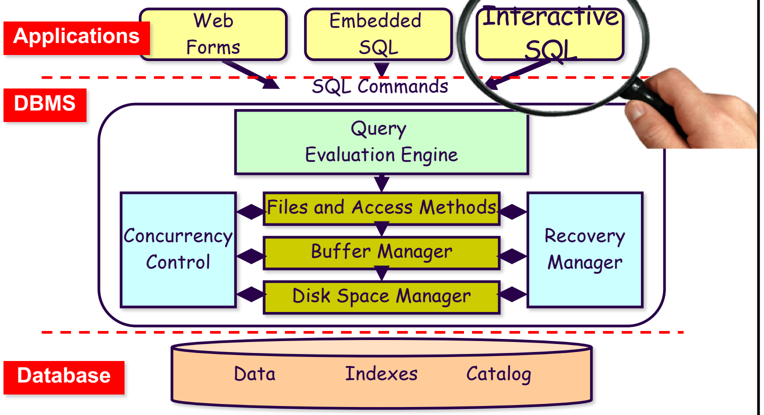
Structured Query Language
SQL - DDL

- SQL Overview
- SQL Datatypes
- DDL statements

Database Management System (DBMS)



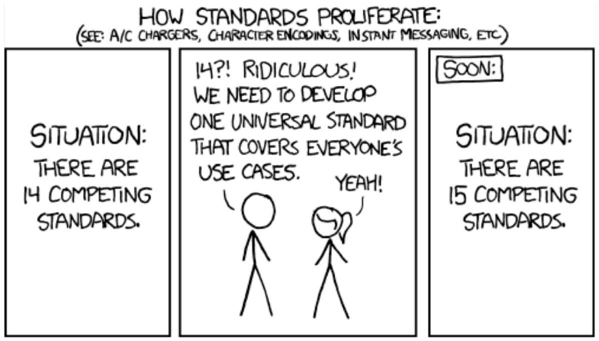
Relational Model - History

- Before: records, pointers, sets, etc.
  - Hierarchical Data Model (IBM IMS,1966-68)
  - Network Data Model (CODASYL DBTG, 1969)
- Introduced by E.F. Codd in 1970
- Revolutionary!
- First systems: 1977-8
  - System R; Ingres
- Turing award in 1981

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 *SIDs*
  - 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

- SQL2, SQL3 support multiple database schemas
- **Catalog** contains the definitions of database 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,...)
  - Authorization
  - Integrity
- Naming of tables: Schema_name.Table_name
- Query: Describe 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)

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)
    -VARCHAR2(n) in Oracle
  - 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: ll
  - 'abc' ll '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)

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 Type & Oracle DATE

- ❑ DATETIME is not a valid ANSI SQL type
- ❑ Not supported by Oracle - Oracle DATE = ANSI TIMESTAMP
- ❑ MySQL DATETIME is used as a TIMESTAMP
  - ▪ MySQL DATETIME supported range is '1000-01-01 00:00:00' to '9999-12-31 23:59:59'
  - ▪ MySQL TIMESTAMP supported range is '1970-01-01 00:00:01' UTC to '2038-01-19 03:14:07' UTC
    - – has varying properties, depending on the MySQL version and the SQL mode the server is running in.
- ❑ Transarc-SQL: No TIMESTAMP
  - ▪ DATETIME: 1753-01-01 to 9999-12-31[hh:mm:ss:nnn]
  - ▪ DATETIME2: 0001-01-01 00:00:00.0000000 to 9999-12-31 23:59:59.9999999

Functions on Dates

- ❑ All systems provide functions under different names
  - ▪ for constructing a date from strings or integers
  - ▪ for extracting out the month, day, or year from a date
  - ▪ for displaying dates in different ways
- ❑ Examples,
  - ▪ CAST(string AS DATE) [SQL2: CAST(<value> AS <type>)]
    e.g., CAST('2002-02-18' AS DATE)
  - ▪ MAKEDATE (int year, int month, int day) or
    DATE (int year, int month, int day)
    e.g., MAKEDATE(1999, 12, 31)
  - ▪ EXTRACT (MONTH/DAY/YEAR FROM <date>) [SQL3]
  - ▪ YEAR(<date>), MONTH(<date>), DAY(<date>)

Constructing Date Functions in Oracle

Oracle	Returns
TO_CHAR(d,format)	character-string equivalent of *d* based on *format*
TO_DATE(s,format)	date corresponding to *s* based on *format*
TO_TIMESTAMP(s, format)	date corresponding to *s* based on *format*

