II) RELATIONAL DATABASES

The Relational Model

- **Domain**: A set of atomic values such as strings, integers, reals, etc.
- **Tuple**: An element of the Cartesian product of the different domains
- **Relation**: A subset of the Cartesian product of the different domains (i.e., a set of tuples)
  - The schema of a relation is composed by its name and a list of attribute names and their domains
- **Database**: A set of relations
  - The database schema is the set of all relation schemas
### A Relational Database Example

#### ARTIST

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Style</th>
<th>Live-Time</th>
<th>Nationality</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Pablo Picasso</td>
<td>Cubism</td>
<td>1881-1973</td>
<td>Spain</td>
</tr>
<tr>
<td>#2</td>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>1840-1926</td>
<td>France</td>
</tr>
<tr>
<td>#3</td>
<td>Kasimir Malevich</td>
<td>Geom. Abstract</td>
<td>1878-1935</td>
<td>Russia</td>
</tr>
<tr>
<td>#4</td>
<td>Alberto Giacometti</td>
<td>Surrealism</td>
<td>1901-1966</td>
<td>Swiss</td>
</tr>
</tbody>
</table>

**ARTIST** (Code:INT, Name:STRING, Style:STRING, Live-Time:STRING, Nationality:STRING)

- **Relation**: Table
- **Cardinality**: Number of rows
- **Arity**: Number of columns
- **Attribute**: Column

---

#### PAINTING

<table>
<thead>
<tr>
<th>Code</th>
<th>Painter</th>
<th>Name</th>
<th>Material</th>
<th>Date</th>
<th>Museum</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>#2</td>
<td>Haystacks at Chailly at Sunrise</td>
<td>Oil on canvas</td>
<td>1865</td>
<td>San Diego Museum of Art</td>
</tr>
<tr>
<td>#2</td>
<td>#2</td>
<td>Wheatstacks (End of Summer)</td>
<td>Oil on canvas</td>
<td>1890-91</td>
<td>The Art Institute of Chicago</td>
</tr>
<tr>
<td>#3</td>
<td>#1</td>
<td>Guernica</td>
<td>Sprawling, black, white, gray canvas</td>
<td>1937</td>
<td>Reina Sofia Nat. Mus. of Cont. Art</td>
</tr>
<tr>
<td>#4</td>
<td>#3</td>
<td>Soldier of the First Division</td>
<td>Oil and collage on canvas</td>
<td>1914</td>
<td>The Museum of Modern Art, N.Y.</td>
</tr>
</tbody>
</table>

**PAINTING** (Code:INT, Painter:INT, Name:STRING, Material:STRING, Date:STRING, Museum:STRING)
Properties of Relations

- There are no duplicate tuples
  - Body of the relation is a mathematical set (a set of tuples), by definition, do not include duplicate elements
- Tuples are unordered
  - Sets in mathematics are not ordered
- Attributes are unordered
  - Each column is identified by its name
- All attribute values are atomic
  - At every row-and-column position in a relation, there always exist only ONE value, not a LIST of values (i.e., relations do NOT contain repeating groups).

Good Relational Schema Design

- Avoid redundancies and guarantee data consistency (i.e., eliminate update and deletion anomalies)?
  - Find Functional (and Multi-Valued) Dependencies
  - Perform Decomposition of Relations
  - Verify Properties of Normalized Relations

<table>
<thead>
<tr>
<th>Properties</th>
<th>3NF</th>
<th>BCNF</th>
<th>4NF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eliminates Redundancy due to Functional Dependencies</td>
<td>Most</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Eliminates Redundancy due to Multi-valued Dependencies</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Preserves Functional Dependencies</td>
<td>Yes</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
<tr>
<td>Preserves Multi-valued Dependencies</td>
<td>Maybe</td>
<td>Maybe</td>
<td>Maybe</td>
</tr>
</tbody>
</table>
Data Query and Manipulation

- **Data Definition Language (DDL):**
  - Creation and modification of a relation schema, integrity constraints and access structures

- **Data Manipulation Language (DML):**
  - Insertion/deletion/modification of relation tuples

- **Data Query Language (QL):**
  - Creation of new relations (intentional) from existing ones

Relational Query Languages

1. **CALCULUS:** found on first order predicate calculus. Two versions of the calculus:
   - Tuple variables
   - Domain variables

