Relational Model Definitions

- A relation is a table with columns and rows.
- An attribute is a named column of a relation.
- A tuple is a row of a relation.
- A domain is a set of allowable values for one or more attributes.
- The degree of a relation is the number of attributes it contains.
- The cardinality of a relation is the number of tuples it contains.
- A relational database is a collection of normalized relations with distinct relation names.

Recall: Relational Integrity

- Integrity rules are used to insure the data is accurate.
- Constraints are rules or restrictions that apply to the database and limit the data values it may store.
  - DBMS checks the constraints
- Types of constraints:
  - Domain constraint - Every value for an attribute must be an element of the attribute's domain or be null.
    - null represents a value that is currently unknown or not applicable.
    - null is not the same as zero or an empty string.
  - Entity integrity constraint - In a base relation, no attribute of a primary key can be null.
  - Key constraint – every relation must have a key; one of them chosen as primary key
  - Referential integrity constraint - If a foreign key exists in a relation, then the foreign key value must match a primary key value of a tuple in the referenced relation or be null.

Referential integrity and Foreign Keys

- Only students listed in the Students relation should be allowed to enroll for courses.
Relational Query Languages

- **Query languages**: Allow specification of schemas and constraints
- Allow manipulation and retrieval of data from a database.
- Relational model supports simple, powerful QLs:
  - Strong formal foundation based on logic.
  - Allows for much optimization.
- Query Languages ≠ programming languages!
  - QLs not expected to be "Turing complete".
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

Formal Query Languages

- Formal query languages are defined as mathematical operators over the set
- What is the advantage of a formal language?
- Relational algebra, Relational calculus are examples
- Procedural vs Non-procedural languages
  - Procedural: what data to fetch from DB and how/where to get the data
  - Non-procedural: what data to fetch from DB
    - System/DBMS needs to figure out the "how"
  - Can have a mix in practice
    - **Relational algebra**: procedural language
    - **Relational calculus**: non-procedural (declarative)

Next: SQL DDL and Constraint specifications

- Specifying schema/table
- Specifying constraints in SQL

SQL: Structured Query Language

The standard language for relational data

- Invented by folks at IBM, esp. Don Chamberlin
- Actually not a great language...
- Beat a more elegant competing standard, QUEL, from Berkeley

Separated into a DML & DDL

SQL DML component based on relational algebra & calculus

- Data definition (DDL) – to define schema/tables
  - Define Schema
  - Define Constraints
SQL Basic Rules...read up on SQL syntax

- Some basic rules for SQL statements:
  - 1) There is a set of reserved words that cannot be used as names for database objects. (e.g. SELECT, FROM, WHERE)
  - 2) SQL is case-insensitive. Only exception is string constants. 'FRED' not the same as 'fred'.
  - 3) SQL is free-format and white-space is ignored.
  - 4) The semi-colon is often used as a statement terminator, although that is not always required.
  - 5) Date and time constants have defined format:
    - Dates: 'YYYY-MM-DD' e.g. '1975-05-17'
    - Times: 'hh:mm:ss[.f]' e.g. '15:00:00'
    - Timestamp: 'YYYY-MM-DD hh:mm:ss[.f]' e.g. '1975-05-17 15:00:00'
  - 6) Two single quotes '' are used to represent a single quote character in a character constant. e.g. 'Master”s'.

The SQL Query Language

- To find all 18 year old students, we can write:

  ```
  SELECT * FROM Students S
  WHERE S.age=18
  ```

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>login</th>
<th>age</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Jones</td>
<td>jones@cs</td>
<td>18</td>
<td>3.4</td>
</tr>
<tr>
<td>53688</td>
<td>Smith</td>
<td>smith@ee</td>
<td>18</td>
<td>3.2</td>
</tr>
</tbody>
</table>

- To find just names and logins, replace the first line:

  ```
  SELECT S.name, S.login
  ```

SQL Query Language: DML

To query and retrieve data from the tables we have a:

- SELECT clause
  - What attributes you want
  - What relations/tables to search
  - What condition/predicate to apply

First.....SQL DDL Overview

- SQL data definition language (DDL) allows users to:
  - add, modify, and drop tables
  - define and enforce integrity constraints
  - enforce security restrictions
  - Create views
SQL Identifiers and Data types...standard definitions you've seen before in other languages

- **Identifiers** are used to identify objects in the database such as tables, views, and columns.
  - The identifier is the name of the database object.
  - Rules for SQL identifiers...read notes
  - Note: Quoted or delimited identifiers enclosed in double quotes allow support for spaces and other characters. E.g. "select"

- Data types: each attribute has associated domain of values – i.e., each column has data type
  - The DBMS can perform implicit data type conversion when necessary
  - Can also do explicit conversion using CAST and CONVERT

SQL also supports **user defined data types**
- CREATE DOMAIN
- Similar to typedef in C?

