An Introduction to SQL

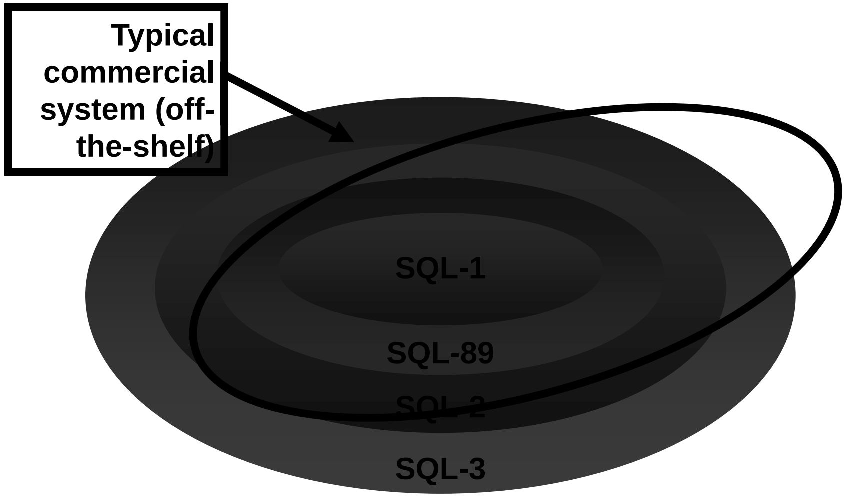
SQL

- The name stands for *Structured Query Language*
- It comes into two flavors:
  - DDL: domain definitions, tables, indexes, authorizations, views, constraints, procedures, triggers.
  - DML: query, insert/delete/update, transactions
- History:
  - First proposal: SEQUEL (IBM Research, 1974)
  - First commercial implementation: SQL/DS (IBM, 1981)

SQL Standardization

- Standardization has been the success of SQL (within ANSI and ISO)
  - Since 1983, standard de facto
  - First official release in 1986 (SQL-1), revised in 1989 (SQL-89)
  - Second release in 1992 (SQL-2 or SQL-92)
  - Third release in 1999 (SQL-3 or SQL:1999)
- SQL-92 has 3 different levels of standardization:
  - Entry SQL (very close to SQL-89)
  - Intermediate SQL
  - Full SQL
- Most commercial systems comply to the entry level specifications, and offer more proprietary extensions for advanced functionalities.

SQL Standards vs. Commercial Systems



Typical commercial system (off-the-shelf)

SQL-1

SQL-89

SQL-2

SQL-3

SQL

- SQL queries are declarative:
  - user says which information should be retrieved, but not how.
- Queries are translated/optimized by the query optimizer into an internal, procedural language.
- Programmer focuses on meaning, not on efficiency.
- This is the most relevant aspect of relational databases.

Schema Definition in SQL

Domains

- A domain specifies the set of permitted values for an attribute:
  - similar to data type definition on a programming language.
- Two categories:
  - Elementary (predefined by the standard, elementary or built-in):
    - SQL-2 comes with 6 types.
  - user-defined.

Elementary Domain, 1

- Characters
  - Character or string;
  - String may have variable length;
  - May use a different character set (e.g.., Latin, Greek, Cyrillic, etc.)
  - Syntax:

    `character` [`varying`] [(*Length*)]
    [`character set` *Set*]
  - Alternatively, know as `char` and `varchar`, for `character` and `character varying`, respectively.
  - Examples:
    - `char(6)`
    - `varchar(50)`

Elementary Domain, 2

- Bit
  - Boolean values (true/false), one bit o a sequence of bits (also variable length)
  - Syntax:

    `bit` [`varying`] [(*length*)]
- Numerical domains:
  - Exact values, integer, float (rational)
  - 4 alternatives:

    `numeric` [(*Precision* [, *Scale*])]

    `decimal` [(*Precision* [, *Scale*])]

    `integer`

    `smallint`

Elementary Domain, 3

- Approximated numerical values:
  - real;
  - floating point : mantix + exp

    `float` [(*Precision*)]

    `real`

    `double precision`

Elementary Domain, 4

- Timestamp
  - `date (year, month, day)`
  - `time` [(*Precision*)] [`with time zone`] : (`hour, minute, second`)
  - `timestamp` [(*Precision*)] [`with time zone`]
- Interval

  `interval` *FirstTimeUnit* [`to` *SecondTimeUnit*]
  - Two groups of time unit:
    - year, month
    - day, hour, minute, second
  - Examples:
    - `interval year to month`
    - `interval second`