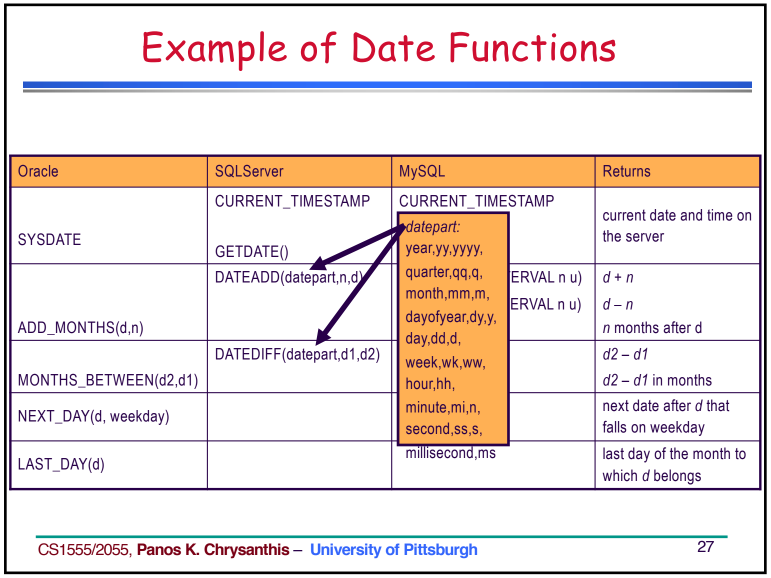
Examples:

- ▪TO_DATE('2011-FEB-18', 'YYYY-MON-DD')
- ▪TO_DATE('02182011' , 'MMDDYYYY')
- ▪TO_CHAR(mydate, DY) → returns sun, mon, tue, wed, thu, fri, sat

Format	Description
MM	Month number
MON	3-letter abbreviation of month
MONTH	Fully spelled-out month
D	Number of days in the week
DD	Number of days in the month
DDD	Number of days in the year
DY	3-letter abbreviation of day of week
DAY	Fully spelled-out day of week
Y, YY, YYY, YYYY	Last 1, 2, 3 or 4 digits of year
HH12, HH24	Hours of the day (1-12 or 0-23)
MI	Minutes of hour
SS	Seconds of minute
AM	Display AM or PM depending on time

Resolving Spec Ambiguity

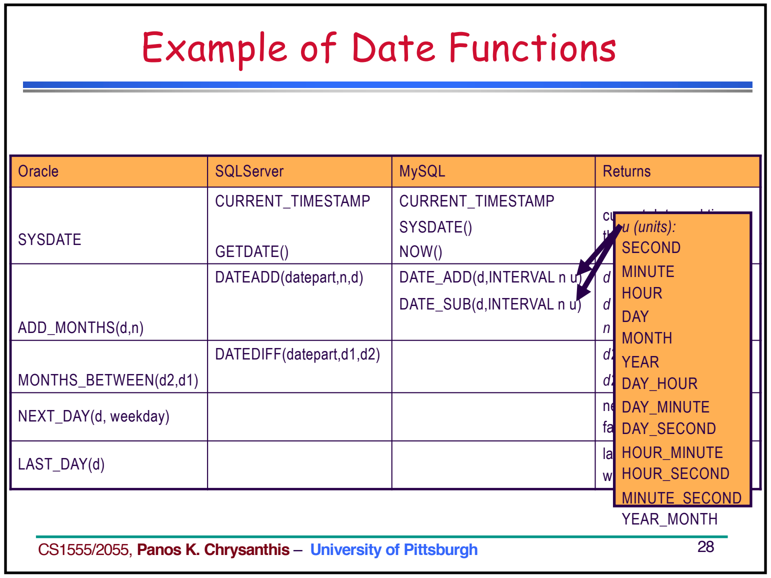
- ❑ TO_DATE('02182011' , 'MMDDYYYY')
- ❑ It parses to the longest keyword.
- ❑ Examples:
  - ▪ 'DYY' = DY and Y
    TO_DATE('WED7', 'DYY') = 01-FEB-17
  - ▪ 'DDDYYYY' = DDD and YYYY
    TO_DATE('3232017', 'DDDYYYY') = 19-NOV-17
  - ▪ 'DYYY' = DY and YY
    TO_DATE('WED17', 'DYYY') = 01-FEB-17



