CS 344: Database Management Systems

Lecture Notes

10th August 2011

Topic:

ER to Relational Conversion

Submitted by

Apul Jain
09010112

Apoorv Kumar
09010111

Relational Model:

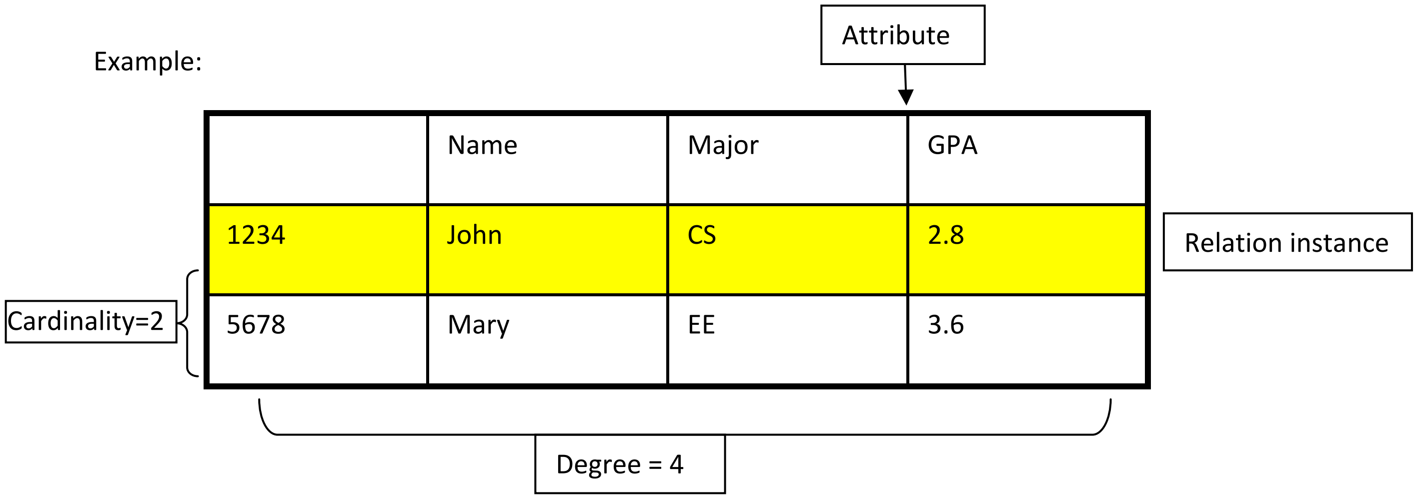
Relational model is a collection of tables representing an E-R database schema. For each entity set and for each relationship set in the database, there is a unique table having the name of the corresponding entity set or relationship set. Each table has multiple columns which correspond to attributes in E-R schema.

A relational model is a tabular representation of ER model. The ER diagram represents the conceptual level of database design intended as a description of real-world entities while a relational schema is at the logical level of database design.

In relational model,

Table	represents	a schema/relation
row	represents	a relational instance (also called tuple)
column	represents	an attribute
cardinality	represents	number of rows
degree	represents	number of columns

Example:



	Name	Major	GPA
1234	John	CS	2.8
5678	Mary	EE	3.6

ER to Relational conversion:

<u>Basic ideas:</u>

Build a table for each entity set.

Build a table for each relationship set.

Make a column in the table for each attribute in the entity set.