---

Example Schema

- Relational database schema:

```
emp (eno,ename,bdate,title,
salary,supereno,dno)
proj (pno,pname,budget,dno)
dep (dno,dname,mgreno)
workson (eno,pno,resp,hours)
```

---

SQL Data Types...similar to prog lang

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOLEAN</td>
<td>TRUE or FALSE</td>
</tr>
<tr>
<td>CHAR</td>
<td>Fixed length string (padded with blanks) e.g. CHAR(10)</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>Variable length string e.g. VARCHAR(50)</td>
</tr>
<tr>
<td>BIT</td>
<td>Bit string e.g. BIT(4) can store '1011'</td>
</tr>
<tr>
<td>NUMERIC or DECIMAL</td>
<td>Exact numeric data type e.g. NUMERIC(7,2) has a precision (max. digits) of 7 and scale of 2 (if of decimals) e.g. 12345.67</td>
</tr>
<tr>
<td>INTEGER</td>
<td>Integer data only</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>Smaller space than INTEGER</td>
</tr>
<tr>
<td>FLOAT or REAL</td>
<td>Approximate numeric data types.</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>Precision dependent on implementation.</td>
</tr>
<tr>
<td>DATE</td>
<td>Stores YEAR, MONTH, DAY</td>
</tr>
<tr>
<td>TIME</td>
<td>Stores HOUR, MINUTE, SECOND</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Stores date and time data</td>
</tr>
<tr>
<td>INTERVAL</td>
<td>Time interval.</td>
</tr>
<tr>
<td>CHARACTER LARGE OBJECT</td>
<td>Stores a character array (e.g. for a document)</td>
</tr>
<tr>
<td>BINARY LARGE OBJECT</td>
<td>Stores a binary array (e.g. for a picture, movie)</td>
</tr>
</tbody>
</table>

---

COMPANY Database Schema

```
<table>
<thead>
<tr>
<th>EMPLOYEE</th>
<th>Frame</th>
<th>Mint</th>
<th>Name</th>
<th>Sex</th>
<th>Bdate</th>
<th>Address</th>
<th>Sex</th>
<th>Salary</th>
<th>Super_salary</th>
<th>Dno</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPARTMENT</td>
<td>Dname</td>
<td>Dnumber</td>
<td>Mgr.name</td>
<td>Mgr.start_date</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPT_LOCATIONS</td>
<td>Dnumber</td>
<td>Dlocation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT</td>
<td>Frame</td>
<td>Dnumber</td>
<td>Location</td>
<td>Dnum</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WORKS_ON</td>
<td>Estart</td>
<td>Edate</td>
<td>Hours</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEPENDENT</td>
<td>Estart</td>
<td>Dependent_name</td>
<td>Sex</td>
<td>Bdate</td>
<td>Relationship</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 5.5  
Schema diagram for the COMPANY relational database schema.
SQL CREATE TABLE

- The `CREATE TABLE` command is used to create a table in the database. A table consists of a table name, a set of fields with their names and data types, and specified constraints.

- The general form is:

```sql
CREATE TABLE tableName (  
    attr1Name attr1Type [attr1_constraints],  
    attr2Name attr2Type [attr2_constraints],  
    ...  
    attrMName attrMType [attrM_constraints],  
    [primary and foreign key constraints]  
);  
```

SQL Constraints - Entity Integrity

- **Entity Integrity constraint** - The primary key of a table must contain a unique, non-null value for each row. The primary key is specified using the `PRIMARY KEY` clause.
  - e.g. `PRIMARY KEY (eno)` (for Emp relation)
  - e.g. `PRIMARY KEY (eno,pno)` (for WorksOn relation)
  - It is also possible to use `PRIMARY KEY` right after defining the attribute in the `CREATE TABLE` statement.

- There can only be one primary key per relation, other candidate keys can be specified using `UNIQUE`:
  - e.g. `UNIQUE (ename)`
Another Example... ‘mini-banner’

- Create Students table
  - Info on students
- Enrolled table holds information about courses that students take.
  - Is sid same field in the two tables??

CREATE TABLE Students
(sid: CHAR(20),
 name: CHAR(20),
 login: CHAR(10),
 age: INTEGER,
 gpa: REAL,
 PRIMARY KEY (sid)) ;

CREATE TABLE Enrolled
(sid: CHAR(20),
 cid: CHAR(20),
 grade: CHAR(2))

Specifying constraints on Enrolled table

- A reasonable condition/constraint:
  “For a given student and course, there is a single grade”

CREATE TABLE Enrolled
(sid: CHAR(20),
 cid: CHAR(20),
 grade: CHAR(2))

Does this schema have any problems ?
A. True
B. False

CREATE TABLE Enrolled2
(sid CHAR(20)
 cid CHAR(20),
 grade CHAR(2),
 PRIMARY KEY (sid),
 UNIQUE (cid, grade) )

SQL Constraints - Referential Integrity

- Referential integrity constraint - Defines a foreign key that references the primary key of another table.
  - If a foreign key contains a value that is not NULL, that value must be present in some tuple in the relation containing the referenced primary key.

- Example: Workson contains two foreign keys:
  - workson.eno references emp.eno
  - workson.pno references proj.pno

- Specify foreign keys using FOREIGN KEY syntax:

  FOREIGN KEY (eno) REFERENCES emp(eno)
### SQL Referential Integrity Example

- The `CREATE TABLE` command for the `workson` relation:

```sql
CREATE TABLE workson (  
  eno CHAR(5),  
  pno CHAR(5),  
  resp VARCHAR(20),  
  hours SMALLINT,  
  PRIMARY KEY (eno,pno),  
  FOREIGN KEY (eno) REFERENCES emp(eno),  
  FOREIGN KEY (pno) REFERENCES proj(pno)  
);```

### SQL Referential Integrity and Updates

- When you try to **INSERT** or **UPDATE** a row in a relation containing a foreign key (e.g. `workson`) that operation is rejected if it violates referential integrity.

- When you **UPDATE** or **DELETE** a row in the primary key relation (e.g. `emp` or `proj`), you have the option on what happens to the values in the foreign key relation (`workson`):
  1. **CASCADE** - Delete (update) values in foreign key relation when primary key relation has rows deleted (updated).
  2. **SET NULL** - Set foreign key fields to NULL when corresponding primary key relation row is deleted.
  3. **SET DEFAULT** - Set foreign key values to their default value (if defined).
  4. **NO ACTION** - Reject the request on the parent table.

### SQL Referential Integrity Example (2)

```sql
CREATE TABLE workson (  
  eno CHAR(5),  
  pno CHAR(5),  
  resp VARCHAR(20),  
  hours SMALLINT,  
  PRIMARY KEY (eno,pno),  
  FOREIGN KEY (eno) REFERENCES emp(eno),  
  FOREIGN KEY (pno) REFERENCES proj(pno)  
);```

### SQL CREATE TABLE Example

- The `CREATE TABLE` command for the `Emp` relation:

```sql
CREATE TABLE emp (  
  eno CHAR(5),  
  ename VARCHAR(30) NOT NULL,  
  bdate DATE,  
  title CHAR(2),  
  salary DECIMAL(9,2),  
  supereno CHAR(5),  
  dno CHAR(5),  
  PRIMARY KEY (eno),  
  FOREIGN KEY (dno) REFERENCES dept(dno)  
);```
Domain Constraints SQL

- Name should not be NULL
- Age > 10 (restrict values in that domain)
- Other constraints...
  - Can specify SQL query

```
CREATE TABLE Students
    (sid CHAR(20),
     name: CHAR(20) NOT NULL,
     login CHAR(10),
     age INTEGER,
     gpa: REAL,
     CHECK (age > 10));
```

SQL CREATE TABLE Full Syntax

- Full syntax of CREATE TABLE statement:

```
CREATE TABLE tableName (
    { attrName attrType {NOT NULL} {UNIQUE} {PRIMARY KEY}
    {DEFAULT value} {CHECK (condition)} }
    [PRIMARY KEY (colList)]
    [FOREIGN KEY (colList) REFERENCES tbl [(colList)],
    {ON UPDATE action}
    {ON DELETE action}] } }
    {[CHECK (condition)] }
);
```