Example of Date Functions

Oracle	SQLServer	MySQL	Returns
SYSDATE	CURRENT_TIMESTAMP GETDATE()	CURRENT_TIMESTAMP	current date and time on the server
ADD_MONTHS(d,n)	DATEADD(datepart,n,d)	ERVAL n u) ERVAL n u)	*d + n* *d – n* *n* months after d
MONTHS_BETWEEN(d2,d1)	DATEDIFF(datepart,d1,d2)		*d2 – d1* *d2 – d1* in months
NEXT_DAY(d, weekday)			next date after *d* that falls on weekday
LAST_DAY(d)			last day of the month to which *d* belongs

datepart:
year,yy,yyyy,
quarter,qq,q,
month,mm,m,
dayofyear,dy,y,
day,dd,d,
week,wk,ww,
hour,hh,
minute,mi,n,
second,ss,s,
millisecond,ms



Example of Date Functions

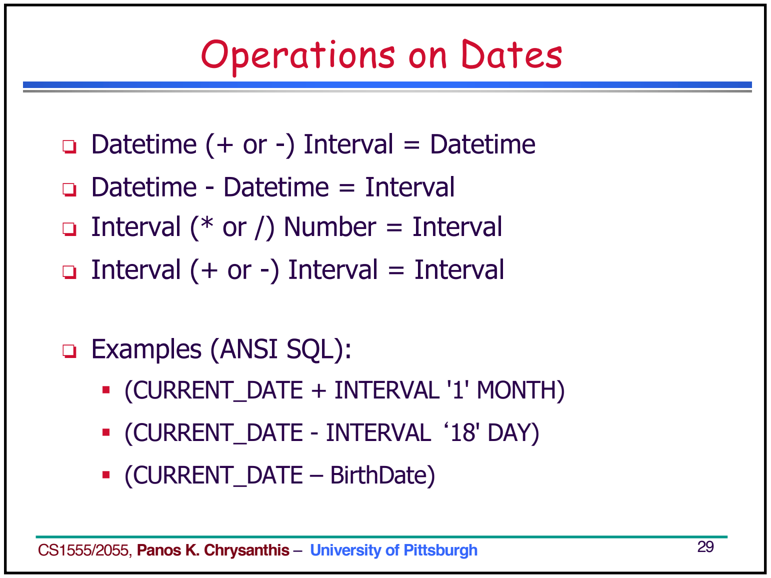
Oracle	SQLServer	MySQL	Returns

SYSDATE	CURRENT_TIMESTAMP
	GETDATE()
		CURRENT_TIMESTAMP
		SYSDATE()
		NOW()
			cu
			th

ADD_MONTHS(d,n)	DATEADD(datepart,n,d)	DATE_ADD(d,INTERVAL n u)
		DATE_SUB(d,INTERVAL n u)
			d
			d
			n

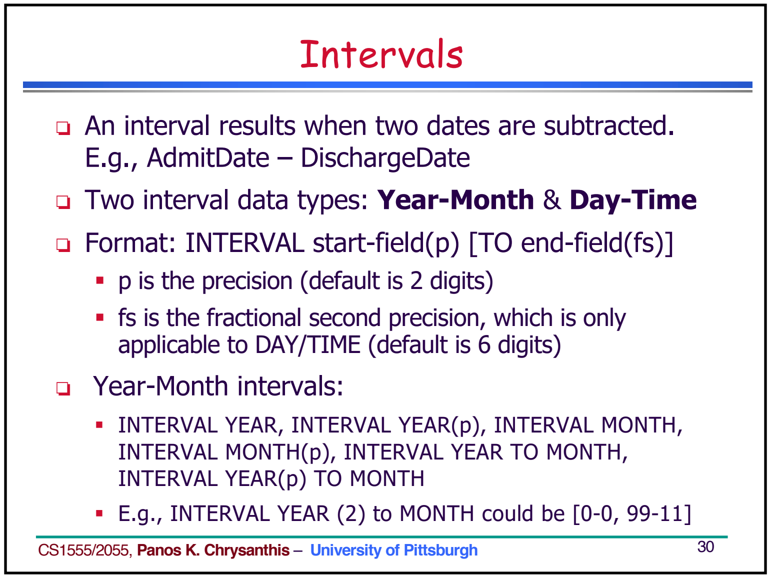
MONTHS_BETWEEN(d2,d1)	DATEDIFF(datepart,d1,d2)		*d2* *d2*
NEXT_DAY(d, weekday)			ne fa
LAST_DAY(d)			la w

u (units):
SECOND
MINUTE
HOUR
DAY
MONTH
YEAR
DAY_HOUR
DAY_MINUTE
DAY_SECOND
HOUR_MINUTE
HOUR_SECOND
MINUTE_SECOND
YEAR_MONTH