Elementary Domain, 5

- New domains in SQL:1999
  - `BLOB` Binary Large Object
  - `CLOB` Character Large Object
- SQL:1999 introduces some contructors (`REF`, `ARRAY`, `ROW`; beyond the pure relational data model, not considered any further)

User Defined, 1

- Similar to data type definition in a programming language: permitted values for an object.
- A domain features:
  - name
  - elementary domain
  - default value
  - constraints
- Syntax:

  `create domain` *DomainName* `as` *ElementaryDomain* [*DefaultValue*] [*Constraints*]
- Example:

```
create domain Mark as smallint default null
```

User Defined, 2

- Wrt programming languages:

  + constraints, default values , basic domains richer

  - no contructor (just renaming of the domain)
- Useful to define domains and to make the schema more readable.

Default Values

- Define the value of an attribute in case no other value is defined at tuple insertion time.
- Syntax:

  `default` < *Value* | `user` | `null` >
- *Value* is a value compatible with the domain, depicted as a constant or as an expression.
- `user` is the login of the user issuing the command.

“null”

`null`

- Polymorphic value (belonging to any domain), meaning “unknown” value:
  - Not known to the database
    - e.g. birth date
  - Does not apply
    - e.g. driving license for under 18
  - Does/doesn’t apply:
    - e.g. driving license for over 18

Intra-relation Constraints

- Constraints rule the values for every instance od the database.
- Intra-relation constraint in one relation (table), and can be further distinguished as tuple-level or table-level
  - `not null` (over one attribute, only: tuple level)
  - `unique`: enables one to define candidate keys (at the table level);
  - syntax:
    - for one attribute, only: `unique`, after the domain;
    - for more attributes: `unique( Attribute {, Attribute } )`
  - `primary key`: defines the primary key (once for a table; implies `not null`); same syntax as for `unique`
  - `check`: see next

Domain Definition

```
create domain NewsPaperPrice as decimal
     default 0,90
     not null
```

Intra-relation Constraints: Examples

- Every couple of attributes Name and Surname univocally identifies every tuple:

```
Name character(20) not null,
Surname character(20) not null,
unique(Name,Surname)
```

- Differenced from (more restrictive):

```
Name character(20) not null unique,
Surname character(20) not null unique,
```

Schema Definition

- A *schema* is a collection of objects:
  - domain, table, index, assertion, view, privilege.
- A schema has a name and an owner.
- Syntax:

  `create schema` [*SchemaName*]
  [[`authorization`] *Authorized*]
  {*SchemaElementDefinition*}

Table Definition

- A SQL tables consists of:
  - an ordered set of attributes;
  - a set of constraints (possibly, empty).
- Command `create table`
  - defines the schema of a relation and creates an empty instance.
- Syntax:

`create table` *TableName*
(
AttributeName Domain [*DefaultValue*] [*Constraints*]
{, *AttributeName Domain* [*DefaultValue*] [*Constraints*] }
[*OtherConstraints*]
)

Example (1)

```
create table Student
 (Id        character(6) primary key,
  Name      varchar(30) not null,
  City      varchar(20),
  Dept      char(3) )
```

Example (2)

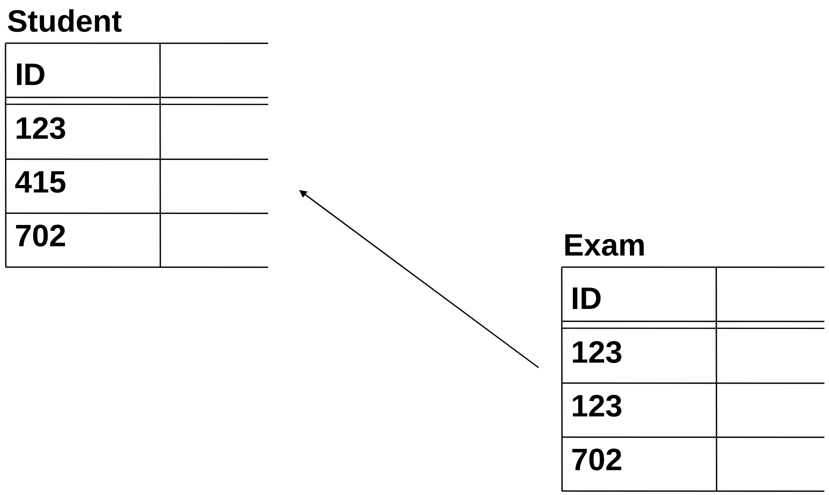
```
create table Exam
 (  Id         char(6),
    Course     char(6),
    Date       date  not null,
    Mark       smallint not null,
    primary key(Id,Course) )

 create table Course
 (  Code     char(6) primary key,
    Name     varchar(30) not null,
    Teacher  varchar(20) )
```

Referential Integrity

Referential Integrity

- `references` and `foreign key` for referential integrity:
  - for one attribute only:
    `references` after the domain
  - for several attributes
    `foreign key` (*Attribute1* {, *Attribute2* })
    `references` …
- Foreign key in the child table must represent values of the key attributes of the parent table.
- Reaction policies can be defined to manage referential integrity violations.