Important: MySQL currently does not support CHECK operation
Can be implemented using TRIGGERS –
will return to this later in course

ALTER TABLE

- The ALTER TABLE command can be used to change an existing table. This is useful when the table already contains data and you want to add or remove a column or constraint.
  - DB vendors may support only parts of ALTER TABLE or may allow additional changes including changing the data type of a column.
- General form:

```
ALTER TABLE tableName
    [ADD [COLUMN] colName dataType {NOT NULL} {UNIQUE}
    {DEFAULT value} {CHECK (condition)} ]
    [DROP [COLUMN] colName {RESTRICT | CASCADE}]
    [ADD [CONSTRAINT [constraintName]] constraintDef]
    [DROP CONSTRAINT constraintName {RESTRICT | CASCADE}]
    [ALTER [COLUMN] SET DEFAULT defValue]
    [ALTER [COLUMN] DROP DEFAULT]
```

ALTER TABLE Examples

Add column location to dept relation:
```
ALTER TABLE dept
    ADD location VARCHAR(50);
```

Add field SSN to Emp relation:
```
ALTER TABLE emp
    ADD SSN CHAR(10);
```

Indicate that SSN is UNIQUE in emp:
```
ALTER TABLE emp
    ADD CONSTRAINT ssnConst UNIQUE(SSN);
```
DROP TABLE

- The command `DROP TABLE` is used to delete the table definition and all data from the database:

  ```sql
  DROP TABLE tableName [RESTRICT | CASCADE];
  ```

- Example:

  ```sql
  DROP TABLE emp;
  ```

Question: What would be the effect of the command:

  ```sql
  DROP TABLE emp CASCADE;
  ```

Database Updates

- Database updates such as inserting rows, deleting rows, and updating rows are performed using their own statements.

  - INSERT
  - UPDATE
  - DELETE

INSERT Multiple Rows

- INSERT statement extended by many databases to take multiple rows:

  ```sql
  INSERT INTO tableName [(column list)]
  VALUES (data value list) [, (values) ]
  ```

- Example:

  ```sql
  INSERT INTO emp (eno, ename) VALUES ('E10', 'Fred'), ('E11', 'Jane'), ('E12', 'Joe')
  ```
**INSERT rows from SELECT**

- Insert multiple rows that are the result of a SELECT statement:

  ```sql
  INSERT INTO tableName [(column list)]
  SELECT ...
  ```

- Example: Add rows to a temporary table that contains only employees with `title = 'EE'`.

  ```sql
  INSERT INTO tmpTable
  SELECT eno, ename
  FROM emp
  WHERE title = 'EE'
  ```

**UPDATE Statement**

- Updating existing rows is performed using UPDATE statement:

  ```sql
  UPDATE tableName
  SET col1 = val1 [,col2=val2...] [WHERE condition]
  ```

- Examples:
  - 1) Increase all employee salaries by 10%.
    ```sql
    UPDATE emp SET salary = salary*1.10;
    ```
  - 2) Decrease salaries of employees in department 'D1' by 8% and put their title after their name.
    ```sql
    UPDATE emp
    SET salary = salary*0.92, ename=concat(ename, ' ', title)
    WHERE dno = 'D1';
    ```

**DELETE Statement**

- Rows are deleted using the DELETE statement:

  ```sql
  DELETE FROM tableName [WHERE condition]
  ```

- Examples:
  - 1) Fire everyone in the company.
    ```sql
    DELETE FROM workson;
    DELETE FROM emp;
    ```
  - 2) Fire everyone making over $35,000.
    ```sql
    DELETE FROM emp
    WHERE salary > 35000;
    ```

**DDL Summary**

- SQL contains a data definition language that allows you to CREATE, ALTER, and DROP database objects such as tables, triggers, indexes, schemas, and views.

- Constraints are used to preserve the integrity of the database:
  - **CHECK** can be used to validate attribute values.
  - **Entity Integrity constraint** - The primary key of a table must contain a unique, non-null value for each row.
  - **Referential Integrity constraint** - Defines a foreign key that references a unique key of another table.

- **INSERT, DELETE, and UPDATE** commands modify the data stored within the database.