Operations on Dates

- 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)



Intervals

- An interval results when two dates are subtracted. E.g., AdmitDate – DischargeDate
- Two interval data types: **Year-Month** & **Day-Time**
- Format: INTERVAL start-field(p) [TO end-field(fs)]
  - p is the precision (default is 2 digits)
  - fs is the fractional second precision, which is only applicable to DAY/TIME (default is 6 digits)
- Year-Month intervals:
  - INTERVAL YEAR, INTERVAL YEAR(p), INTERVAL MONTH, INTERVAL MONTH(p), INTERVAL YEAR TO MONTH, INTERVAL YEAR(p) TO MONTH
  - E.g., INTERVAL YEAR (2) to MONTH could be [0-0, 99-11]

Intervals…

- DAY-TIME intervals: the fields can be a contiguous selection from DAY, HOUR, MINUTE, SECOND
- E.g.,
  - INTERVAL DAY TO HOUR
    - [0:0, 99:23] (day:hours)
  - INTERVAL DAY(1) TO HOUR
    - [0:0, 9:23] (days:hours)
  - INTERVAL DAY TO MINUTE
    - [0:0:0, 99:23:59] (days:hours:minutes)
  - INTERVAL SECOND(8)
  - INTERVAL DAY(5) to SECOND(10)
  - INTERVAL MINUTE(3) to SECOND

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 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)
  )
   IS TABLESPACE {tablespace | users};        -- In Oracle
   ON {filegroup | DEFAULT};                  -- In SQLServer
```

Table Schema (MySQL)

```
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
```

Other options: ARCHIVE, CSV, HEAP, Memory, myisam, ndbcluster

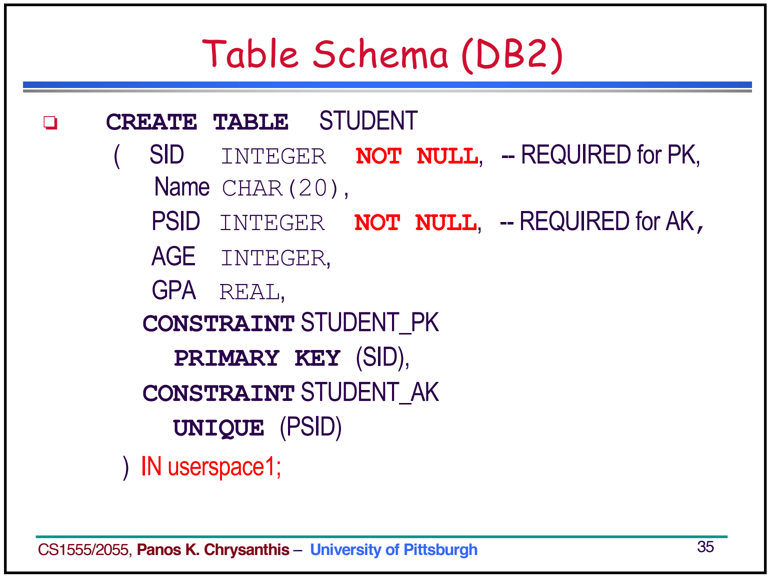
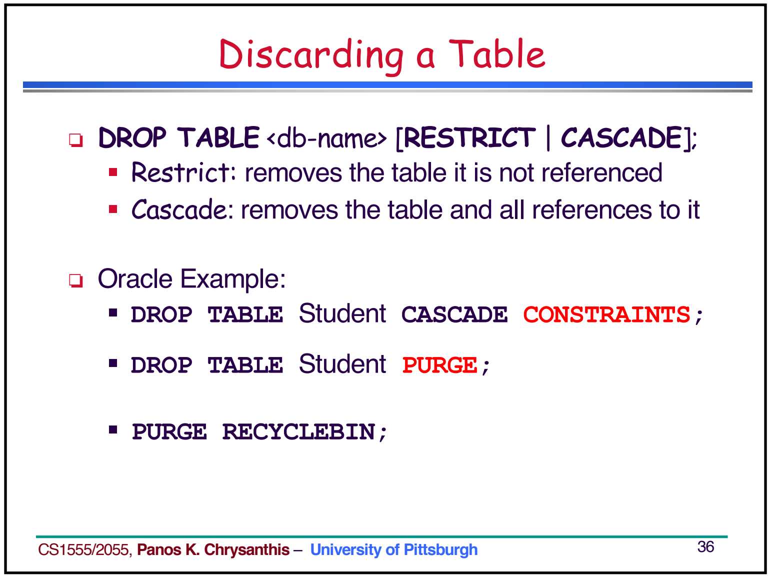


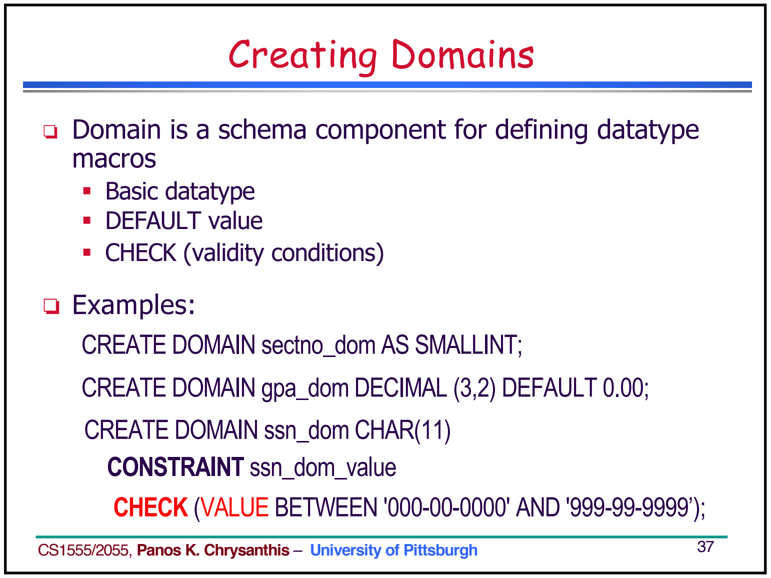
Table Schema (DB2)

