A relationship, I think, is like a shark, you know? It has to constantly move forward or it dies. And I think what we got on our hands is a dead shark.

Woody Allen (from Annie Hall, 1979)

Databases Model the Real World

- “Data Model” allows us to translate real world things into structures computers can store
- Many models: Relational, E-R, O-O, Network, Hierarchical, etc.
- Relational
  - Rows & Columns
  - Keys & Foreign Keys to link Relations

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>53666</td>
<td>Carnatic101</td>
<td>C</td>
</tr>
<tr>
<td>53666</td>
<td>Reggae203</td>
<td>B</td>
</tr>
<tr>
<td>53666</td>
<td>Topology112</td>
<td>A</td>
</tr>
<tr>
<td>53666</td>
<td>History105</td>
<td>B</td>
</tr>
</tbody>
</table>

Enrolled Students

<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@eecs</td>
<td>18</td>
<td>3.2</td>
</tr>
<tr>
<td>53650</td>
<td>Smith</td>
<td>smith@math</td>
<td>19</td>
<td>3.8</td>
</tr>
</tbody>
</table>

Steps in Database Design

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level descr (often done w/ER model)
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency, normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what, and how

Conceptual Design

- What are the entities and relationships in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the integrity constraints or business rules that hold?
- A database `schema' in the ER Model can be represented pictorially (ER diagrams).
- Can then map an ER diagram into a relational schema.

ER Model Basics

- **Entity**: Real-world object, distinguishable from other objects. An entity is described using a set of attributes.
- **Entity Set**: A collection of similar entities. E.g., all employees.
  - All entities in an entity set have the same set of attributes. (Until we consider hierarchies, anyway!)
  - Each entity set has a key (underlined).
  - Each attribute has a domain.

ER Model Basics (Contd.)

- **Relationship**: Association among two or more entities. E.g., Attishoo works in Pharmacy department.
  - relationships can have their own attributes.
- **Relationship Set**: Collection of similar relationships.
  - An n-ary relationship set R relates n entity sets $E_1, ..., E_n$; each relationship in R involves entities $e_1 \in E_1$, ..., $e_n \in E_n$
ER Model Basics (Cont.)

• Same entity set can participate in different relationship sets, or in different “roles” in the same set.

Participation Constraints

• Does every employee work in a department?
• If so, this is a participation constraint
  - the participation of Employees in Works_In is said to be total (vs. partial)
  - What if every department has an employee working in it?
• Basically means “at least one”

Means: “exactly one”

Binary vs. Ternary Relationships

If each policy is owned by just 1 employee:

Key constraint on Policies would mean policy can only cover 1 dependent!

• Think through all the constraints in the 2nd diagram!

Better design

Key Constraints

An employee can work in many departments; a dept can have many employees.

In contrast, each dept has at most one manager, according to the key constraint on Manages.

Weak Entities

A weak entity can be identified uniquely only by considering the primary key of another (owner) entity.

- Owner entity set and weak entity set must participate in a one-to-many relationship set (one owner, many weak entities).
- Weak entity set must have total participation in this identifying relationship set.

Means: “exactly one”

Binary vs. Ternary Relationships (Contd.)

• Previous example illustrated a case when two binary relationships were better than one ternary.

• An example in the other direction: a ternary relation Contracts relates entity sets Parts, Departments and Suppliers, and has descriptive attribute quantity.
  - No combination of binary relationships is an adequate substitute.
Binary vs. Ternary Relationships (Contd.)

- S "can-supply" P, D "needs" P, and D "deals-with" S does not imply that D has agreed to buy P from S.
- How do we record qty?

Aggregation

Used to model a relationship involving a relationship set.
Allows us to treat a relationship set as an entity set for purposes of participation in (other) relationships.

ISA (‘is a’) Hierarchies

- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.
  - Overlap constraints: Can Simon be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
  - Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)
  - Reasons for using ISA:
    - To add descriptive attributes specific to a subclass.
    - i.e. not appropriate for all entities in the superclass
    - To identify entities that participate in a particular relationship
    - i.e. not all superclass entities participate

Conceptual Design Using the ER Model

- ER modeling can get tricky!
  - Design choices:
    - Should a concept be modeled as an entity or an attribute?
    - Should a concept be modeled as an entity or a relationship?
    - Identifying relationships: Binary or ternary? Aggregation?
  - Note constraints of the ER Model:
    - A lot of data semantics can (and should) be captured.
    - But some constraints cannot be captured in ER diagrams.
    - We’ll refine things in our logical (relational) design

Review - Our Basic ER Model

- Entities and Entity Set (boxes)
- Relationships and Relationship sets (diamonds)
  - binary
  - n-ary
- Key constraints (1-1,1-M, M-M, arrows on 1 side)
- Participation constraints (bold for Total)
- Weak entities - require strong entity for key
- Aggregation - an alternative to n-ary relationships
- ISA hierarchies - abstraction and inheritance

Entity vs. Attribute

- Should address be an attribute of Employees or an entity (related to Employees)?
  - Depends upon how we want to use address information, and the semantics of the data:
    - If we have several addresses per employee, address must be an entity (since attributes cannot be set-valued).
    - If the structure (city, street, etc.) is important, address must be modeled as an entity (since attribute values are atomic).
Entity vs. Attribute (Cont.)

