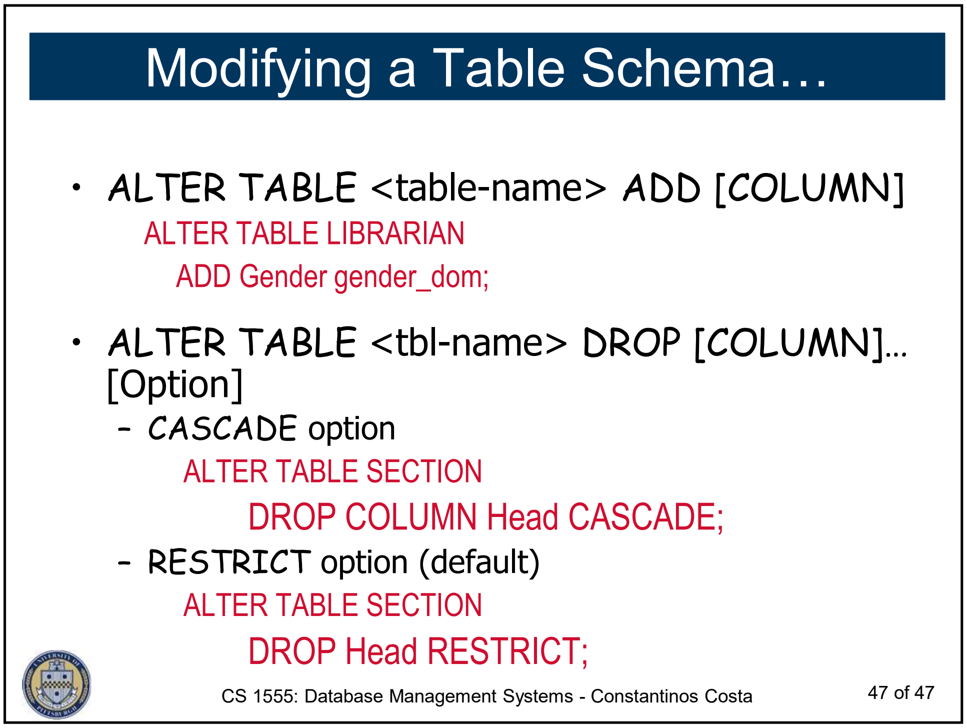
Table Schema Evolution

- The ALTER command allows to alter the domain of an attribute, add and drop an attribute or constraint
- ALTER TABLE <table-name> ALTER [COLUMN]
  - Domain change of an attribute

    E.g., ALTER TABLE Student
    ALTER QPA DECIMAL(4,2);
    - Warning: Type Narrowing is possible as in C/C++
  - Set or drop the default value of an attribute

    E.g.1, ALTER TABLE SECTION
    ALTER COLUMN Head DROP DEFAULT;

    E.g.2, ALTER TABLE SECTION
    ALTER Head SET DEFAULT NULL;



Modifying a Table Schema…

- ALTER TABLE <table-name> ADD [COLUMN]

```
ALTER TABLE LIBRARIAN
    ADD Gender gender_dom;
```

- ALTER TABLE <tbl-name> DROP [COLUMN]… [Option]
  - CASCADE option

```
ALTER TABLE SECTION
        DROP COLUMN Head CASCADE;
```

  - RESTRICT option (default)

```
ALTER TABLE SECTION
        DROP Head RESTRICT;
```