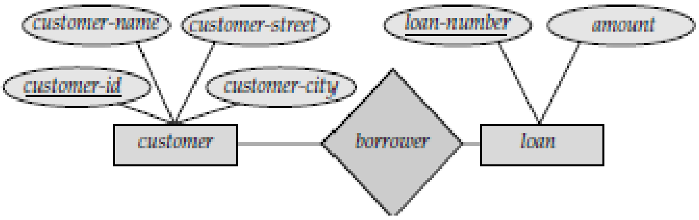
Generate primary key

<u>Tabular representation of Strong entity set:</u>

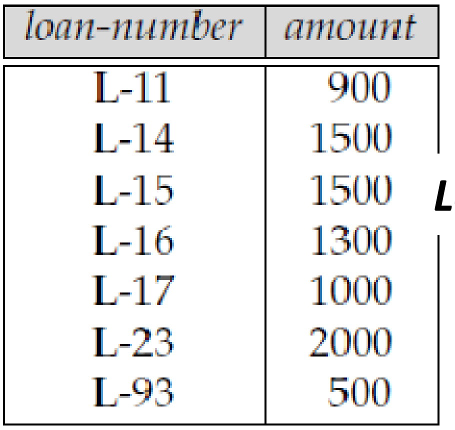
Let E be a strong set with descriptive attributes a_1 , a_2 ,………, a_n. Represent this entity by a table called *E* having n distinct columns, each corresponding to one of the n attributes of *E*.

Each row in this table corresponds to one entity of set *E*.

Example:

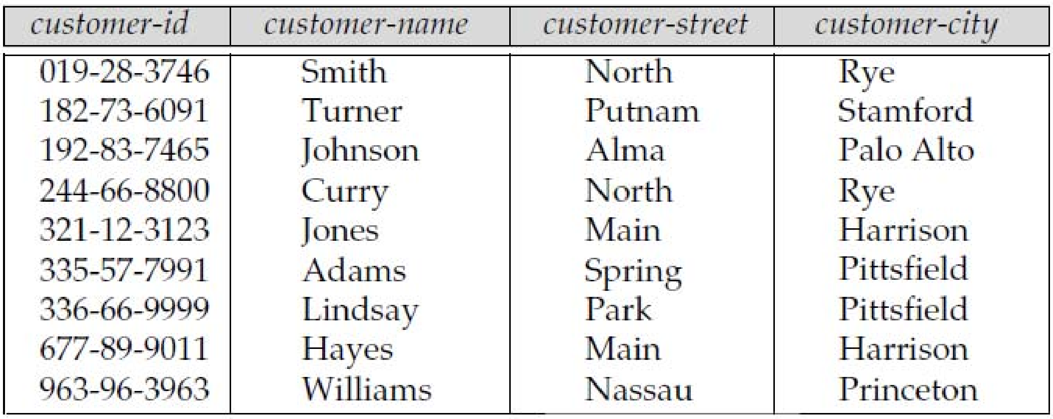


E-R diagram corresponding to customer and loans.



loan-number	*amount*
L-11	900
L-14	1500
L-15	1500
L-16	1300
L-17	1000
L-23	2000
L-93	500

Loan **table**



customer-id	*customer-name*	*customer-street*	*customer-city*
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

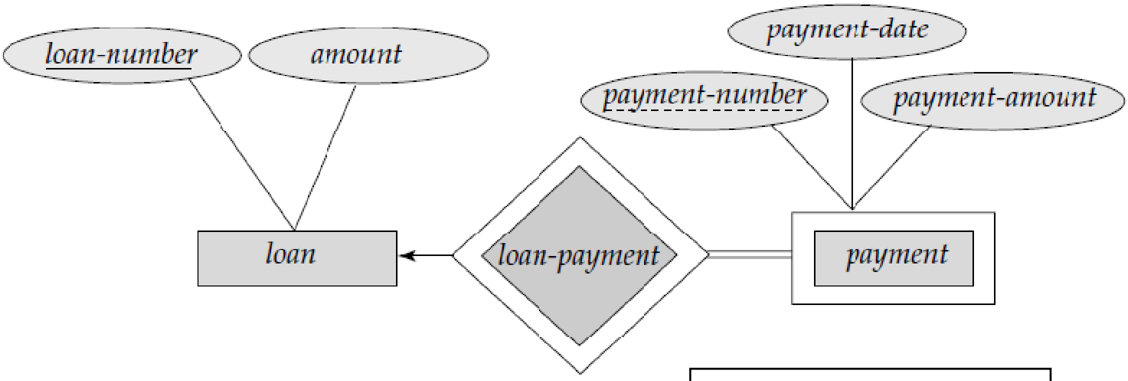
customer **table**

Tabular representation of weak entity set:

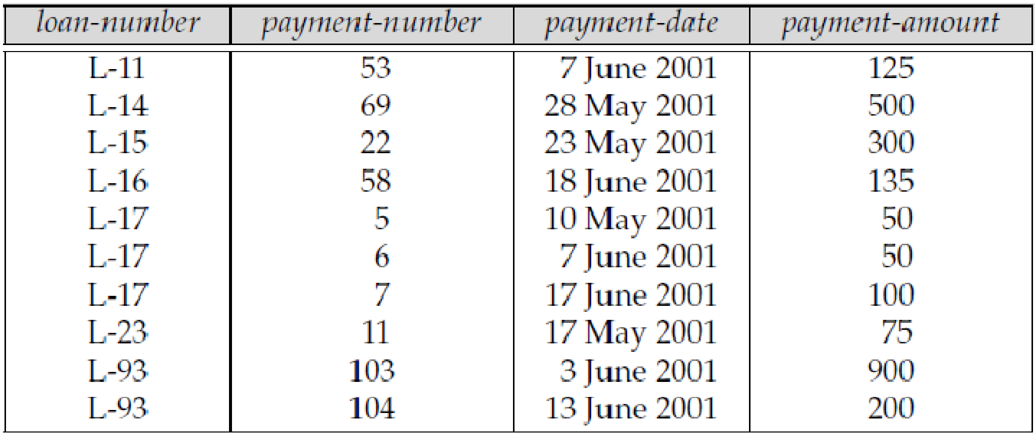
Let *A* be the weak entity set with attributes a_1 ,a_2,………,a_m and let B be the strong entity set on which A depends. Let the primary key of B consists of b_1 ,b_2,………,b_n. Then entity set A is represented by a table having one column for each of the attribute in the set:

$$\{a_1 , a_2,………, a_m \} \text{ U } \{b_1 , b_2,………, b_m\}$$

Example:



ER diagram loan-payment



loan-number	payment-number	payment-date	payment-amount
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

Table corresponding to weak entity set payment

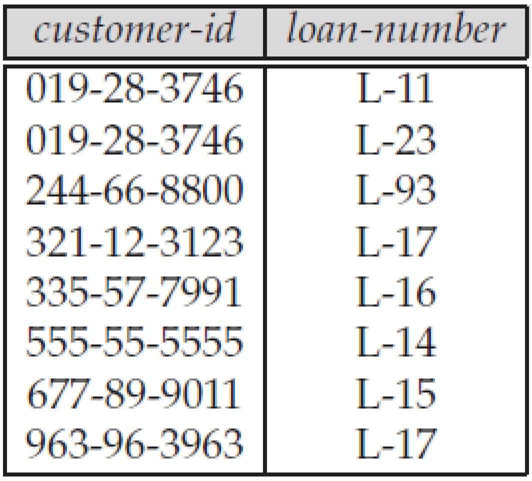
payment entity set has three attributes: *payment-number*, *payment-date*, and *payment-amount*.
The primary key of the *loan* entity set, on which *payment* depends, is *loan-number*. Thus, we represent *payment* by a table with four columns labelled *loan-number*, *payment-number*, *payment-date*, and *payment-amount*.