Example: Student - Exam



Student

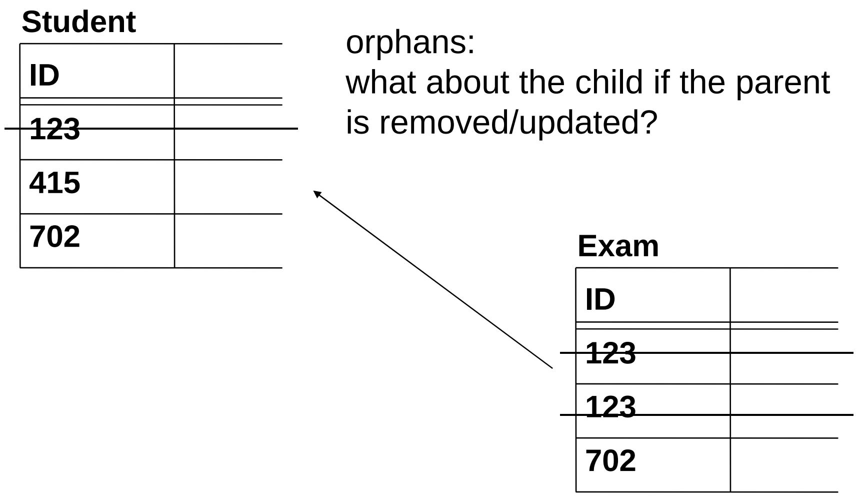
ID	
123	
415	
702	

Exam

ID	
123	
123	
702	

Orphans

orphans:
what about the child if the parent is removed/updated?



Student

ID	
~~123~~	
415	
702	

Exam

ID	
~~123~~	
~~123~~	
702	

Orphans

- Reactions apply on child table(s) after the changes to the parent table.
- Violations can derive from (1) updates of the attribute under consideration OR (2) tuple deletion
- Reaction policies:
  - `cascade`: propagates the update
  - `set null`: forces to null the value of the attribute
  - `set default`: forces the attribute to the default value
  - `no action`: operation is blocked
- Reactions can depend on the occurring event. Syntax:

```
on <  delete | update >
   < cascade | set null | set default | no action >
```

Orphans: Delete

What about exams when I delete a student?

- **`cascade`**
  exams will be deleted, too
- **`set null`**
  the Student.Id is set to NULL for the exams belonging to the deleted student
- **`set default`**
  the Student.Id is set to the default value for the exams belonging to the deleted student
- **`no action`**
  the student can’t be deleted (the delete command is aborted)

Orphans: Update

What about exams when I update the Id of a student?

- **`cascade`**
  Id in the Exam table is updated for the exams belonging to the updated student
- **`set null`**
  Id in the Exam table is set to NULL for the exams belonging to the updated student
- **`set default`**
  Id in the Exam table is set to the default value for the exams belonging to the updated student
- **`no action`**
  Id of the student can’t be updated (the update command is aborted)

Table Definition

```
create table Exam
   ( ....
     ....
     foreign key Id
       references Student
        on delete cascade
        on update cascade)
```

One Child of Several Parents

```
create table Exam
( ....
    primary key(Id,Code)
    foreign key Id
      references Student
        on delete cascade
        on update cascade
    foreign key Code
      references Course
        on delete no action
        on update no action )
```

A Wrong Intance

Id	Name	City	Dept
123			
415			
702			

Exam

	Id	Course Id	Date	Mark
	123	1	7-9-97	10
	123	2	8-1-98	8
key violation	~~123~~	~~2~~	~~1-8-97~~	~~8~~
	702	2	7-9-97	10
NULL violation	~~702~~	~~1~~	~~NULL~~	~~NULL~~
referencial integrity violation	~~714~~	~~1~~	~~7-9-97~~	~~8~~

