The Relational Model

- database consists of several tables (relations)
- columns in each table are named by attributes
- each attribute has an associated domain (set of allowed values)
- data in each table consists of a set of rows (tuples) providing values for the attributes

Example

<table>
<thead>
<tr>
<th>Relation name</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT</td>
<td>Name</td>
</tr>
<tr>
<td>Benjamin Boyer</td>
<td>305-61-2435</td>
</tr>
<tr>
<td>Katherine Ashly</td>
<td>381-62-1245</td>
</tr>
<tr>
<td>Dick Davidson</td>
<td>422-11-2320</td>
</tr>
<tr>
<td>Charles Cooper</td>
<td>489-22-1100</td>
</tr>
<tr>
<td>Barbara Benson</td>
<td>533-69-1238</td>
</tr>
</tbody>
</table>
Relation Schema
“type declaration”

- Relation name
- Set of attributes
- Domain of each attribute
- Integrity constraints

Example
CUSTOMER (Cust-id, Cust-name, Address, Phone#)

integer char strings 7-digits

Attribute Types

- Each attribute of a relation has a name
- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- Sometimes, the special value **null** is considered a member of every domain
Relation Instance

An *instance* of a relation schema is the current content of the relation: a *set of rows* (tuples) over the attributes, with values from the attribute domains.

<table>
<thead>
<tr>
<th>customer_name</th>
<th>customer_street</th>
<th>customer_city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>

More on tuples

**Notation:**
- We refer to *component values* of a tuple *t* by $t(A_i) = v_i$ (the value of attribute $A_i$ for tuple $t$), also called coordinates.
Example

<table>
<thead>
<tr>
<th>customer_name</th>
<th>customer_street</th>
<th>customer_city</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jones</td>
<td>Main</td>
<td>Harrison</td>
</tr>
<tr>
<td>Smith</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Curry</td>
<td>North</td>
<td>Rye</td>
</tr>
<tr>
<td>Lindsay</td>
<td>Park</td>
<td>Pittsfield</td>
</tr>
</tbody>
</table>

\[ t = \langle \text{Smith}, \text{North}, \text{Rye} \rangle \]
\[ t(\text{customer_name}) = \text{Smith} \]
\[ t(\text{customer_street}) = \text{North} \]
\[ t(\text{customer_city}) = \text{Rye} \]

attributes and tuple values are generally assumed to be ordered

Relations are Unordered Sets

The tuples are not considered to be ordered, even though they appear to be so in the displayed tabular form.

<table>
<thead>
<tr>
<th>account_number</th>
<th>branch_name</th>
<th>balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-101</td>
<td>Downtown</td>
<td>500</td>
</tr>
<tr>
<td>A-215</td>
<td>Mianus</td>
<td>700</td>
</tr>
<tr>
<td>A-102</td>
<td>Perryridge</td>
<td>400</td>
</tr>
<tr>
<td>A-305</td>
<td>Round Hill</td>
<td>350</td>
</tr>
<tr>
<td>A-201</td>
<td>Brighton</td>
<td>900</td>
</tr>
<tr>
<td>A-222</td>
<td>Redwood</td>
<td>700</td>
</tr>
<tr>
<td>A-217</td>
<td>Brighton</td>
<td>750</td>
</tr>
</tbody>
</table>
Database

- A database consists of one or several relations
- Information about an application is usually broken up into parts, with each relation storing one part of the information
  
  - `account`: stores information about accounts
  - `depositor`: stores information about which customer owns which account
  - `customer`: stores information about customers
- Storing all information as a single relation such as `bank (account_number, balance, customer_name, ..)` is possible but not desirable:
  results in repetition of information and the need for null values

Relational Integrity Constraints

- Constraints are *conditions* that must hold on *all* valid relation instances of a database
- Some common types of constraints:
  1. **Key** constraints
  2. **Entity integrity** constraints
  3. **Referential integrity** constraints
Key Constraints

- **Superkey** of R: A set of attributes SK of R such that no two tuples in any valid relation instance r(R) will have the same value for SK. That is, for all distinct tuples t1 and t2 in r(R), t1(SK) ≠ t2(SK).

- **Key** of R: A "minimal" superkey; that is, a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey.

  Example: The CAR relation schema:
  CAR(State, Reg#, SerialNo, Make, Model, Year)
  has two keys Key1 = {State, Reg#}, Key2 = {SerialNo}.
  {SerialNo, Make} is a superkey but not a key.