Tabular representation of Relationship sets

Let R be a relationship set and let a_1 , a_2,………, a_m be the set of attributes formed by the union of primary keys of each of the entity sets participating in *R*, and let the descriptive attributes (if any) of *R* be b_1, b_2, . . . , b_n. We represent this relationship set by a table called *R* with one column for each attribute of the set:

$$\{a_1, a_2, . . . , a_m\} \cup \{b_1, b_2, . . . , b_n\}$$

Example:



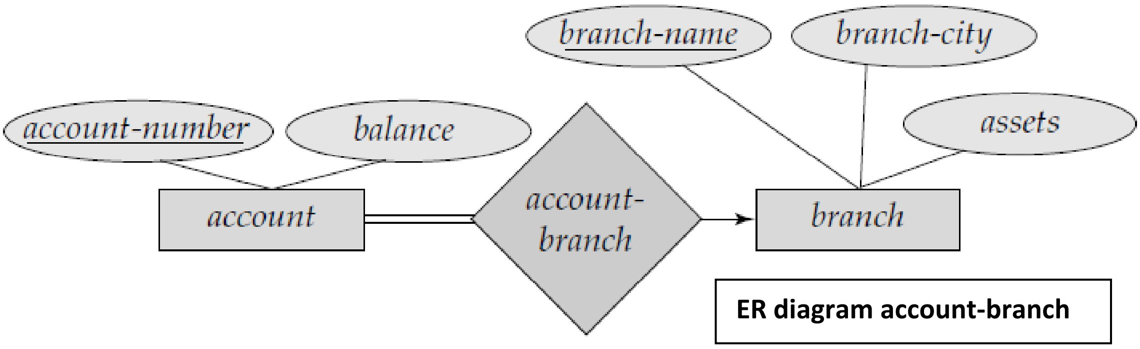
customer-id	*loan-number*
019-28-3746	L-11
019-28-3746	L-23
244-66-8800	L-93
321-12-3123	L-17
335-57-7991	L-16
555-55-5555	L-14
677-89-9011	L-15
963-96-3963	L-17

The *borrower* table

Combination of Tables:

Consider a many-to-one relationship set *account-branch* from entity set *account* to entity set *branch*. Participation of *account* in the relationship is total; that is, every entity *a* in the entity set *account* must participate in the relationship *account-branch*. Then we can combine the tables *account and account-branch to* form a single table consisting of the union of columns of both tables.

Example:



ER diagram account-branch

In above ER participation of *account* in the *account-branch* is total. Hence, an account cannot exist without being associated with a particular branch. Further, the relationship set *account-branch* is many to one from *account* to *branch*.

Therefore, we can combine the table for *account-branch* with the table for *account* and require only the following two tables:

- *account*, with attributes *account-number*, *balance*, and *branch-name*
- *branch*, with attributes *branch-name*, *branch-city*, and *assets*

account-number	balance	branch-name
1210801	4,84,000	SBI, Guwahati

account *table*

branch-name	branch-city	assets
SBI, Guwahati	Guwahati	29092819

Branch *table*

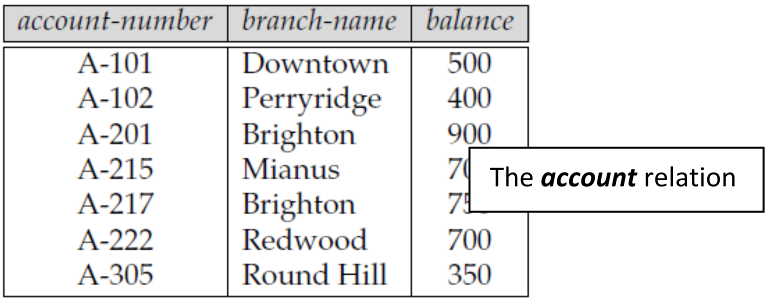
Database schema:

Database schema: logical design of database

Database instance: snapshot of data in database

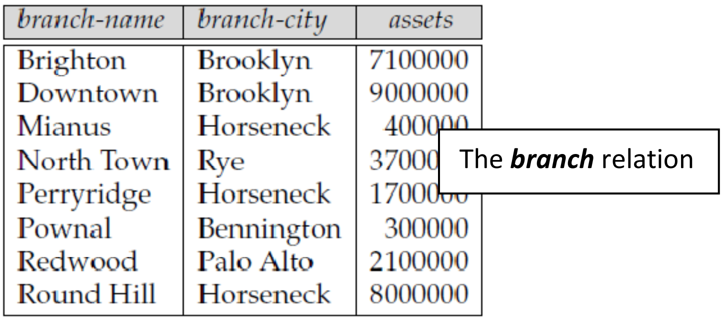
The concept of a relation corresponds to the programming-language notion of a variable. The concept of a **relation schema** corresponds to the programming-language notion of type definition.