- Works_In2 does not allow an employee to work in a department for two or more periods.
- Similar to the problem of wanting to record several addresses for an employee: we want to record several values of the descriptive attributes for each instance of this relationship.

Entity vs. Relationship

OK as long as a manager gets a separate discretionary budget (dbudget) for each dept.
What if manager’s dbudget covers all managed depts? (can repeat value, but such redundancy is problematic)

These things get pretty hairy!

- Many E-R diagrams cover entire walls!
- A modest example:

A Cadastral E-R Diagram

- Many E-R diagrams cover entire walls!
- A modest example:

Logical DB Design: ER to Relational

- Entity sets to tables.

CREATE TABLE Employees

<table>
<thead>
<tr>
<th>ssn</th>
<th>name</th>
<th>lot</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>48</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
</tr>
</tbody>
</table>

A Cadastral E-R Diagram

cadastral: showing or recording property boundaries, subdivision lines, buildings, and related details

Source: US Dept. Interior Bureau of Land Management, Federal Geographic Data Committee Cadastral Subcommittee
http://www.fairview-industries.com/standardmodule/cad-erd.htm
### Relationship Sets to Tables

- In translating a many-to-many relationship set to a relation, attributes of the relation must include:
  1. Keys for each participating entity set (as foreign keys). This set of attributes forms a **superkey** for the relation.
  2. All descriptive attributes.

```sql
CREATE TABLE Works_In(
    ssn  CHAR(1),
    did  INTEGER,
    since  DATE,
    PRIMARY KEY (ssn, did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

<table>
<thead>
<tr>
<th>ssn</th>
<th>did</th>
<th>since</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>51</td>
<td>1/1/91</td>
</tr>
<tr>
<td>123-22-3666</td>
<td>56</td>
<td>3/3/93</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>51</td>
<td>2/2/92</td>
</tr>
</tbody>
</table>

### Translating ER with Key Constraints

- Since each department has a unique manager, we could instead combine Manages and Departments.

```sql
CREATE TABLE Manages(
    ssn  CHAR(11),
    did  INTEGER,
    since  DATE,
    PRIMARY KEY  (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments)
```

### Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to CHECK constraints).