- If a relation has several candidate keys, one is chosen to be the primary key.

### Key Constraints

<table>
<thead>
<tr>
<th>CAR</th>
<th>LicenseNumber</th>
<th>EngineSerialNumber</th>
<th>Make</th>
<th>Model</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Texas ABC-739</td>
<td>A63932</td>
<td></td>
<td>Ford</td>
<td>Mustang</td>
<td>96</td>
</tr>
<tr>
<td>Florida TVP-347</td>
<td>B43696</td>
<td></td>
<td>Oldsmobile</td>
<td>Cutlass</td>
<td>99</td>
</tr>
<tr>
<td>New York MPO 22</td>
<td>X83564</td>
<td></td>
<td>Oldsmobile</td>
<td>Delta</td>
<td>95</td>
</tr>
<tr>
<td>California 432-TFY</td>
<td>C43742</td>
<td></td>
<td>Mercedes</td>
<td>150-D</td>
<td>93</td>
</tr>
<tr>
<td>California RSK-629</td>
<td>Y123905</td>
<td></td>
<td>Toyota</td>
<td>Camry</td>
<td>98</td>
</tr>
<tr>
<td>Texas RSK-629</td>
<td>U038935</td>
<td></td>
<td>Jaguar</td>
<td>XJS</td>
<td>98</td>
</tr>
</tbody>
</table>

The primary key attributes are underlined.
## Employee Data

<table>
<thead>
<tr>
<th>FNAME</th>
<th>LNAME</th>
<th>SSN</th>
<th>DATE</th>
<th>ADDRESS</th>
<th>SEX</th>
<th>SALARY</th>
<th>SUPERSSN</th>
<th>DINO</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>Smith</td>
<td>2323456789</td>
<td>12/04/1999</td>
<td>721 Franklin, Houston, TX</td>
<td>M</td>
<td>30000</td>
<td>2345678901</td>
<td>123</td>
</tr>
<tr>
<td>Traci</td>
<td>King</td>
<td>2456789012</td>
<td>03/05/2000</td>
<td>321 Main, Houston, TX</td>
<td>F</td>
<td>20000</td>
<td>3456789012</td>
<td>456</td>
</tr>
<tr>
<td>Alice</td>
<td>Johnson</td>
<td>3567890123</td>
<td>01/06/2001</td>
<td>123 Oak, Springfield, TX</td>
<td>F</td>
<td>15000</td>
<td>4567890123</td>
<td>567</td>
</tr>
<tr>
<td>Jennifer</td>
<td>Wilson</td>
<td>4678901234</td>
<td>04/07/2002</td>
<td>234 Bay, Dallas, TX</td>
<td>M</td>
<td>25000</td>
<td>5678901234</td>
<td>678</td>
</tr>
<tr>
<td>Ramon</td>
<td>Nguyen</td>
<td>5789012345</td>
<td>07/08/2003</td>
<td>345 Pine, Austin, TX</td>
<td>M</td>
<td>30000</td>
<td>6789012345</td>
<td>789</td>
</tr>
<tr>
<td>Joyce</td>
<td>Singh</td>
<td>6890123456</td>
<td>10/09/2004</td>
<td>456 Elm, Houston, TX</td>
<td>F</td>
<td>20000</td>
<td>7890123456</td>
<td>890</td>
</tr>
<tr>
<td>Arnold</td>
<td>Johnson</td>
<td>7901234567</td>
<td>13/10/2005</td>
<td>567 Oak, Austin, TX</td>
<td>M</td>
<td>25000</td>
<td>8901234567</td>
<td>901</td>
</tr>
<tr>
<td>James</td>
<td>Dunn</td>
<td>8901234567</td>
<td>15/11/2006</td>
<td>678 Pine, Houston, TX</td>
<td>M</td>
<td>20000</td>
<td>9012345678</td>
<td>012</td>
</tr>
</tbody>
</table>