```
❑  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

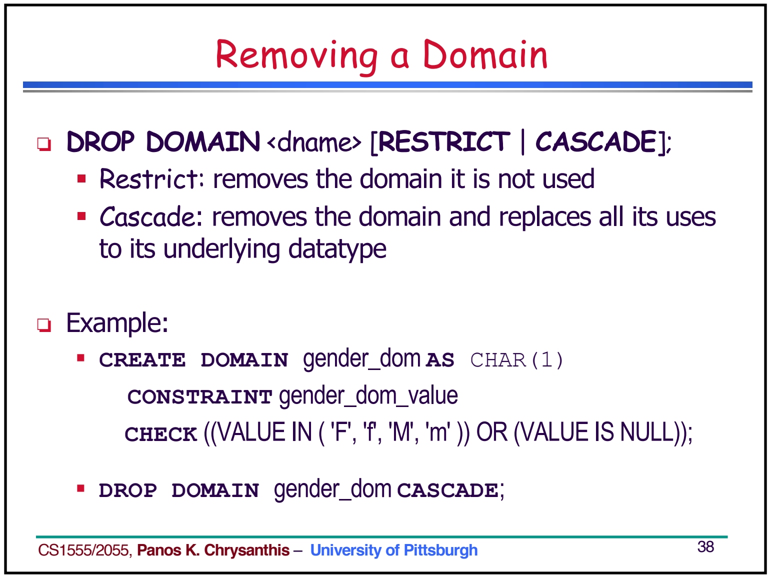
- **DROP TABLE** <db-name> [**RESTRICT** | ***CASCADE***];
  - *Restrict*: removes the table it is not referenced
  - *Cascade*: removes the table and all references to it
- Oracle Example:
  - `DROP TABLE Student CASCADE CONSTRAINTS;`
  - `DROP TABLE Student PURGE;`
  - `PURGE RECYCLEBIN;`