```sql
CREATE TABLE Dept_Mgr(
    did  INTEGER,
    dname  CHAR(20),
    budget  REAL,
    ssn  CHAR(11) NOT NULL,
    since  DATE,
    PRIMARY KEY  (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE NO ACTION)
```

### Review: Key Constraints

- Each dept has at most one manager, **according to the key constraint** on Manages.

### Review: Participation Constraints

- **Does every department have a manager?**
  - If so, this is a **participation constraint**: the participation of Departments in Manages is said to be **total** (vs. partial).
  - Every `did` value in Departments table must appear in a row of the Manages table (with a non-null `ssn` value!)

### Review: Weak Entities

- A **weak entity** can be identified uniquely only by considering the primary key of another (owner) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this **identifying** relationship set.
Translating Weak Entity Sets
- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all owned weak entities must also be deleted.

CREATE TABLE Dep_Policy (  
  pname CHAR(20),  
  age INTEGER,  
  cost REAL,  
  ssn CHAR(11) NOT NULL,  
  PRIMARY KEY (pname, ssn),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE CASCADE)

Translating ISA Hierarchies to Relations
- General approach:
  - 3 relations: Employees, Hourly_Emps and Contract_Emps.
    - Hourly_Emps: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (hourly_wages, hours_worked, ssn); must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
    - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.
  - Hourly_Emps: ssn, name, lot, hourly_wages, hours_worked.
    - Each employee must be in one of these two subclasses.

Other SQL DDL Facilities
- Integrity Constraints (ICs) - Review
  - An IC describes conditions that every legal instance of a relation must satisfy.
    - Inserts/deletes/updates that violate IC’s are disallowed.
    - Can be used to ensure application semantics (e.g., sid is a key), or prevent inconsistencies (e.g., sname has to be a string, age must be < 200)
- Types of IC’s: Domain constraints, primary key constraints, foreign key constraints, general constraints.
  - Domain constraints: Field values must be of right type. Always enforced.
  - Primary key and foreign key constraints: you know them.

CREATE TABLE Sailors (  
  sid INTEGER,  
  sname CHAR(10),  
  rating INTEGER,  
  age REAL,  
  PRIMARY KEY (sid),  
  CHECK (rating >= 1 AND rating <= 10)
)

CREATE TABLE Reserves (  
  sname CHAR(10),  
  bid INTEGER,  
  day DATE,  
  PRIMARY KEY (bid, day),  
  CONSTRAINT noInterlakeRes  
  CHECK (`Interlake' <> (SELECT B.bname  
  FROM Boats B  
  WHERE B.bid=bid)))

Now you try it
University database:
- Courses, Students, Teachers
- Courses have ids, titles, credits, ...
- Courses have multiple sections that have time/ rm and exactly one teacher
- Must track students’ course schedules and transcripts including grades, semester taken, etc.
- Must track which classes a professor has taught
- Database should work over multiple semesters

Review: ISA Hierarchies
- As in C++, or other PLs, attributes are inherited.
- If we declare A ISA B, every A entity is also considered to be a B entity.
  - Overlap constraints: Can Joe be an Hourly_Emps as well as a Contract_Emps entity? (Allowed/disallowed)
  - Covering constraints: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? (Yes/no)
Constraints Over Multiple Relations

CREATE TABLE Sailors
(D sid INTEGER,
 sname CHAR(10),
 rating INTEGER,
 age REAL,
 PRIMARY KEY (sid),
 CHECK
 (SELECT COUNT (S.sid) FROM Sailors S)
 + (SELECT COUNT (B.bid) FROM
   Boats B) < 100 )

• Awkward and wrong!
• Only checks sailors!
• Only required to hold if the associated table
  is non-empty.
• ASSERTION is the right solution; not
  associated with either table.
• Unfortunately, not supported in many
  DBMSs.
• Triggers are another solution.

CREATE ASSERTION smallClub
CHECK
 ( (SELECT COUNT (S.sid) FROM Sailors S)
 + (SELECT COUNT (B.bid) FROM
   Boats B) < 100 )

Number of boats
plus number of
sailors is < 100

Or, Use a Trigger

• Trigger: procedure that starts automatically if specified
  changes occur to the DBMS
• Three parts:
  - Event (activates the trigger)
  - Condition (tests whether the triggers should run)
  - Action (what happens if the trigger runs)
• Triggers (in some form) are supported by most DBMSs;
  Assertions are not.
• Support for triggers is defined in the SQL:1999
  standard.

Triggers

CREATE TRIGGER trigger_name
ON TABLE
  [FOR {INSERT, UPDATE, DELETE}]
  [WITH APPEND]
AS
sql-statements

• Cannot be called directly – initiated by events on the
  database.
• Can be synchronous or asynchronous with respect to
  the transaction that causes it to be fired.

Summary: Triggers, Assertions, Constraints

• Very vendor-specific (although standard has been
  developed).
• Triggers vs. Constraints and Assertions:
  - Triggers are “operational”, others are declarative.
• Triggers can make the system hard to understand if
  not used with caution.
  - ordering of multiple triggers
  - recursive/chain triggers
• Triggers can be hard to optimize.
• But, triggers are also very powerful.
• Use to create high-performance, “active” databases.

Triggers: Example

CREATE TRIGGER member_delete
ON member FOR DELETE
AS
IF (Select COUNT (*) FROM loan INNER JOIN deleted
ON loan.member_no = deleted.member_no) > 0
BEGIN
  PRINT 'ERROR - member has books on loan.'
  ROLLBACK TRANSACTION
END
ELSE
DELETE reservation WHERE reservation.member_no =
deleted.member_no

Summary of Conceptual Design

• Conceptual design follows requirements analysis,
  – Yields a high-level description of data to be stored
• ER model popular for conceptual design
  – Constructs are expressive, close to the way people think
    about their applications.
  – Note: There are many variations on ER model
    • Both graphically and conceptually
• Basic constructs: entities, relationships, and attributes (of
  entities and relationships).
• Some additional constructs: weak entities, ISA hierarchies,
  and aggregation.
Summary of ER (Cont.)

• Several kinds of integrity constraints:
  - key constraints
  - participation constraints
  - overlap/covering for ISA hierarchies.
• Some foreign key constraints are also implicit in the definition of a relationship set.
• Many other constraints (notably, functional dependencies) cannot be expressed.
• Constraints play an important role in determining the best database design for an enterprise.

Summary of ER (Cont.)

• ER design is subjective. There are often many ways to model a given scenario!
• Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - Entity vs. attribute, entity vs. relationship, binary or n-ary relationship, whether or not to use ISA hierarchies, aggregation.
• Ensuring good database design: resulting relational schema should be analyzed and refined further.
  - Functional Dependency information and normalization techniques are especially useful.