## Works On

<table>
<thead>
<tr>
<th>ESSN</th>
<th>DEPT NUMBER</th>
<th>DESCRIPTION</th>
<th>HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456</td>
<td>1</td>
<td>123456</td>
<td>10.5</td>
</tr>
<tr>
<td>234567</td>
<td>2</td>
<td>234567</td>
<td>7.0</td>
</tr>
<tr>
<td>345678</td>
<td>3</td>
<td>345678</td>
<td>10.0</td>
</tr>
<tr>
<td>456789</td>
<td>4</td>
<td>456789</td>
<td>10.0</td>
</tr>
<tr>
<td>567890</td>
<td>5</td>
<td>567890</td>
<td>10.0</td>
</tr>
<tr>
<td>678901</td>
<td>6</td>
<td>678901</td>
<td>10.0</td>
</tr>
<tr>
<td>789012</td>
<td>7</td>
<td>789012</td>
<td>10.0</td>
</tr>
<tr>
<td>890123</td>
<td>8</td>
<td>890123</td>
<td>10.0</td>
</tr>
</tbody>
</table>

## Dependent Data

<table>
<thead>
<tr>
<th>ESSN</th>
<th>DEPENDENT NAME</th>
<th>SEX</th>
<th>DATE</th>
<th>RELATIONSHIP</th>
<th>HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>234567</td>
<td>Alex</td>
<td>F</td>
<td>10/01/2001</td>
<td>DAUGHTER</td>
<td>10.0</td>
</tr>
<tr>
<td>345678</td>
<td>Theodore</td>
<td>M</td>
<td>05/02/2002</td>
<td>SON</td>
<td>15.0</td>
</tr>
<tr>
<td>456789</td>
<td>Joy</td>
<td>F</td>
<td>06/03/2003</td>
<td>SPOUSE</td>
<td>10.0</td>
</tr>
<tr>
<td>567890</td>
<td>Alice</td>
<td>M</td>
<td>07/04/2004</td>
<td>SPOUSE</td>
<td>10.0</td>
</tr>
<tr>
<td>678901</td>
<td>Michael</td>
<td>M</td>
<td>08/05/2005</td>
<td>SON</td>
<td>10.0</td>
</tr>
<tr>
<td>789012</td>
<td>Anne</td>
<td>F</td>
<td>09/06/2006</td>
<td>DAUGHTER</td>
<td>10.0</td>
</tr>
<tr>
<td>890123</td>
<td>David</td>
<td>M</td>
<td>10/07/2007</td>
<td>SPOUSE</td>
<td>10.0</td>
</tr>
<tr>
<td>901234</td>
<td>Susan</td>
<td>F</td>
<td>11/08/2008</td>
<td>null</td>
<td>15.0</td>
</tr>
</tbody>
</table>

1/9/2020
Entity Integrity

- The **primary key attributes** PK of each relation schema R in S cannot have null values in any tuple. This is because PK values are used to *identify* the individual tuples.

  \[ t(A) \neq \text{null} \text{ for every tuple } t \text{ in} \]
  \[ \text{a valid instance of } R, \text{ where } A \text{ is in } PK \]

  Note: Other attributes of R may be similarly constrained to disallow null values, even though they are not members of the primary key.

Referential Integrity
Referential Integrity

- A constraint involving two relations of the database (the previous constraints involve a single relation).
- Used to specify a relationship among tuples in two relations: the referencing relation and the referenced relation.
- Tuples in the referencing relation $R_1$ have attributes FK (called foreign key attributes) that reference the primary key attributes PK of the referenced relation $R_2$. A tuple $t_1$ in $R_1$ is said to reference a tuple $t_2$ in $R_2$ if $t_1(FK) = t_2(PK)$.
- A referential integrity constraint can be displayed in a relational database schema as a directed arc from $R_1$.FK to $R_2$.PK.

Referential Integrity Constraint

Statement of the constraint
The value in the foreign key column(s) FK of the referencing relation $R_1$ can be either
(1) a value of a primary key PK in the referenced relation $R_2$ or (2) null.
Other Types of Constraints

- Semantic Integrity Constraints: based on application semantics and cannot be expressed by the model per se
  - e.g., “the max. no. of hours per employee for all projects he or she works on is 40 hrs per week”

- A constraint specification language may have to be used to express these

  SQL provides assertions and triggers

Update Operations on Relations

- INSERT a tuple.
- DELETE a tuple.
- MODIFY a tuple.

- Integrity constraints should not be violated by the update operations.
- Several update operations may have to be grouped together.
In case of integrity violation, several actions can be taken:
- Cancel the operation that causes the violation (REJECT option)
- Perform the operation but inform the user of the violation
- Trigger additional updates so the violation is corrected
- Execute a user-specified error-correction routine