2. **ALGEBRA:** found on a fixed set of operators allowing manipulation of relations. Two classes of operators:
   - Primitive (projection, selection, union, difference, Cartesian product)
   - Derived (intersection, division, join)

- **THEOREME:** The Relational Calculus and Algebra have the same expressive power
The Relational Algebra

The SQL Standard

- Data manipulation is centered around a general filter for the construction of new relations from existing ones in the base
  - SELECT attribute(s) of the resulting relation
  - FROM relations (and variables) defining the scope of the research
  - [WHERE boolean condition(s) on relation attribute(s)]
  - [[NOT] [EXISTS | IN] subqueries]
  - [GROUP BY attribute(s)]
  - [HAVING boolean condition(s)]
  - [ORDER BY attribute(s) [ASC | DESC]] ]

- The SQL filter can handle only tuples with atomic values
  - the result of a query is always a set of tuples (i.e., a relation)
- **NOTE:** SQL (without functions) has the same expressive power with relational algebra and calculus
Query Example

- **Find the paintings of French Artists**

  **SQL**
  
  ```sql
  SELECT p.name
  FROM ARTIST a, PAINTING p
  WHERE a.nationality = "France"
  and p.painter = a.code
  ```

  **Calculus**
  
  `{n | PAINTING(c, p, n, t, d, m), ARTIST (a, p, s, l, na),
  na="France"}
  ```

  **Algebra**
  
  $$\Pi_{\text{name}} (\sigma_{\text{nationality}=\text{"France"}}
  \text{ARTIST } \times_{\text{code=painter}} \text{PAINTING})$$

Advantages of RDBMS

- **Simple Representation**
  - Computerized files are viewed as tables where the rows represent the records and the columns the fields

- **Declarative Access**
  - User specifies by queries what data are wanted and the DBMS figure out where and how to access the data

- **Formal Foundations**
  - Set theory and First Order logic predicate calculus

- **Standardization**
  - The SQL standard (SQL1 86, SQL1 89, SQL2 92, SQL3 98)

- **Mature Technology**
  - A variety of products: DB2, ORACLE, INGRESS, SYBASE, MS-ACCESS, etc.
Drawbacks of RDBMS

- **Poor Data Representation** (DDL tables with atomic values) => Novel applications (Web, GIS, multi-media & spatio-temporal) require:
  - Complex Objects
    - Design Objects, Documents, Physical Systems, ...
  - Multiplicity of Data Types
    - Numbers, Text, Images, Audio, Video, ...
  - Extensible Data Models
- **Limited Data Manipulation** (DML is not Turing-complete) => Difficult integration of Query Languages with Host PL (e.g., C+SQL) for:
  - Better interfaces
  - More visualization & interactivity & animation
  - Different paradigms for search (querying & browsing)

➲ Low productivity and application performances

Poor Data Representation: Example

<table>
<thead>
<tr>
<th>Artist</th>
<th>Style</th>
<th>Life-Time</th>
<th>Influenced by</th>
</tr>
</thead>
<tbody>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>1840-1926</td>
<td>Eugène Boudin</td>
</tr>
<tr>
<td>Claude Monet</td>
<td>Impressionism</td>
<td>1840-1926</td>
<td>Edouard Manet</td>
</tr>
<tr>
<td>Edouard Manet</td>
<td>Impressionism</td>
<td>1832-1883</td>
<td>Gustave Courbet</td>
</tr>
<tr>
<td>Gustave Courbet</td>
<td>Realism</td>
<td>1819-1877</td>
<td>Michelangelo Merisi da Caravaggio</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- How we can represent that:
  - Artists have been influenced by a set of Persons: Monet influences are Boudin and Manet (no bulk data)
  - Artists influences may be recursive: Monet has been influenced by Manet whose influence was Courbet, influenced in his turn by Caravaggio (no recursion)
  - Artists as well as their influences are not simple strings (or integers): nothing guarantee that “Caravaggio” is an Artist (no typing)

➲ Modeling information has to be moved from the data part to the schema part
The Impedance Mismatch Problem

- In order to develop a database application we need two languages:
  - C or C++ and SQL
- Programming Languages (PL):
  - provide mechanisms (e.g., a pointer, array, set)
  - limit controls (depends also on the typing system)
  - is data-item-at-time oriented (imperative)
  - designed for the main memory
- Database Languages (DL):
  - provide policies (e.g., functions, access methods, integrity constraints)
  - favor controls
  - is set-at-time oriented (assertional)
  - designed for the secondary memory

➲ Incompatibility of Styles and Translation/Communication Costs
- A programmer must deal with two languages
- Data must converted from the database to the program work space

Programming in C+SQL: Example ORACLE