Creating Domains

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

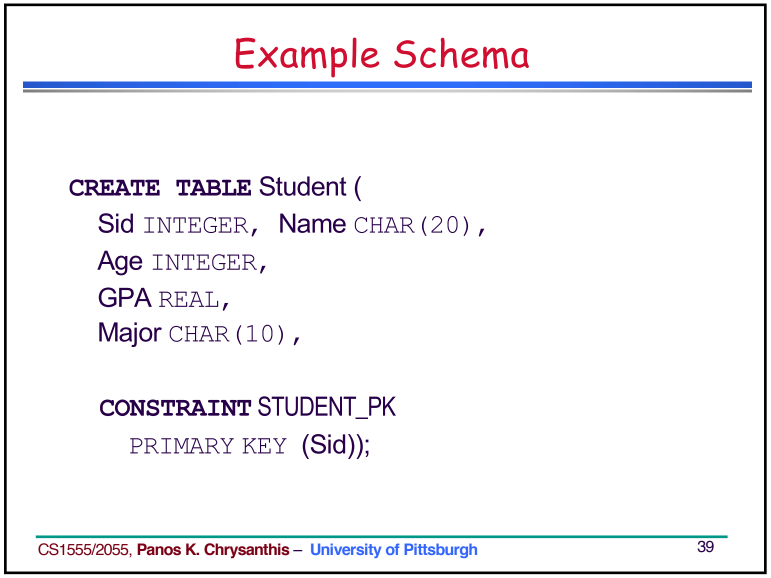
```
CREATE DOMAIN sectno_dom AS SMALLINT;

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

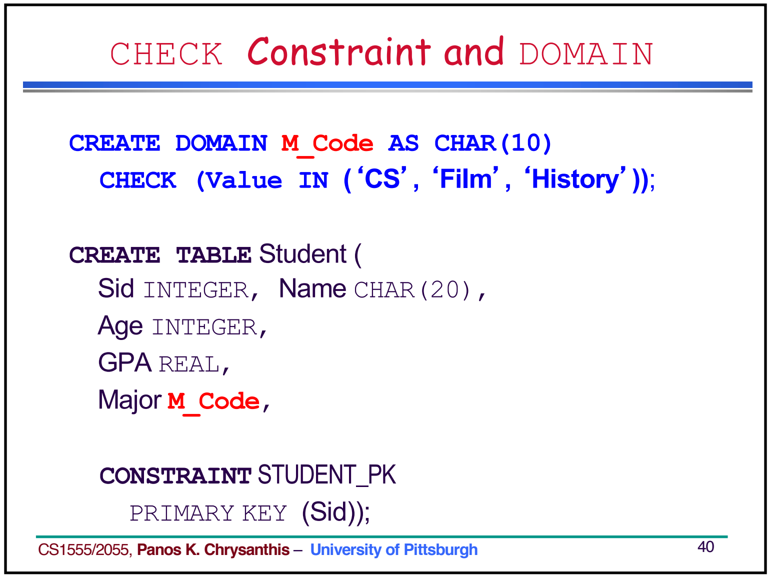
- **DROP DOMAIN** <dname> [**RESTRICT** | ***CASCADE***];
  - *Restrict*: removes the domain it is not used
  - *Cascade*: removes the domain and replaces all its uses to its underlying datatype
- Example:

  -
    ```
    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;`



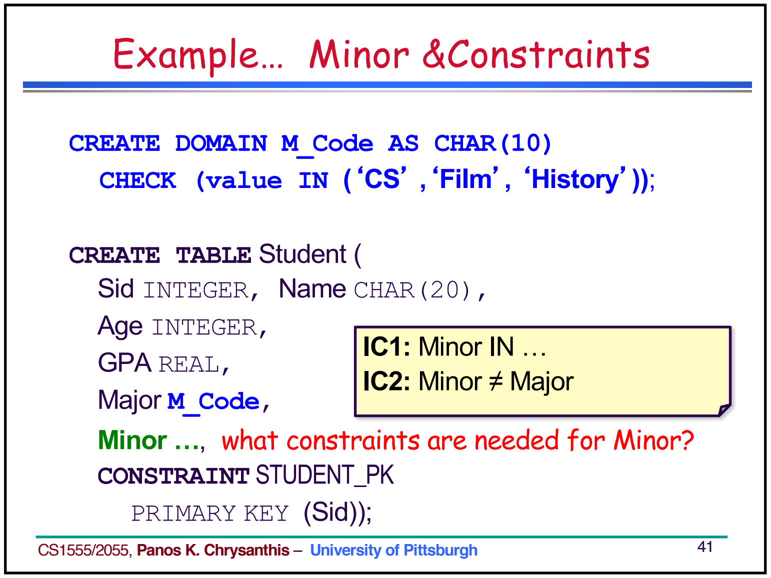
Example Schema

```
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

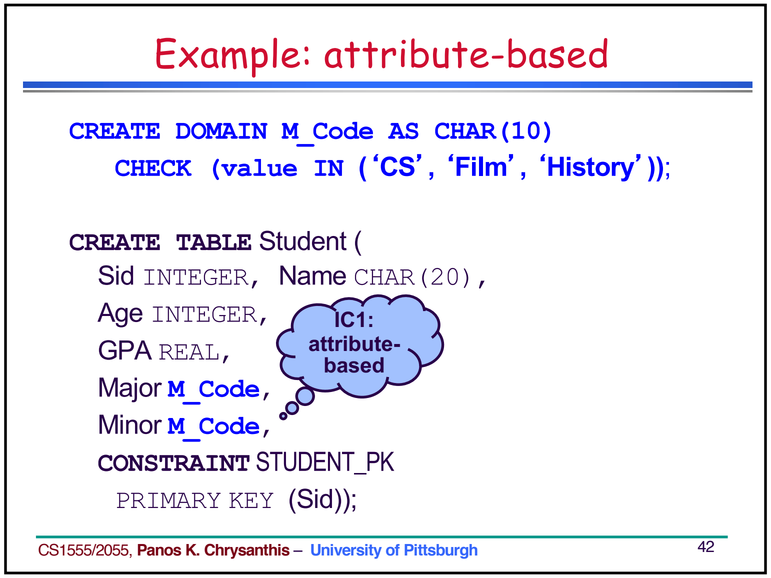
```
CREATE DOMAIN M_Code AS CHAR(10)
   CHECK (Value IN (‘CS’, ‘Film’, ‘History’));