Examples of relation schema:



account-number	*branch-name*	*balance*
A-101	Downtown	500
A-102	Perryridge	400
A-201	Brighton	900
A-215	Mianus	70
A-217	Brighton	75
A-222	Redwood	700
A-305	Round Hill	350

The ***account*** relation



branch-name	*branch-city*	*assets*
Brighton	Brooklyn	7100000
Downtown	Brooklyn	9000000
Mianus	Horseneck	400000
North Town	Rye	37000
Perryridge	Horseneck	1700000
Pownal	Bennington	300000
Redwood	Palo Alto	2100000
Round Hill	Horseneck	8000000

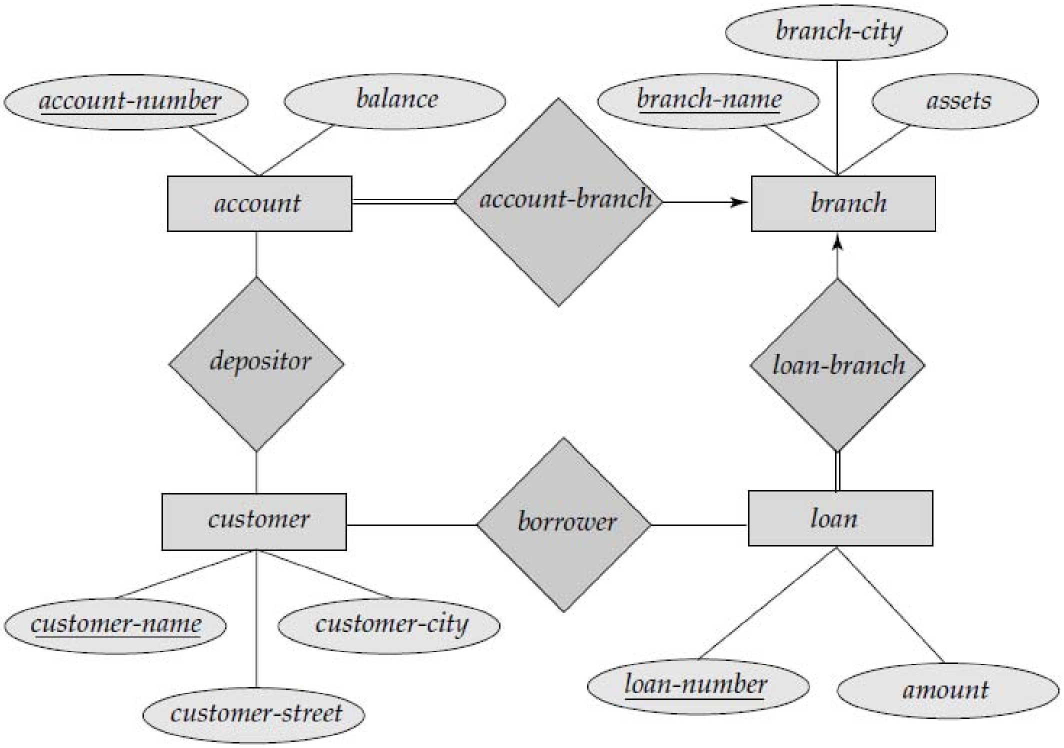
The ***branch*** relation

Account-schema = (*account-number*, *branch-name*, *balance*)