```
#include <stdio.h>
EXEC SQL BEGIN DECLARE SECTION;
  VARCHAR user[20];
  VARCHAR password[20];
  VARCHAR protocol[12];
  VARCHAR vstyle[32];
  VARCHAR vname[32];
EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;
main() {
  /* connection to Oracle */
  strcpy(user.arr,"christop");
  user.len=strlen(user.arr);
  strcpy(password.arr,"xxxxxx");
  password.len=strlen(password);
  strcpy(protocol.arr,"P:host");
  protocol.len=strlen(protocol.arr);}
```
EXEC SQL DECLARE fineart_base DATABASE;
EXEC SQL CONNECT :user IDENTIFIED BY :password
   AT : fineart_base USING :protocol;
/*execute an Oracle SQL query*/
strcpy(vstyle.arr="Impressionism");
vstyle.len=strlen(vstyle.arr);
EXEC SQL DECLARE cursor_artist CURSOR FOR
   SELECT name
   FROM FINE_ART
   WHERE Style=:vstyle;
EXEC SQL OPEN cursor_artist;
printf("Impressionist Artists\n");
for(;;)
   {EXEC SQL WHENEVER NOT FOUND DO break;
    EXEC SQL FETCH cursor_artist INTO :vname;
    printf("%s\n", vname.arr);
   }
EXEC SQL CLOSE cursor_artist;
EXEC SQL COMMIT WORK RELEASE;

**ORACLE embedded SQL Preprocessor**
The Road From Relational to Object DBMS

- Richer Data Structuring:
  - complex values/objects and object-oriented models

- Advanced Data Management:
  - extended-relational languages with ADTs
  - persistent programming languages
  - object-oriented database languages
Evolution of Data Management Systems

Increasing Database Functionality & Easiness of Use

- **Application**
  - PL
  - SQL
  - RDBMS

- **Application**
  - PL
  - Extended SQL
  - RDBMS

- **Application**
  - OPL
  - ODBMS

- **Application**
  - OPL
  - ODBMS

---

OBDMS: Integration of Technologies

- **Database Technology**:
  - persistence
  - secondary storage management
  - data sharing (concurrency)
  - data reliability (recovery)
  - declarative query language
  - data security
  - distribution, etc.

- **Advanced Data Modeling**:
  - complex values/objects
  - classification
  - generalization/specialization (inheritance)

- **Object-Oriented Languages**:
  - objects
  - encapsulation
  - methods
  - overloading
  - extensibility
# A Brief History of DB Models & Access Paradigms

<table>
<thead>
<tr>
<th>DATE</th>
<th>MODEL</th>
<th>ACCESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>50's</td>
<td>FILE ORIENTED SYSTEMS (COBOL)</td>
<td>imperative</td>
</tr>
<tr>
<td>mid-60's</td>
<td>HIERARCHICAL (IMS IBM)</td>
<td>navigational</td>
</tr>
<tr>
<td>mid-60's</td>
<td>NETWORK (IDMS)</td>
<td>navigational</td>
</tr>
<tr>
<td>70</td>
<td>RELATIONAL (Codd)</td>
<td>assertional</td>
</tr>
<tr>
<td>76</td>
<td>ENTITY-RELATIONSHIP (Chen)</td>
<td>conceptual</td>
</tr>
<tr>
<td>77</td>
<td>FUNCTIONAL (Backus)</td>
<td>functional</td>
</tr>
<tr>
<td>early-80's</td>
<td>DEDUCTIVE (DATALOG)</td>
<td>logical</td>
</tr>
<tr>
<td>mid-80's</td>
<td>EXTENDED RELATIONAL</td>
<td>assertional + navigational</td>
</tr>
<tr>
<td>early-90's</td>
<td>OBJECT ORIENTED (ODMG)</td>
<td>navigational + assertional</td>
</tr>
<tr>
<td>mid-90's</td>
<td>OBJECT-RELATIONAL (SQL3)</td>
<td>navigational + assertional</td>
</tr>
<tr>
<td>late-90's</td>
<td>SEMISTRUCTURED (XML)</td>
<td>navigational</td>
</tr>
</tbody>
</table>