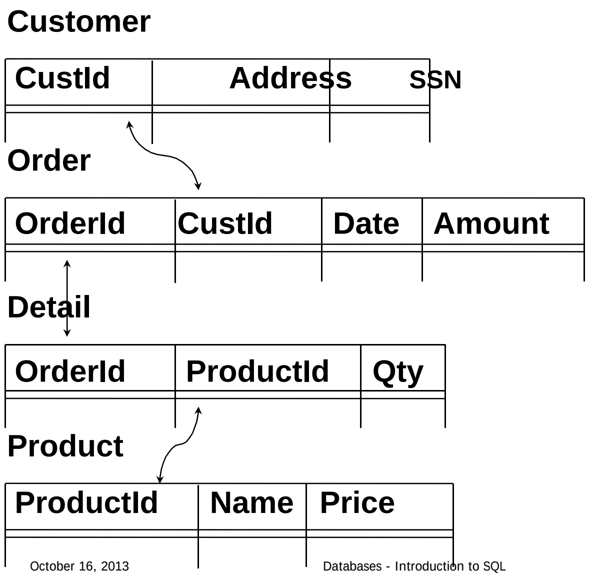
A Correct Instance

Id	Name	City	Dept
123			
415			
702			

Exam

Id	Course Id	Date	Mark
123	1	7-9-97	10
123	2	8-1-98	8
702	2	7-9-97	10

Order Management



Customer

```
create table Customer
 ( CustId    char(6)  primary key,
   Address   char(50),
   SSN       char(12) unique )
```

Warning! Differences exist:

```
CustId primary key, SSN unique
primary key (CustId, SSN)
CustId unique, SSN primary key
```

Order

```
create table Order
(  OrderId   char(6)  primary key,
   CustId    char(6)  not null
                     default='999999',
   Date     date,
   Amount   integer,
    foreign key CustId
      references Customer
         on delete set default
         on update set default)
```

Detail

```
create table Detail
(  OrderId  char(6),
   ProductId char(6),
   Qty      smallint,
    primary key(OrderId, ProductId)
    foreign key OrderId
      references Order
        on delete cascade
        on update cascade
    foreign key ProductId
      references Product
        on delete no action
        on update no action)
```

Product

```
create table Product
 ( ProductId char(6)  primary key,
   Name      char(20),
   Price     smallint )
```

Index

- An index makes data access efficient:
  - – create index
  - – e.g.: create index DataIx on Order(Date)

  - – create unique index
  - – **e.g.**: create unique index OrdKey on Order(Orderid)

Schema Updates

Schema Updates

- They are needed to maintain the database structure in case of new requirements, needs.
- SQL commands:
  - `alter (alter domain ..., alter table …)` modify existing objects
  - `drop`

    ```
    drop <schema|domain|table|view|assertion >
          ComponentName [ restrict | cascade ]
    ```

    remove existing objects

Object Update - DDL

- **`alter`**
  - Applies to domains and tables
    - **e.g.:**
      ```
      alter table Order
         add column InvoiceNumber char(6)
      ```
    - **e.g.:**
      ```
      alter table Order
         alter column Amount
         set default 0
      ```
    - **e.g.:**
      ```
      alter table Order
         drop column Date
      ```

Object Deletion - DDL

- **`drop`**
  - Applies to domains, tables, indexes, views, assertions, procedures, trigger
    - **e.g.**: **`drop table Order`**
    - **e.g.: `drop index DataIx`**

- Options `restrict` and `cascade`
  - **`restrict`**: aborts the `drop` command if objects include any instance
  - **`cascade`**: extends the `drop` command to remove linked objects, too

Catalogs

- The catalog is a data dictionary, i.e. a description of the structures of the data stored by the database.
- Is its, in turn, a relational table:
  - every users knows the relational data;
  - the system manages relational tables in an efficient way;
  - every system stores its catalog by its data model. An object oriented DBMS will store its catalogue by objects.
- The SQL-2 standard organizes the catalog into two levels:
  - **`Definition_Schema`** (made up by tables)
  - **`Information_Schema`** (made up by views)

Information Schema

- Th Information_Schema includes views such as:
  - Domains
  - Domain_Constraints
  - Tables
  - Views
  - Columns
  - ....
- SQL-2 comes with 23 views.

The View “Columns”

- As an example, the view “Columns” may have a structure like the following one:
  - Table_Name
  - Column_Name
  - Ordinal_Position
  - Column_Default
  - Is_Nullable

  (ad many others attributes)

The Catalog is Reflexive

- The catalog is reflexive (the data structures of the catalog are described by the catalog itself)
- Every DDL command is performed by suitable DML commands which operate on the schema of the database.
- This does **NOT** mean the DDL is unuseful!
- The DDL enables the user to describe objects of the schema in a way that is:
  - reliable;
  - consistent;
  - efficient;
  - portable
- **NEVER** make changes directly to the catalog (or you will have to reinstall everything)