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 …,  what constraints are needed for Minor?
   CONSTRAINT STUDENT_PK
      PRIMARY KEY (Sid));
```

IC1: Minor IN …
IC2: Minor ≠ Major



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

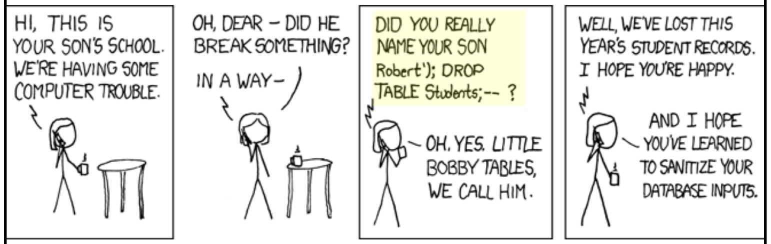
Example: attribute- and tuple-based

```
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

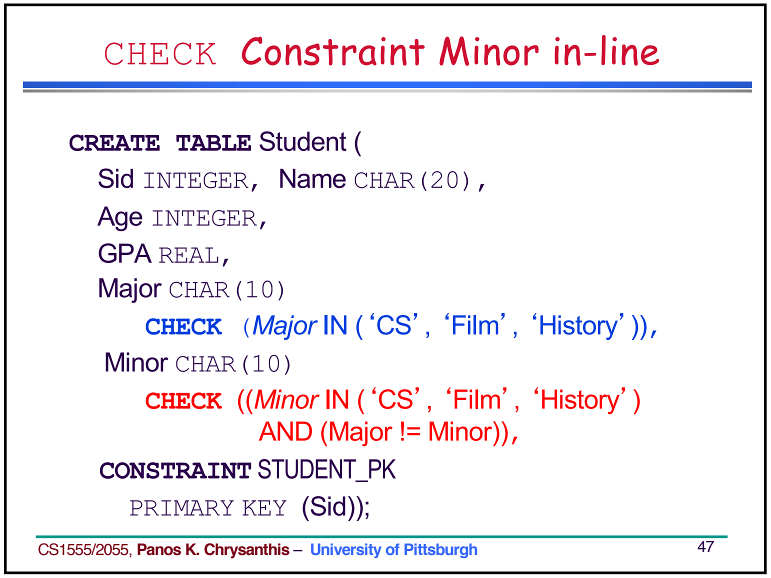
IC2: tuple-based

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.

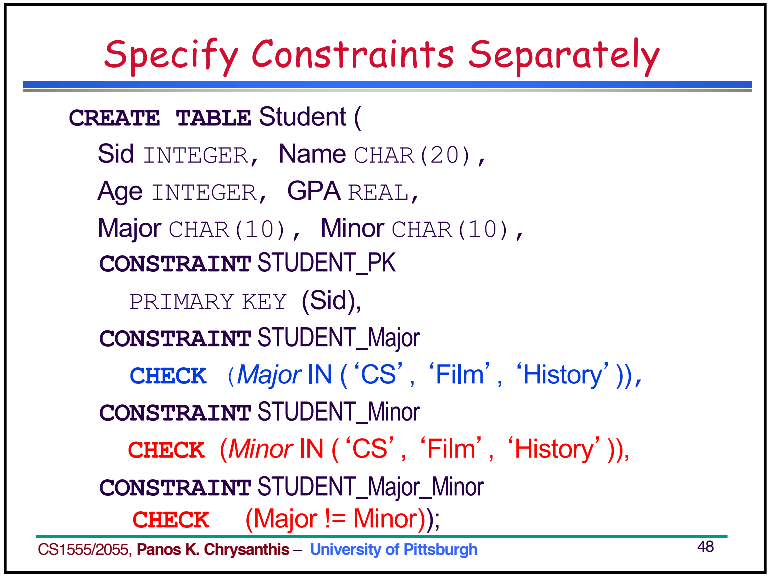
CHECK Constraint Major in-line

```
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));
```