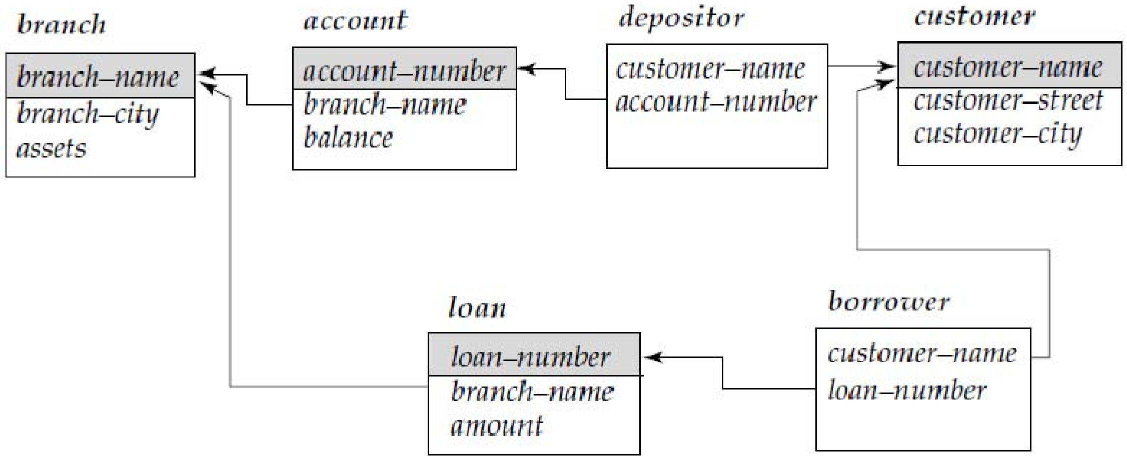
Branch-schema = (*branch-name*, *branch-city*, *assets*)

In general, a relation schema consists of a list of attributes and their corresponding domains.

Schema diagram:



E-R diagram for the banking enterprise.



Schema diagram for the banking enterprise.

Query Languages:

A **query language** is a language in which a user requests information from the database. It is used to define relational schema. SQL is widely used query language.

Defining schema in SQL:

The CREATE TABLE statement is used to define a new table.To create the Students relation, we can use the following statement:

```
CREATE TABLE Students ( sid CHAR(20),
                                 name CHAR(30),
                                 login CHAR(20),
                                 age INTEGER,
                                    gpa REAL )
```

Tuples are inserted using the INSERT command. We can insert a single tuple into the Students table as follows:

```
INSERT
INTO Students (sid, name, login, age, gpa)
VALUES (53688, `Smith', `smith@ee', 18, 3.2)
```

We can delete tuples using the DELETE command. We can delete all Students tuples with *name* equal to Smith using the command:

```
DELETE
FROM Students S
WHERE S.name = `Smith'
```

We can modify the column values in an existing row using the UPDATE command. For example, we can increment the age and decrement the gpa of the student with *sid* 53688:

```
UPDATE Students S
SET S.age = S.age + 1, S.gpa = S.gpa - 1
WHERE S.sid = 53688
```

Key constraints:

In SQL we can declare that a subset of the columns of a table constitute a key by using the UNIQUE constraint. At most one of these `candidate' keys can be declared to be a *primary key*, using the PRIMARY KEY constraint

```
CREATE TABLE Students ( sid CHAR(20),
name CHAR(30),
login CHAR(20),
age INTEGER,
gpa REAL,
UNIQUE (name, age),
CONSTRAINT StudentsKey PRIMARY KEY (sid) )
```

This definition says that *sid* is the primary key and that the combination of *name* and *age* is also a key.

Foreign Key Constraints

Suppose we have relation: Enrolled(*sid:* string, *cid:* string, *grade:* string) having foreign key *sid* . SQL query for this will be:

```
CREATE TABLE Enrolled ( sid CHAR(20),
cid CHAR(20),
grade CHAR(10),
PRIMARY KEY (sid, cid),
FOREIGN KEY (sid) REFERENCES Students )
```

The foreign key constraint states that every *sid* value in Enrolled must also appear in Students, that is, *sid* in Enrolled is a foreign key referencing Students.

================================END===============================