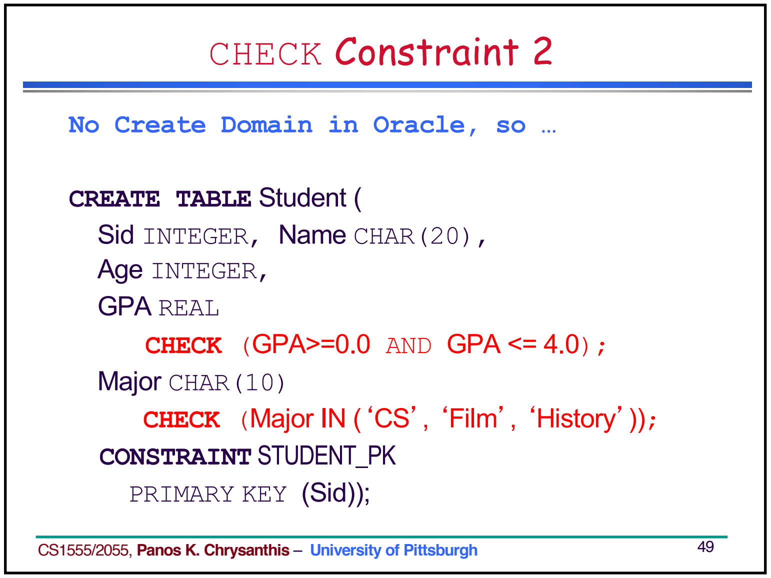
CHECK Constraint Minor in-line

```
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

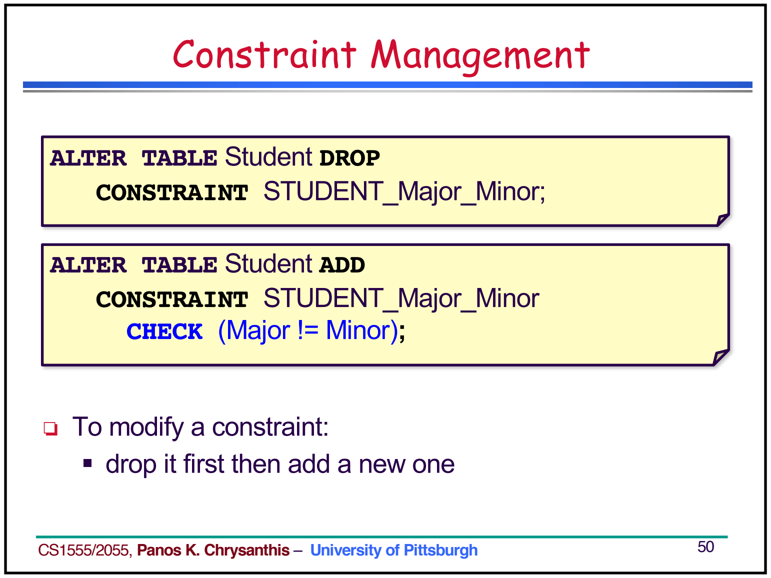
```
CREATE TABLE Student (
    Sid INTEGER,  Name CHAR(20),
    Age INTEGER,  GPA REAL,
    Major CHAR(10),  Minor CHAR(10),
    CONSTRAINT STUDENT_PK
       PRIMARY KEY  (Sid),
    CONSTRAINT STUDENT_Major
       CHECK  (Major IN ('CS', 'Film', 'History')),
    CONSTRAINT STUDENT_Minor
       CHECK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



CHECK Constraint 2

No Create Domain in Oracle, so …

```
CREATE TABLE Student (
    Sid INTEGER,  Name CHAR(20),
    Age INTEGER,
    GPA REAL
         CHECK  (GPA>=0.0  AND  GPA <= 4.0);
    Major CHAR(10)
         CHECK  (Major IN ('CS', 'Film', 'History'));
    CONSTRAINT STUDENT_PK
       PRIMARY KEY  (Sid));
```



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;

Table Schema Evolution in Oracle

- ALTER TABLE <table-name> MODIFY [COLUMN]
  - Domain change of an attribute

    E.g., ALTER TABLE Student
    MODIFY QPA DECIMAL(4,2);
  - Set or drop the default value of an attribute

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

    E.g.2, ALTER TABLE SECTION
    MODIFY 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;